The Impact of mobility in SDN-based protocols: evaluation and solutions

Master Thesis

MD Ashik Rezwan

Matriculation Number

3164894

This work was submitted to the

Institute of Computer Science IV

University of Bonn, Germany

Adviser(s):

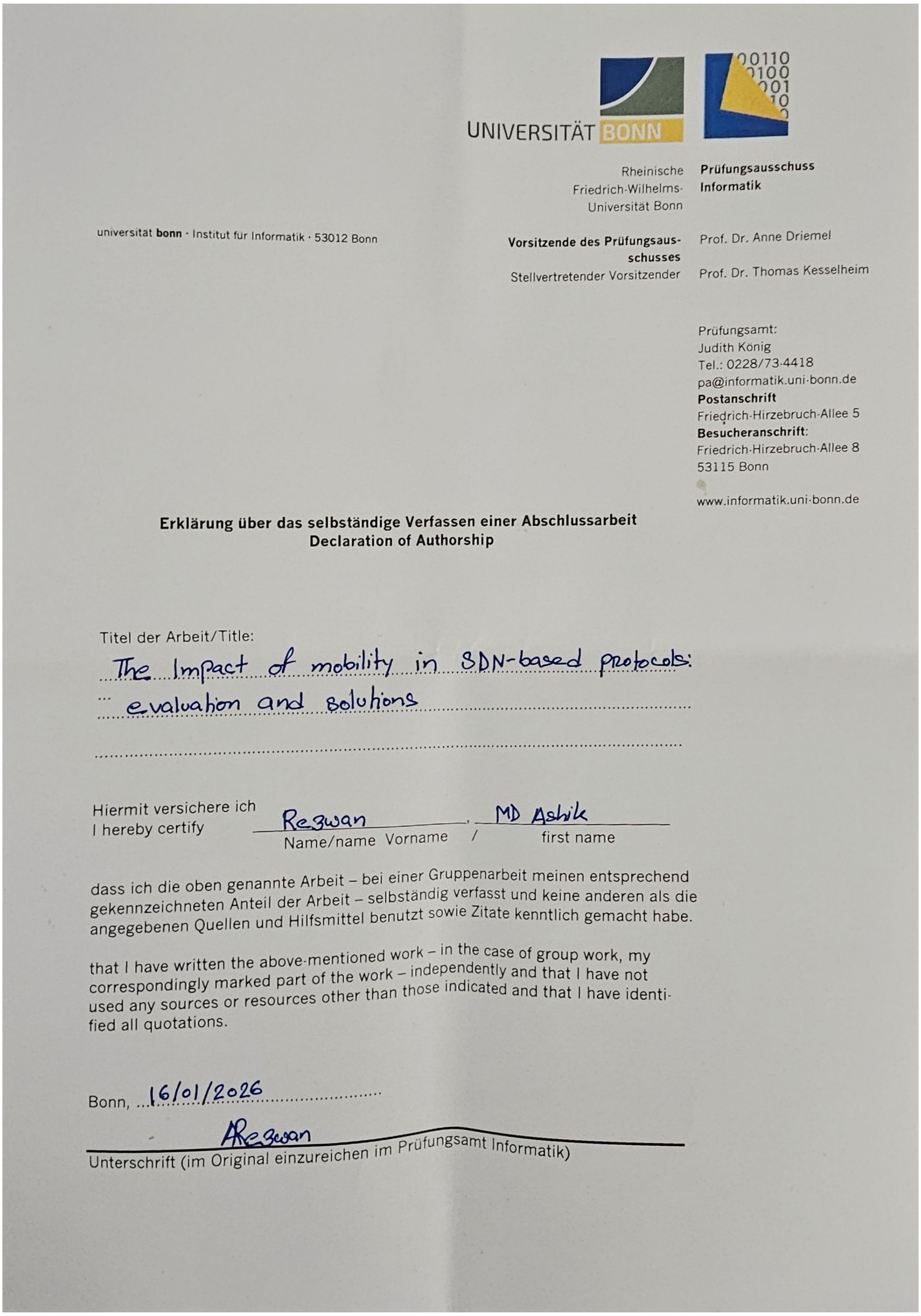
Dr. Paulo Henrique Lopes Rettore
Dr. Bruno P. Santos
Mr. Philipp Zißner

Examiners:

Prof. Dr. Michael Meier and Dr. Paulo Henrique Lopes Rettore

Registration date: 18-07-2025
Submission date: 19-01-2026

In collaboration with the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE), Bonn, Germany



UNIVERSITÄT BONN

Rheinische
Friedrich-Wilhelms-
Universität Bonn

Prüfungsausschuss
Informatik

universität **bonn** · Institut für Informatik · 53012 Bonn

Vorsitzende des Prüfungsaus-schusses Prof. Dr. Anne Driemel

Stellvertretender Vorsitzender Prof. Dr. Thomas Kesselheim

Prüfungsamt:
Judith König
Tel.: 0228/73-4418
pa@informatik.uni-bonn.de
Postanschrift
Friedrich-Hirzebruch-Allee 5
Besucheranschrift:
Friedrich-Hirzebruch-Allee 8
53115 Bonn

www.informatik.uni-bonn.de

Erklärung über das selbständige Verfassen einer Abschlussarbeit
Declaration of Authorship

Titel der Arbeit/Title:
The Impact of mobility in SDN-based protocols:
evaluation and solutions

Hiermit versichere ich
I hereby certify Regwan , MD Ashik
Name/name Vorname / first name

dass ich die oben genannte Arbeit – bei einer Gruppenarbeit meinen entsprechend gekennzeichneten Anteil der Arbeit – selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

that I have written the above-mentioned work – in the case of group work, my correspondingly marked part of the work – independently and that I have not used any sources or resources other than those indicated and that I have identified all quotations.

Bonn, 16/01/2026

ARegwan
Unterschrift (im Original einzureichen im Prüfungsamt Informatik)

Abstract

As Software-Defined Networking (SDN) evolves, its ability to handle highly mobile environments has become a major challenge. This study examines how three popular SDN controllers: Open Network Operating System (ONOS), RYU, and Open-FLow (OF), perform when paired with four distinct mobility models: Manhattan Grid (MG), Gauss-Markov (GM), Random Direction (RD), and Random Waypoint (RWP). The primary goal was to determine if transitioning from a traditional reactive flow management system to a proactive one could mitigate the performance drops that typically occur when network layouts change.

The results show that while Software-Defined Network (SDN) can scale, standard reactive setups really struggle with high-mobility models like RWP. Because RWP is so unpredictable, it consistently caused the highest packet loss in almost every test. One of the most interesting findings was a major shift in how the RYU controller performed. While RYU was actually the weakest performer in small or medium-sized networks, it proved to be much more robust and scalable in large-scale, high-stress situations, provided it was used with a proactive strategy. In contrast, ONOS and OF were more reliable for smaller setups, but they eventually hit a performance ceiling and were outperformed by RYU under heavy stress.

Ultimately, this research suggests that a proactive approach is essential for keeping a network functional as it gets faster and more crowded. While SDN architecture can technically handle a high number of nodes, the packet loss seen in reactive mode shows that standard SDN isn't fully reliable for high-mobility use cases yet. These findings offer a practical guide for choosing the right controller and flow management style based on the specific scale and movement patterns of a network.

Acknowledgments

My deepest thanks go to Dr. Paulo Henrique Lopes Rettore, Dr. Bruno P. Santos, and Mr. Philipp Zißner for their mentorship and support during my Master's degree. Their expertise and feedback were instrumental in shaping this thesis.

Contents

1

Introduction

The emergence of SDN, has evolved into a disruptive paradigm of modern networking that has overcome some of the limitations inherent to traditional network architectural models. The escalating need for scalable, programmable, and affordable networks has exposed many of the disadvantages of legacy network infrastructures, that typically have been inflexible and difficult to manage [5]. The key innovation of SDNs is to de-couple the Control Plane from the Data Plane; thus providing the capability to centrally control and program the network [5]. This new architecture provides greater flexibility in managing resources within the network, better network automation capabilities and increased responsiveness to changing traffic demands.

SDN, initially used in data centers and wired network environments through the use of Open Networking Foundation (ONF) created openflow protocol [5], is now being used in wireless and mobile network environments as a result of extending the principles of SDN to those environments, creating software-defined wireless networks Software-Defined Wireless Networks (SDWN) [20]. The growth in the demand for seamless connectivity between mobile device ecosystems and iot environments [10] has prompted this expansion. While SDN has been successful in transforming fixed wired network infrastructure into dynamic network infrastructures, it will face additional challenges in adapting to mobile environments.

Mobility is one of the main difficulties in wireless SDN, since wireless SDN does not have a fixed environment. Mobility introduces high levels of change (i.e., frequent topological changes), unreliability (i.e., unpredictable link failures) and a need for continuous handover between network devices. All three characteristics of mobile networks create inefficiencies in the placement of controllers within wireless SDN-based architectures, increase latencies, and add overhead to the control plane of wireless SDN-based architectures [15]. In addition, existing SDN architectures are designed primarily with fixed infrastructure in mind and lack a means to effectively manage dynamic routing, handover and real-time network configuration [22]. Therefore, it is essential to understand how SDN will be affected by mobility and what modifications are needed to make SDN more adaptable as mobile networks continue to evolve.

The purpose of this research is to assess the effects of mobility on SDN based protocols and identify limitations in current SDN implementations and potential solutions to increase network performance, reliability and handover efficiency. This study examines the performance of SDN in a mobile environment, specifically examining the critical areas of latency, handover time and packet loss, and identifies the critical areas of limitation and bottleneck that occur due to mobility. Finally, this study provides solutions and improvements to the adaptability of SDN in highly dynamic networks and validate these solutions using simulations and experiments and compare the performance of multiple SDN models in a variety of mobility patterns.

1.1 Problem Statement

The need for effective and dependable network topologies has increased due to the quick spread of wireless devices and the Internet of Things (IoT), especially as we move toward small and microcells. Mobility management now faces significant obstacles due to this change, including higher latency, increased packet loss, and interruptions to real-time communication during handovers.

The advancement of network administration has been made possible by Software-Defined Network (SDN) and OF, which offer centralized control and flexibility. Even with their fundamental advantages, there is still much to learn about how well they manage mobility in crowded, dynamic settings. SDN-enabled mobile networks face several challenges, including long handover times, scalability constraints, and elevated control-plane load. While most research focuses on enhancing SDN for mobility management, it does not fully examine the baseline performance of common SDN systems that have not been updated. Due to this gap, the impact of mobility on SDN protocols remains unclear, making it difficult to identify where modifications would most effectively enhance their performance in mobile scenarios. In this sense, the following research questions arise:

- How effective are standard SDN protocols in managing dynamic and rapidly changing network conditions?
- To what extent do standard SDN implementations address the Scalability (SC) and Reliability (RE) requirements of large-scale mobile network infrastructures?

1.2 Solution

We used Mininet-WiFi to simulate an environment with various mobile nodes (4, 32 & 64) and various Access Points (APS) (4 & 16) all connected in a mesh network in order to address these problems. We recorded data in accordance with various mobility patterns that represent different real-life scenarios. The experiment has been conducted with multiple controllers for ten rounds in total. To provide a clear image of the scenarios, their average is then calculated. In order to investigate whether the outcome might be improved, we later conducted the same experiment using the flow entries from the normal experiment (reactive) as pre-installed (proactive) flow entries. All of our findings center on how proactive experiences enhance reactive ones.

1.3 Thesis Structure

This thesis is arranged in the following manner. After the introduction comes the background of the thesis, along with a description of the mobility and the scenarios they depict in real life. Section 2.1 provides a review of related works, highlighting existing research on SDN in mobile environments. The system design are presented in Sections 3. Section 4 provides the experimental setup used for all analysis and a detailed evaluation of the experimental results. Finally comes the conclusion in section 5, and also outlines the future work.

2

Background

Network agility is achieved using SDN, which divides the data plane (forwarding hardware) from the control plane (brain). In a typical scenario, when a mobile station travels between Access Points (APs), the switch must send an OpenFlow Packet-In message to the controller to query it. In high-mobility settings, this round-trip communication introduces ”control plane latency” that often exceeds the handover window, resulting in substantial packet loss. Proactive flow management, in which the controller pre-installs forwarding rules in the destination switches prior to the mobile node finishing its transition, has been used to lessen this.

Mininet-WiFi, which expands on conventional SDN emulation by adding wireless propagation models and mobility patterns, is used to assess the effectiveness of these mobility management solutions. This environment enables a direct comparison of common OF implementations by modeling real-world wireless challenges such as signal fading and handover triggers. This study's main focus is on how pre-installing the flow entries avoids the conventional SDN request-response cycle, potentially preserving a continuous data flow and stabilizing packet loss metrics under abrupt topology changes.

This study uses four different mobility models within Mininet-WiFi (MG, GM, RWP & RD) to thoroughly assess the effect of movement on SDN performance. These models specify the station's accelerations, velocities, and trajectories, which directly determine the handover occurrences. The study provides an in-depth overview of how different OF controllers handle varying degrees of topology instability by testing across multiple models, ensuring the advantages of pre-installed flow entries remain comparable across both predictable linear paths and highly dynamic, stochastic movement patterns.

The particular features of the four selected models are defined as follows in order to appropriately depict various deployment environments:

- Random Direction: The RD Mobility Model is a stochastic movement pattern where nodes independently select a random direction and speed to travel along straight walk segments. In this model, movement consists of a ”move phase,” where the node maintains a constant velocity and direction, followed by an optional ”pause phase” of a specified duration. When a node reaches the simulation boundary, it follows a predefined border behavior, such as wrapping around to the opposite side of the field or reflecting back into the area. This model is frequently chosen for network simulations because it ensures that node positions and directions are uniformly distributed throughout the movement space in a steady state. In real life, this model depicts unconstrained movement in open environments, such as pedestrians in a wide open plaza, animals in a field, or autonomous robots in a disaster recovery zone, where movement is not restricted by a road grid or specific paths but is eventually limited by physical or geographic boundaries. [4, 7, 13]

- Random WayPoint: The RWP mobility model is a random movement pattern where a mobile node travels along a ”zigzag” path consisting of straight-line segments between destinations known as waypoints. In this model, a node chooses its next waypoint uniformly at random within a predefined convex area and moves directly toward it at a velocity selected from a random distribution. Upon reaching the waypoint, the node may remain stationary for a predefined ”pause” or ”thinking” time before choosing a new destination and resuming its motion. Over time, this movement leads to a nonuniform spatial distribution where nodes are most likely to be found in the center of the area, a phenomenon known as the ”border effect”. [3, 9]

  In real life, the RWP model depicts pedestrian movement patterns and the behavior of mobile entities moving within a constrained field. Specific environments it represents include:

  - Open Public Spaces: People walking in plazas or parks where they choose a destination and move straight toward it. [9]
  - Office Buildings: The movement of employees or visitors within large commercial structures. [9]
  - Sensor Dispersal: Scenarios where mobile sensors are spread or distributed across a region. [3]

- Manhattan Grid: The MG Model is a grid-based vehicular movement pattern designed to emulate the complex road topology of an urban city environment. In this model, the map consists of organized horizontal and vertical streets, typically composed of two lanes for each direction—north/south and east/west. Unlike random models, movement is strictly restricted to these predefined grid lanes where vehicles move straight, turn left, or turn right at intersections based on specific probabilities, often set at 0.5 for straight and 0.25 for each turn. This model is an ideal choice for vehicular ad-hoc network (VANET) research because it provides a realistic simulation of high-density city scenarios where geographic constraints like buildings, trees, and roadblocks act as obstacles to communication. Furthermore, it accounts for time and space limitations by making a vehicle’s velocity dependent on the previous time slot and the speed of the preceding node, accurately reflecting the constrained, non-random nature of real-life urban traffic. [8, 16]

- Gauss Markov: The GM mobility model is a synthetic movement pattern in which the current speed and direction of a mobile node are directly influenced by its speed and direction in the preceding time interval. The model uses a single tuning parameter, α, to represent different degrees of randomness; setting α to 0 results in maximum randomness (Brownian motion), while setting it to 1 produces a linear motion. By utilizing this temporal correlation, the GM model generates realistic movement by ensuring that speed and direction changes are not abrupt. In a real-life context, it depicts intentional movement patterns where objects or people have defined target destinations, making it highly applicable for simulating the behavior of autonomous mobile anchor nodes in surveillance regions or high-speed automobiles where current velocity and location are strong indicators of the future state. [1, 25]

2.1 Related Work

Research on using SDN for mobility management is gaining momentum and will lead to enhancements in Handover Efficiency (HE), Network Scalability (NE) and user Quality of experience for diverse applications (QoE) in modern wireless environments. In the past, there have been many research areas related to mobility in SDN that have identified a number of problems with mobility in wireless systems including: pure SDN methodologies; hybrid SDN architecture methodologies; and proactive versus reactive methodologies in SDN; and SDN applications in next-generation wireless systems. However, despite the extensive research done so far on how to use SDN to develop new ideas and optimize existing ideas related to mobility in wireless systems, little research has been done comparing SDN implementations such as OF, with respect to their performance in mobile wireless system environments. The lack of evaluation of these mainstream SDN implementations leaves a number of open questions related to their upper bounds on performance and their operating characteristics in real-world mobility scenarios.

2.1.1 SDN-Based Mobility Management: Progress and Limitations

While early studies focused on SDN-driven mobility management as an alternate to PMIPv6 for lowering signaling load, and handover delay; Kuljaree et al. [22] have shown that SDN will increase reliability and throughput by enabling flow rerouting at runtime; however, Shri et al. [18] presented fast split, an SDN-based IP mobility solution that enables faster handovers by dynamically optimizing paths proactively. However, this research was limited to simulation studies and were only conducted on small scale trials and did not address the issue of large scale trial deployments and scalability.

Therefore, later studies attempted to improve scalability by focusing on distributed SDN-based architectures. Tien et al. [14] presented S-dmm, a SDN-driven Distributed Mobility Management (DMM) integration to provide fine-grained flow level control in 5G network architecture. Similarly, Sanchez et al. [17] proposed a hierarchical controller architecture for S-dmm that reduced the signaling overhead in large scale IEEE 802.11 networks. Even though both approaches had their advantages, the majority of them required extensive changes to the original SDN framework and lacked testing of standard OF-based SDN frameworks in actual-world mobile environments.

More recently Lo Bello et al. [2] developed FTS-SDN, a SDN-driven TSCH protocol for Industrial Wireless Sensor Networks (IWSNs) using SDN-wise. The authors evaluated the performance of the proposed solution in terms of maintaining the bound of latency due to mobility on a commercial hardware platform. While the authors reported positive results, they utilized a proprietary SDN protocol (SDN-wise), instead of utilizing mainstream OF-based SDN controllers. Therefore, the study shows that there is still a need for evaluating standard OF-based SDN frameworks in mobility driven scenarios.

2.1.2 Hybrid SDN Solutions for Incremental Deployment

Although there is a growing trend towards developing hybrid SDN architectures that blend SDN with traditional systems due to the complexity associated with fully migrating all SDN components, the authors are also examining the use of hybrid SDN frameworks to leverage the benefits of SDN while still supporting legacy systems. In addition to enabling dynamic traffic offload between LTE and Wi-Fi, along with SDN-based network slicing and predictive handover management, Silva et al. [19] provided a hybrid SDN framework for 4G/5G networks and a number of other applications, while providing backwards-compatibility for existing Evolved Packet Core (EPC) systems.

These hybrid solutions allow for greater ease of SDN integration into legacy systems; however, they may hide some of the limitations of implementing SDN with legacy systems. Hybrid solutions are able to support both OF-based control logic, and at least one of the two following alternatives: replacement of OF-based control logic with alternative logic or abstraction of OF-based control logic. Hybrid solutions do not make it clear how current mainstream SDN controllers (such as OpenDaylight and ONOS) will handle mobile users, which use different access technologies to connect to a SDN-enabled network. Therefore, although hybrid solutions provide some practical advantages for integrating SDN into legacy systems, they provide very little insight into the SC and RE of unaltered SDN layers when multiple access technologies are used to dynamically move data across a SDN-enabled network.

2.1.3 Proactive vs. Reactive Handover Optimization in SDN-Based Mobility

A variety of techniques have been developed to improve the speed of proactive handover optimization to reduce the amount of time required to transfer packets between networks in highly dynamic environments. The first study related to optimizing handovers in a dynamic environment were conducted by Chen et al. [6], who created a mobility-aware SDN called M-SDN. They demonstrated that if a device's predicted future location can be determined through the use of location prediction techniques, then the amount of time lost when a user transitions from one network to another can be significantly reduced. Subsequently, Clarissa et al. [12] and Labraoui et al. [11] extended this work by developing proactive path determination techniques and utilizing dynamic flow management in both wireless mesh and vehicular networks.

Later, Silva et al. [19] developed a new type of hybrid proactive/reactive handover technique. This hybrid technique includes proactive decision making regarding certain decisions (for example, selecting a Wi-Fi network), but utilizes reactive techniques to manage flows after the handover is complete. Although this provides a balance between determining the correct decision and reducing the level of control overhead, the performance of this technique has never been evaluated using standard OF-based SDN controllers, especially in environments where devices move rapidly and experience many handovers.

Tong et al. [24] designed a novel handover architecture for Software-Defined Heterogeneous Networks (SDHetNets), which incorporates three critical innovations: (1) Echo State Networks for predicting user movement, (2) a fuzzy-AHP decision-making method for selecting the best network, and (3) multipath Transmission Control Protocol (TCP) to stabilize connections. Although Tong et al.'s solution achieved better efficiency in handover operations and higher reliability in services, the solution was developed using proprietary SDN control logic, not a well-established platform such as RYU, ONOS, or OpenDaylight. This raises questions about the ability of the solution to operate successfully with systems such as RYU, ONOS, or OpenDaylight.

2.1.4 SDN for Next-Generation Networks: 5G and Vehicular Applications

SDN is becoming increasingly important in managing mobility in 5G and vehicle networks. The works of Kuljaree et al. [23] have presented SDNVMM for vehicular video streaming as well as Sorn et al. [21] have suggested SDN-NEMO for mobile network resilience. The two studies indicate that there is a lot of promise with SDN in highly mobile contexts. However, both are often implemented using customized extensions to SDN systems, and many do not provide information about how the implementations were made using standardized SDN components. Furthermore, few of them investigate the behavior of control plane latency, flow rule updates, and topology change detection using OF. These elements are very important when evaluating performance in real-world environments.

2.1.5 The Findings

Table 2.1 provides a detailed comparison of the reviewed studies, including their deployment scenarios. Most works primarily address Latency Reduction (LR), HE, SC, and Performance Optimization (PO), with some also improving RE and Reduced Signaling Overhead (RSO). However, key challenges such as Scalability with larger network (LSD) and QoE remain largely unexplored. These gaps underscore the need for further investigation into the limitations and optimization opportunities of standard SDN implementations under dynamic mobility conditions.

Category	**Focus Area**								**Network Aspects**				
	LR	HE	SC	PO	RE	RSO	LSD	QoE	Topology	Tools	Env.	Controller	Switch
SDN-Based Mobility Management													
[22]	✓	✓	✗	✓	✗	✗	✗	✗	Linear	Mininet	Emulation	RYU	-
[18]	✓	✓	✓	✓	✗	✓	✗	✗	Grid,Random	Mininet	Emulation	Floodlight	36/42
[14]	✓	✓	✓	✓	✓	✓	✗	✗	Linear	Mininet	Emulation	RYU	5
[17]	✓	✓	✓	✓	✓	✓	✗	✗	-	-	Emulation	RYU	-
[2]	✓	✓	✗	✓	✓	✓	✗	✗	Star/Mesh	Cooja/Contiki	Emulation	OpenFlow	-
Hybrid SDN Solutions													
[19]	✓	✓	✗	✓	✗	✓	✗	✓	Hybrid	-	Simulation	RYU	-
Proactive vs. Reactive Handover Optimization													
[6]	✓	✓	✗	✓	✓	✓	✗	✗	Star	-	Emulation	ONOS	-
[12]	✓	✓	✓	✓	✗	✗	✓	✗	Hybrid	Mininet	Emulation	Floodlight	19/89/178
[11]	✗	✗	✓	✓	✓	✗	✗	✗	Mesh	CORE	Emulation	4,NEON	8
[24]	✓	✓	✗	✓	✓	✗	✗	✓	-	Mininet-WiFi	Emulation	OpenFlow	-
SDN for Next-Generation Networks													
[23]	✓	✓	✓	✓	✓	✗	✗	✗	-	NS3,SUMO	Simulation	OpenFlow	-
[21]	✓	✓	✓	✓	✓	✓	✗	✗	Mesh	-	Simulation	OpenFLow	-

Latency Reduction (LR); Handover Efficiency (HE); Scalability (SC); Performance Optimization (PO); Reliability (RE); Reduced Signaling Overhead (RSO); Scalability with larger network (LSD); Quality of experience for diverse applications (QoE)

Table 2.1 Findings regarding literature review

3

Design

The experimental design is divided into 2 phases: 'Deployment of the Experimentation Infrastructure' and 'Evaluation'. This structure ensures that every stage of network configuration and performance analysis is carefully evaluated. Figure 3.1 shows the design of the initial phase.

3.1 Deployment of the Experimentation Infrastructure

This initial phase focuses on establishing a stable network environment for executing the simulation. It is subdivided into three more phases: 'Infrastructure Test', 'Experimental Phase', and 'Pre-Installation'.

3.1.1 Infrastructure Test

There are three main parts to this phase, that serve as the foundation for a successful experiment. First, we introduce a SDN controller, APS, and a few stationary stations spread out throughout the grid. Next, we use the pingall command to check connectivity, which is the next step. If communication between nodes fails after executing the command, we adjust the APS location by moving them closer until a proper spacing is found, at which point communication is completely successful. After it has been corrected, we proceed to the experimental phase.

3.1.2 Experimental Phase:

Once we passed the checking phase, we scaled up the network with different numbers of stations and APS connected by a mesh topology as part of our performance

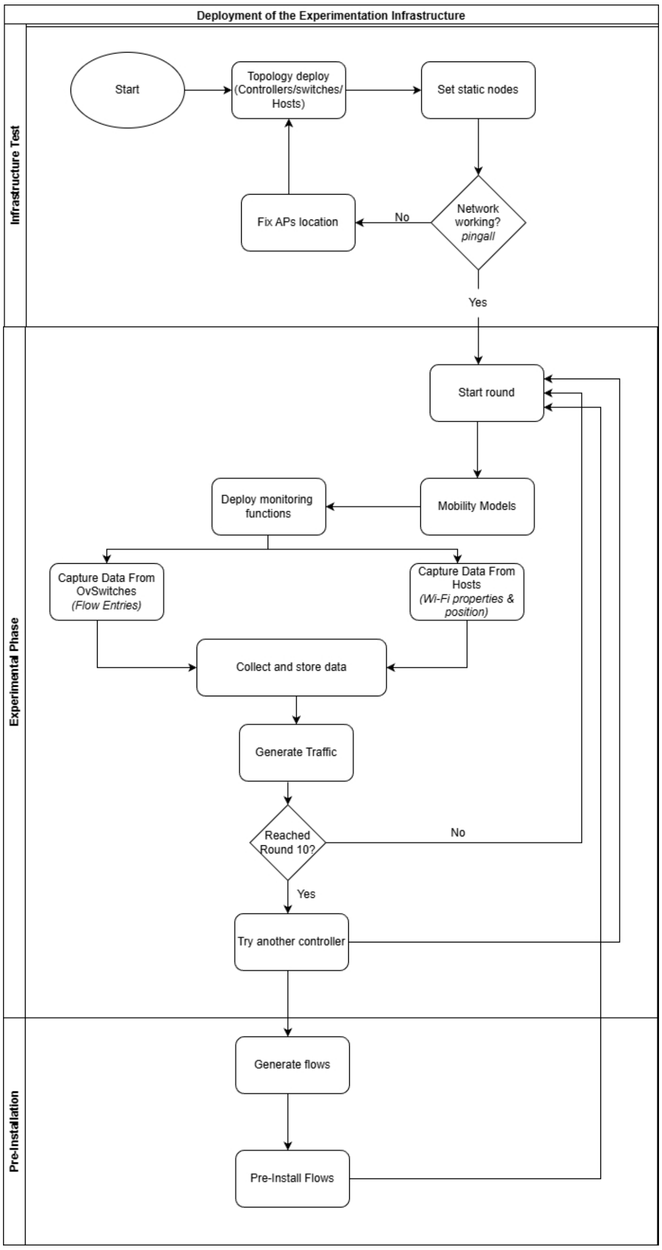


Figure 3.1 Deployment of the Experimentation Infrastructure.

investigation. We used four mobility models (RD, RWP, MG, and GM) to simulate real-world movement while evaluating three different SDN controllers: the basic Mininet controller (OF controller), RYU, and ONOS. To generate traffic, we used Distributed Internet Traffic Generator (DITG) and Internet Performance Working Group (iPerf) in a pattern in which even-numbered stations served as receivers and odd-numbered stations as senders. In this phase, we concentrate mainly on three tasks: introducing mobility, starting communication, and capturing data from both APS and stations.

3.1.2.1 Mobility Models

The first task of the experiment was to implement mobility patterns on every network node. Mobility parameters were changed during the experiment to mimic different real-world situations rather than employing a static configuration.

To ensure experimental reproducibility and consistency across varying node densities, a custom Python-based trajectory generator, the MobilityGenerator class, was developed. This framework automates the creation of .dat trajectory files, allowing the same movement patterns to be replayed across multiple experimental iterations. The system is defined by a modular architecture that supports four stochastic mobility models and a unique initialization sequence termed the Intro Phase.

- Phase 1 (Intro Phase): Unlike standard simulations that begin with a static or arbitrary node distribution, this framework implements a 15-second dynamic entry sequence. All nodes originate from a central coordinate $P_{fixed} = (60, 60)$ and transition toward their model-specific starting positions. This allows the stations to be assigned to their specific APS before the experiment starts communication.

  The core movement logic is governed by the Euclidean distance formula:

  $$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

  Through the _move_towards function, the system calculates incremental coordinate updates based on a velocity vector adjusted to ensure all nodes reach their designated operational starting points simultaneously. This phase effectively simulates the deployment of mobile units into the field of interest.

- Phase 2 (Mobility Model): The framework implements four distinct models to simulate various operational environments:
  - Random Waypoint: Nodes move toward randomly selected destinations within the simulation boundaries at a constant velocity. Upon arrival, a new waypoint is instantly generated. RWP serves as a benchmark for simulating unpredictable, jerky user movement.
  - Gauss-Markov: To achieve more realistic, fluid trajectories, this model introduces temporal dependency using a tuning parameter $\alpha = 0.75$. The current velocity is a weighted average of the previous state and a Gaussian noise component, providing the nodes with ”memory.” Boundary interaction is managed via reflection, where nodes ”bounce” off the simulation edges.

  - Manhattan Grid: Tailored for urban topographies, this model constrains movement to a grid with a 50-unit block size. Nodes are initialized at intersections and restricted to cardinal directions (North, South, East, West). At each junction, nodes randomly select a neighboring intersection, mimicking vehicular or pedestrian traffic in a city layout.
  - Random Direction: To mitigate the ”density peak” phenomenon, a common RWP issue where nodes cluster in the center, the RD model requires nodes to travel in a straight line at a chosen angle until a boundary is reached. A new direction is only selected after a ”wall” collision, ensuring a uniform spatial distribution across the field.

- Phase 3 (Data Persistence and Configuration): The framework offers granular control over the simulation environment, including the configuration of speed and the spatial boundaries ($x_{min}, y_{min}, x_{max}, y_{max}$). Upon completion, the save_output method exports time-series positional data into standardized .dat files for each individual node. This structured output ensures seamless integration with network simulators and post-simulation visualization tools.

```
Input: Num Nodes $N$, Boundaries $[x_{min}, x_{max}, y_{min}, y_{max}]$, Simulation Time $T$, Velocity $s$, Model $M$, Intro Duration $t_{intro}$, Fixed Start $P_{fixed}$, GM parameter $\alpha$, Mean velocity $\mu$
Output: Spatio-temporal Node Trajectories $\mathcal{D}$ (Standardized .dat format)
Initialization and Pre-calculation
Initialize empty trajectory list $\mathcal{D}_i$ for each node $i \in \{1 \ldots N\}$
Determine model-specific start positions $P_{start}[i]$ based on model $M$
for each node $i \in \{1 \ldots N\}$ do
    $\mathcal{D}_i[0] \leftarrow P_{fixed}$                    % All nodes begin at (60, 60)
end for
Phase 1: Intro Phase (Deployment Transition)
for each node $i \in \{1 \ldots N\}$ do
    $dist \leftarrow \sqrt{(P_{start}[i].x - P_{fixed}.x)^2 + (P_{start}[i].y - P_{fixed}.y)^2}$
    $s_{intro} \leftarrow dist/t_{intro}$          % Calculated to ensure simultaneous arrival
    for $t = 1\ t_{intro}$ do
        $P_{next} \leftarrow \text{CalculateStep}(\mathcal{D}_i[t-1], P_{start}[i], s_{intro})$
        $\mathcal{D}_i[t] \leftarrow P_{next}$
    end for
end for
Phase 2: Mobility Model (Stochastic Simulation)
for $t = (t_{intro} + 1)\ (t_{intro} + T)$ do
    for each node $i \in \{1 \ldots N\}$ do
        if $M =$ 'Random Waypoint' then
            Move toward Target; if reached then Target $\leftarrow \text{Random}(x, y)$ $M =$ 'Gauss-Markov'
            $v_t \leftarrow \alpha v_{t-1} + (1 - \alpha)\mu + \sqrt{1 - \alpha^2}\mathcal{N}(0, 1)$
            Update $P_{next}$ and apply boundary reflection $M =$ 'Manhattan Grid'
            Constrain to grid; if intersection then Select random neighbor $M =$ 'Random Direction'
            Move at $\theta$; if boundary hit then $\theta \leftarrow \text{Uniform}(0, 2\pi)$
        end if
        $\mathcal{D}_i[t] \leftarrow P_{next}$
    end for
end for
Phase 3: Data Persistence and Configuration
Export $\mathcal{D}$ to standardized .dat files                 % Save_output method
return $\mathcal{D}$
```

Algorithm 1 Mobility Generation Framework

3.1.2.2 Deploy monitoring functions

This phase concentrates on gathering comprehensive data on the network's performance under various circumstances. In this stage, relevant information is collected, and it is methodically arranged for further examination. The data captured here are Flow entries from APS and from stations where we captured WiFi properties and the position of the nodes.

Capture Data From OvSwitches

Due to architectural differences in how SDN controllers manage and expose flow telemetry, specialized monitoring scripts were developed for each controller. These agents run concurrently with the primary simulation script to ensure consistent, accurate flow-state acquisition.

- Mininet Default Controller (OF): To capture the flows, the OF pipeline is polled frequently across all simulated APS via the flow capture mechanism, which is implemented as an asynchronous telemetry agent. The system has a multi-threaded design, creating concurrent threads to run 'ovs-ofctl dump-flows' commands for every APS instance. This allows the system to take a nearly simultaneous snapshot of the global network state every second. A parsing engine is then used to separate match fields, flow counters, and action sets (such as output ports and packet modification instructions) from these unprocessed ASCII snapshots and convert them into an organized relational format. The agent uses a mutual exclusion (mutex) lock for thread-safe I/O operations and a deduplication algorithm based on a composite unique key (flow cookie, network addresses, and durations) to preserve data integrity and optimize storage during extended simulation rounds. This ensures that only unique state transitions are archived with their corresponding experimental round identifiers and synchronized timestamps.

- RYU controller: A centralized telemetry agent that communicates with the Northbound REST API of the SDN controller facilitates the collection of flow information. The agent periodically polls all APS, each identifiable by its 64-bit Data Path Identifier (DPID), focusing on the controller's statistical endpoints (specifically, the /stats/flow/ URI). The agent sends an HTTP GET request every second to retrieve detailed flow descriptors in JSON format. These descriptors are then analyzed to extract important performance characteristics, such as flow durations, byte quantities, and granular packet counts. This information is stored in a structured, CSV-based relational database after being mapped to specific match criteria, such as ingress ports and MAC address pairings. A continuous, synchronized longitudinal dataset that links network load to specific experimental rounds is ensured by the implementation's robust exception handling for network timeouts and HTTP errors.

- ONOS controller: A dedicated flow-state monitoring agent that extracts and normalizes the OpenFlow 1.3 pipeline state in real-time, supports the experimental data gathering. Using a local Southbound interface, this agent calls 'ovs-ofctl dump-flows' to get immediate snapshots of the forwarding tables and 'ovs-vsctl' to retrieve and store persistent Datapath Identifiers (DPIDs). Match-action tuples, flow priorities, and statistical counters ($n\_packets$, n_bytes) are isolated from the raw ASCII stream by a regular-expression-based parsing engine that converts complicated datapath syntax into a normalized schema. The agent re-formats this data into a standard CSV structure. The system operates on a 1.0 Hz polling frequency, appending each state transition to a longitudinal dataset tagged with high-precision timestamps and experimental round identifiers, thereby enabling the correlation of specific flow-table modifications with the dynamic movement of mobile stations.

Capture Data From Hosts

Two distinct datasets were extracted from the mobile stations to characterize network performance and physical dynamics. First, link-layer telemetry was gathered via the iwconfig utility to monitor association and disassociation events between stations and APS, providing a granular view of handover scenarios. Second, the spatial coordinates of each station were recorded at one-second intervals. This positional data enables the derivation of critical mobility statistics, including Euclidean distances to infrastructure nodes, speed, and acceleration.

- Position: A specialized background monitoring thread was implemented to record each mobile station's real-time Cartesian coordinates, enabling correlation between network performance and actual node movement. The tracking method retrieves the instantaneous (x, y, z) positions of each target node by querying the Mininet-WiFi simulation engine at a sampling frequency of 1.0 Hz. These spatial data points are stored in station-specific repositories after being precisely timestamped to the microsecond level. The subsequent derivation of crucial mobility metrics, such as inter-node Euclidean distance, node speed, and acceleration patterns, is crucial for examining how route loss and physical displacement affect packet delivery ratios.
- WiFi Properties: In parallel with spatial tracking, the experimental framework utilizes a persistent monitoring agent to record the link-layer status of the wireless interfaces via the iwconfig utility. To retrieve essential association parameters, such as the Service Set Identifier (SSID), operational frequency, and the Medium Access Control (MAC) address of the associated APS, asynchronous system calls are used. The agent tracks the station's movement across different infrastructure nodes and separates handover events using regular expression-based parsing of the raw command-line output. The resulting dataset provides a longitudinal view of the association state and connection persistence, which, when synchronized with the mobility logs, allows for a comprehensive evaluation of handover triggers.

3.1.2.3 Generate Traffic

In order to establish a dynamic communication environment, User Datagram Protocol (UDP) traffic is introduced at this phase. Nodes in pairs exchange packets of a fixed size; the total volume is determined by the magnitude of the scenario, with

140 packets for small-scale scenarios and 299 packets for bigger ones. For nodes to continue actively exchanging data during the simulation, the communication layer is important.

3.1.3 Pre-Installation

This phase focuses on pushing the already generated flows captured from the reactive method into the APS before communication starts between nodes. A uniform injection approach was not possible since the three different SDN controllers used different data formats for flow capture. To ensure a smooth restoration of flow entry into the APS, three distinct proactive approaches were developed to account for the unique data recording structures of each controller. Once the flows are pre-installed, we repeat the Experimental phase to run and collect data generated by the proactive method.

3.1.3.1 Pre-Install OpenFlow controller

The experimental framework uses a Proactive Flow Pre-installation technique to assess how control-plane overhead affects network performance. Before any data transmission is initiated, each APS must manually populate its OpenFlow tables. Asynchronous flow entries are created via the add-flow command by mapping particular match criteria, such as source/destination IP addresses and UDP destination ports, to forwarding instructions based on a predetermined set of communication pairs. Additionally, a high-priority ARP flow with a FLOOD action is pre-installed to guarantee network-wide reachability. The conventional request-response cycle between the switch and the controller (the Packet-In and Flow-Mod exchange) is circumvented by utilizing the set-fail-mode command to switch the APS to standalone (or secure) mode. By pre-loading these rules, the framework isolates the data plane's raw performance and prevents the switch from falling back to controller dependency during the experiment.

A validation sub-process is carried out to guarantee the datapath's integrity after proactive flows are injected. To obtain a snapshot of the current flow table from each APS, the system calls 'ovs-ofctl dump-flows'. After tokenizing this raw output, a customized parsing engine extracts key-value pairs for action sets and match fields, which are subsequently stored in a normal CSV relational format.

```
Input: Network Object $G$, Set of Communication Triplets $P = \{(s,r,p)_1,\dots,(s,r,p)_n\}$, Experimental Round $R$, Protocol $T$
Output: Proactive Flow Instantiation
Initialization
Initialize empty list $L_{flows}$
$f_{arp} \leftarrow \{priority = 100, arp, actions = FLOOD\}$
Add $f_{arp}$ to $L_{flows}$
Bi-directional Rule Generation
for each $pair(s,r,p)$ in $P$ do
    $IP_s \leftarrow s.IP$, $IP_r \leftarrow r.IP$
    $f_{fwd} \leftarrow \{priority = 500, proto = T, src = IP_s, dst = IP_r, tp\_dst = p, action = FLOOD\}$
    $f_{rep} \leftarrow \{priority = 500, proto = T, src = IP_r, dst = IP_s, tp\_src = p, action = FLOOD\}$
    Add $f_{fwd}$ and $f_{rep}$ to $L_{flows}$
end for
Deployment and Verification
for each Access Point $AP_i$ in $G$ do
    Set $AP_i$ fail-mode to $standalone$
    for each $flow$ in $L_{flows}$ do
        Execute Southbound Command: ovs-ofctl add-flow $AP_i, flow$
    end for
    Telemetry Archiving
    $D_{raw} \leftarrow$ ovs-ofctl dump-flows $AP_i$
    Call ParseAndSaveFlows($AP_i, D_{raw}, R$)
end for
```

Algorithm 2 Pre-Installation of flowentries in OpenFlow controller

3.1.3.2 Pre-Install RYU controller

In order to evaluate network performance under predetermined traffic patterns, the experimental framework uses a historical replay method. Control plane synchronization, proactive protocol configuration, and restorative flow injection are the three synchronized processes used to accomplish this.

Initially, the system polls the SDN controller's REST API using a blocking synchronization function (wait_for_switches). This guarantees that before any data is pushed, all Datapath Identifiers (DPIDs) are active and the global topology is fully registered.

Second, the system executes a proactive configuration step to ensure reachability. Using the ovs-ofctl utility, an ARP flooding rule (”priority=90,arp,actions=flood”) is injected into each switch via OpenFlow 1.3. This prevents packet loss during the address resolution phase, allowing the replay to proceed without controller intervention for basic discovery.

Finally, the system extracts the exact flow-matching criteria (InPort, SrcMAC, DstMAC) and action sets from a particular previous experimental round by parsing historical CSV information. After that, these entries are sent to the controller's /stats/flowentry/add endpoint in JSON-formatted OpenFlow FlowMod messages.

```
Input: Network Object $G$, CSV Database $D$, Target Round $R_{target}$, Controller $C$
Output: Proactive Flow Instantiation
Phase 1: Controller Synchronization
while Controller $C$ has not registered all $AP \in G$ do
    Poll $C.API$ for connected $DPIDs$
    Wait 1 second                                   % Prevent API flooding
end while
Wait 5 seconds for controller stability
Phase 2: Proactive Protocol Configuration
for each Access Point $AP_i$ in $G$ do
    Execute shell command on $AP_i$:
    ovs-ofctl add-flow $AP_i$ -O OpenFlow13 "priority=90,arp,actions=flood"
end for
Phase 3: Restorative Flow Injection
for each Access Point $AP_i$ in $G$ do
    $Live\_DPID \leftarrow G.getDPID(AP_i)$
    Open CSV file $D[AP_i]$
    for each $row$ in $D[AP_i]$ do
        if $row.Round == R_{target}$ and $row$ is unique then
            Construct Flow Object $F$:
                $Match \leftarrow \{InPort, SrcMAC, DstMAC\}$
                $Action \leftarrow Parse(row.Action)$
            Post $F$ to $C.API$ via /stats/flowentry/add
        end if
    end for
end for
```

Algorithm 3 Pre-Installation of Flowentries in RYU controller

3.1.3.3 Pre-Install ONOS controller

A proactive flow instantiation technique is designed that uses past traffic data to populate the OpenFlow tables of network nodes (APS and switches). The script creates distinct flow rules by extracting match fields like MAC addresses and ingress ports from CSV-based flow records. The OpenFlow 1.3 protocol is then used by the ovs-ofctl software to push these rules to the switches. This implementation's "headless" architecture is a vital component; it adds a proactive ARP flooding rule (priority 100) to every Access Point and filters out any rules that call for controller intervention. By doing this, the latency overhead of reactive "packet-in" queries to a central controller is eliminated, allowing the network to carry out autonomous forwarding and address resolution.

The approach uses a deduplication logic with a hashing set to guarantee that each distinct flow-match-action triplet is installed only once in order to maximize efficiency. By doing this, redundant configuration commands are avoided, and the virtual switches' TCAM (Ternary Content-Addressable Memory) resources are preserved. The system creates a deterministic networking environment appropriate for performance benchmarking and edge-computing scenarios where controller reachability may be limited by pre-installing these forwarding pathways before the simulation starts.

```
Input: Network Topology $G$, CSV Repository $D$, Round $R$
Output: Proactive Flow Instantiation
for each Access Point $AP_i$ in $G$ do
    $f \leftarrow$ Load CSV for $AP_i$ from $D$
    $S_{unique} \leftarrow \emptyset$                                % Initialize deduplication set
    for each $row$ in $f$ do
        $M \leftarrow$ ConstructMatch($row.In\_Port, row.MAC$)
        $A \leftarrow$ NormalizeAction($row.Action$)
        if $A$ involves Controller or $\{M, A, row.P\} \in S_{unique}$ then
            continue                                          % Filter reactive dependencies
        end if
        Deploy flow via Southbound API: ovs-ofctl add-flow
        Add $\{M, A, row.P\}$ to $S_{unique}$
    end for
end for
Inject global ARP rule: priority=100, arp, action=flood      % Headless baseline
```

Algorithm 4 Pre-Installation of Flowentries in ONOS controller

3.2 Evaluation Phase

Once all data is collected using the reactive method and proactive, we move on to the next phase. Here, we divided the whole design into two parts: 'Merge and Analyze' and 'Conclusion' phase.

3.2.1 Merge and Analyze

In this section, we focus on refining the data and making it suitable for analysis. This section can be further classified into 3 phases: Data Merging and Statistical Analysis, Analyze Data, and Visualization (Graph Plotting).

3.2.1.1 Data Merging and Statistical Analysis

The primary objective of this stage is to combine various datasets into a single structure. We combine the WiFi characteristics of the mobile nodes (gathered from iwconfig) with their respective physical coordinates and DITG summary files using automated scripts. Because it provides precise measurements, including packet loss, bitrate, jitter, and latency, the D-ITG data is essential for packet-level analysis. We produce an extensive dataset by syncing these files using exact timestamps.

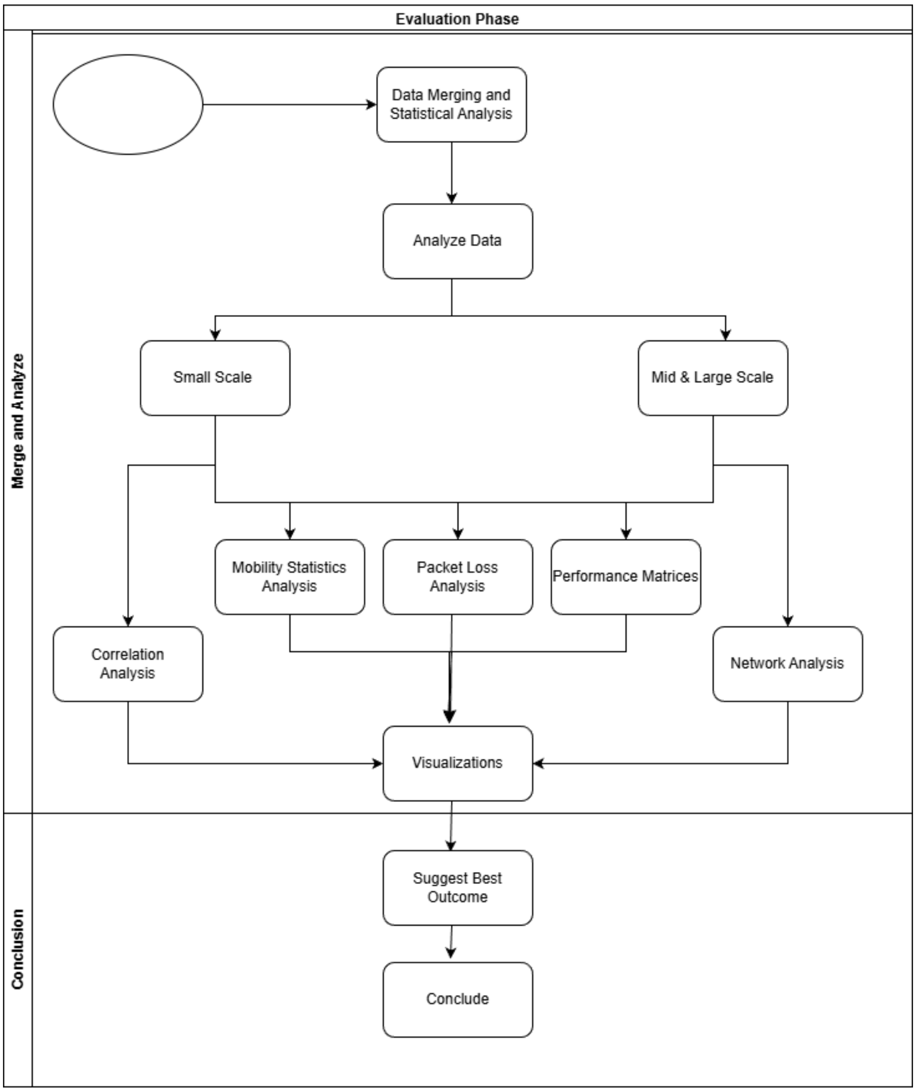


Figure 3.2 Evaluation Phase.

In order to make handover analysis easier and for calculating the distance from APS, we add two more features into the data. An association state that consists of a binary column is created, with '0' denoting a disconnected state and '1' denoting an association with an APS. Another for APS coordinate, which uses the corresponding SSID to map the fixed coordinates of the APS to the mobile stations. We also extract a number of physical measures that are crucial for assessing network performance in connection to mobility using the combined positional data.

Euclidean Distance

The 3D Pythagorean theorem is used to calculate the distance traveled between two time steps or the distance between the mobile station and the AP. The straight-line displacement between coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2) is thus given:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Instantaneous Speed

The rate of displacement over a given time interval (Δt) is used to compute velocity. We define speed as follows because the data is taken at distinct intervals:

$$v = \frac{d}{\Delta t}$$

Acceleration

Acceleration is computed by measuring the change in velocity (Δv) during the elapsed period in order to understand the dynamics of the node's movement:

$$a = \frac{v_2 - v_1}{\Delta t}$$

3.2.1.2 Analyze Data

We created a script that uses the DITG log files and the statistical files for additional analysis before converting them to plots to improve clarity. Here, we concentrate on five unique aspects of the station. We compute their correlation analysis (only for small-scale scenarios), packet loss analysis, performance metrics, mobility statistics, and associated station analysis (for mid and large-scale only).

Packet Loss Analysis

We used a generic formula to determine the average packet loss (%) across various scenarios and varying numbers of packets (140 for small-scale and 299 for mid- to large-scale). To gain an understanding of the entire scene, we focused on the average of a percentage of stations at the mid-to-large scale, while focusing on individual stations at the small scale. The following is the equation that was used in the analysis:

$$L_{\%} = \left(\frac{total\_packet - \left(\frac{1}{n}\sum_{i=1}^{n} P_i\right)}{total\_packet} \right) \times 100$$

At the mid and large scales, the same methodology was applied, calculating the average across all stations rather than focusing on individual nodes. This approach provides an overall evaluation of the entire network scenario. The formula used is:

$$L_{Scenario} = \frac{1}{N}\sum_{j=1}^{N} \bar{L}_{\%j}$$

Where,

- $L_{\%}$: Average Packet Loss Percentage
- total_packet: Total Expected Packets
- n: Sample size
- P_i: Received Packets per Sample
- $\sum_{i=1}^{n} P_i$: Aggregate Received Packets
- $\frac{1}{n}\sum P_i$: Mean Received Packets ($\bar{P}_{recv}$)
- $L_{Scenario}$: The total average packet loss for the entire scenario
- N: The total number of receivers in the sample
- $\bar{L}_{\%j}$: The average packet loss percentage for station j

Later, we visualize this with 95% Confidence Intervals (CI) as vertical error bars to measure the average results variance and reliability. All the bar plots are made according to the reactive and proactive controllers used, along with the mobility used.

Performance Metrices

Here, we focused mainly on generating plots for Connection, Delay, Jitter, Packet Loss, Bitrate, Speed, and Distance in time series representation and then forming an overall average performance table.

To calculate the connection graph, we determined the mean for each second across all test rounds by filtering the data for connected states (Connection == 1). The first row uses an initial function, whereas the remaining rows do not. The ideal value (1) is where it begins. The line falls to zero for the duration of the handover if the Handover Analysis finds a frequent disconnection (occurring in ≥ 5 rounds at the same time cluster).

For other metrics, we plotted as continuous line charts. The script calculates the mean value across all rounds for every second and represents them in the plots.

Once all the graph is plotted, we formed a table where the average (handover, handover time, delay, speed, distance, packet loss (%), and packets received) for all the rounds is presented. This highlights the performance of all the stations in different mobility according to the controller used in the experiment. The handover time in the table is a weighted Mean and is calculated using the formula:

$$T_{avg} = \frac{\sum(\text{Count}_i \times \text{Duration}_i)}{\sum \text{Count}_i}$$

Where,

- Duration_i: How long a single disconnection lasted
- Count_i: How many times that specific disconnection happened across all your test runs
- $\sum \text{Count}_i$: Total Number of Handovers

Mobility Statistics & Network Analysis

To conduct the study of mobility statistics, a statistical aggregation is employed to gather detailed temporal data into comprehensible performance measures. The files here used were the statistics file used for mobility statistics and DITG summary file used for network analysis. For each receiver node, raw telemetry data (distance, speed and acceleration) for mobility and (delay, jitter and bitrate) for network, sampled at discrete time intervals are then averaged over the entire duration of the simulation in order to describe the steady state operation of a station i; this average value $\bar{x}_i$ is the principle attribute of the node and is used to smooth out transient data to allow for comparative analysis between different mobility models. The formula for this is shown below.

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{i,j}$$

Where,

- $x_{i,j}$: The raw, instantaneous value of the metric (such as Distance to AP) captured for station i at time-step j
- n: The total number of temporal samples or rows extracted from the CSV file for that specific station
- $\bar{x}_i$: The resultant average value for station i

Box-and-whisker plots was used in this framework to evaluate the dispersion and the consistency of network performance across all nodes. The ”central box” represents the interquartile range ($IQR = Q_3 - Q_1$). This is the area where the middle 50 percent of station averages exist. The median (Q_2) is represented by a horizontal line in the box and provides a measure of the central tendency that is less affected by outliers than a mean would be. Additionally, the whiskers represent the farthest data points from the median that fall within the limits of $Q_{1/3} \pm 1.5 \times IQR$. Nodes that have performance metrics outside of these limits are considered statistical outliers and can thus be identified for extreme mobility cases or for connectivity failures in a particular SDN controller environment.

Correlation Analysis

This analysis is performed mainly for Experiments 1 and 2, where the same experiment is repeated at increasing speeds. The correlation was determined by measuring the statistical relationship between the Scenario Speed (independent variable) and Network Performance Metrics (dependent variable, such as Packet Loss). It uses the Pearson Correlation Coefficient (r) to determine whether increasing movement speed leads to a predictable increase or decrease in network failures.

The script uses the Pearson formula to compute the correlation.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

Here,

- x: Represents the Mean Speed of the stations for both scenarios (0 or 1).
- y: Represents the mean packet_loss (or other metrics) recorded for that round.
- r: The resulting value between -1 and +1.

The script then uses the resulting r value to drive the colors in a Heatmap. The following interpretations are derived from the heatmap.

- Positive Correlation ($r > 0$): Represented in Red. This indicates that as speed increases, the metric also increases.
- Negative Correlation ($r < 0$): Represented in Blue. This suggests that the network performed better at higher speeds (or was unaffected by the change).
- Strength: Values closer to 1.0 or -1.0 show a very strong, predictable relationship, while values near 0 show that speed had almost no consistent impact on network metrics for that specific controller.

3.2.1.3 Visualization

We created multiple visualizations of our data for different analyses as well. Packet loss is represented by bar charts with 95 % CI. Performance metrics station data are presented graphically in time series formats, summarized in tables; Heat Maps have been used to show correlation of related variables, Box Whisker Plots have been used to represent Mobile Statistics along with outliers.

3.2.2 Conclusion

We compare each outcome after analyzing every experiment and creating a graph of it. Since all of the results were controller-based and concentrated on all of the mobility used, we can conclude our research by determining which scenario the controller performed best in, as well as whether the proactive approach has enhanced the reactive approach.

4

Evaluation

4.1 Experimental Setup

The study involved five different experimental setups: small (4 APS, 4 stations), mid (16 APS, 32 stations), and large (16 APS, 64 stations).

Scale	ID	Configuration	AP Spacing	Traffic Tool	γ	Setup	Rounds
Small	Exp A	4 APS, 4 stations	100 m	DITG	3.7	All Mobile (Default speed)	10
	Exp B	4 APS, 4 stations	100 m	DITG	3.7	All Mobile (Double speed)	10
Mid	Exp C	16 APS, 32 stations	65 m	DITG, iPerf	3.9	Receiver stationary, Sender Mobile	5
Large	Exp D	16 APS, 64 stations	65 m	DITG, iPerf	3.9	All Mobile	10
	Exp E	16 APS, 64 stations	65 m	DITG, iPerf	3.9	Receiver stationary, Sender Mobile	10

Table 4.1 Detailed Experimental Setup and Simulation Parameters

Each simulation ran for 300 seconds for mid-to-large-scale deployments (16 APS, 32, and 64 stations) and 140 seconds for small-scale scenarios (4 APS and 4 stations). Each experiment was set up with mesh topology using Channel 5 and APS were placed at a spacing of 100 m for small scale and 65 m for mid and large scale.

Every data collection station was equipped with a dual-link interface: a wired link for out-of-band signaling with the DITG and a wireless link for experimental data communication. Receiver processes could be stopped gracefully without affecting the wireless test data thanks to this arrangement. Each communicating pair was given a unique port to prevent interference. UDP packets having a payload of 512 bits were transmitted at a rate of 1 packet/s using DITG and iPerf. Mainly the stations which we want to collect data used DITG and the rest used iPerf, keeping the same properties.

To evaluate the effect of flow rule management, the experimental technique was split into two stages. Pre-installed flow entries (proactive method) were used in the second phase, whereas reactive flow installation was used in the first. Every mobility scenario was repeated for 5 to 10 rounds in each phase. The controller was rotated before data collection started again, once all mobility patterns in the first phase were finished. Mobility was simulated using two distinct models: a 'defined mobility' model (only for small scale), which involved deterministic linear movement between specific start and stop coordinates, and a 'replaying mobility' model. The latter was utilized to represent all other mobility patterns by executing coordinates from a trace file at a rate of one coordinate per second (speed set to 1). Environmental conditions were simulated using the log-distance propagation model ($\gamma = 3.7$ and 3.9), with signal interference managed by wmediumd (noise threshold of -91 dBm and fading coefficient of 3).

Mobility Models

We have tested SDN Controller on four different mobility models that help us to analyze how SDN Controller performs on all types of mobility. These models bridge the gap between simulation and reality by depicting urban traffic, open-space pedestrian flow, and high-momentum vehicular movement. By testing under conditions ranging from random search-and-rescue paths to rigid city layouts, we identified how varying degrees of movement predictability affect SDN management. Further details of which mobility represent which real-life enviornment is given in Table 4.2.

Mobility Model	**Real-Life Environment**	**Ideal Use Case / Application**
Manhattan Grid	Structured City Streets	Smart city traffic management
Random Waypoint	Open Public Spaces (Campuses / Parks)	Human pedestrian crowds or mobile user behavior in malls
Random Direction	Border or Area Patrol	Search and rescue missions or automated cleaning robots
Gauss-Markov	Highways / Open Air	High-speed vehicles and drones

Table 4.2 Mapping Mobility Models to Real-Life Environments

4.2 Experiments

To identify how increasing the number of nodes impacts packet loss, several tests were carried out. The next sections contain the configuration information and related outcomes for each of these test cases.

4.2.1 Experiment A

A small-scale scenario with four (APS) and four mobile stations was used for the first assessment. Starting at coordinates $[100, 100]$, the APS were positioned with consistent 100 m spacing. An effective transmission radius of roughly 110m was obtained by setting the log-distance path-loss exponent to 3.7 to replicate a realistic wireless environment. Traffic was sent for a duration of 140 s, with sta1 and sta4 serving as senders and sta2 and sta3 as receivers. All the analyses here were done based on each station, and the outcome is listed below.

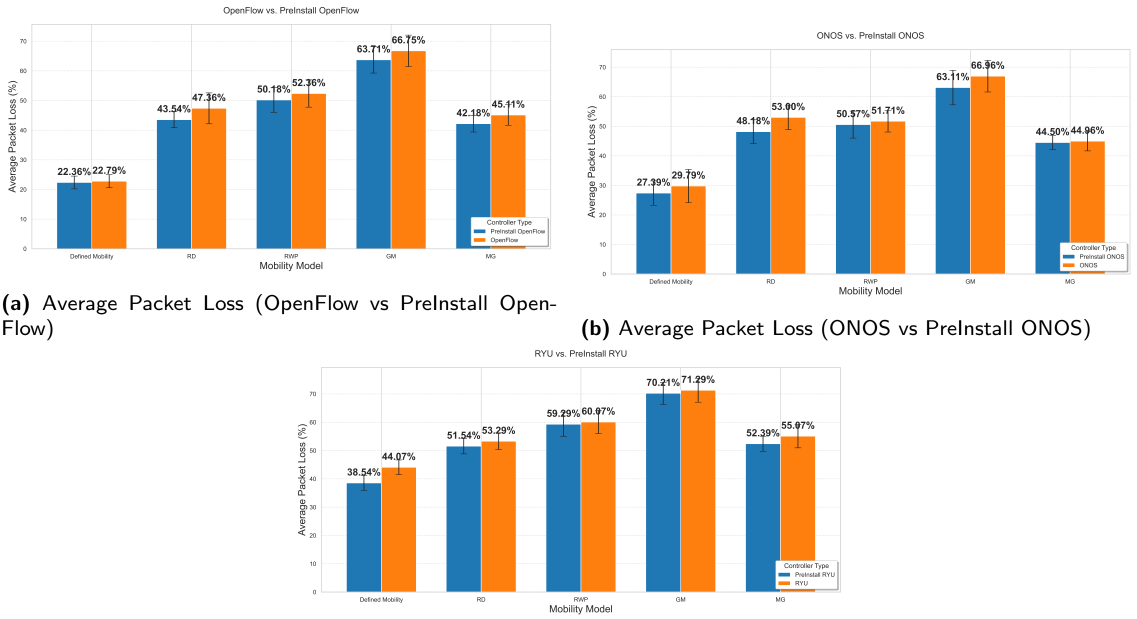
4.2.1.1 The Findings

Packet Loss

We have identified a clear, positive effect on the overall packet loss percentage due to proactively setting up the network. We have also seen that the proactive setup has reduced packet loss by 1-6% compared to the reactive setup for each controller. OF has shown the best average loss rate across our testing, especially in the Defined Mobility (DM) model, where it was possible to lower loss rates to 22.36% in proactive mode vs. 22.79% in reactive mode. In addition, even under extremely stressful conditions in the GM model, we have consistently observed that a proactive methodology can lower loss rates by about 1-3%. RYU lost about 1.08% less than reactive mode (from a high of 71.29%) in the GM model using the proactive methodology, while ONOS lowered the loss rate by the largest amount, from 66.96% in reactive mode down to 63.11% in proactive mode for this mobility.

In terms of challenge to the controllers, the RWP mobility model ranked second after the GM model, with average loss rates ranging from about 52% for both ONOS and OF to 60% for RYU. Although proactive methodology has provided some improvement in RWP, the total gain has been quite small, averaging around 1-1.5%. Thus, it appears that all three controllers are struggling significantly to adapt to the rapid, uncoordinated changes in speed and direction found in both the GM and RWP models. Notably, the largest improvement in performance from a proactive methodology was observed in the RD mobility model, which showed the largest increase among all tested mobility patterns.

In terms of packet loss, we found that for this small-scale scenario, OF performs better, followed by ONOS, and the worst-performing scenario is RYU. A further detailed comparison of these performance metrics is provided in Fig. 4.8.



(a) Average Packet Loss (OpenFlow vs PreInstall OpenFlow)

(b) Average Packet Loss (ONOS vs PreInstall ONOS)

(c) Average Packet Loss (RYU vs PreInstall RYU)

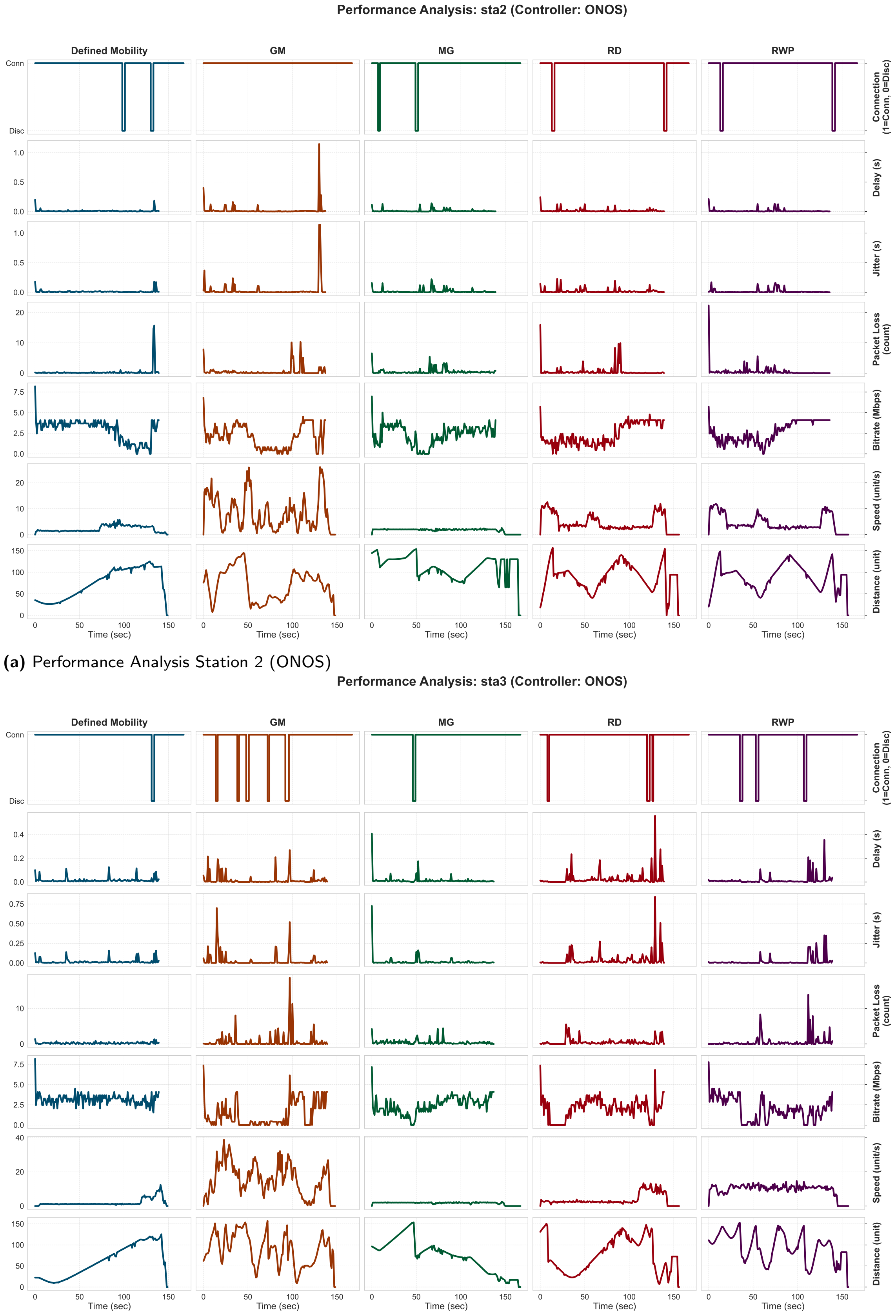
Figure 4.1 Comparative Analysis of Average Packet Loss

Performance Analysis

Here we present the performance of each station individually and state the observation as follows.

The network dynamics at Station 2 resemble a busy yet controllable path, where the main obstacles are rational and require quick decision-making. Metrics show high-density noise in jitter and delay combined with jagged bitrate in the reactive "stressed navigator" setup, especially in RD and GM models. This suggests the controller is consistently struggling to keep up with node movement. Nonetheless, the proactive configuration acts as a "smooth path". By pre-installing flows and eliminating processing overhead, the system clearly mitigates this noise, lowering the frequency and magnitude of jitter spikes while preserving a stronger connection status, thereby "smoothing" the path for data transfer.

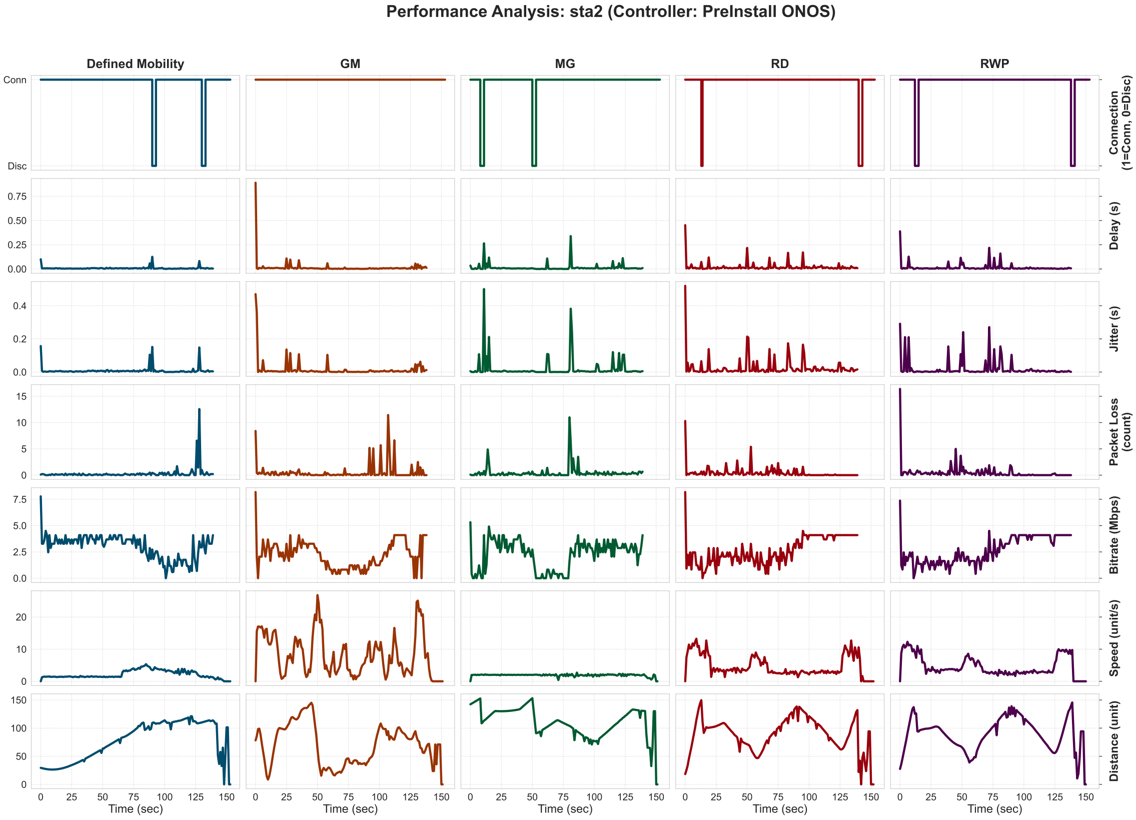
Station 3's network dynamics are an example of a high-velocity environment where the main obstacle is the station's quick physical displacement. As the controller in the reactive ONOS arrangement struggles to keep up with the numerous handovers required by the station's fast speed. This shows up as irregular bitrate swings and dense clusters of jitter and packet loss, especially in the GM and RWP models. On the other hand, the PreInstall ONOS setup anticipates the station's trajectory by proactively placing flows. The proactive approach greatly reduces surrounding noise, isolates jitter spikes, and normalizes the bitrate to guarantee a more robust data flow across a high-mobility landscape, even though physical distance still results in unavoidable connection loss. Figure 4.3 illustrates the performance analysis for both stations using ONOS and pre-install ONOS controller.



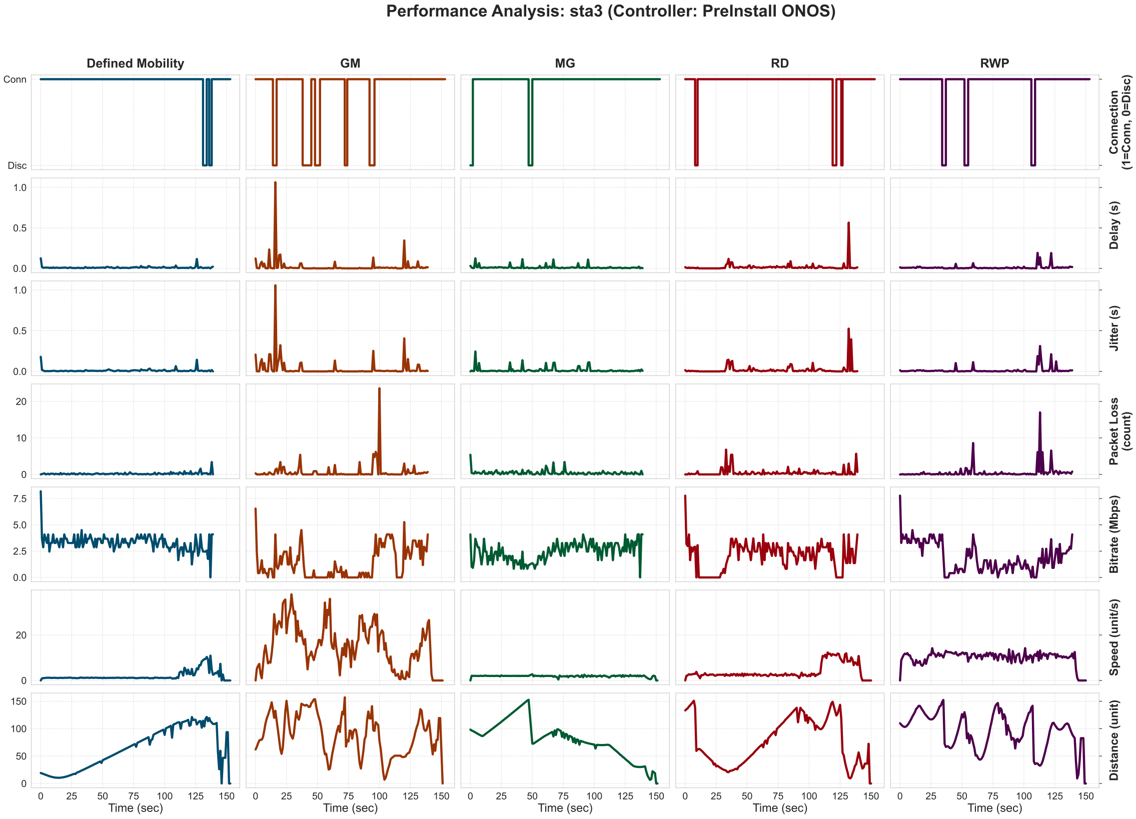
(a) Performance Analysis Station 2 (ONOS)

(b) Performance Analysis Station 3 (ONOS)

Figure 4.2 Performance Analysis (ONOS)



(a) Performance Analysis Station 2 (Pre-Install ONOS)

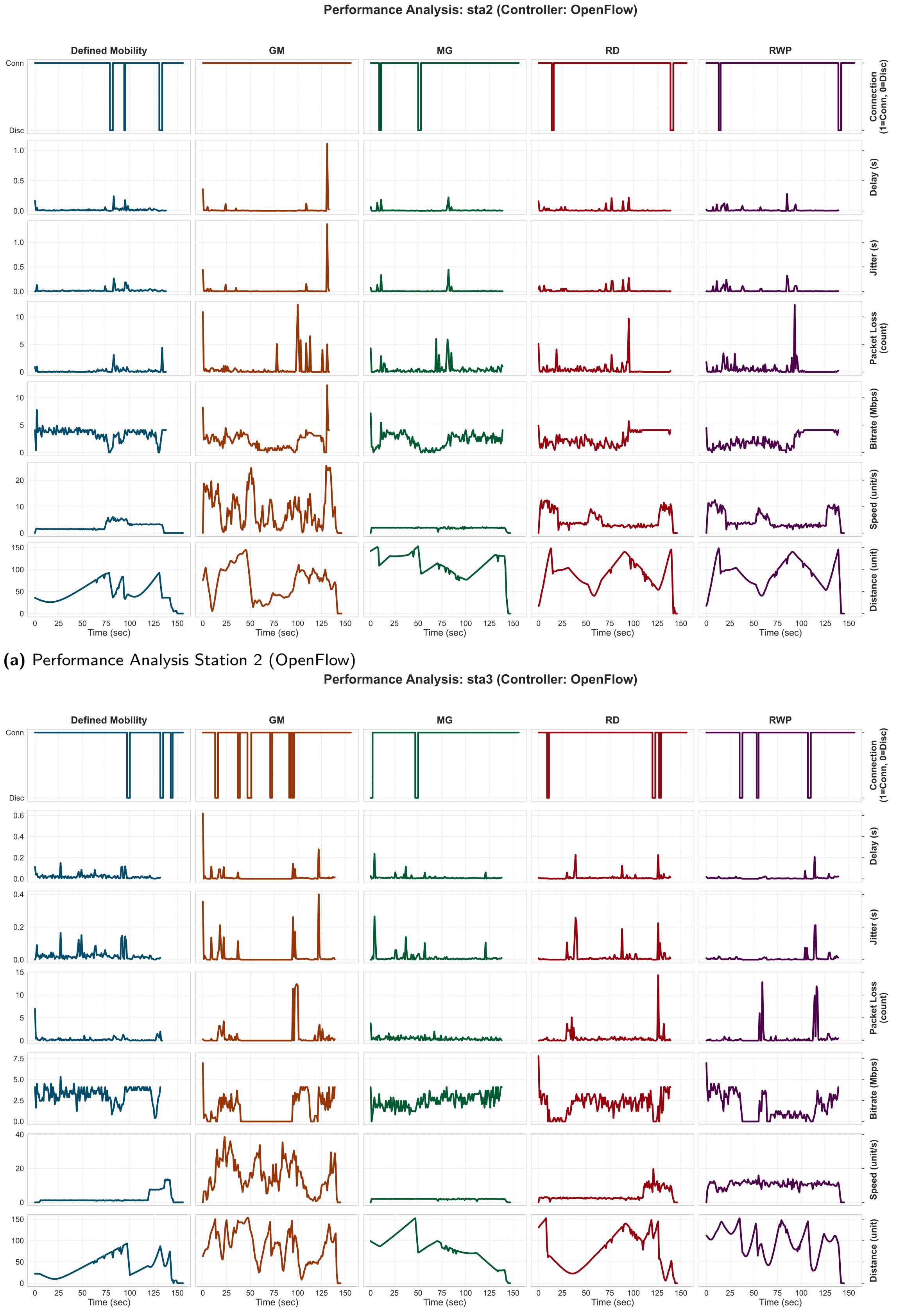


(b) Performance Analysis Station 3 (Pre-Install ONOS)

Figure 4.3 Performance Analysis (Pre-Install ONOS)

A shift from reactive instability to a more stable performance profile is shown in the comparison of Station 2's OF and PreInstall OF settings. The network exhibits an "event-driven lag" in the reactive OF configuration, where mobility-induced handovers, especially noticeable in the GM and RWP models, cause large vertical spikes in Delay and Jitter, peaking at roughly 1.0s and 0.28s. Due to the reactive controller's difficulty processing flow requests during node movement, these latency surges are directly linked to dense Packet Loss clusters that can reach up to 13 units and notable Bitrate (Mbps) decreases. On the other hand, the PreInstall OF setup acts as a "stabilizing filter". Although physical disconnections still happen, as the connection row illustrates, the proactive installation of flow rules decreases jitter spikes to less than 0.25 seconds and keeps the bitrate steadier.

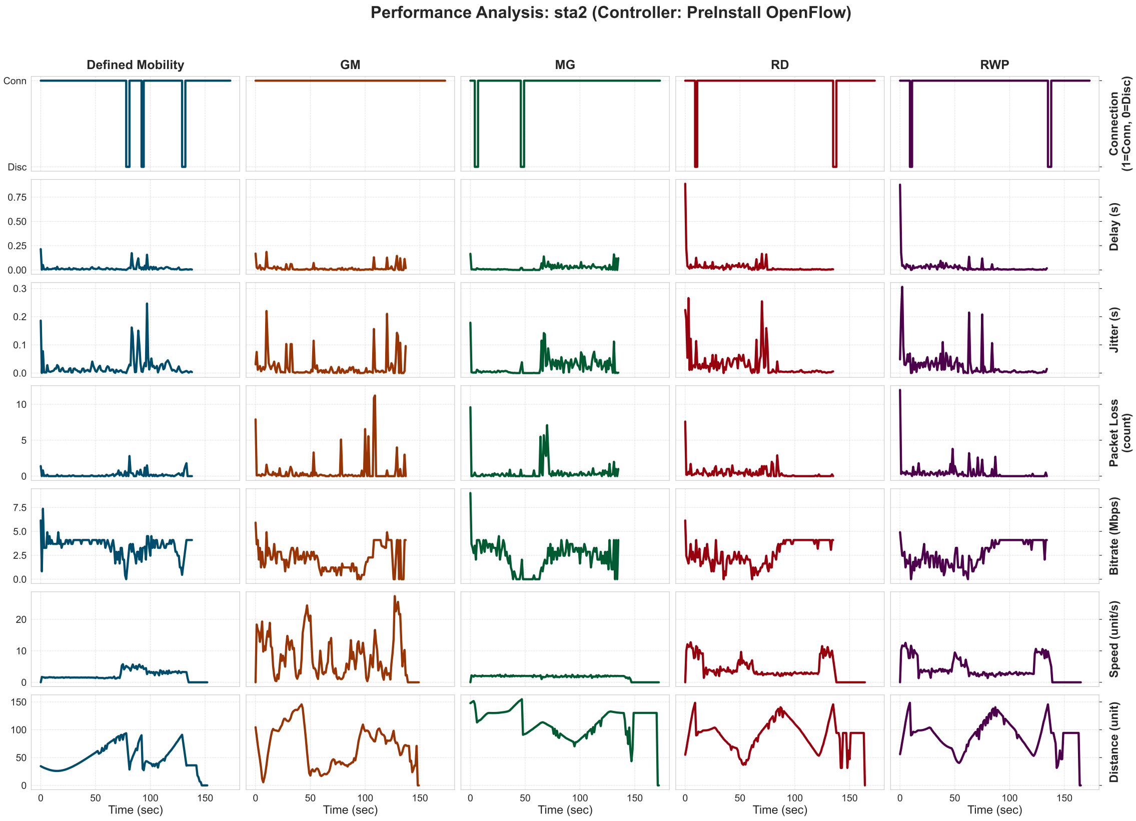
For Station 3, the performance profile mirrors the trends observed in the previous station, as the reactive OF configuration suffers from significant "event-driven lag" during high-speed transitions. Specifically, under the GM mobility model, Delay and Jitter reach peak values of approximately 0.6s and 0.4s, respectively. These latency surges are directly associated with dense Packet Loss clusters that frequently exceed 20 units, alongside a fragmented Bitrate (Mbps) profile that fails to maintain a stable baseline. Consistent with the results for Station 2, the PreInstall OF method acts as a stabilizing force, effectively dampening these spikes and smoothing the overall throughput by eliminating the signaling overhead of reactive flow requests. Figure 4.3 illustrates the performance analysis for both stations using OF and a pre-install OF controller.



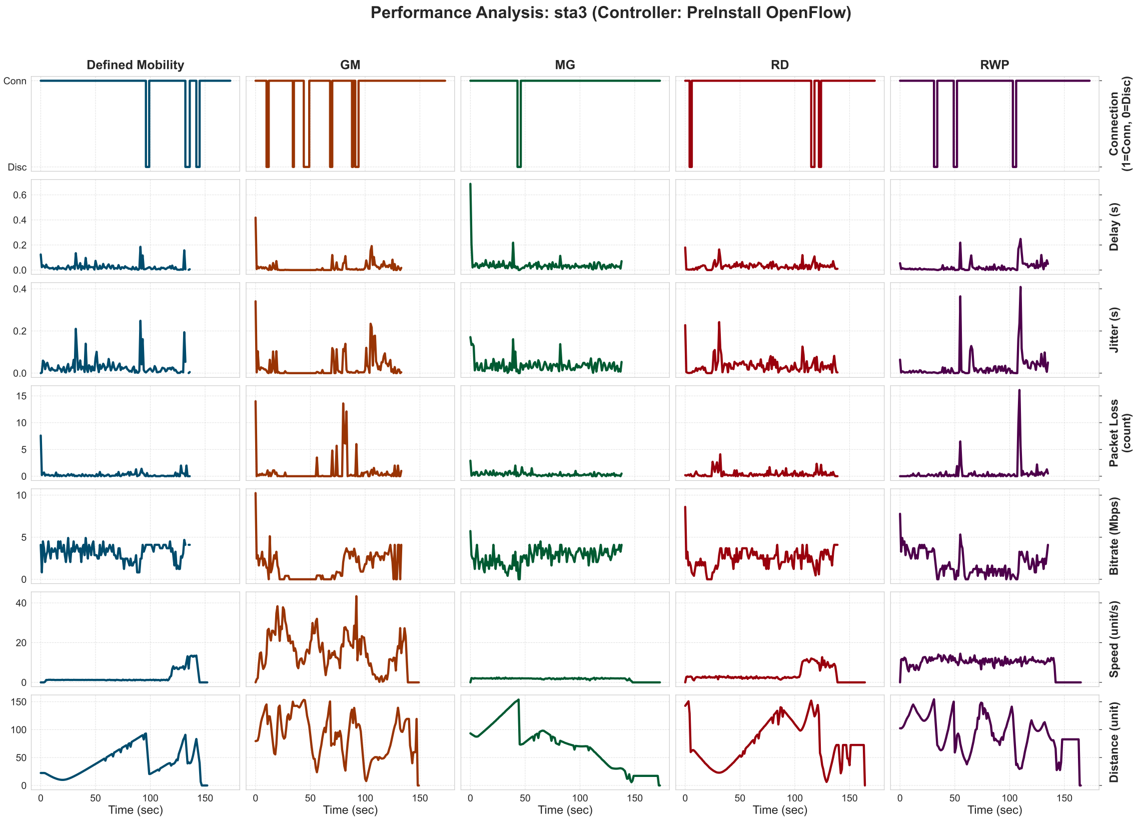
(a) Performance Analysis Station 2 (OpenFlow)

(b) Performance Analysis Station 3 (OpenFlow)

Figure 4.4 Performance Analysis (OpenFlow)



(a) Performance Analysis Station 2 (Pre-Install OpenFlow)

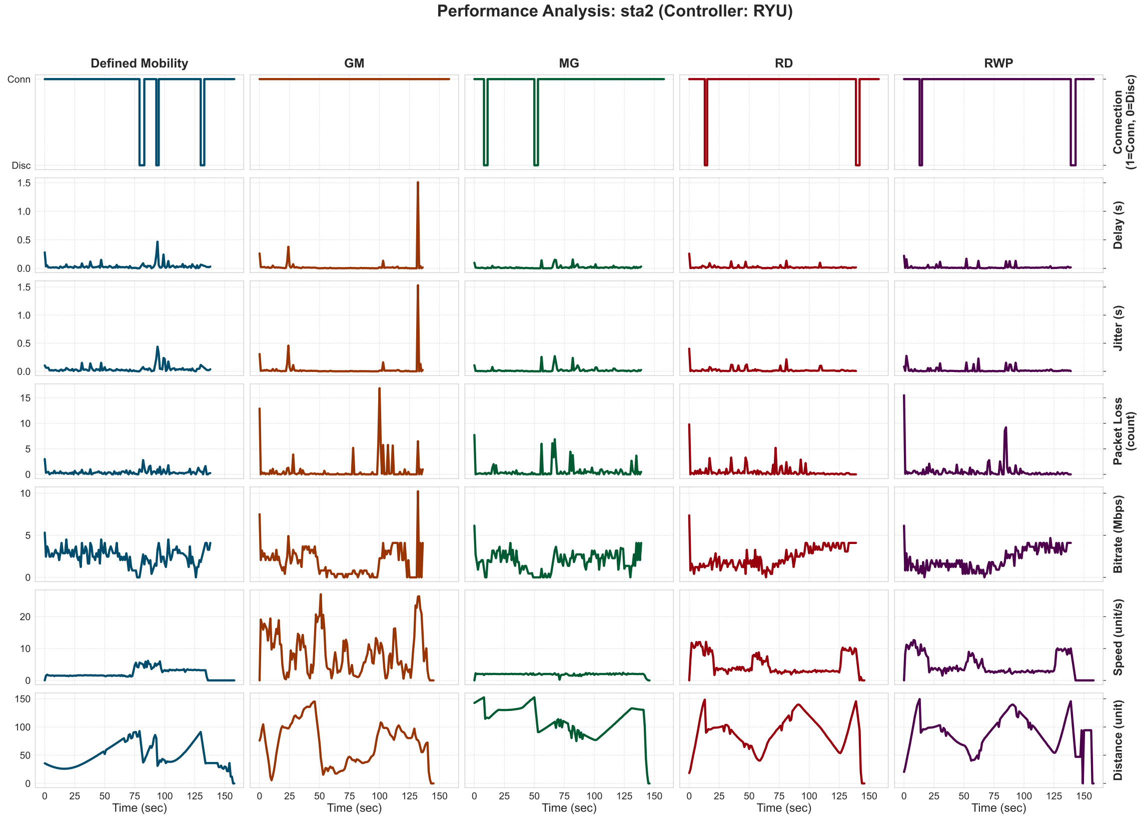


(b) Performance Analysis Station 3 (Pre-Install OpenFlow)

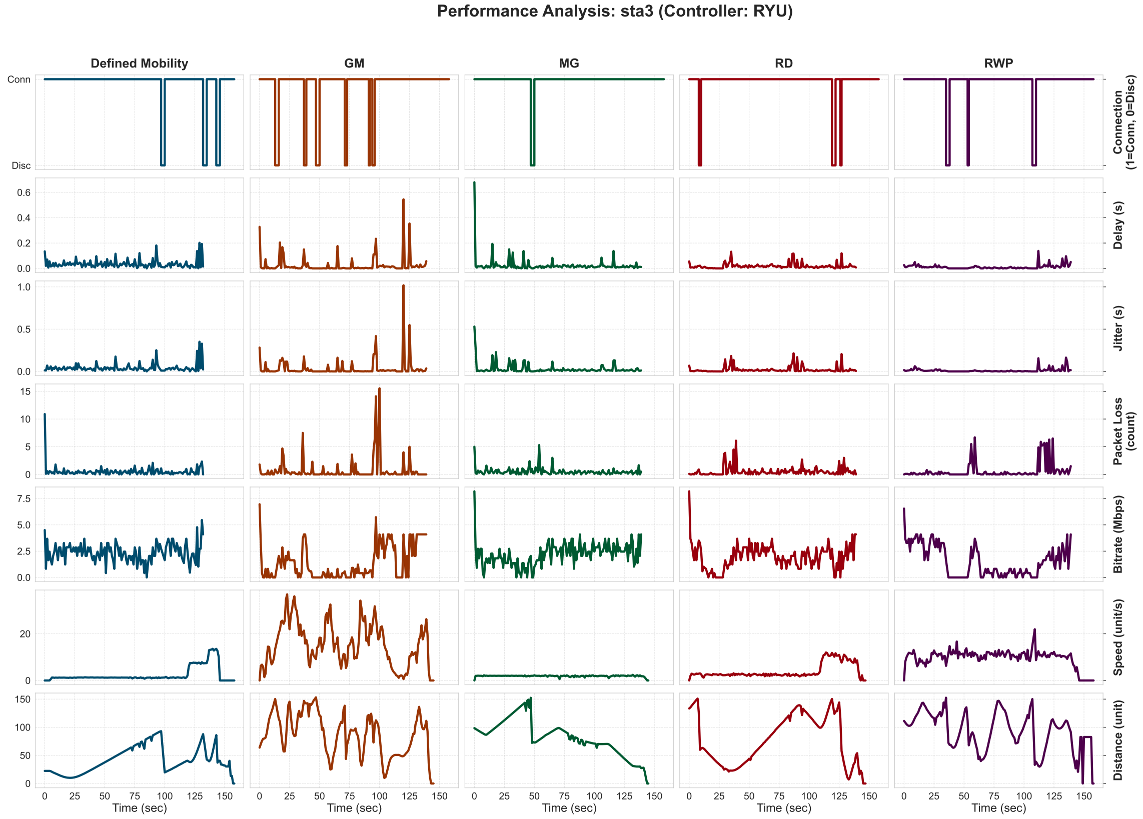
Figure 4.5 Performance Analysis (Pre-Install OpenFlow)

The performance analysis for Station 2 illustrates a transition from reactive signaling bottlenecks to proactive stability when moving from RYU to Preinstall RYU. This pattern is particularly evident in the Defined Mobility and GM models. Specifically, under these models, Delay and Jitter reach extreme peaks of 0.5s and 1.5s, respectively. These surges are directly associated with Packet Loss clusters of up to 18 units and deep, jagged drops in the Bitrate (Mbps) profile, reflecting the controller's struggle to manage flow rules in real-time during node movement. Using the proactivae method, these have been stabilized with Delay and Jitter reaching a value of 0.3s, and also maintain a more resilient Bitrate profile.

The performance analysis for Station 3 reveals a familiar trend of proactive stabilization, though with specific anomalies in certain mobility models. Under the reactive RYU configuration, the station suffers particularly in the GM model where Delay and Jitter reach extreme peaks of 0.6s and 1.0s, respectively. These surges are accompanied by massive Packet Loss bursts of up to 25 units. While the Preinstall RYU method successfully stabilizes these metrics for most scenarios, it encounters a specific struggle under the RD mobility model. In this instance, the proactive approach actually sees Delay and Jitter peak at 0.6s and 0.8s, whereas the reactive setup maintained lower values of approximately 0.2s. Furthermore, the proactive Packet Loss in the RD model increases to 10 units from the reactive baseline of 6 units, indicating a unique performance degradation for this specific mobility pattern when using the pre-installation method. Figure 4.3 illustrates the performance analysis for both stations using RYU and a pre-install RYU controller.

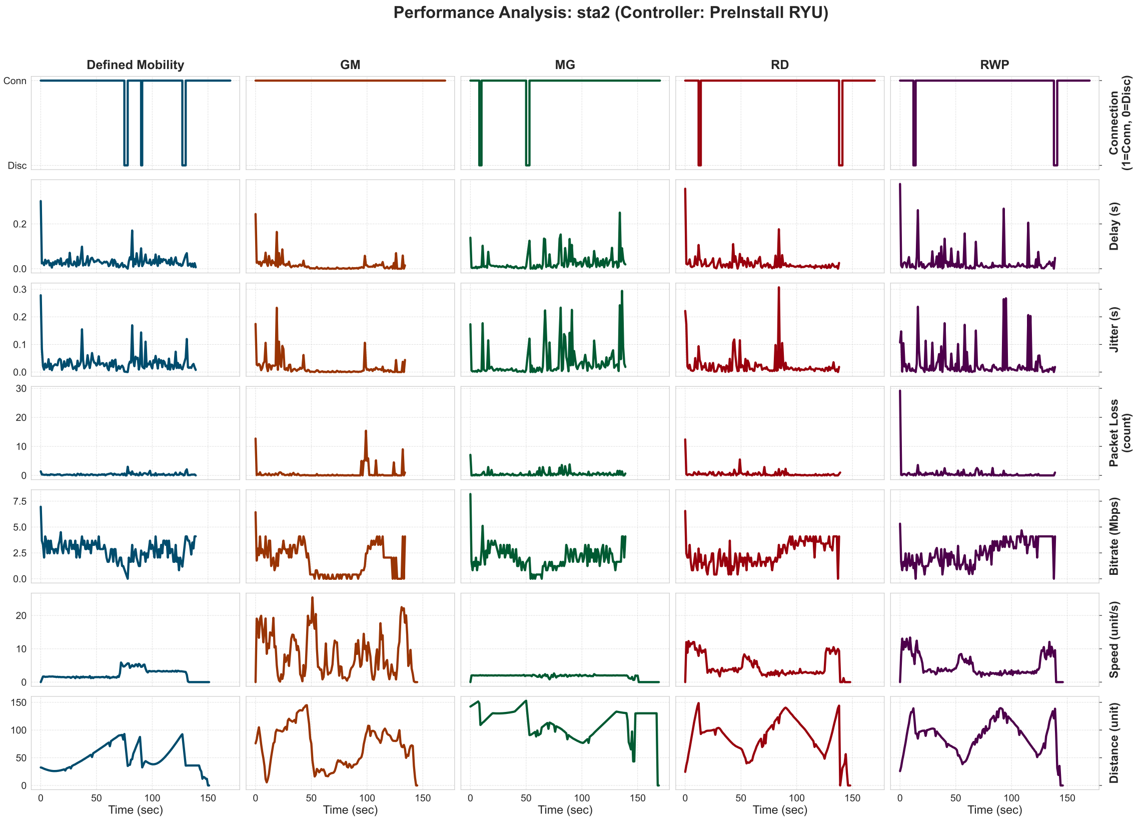


(a) Performance Analysis Station 2 (RYU)

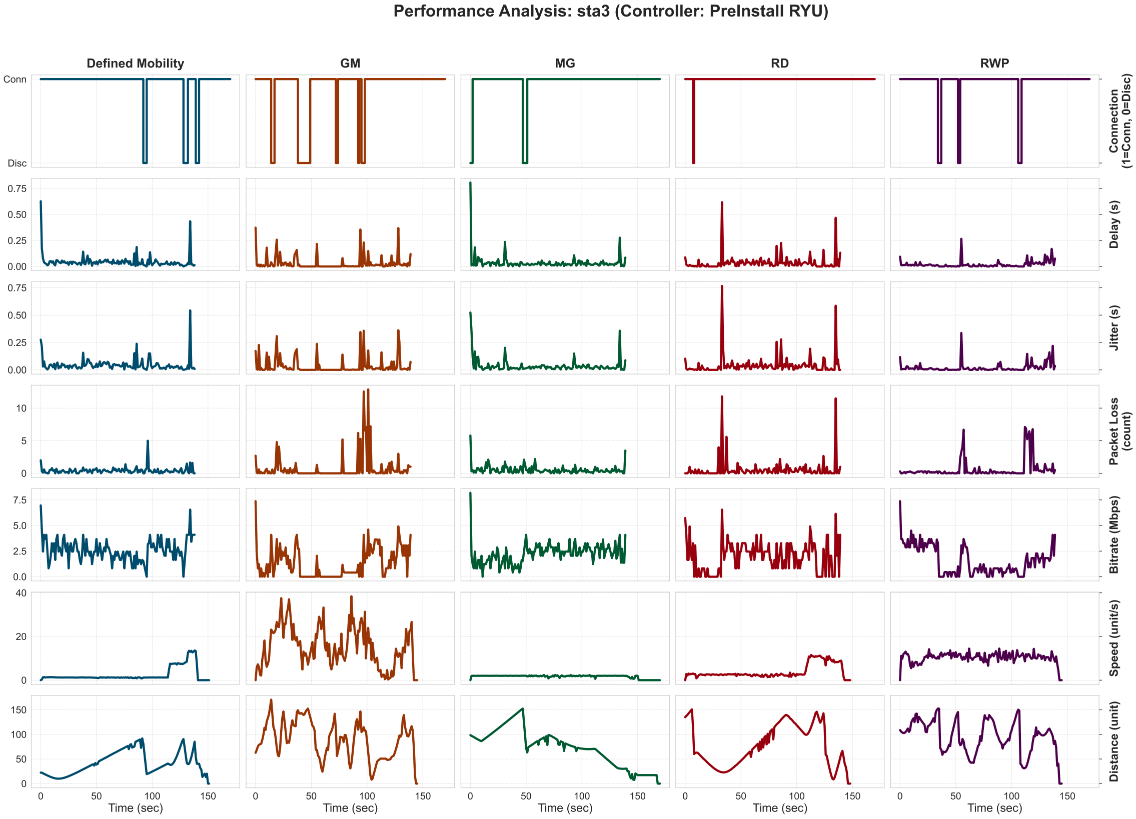


(b) Performance Analysis Station 3 (RYU)

Figure 4.6 Performance Analysis (RYU)



(a) Performance Analysis Station 2 (Pre-Install RYU)



(b) Performance Analysis Station 3 (Pre-Install RYU)

Figure 4.7 Performance Analysis (Pre-Install RYU)

Due to its high node speeds, the GM mobility model generated the largest packet loss, followed by RWP. Tables 4.3 contain detailed comparison statistics for each mobility model under ONOS controllers and for further controller see A.1, and A.2 in A.

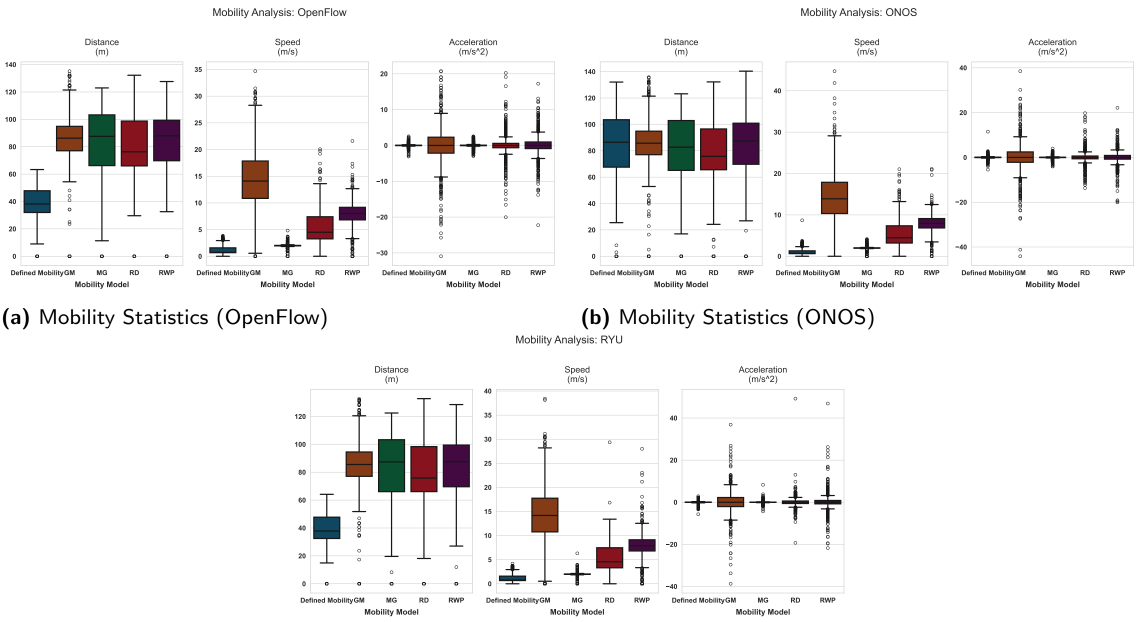
Mobility	St.	Ctrl.	Avg HO	HO Time (s)	Speed (m/s)	Accel. (m/s^2)	Dist. (m)	Loss (%)	Loss (Ct)	Total Pkt	Delay (s)
Def. Mob.	sta2	R	2	3.27	2.24	-0.003	74.52	34.71	49	91	0.011
		P	2	3.21	2.27	-0.008	77.55	33.36	47	93	0.0082
	sta3	R	1	2.44	2.06	0.064	60.30	24.86	35	105	0.015
		P	2	2.67	2.092	0.009	63.32	21.43	30	110	0.0109
GM	sta2	R	0	3.25	8.49	-0.10	71.03	57.36	80	60	0.016
		P	0	3.25	8.44	-0.12	70.97	54.86	77	63	0.013
	sta3	R	5	2.94	15.63	-0.009	89.70	76.57	107	33	0.0164
		P	5	4.87	15.67	0.0544	90.03	71.36	100	40	0.019
MG	sta2	R	2	2.75	1.90	-0.003	112.47	46.36	65	75	0.0133
		P	2	2.85	1.92	-0.008	112.46	45.43	64	76	0.0134
	sta3	R	1	2.83	1.87	-0.006	79.60	43.57	61	79	0.014
		P	2	2.50	1.91	-0.003	81.70	43.57	61	79	0.013
RD	sta2	R	2	2.85	4.85	-0.01	89.55	52.00	73	67	0.015
		P	2	1.88	4.87	0.007	91.70	42.79	60	80	0.022
	sta3	R	3	2.38	3.99	0.038	77.10	54.00	76	64	0.022
		P	3	2.067	3.98	0.031	77.67	53.57	75	65	0.015
RWP	sta2	R	2	2.88	4.85	-0.005	90.03	49.07	69	71	0.013
		P	2	3.10	4.87	-0.023	90.29	46.36	65	75	0.018
	sta3	R	3	3.00	10.27	-0.016	95.46	54.36	76	64	0.014
		P	3	2.90	10.37	0.008	95.28	54.79	77	63	0.011

Table 4.3 Performance Metrics for ONOS (R) and Pre-Install ONOS (P) Controllers across Mobility Models

Mobility Statistics

In our study of node motion, we discovered that a direct relationship existed between node mobility characteristics (i.e., distance, speed, and acceleration) and the overall stability of the network. In each of the testing scenarios, the GM model provided the greatest difficulty for all controllers due to the high variability in speed and the presence of extreme outliers. This unpredictability of movement was also responsible for the highest loss rates in each scenario as a result of the frequent topological changes that occurred within the network. Conversely, the DM model had a much smaller and more uniform speed distribution than the GM model with less frequency of speed shift; consequently, the controllers were able to adhere to flow rules more successfully and maintain the lowest recorded loss rates in each scenario.

Among all the OF demonstrates the most resilience, with a stable distribution of speeds and distances across all simulations, which translates to superior packet retention. ONOS has an average to above-average performance, but with a greater sensitivity than others to speed spike events, particularly in simulation models such as GM and RD. The greatest variance was found in RYU, which showed the largest variation in both speed and acceleration processing. RYU lacks statistical consistency and thus may be overwhelmed by rapid node movement and therefore demonstrate the poorest performance among all the controller platforms in our study.



(a) Mobility Statistics (OpenFlow)

(b) Mobility Statistics (ONOS)

(c) Mobility Statistics (RYU)

Figure 4.8 Mobility Statistics of All Controllers

4.2.2 Experiment B

The second experiment was conducted using an identical physical setup and motion parameters to those of experiment 1. The travel time between stations was doubled by increasing the station speed feature to 2 units/second. To provide consistency for this increased speed, the running time of DITG was decreased to 70 seconds (from 140), and the waiting time was shortened from 15 to 7 seconds. The goal of this second test was to examine how increased speed influences packet loss while maintaining other test conditions constant.

4.2.2.1 The Findings

Here, we focused on comparing the stations from experiments 1 and 2 rather than assessing their performance. Speed 0 denotes data obtained at a standard speed, i.e., experiment 1, while speed 1 corresponds to this experiment.

Packet Loss

Our results indicate that increasing speed significantly increases packet loss when applied to the OF controller. The GM mobility model showed the most significant degradation, with packet loss rising to roughly 74% from 69% in the previous experiment. Additionally, compared to earlier findings, the pre-installation of flow entries showed a more pronounced performance difference, especially in the GM, and MG mobility models. Therefore, we can conclude that pre-installation has a stronger influence on improved speed, with the difference greater than in the baseline experiment, where the improvement was significant even with pre-installation.

We observed nearly opposite results when comparing the RYU controller, as in most cases increasing speed lead to less packet loss compared to the OF controller. The most significant performance degradation occurred in GM mobility, where performance declined by over 7% from a peak packet loss of 78%. In this case, pre-installing the flow entries has more influence than what we obtained from the pre-install OF controller, but the peak in GM mobility is a bit higher (71.8%) than what we obtained from the OF controller (71.0%). Even with all the performance improvement the RYU controller still achieved the highest packet loss in all scenario compared to the OF controller. Apart from that, the effects of enhanced speed are nearly identical to those of normal speed.

When analyzing for the ONOS controller as the station moves at an increasing speed, we see that it has much greater efficiency. Packet loss for the reactive experiment dropped from 29.8% to 23.4% (DM), and 53% to 48.9% (RD). Overall, the proactive experiments performed better than the corresponding baseline experiments, indicating improved resiliency. Among all models, the GM mobility model had the highest packet loss. However, its packet loss rate was always lower than that of the other controllers. The proactive MG mobility model showed the largest drop in packet loss of 16.5% with respect to the baseline experiment and achieved the best performance results for all of the controllers evaluated.

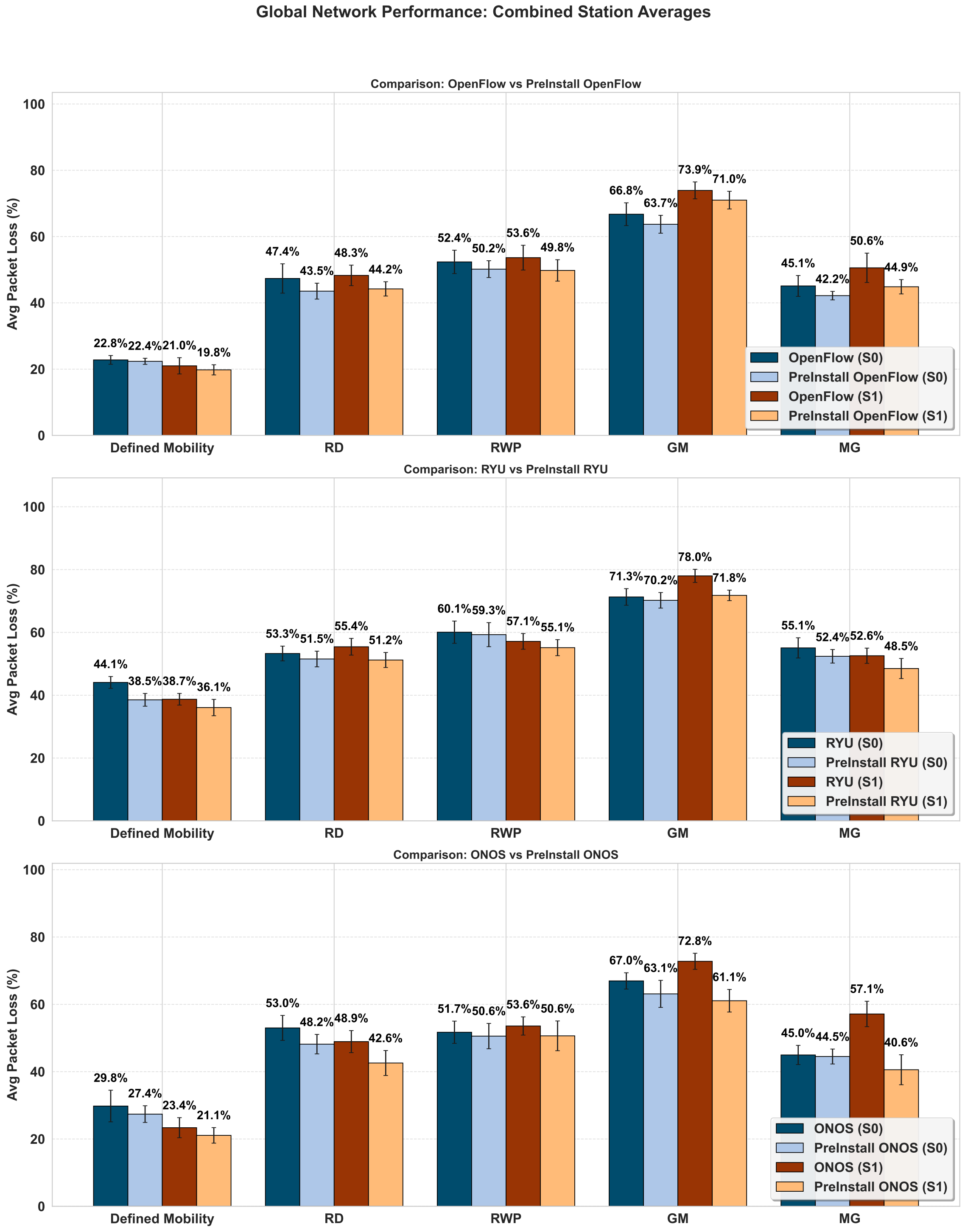


Figure 4.9 Comparison between speed 0 and speed 1 for all controllers.

Increasing node speed and using the RYU controller improve performance, with almost all mobility achieving lower packet loss. We also observed better performance in DM and RD mobility using ONOS. In terms of controller OF and ONOS achieved almost similar results, whereas RYU struggles as it shows greater packet loss compared to other controllers. Since GM showed the highest packet loss across all scenarios, we did not observe any improvement in performance at higher speeds or with a different controller, unlike other mobility.

Correlation

To further analyze the impact of speed, a correlation heat map was done on the reactive controllers. The heat map results are as follows:

- ONOS Controller: ONOS has significant sensitivity to speed at its base level, but that sensitivity greatly diminishes as speeds increase. In the DM case, ONOS has a nearly perfect negative relationship with bitrate (-0.96) and a nearly perfect positive relationship with loss (0.95), indicating that there is a very predictable and large decrease in performance with each incremental increase in speed, within low speed range. At twice the speed, those relationships collapse (bitrate is now at -0.10), indicating that ONOS hits some kind of saturation point, beyond which further increases in speed do little to degrade the already poor performance metrics.

  The MG case is an exception; at normal speed, bitrate has a positive correlation (0.55) of the same order as the DM case, possibly indicating ONOS can deal with structured grid movement better than random movements.

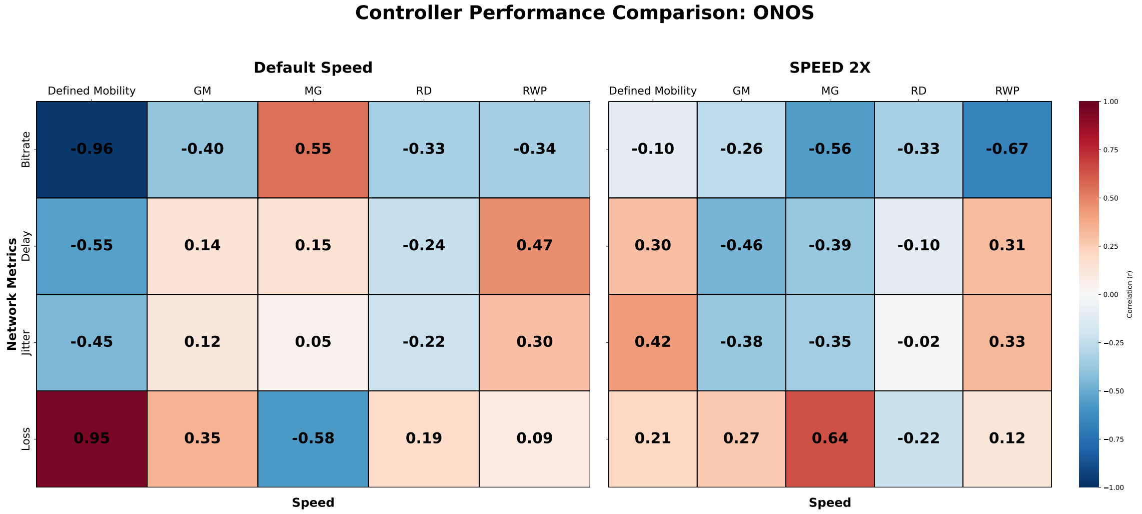


Figure 4.10 Correlation Analysis: Speed 0 vs. Speed 1 for ONOS Controller

- RYU Controller: RYU was the most unstable of the three controllers, with many of its relationships being reversed as the velocity doubled. The default speed showed a moderate to strong negative correlation for bitrate (-0.5), delay (-0.69) for DM. The relationship flipped dramatically; the bitrate went from a -0.5 to a +0.32 correlation. A flip in the relationship signifies that the controller has lost its ability to handle traffic predictably, as the overhead associated with high velocity may have triggered nonlinear delay processing.

  RYU's packet loss metrics were also very volatile. Although they remained relatively stable in the RWP model, the correlation between loss and speed flipped in the MG model from 0.12 to -0.44. Such variability suggests that RYU may struggle to maintain consistent flow table synchronization, as node handovers occur at high frequencies.

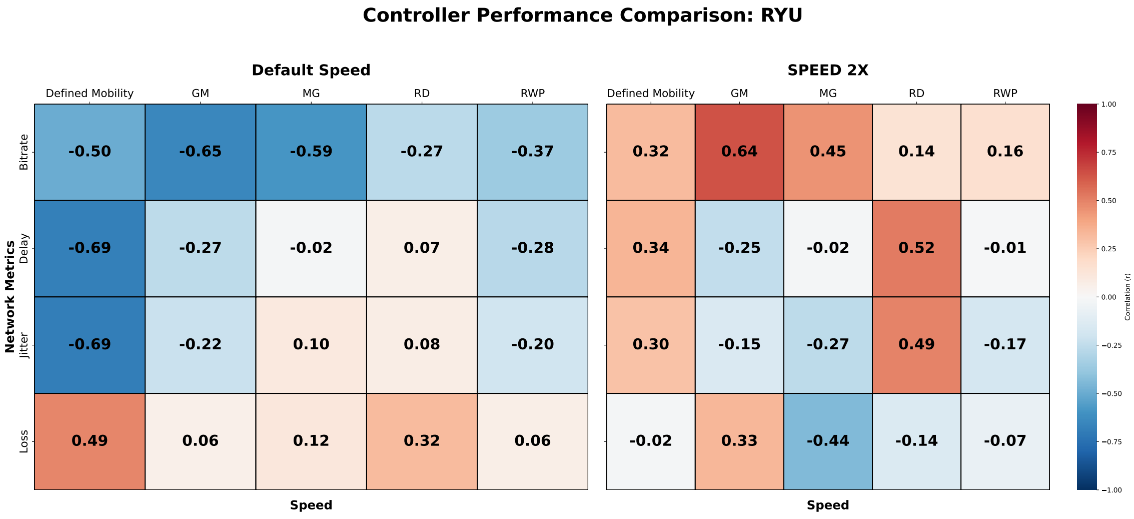


Figure 4.11 Correlation Analysis: Speed 0 vs. Speed 1 for RYU Controller

- OF Controller: The OF implementation exhibited a linear degradation pattern, where sensitivity to speed typically increased with increasing speed. Unlike ONOS, OF becomes notably more sensitive at higher speeds. In DM, the correlation for packet loss moves from -0.34 to -0.64 at double the speed. This suggests that the controller does not reach a saturated point but rather continues to degrade as the dynamism of the network environment increases.

  OF is uniquely sensitive to the pattern of movement as it struggles significantly in structured environments like the MG at default speed (loss correlation of 0.66), but shows better resilience in RWP model until speeds are doubled, where loss correlation then jumps from -0.19 to 0.58.

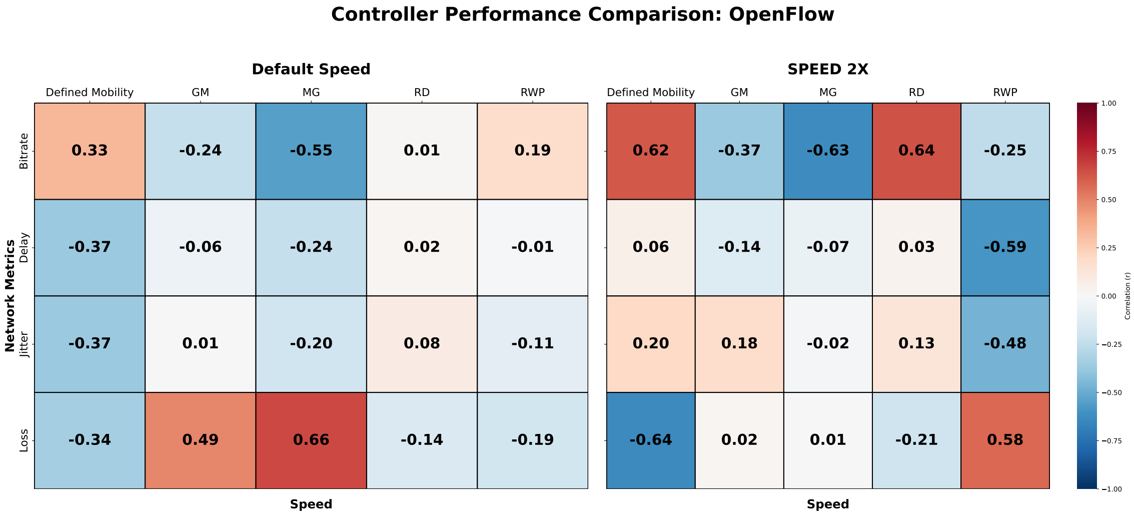


Figure 4.12 Correlation Analysis: Speed 0 vs. Speed 1 for OpenFlow Controller

These results indicate that ONOS would be the best choice for high-speed environments as it shows a performance saturation point where increasing the speed does not further degrade the performance, with the exception in MG mobility. OF would provide the most predictable degradation as speed increases, making it the best option for networks with moderate, consistent speed increases. Finally, RYU would be the least desirable option for high mobility networks, as the controller's response to speed increases is nonlinear and thus difficult to predict.

4.2.3 Experiment C

The design used in this study utilized 16 APS, and 32 stations. This design had an even number of stations acting as stationary receivers close to the APS, and an odd number of stations acting as mobile senders using the mobility models. We transmitted data for 300 seconds at one packet per second at 512 bits through all stations, but for evaluation, we only considered 16 stations. Each mobility pattern was repeated for 5 rounds in the study.

4.2.3.1 The Findings

Like all other experiments, the average packet loss across all stations was done using 95% CI. The study further investigated how stations spread across each APS. Finally, we provide the performance metrics of all the stations combined and group them by senders and receivers.

Packet Loss

We showed a consistent reduction in packet loss across all the mobile scenarios using the proactive method in OF controller. With a lowest packet loss achieved in GM mobility (55.2%) and highest in MG mobility (65.9%) using the reactive method. This has been reduced to 42.3% and 55.9%, respectively, using the proactive method. Across all the mobility scenarios using the proactive method, we got an average reduction ranging from 7.6% (RD) to a maximum of 12.9% (GM).

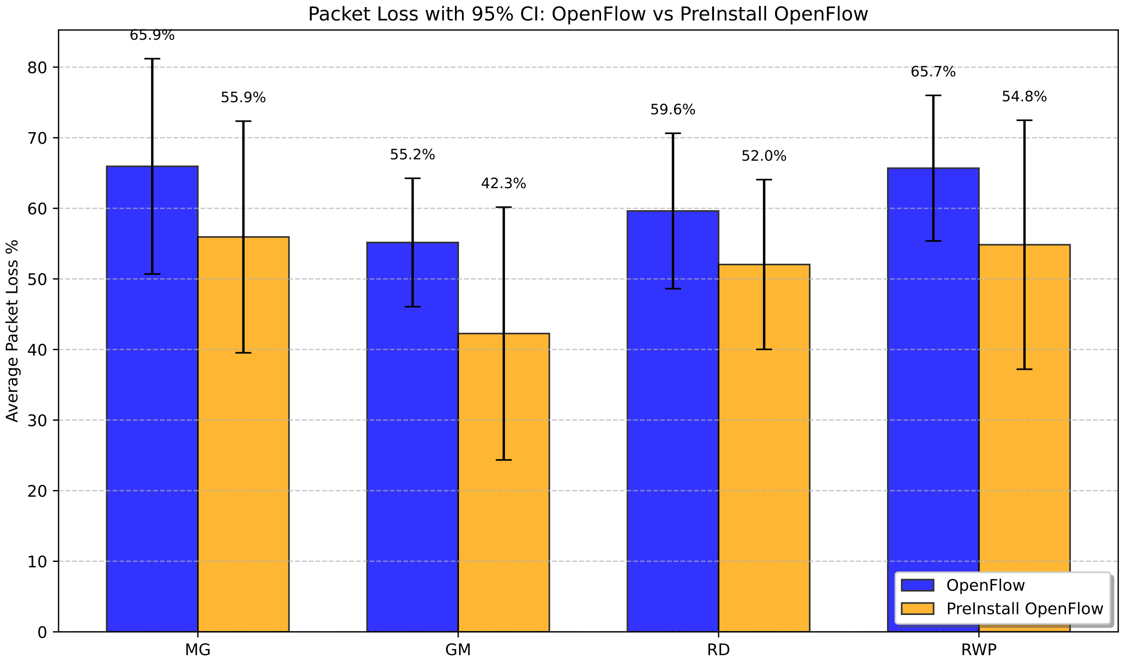


Figure 4.13 Overall packet loss using OpenFlow and Pre-Install OpenFlow controller

According to the data using ONOS, in every mobile context, the proactive approach performed better than the reactive setup. The MG scenario reported the highest packet loss of 65.8% using the reactive configuration, while the RD scenario recorded the lowest packet loss of 56.7%. When the proactive approach was used, these numbers decreased to 64.5% and 43.6%. Overall, the proactive approach reduced packet loss by about 9% with the lowest being in MG (1.3%). But in the GM and RD situations, when the uncertainty intervals are still rather wide, the CI indicates a high degree of variation.

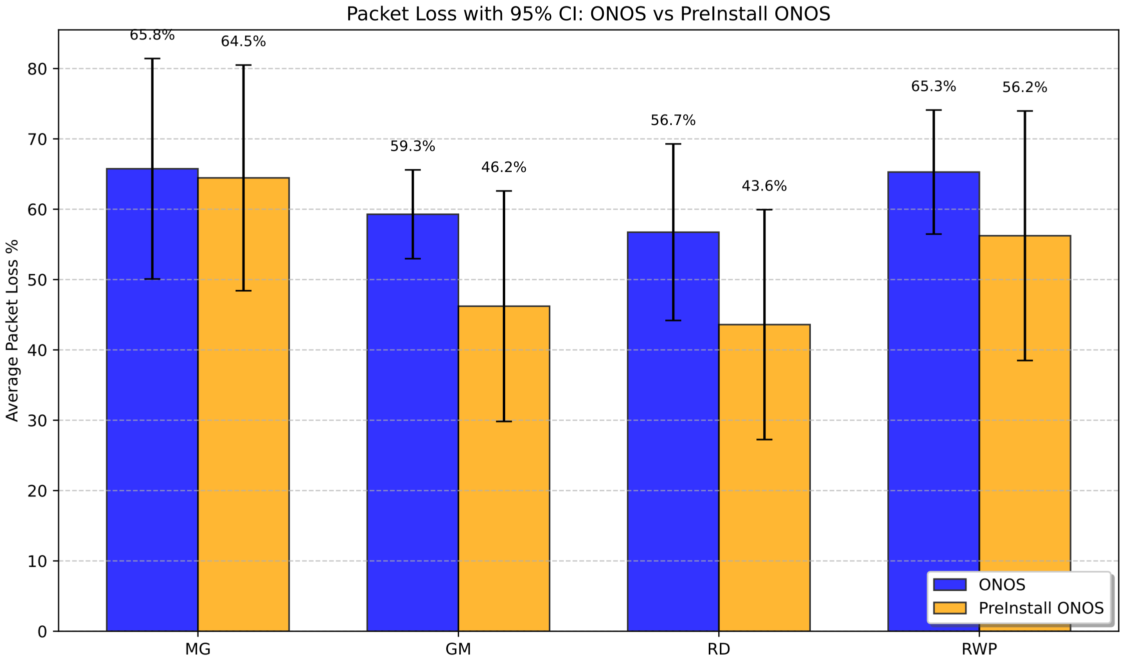


Figure 4.14 Overall packet loss using ONOS and Pre-Install ONOS controller

Regarding the RYU controller, the data indicates that it experiences the highest packet loss among all tested controllers across every scenario, specifically within RD mobility. In the RD scenario, it hits a peak average packet loss of 68.6% using the reactive approach, which is then lowered by nearly 20% through the proactive method. RYU achieves its highest reactive performance with an average loss of 72.2% in MG mobility. Within this scenario, the transition to the proactive method yields an improvement of 10.3%, representing a significant gain compared to the improvements seen in other controllers for the same scenario.

Among all three controllers, ONOS and OF performed almost at the same level in terms of overall network stability, while RYU showed the most significant improvement when switching to a proactive method. In terms of mobility, all three controllers struggled most while maintaining network stability in more structured mobility, i.e., MG model, followed by the more erratic pattern of RWP.

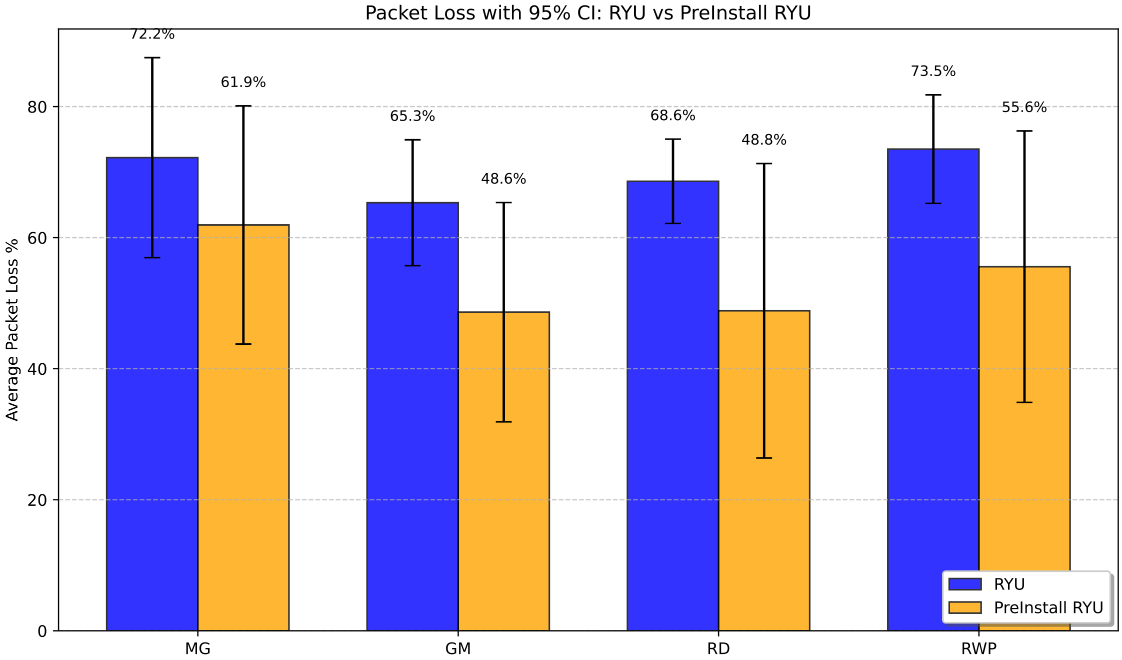


Figure 4.15 Overall packet loss using RYU and Pre-Install RYU controller

Network Analysis

- Delay: The graph shows that ONOS consistently stands out as the most reliable performer. In almost every mobility scenario, especially RD and RWP. It maintains a lower median delay compared to its peers. It also shows how the GM model pushes all three controllers to their limits. As we can see, the median delays climb, and the whiskers stretch out, indicating a significant increase in packet delivery volatility. OF generally struggles the most here, often showing the highest latency peaks. This suggests that as nodes move in more complex patterns, the OF architecture takes longer to re-establish stable paths than ONOS framework.

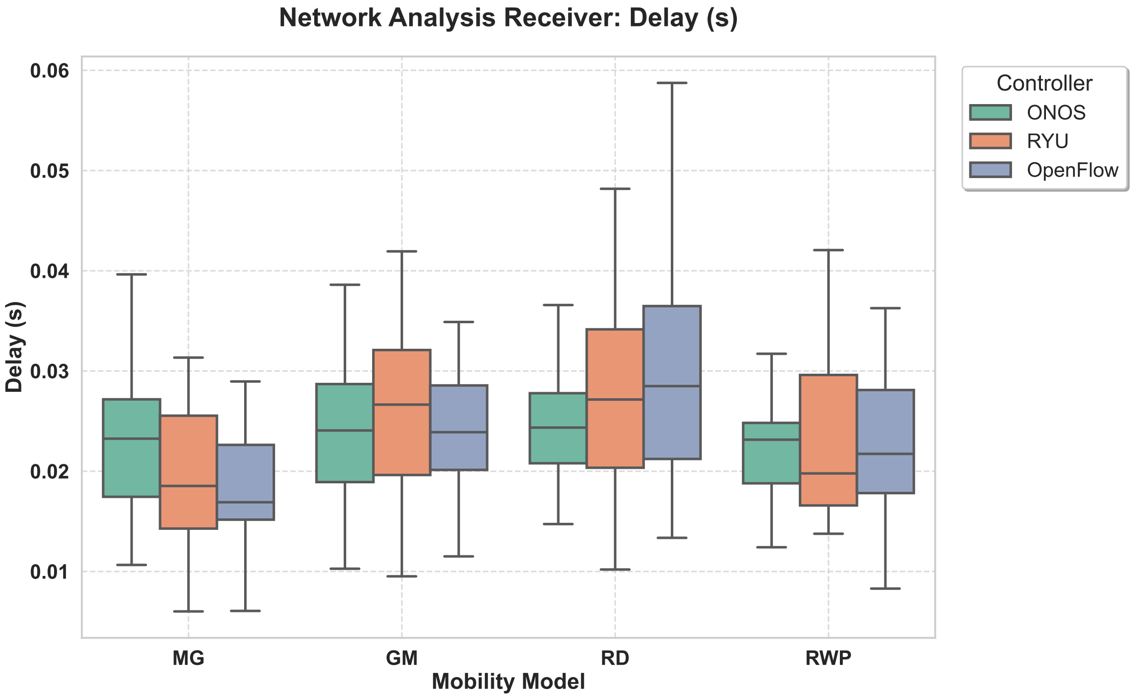


Figure 4.16 Analysis of Delay for All Controllers

- Jitter: The jitter trends tell a similar story about stability. ONOS shows a tighter box plot, which means the variation between packet arrival times is kept to a minimum. On the other hand, the GM model once again causes issues, as it causes OF to experience massive jitter spikes exceeding 0.05s. While RYU and OF perform somewhat similarly in the RWP model, they both show much wider distributions than ONOS, indicating that they aren't quite as efficient at handling the rapid topology shifts that happen when nodes suddenly change direction.

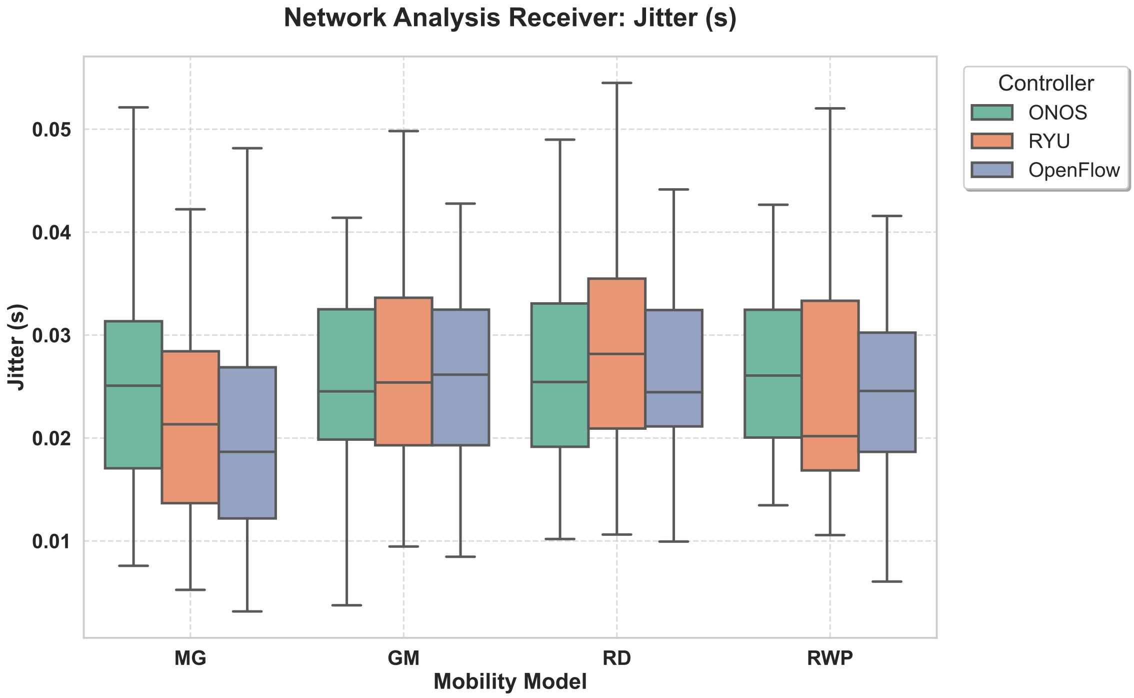


Figure 4.17 Analysis of Jitter for All Controllers

- Bitrate: The MG appears to be the best option when it comes to raw throughput, likely because the structured movement allows for more stable connections. In this case, RYU continuously lags behind with a lower median, while ONOS and OF are closely matched, both pushing bitrates toward the higher end of the range. In terms of RD, we can see a visible drop in bitrate across the board. This is likely because nodes are more prone to hitting the edges of the simulation area and losing signal. Even in these tough conditions, ONOS manages to maintain a slightly better edge, though it does show more variance.

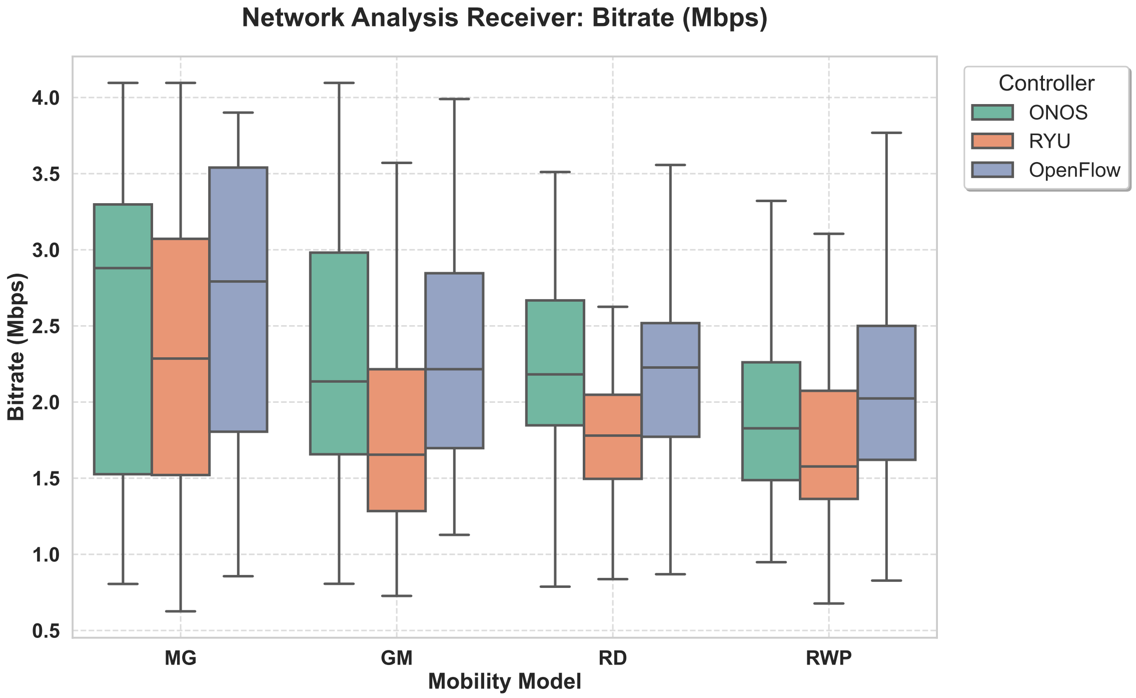


Figure 4.18 Analysis of Bitrate for All Controllers

Performance Metrics

Analysing the performance of each node we figured that MG and GM have the lowest number of handovers (4) compared to RWP and RD (6). Even having similar handovers, the handover time across MG and RWP is comparatively higher than other mobility models, where RWP has the highest average handover time across all controllers. This higher handover time contributes to getting a higher packet loss in these scenarios, along with their mobility patterns.

Mobility	Ctrl.	Role	Avg HO	Avg HO Time (s)	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	0	0.000	25.934	0.000	0.000	59.281
	P	Receiver	0	0.000	25.934	0.000	0.000	46.212
	R	Sender	4	3.195	53.739	1.125	0.003	—
	P	Sender	5	4.066	52.991	1.055	-0.006	—
MG	R	Receiver	0	0.000	56.287	0.000	0.000	65.753
	P	Receiver	0	0.000	56.311	0.000	0.000	64.454
	R	Sender	4	3.297	41.981	1.809	0.007	—
	P	Sender	4	3.007	42.410	1.963	0.004	—
RD	R	Receiver	0	0.000	25.902	0.000	0.000	56.733
	P	Receiver	0	0.000	25.928	0.000	0.000	43.595
	R	Sender	6	3.226	47.475	1.945	0.004	—
	P	Sender	6	3.314	47.357	1.938	0.003	—
RWP	R	Receiver	0	0.000	25.893	0.000	0.000	65.284
	P	Receiver	0	0.000	25.870	0.000	0.000	56.229
	R	Sender	6	4.905	51.262	1.908	0.003	—
	P	Sender	7	3.469	50.997	1.842	-0.007	—

Table 4.4 Average Performance Metrics of all stations for ONOS (R) and PreInstall ONOS (P) across Mobility Models

Mobility	Ctrl.	Role	Avg HO	Avg HO Time (s)	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	0	0.000	25.933	0.000	0.000	55.159
	P	Receiver	0	0.000	25.780	0.000	0.000	42.260
	R	Sender	4	4.584	53.669	1.122	-0.000	—
	P	Sender	5	3.290	53.398	1.109	0.001	—
MG	R	Receiver	0	0.000	56.315	0.000	0.000	65.947
	P	Receiver	0	0.000	56.128	0.000	0.000	55.939
	R	Sender	4	4.156	42.464	1.964	0.004	—
	P	Sender	4	3.100	42.675	1.939	0.002	—
RD	R	Receiver	0	0.000	25.946	0.000	0.000	59.624
	P	Receiver	0	0.000	25.874	0.000	0.000	52.040
	R	Sender	6	2.829	47.675	1.899	0.001	—
	P	Sender	6	2.882	47.453	1.927	0.000	—
RWP	R	Receiver	0	0.000	25.910	0.000	0.000	65.686
	P	Receiver	0	0.000	25.803	0.000	0.000	54.839
	R	Sender	6	4.143	51.139	1.874	-0.001	—
	P	Sender	7	3.380	51.227	1.909	0.003	—

Table 4.5 Average Performance Metrics of all stations for OpenFlow (R) and PreInstall OpenFlow (P) across Mobility Models

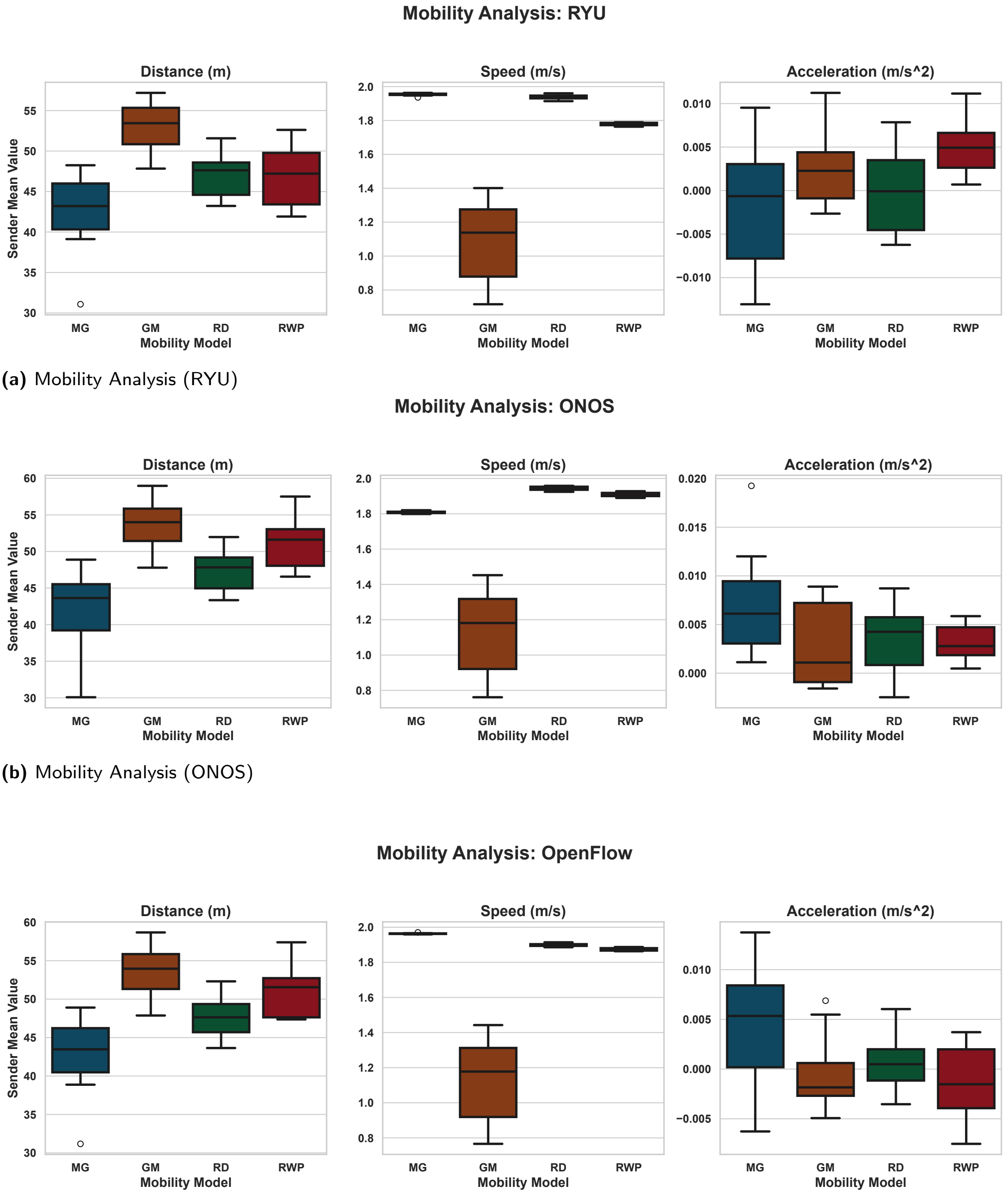
Mobility	Ctrl.	Role	Avg HO	Avg HO Time (s)	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	0	0.000	25.807	0.000	0.000	65.334
	P	Receiver	0	0.000	25.835	0.000	0.000	48.629
	R	Sender	4	4.092	53.083	1.081	0.002	—
	P	Sender	5	4.199	53.569	1.114	-0.001	—
MG	R	Receiver	0	0.000	25.912	0.000	0.000	72.214
	P	Receiver	0	0.000	25.842	0.000	0.000	61.931
	R	Sender	4	5.179	42.252	1.953	-0.002	—
	P	Sender	4	2.859	42.249	1.953	0.002	—
RD	R	Receiver	0	0.000	25.847	0.000	0.000	68.604
	P	Receiver	0	0.000	25.911	0.000	0.000	48.846
	R	Sender	6	3.836	47.174	1.939	-0.000	—
	P	Sender	6	2.896	47.537	1.952	0.008	—
RWP	R	Receiver	0	0.000	23.776	0.000	0.000	73.512
	P	Receiver	0	0.000	25.919	0.000	0.000	55.569
	R	Sender	6	7.319	47.062	1.778	0.005	—
	P	Sender	7	3.106	51.560	1.928	0.002	—

Table 4.6 Average Performance Metrics of all stations for RYU (R) and PreInstall RYU (P) across Mobility Models

Mobility Statistics

Spatial distance does not explain why networks fail based on the results of the analysis. Although the GM model has the highest average distance (approximately 55 m), it produces the least amount of packet loss between controllers. Conversely, while the MG model has the tightest cluster (lowest mean distance), it has the largest number of packets lost. Therefore, it appears that how predictable and how smoothly a mobility trace progresses is significantly more important than simply being close to a SDN controller for maintaining its path to the controller.

Each model displays a similar speed and acceleration profile regardless of which SDN controller was used, and each model shows a significant range in speed when comparing all of the controllers. Small variations exist between the two controllers in terms of acceleration, such as RYU's slightly wider acceleration whiskers as compared to ONOS. However, the small variation in acceleration does not change the basic behavior of the mobility models. The data indicates that the greatest challenges to reactive SDN processes arise from high volatility of movement. The RWP and MG models have characteristics of rapid directional changes and large shifts in acceleration that result in the largest amount of link breaks, thus producing the largest amount of packet loss.



(a) Mobility Analysis (RYU)

(b) Mobility Analysis (ONOS)

(c) Mobility Analysis (OpenFlow)

Figure 4.19 Mobility Statistics Analysis For Each Reactive Controller.

4.2.4 Experiment D

To evaluate controller performance under high-density stress, the experiment was expanded to include 64 mobile stations and 16 APS. In this setup, even-numbered stations served as receivers and odd-numbered stations as senders. To offer a representative examination of the entire network environment, data gathering was focused on five particular station pairs. The experimental environment was highly volatile, as mobility was enabled for all 64 stations.

4.2.4.1 The Findings

After data collection, we concentrated on calculating the average packet loss across all stations in a bar plot with a 95% CI.

Packet Loss

We showed a consistent reduction in packet loss across all the mobile scenarios using the proactive method. With a lowest packet loss achieved in MG mobility (74%) and highest in RD mobility (86.4%) using the reactive OF method. This has been reduced to 69.1% and 81.2% using the proactive method. Across all the mobility using the proactive method, we got an average of almost 5% reduction in packet loss. Even with a lower mean, the CI suggested a high variance in performance.

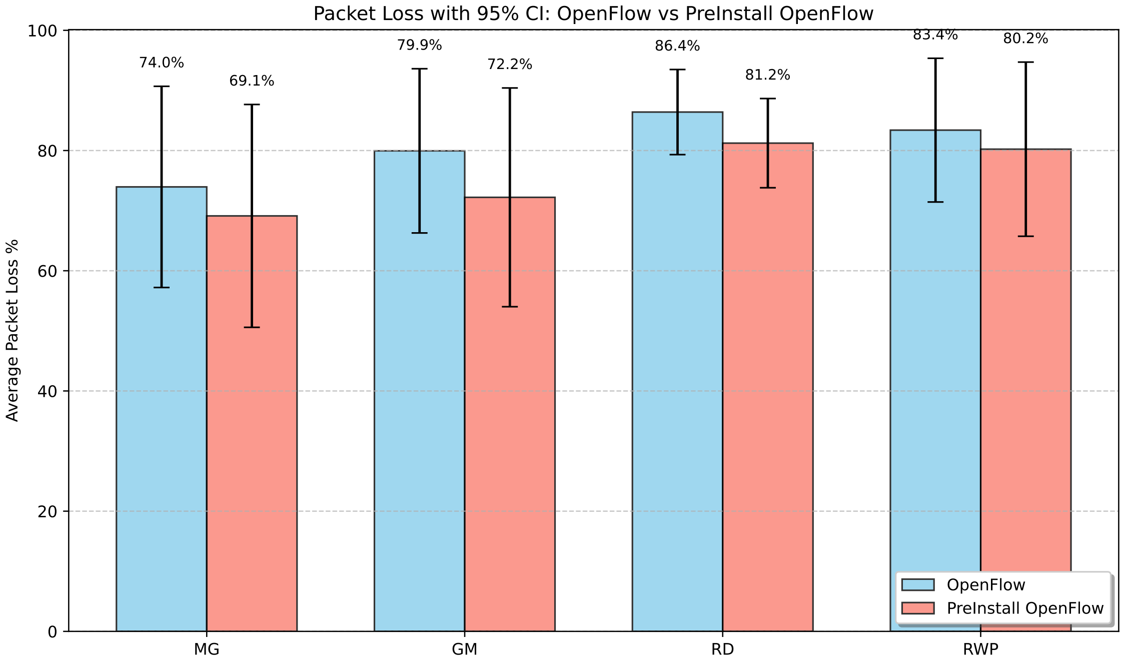


Figure 4.20 Overall packet loss using OpenFlow and Pre-Install OpenFlow controller

Using ONOS we found a mixed performance. In some cases, it performed better than standard OF but slightly higher in certain scenarios. Across GM, RD, and RWP, ONOS maintains an average loss rate between 83% and 88%. Using a proactive method, it achieved a notable improvement in MG mobility, with a 67.2% loss, the lowest among all three controllers. The average improvement through all mobility ranges from 3% to 7% in this case. Analyzing the CI for the proactive method, we found a tighter CI for RD and RWP compared to GM mobility.

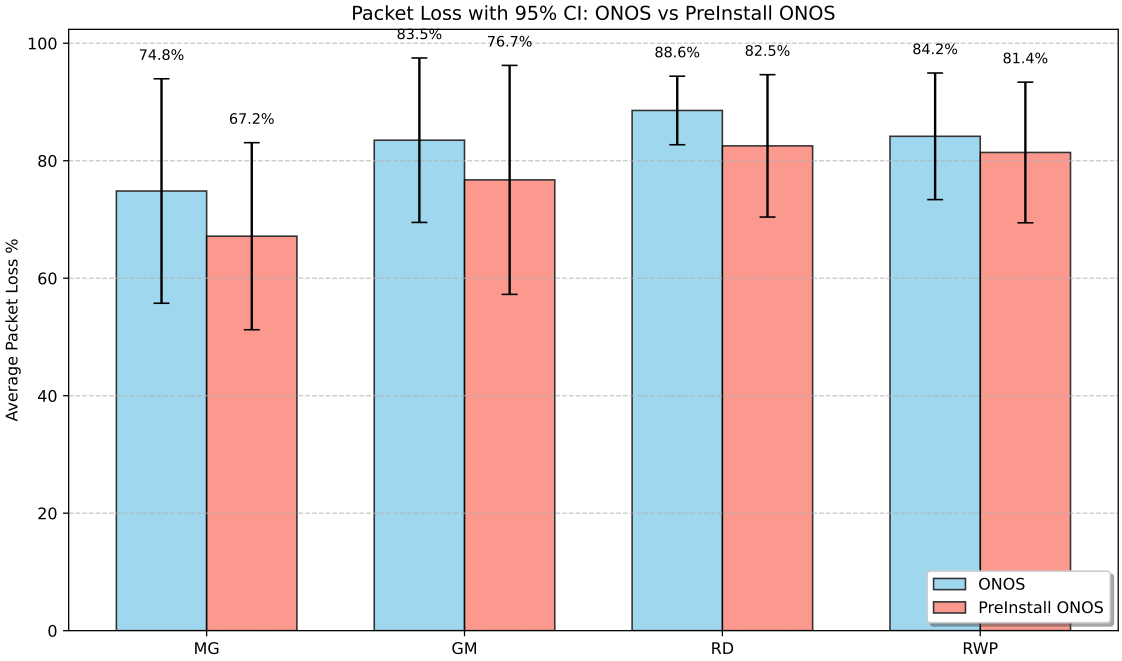


Figure 4.21 Overall packet loss using ONOS and Pre-Install ONOS controller

Among all the controllers, RYU shows the highest packet loss in all scenarios, particularly in RD mobility. It reaches a maximum average packet loss of 90.9% in RD scenario, but is reduced by more than 10% using the proactive method. The MG appears to be the most 'controller-friendly', achieving an average of approximately 75% across all controllers. Here, the average improvement ranges from 3% to 10%, which is the highest among the three controllers.

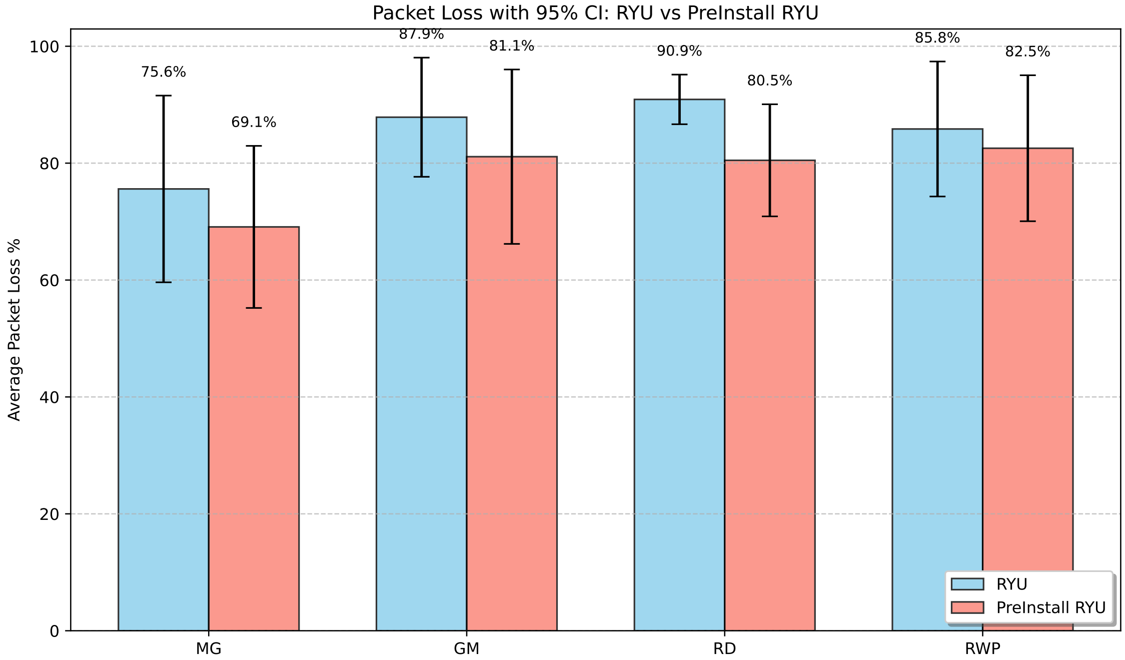


Figure 4.22 Overall packet loss using ONOS and Pre-Install ONOS controller

According to our analysis, when we increase the number of nodes, it shows a higher packet loss in RD and RWP followed by GM mobility. MG showed the lowest loss among all controllers with an average of 75%. Also, when we increased the node, we observed an almost similar performance in all controllers, which was not seen when we used smaller scenarios, where RYU was constantly performing worse than the others.

Network Analysis

- Delay: The analysis shows that ONOS consistently achieves the lowest median latency across different mobility models, particularly in the RD and RWP models where its performance is most stable. In contrast, the GM model presents the most challenge for all controllers, resulting in higher median delays and a wider spread of data, with OF exhibiting the highest peak latency values. The increased variability (greater whiskers) in the MG and GM models demonstrates that the structured movement patterns produce more flow recomputations events, which are handled by OF and RYU with less effectiveness than ONOS.

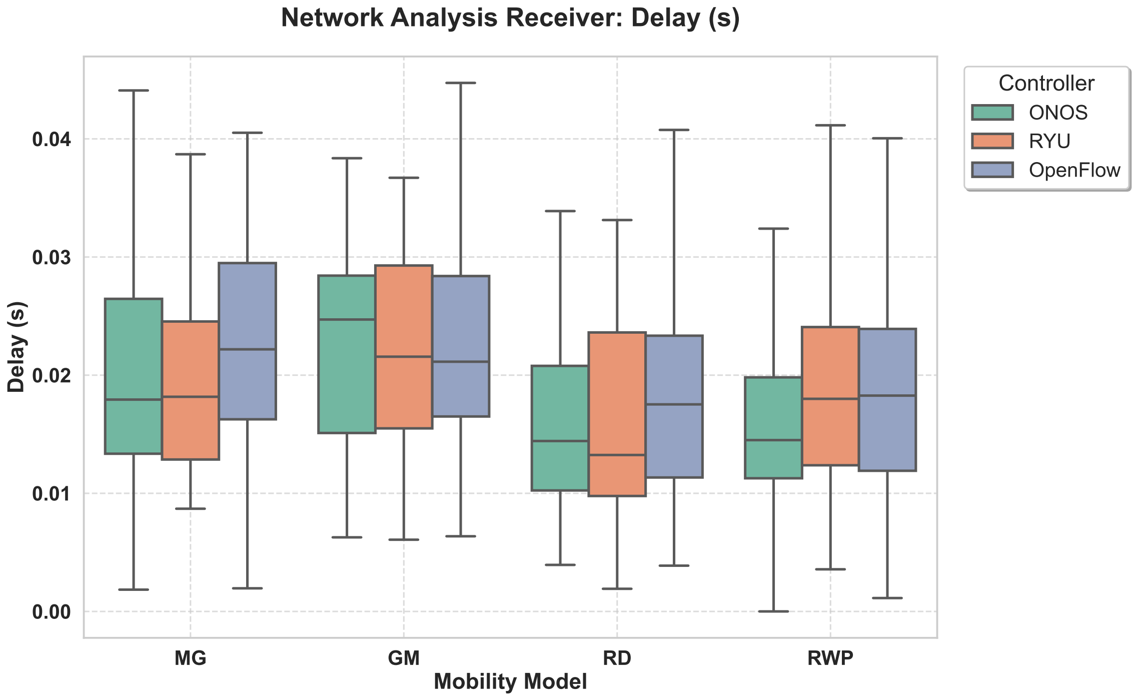


Figure 4.23 Analysis of Delay for All Controllers

- Jitter: The evaluation shows that ONOS provides the most consistent packet delivery intervals, maintaining a lower and more compact distribution across most scenarios. The GM model remains the most unpredictable environment as it causes OF to reach jitter peaks above 0.05s, which could severely impact real-time communication. While the RWP model shows relatively comparable median jitter for all three controllers, RYU and OF display significantly higher maximum values, indicating that they struggle to maintain stable timing as nodes change direction and speed randomly.

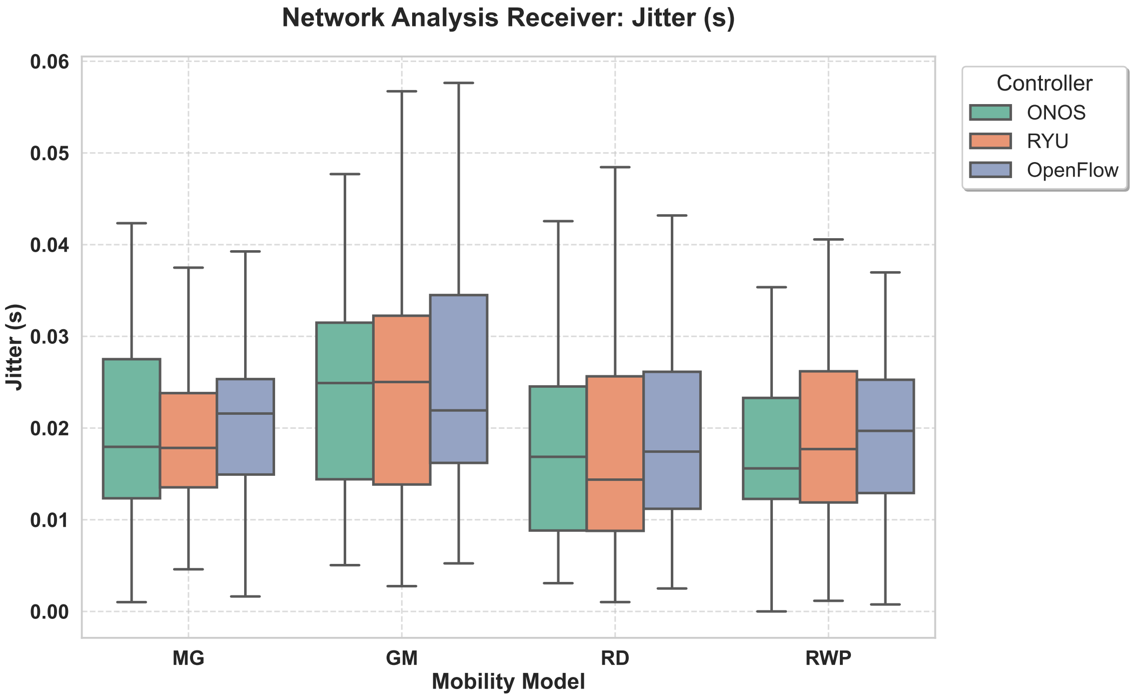


Figure 4.24 Analysis of Jitter for All Controllers

- Bitrate: ONOS and OF controllers were able to achieve median bitrate performance better than Ryu in both GM and MG. However, all three controllers perform poorly when using the RD model. Although ONOS shows the highest overall bitrate with peaks exceeding 3.0 Mbps in the MG model, its performance displays higher variance, suggesting that while it can maximize throughput, it remains sensitive to the rapid topology changes inherent in high-mobility scenarios.

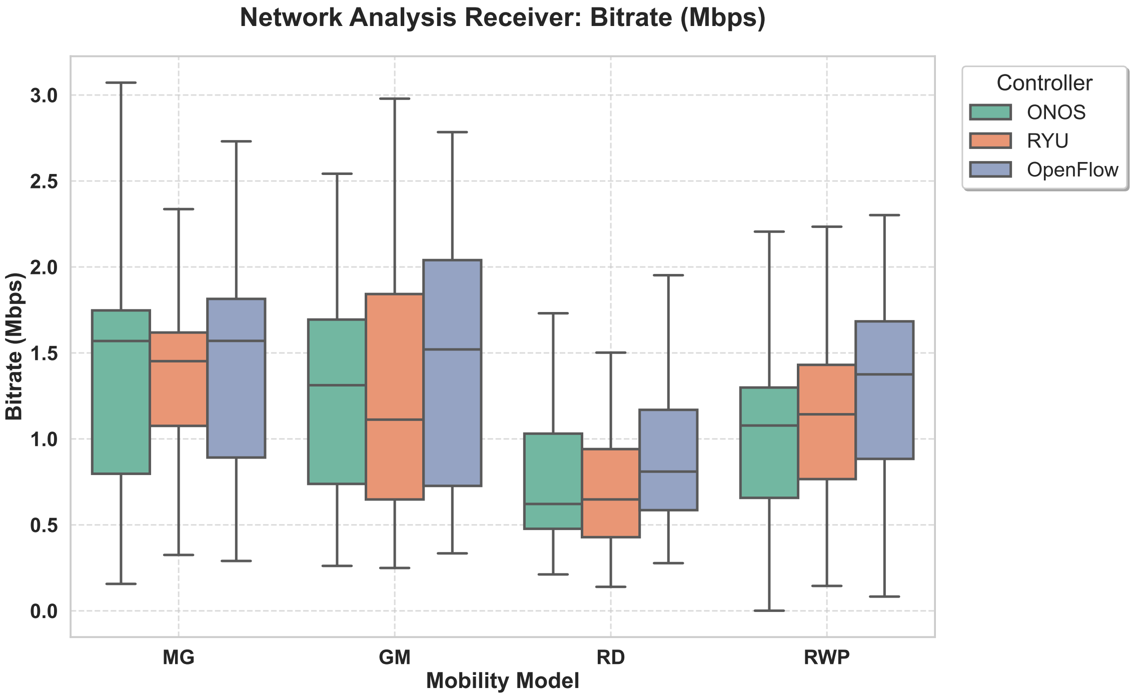


Figure 4.25 Analysis of Bitrate for All Controllers

Performance Metrics

According to the analysis, we see an average of 6 to 8 handovers in GM and MG, whereas RWP and RD have 10 to 12 handovers in the reactive approach. We see an almost similar average speed and average distance for all mobility with a slightly higher distance in RWP. This suggests that they had an almost similar signal in all mobility models. Along with the average handover, the average time of each handover in RWP and RD is comparatively higher than other mobility. This number of handovers and their average time tends to peak the packet loss in these mobilities.

Mobility	Ctrl.	Role	Avg HO	Avg HO Time (s)	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	3	3.662	43.567	1.950	-0.005	83.492
	P	Receiver	3	3.635	43.469	1.954	0.001	76.737
	R	Sender	3	3.689	41.968	1.949	-0.001	—
	P	Sender	3	4.233	41.994	1.956	0.002	—
MG	R	Receiver	4	4.007	44.051	1.952	-0.001	74.843
	P	Receiver	4	3.292	43.991	1.956	-0.004	67.151
	R	Sender	4	7.884	46.057	1.949	0.000	—
	P	Sender	4	4.133	46.269	1.955	-0.002	—
RD	R	Receiver	6	5.650	48.349	1.955	-0.003	88.555
	P	Receiver	7	3.104	48.437	1.952	0.000	82.528
	R	Sender	6	4.686	49.784	1.953	-0.001	—
	P	Sender	7	3.397	49.766	1.943	0.001	—
RWP	R	Receiver	5	4.464	50.630	1.916	-0.007	84.155
	P	Receiver	5	4.634	50.528	1.950	-0.003	81.405
	R	Sender	5	5.398	53.962	1.922	-0.006	—
	P	Sender	6	3.770	54.206	1.949	-0.004	—

Table 4.7 Average Performance Metrics of all stations for ONOS (R) and PreInstall ONOS (P) across Mobility Models

Mobility	Ctrl.	Role	Avg HO	Avg HO Time (s)	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	3	4.065	43.964	1.883	-0.006	79.949
	P	Receiver	3	4.731	43.245	1.693	0.012	72.214
	R	Sender	3	3.424	41.867	1.878	-0.004	—
	P	Sender	3	6.105	41.245	1.632	0.010	—
MG	R	Receiver	4	8.208	44.131	1.869	0.001	73.950
	P	Receiver	3	4.198	42.742	1.593	0.015	69.125
	R	Sender	4	4.732	46.078	1.869	0.003	—
	P	Sender	3	3.543	45.365	1.590	0.014	—
RD	R	Receiver	6	4.765	48.186	1.932	-0.001	86.401
	P	Receiver	6	4.517	47.842	1.905	0.013	81.231
	R	Sender	6	5.064	49.483	1.927	0.002	—
	P	Sender	7	4.164	49.221	1.894	0.010	—
RWP	R	Receiver	5	5.236	50.154	1.742	-0.001	83.394
	P	Receiver	4	4.766	49.500	1.392	0.016	80.227
	R	Sender	5	6.814	53.905	1.746	-0.002	—
	P	Sender	4	6.220	53.329	1.395	0.016	—

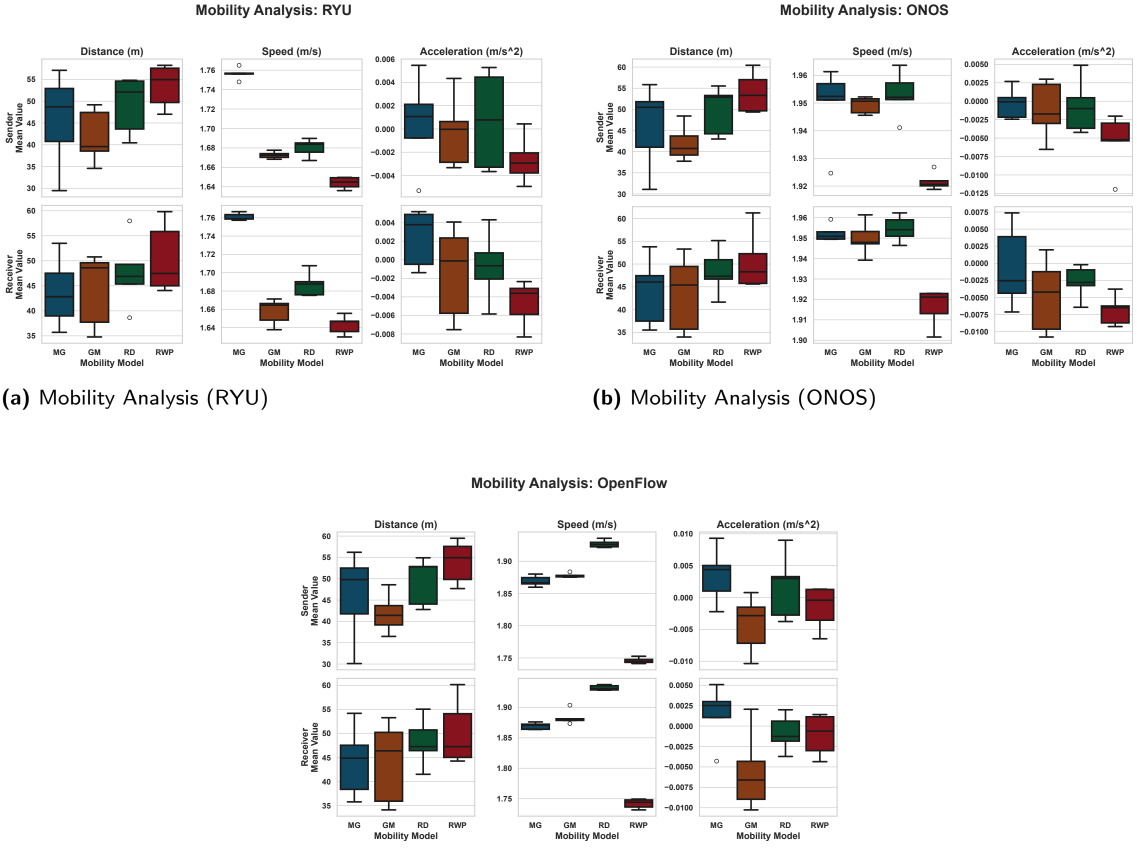
Table 4.8 Average Performance Metrics of all stations for OpenFlow (R) and PreInstall OpenFlow (P) across Mobility Models

Mobility	Ctrl.	Role	Avg HO	Avg HO Time (s)	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	2	3.806	44.299	1.658	-0.001	87.858
	P	Receiver	3	4.064	43.546	1.942	-0.001	81.104
	R	Sender	3	5.110	41.856	1.673	0.000	—
	P	Sender	3	5.250	41.751	1.940	0.000	—
MG	R	Receiver	3	5.806	43.702	1.761	0.002	75.586
	P	Receiver	4	3.151	44.145	1.954	0.004	69.090
	R	Sender	3	7.090	45.783	1.756	0.001	—
	P	Sender	4	2.683	46.110	1.950	0.000	—
RD	R	Receiver	5	8.346	47.628	1.687	-0.001	90.902
	P	Receiver	6	3.723	48.580	1.916	-0.001	80.482
	R	Sender	5	9.803	49.111	1.680	0.001	—
	P	Sender	7	3.692	49.656	1.907	-0.002	—
RWP	R	Receiver	4	4.880	50.441	1.643	-0.005	85.843
	P	Receiver	5	4.266	50.473	1.940	-0.000	82.549
	R	Sender	4	9.306	53.494	1.644	-0.003	—
	P	Sender	5	4.964	54.104	1.939	0.001	—

Table 4.9 Average Performance Metrics of all stations for RYU (R) and PreInstall RYU (P) across Mobility Models

Mobility Statistics

Across all controllers, RWP consistently results in the greatest distance between nodes, while GM maintains the shortest distance. As maximum node distances (up to 60m) are reached in this mobility, and steady speeds stretch the signal range to its limit, packet loss is notably higher. Variation in acceleration further stresses controller performance, particularly in the GM model, which exhibits the highest volatility. While ONOS proves the most efficient by maintaining the highest speeds with comparable loss, RYU demonstrates the lowest efficiency, suffering a peak packet loss of 90.9% in the RD model despite operating at lower average speeds.



(a) Mobility Analysis (RYU)

(b) Mobility Analysis (ONOS)

(c) Mobility Analysis (OpenFlow)

Figure 4.26 Mobility Statistics Analysis For Each Reactive Controller

4.2.5 Experiment E

In this experiment, we maintained all the same characteristics used in experiment D, but instead of keeping all the stations mobile, we set only the sender nodes as mobile nodes and placed the receiver close to the APS for better signal. Also, here we collected data from 20 stations instead of 10 stations in the last experiment. We averaged the packet loss for all the stations and plotted the analysis using the bar plots with 95% CI.

4.2.5.1 The Finding:

We present the average packet loss of all 10 receivers below.

Packet Loss

The OF controller demonstrated a consistent reduction in packet loss when we utilized the proactive method in all scenarios. Reactive OF experienced the highest average packet loss in the RD model at 80.6%, while the MG model shows the lowest at 76.6%. The most significant improvement occurred in the GM model, where loss drops from 78.7% to 72.1%. Despite lower average loss, the MG model with the proactive method showed a wider CI compared to other models. Here we got an average improvement ranging from a minimum of 1.3% in RD to a maximum of 6.6% in GM mobility.

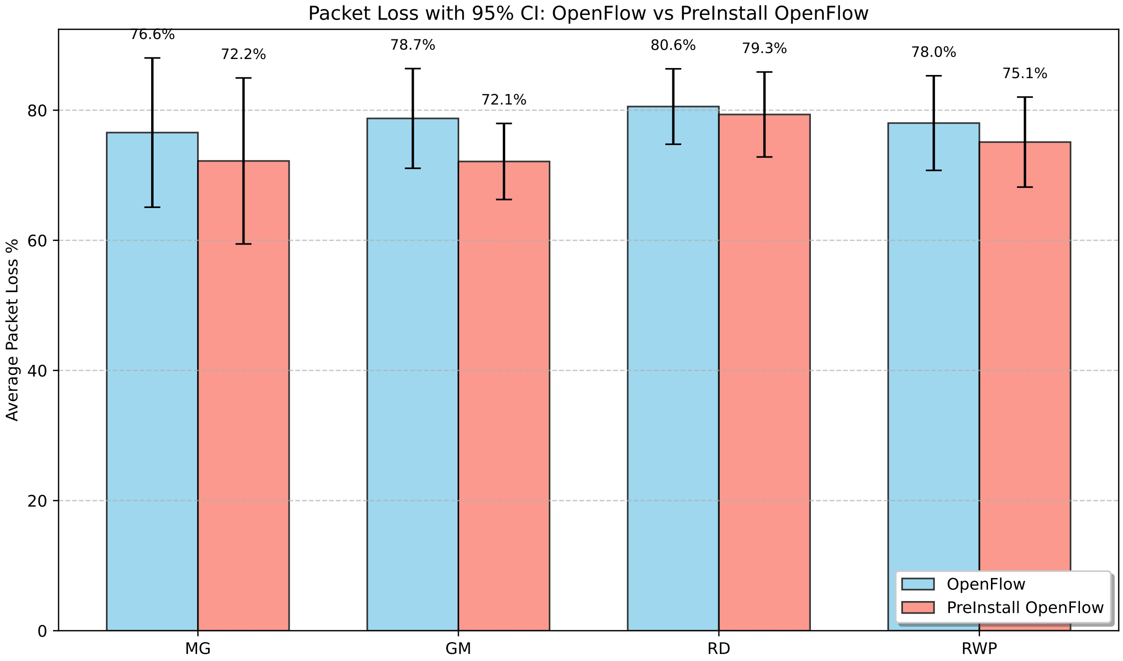


Figure 4.27 Overall packet loss using OpenFlow and Pre-Install OpenFlow controller

Using reactive ONOS we achieved better performance in all mobility except RWP, where there's a slight rise of packet loss from 78% (OF) to 78.9% (ONOS). Using the proactive method ONOS showed its best performance in GM and RD model, with loss rates reduced to 66.4% and 67.8%, respectively. Here the average loss ranged from 2% in MG to 9.6% in GM model.

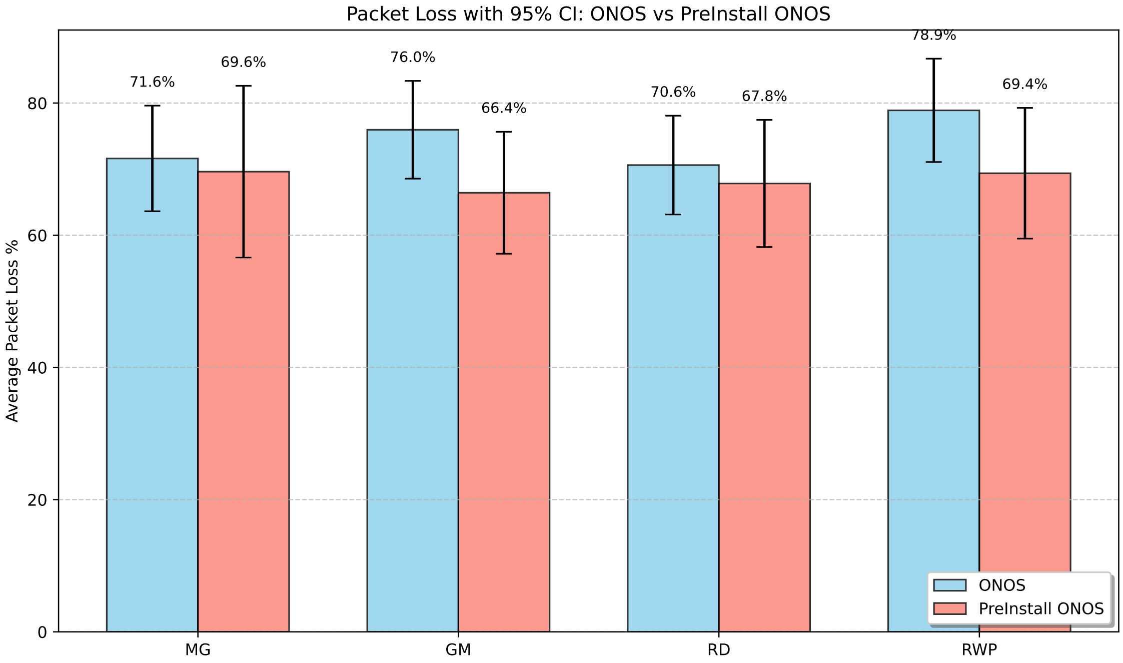


Figure 4.28 Overall packet loss using ONOS and Pre-Install ONOS controller

RYU showed the greatest improvement with the proactive method among all three controllers, with an average ranging from a minimum of 13.1% to a maximum of 17.9%. The reactive analysis showed the greatest loss in the GM scenario, at 83.9%. Using the proactive approach, the loss is reduced from 83.0% to 66.5% in the RD scenario, and from 82.5% to 66.2% in the RWP scenario. The lowest for this controller was achieved for MG model (63.5%). Even with a high packet-loss rate in the reactive method, RYU showed promising results with the proactive method compared to other controllers.

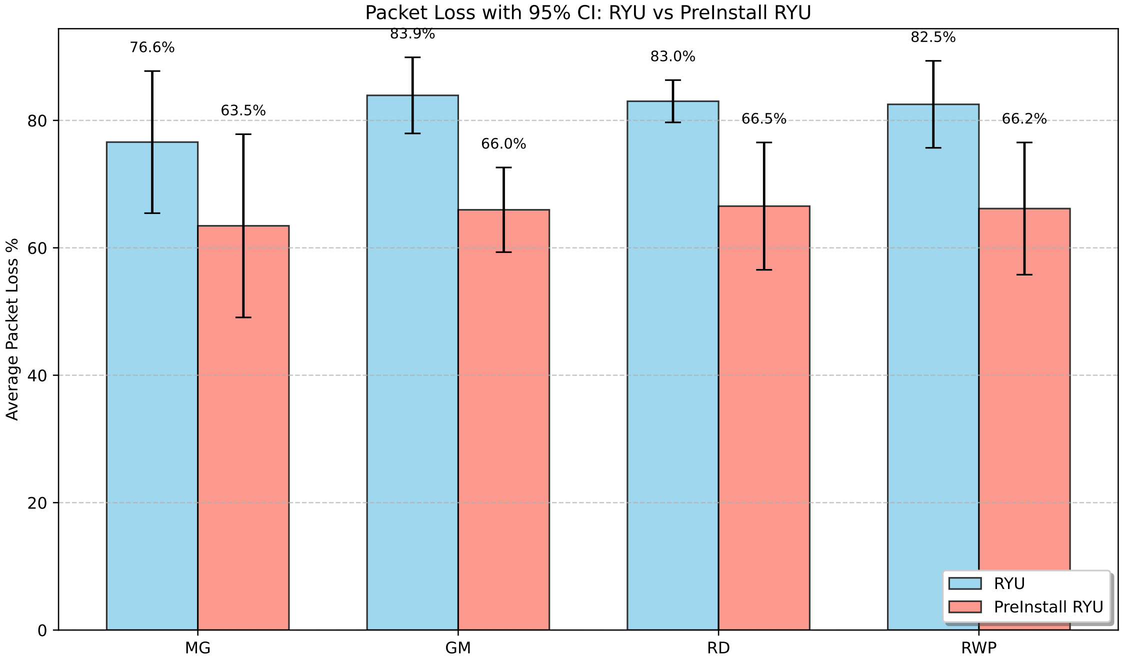


Figure 4.29 Overall packet loss using RYU and Pre-Install RYU controller

According to the analysis, it showed that RWP has a comparatively higher packet loss throughout all controllers, followed by GM. For RD, a mixed performance has been noted, as ONOS achieved the lowest packet loss while the other controllers achieved the highest. Also it can be seen that compared to other controllers ONOS has an overall better performance in the reactive scenario for all mobility, and in terms of proactive RYU performed the best as it achieved the lowest packet drop in all mobility.

Network Analysis

- Delay: The comparison of the network delay among the three controllers in this study shows that the ONOS controller was the best performing with the least amount of median delay time in each of the different mobility test cases. Although all controllers exhibit a performance degradation under the GM model, ONOS maintained the smallest interquartile range of delay time in this model. Conversely, the OF controller experienced the largest total delay time and was also the most sensitive to node movement, especially in GM and MG mobility models. These results indicate that the OF controller's ability to update flow entries is less than ideal in high-dynamic environments.

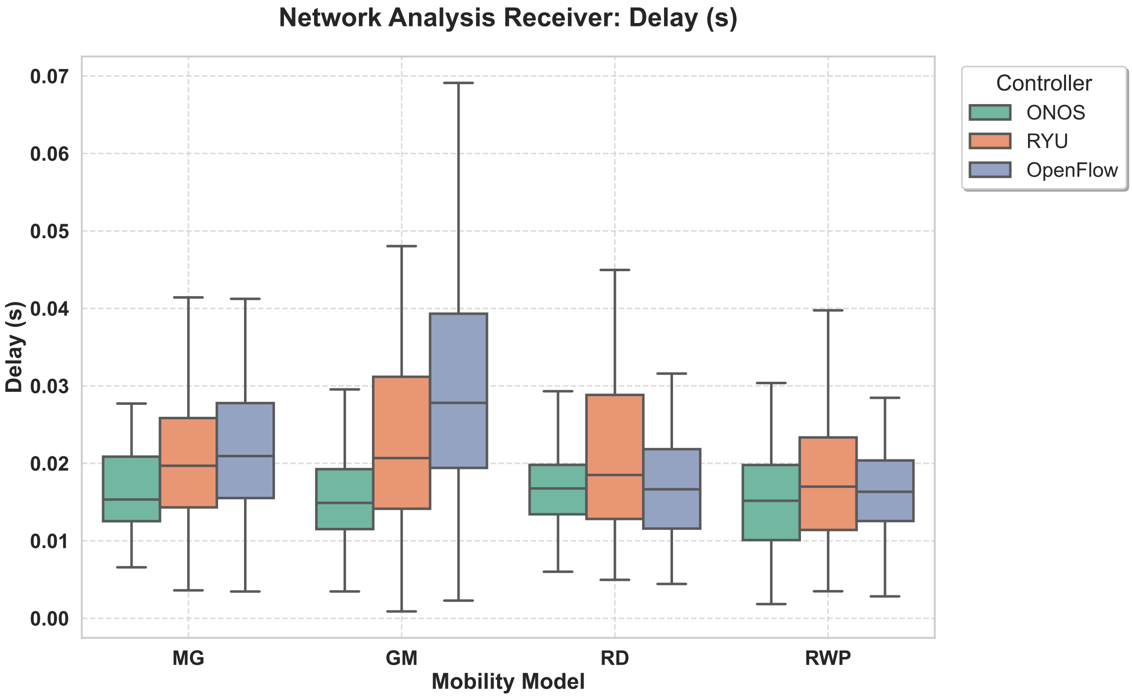


Figure 4.30 Analysis of Delay for All Controllers

- Jitter: As indicated in the delay analysis, the results of the jitter analysis provided additional evidence of the differences between the three controllers' performance in terms of how quickly they delivered packets and how consistent the packet delivery intervals were. The data showed that ONOS was able to provide a more stable and predictable delivery interval for packets. As previously mentioned, the GM mobility model provided the most challenging environment for the three controllers, producing extreme jitter peak times greater than .06 seconds for the OF controller. Additionally, the larger whisker sizes found in the jitter distributions for RYU and OF for each of the different mobility models further emphasized the lack of predictability in their flow management during node transitions. On the other hand, the smaller whisker sizes for ONOS for each of the mobility models demonstrated a more reliable approach to managing frequent topology updates.

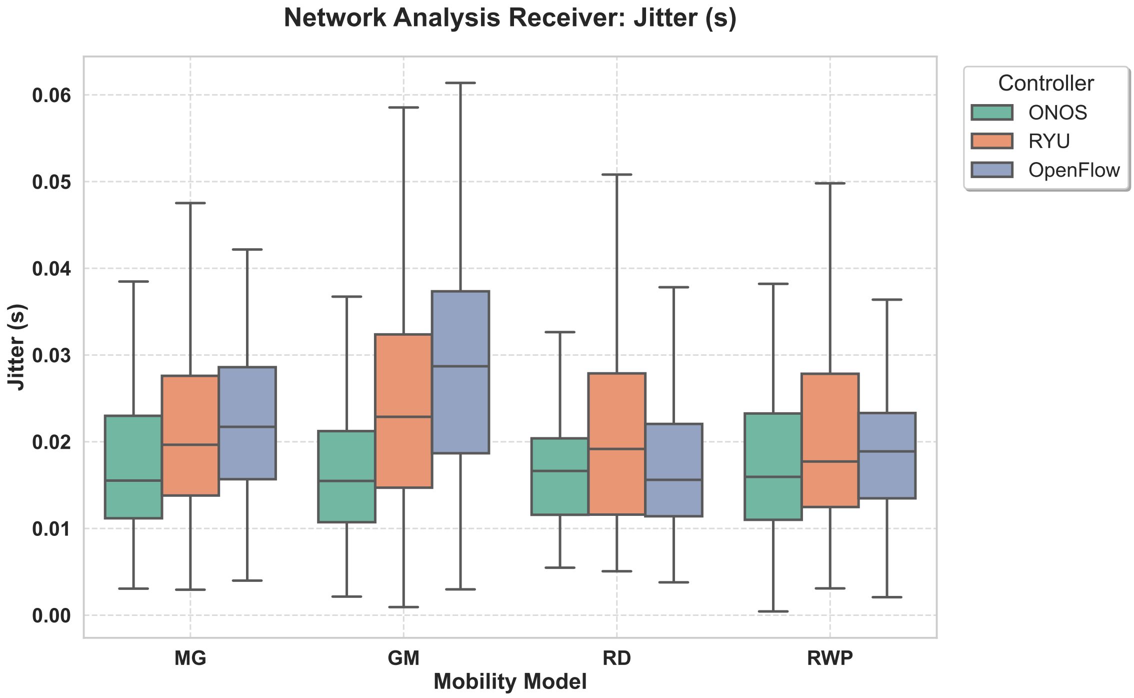


Figure 4.31 Analysis of Jitter for All Controllers

- Bitrate: ONOS provided the best bitrate performance, especially when the MG model was used, with bitrate performance peaking over 4.0 Mb/s, and provided better performance than RYU and OF, which struggled to achieve effective bitrate performance when simulating node movement using the GM model. Similar to the previous results, although all three controllers experienced a decrease in throughput due to node movement in the GM model, ONOS achieved the highest median bitrate performance of the three controllers. However, similar to the previous results, the higher variances in the bitrate performance for ONOS for the MG and RD models demonstrate that, although ONOS is capable of achieving the highest bitrate performance when the nodes move in an optimal manner and through optimal network conditions, the performance remains dependent upon the physical constraints and the node density variations present in each of the different mobility models.

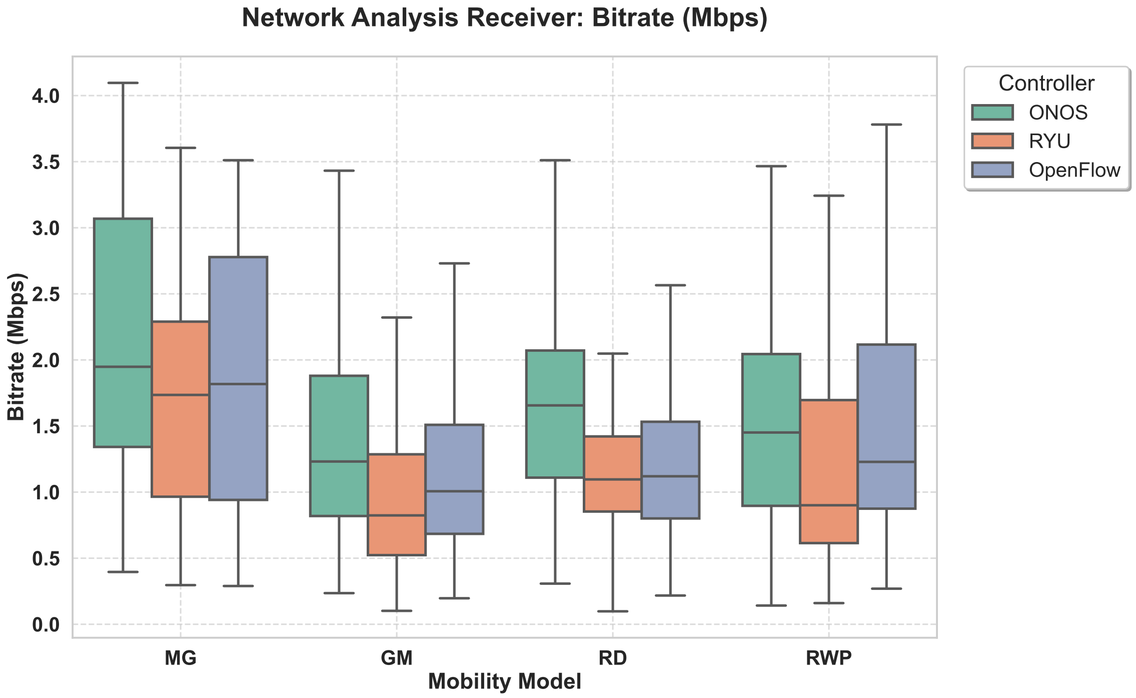


Figure 4.32 Analysis of Bitrate for All Controllers

Performance Metrics

As the setup uses the node pattern used in Experiment D, the number of handovers for the sender is similar to that experiment. By eliminating mobility from half of the stations, the system experiences less stress, leading to a more seamless handover. This configuration improves reliability, decreasing packet loss by an average of 5% to 10%. Despite the overall reduction in movement, RWP continues to demonstrate the poorest performance, while GM and RD show a mixed performance.

Mobility	Ctrl.	Role	Avg HO	Avg HO Time	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	0	0.000	20.741	0.000	0.000	75.955
	P	Receiver	0	0.000	20.329	0.000	0.000	66.427
	R	Sender	3	4.539	54.885	0.658	0.003	—
	P	Sender	4	3.972	54.503	1.051	-0.006	—
MG	R	Receiver	0	0.000	57.899	0.000	0.000	71.616
	P	Receiver	0	0.000	57.855	0.000	0.000	69.618
	R	Sender	3	11.709	42.963	1.916	-0.000	—
	P	Sender	3	3.826	42.940	1.958	0.004	—
RD	R	Receiver	0	0.000	25.460	0.000	0.000	70.612
	P	Receiver	0	0.000	25.469	0.000	0.000	67.831
	R	Sender	3	4.877	45.130	1.253	0.005	—
	P	Sender	5	3.893	46.349	1.779	0.004	—
RWP	R	Receiver	0	0.000	25.501	0.000	0.000	78.892
	P	Receiver	0	0.000	25.481	0.000	0.000	69.379
	R	Sender	6	6.500	51.278	1.952	0.001	—
	P	Sender	7	3.762	51.132	1.939	-0.002	—

Table 4.10 Average Performance Metrics of all stations for ONOS (R) and PreInstall ONOS (P) Controllers across Mobility Models

Mobility	Ctrl.	Role	Avg HO	Avg HO Time (s)	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	0	0.000	25.460	0.000	0.000	78.737
	P	Receiver	0	0.000	24.937	0.000	0.000	72.120
	R	Sender	4	6.012	54.214	1.041	-0.004	—
	P	Sender	4	4.234	54.693	1.008	0.002	—
MG	R	Receiver	0	0.000	25.442	0.000	0.000	76.560
	P	Receiver	0	0.000	25.131	0.000	0.000	72.201
	R	Sender	3	4.111	43.014	1.833	-0.002	—
	P	Sender	3	3.735	42.505	1.511	0.010	—
RD	R	Receiver	0	0.000	25.393	0.000	0.000	80.564
	P	Receiver	0	0.000	25.784	0.000	0.000	79.344
	R	Sender	5	4.149	46.299	1.791	0.000	—
	P	Sender	5	5.253	45.768	1.576	0.005	—
RWP	R	Receiver	0	0.000	25.476	0.000	0.000	78.022
	P	Receiver	0	0.000	25.200	0.000	0.000	75.104
	R	Sender	6	5.827	51.336	1.914	0.004	—
	P	Sender	6	4.697	50.812	1.895	0.005	—

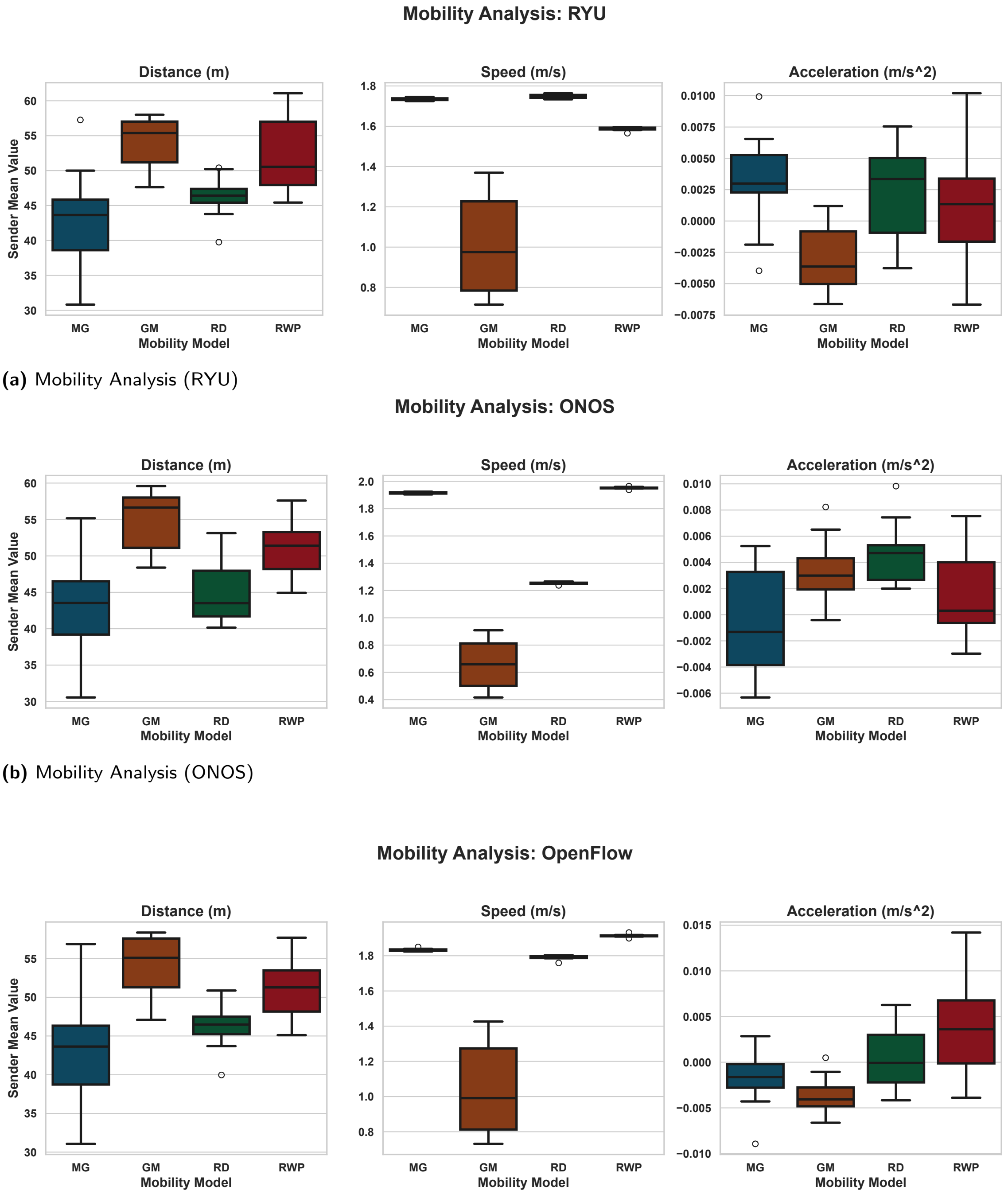
Table 4.11 Average Performance Metrics of all stations for OpenFlow (R) and PreInstall OpenFlow (P) across Mobility Models

Mobility	Ctrl.	Role	Avg HO	Avg HO Time (s)	Avg Dist.	Avg Speed	Avg Accel.	Avg Loss (%)
GM	R	Receiver	0	0.000	25.442	0.000	0.000	83.929
	P	Receiver	0	0.000	25.436	0.000	0.000	65.966
	R	Sender	4	7.620	54.145	1.012	-0.003	—
	P	Sender	4	6.624	54.382	1.045	-0.001	—
MG	R	Receiver	0	0.000	25.415	0.000	0.000	76.592
	P	Receiver	0	0.000	25.429	0.000	0.000	63.454
	R	Sender	3	4.257	42.877	1.736	0.003	—
	P	Sender	3	7.124	42.901	1.922	-0.001	—
RD	R	Receiver	0	0.000	25.425	0.000	0.000	83.008
	P	Receiver	0	0.000	25.435	0.000	0.000	66.542
	R	Sender	5	5.114	46.259	1.749	0.002	—
	P	Sender	5	3.438	46.524	1.937	-0.002	—
RWP	R	Receiver	0	0.000	25.381	0.000	0.000	82.522
	P	Receiver	0	0.000	25.437	0.000	0.000	66.161
	R	Sender	5	9.137	52.253	1.587	0.001	—
	P	Sender	7	3.627	51.142	1.936	-0.003	—

Table 4.12 Average Performance Metrics of all stations for RYU (R) and PreInstall RYU (P) across Mobility Models

Mobility Statistics

One major outcome was that the GM model has a very significant dependence on which controller is being used. Specifically, the ONOS environment resulted in GM having significantly slower average sender speeds ($0.65m/s$) than both RYU and OF, which maintained an average speed of approximately $1.0m/s$. However, the RWP model acts as a benchmark as it consistently produced the fastest and most stable sender speeds in all tested environments, maintaining a high range between $1.6m/s$ and $1.9m/s$.



(a) Mobility Analysis (RYU)

(b) Mobility Analysis (ONOS)

(c) Mobility Analysis (OpenFlow)

Figure 4.33 Mobility Statistics Analysis For Each Reactive Controller

5

Conclusion

These studies findings show that using a proactive approach to manage traffic clearly produces better performance than traditional reactive approaches, especially in networks that have complex structures. In smaller networks, the benefits of proactive flow management were insignificant. When network node speeds were doubled, significant performance degradation was noticed, and both the GM and RWP models suffered, as they naturally exhibit high rates of mobility and rapidly changing network topologies.

When the motion for the nodes in large-scale environments was reduced by half, a significant increase in performance occurred, but even then, the RWP model produced the largest amount of packet loss, almost 80% loss on every controller tested. This indicates that reactive setups can't keep up with the random direction and constant speed changes inherent in the RWP model. The proactive setup alleviated many of these problems, and when subjected to extreme stress testing conditions in large-scale environments, the RYU controller was found to be the best performing, with an improvement of almost 12% in the proactive experiment and also outperforming the proactive experiments of other controllers by an average of almost 2%. This contrasts with mid-scale scenarios, where ONOS and OF provided better reliability than RYU, particularly when dealing with the high packet loss triggered by the MG and RWP models, both having a loss of over 65% in the reactive case and RYU showing a loss of around 73%. This performance was improved using the proactive case where OF outperforms all with an average of 55% in both of this mobility compared to ONOS an average of 60% and RYU 59%.

Results indicate that the RWP model poses the greatest challenge to SDN stability, highlighting a significant struggle for controllers to maintain performance in open-space environments such as campuses or parks. This was followed by high-speed, momentum-based movement typical of highways GM. Conversely, structured movement patterns, such as those found in urban city centers MG or systematic patrols RD, yielded moderate but fluctuating performance levels. These findings are summarized in Table 5.1.

Network Scale	Method	Worst Performance Environment	Best Controller
Small Scale	Reactive	Open-Space (Highway & Park)	OF
	Proactive	Open-Space (Highway & Park)	OF
Mid Scale	Reactive	City Centers, Parks & Campuses	ONOS / OF
	Proactive	City Centers, Parks & Campuses	OF
Large Scale	Reactive	Parks & Campuses	ONOS / OF
	Proactive	Parks & Campuses	RYU

Table 5.1 Summary of Experimental Results across SDN Controllers and Mobility Models

A major consideration when selecting an SDN controller is that while ONOS and OF perform efficiently for small to medium-scale applications, RYU is well-suited for large-scale, high-stress environments when used with proactive traffic management techniques. Thus, while SDN is very scalable and capable of supporting a large number of nodes, the packet loss experienced in this study shows that traditional, reactive SDN architecture is still unreliable for high mobility applications unless active traffic management strategies are integrated into the design.

Reproducibility

All experimental scripts used to produce, evaluate, and plot generation in this thesis can be found in this Git repository: `https://github.com/prettore/Mobility_in_SDN`. The repository contains the whole experimental pipeline followed to produce the result used in this study.

5.1 Future Work

This study establishes an initial basis to understand the effectiveness of SDN controllers with respect to performance in mobile environments through a proactive methodology. However, there are numerous other fields that need to be explored as part of future research. One area of research would include identifying specific architectural obstructions responsible for the performance degradation of SDN controllers when operating within high mobility environments. Additionally, another direction of research could compare a scenario combined with multiple SDN controller architectures into a single environment to assess the combined performance. The third path of research would analyze the effect of hardware diversity (i.e., how the performance is affected by the use of different physical components) on the overall performance of the system.

Bibliography

[1] ARIYAKHAJORN, J., WANNAWILAI, P., AND SATHITWIRIYAWONG, C. A comparative study of random waypoint and gauss-markov mobility models in the performance evaluation of manet. In *2006 International Symposium on Communications and Information Technologies* (2006), IEEE, pp. 894–899.

[2] BELLO, L. L., LOMBARDO, A., MILARDO, S., PATTI, G., AND RENO, M. Experimental assessments and analysis of an sdn framework to integrate mobility management in industrial wireless sensor networks. *IEEE Transactions on Industrial Informatics 16*, 8 (2020), 5586–5595.

[3] BETTSTETTER, C., RESTA, G., AND SANTI, P. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on mobile computing 2*, 3 (2003), 257–269.

[4] CAROFIGLIO, G., CHIASSERINI, C.-F., GARETTO, M., AND LEONARDI, E. Route stability in manets under the random direction mobility model. *IEEE transactions on Mobile Computing 8*, 9 (2009), 1167–1179.

[5] CHAURASIA, A., MISHRA, S. N., AND CHINARA, S. Performance evaluation of software-defined wireless networks in it-sdn and mininet-wifi. In *2019 1st International Conference on Advances in Information Technology (ICAIT)* (2019), pp. 315–319.

[6] CHEN, C., LIN, Y.-T., YEN, L.-H., CHAN, M.-C., AND TSENG, C.-C. Mobility management for low-latency handover in sdn-based enterprise networks. In *2016 IEEE Wireless Communications and Networking Conference* (2016), pp. 1–6.

[7] GLOSS, B., SCHARF, M., AND NEUBAUER, D. A more realistic random direction mobility model. *TD (05) 52* (2005), 13–14.

[8] HANGGORO, A., AND SARI, R. F. Performance evaluation of the manhattan mobility model in vehicular ad-hoc networks for high mobility vehicle. In *2013 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)* (2013), pp. 31–36.

[9] HYYTIÄ, E., AND VIRTAMO, J. Random waypoint mobility model in cellular networks. *Wireless Networks 13*, 2 (2007), 177–188.

[10] KHAN, M. A., DANG, X. T., DOERSCH, T., AND PETERS, S. Mobility management approaches for sdn-enabled mobile networks. *Annals of Telecommunications 73* (2018), 719–731.

[11] Labraoui, M., Boc, M. M., and Fladenmuller, A. Self-configuration mechanisms for sdn deployment in wireless mesh networks. *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (2017), 1–4. `https://api.semanticscholar.org/CorpusID:21740130`.

[12] Marquezan, C. C., Despotovic, Z., Khalili, R., Perez-Caparros, D., and Hecker, A. Understanding processing latency of sdn based mobility management in mobile core networks. In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (2016), pp. 1–7.

[13] Nain, P., Towsley, D., Liu, B., and Liu, Z. Properties of random direction models. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* (2005), vol. 3, IEEE, pp. 1897–1907.

[14] Nguyen, T.-T., Bonnet, C., and Harri, J. Sdn-based distributed mobility management for 5g networks. In *2016 IEEE Wireless Communications and Networking Conference* (2016), pp. 1–7.

[15] Pentikousis, K., Wang, Y., and Hu, W. Mobileflow: Toward software-defined mobile networks. *IEEE Communications Magazine 51*, 7 (2013), 44–53.

[16] Ramakrishan, B., Joe, M. M., and Nishanth, R. B. Modeling and simulation of efficient cluster based manhattan mobility model for vehicular communication. *Journal of emerging technologies in web intelligence 6*, 2 (2014), 253–261.

[17] Sanchez, M. I., de la Oliva, A., and Mancuso, V. Experimental evaluation of an sdn-based distributed mobility management solution. In *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture* (New York, NY, USA, 2016), MobiArch '16, Association for Computing Machinery, p. 31–36. `https://doi.org/10.1145/2980137.2980138`.

[18] Shrivastava, P., and Kataoka, K. Fastsplit: Fast and dynamic ip mobility management in sdn. In *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)* (2016), pp. 166–172.

[19] Silva, R., Santos, D., Meneses, F., Corujo, D., and Aguiar, R. L. A hybrid sdn solution for mobile networks. *Computer Networks 190* (2021), 107958. `https://www.sciencedirect.com/science/article/pii/S1389128621000931`.

[20] Sornlertlamvanich, P., Ang-Chuan, T., Sae-Wong, S., Kamolphiwong, T., and Kamolphiwong, S. Sdn-based network mobility. In *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)* (2016), pp. 1–6.

[21] Sornlertlamvanich, P., Ang-Chuan, T., Sae-Wong, S., Kamolphiwong, T., and Kamolphiwong, S. Sdn-based network mobility. In *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)* (2016), pp. 1–6.

[22] Tantayakul, K., Dhaou, R., and Paillassa, B. Impact of sdn on mobility management. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)* (2016), pp. 260–265.

[23] Tantayakul, K., Dhaou, R., and Paillassa, B. Sdn aided mobility management for connected vehicle networks.

[24] Tong, H., Wang, T., Zhu, Y., Liu, X., Wang, S., and Yin, C. Mobility-aware seamless handover with mptcp in software-defined hetnets. *IEEE Transactions on Network and Service Management 18*, 1 (2021), 498–510.

[25] ZHONG, Z., Da-Yong, L., Shao-Qiang, L., Xiao-Ping, F., and Zhi-Hua, Q. An adaptive localization approach for wireless sensor networks based on gauss-markov mobility model. *Acta Automatica Sinica 36*, 11 (2010), 1557–1568.

List of Figures

List of Tables

A

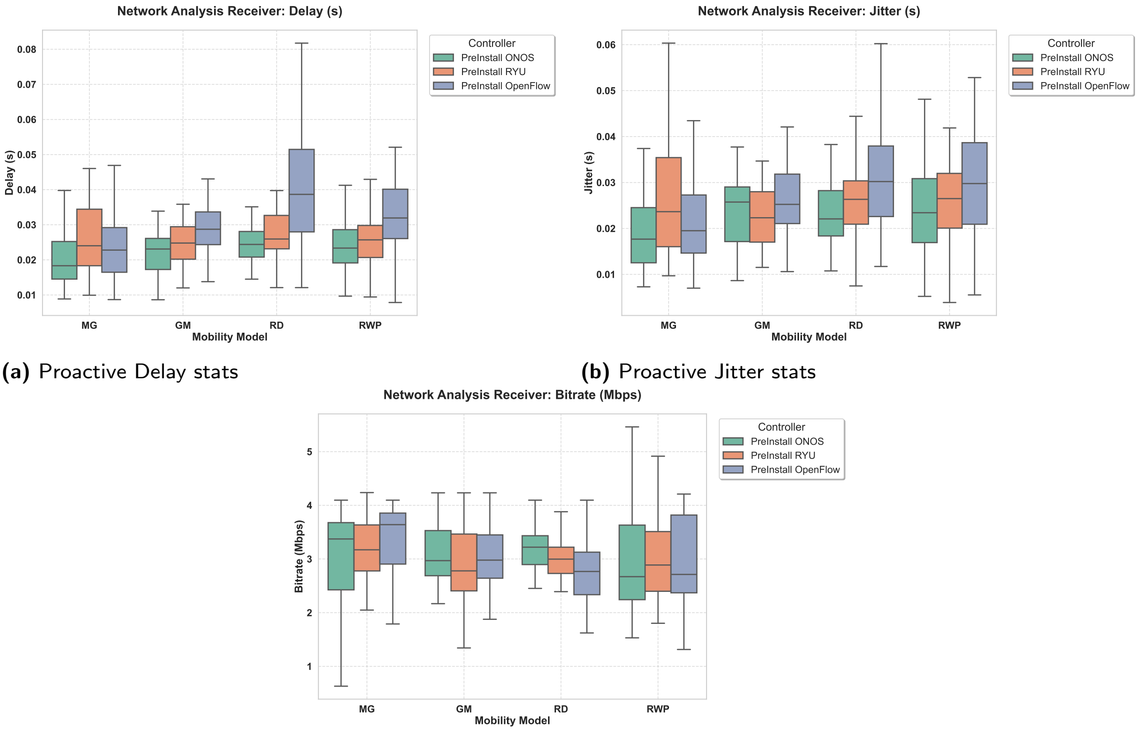
Supplementary Material

Mobility	St.	Ctrl.	Avg HO	HO Time (s)	Speed (m/s)	Accel. (m/s^2)	Dist. (m)	Loss (%)	Loss (Ct)	Total Pkt	Delay (s)
Def. Mob.	sta2	R	3	2.40	2.33	-0.011	48.23	19.21	27	113	0.019
		P	3	2.60	2.29	-0.010	48.48	18.36	26	114	0.018
	sta3	R	2	2.92	2.39	0.074	39.38	26.36	37	103	0.023
		P	3	3.31	2.44	0.067	40.18	26.36	37	103	0.023
GM	sta2	R	0	2.33	8.53	-0.148	70.27	59.14	83	57	0.010
		P	0	2.00	8.23	-0.113	70.89	57.21	80	60	0.016
	sta3	R	6	3.00	15.79	0.090	88.57	74.36	104	36	0.016
		P	6	2.89	15.65	0.013	91.35	70.21	98	42	0.022
MG	sta2	R	2	3.00	1.91	0.002	111.61	47.29	66	74	0.011
		P	2	3.05	1.89	-0.006	112.48	47.21	66	74	0.022
	sta3	R	2	2.39	1.90	0.001	81.66	42.93	60	80	0.012
		P	1	3.40	1.88	-0.003	78.92	37.14	52	88	0.041
RD	sta2	R	2	2.12	4.89	0.020	89.36	42.71	60	80	0.015
		P	2	2.25	4.57	-0.058	91.62	41.50	58	82	0.032
	sta3	R	3	2.13	3.93	-0.000	77.71	52.00	73	67	0.012
		P	3	2.00	3.91	-0.024	76.76	45.57	64	76	0.031
RWP	sta2	R	2	1.95	4.91	0.024	89.83	46.64	65	75	0.015
		P	2	2.30	4.58	-0.058	91.48	44.21	62	78	0.030
	sta3	R	3	2.70	10.34	0.012	95.74	58.07	81	59	0.008
		P	3	2.93	10.16	-0.059	95.24	56.14	79	61	0.025

Table A.1 Performance Metrics for OpenFlow (R) and Pre-Install OpenFlow (P) Controllers

Mobility	St.	Ctrl.	Avg HO	HO Time (s)	Speed (m/s)	Accel. (m/s^2)	Dist. (m)	Loss (%)	Loss (Ct)	Total Pkt	Delay (s)
Def. Mob.	sta2	R	3	2.87	2.24	-0.031	48.43	40.64	57	83	0.034
		P	3	2.33	2.31	-0.006	49.34	35.57	50	90	0.033
	sta3	R	3	3.12	2.46	0.012	40.42	47.50	66	74	0.035
		P	3	3.37	2.56	0.001	41.04	41.50	58	82	0.046
GM	sta2	R	0	2.25	8.63	-0.070	71.03	64.71	91	49	0.018
		P	0	3.67	8.31	-0.069	70.21	64.29	90	50	0.015
	sta3	R	5	3.04	15.88	0.117	88.88	77.86	109	31	0.021
		P	5	5.57	15.47	-0.148	88.99	76.14	107	33	0.033
MG	sta2	R	2	6.00	1.91	-0.002	111.70	59.86	84	56	0.018
		P	2	2.85	1.90	-0.001	112.69	55.29	77	63	0.025
	sta3	R	1	5.93	1.90	0.003	81.86	50.29	70	70	0.022
		P	2	2.82	1.89	0.002	80.75	49.50	69	71	0.032
RD	sta2	R	2	2.32	4.93	0.043	89.85	49.86	70	70	0.020
		P	2	2.38	4.86	-0.048	89.95	49.93	70	70	0.023
	sta3	R	3	2.38	4.02	0.066	78.09	56.71	79	61	0.019
		P	3	2.50	3.96	-0.006	76.82	53.14	74	66	0.036
RWP	sta2	R	2	2.06	4.84	0.015	89.72	56.50	79	61	0.019
		P	2	2.30	4.83	0.006	88.97	56.00	78	62	0.028
	sta3	R	3	2.67	10.32	0.091	95.58	63.64	89	51	0.011
		P	3	3.32	10.34	0.117	94.69	62.57	88	52	0.019

Table A.2 Performance Metrics for RYU (R) and Pre-Install RYU (P) Controllers across Mobility Models

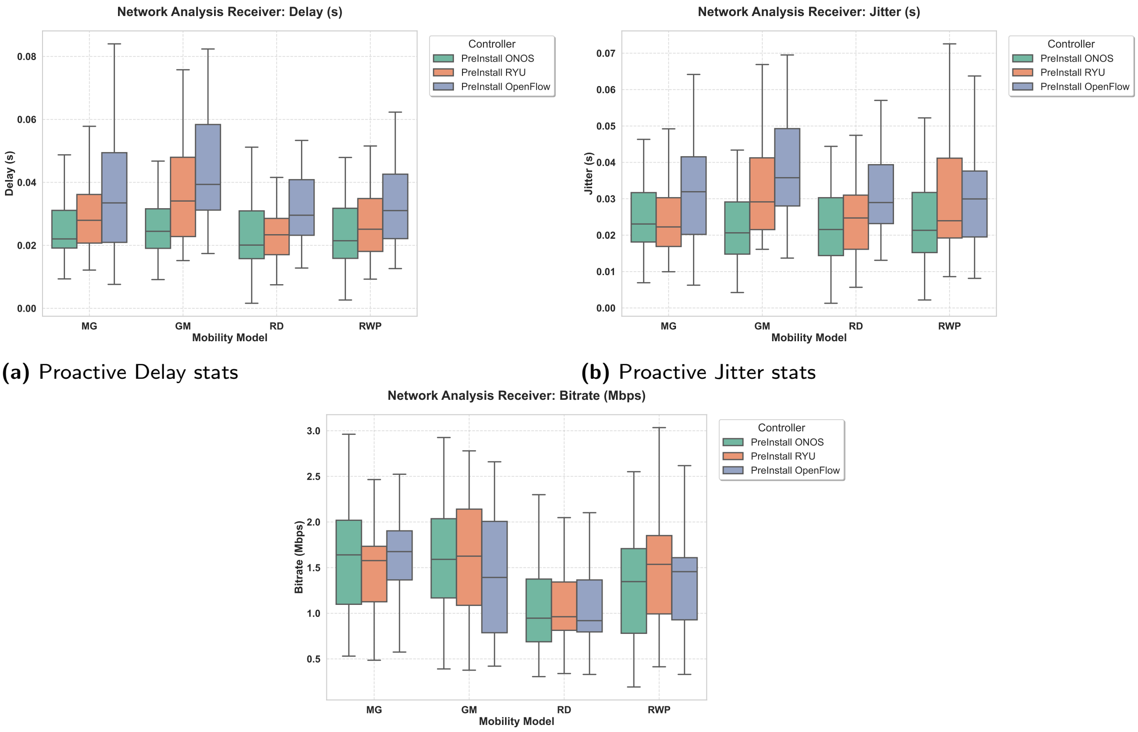


(a) Proactive Delay stats

(b) Proactive Jitter stats

(c) Proactive Bitrate stats

Figure A.1 Network Statistics of Proactove Controller Experiment C

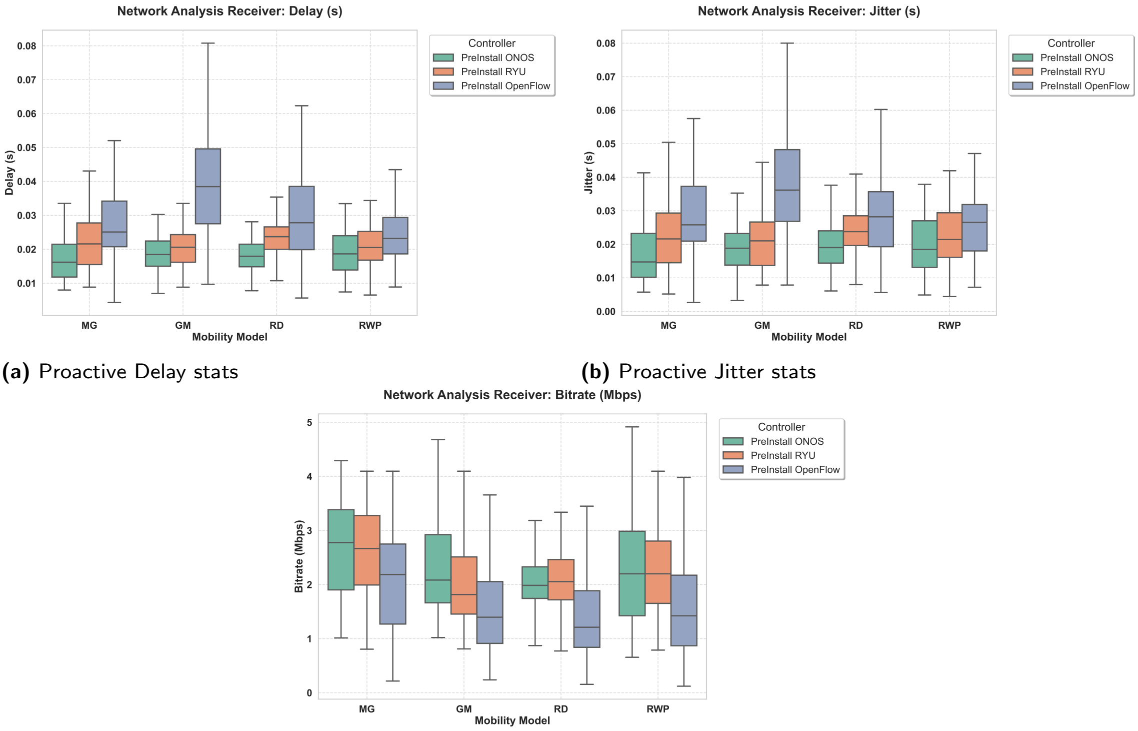


(a) Proactive Delay stats

(b) Proactive Jitter stats

(c) Proactive Bitrate stats

Figure A.2 Network Statistics of Proactove Controller Experiment D



(a) Proactive Delay stats

(b) Proactive Jitter stats

(c) Proactive Bitrate stats

Figure A.3 Network Statistics of Proactove Controller Experiment E