

Implementation of Fault-Resilient Network Slicing using GNS3-Mininet Hybrid Testbed

Hyeon Ho Lee*, Won Seok Choi*, Hyeong Bin Choi*, Seong Gon Choi*

* Information & Communication Engineering, Chungbuk University, Cheongju-si, Chungcheongbuk-do, Korea
hhl9438@chungbuk.ac.kr, choiws@chungbuk.ac.kr, choibin2463@gmail.com, choisg@chungbuk.ac.kr

Abstract— We implement and validate an SDN based fault-resilient network slicing mechanism on a GNS3-Mininet hybrid testbed. URLLC services demand guaranteed QoS and ultra-low latency, yet current implementations lack swift and reliable slice recovery solutions when network failures occur in resource constrained dynamic environments. Therefore, we utilize an SDN controller and RESTCONF to ensure slice service continuity through centralized topology monitoring and dynamic traffic rerouting to backup paths. The mechanism monitors changes in network topology state, identifies missing nodes via topology snapshot comparison, triggers path reconfiguration algorithms, and installs backup flow rules. Our testbed combines GNS3 for realistic network device behavior with Mininet for scalable host emulation, enabling comprehensive end-to-end verification. We demonstrate the feasibility and effectiveness of the fault-resilient slicing mechanism through experimental validation in a laboratory environment.

Keywords— *Software Defined Networking (SDN), Network Slicing, GNS3, Mininet, Fault-Resilient*

I. INTRODUCTION

The emergence of 5G networks has established network slicing as a key technology for creating multiple logical networks on a single physical infrastructure. Network slicing enables operators to provide differentiated services tailored to specific requirements, ranging from enhanced mobile broadband to massive IoT deployments. Among these services, Ultra Reliable Low Latency Communication (URLLC) demands the most stringent performance guarantees, requiring sub-millisecond latency and 99.999% reliability [5]. Software-Defined Networking (SDN) and Network Function Virtualization (NFV) have emerged as enabling technologies for implementing dynamic network slicing, providing centralized control and flexible resource management essential for meeting diverse service requirements.

Existing network slicing research has primarily focused on theoretical algorithms or partial implementations, with relatively little attention given to automated fault recovery mechanisms. Real-time fault detection and automatic recovery are limited in practical implementation, and current approaches often rely on manual intervention during network failures, making it difficult to guarantee the service continuity required for URLLC services. Most existing approaches either handle faults at the network-wide level or focus on congestion-based triggers, lacking granular fault management at the slice level. Furthermore, current implementations either

require manual intervention for recovery or fail to provide automatic restoration mechanisms to the original path after fault resolution. Research into the practical implementation and validation of automated fault detection alongside dynamic path reconfiguration, particularly in testbed environments, remains an area requiring further exploration.

In this paper implements and validates an SDN based fault recovery network slicing mechanism with automated recovery capabilities on a GNS3 [6] and Mininet [7] hybrid testbed. Our implementation detects node failures using centralised topology monitoring via the SDN controller and automatically reconfigures traffic paths using the RESTCONF [8] API. To combine realistic network infrastructure emulation with scalable edge network configuration, we integrated GNS3 for the backbone network and Mininet for the access network. We implemented a proof-of-concept prototype demonstrating automatic slice recovery without manual intervention.

The key contributions of this research are as follows: (1) Implementation and deployment of a GNS3-Mininet hybrid testbed for realistic network slicing validation, (2) Implementation and experimental validation of a RESTCONF based automated fault detection and dynamic path recovery mechanism, (3) Functional validation of VLAN based slice isolation and seamless traffic rerouting during node failures.

This paper is organised as follows. Section II reviews related work. Section III presents the system design and fault resilience mechanisms. Section IV describes the implementation and validation results. Section V concludes the paper.

II. RELATED WORK

Network slicing has been extensively studied from both architectural and implementation perspectives. Barakabitze et al. provide a comprehensive survey of SDN and NFV based network slicing architectures, presenting key challenges such as slice isolation, resource allocation, and inter slice communication [1]. Their analysis provides a foundation for understanding the technical requirements of actual slice implementations.

Recent studies have explored various approaches to fault tolerance within network slicing environments. Han et al. developed a hybrid GNS3 and Mininet-WiFi testbed to implement a fault-tolerant routing algorithm for SDN backbone networks [2]. Their approach successfully

demonstrated automatic path rerouting with an average recovery time of 30 seconds. However, this solution operates at the network-wide level rather than providing slice-specific recovery mechanisms, limiting its applicability to differentiated service requirements. Hefe and Costa-Requena proposed SDN-managed network slicing for mobile backhaul networks, implementing automatic path adjustment for URLLC user groups upon congestion detection [3]. While demonstrating practical feasibility, their approach focuses on congestion-based rerouting rather than node failure scenarios and lacks an automatic restoration mechanism post-recovery. Zheng et al. present a theoretical framework for deterministic fault-tolerant service-function slicing using k-connected protection graphs [4]. Although their approach was validated through mathematical simulations, the lack of implementation in network emulators or testbed environments limits the practical verification of the proposed mechanism.

Most existing approaches handle failures at the network level or focus on congestion-based triggers, lacking granular failure management at the slice level. Furthermore, current implementations either require manual intervention for recovery or fail to provide automatic restoration to the original path after failure resolution.

Therefore, we intend to implement and validate a slice-based fault management mechanism within the GNS3-Mininet hybrid testbed. This implementation enables automated fault detection and dynamic path recovery through RESTCONF based SDN controller interactions, whilst maintaining slice isolation throughout the fault recovery cycle.

III. PROPOSED METHOD

In this section we describe the overall system design and the implemented fault resilience method.

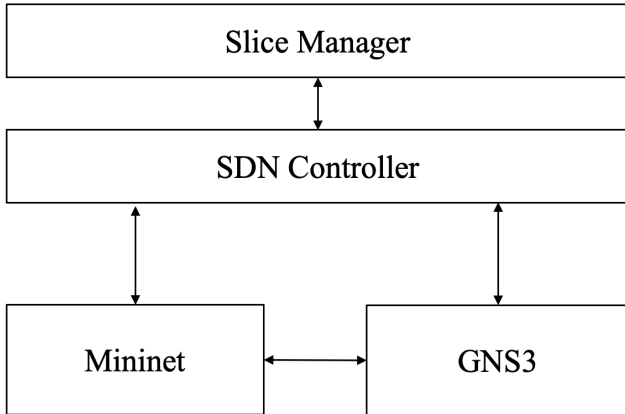


Figure 1. System Architecture

A. System Design

Figure 1 shows the configuration of our implemented SDN based network slicing system. The slice manager manages the network slices, an independently deployed SDN controller provides centralized control, and a hybrid data plane composed of Mininet and GNS3 provides the emulation environment. The SDN controller communicates with both platforms to form an integrated testbed.

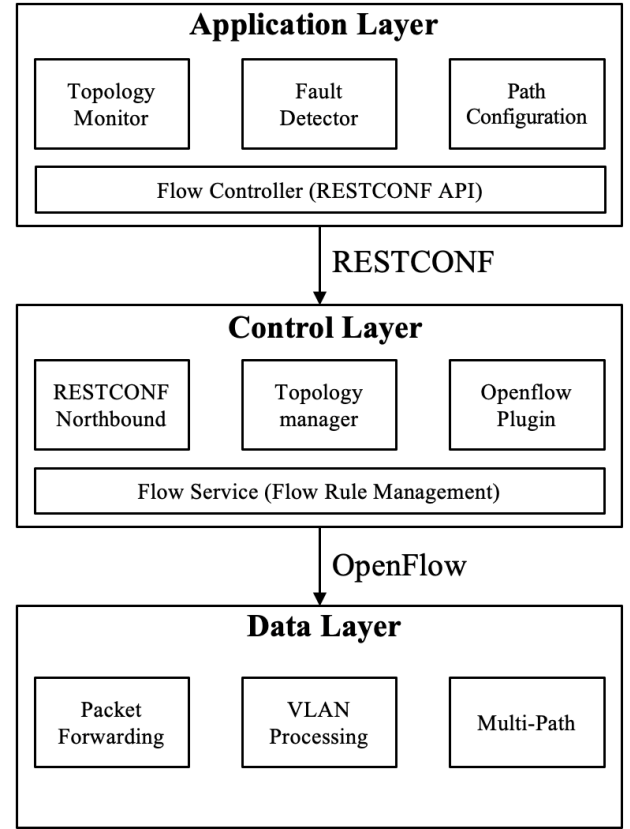


Figure 2. 3-Layer Architecture

Figure 2 describes the detailed three-layer architecture for fault-resilient network slicing. The Application Layer includes modules for topology monitoring, fault detection, and path configuration scenarios that interface with the Flow Controller via RESTCONF. The Control Layer integrates the RESTCONF Northbound interface with OpenFlow based flow rule management, enabling dynamic slice reconfiguration. The Data Layer implements packet forwarding, VLAN processing, and multi-path routing functions via OpenFlow switches.

B. Fault Resilience Method

Algorithm1: Topology Monitoring

Input: Δt

Output: Fault Node N_{fault}

```

1:  $T_{prev} \leftarrow \emptyset$ 
2: while true do
3:    $T_{current} \leftarrow GET\_TOPOLOGY()$ 
4:   if  $T_{prev} \neq \emptyset$  then
5:      $Missing \leftarrow T_{prev} \setminus T_{current}$ 
6:     if  $Missing \neq \emptyset$  then
7:        $N_{fault} \leftarrow FIRST(Missing)$ 
8:        $TRIGGER\_RECONFIGURATION(N_{fault})$ 
9:     end if
10:  end if
11:   $T_{prev} \leftarrow T_{current}$ 
12:   $WAIT(\Delta t)$ 
13: end while

```

To enable proactive fault detection, the topology monitor continuously polls the network topology through RESTCONF API at regular intervals Δt . Algorithm 1 describes the monitoring process that compares consecutive topology snapshots to identify disconnected nodes. By computing the set difference between T_{prev} and $T_{current}$, the algorithm efficiently detects missing nodes without imposing excessive controller overhead. Upon detecting a fault node N_{fault} , the system immediately triggers the path reconfiguration mechanism described in Algorithm 2.

Algorithm 2: Path Reconfiguration

Input: Fault Node N_{fault} , Original Slice S

Output: Backup Slice S'

```

1: function RECONFIGURE( $N_{fault}, S$ )
2:    $P_{original} \leftarrow S.path$ 
3:   if  $N_{fault} \notin P_{original}$  then return  $S$ 
4:    $N_{available} \leftarrow N \setminus N_{fault}$ 
5:    $P_{backup} \leftarrow FIND\_PATH(S.src, S.dst, N_{available})$ 
6:   REMOVE_FLOWS( $P_{original}, S.ID$ )
7:   INSTALL_FLOWS( $P_{backup}, S.ID$ )
8:
9:   return  $S'$  with  $path = P_{backup}$ 
10: end function

```

Upon receiving a fault notification from Algorithm 1, the path calculator module executes dynamic reconfiguration using the pre-calculated backup paths stored in the slice configuration. Algorithm 2 describes the RECONFIGURE function, which verifies whether the faulted node N_{fault} exists on the original slice path, retrieves the pre-calculated backup path via FIND_PATH from the slice design, and performs atomic flow rule migration. The controller minimises failover latency by eliminating real-time path calculation overhead, removing unused forwarding entries via REMOVE_FLOWS and installing backup path flows via INSTALL_FLOWS.

Algorithm 3: Path Restoration

Input: Restored node N_R , Backup Slice S'

Output: Restored Slice S

```

1: function RESTORE( $N_R, S'$ )
2:    $P_{original} \leftarrow GET\_ORIGINAL\_PATH(S')$ 
3:   if  $N_R \notin P_{original}$  then return
4:   WAIT(delay)
5:
6:   if NOT IS_NODE_HEALTHY( $N_R$ ) then return
7:   REMOVE_FLOWS( $P_{backup}, S.ID$ )
8:   INSTALL_FLOWS( $P_{original}, S.ID$ )
9:   return  $S$ 
10: end function

```

To maintain optimal resource utilization, the system automatically restores slices to their original paths when failed

nodes recover. Algorithm 3 describes the RESTORE function, which periodically monitors restored node N_R via topology updates, verifies node status using IS_NODE_HEALTHY after a stabilisation delay, and performs reverse flow migration. The controller removes temporary backup flow flows via REMOVE_FLOWS and reinstalls the original forwarding entries via INSTALL_FLOWS, thereby returning the slice to its designated primary path without service interruption.

IV. IMPLEMENTATION

In this section describes the implementation environment, network topology and configuration details, and finally the experimental results, enabling comprehensive verification of the automatic fault detection and recovery functionality.

A. Testbed

To validate the fault-resilient network slicing mechanism, we implemented PoC testbed on a single-host system equipped with an Intel i7-12700 processor, 32GB RAM, and running Ubuntu 22.04. This testbed integrates GNS3 and Mininet 2.3.0 as complementary emulation platforms, with the OpenDaylight(ODL) [9] Scandium release serving as the SDN controller. All network switches utilized OpenvSwitch(OVS) 2.17 to provide OpenFlow-based switching functionality across both emulation domains.

The configuration comprised a total of 8 OpenFlow switches, with 2 switches deployed in the Mininet domain and 6 switches in the GNS3 domain. Five hosts were deployed for packet generation and verification, including one Mininet host and four VPCS hosts within GNS3. Finally, the Slice Manager was implemented based on Python 3.10.

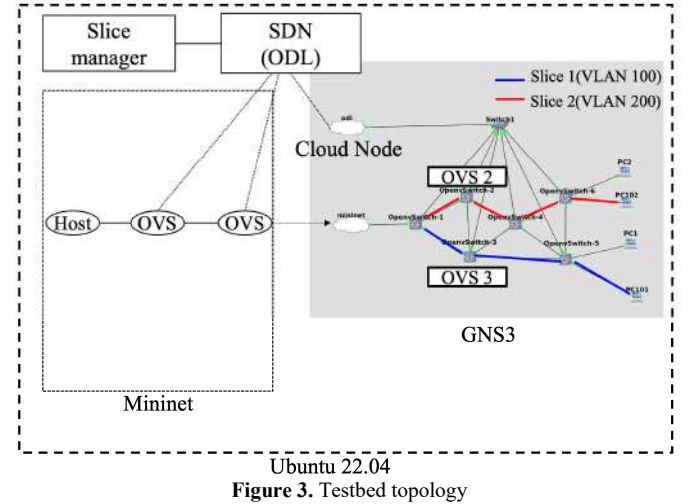


Figure 3. Testbed topology

Figure 3 describes the implemented testbed topology integrating Mininet and GNS3 domains. The Mininet domain consists of two OVS connecting hosts to the GNS3 infrastructure. The GNS3 domain comprises six interconnected OVS and four VPCS endpoint hosts. The ODL controller and Mininet platform integrate with the GNS3 infrastructure via the Cloud Node provided by GNS3. The

testbed implements two separate network slices using VLANs as identifiers. Slice 1, distinguished by VLAN 100, uses the blue path, while Slice 2, distinguished by VLAN 200, uses the red path. The Slice Manager and ODL controller provide centralized management across both domains.

B. Validation Scenario

We validate the fault-resilient network slicing mechanism through three experimental scenarios that demonstrate automatic fault detection, path recovery, and restoration capabilities.

During normal operation, Slice 2, identified as VLAN 200, routes traffic through the primary path using OVS 2, OVS 4, and OVS 6. If OVS 2 fails, the slice manager continuously monitors the network topology via RESTCONF API queries to the ODL controller. Upon detecting a missing node, the system identifies the affected slice and triggers the reconfiguration process.

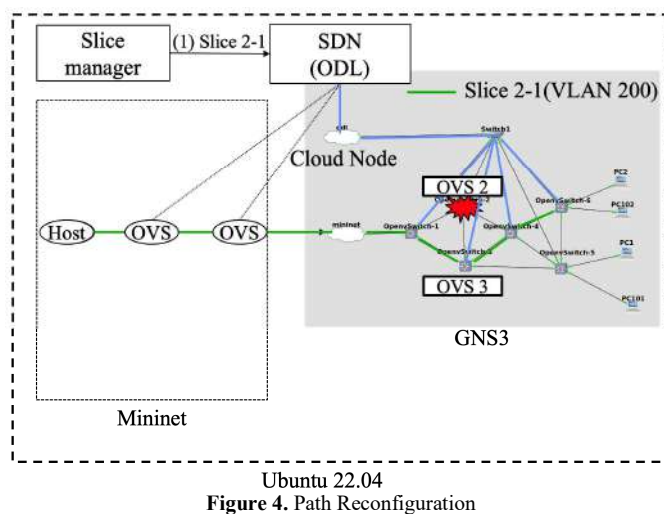


Figure 4. Path Reconfiguration

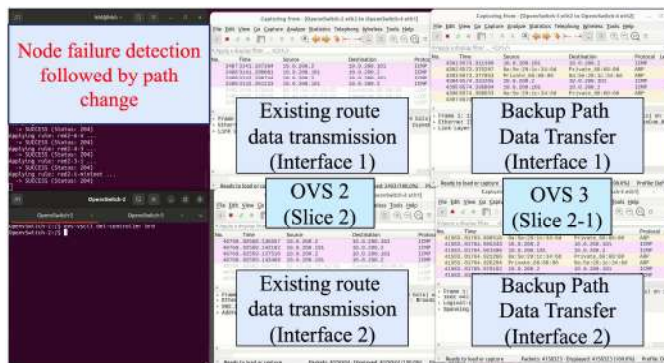


Figure 5. Path Reconfigure Validation

Figure 4 shows the reconfigured network topology after the OVS 2 failure. Slice 2 traffic is rerouted via the backup path as Slice 2-1, passing through OVS 1, OVS 3, OVS 4, and OVS 6. Figure 5 shows an actual packet capture during the failover process, demonstrating how the slice manager detects the failure at the left terminal and traffic switches from the OVS 2 interface to the OVS 3 interface. The Wireshark

capture confirms that VLAN 200 traffic successfully migrated from the primary path experiencing the failure to the backup path while maintaining slice isolation.

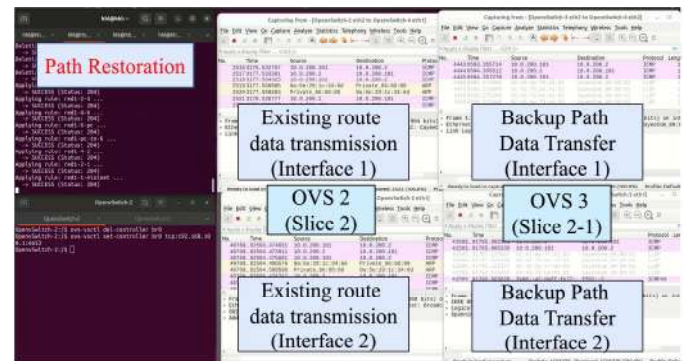


Figure 6. Path Restoration Validation

Figure 6 shows the path restoration process when OVS 2 recovers. The left panel displays the slice manager terminal log detecting the OVS 2 restoration and initiating path recovery to the original route. The Wireshark capture demonstrates reverse traffic migration, where VLAN 200 traffic transitions from the temporary backup path to the restored primary path via OVS 2, OVS 4, and OVS 6. The packet capture validates the automatic recovery mechanism.

V. CONCLUSIONS

We present the implementation and validation of a fault-recovery network slicing mechanism using a GNS3-Mininet hybrid testbed. We successfully integrated two complementary emulation platforms to demonstrate practical slice level fault management capabilities through automatic detection, recovery, and restoration. Experimental verification confirmed node fault detection, automatic path recovery execution using backup paths, and service restoration to the original path upon node recovery. Packet captures validated successful traffic migration while maintaining VLAN segmentation, demonstrating the feasibility of slice level fault management in a hybrid emulation environment. Future research will focus on integrating AI based mechanisms for intelligent path computation and optimization in real traffic scenarios, enabling adaptive resource allocation and predictive fault management in production network environments.

ACKNOWLEDGMENT

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00397979, Development of tera-level ultra-precision transmission network system technology, 50%) and by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. RS-2020-NR049604, 50%).

REFERENCES

- [1] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, 106984, 2020.
- [2] M. P. Han, S. Y. Htet and L. Wuttistikulkij, "Hybrid GNS3 and Mininet-WiFi Emulator for SDN Backbone Network Supporting Wireless IoT Traffic," 2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Phuket, Thailand, 2022, pp. 768-771, doi: 10.1109/ITC-CSCC55581.2022.9895019.
- [3] A. Hefele and J. Costa-Requena, "SDN managed Network Slicing in Mobile Backhaul," 2020 International Conference on Electronics, Information, and Communication (ICEIC), Barcelona, Spain, 2020, pp. 1-8, doi: 10.1109/ICEIC49074.2020.9051225.
- [4] D. Zheng, C. Peng, B. Wang and X. Cao, "Towards Deterministic Fault-Tolerant Service Function Slicing in Edge Networks," 2022 International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 2022, pp. 1-8, doi: 10.1109/ICCCN54977.2022.9868901.
- [5] W. Hamdi, C. Ksouri, H. Bulut and M. Mosbah, "Network Slicing-Based Learning Techniques for IoV in 5G and Beyond Networks," in *IEEE Communications Surveys & Tutorials*, vol. 26, no. 3, pp. 1989-2047, thirdquarter 2024, doi: 10.1109/COMST.2024.3372083.
- [6] GNS3. [Online]. Available from: <https://gns3.com>.
- [7] Mininet. [Online]. Available: <http://mininet.org/>
- [8] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," RFC 8040, Internet Engineering Task Force, Jan. 2017. [Online]. Available: <https://www.rfc-editor.org/info/rfc8040>
- [9] Open Daylight. [Online]. Available: <https://www.opendaylight.org/>



Hyeon Ho Lee received B.S. degree in college of Information & Communication Engineering from Chungbuk National University in 2024. He is currently pursuing the M.S. degree in Radio Communication Engineering. Chungbuk National University. His research interests include network security, IoT, SDN and NFV.



Won Seok Choi received B.S. and Ph.D. degree in the College of Electrical and Computer Engineering, Chungbuk National University, Korea in 2008 and 2014 respectively. He is currently researcher in Research institute of Computer and Information Communication, Chungbuk National University. His research interests include vehicle network, energy saving network, SDN, NFV, NGN and AI.



Hyeong Bin Choi is currently pursuing the B.S. degree in college of information & Communication Engineering from Chungbuk National University, Korea. His research interests include high-speed network architecture, SDN and NFV.



Seong Gon Choi received B.S. degree in Electronics Engineering from Kyungpook National University in 1990, and M.S. and Ph.D. degree from KAIST in Korea in 1999 and 2004, respectively. He is currently a professor in College of Electrical & Computer Engineering, Chungbuk National University. His research interests include V2X, AI, smart grid, IoT, mobile communication, high-speed network architecture and protocol.