An Intelligent SDN-Based Framework for Seamless Mobility and Migration in Hybrid Networks

*University of Information Technology – UIT

1st Le Ngoc Kieu Anh
UIT University
Thu Duc, Viet Nam
22520047@gm.uit.edu.vn

2nd Phung Viet Bac
UIT University)
Thu Duc, Viet Nam
22520089@gm.uit.edu.vn

3rd Tran Phuoc Dai
UIT University
Thu Duc, Viet Nam
22520184@gm.uit.edu.vn

4th Tran Hoai Phu
UIT University
Thu Duc, Viet Nam
22521106@gm.uit.edu.vn

***Abstract*—Software-defined networking (SDN) is experiencing rapid growth, driving demands for mobility and seamless migration to ensure both performance and system flexibility. In traditional networks, the use of static architectures imposes significant constraints on traffic management and orchestration whenever devices or services are relocated; while SDN offers highly centralized control, optimizing packet forwarding and maintaining uninterrupted connectivity during migration remain challenging. To address these issues, we propose a solution that integrates intelligent routing algorithms with dynamic resource allocation on an SDN platform. Experimental results show that our approach significantly reduces migration downtime and packet-loss rates, improves controller response time and throughput, and noticeably lowers end-to-end latency compared with conventional SDN migration methods. This solution can be applied to data center networks, 5G networks, and beyond, helping to improve user experience as well as overall network performance.**

***Index Terms*—SDN, Seamless Mobility, Migration, Controller Handover, Host Mobility.**

I. Introduction

In today's digital era, network systems have become increasingly important, serving as the foundation for many sectors—from communications and finance to industry and services. However, traditional network architectures, which are rigid and hardware-centric, face significant challenges in scaling and flexible management. To address these issues, Software-Defined Networking (SDN) has emerged, offering a new approach that separates the control plane from the data plane and enables more effective programmability and management. SDN has already had a major real-world impact—for example, large data centers operated by Google, Facebook, and Amazon have adopted SDN to optimize traffic routing, reduce latency, and enhance service performance. Despite these benefits, ensuring seamless mobility when devices or services move across the network remains a formidable challenge. Optimizing the migration process without interrupting connectivity is therefore a critical problem to solve. In this paper, we propose a method to support seamless mobility in SDN to improve both network performance and user experience.

A. Current problems of traditional network

Traditional networks, based on rigid hardware-centric architectures, face significant challenges in supporting the dynamic mobility and flexible relocation required in modern systems [1]. These conventional infrastructures tightly couple the control and data planes within individual network devices, making them inherently static and difficult to adapt. As a result, any attempt to reconfigure routing paths, relocate services, or scale out infrastructure typically involves manual intervention, leading to increased operational complexity and potential for misconfiguration.

Moreover, traditional networks lack a centralized view of the system state, which makes real-time decision-making and global optimization nearly impossible. This limitation becomes especially problematic in scenarios involving high mobility—such as cloud-based services, mobile users, or dynamic virtual machine (VM) migration—where seamless handover and uninterrupted connectivity are critical.

1) **Complex architecture and distributed management:** In conventional networks, the control and data planes are tightly integrated within networking devices such as routers and switches [2]. Each device makes independent routing decisions based on local configurations, resulting in operational complexity. This is further compounded by vendor-specific command-line interfaces, making centralized management and policy enforcement cumbersome and error-prone [1], [2].
2) **Limited flexibility and adaptability:** The static nature of traditional networks limits their ability to support dynamic service demands. Achieving consistent policy enforcement across disparate hardware often requires extensive manual configuration, reducing scalability and adaptability [3].
3) **Challenges in mobility and migration support:** Traditional architectures are ill-equipped to maintain seamless connectivity for mobile users. Handover decisions in legacy WLANs are typically based on signal strength alone, which may not guarantee optimal performance [4]. Moreover, IP addresses are tied to physical

locations, making session continuity difficult during host relocation [5], [6]. The migration of applications or services in legacy environments—especially under cloud or virtualized infrastructures—often incurs high latency and packet loss [7].

4) **Scalability and performance bottlenecks:** As traffic volume and the number of devices increase, traditional networks struggle to scale effectively. Bandwidth limitations and latency constraints hinder performance for modern use cases such as video streaming and IoT [6].

B. The necessity of implementing SDN with Seamless Mobility and Migration

In today's hybrid and multi-cloud infrastructures, virtual machines and workloads frequently migrate between physical servers or data centers. Maintaining uninterrupted connectivity during these transitions—without manual reconfiguration—is a major challenge for traditional architectures. Software-Defined Networking (SDN) addresses this issue by decoupling control and forwarding, allowing centralized orchestration and automated configuration. SDN supports seamless migration by preserving IP addresses, security policies, and flow rules during mobility events, thus minimizing human error and service disruption [3].

In mobile and edge environments, devices such as IoT sensors and smartphones continuously change locations. SDN can dynamically reroute flows based on bandwidth, latency, and system load, maintaining Quality of Service (QoS) in latency-sensitive applications like VoIP or video streaming [8]. It also enables load balancing across network nodes, reducing congestion and enhancing overall performance [9].

Additionally, SDN improves security during migration. Access Control Lists (ACLs), firewalls, and other policies can be updated automatically, mitigating risks of DDoS attacks or unauthorized access [10]. Advanced traffic engineering techniques in SDN further optimize bandwidth usage and network reliability [9].

C. Research Objectives

This study aims to provide a comprehensive and practical understanding of implementing seamless mobility and migration in Software-Defined Networking (SDN) environments. The primary goal is to simulate the relocation of network components such as hosts and switches while maintaining uninterrupted connectivity.

To achieve this, the research focuses on visually modeling host and station mobility using hybrid wired and wireless topologies. It also examines switch migration scenarios to analyze how the SDN controller reacts and reconfigures routing dynamically. Through these visual simulations, the study enables deeper exploration of SDN behavior under mobility conditions, offering insight into flow recalculation, path reinstallation, and controller responsiveness.

In addition, the study evaluates existing approaches and proposes feasible strategies for optimizing the migration process in SDN. It aims to address practical challenges such as minimizing packet loss, reducing downtime, and ensuring stable communication during transitions in dynamic, real-world network environments.

II. BACKGROUND AND RELATED WORK

Seamless mobility refers to the ability to maintain uninterrupted network connectivity as devices move across different access points or network domains. This capability is particularly important in modern environments such as high-speed railways, enterprise WLANs, and vehicular networks, where frequent handovers pose serious challenges for consistent Quality of Service (QoS) and reliability.

Recent studies have explored SDN-based frameworks to address mobility and migration challenges in various contexts.

In [11], the authors propose an SDN-based mobility management framework tailored for Beyond 5G (B5G) services in high-speed railway systems. Their work focuses on optimizing frequent handovers due to high mobility, ensuring low-latency and reliable connections for real-time applications.

Similarly, the study in [12] presents an SDN architecture for enterprise WLANs. The framework enables seamless transitions between access points, thereby maintaining stable connectivity for mobile users in office and campus networks.

In the vehicular domain, [13] investigates the application of SDN to support seamless handovers in vehicular ad hoc networks (VANETs). The unpredictable and high-speed movement of vehicles demands robust mechanisms to preserve session continuity, which the SDN control plane can manage through centralized route updates and flow management.

These studies collectively demonstrate the potential of SDN to improve mobility handling across different network types. However, most existing works focus on large-scale, domain-specific deployments and often rely on tightly integrated frameworks that are complex for newcomers to reproduce or adapt for experimentation.

In contrast, this paper emphasizes a fundamental and accessible approach to simulate both mobility and migration using Mininet/Mininet-WiFi and the Ryu controller. The objective is to provide a simplified yet effective model that visualizes host and switch transitions, enabling researchers and students to better understand the behavior of SDN during mobility events. Furthermore, the proposed solution incorporates flow recalculation and ARP-based host discovery mechanisms to ensure continuity and responsiveness during network transitions.

III. PROPOSED TOOLS AND ARCHITECTURE

This section presents the tools and architectural design proposed to implement seamless mobility and migration in Software-Defined Networking (SDN) environments. The solution integrates Mininet for wired network migration simulation, Mininet-WiFi for wireless mobility emulation, and Ryu as the SDN controller. This combination ensures continuous connectivity and dynamic routing optimization when hosts or mobile stations change their positions. The proposed

architecture is designed to address both switch migration and seamless handover between wireless access points, and is further validated through experimental models and performance evaluation.

A. Deployment Tools

Mininet: Mininet is an open-source network emulator that allows the creation of virtual network topologies on a single Linux-based machine. It uses lightweight virtualization to simulate network entities such as hosts, switches, routers, and links as software instances rather than physical hardware. Mininet is widely used for simulating SDN, Ethernet, Wi-Fi, and hybrid networks. Its key advantages include the ability to emulate large-scale networks and perform rapid testing. Additionally, Mininet supports seamless integration with other SDN tools such as OpenFlow and the Ryu controller, making it an efficient platform for network research and development.

Mininet-WiFi: Mininet-WiFi is an extension of Mininet that provides wireless network emulation capabilities within SDN architectures. It introduces support for wireless components such as Access Points (APs) and Stations (STAs), enabling the simulation of complex wireless scenarios, including device mobility, channel variability, and wireless routing protocols. Mininet-WiFi facilitates in-depth experimentation with wireless performance factors and integrates naturally with Mininet to support hybrid wired-wireless topologies.

Ryu: Ryu is an open-source SDN controller framework written in Python. It provides a flexible platform for developing SDN control applications and supports a variety of southbound protocols including OpenFlow, OVSDB, and NetConf. Ryu enables centralized control of network devices, offering programmable and reactive flow management. Its modular architecture and active community make it suitable for both research and production-grade SDN environments.

B. Deployment Scenario

The proposed system architecture is based on a layered Software-Defined Networking (SDN) model designed to support seamless mobility and dynamic resource migration in virtualized environments. The architecture is logically divided into three main functional layers: the data plane, the control plane, and the application plane.

Data Plane: The data plane comprises OpenFlow-enabled network devices such as Open vSwitch (OVS) instances, wireless Access Points (APs), and host machines. These components are responsible for forwarding packets based on flow rules established by the control plane. By employing a flow-based forwarding paradigm, the architecture achieves a complete decoupling between the forwarding logic and control decisions, enabling more flexible and programmable traffic management.

In hybrid topologies that combine wired and wireless networks, Mininet-WiFi is used to emulate mobile entities such as wireless stations (STAs) and access points (APs). This simulation environment enables the creation of dynamic mobility scenarios in a resource-efficient and reproducible manner, allowing researchers to analyze mobility patterns and handover effects in a controlled setup.

Control Plane: The control layer is implemented using the Ryu SDN controller, an open-source Python-based framework. Ryu handles core control functions such as topology discovery, path computation, policy enforcement, and real-time event response. It communicates with both Mininet and Mininet-WiFi via the OpenFlow protocol to receive real-time network state information, including bandwidth utilization, link status, and latency.

Based on this information, Ryu dynamically recalculates optimal paths and updates flow rules to maintain connectivity during topology changes caused by host migration or mobility events. The controller ensures minimal delay and packet loss during link failures or handover transitions.

Application Plane: The application layer includes custom modules developed as Ryu extensions to manage mobility and resource migration. These modules are responsible for handling high-level logic such as triggering ARP requests, detecting host relocation, and coordinating policy updates across the network. This layer provides the programmability needed to implement seamless handover strategies and migration workflows in virtualized SDN environments.

The overall deployment model is illustrated in Figure 1 for the migration scenario and Figure 2 for the mobility scenario. In both cases, the network is structured around a centralized controller, switches, and end-hosts. In the mobility architecture, wireless APs and mobile STAs are also included to represent real-world wireless connectivity contexts.

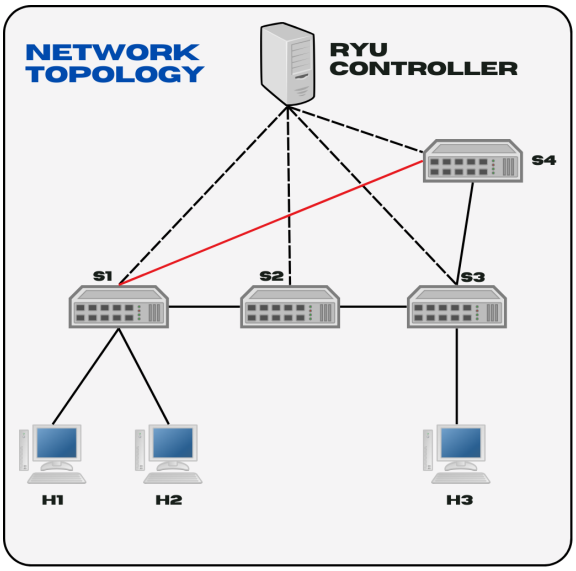

Fig. 1. Migration Network topology for SDN

Migration Scenario:

- In the migration scenario, the network topology consists of a Ryu controller managing four OpenFlow-enabled switches (S1, S2, S3, and S4). Initially, the switches are interconnected in a linear topology: S1 is connected to

S2, S2 to S3, and S3 to S4. Host H1 is connected to S1, while hosts H2 and H3 are connected to S3 and S4, respectively.

- The experiment begins by initiating ICMP ping traffic from H1 to H2 and H3 to verify baseline connectivity across the network. Once connectivity is established, a simulated migration event is triggered: the link between S1 and S2 is removed, and a new link is added between S1 and S4. Simultaneously, host H2 is logically relocated from its original position to be associated with switch S2.
- This topology change emulates the dynamic migration of both switches and hosts within the SDN infrastructure. Following the migration, connectivity tests are repeated to assess whether the controller can successfully recompute paths and maintain communication between H1 and the relocated H2, as well as with H3.The success of this migration process is evaluated based on the controller's ability to quickly detect the topology change, remove outdated flow rules, and install new paths with minimal delay and no packet loss. The updated migration scenario is illustrated in Figure 2.

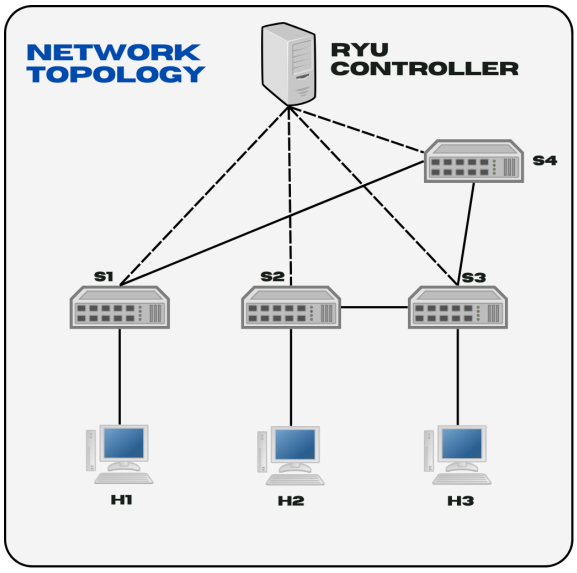

Fig. 2. Topology after migration

- In addition, the architecture is extended to support a Mobility Network scenario using Mininet-WiFi, which enables the emulation of wireless components such as Access Points (APs) and mobile Stations (STAs). In this setup, a single Ryu controller manages the overall flow control and data handling across the network. The topology consists of one OpenFlow switch connected to two wireless APs—AP1 and AP2. The same switch is also connected to a wired host (H1), which is used to simulate traffic interactions with the mobile nodes. Two mobile wireless stations, STA1 and STA2, are included in the topology to represent wireless client devices such as laptops, smartphones, or any Wi-Fi-capable endpoints. These stations can dynamically associate with either AP, allowing the emulation of realistic wireless mobility scenarios.
- In Mininet-WiFi, STAs act as network nodes capable of connecting to nearby APs based on coverage area and movement behavior. Once the simulation is initialized, STA1 and STA2 begin moving from the coverage zone of AP1 toward that of AP2, simulating a handover between wireless cells. During the mobility process, continuous ICMP ping tests are performed between H1 and each STA (STA1, STA2), as well as between the STAs themselves, to assess packet delivery, path stability, and latency throughout the handover. This scenario allows for evaluating the SDN controller's responsiveness in detecting wireless movement, updating flow entries, and maintaining seamless communication during dynamic association changes.

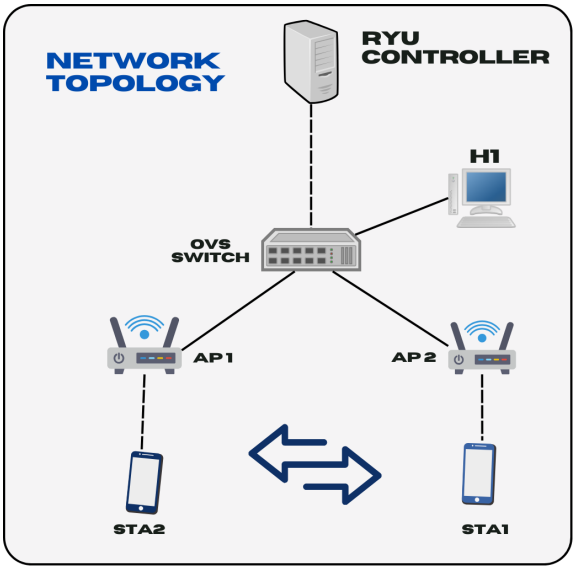

Fig. 3. Mobility Network Topology for SDN

IV. SOLUTIONS AND EVALUATION

This section details the experimental configuration and evaluates the proposed solution's effectiveness in enabling seamless mobility and migration in SDN environments. The objective is to demonstrate how the controller dynamically adapts to topology changes and maintains uninterrupted connectivity during host or link transitions.

A. Smart routing solution for seamless mobility and migration in SDN

In the paper, we propose a solution called *"Flexible Controller"*, enables intelligent reaction to dynamic events in the network. The controller handles topology changes, recalculates paths, and updates flow rules automatically in response to mobility or migration.

The solution consists of four key components:

- Specify the topology of the whole network.

- Calculate the shortest path from the source host to the destination host and install flows on the switches.
- Remove flows when the old shortest path is not available.
- The controller sends ARP Request packets to the host that has moved to a new location.

B. Specify the topology of the whole network

The controller utilizes Ryu's event-driven topology module, particularly events such as **EventSwitchEnter**, **EventLinkAdd**, and **EventLinkDelete**, to construct and update the network model. This allows real-time awareness of switch presence and interconnections, forming the basis for route computation.

C. Calculate the shortest path from the source host to the destination host and install flows on the switches

Schedule for the controller to send **OFPPortStatsRequest** packets periodically. As soon as the controller receives reply packets, it will measure the bandwidth between two hosts based on the tx_bytes information and store it in memory.

When one host wants to ping another host, the controller uses the Dijkstra algorithm with bandwidth as cost to calculate the shortest path [14]. Then, the flow will be installed on the switches directly for both forward and reverse directions.

D. Remove flows when the old shortest path is not available

When a link failure or host migration is detected, the controller identifies all paths affected by the failure. It removes outdated flow entries using OFPFlowMod messages and triggers recomputation of alternate paths to restore connectivity.

The algorithm to find the affected path is presented as follows:

paths ← Information on all active paths
for *path* **in** *paths*:
    for *data* **in** the component of the *path*:
        if the *data* contains the down port:
            add *path* to affected paths
        end if
    end for
return affected paths

E. The controller sends ARP Request packets to the host that has moved to a new location

The controller cannot know exactly which host connects to which switch. Therefore, the first task is to recognize which host has moved, and send ARP Request packets to the host that has moved to a new location. This task can be resolved by this algorithm:

hosts ← Information about all the hosts and the switch links to that host
downPortSwitch ← Information about the switch including the port that has been down
for *host* **in** *hosts*:
    switch ← The switch links to the host
    if *downPortSwitch* is the same as the *switch*:
        return *host*
    end if
end for

After the recognition process is done, the controller creates ARP Request packets with the source IP and source MAC, which contain the information of all hosts except the relocated host, the destination IP is IP of the relocated host, and the destination MAC is 00:00:00:00:00:00. No matter where the host moves to, as long as this host receives the ARP Request packet, it will reply with an ARP Reply packet. All the hosts have communicated with this host will update its ARP table and continue communicating normally.

F. Implementation results

The experimental setup was deployed according to Figures 1, 2, and 3. In the migration scenario, after severing the link between S1 and S2 and reattaching S1 to S4, the relocated host continued to communicate without interruption, demonstrating the controller's ability to re-establish flows.

```
mininet> movesw
>>> Interrupt s1-s2 and connect s1-s4
>>> Completed: s1 connected to s4
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> moveho
>>> Migrating h2 from s1 to s2
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

Fig. 4. Testing ping for migration

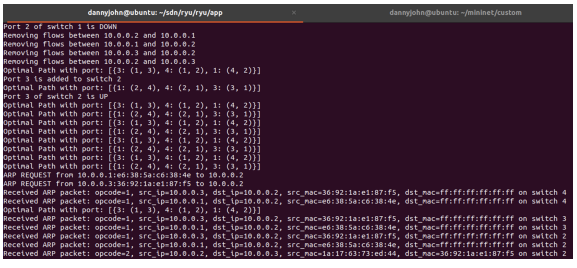

Fig. 5. Active log of controller

Regarding the Mininet environment, the performance and the loss-packet rate are over expectations. However, in reality,

nothing ensures that the network is still stable due to distance, noise, signal strength, bandwidth, etc.

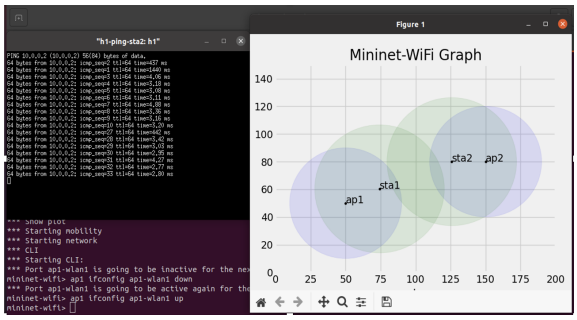

Fig. 6. Testing ping for mobility

Performance was satisfactory under Mininet conditions, with negligible packet loss and acceptable latency. However, real-world deployments may experience additional challenges due to signal noise, wireless interference, or varying link quality, which are not fully represented in the emulation environment.

V. CONCLUSION AND FUTURE WORK

This research has successfully implemented an SDN-based solution using Ryu and Mininet/Mininet-WiFi, enabling the effective management of seamless mobility and migration. The proposed "Flexible Controller" architecture has demonstrated its ability to maintain stable connectivity throughout simulated scenarios, while also providing a practical and intuitive approach to SDN. However, the current solution still has some limitations, such as the lack of full IPv6 support and the potential risk of controller overload in certain scenarios.

In future work, we will focus on addressing these limitations to enhance the system. Another significant and promising direction involves extending the "Flexible Controller" architecture and its developed control principles for application in satellite networks, particularly for Low Earth Orbit (LEO) satellite systems. This will include adapting the network topology management mechanisms, path computation algorithms, and handover processing procedures to suit the dynamic nature and specific requirements of satellite networks. The objective is to develop a comprehensive solution capable of ensuring seamless and efficient connectivity, thereby extending the applicability of the current SDN model to space-based environments.

ACKNOWLEDGMENT

The authors would like to thank Dr.Phan Xuan Thien for valuable guidance and insightful discussions throughout the project. We also acknowledge the technical support provided by UIT's infrastructure team during the experimental setup.

REFERENCES

[1] F. Liu, G. Kibalya, S. V. N. S. Kumar, and P. Zhang, "Challenges of traditional networks and development of programmable networks," in *Software Defined Internet of Everything*, B. B. Gupta, D.-N. Le, and A. E. Hassanien, Eds. Springer, 2022, pp. 37–61. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-89328-6_3

[2] B. S. E. Zoraida and G. Indumathi, "A comparative study on software-defined network with traditional networks," *TEM Journal*, vol. 13, no. 1, pp. 167–176, 2024. [Online]. Available: https://www.temjournal.com/content/131/TEMJournalFebruary2024_167_176.html

[3] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013. [Online]. Available: https://doi.org/10.1109/MCOM.2013.6461195

[4] S. M. M. Gilani, T. Hong, W. Jin, G. Zhao, H. M. Heang, and C. Xu, "Mobility management in ieee 802.11 wlan using sdn/nfv technologies," *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, no. 67, 2017. [Online]. Available: https://link.springer.com/article/10.1186/s13638-017-0856-9

[5] C. Perkins, "Ip mobility support," https://datatracker.ietf.org/doc/html/rfc2002, 1996, rFC 2002.

[6] M. Lali, G. Mustafa, and M. Awais, "Performance evaluation of software defined networking vs. traditional networking," *The Nucleus*, vol. 54, no. 1, pp. 27–32, 2017. [Online]. Available: https://thenucleuspak.org.pk/index.php/Nucleus/article/view/95

[7] R. Duo, C. Wu, T. Yoshinaga, J. Zhang, and Y. Ji, "Sdn-based handover scheme in cellular/ieee 802.11p hybrid vehicular networks," *Sensors*, vol. 20, no. 4, p. 1082, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/4/1082

[8] B. Chen and M. Ruffini, "Resource cooperation in mec and sdn based vehicular networks," *arXiv preprint arXiv:2308.04564*, 2023.

[9] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, and M. Zhu, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM*, 2013, pp. 3–14.

[10] E. Marin, J. Martins, S. K. Fayaz, J. Mirkovic, and R. de Oliveira, "An in-depth look into sdn topology discovery mechanisms: Novel attacks and practical countermeasures," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, pp. 1735–1747.

[11] J. Kim, Y. H. Lee, and K. H. Kim, "Sdn-based seamless mobility management for b5g services in high-speed railways," *IEEE Access*, vol. 8, pp. 208 282–208 292, 2020.

[12] S. B. Islam, H. S. Hassanein, and M. Maheswaran, "An sdn framework for seamless mobility in enterprise wlans," *Journal of Network and Computer Applications*, vol. 88, pp. 157–166, 2017.

[13] A. Benslimane, T.-W. Um, and T. Larsson, "Exploring software defined networks for seamless handovers in vehicular networks," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 62–69, 2017.

[14] A. Kumar, "Sdn-based load balancing," https://github.com/akshayxml/SDN-based-Load-Balancing, 2020, accessed: 2024-05-12.