Xi'an Jiaotong-Liverpool University
西交利物浦大学

SCHOOL OF INTERNET OF THINGS

IOT305TC FINAL YEAR PROJECT

MPTCP-Based Link Congestion Detection and Traffic Control in Software-Defined Networks

Final Thesis

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Engineering

Student Name	:	Yikun Li
Student ID	:	1929262
Supervisor	:	Dr. Bong-Hwan Oh
Assessor	:	

Abstract

This project aims to enhance the performance of Multipath TCP (MPTCP) in congestion situations within Software-Defined Networking (SDN) environments. The main objectives are to develop a congestion detection algorithm and a traffic control mechanism based on MPTCP. The algorithm will accurately detect congestion levels and dynamically reroute congested subflows to non-congested paths for congestion balancing. The goal is to improve MPTCP's throughput and reduce latency, enhancing its performance in various network conditions. The methodology involves implementing an SDN controller using the Ryu framework and OpenFlow v1.3, along with Mininet for network simulation. The controller will handle IP forwarding, MAC-port learning, and flow table entries to facilitate packet forwarding and communication. Additionally, loop avoidance mechanisms such as the Spanning Tree Protocol will be employed to prevent broadcast storms in looped topologies. The project will evaluate the proposed solution through simulations or experiments to assess its effectiveness in improving MPTCP's performance during congestion.

Keywords: Multi-path TCP; Software-Defined Networking; Congestion Detection; Traffic Control

Acknowledgements

First of all, I would like to express my sincere gratitude to Dr. Bong-Hwan, who led me to the field of networking and gave me a lot of opportunities to understand the advanced technologies of networking such as SDN. He also provided me with many ideas for my final year project. I can say that without him, this FYP would not have been born.

Secondly, I would like to express my deep gratitude to my family and friends who have been with me all the way, who have helped me to avoid being caught up in these negative emotions when I was deeply stressed and powerless. At the same time, they have also provided me with a lot of inspiration by some coincidence.

Finally, I would like to thank the school of IoT. The school provided us with much help during our studies and also provided us with a FYP room so that we could have a regular place to work on our projects at any time while we were busy with FYP.

Contents

List of Tables

List of Figures

List of Acronyms

Term	Initial components of the term
TCP	Transmission Control Protocol
SPTCP	Single-path TCP
MPTCP	Multi-path TCP
SDN	Software-defined Networking
LIA	Linked Increases Algorithm
OLIA	Opportunistic Linked Increases Algorithm
D-OLIA	Dynamic OLIA
Balia	Balanced Linked Adaptation Algorithm
CP Scheduling	Constraint-based Proactive Scheduling
QoS	Quality of Service
dpid	Data Path ID

Please insert one term per table row; this will ensure appropriate spacing and alignment between each term and its components. It will also allow you to sort the terms alphabetically.

Chapter 1

Introduction

1.1 Motivation, Aims and Objective

Motivation

TCP/IP communication is currently restricted to a singular path per connection, despite the presence of multiple paths between peers. The utilization of these multiple paths concurrently for a TCP session can enhance resource allocation within the network, thereby improving the user experience through increased throughput and resilience against network failures. Consequently, the development of Multipath TCP (MPTCP) has addressed this limitation by enabling the simultaneous utilization of multiple paths between peers [1].

For applications that lack MPTCP awareness, MPTCP operates similarly to TCP. However, when additional paths are discovered between peers, MPTCP establishes additional sessions, known as MPTCP subflows, on these paths. These subflows, combined with the existing sessions, continue to appear as a single connection to the applications at both ends [2]. This process is depicted in Figure 1-1.

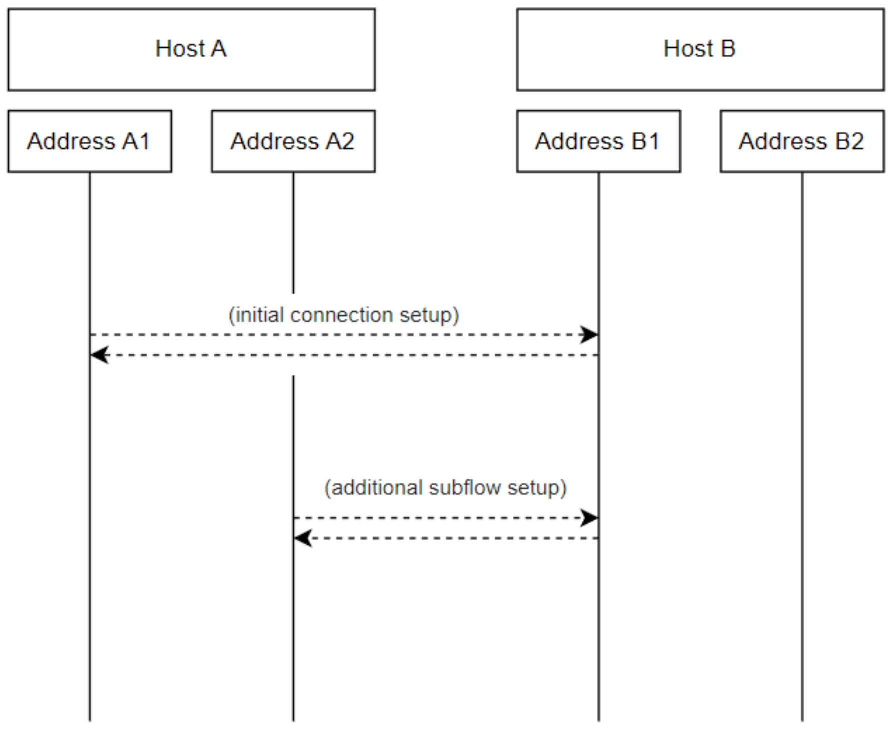

Figure 1-1: MPTCP Usage Scenario

The redundancy offered by Multipath TCP allows for the inverse multiplexing of resources, resulting in an increased TCP throughput equivalent to the aggregate capacity of all available link-level channels [3]. This stands in contrast to standard TCP, which operates on a single channel requirement. Consequently, the utilization of Multipath TCP significantly enhances network throughput and reduces latency.

This advantage proves particularly valuable in wireless network environments, such as scenarios where both Wi-Fi and cellular networks are simultaneously utilized [4]. Moreover, in addition to the achieved throughput gains facilitated by Multipath TCP, users have the flexibility to add or remove connections at their discretion without interrupting the end-to-end TCP connection.

SDN represents an innovative approach to network design, implementation, and management, which involves separating the network control (control plane) from the forwarding process (data plane) to improve user experience. This separation enables network administration and management to become simple, as the SDN architecture segregates the routing and forwarding decisions of network elements, such as routers, switches, and access points, from the data plane [5]. Meanwhile, the control plane focuses only on information related to logical network topology, traffic routing, and similar aspects. The data plane is then responsible for managing the network traffic according to the established configuration in the control plane.

The network segmentation offered by SDN yields numerous advantages in terms of network flexibility and controllability. This is primarily due to the centralized and intelligent implementation that provides enhanced visibility into the network for network management and maintenance purposes. Additionally, SDN significantly improves network control and responsiveness. In contrast, traditional network architectures lack these capabilities.

In traditional network architectures, the control plane and data plane are tightly coupled within network devices such as routers, switches, and access points. The control plane is responsible for making routing decisions and managing network protocols, while the data plane handles the forwarding of network traffic based on these decisions. However, the lack of separation between the control and data planes limits flexibility, controllability, and the ability to customize network functions.

On the other hand, SDN architecture decouples the control plane from the data plane. The control plane is centralized and logically separated from the data plane, which allows for more efficient network management. The control plane, often implemented through a controller, is responsible for managing network policies, traffic routing, and other high-level network functions. In contrast, the data plane focuses solely on forwarding network packets according to the instructions received from the control plane.

The separation of the control and data planes in SDN architecture provides several advantages over traditional network architectures. It allows for centralized network management, where administrators can easily configure and manage the network from a single point of control. This centralized control enhances network visibility, making it easier to monitor and troubleshoot network issues. Moreover, SDN enables dynamic network reconfiguration and adaptation to changing traffic patterns and requirements, leading to improved network responsiveness.

In Figure 1-2, a visual representation of the comparison between traditional network architecture and SDN architecture can be observed. It illustrates the separation of the control and data planes in SDN, highlighting the centralized control and flexibility it offers, compared to the tightly coupled nature of traditional network architectures.

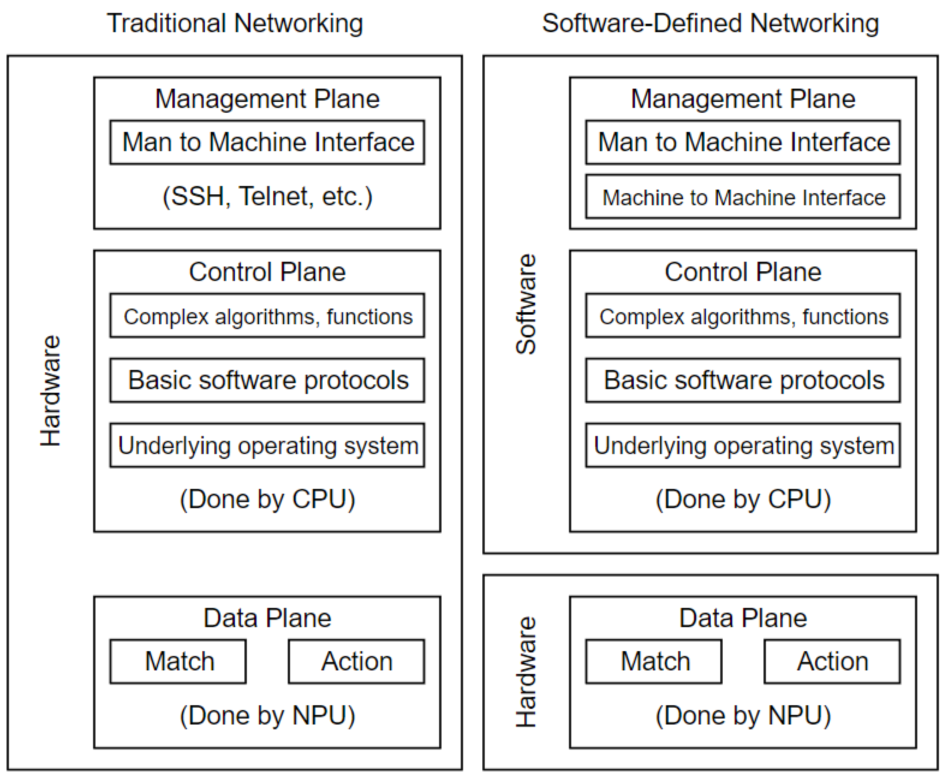

Figure 1-2: Traditional Architecture vs. SDN Architecture

MPTCP utilizes multiple paths concurrently for a single TCP connection, which can lead to a substantial number of out-of-order packets, especially when the paths have varying bandwidths and latencies [6]. In scenarios where congestion arises on a path, large disparities in bandwidth and latency between paths are common and consequential. Consequently, MPTCP may not perform as effectively as a single TCP link under such circumstances. This finding has been corroborated through experimentation involving the introduction of distinct congestion links to a path.

In Figure 1-3, it is evident that MPTCP's performance falls short of that of a single TCP link when the congestion link reaches a capacity of 99Mbit/s. This outcome highlights the potential limitations of MPTCP in situations where significant congestion occurs.

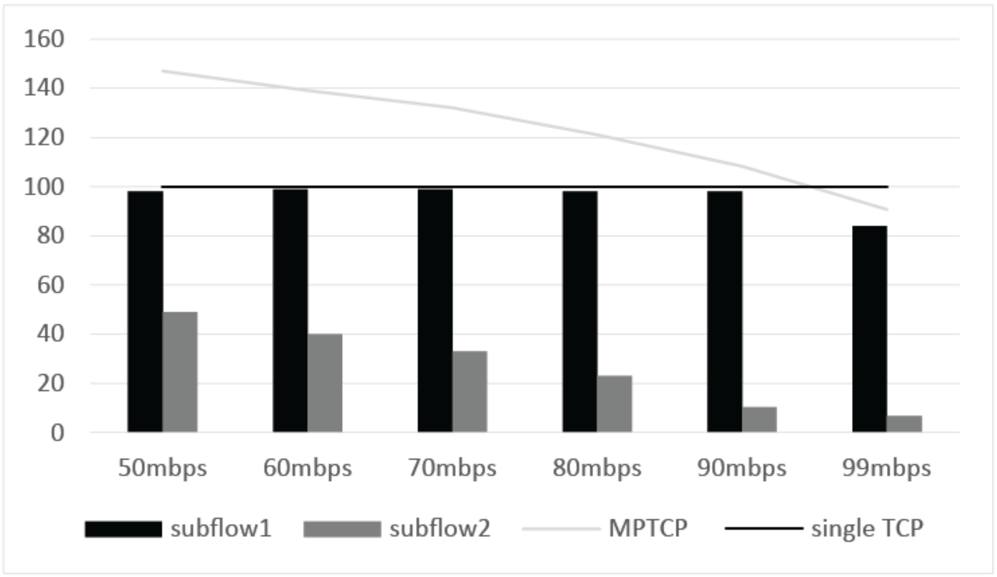

Figure 1-3: Experiment Data

Hence, it is evident that the current implementation of MPTCP not only compromises the throughput of TCP links in the network but also fails to ensure optimal performance for MPTCP itself. Thus, it is apparent that MPTCP is not Pareto optimal in its current state.

In the context of congestion control algorithms employed by MPTCP, the Internet Engineering Task Force (IETF) has outlined three primary goals [7], as shown in Table 1-1:

Goals	Description
Goal 1: Improve Throughput	A multipath flow should perform at least as well as a single path flow would on the best of the paths available to it.
Goal 2: Do no harm	A multipath flow should not take up more capacity from any of the resources shared by its different paths than if it were a single flow using only one of these paths. This guarantees it will not unduly harm other flows.
Goal 3: Balance congestion	A multipath flow should move as much traffic as possible off its most congested paths, subject to meeting the first two goals.

Table 1-1: Desirable Properties of Congestion Control Algorithm

Among these goals, only Goal 1 and Goal 2 have been completely implemented in the congestion control algorithm used by MPTCP. However, Goal 3 has not been fully satisfied in the design of the algorithm [7]. Therefore, it is highly likely that the performance degradation observed in MPTCP when a subflow experiences congestion is primarily attributed to its "Linked Increase" congestion control algorithm.

The resolution of this problem would indeed enhance MPTCP's performance during congestion, leading to increased efficiency in communication networks. Particularly in the context of Wide Area Networks (WANs), MPTCP is employed to enhance performance over long distances. By improving MPTCP's congestion control mechanism, communication between geographically dispersed devices can be rendered more reliable. This improvement has the potential to have a significant impact on technologies like the Internet of Things (IoT), which rely on larger bandwidth and lower latency for seamless operation.

Aims

In light of the mentioned congestion control algorithm, the objective of this project is to develop a congestion detection and traffic control algorithm based on MPTCP within the context of SDN. The aim is to meet Goal 3 of MPTCP's design, which pertains to balancing congestion.

The proposed algorithm will follow a two-step process. Firstly, it will detect congestion within the MPTCP subflow and assess whether it has reached a level that impairs the overall performance of the MPTCP connection. Once congestion at a significant level is identified, the algorithm will dynamically reroute the congested subflow through alternative non-congested paths to achieve congestion balancing. This approach aims to distribute the network load across available paths effectively. As a result, the overall performance degradation of the MPTCP connection will be mitigated, allowing for its recovery.

The objective of this algorithm is to enhance MPTCP's performance by refining its congestion control mechanism, aiming to achieve Pareto optimality. By optimizing MPTCP's congestion control algorithm, the algorithm seeks to increase MPTCP's throughput even in congested states while simultaneously reducing latency. This improvement will enable MPTCP to achieve a more efficient balance between these two important performance metrics, enhancing its overall performance in various network conditions, including congested states.

Objective

To achieve the goals of this project, the entire process can be divided into several key objectives:

Objective 1: Develop a congestion detection algorithm based on MPTCP:
This objective entails the design and implementation of an algorithm capable of accurately detecting link congestion within a SDN environment. The algorithm will assess congestion levels by considering factors such as throughput. It is crucial to ensure that the algorithm exhibits efficiency, scalability, and adaptability to dynamic network conditions. Furthermore, the algorithm should operate independently of the main process, thereby maintaining its autonomous functionality within the network infrastructure.

Objective 2: Design adaptive flow control mechanisms for MPTCP flows:
The proposed mechanism should possess the capability to assess the severity of congestion within a subflow and make informed decisions regarding whether detouring is necessary. In the presence of severe congestion, the mechanism should facilitate the completion of the detour process for the subflow. By accomplishing congestion balancing through detouring, the objective is to optimize overall network performance and utilization. This process aims to distribute network traffic effectively, alleviating congestion and enhancing the efficient utilization of network resources.

Objective 3: Evaluate the performance of the proposed solution:
To evaluate the effectiveness and efficiency of the developed congestion detection algorithms and traffic control mechanisms, it is essential to conduct simulations or experiments. During these simulations or experiments, subflows in congestion links should be introduced to an idle path while the network is operating under normal conditions. This scenario will allow for an assessment of whether the proposed method has improved MPTCP network performance during congestion.

By comparing relevant data, such as throughput measurements obtained before implementing the method, the impact of the congestion control mechanisms can be evaluated. The comparison will help determine whether the method has resulted in enhanced network performance under congested conditions. Through these simulations or experiments, the effectiveness and efficiency of the developed congestion detection algorithms and traffic control mechanisms can be quantitatively assessed, providing valuable insights into their performance within MPTCP networks.

1.2 Literature Review

The field of computer networking has experienced significant growth in recent years, driven by the widespread adoption of data-intensive applications and the escalating need for high-performance networks [8]. Consequently, the issue of congestion control has become increasingly critical to ensure optimal utilization of network resources and maintain a high Quality of Service (QoS). In response to this challenge, numerous congestion control algorithms have been developed, each with its specific objectives and operational mechanisms.

This literature review aims to investigate a selection of previous congestion control algorithms, providing an examination of their underlying principles, operational characteristics, and performance evaluations. By comprehensively understanding the strengths, limitations, and advancements of these algorithms, can lay the groundwork for the development of an effective congestion control mechanism within the context of our research focus: MPTCP-Based Link Congestion Detection and Traffic Control in SDN.

Through this literature review, it will explore the evolution of congestion control algorithms, highlighting key contributions and advancements in the field. By analyzing their operational principles and evaluating their performance under various scenarios, can gain valuable insights into their effectiveness in addressing congestion-related challenges. This knowledge will inform the development of our proposed congestion control mechanism, tailored specifically to the MPTCP framework within the SDN environment.

Ultimately, this literature review seeks to synthesize existing research and identify gaps in current congestion control approaches. By building upon prior knowledge and leveraging the advancements in the field, aim to contribute to the development of a robust and efficient congestion control mechanism for MPTCP networks in SDNs.

LIA

The first and fundamental congestion control algorithm used in current versions of MPTCP is the "Linked Increases" Algorithm (LIA). LIA operates by dynamically managing the congestion window size for subflows, necessitating a balance between achieving optimal congestion balancing and maintaining responsiveness. However, the current implementation of LIA faces challenges in achieving both optimal congestion balancing and responsiveness simultaneously [7]. This tradeoff prevents the full pooling of network resources, resulting in several issues.

One problem arises when upgrading TCP users to MPTCP, as it may lead to a significant decrease in the total network throughput without providing any benefits to any individual users. This characteristic renders MPTCP non-Pareto optimal [9]. Additionally, MPTCP users may exhibit overly aggressive behavior towards TCP users, further impacting network performance [10].

To address these limitations, subsequent congestion control methods have been developed to enhance or replace the LIA algorithm. These methods can be categorized into different approaches, each offering unique strategies to improve MPTCP's congestion control. These advancements aim to optimize congestion balancing, responsiveness, and resource pooling within the MPTCP framework, ultimately enhancing network performance and efficiency.

OLIA

An alternative to LIA is the Opportunistic Linked Increases Algorithm (OLIA), which utilizes a similar window-based congestion control mechanism and is considered a variant of LIA's Coupled algorithm [11]. However, OLIA takes a different design approach, aiming to achieve optimal resource pooling, congestion balancing, and responsiveness without sacrificing the congestion balancing capability for responsiveness [9].

OLIA successfully addresses some of the limitations of LIA and has been proven to be Pareto optimal, while retaining the responsiveness and non-jitter characteristics of LIA [9]. However, in the current dynamic nature of the Internet with its transient traffic patterns, OLIA often suffers from low throughput due to its inability to effectively utilize the underlying network resources [11].

To overcome this challenge, further enhancements have been proposed, such as the D-OLIA method. These improvements aim to enhance the performance of OLIA by addressing its limitations and making better use of the underlying network resources. By refining the congestion control mechanisms and adapting them to the dynamic nature of the Internet, these methods strive to optimize throughput and improve the overall performance of MPTCP networks.

Balia

The Balia algorithm offers a solution to address the non-Pareto optimality issue of LIA and the limited responsiveness of OLIA in certain network scenarios. It is also a window-based congestion control algorithm specifically designed for MPTCP [12]. The key distinction of Balia lies in its novel design framework, which provides a structural understanding of the MPTCP algorithm [13].

Balia approaches the problem by mathematically demonstrating an inherent tradeoff between TCP friendliness and responsiveness, as well as between responsiveness and window oscillations. It theoretically proves that it is impossible to maximize the performance of all three metrics simultaneously. Therefore, Balia aims to achieve acceptable levels of window oscillations to improve both TCP friendliness and responsiveness.

In contrast to the Coupled algorithm, which can exhibit instability and have multiple equilibria, the Balia algorithm is designed to determine the existence, uniqueness, and stability of network equilibria [12]. Through extensive testing, Balia has demonstrated a favorable balance between these properties, making it a promising congestion control mechanism for MPTCP networks.

By addressing the tradeoff between TCP friendliness, responsiveness, and window oscillations, Balia provides a refined approach to congestion control that enhances network performance and optimizes the behavior of MPTCP connections.

CP Scheduling

To address the performance degradation caused by a significant number of out-of-order packets resulting from congestion, an alternative approach involves improving scheduling methods to minimize the impact of packet reordering and optimize overall transmission performance within the MPTCP layer.

One such method is CP scheduling (Constraint-based proactive scheduling), which tackles the buffer blocking problem that arises when the receiver's (or sender's) buffer becomes filled with out-of-order packets [14]. CP scheduling is a proactive solution that aims to efficiently schedule packets across paths by considering factors such as buffer size and performance differences between paths.

By estimating the number of scrambled packets based on buffer size and path performance, CP scheduling determines the optimal path for packet transmission. In scenarios where there are significant performance differences between paths, MPTCP with CP scheduling achieves comparable or even better throughput than a single TCP connection on the fastest path. On the other hand, when path performances are similar, MPTCP with CP scheduling fully utilizes the capacity of all available paths [14]. This allows for efficient packet transmission, regardless of buffer size or performance discrepancies across multiple paths.

Through the use of CP scheduling, MPTCP can effectively manage packet scheduling and minimize the negative impact of congestion-induced reordering, resulting in improved overall transmission performance and efficient utilization of available network paths.

Dynamic Path Control by SDN

The advent of SDN has revolutionized network management, introducing a new approach to controlling and operating networks [15]. SDN opens up a multitude of possibilities for addressing the performance degradation issue, including the dynamic control of MPTCP paths through SDN.

In this context, SDN applications can leverage the capabilities of the SDN controller to gather estimated capacity information for each path. By analyzing this information, poor-performing links that contribute significantly to the reordering queue size at the MPTCP layer can be identified. When a path is deemed to have insufficient capacity compared to other available paths, the SDN controller can intelligently remove that path from MPTCP to prevent excessive out-of-order packet blocking [6]. Once the capacity of the path reaches a certain threshold, it can be reconnected to the MPTCP network.

This dynamic path control mechanism, enabled by the support of the SDN controller, allows MPTCP to adaptively manage subflows based on the available capacity of the connected paths. By dynamically switching between the best SPTCP (single-path TCP) and MPTCP configurations, MPTCP can maximize download rates. This approach leverages the capabilities of SDN to enhance the performance and efficiency of MPTCP in handling network congestion and reordering issues, resulting in improved overall transmission rates.

From the several methods mentioned above for solving the MPTCP performance degradation problem caused by the congestion link, the following major approaches can be seen:

Methods	Approach
LIA, OLIA, Balia	Window-Based Congestion Control
CP scheduling	Scheduling Method
Dynamic MPTCP Path Control	SDN

Table 1-2: Congestion Control Methods and its Approach

Certainly, the Window-Based Congestion Control approach has shown significant progress in addressing congestion-related issues. Its efficient mechanisms have demonstrated improved performance in managing network congestion.

Moreover, the Scheduling Method has proven to be effective, offering simple and efficient solutions to various challenges. This method presents opportunities for developing innovative approaches that can further enhance network performance and resource allocation.

By combining the benefits of Window-Based Congestion Control and the efficiency of the Scheduling Method with the SDN approach, novel and effective strategies can be developed to optimize network performance, improve resource allocation, and address congestion-related challenges.Given the advancements in SDN, this project aims to leverage the capabilities of SDN in proposing new solutions.

The SDN-based solutions proposed in this project will utilize the programmability and centralized control provided by SDN architectures. This will enable the development of intelligent algorithms and mechanisms that can dynamically adapt to changing network conditions, efficiently allocate resources, and optimize congestion control.

By harnessing the power of SDN, this project aims to contribute to the ongoing efforts in improving network performance, enhancing resource utilization, and providing efficient solutions to congestion-related issues in modern computer networks.

Indeed, the concept of diverting traffic to the least congested path is a fundamental principle in congestion control [16], [17]. This project will adopt this principle and leverage the capabilities of SDN to effectively detour MPTCP subflows that experience severe congestion. By utilizing SDN's programmability and centralized control, the project will identify the most suitable and idle path to redirect the traffic from congested subflows.

Through the use of SDN, the project will dynamically monitor network conditions and congestion levels, allowing for real-time detection of severe congestion in MPTCP subflows. Upon identifying such congestion, the SDN controller will initiate the detour process, rerouting the traffic to the most suitable path that offers the least congestion.

By implementing this approach, the project aims to alleviate congestion-related issues and improve the overall performance of MPTCP subflows. By dynamically selecting the optimal path, the project seeks to maximize network utilization, enhance throughput, and reduce latency.

The utilization of SDN in detouring congested MPTCP subflows demonstrates the potential for intelligent and adaptive congestion control mechanisms. By leveraging SDN's capabilities, the project aims to enhance the efficiency and effectiveness of congestion control in multipath communication, ultimately improving the user experience and optimizing network performance.

Chapter 2

Methodology and Results

2.1 Methodology

The project will be divided into two main parts: Congestion Detection and Traffic Control. Given the project's foundation in SDN network design, the development will involve creating an SDN controller that encompasses both of these functions. To facilitate this, the project will utilize the Ryu framework, coupled with OpenFlow v1.3, to implement the SDN controller.

To simulate the SDN network environment, Mininet will be employed. Mininet provides a platform for emulating a network topology and allows for testing and validation of the developed algorithms and mechanisms. The testing and development environment for the project will be Linux 4.19.234.mptcp, which offers the necessary support for Multipath TCP (MPTCP) functionality.

In the Congestion Detection phase, the SDN controller will employ congestion detection algorithms and techniques to identify congestion levels within the network. This will involve monitoring factors such as throughput. The goal is to accurately detect congestion and assess its severity in real-time.

In the Traffic Control phase, the SDN controller will leverage its programmability to control and manage the traffic flows within the network. This will involve dynamically detouring MPTCP subflows experiencing severe congestion to alternative paths that offer better network conditions. The SDN controller will make use of the information obtained from the Congestion Detection phase to make informed decisions regarding traffic redirection.

Throughout the project, the Linux 4.19.234.mptcp environment will provide the necessary platform for testing, development, and evaluation of the proposed congestion detection and traffic control mechanisms.

By combining the Ryu framework, OpenFlow v1.3, Mininet, and the Linux 4.19.234.mptcp environment, the project aims to develop and validate an efficient and effective SDN-based congestion control solution for MPTCP networks.

In the initial phase of the project, the focus is on establishing basic network communication within the SDN environment. As the project utilizes the Ryu framework for SDN controller development, the controller's logic and functionalities will align with the Ryu framework.

To enable basic communication in the SDN network, the first step is to ensure that IP forwarding flow rules are added to all switches in the network. This process will be automated and handled by the SDN controller, eliminating the need for manual intervention.

The SDN controller, developed using the Ryu framework, will have the capability to interact with the network switches using the OpenFlow protocol. It will dynamically program the switches by installing appropriate flow rules that enable IP forwarding. These flow rules will define how the switches should handle incoming packets and direct them to their intended destinations.

By automatically adding IP forwarding flow rules to all switches, the SDN controller establishes the foundation for network communication within the SDN environment. This allows packets to be forwarded between different hosts or devices connected to the network, ensuring connectivity and enabling subsequent stages of the project, such as congestion detection and traffic control, to be implemented effectively.

After the initial setup of the network, when the controller sends a feature request message to the switch and receives a response, it signifies the establishment of the communication between the controller and the switch. At this stage, it is important to configure the switches with appropriate flow table entries to handle different types of packets.

To ensure proper packet processing, three Table-Miss flow table entries are added to all switches in the network: Table-Miss, ARP Table-Miss, and IP Table-Miss. Each entry serves a specific purpose in managing packet forwarding and processing.

1. Table-Miss: This flow table entry has a match field of ALL, which means it will match any packet that hasn't been matched by other higher-priority flow table entries. The purpose of the Table-Miss entry is to process non-essential packets, such as control traffic or monitoring data. When a packet matches this entry, the switch will automatically drop the packet, preventing it from interfering with the normal operation of the network.

2. ARP Table-Miss: This flow table entry is used to match ARP packets that do not correspond to IP packets but are necessary for maintaining regular network communication. ARP (Address Resolution Protocol) is responsible for mapping IP addresses to MAC addresses. The ARP Table-Miss entry ensures that ARP packets are correctly processed and forwarded within the network.

3. IP Table-Miss: This flow table entry handles IP packets that have not been matched by any other flow table entry in the switch. When an IP packet is not matched by specific rules, it is sent to the IP Table-Miss entry. This entry typically directs the switch to send the packet to the controller for further processing or decision-making. It is important to assign a higher priority to the IP Table-Miss entry than the Table-Miss entry to ensure that IP packets are given priority and properly processed by the controller.

By adding these Table-Miss, ARP Table-Miss, and IP Table-Miss flow table entries to the switches, the controller ensures that non-essential packets are dropped, ARP packets are handled appropriately, and unmatched IP packets are forwarded to the controller for subsequent operations. This setup establishes a solid foundation for the SDN controller to manage and control the network effectively.

When an ofp_event.EventOFPPacketIn event occurs, it indicates that a packet has been sent to the controller for processing. The packet_in_handler function is responsible for handling these packets and performing the necessary operations. Here's an overview of the packet processing steps:

1. Recording Port: The controller first records the port from which the packet was received. This information is important for learning the MAC-port pairs of the switches.

2. Extracting MAC Addresses: The controller extracts the source and destination MAC addresses from the packet. These addresses provide essential information for forwarding the packet to its intended destination.

3. MAC-Port Pair Learning: The controller stores the MAC-port pair of the switch in its internal memory. This mapping indicates that the source MAC address of the packet corresponds to the incoming port. This process helps the controller learn the network topology and facilitate future packet forwarding.

4. Destination MAC Address Lookup: The controller checks if it knows the outgoing port corresponding to the destination MAC address of the packet. If the mapping exists, the packet can be forwarded directly to the appropriate port. Otherwise, flooding is required to find the outgoing port.

5. Flooding: In the case of flooding, the controller sends the packet to all ports except the incoming port. This allows the packet to reach all switches in the network, ensuring that the destination host receives it. Other hosts that are not the intended recipients will automatically drop the packet.

6. ACK Message and Flow Table Entry: Upon receiving the packet, the destination host sends an ACK (Acknowledgment) message in the reverse direction. This ACK packet undergoes the same processing steps as mentioned above. However, since the MAC-port pair of the destination host already exists in the switch, the packet is forwarded directly, and a flow table entry is inserted into the switch. This entry helps optimize future packet forwarding for similar packets.

7. Flow Table Entry for ARP Packets: The creation of flow table entries for ARP packets follows a similar procedure as that for IP packets. The controller handles ARP packets and ensures that appropriate flow table entries are inserted in the switch for efficient processing and forwarding.

By following these steps, the controller learns the network topology, determines the appropriate outgoing ports for packet forwarding, and establishes flow table entries to optimize packet handling in the SDN network.

After completing the aforementioned process, IP forwarding rules are established, enabling the SDN network to operate and communicate effectively. The network now possesses the fundamental functionalities required for proper network operation. Consequently, the flow table within the switch should exhibit the following structure:

Flow Rules	Priority	Actions
IP Forwarding Rules	3	Outgoing to port

ARP Forwarding Rules	3	Outgoing to port
IP Table-Miss	2	Controller
ARP Table-Miss	1	Controller
Table-Miss	0	Drop

Table 2-1: Flow Table Structure in Preparing Stage

The flow table consists of various entries, each denoting a specific match condition and the corresponding action to be executed. The "Table-Miss" entry serves as a default rule, instructing the switch to drop packets that do not match any other entries in the flow table. The "ARP Table-Miss" entry directs ARP packets to the controller for further processing, while the "IP Table-Miss" entry handles IP packets by forwarding them to the controller.

Additionally, the IP flow table contains entries mapping specific destination MAC addresses to their respective outgoing ports. These entries enable the switch to forward packets directly to their intended destinations, eliminating the need for flooding. The table may include multiple such entries, depending on the number of known destination MAC addresses in the network.

With the establishment of these IP forwarding rules, the SDN network is equipped with the necessary mechanisms to ensure efficient packet forwarding and communication throughout the network.

However, in scenarios where the network topology includes loops or rings, a phenomenon known as a "Broadcast Storm" can occur, which leads to continuous packet flooding within the network. This situation arises due to the presence of redundant paths, causing packets to circulate indefinitely and resulting in a dead loop.

The broadcast storm occurs as follows: When a broadcast or unknown unicast packet enters the network, the switch forwards it to all its connected ports except the one it arrived on. In a looped network, the forwarded packet reaches another switch, which in turn broadcasts it to all its connected ports, including the one it received from. This process continues, causing the packet to circulate repeatedly among the switches and propagate throughout the network. As a result, network resources become overwhelmed, leading to degraded performance and potential network failures.

To address the issue of broadcast storms in looped topologies, a loop avoidance mechanism is required. One widely used approach is the Spanning Tree Protocol (STP), which aims to create a loop-free topology by dynamically disabling redundant paths. STP operates by electing a root bridge and calculating the shortest path from each switch to the root. It then disables specific ports to eliminate loops while maintaining connectivity.

By deploying STP or similar loop avoidance mechanisms in the SDN network, the broadcast storm issue can be mitigated. These mechanisms ensure that only the necessary paths are active while blocking redundant paths, effectively preventing loops and maintaining a stable and efficient network environment.

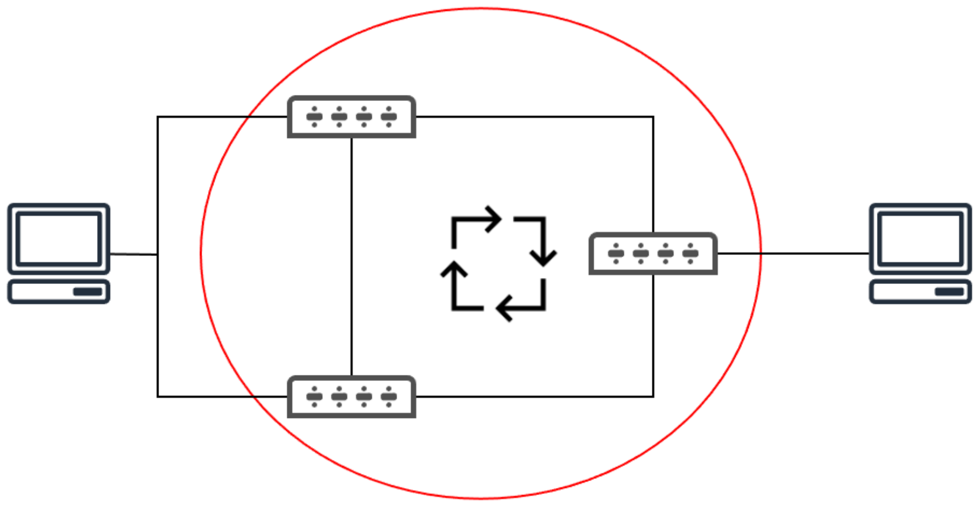

Figure 2-1: Topology Contains Loop

Hence, during each packet_in event for an incoming packet, the controller must additionally verify whether the MAC address of the host from which the packet originated exists in the MAC-port table. If a match is found, the controller compares the port on which the packet entered the switch with the port recorded in the MAC-port table. If the ports differ, the controller can deduce that the packet reached this switch after being flooded by another switch. To prevent a broadcast storm, the controller instructs the switch to discard the packet, effectively halting the flooding process at that switch location. Through this mechanism, the forwarding of packets is ensured along a single path, even in the presence of a circular topology.

By implementing this logic, the controller actively monitors and manages the network to avoid broadcast storms caused by looping topologies. It intelligently detects and terminates the flooding of packets to prevent excessive network congestion and disruptions. This approach guarantees that packets are forwarded through the network in a controlled manner, adhering to the designated paths and maintaining network stability and performance.

Once the fundamental network operations are established, the next step involves initiating congestion detection. The monitoring of congestion will primarily focus on flows, necessitating the addition of TCP forwarding flows on specific switches. In this project, the chosen switches are the edge switches, which directly connect to the hosts. Therefore, the initial task is to identify these switches and include TCP Table-Miss entries in them, preparing them to receive TCP flow rules.

By adding TCP Table-Miss entries to the edge switches, they become primed to accommodate the insertion of TCP flow rules. This step ensures that these switches are equipped to handle TCP traffic and participate in the congestion detection process effectively.

To locate the edge switches, the port method is employed, which involves utilizing three APIs provided by the Ryu framework: **get_host**, **get_switch**, and **get_link**. These APIs are utilized to gather information about all hosts, switches, and links present in the network, respectively. By obtaining topology information about all network components, the controller can identify the various parts of the network.

During this process, the controller maintains two dictionaries: **link_to_port** and **host_or_switch**. The **link_to_port** dictionary is responsible for storing information regarding both ends of a link, including the datapath ID (dpid) and port number. On the other hand, the **host_or_switch** dictionary records whether a device is directly connected to a host or a switch, specifically identifying it as an edge switch.

By comparing the dpid and port number of both ends of a link with the corresponding host information, the controller can determine whether a switch is directly connected to a host. If a match is found, the controller marks the corresponding dpid in the **host_or_switch** dictionary, indicating that the switch is an edge switch. This information is crucial for subsequent steps, such as adding TCP Table-Miss entries to the identified edge switches.

After identifying the edge switches, their datapaths can be determined using their datapath IDs (dpids). The controller can then proceed to add TCP Table-Miss entries directly to these switches. It is important to assign a higher priority to this flow rule compared to the IP forwarding rules added previously. This ensures that TCP packets will match the TCP-related flow rules first.

When a packet matches the TCP Table-Miss entry, it will be forwarded to the controller, triggering the packet_in_handler once again. This packet_in_handler will include packet filtering conditions, but these conditions will have a higher priority. Therefore, after all the flow tables have been added, the structure of the flow table in the edge switches should resemble the following:

Flow Rules	Priority	Actions
TCP Forwarding Rules	5	Outgoing to port
TCP Table-Miss	4	Controller
IP Forwarding Rules	3	Outgoing to port
ARP Forwarding Rules	3	Outgoing to port
IP Table-Miss	2	Controller
ARP Table-Miss	1	Controller
Table-Miss	0	Drop

Table 2-2: Final Flow Table Structure

With the flow table structure mentioned above, the TCP forwarding rules, including the TCP Table-Miss entry with higher priority, enable the monitoring of MPTCP packets at each location within the network. By matching the MPTCP packets with the TCP forwarding rules first, the throughput can be effectively monitored and measured at each specific location.

This flow table structure ensures that MPTCP packets are directed to the appropriate flow rules for further processing, allowing for accurate monitoring of throughput. This monitoring capability provides valuable insights into the performance and congestion levels at different points in the network, facilitating congestion detection and traffic control mechanisms.

The traffic monitoring and control approach in this project incorporates a concurrent monitoring process that runs separately from the main process. This dedicated monitoring process periodically sends OFPFlowStatsRequest requests to the switches using the datapath information. By leveraging priority filtering, the project can specifically target the flow rules of interest and retrieve traffic statistics associated with these flow rules.

The monitoring process calculates the traffic usage of a specific flow rule by subtracting the number of bits obtained during each request from the number of bits obtained in the previous request. This calculation provides valuable insights into the traffic flow through the respective flow rule over a fixed period of time. Implementing a separate monitoring process ensures that traffic information can be obtained at regular intervals without blocking the main process, allowing for continuous monitoring of flow table information.

In addition to monitoring, the project also employs a concurrent traffic control mechanism. When the monitoring process detects a link that exceeds a certain congestion threshold, the traffic control mechanism is activated. The primary objective of traffic control is to insert flow rules with the highest priority into a specific switch. These high-priority flow rules temporarily alter the flow direction, effectively detouring the subflow experiencing severe congestion.

To avoid blocking the main process, the traffic control mechanism operates at regular intervals. This concurrent approach enables ongoing monitoring of the network's congestion status without disrupting the main process. By periodically evaluating congestion levels and dynamically adjusting the flow direction through the insertion of high-priority flow rules, the project aims to alleviate congestion and optimize the overall network performance.

By integrating a concurrent traffic monitoring process and a traffic control mechanism, the project can effectively monitor and respond to congestion events in the SDN network, ensuring smooth and efficient operation.

The experiment will be conducted in a simulated environment using Mininet with the Linux 4.19.234.mptcp platform. To mimic the network infrastructure, a network topology will be created within the Mininet simulation. The simulated topology will consist of a set of interconnected switches and hosts, forming a structured network, as following:

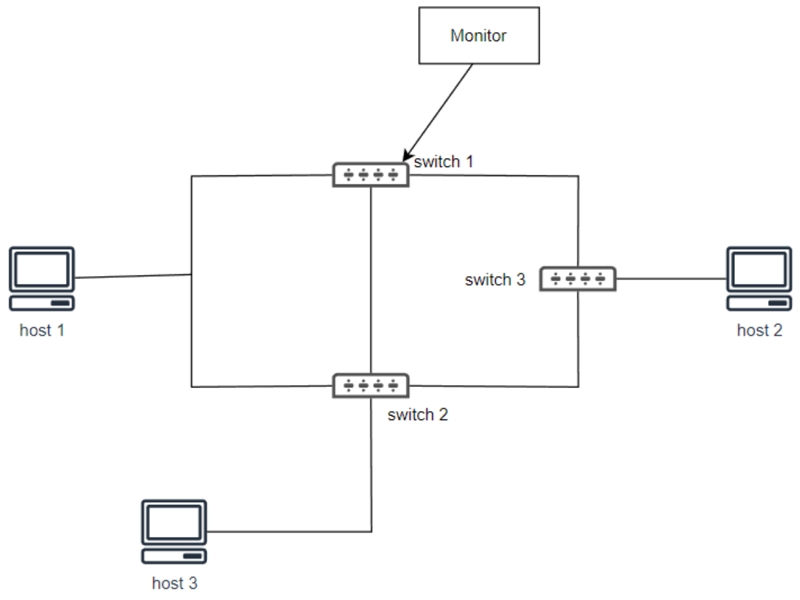

Figure 2-2: Topology of Experiment

Where host 1 will be the MPTCP client, host 2 will be the MPTCP server, and host 3 will be used to simulate the generation of congestion links. monitor will be used to monitor the traffic of the MPTCP flow rule from host 1 to host 2 in switch 1.

2.2 Results

This section contains major results achieved through the project.

At the start of the experiment, MPTCP passed through the path shown in the following diagram:

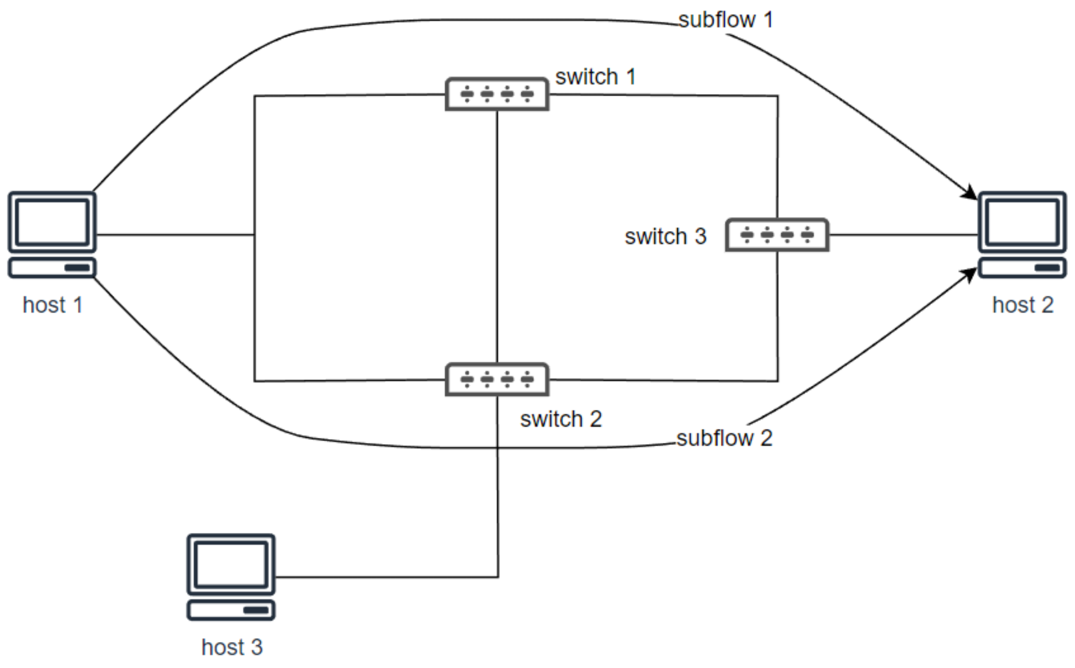

Figure 2-3: Subflow Paths Before Detour

The data collected and monitored by the monitoring process is visualized in the form of a line graph, as depicted in the figure below:

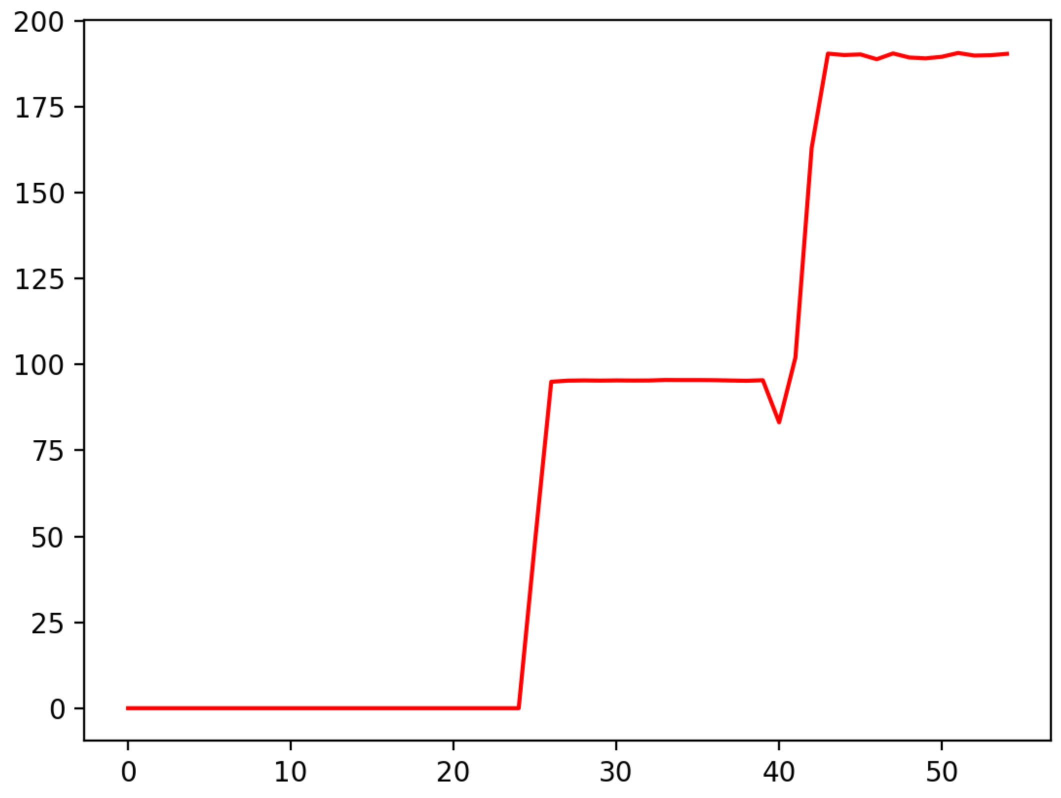

Figure 2-4: Throughput in Monitor

A notable observation can be made from the line graph, wherein a significant performance degradation is evident at a particular point in time before a subsequent substantial increase in throughput occurs. This occurrence coincides with the implementation of the detour method, providing evidence that severe congestion in one subflow adversely affects the performance of the other subflow.

Upon activating the traffic control concurrent process, new flow rules are inserted into the target switch, resulting in the diversion of subflow 2 and a modification of its path, as depicted in the accompanying figure.

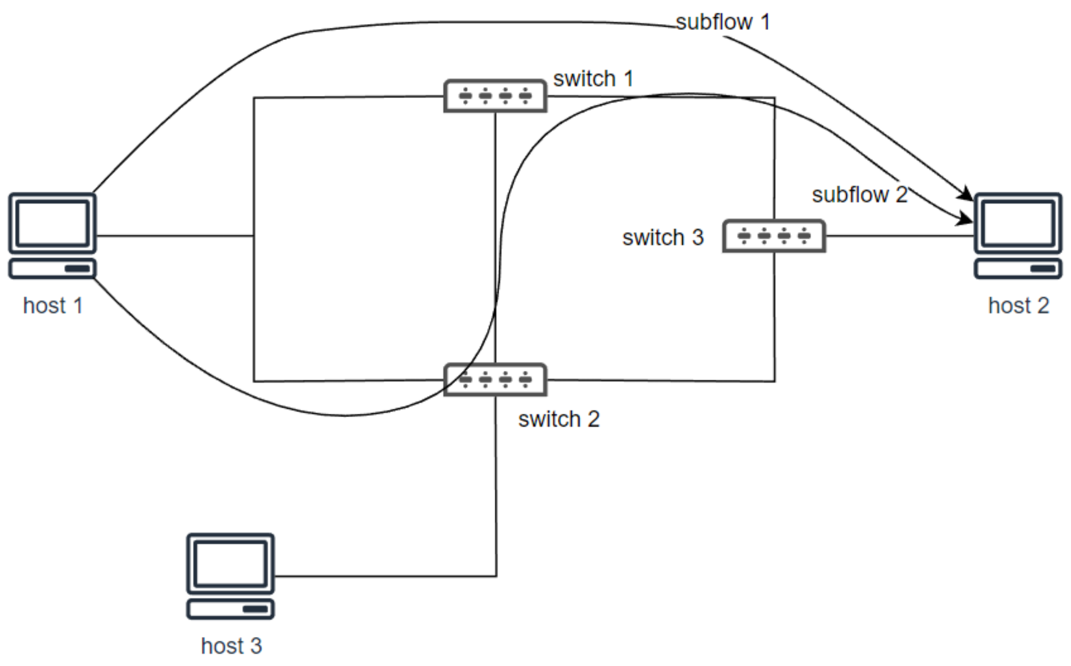

Figure 2-5: Subflow Paths After Detour

Notably, after the detour is initiated, subflow 2 coincidentally utilizes the flow rules of the path being monitored by the monitoring process. Consequently, the substantial rise in throughput reflects the cumulative throughput of both subflow 1 and subflow 2. This observation signifies the successful invocation of the detour method and the subsequent recovery of MPTCP performance.

This finding underscores the effectiveness of the detour approach in mitigating congestion and enhancing the overall performance of MPTCP-based networks. By dynamically rerouting subflows and redistributing traffic load, the detour method effectively alleviates congestion-induced performance degradation, leading to improved throughput and enhanced network performance.

Chapter 3
Conclusion and Future Work

3.1 Conclusion

In conclusion, this dissertation focused on the development of an SDN-based congestion control solution for Multipath TCP (MPTCP) networks. The project was divided into two main parts: Congestion Detection and Traffic Control. By utilizing the Ryu framework, OpenFlow v1.3, Mininet, and the Linux 4.19.234.mptcp environment, an SDN controller was created to encompass both functions.

In the Congestion Detection phase, the SDN controller employed congestion detection algorithms to monitor factors such as throughput and identify congestion levels in real-time. The Traffic Control phase leveraged the programmability of the SDN controller to manage and control traffic flows. By dynamically detouring MPTCP subflows experiencing severe congestion to alternative paths, the controller optimized network conditions.

The initial setup focused on establishing basic network communication within the SDN environment using IP forwarding flow rules. The SDN controller, developed using the Ryu framework, interacted with the switches using the OpenFlow protocol to install appropriate flow rules for packet forwarding.

To ensure proper packet processing, Table-Miss, ARP Table-Miss, and IP Table-Miss flow table entries were added to the switches. These entries handled non-essential packets, ARP packets, and unmatched IP packets, respectively. By automatically adding these entries, the SDN controller facilitated network connectivity and laid the foundation for congestion detection and traffic control mechanisms.

The project also addressed the issue of broadcast storms in looped topologies by implementing a loop avoidance mechanism such as the Spanning Tree Protocol (STP). By actively monitoring the network and instructing switches to discard flooded packets, the controller prevented excessive network congestion and disruptions, maintaining stability and performance.

The congestion detection phase involved adding TCP Table-Miss entries to edge switches and employing priority-based filtering to monitor MPTCP packets. Through a separate monitoring process, traffic statistics were periodically retrieved from flow rules, enabling accurate measurement of traffic usage and congestion levels.

In conclusion, this dissertation successfully developed an SDN-based congestion control solution for MPTCP networks. The implemented SDN controller, in conjunction with the Ryu framework, OpenFlow v1.3, Mininet, and the Linux 4.19.234.mptcp environment, demonstrated efficient congestion detection and traffic control capabilities. The project's contributions in addressing congestion and optimizing network performance provide valuable insights for future research and advancements in SDN-based networking.

3.2 Future Work

The preceding sections of this paper have presented an innovative SDN-based congestion control solution for multipath TCP networks. The developed solution leverages software-defined networking principles to detect congestion and dynamically reroute traffic across available paths, effectively mitigating congestion-related performance issues. While the proposed solution demonstrates promising results in small-scale network environments, there are several avenues for future work to enhance and extend its capabilities.

In this section, will outline the potential future directions that can be pursued to further improve the effectiveness and applicability of the SDN-based congestion control solution.

These future work areas encompass advanced congestion detection algorithms, optimization of traffic control mechanisms, integration of Quality of Service (QoS) mechanisms, evaluation in large-scale networks, security and robustness analysis, deployment in real-world environments, and standardization efforts. By addressing these areas, aims to refine the solution and contribute to the advancement of SDN-based congestion control techniques in multipath TCP networks. Through these endeavors, aim to enhance the accuracy and responsiveness of congestion detection, optimize traffic control decisions for efficient resource utilization, prioritize critical applications through QoS mechanisms, evaluate the solution's performance in large-scale networks, ensure security and robustness in the face of potential threats, validate the solution through real-world deployments, and foster industry adoption through standardization efforts.

Enhancing Congestion Detection Algorithms: The current project focuses on implementing basic congestion detection algorithms and techniques. In future work, more advanced and sophisticated congestion detection algorithms can be explored and integrated into the SDN controller. These algorithms could leverage machine learning techniques, such as anomaly detection or predictive models, to improve the accuracy and responsiveness of congestion detection in real-time.

Optimizing Traffic Control Mechanisms: The traffic control phase of the project involves detouring MPTCP subflows to alternative paths based on congestion levels. Future work can involve optimizing the traffic control mechanisms to ensure efficient utilization of network resources. This can include developing dynamic routing algorithms that consider factors such as available bandwidth, latency, and path reliability to make informed decisions on traffic redirection. Additionally, the project can explore load balancing techniques to evenly distribute traffic across available paths, further enhancing network performance.

Integration of Quality of Service (QoS) Mechanisms: To enhance the overall user experience and prioritize critical applications, future work can focus on integrating

Quality of Service (QoS) mechanisms into the SDN controller. QoS mechanisms can prioritize certain types of traffic, such as real-time communication or video streaming, over others during periods of congestion. This integration can involve assigning different levels of service based on application requirements and dynamically adjusting traffic control decisions based on QoS priorities.

Evaluation in Large-Scale Networks: The current project primarily focuses on developing and validating the SDN-based congestion control solution in a small-scale network environment using Mininet. Future work can involve scaling up the network topology and evaluating the solution in large-scale networks with a higher number of hosts, switches, and paths. This evaluation will provide valuable insights into the performance and scalability of the proposed solution and its applicability in real-world scenarios.

Security and Robustness Analysis: As SDN-based networks become more prevalent, it is crucial to assess the security and robustness of the proposed solution. Future work can involve conducting thorough security analysis, identifying potential vulnerabilities, and implementing appropriate security measures to protect the SDN controller and network infrastructure from attacks. Additionally, robustness analysis can be performed to evaluate the solution's resilience to failures, such as link or switch failures, and devise mechanisms for quick recovery and network stability.

Deployment and Real-World Testing: To validate the effectiveness of the proposed SDN-based congestion control solution, future work can focus on deploying the solution in a real-world network environment. This deployment can involve collaborating with network operators or organizations to implement the solution in their SDN-enabled networks and conducting extensive field testing. The real-world testing will provide valuable feedback and insights for further refinement and optimization of the solution.

Standardization and Industry Adoption: To ensure widespread adoption and compatibility with different SDN frameworks and equipment, future work can involve standardization efforts and engagement with relevant industry bodies. By aligning the proposed solution with industry standards, it becomes more attractive for integration into commercial SDN deployments. Collaboration with industry stakeholders can help drive adoption and facilitate the development of interoperable and scalable SDN-based congestion control solutions.

References

[1] B. Y. L. Kimura, D. C. S. F. Lima, and A. A. F. Loureiro, ‘Packet Scheduling in Multipath TCP: Fundamentals, Lessons, and Opportunities’, *IEEE Syst. J.*, vol. 15, no. 1, pp. 1445–1457, Mar. 2021, doi: 10.1109/JSYST.2020.2965471.

[2] A. Ford, C. Raiciu, M. J. Handley, and O. Bonaventure, ‘TCP Extensions for Multipath Operation with Multiple Addresses’, no. 6824. in Request for Comments. RFC Editor, Jan. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6824

[3] S. K. Nandi, ‘MPTCP Performance with Various Configurations, Path Failures and Recovery’, in *2018 3rd International Conference for Convergence in Technology (I2CT)*, Pune: IEEE, Apr. 2018, pp. 1–6. doi: 10.1109/I2CT.2018.8529504.

[4] S. R. Pokhrel and M. Mandjes, ‘Improving Multipath TCP Performance over WiFi and Cellular Networks: An Analytical Approach’, *IEEE Trans. Mob. Comput.*, vol. 18, no. 11, pp. 2562–2576, Nov. 2019, doi: 10.1109/TMC.2018.2876366.

[5] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, ‘Software-defined networking (SDN): a survey: Software-defined networking: a survey’, *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5803–5833, Dec. 2016, doi: 10.1002/sec.1737.

[6] H. Nam, D. Calin, and H. Schulzrinne, ‘Towards dynamic MPTCP Path control using SDN’, in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, South Korea: IEEE, Jun. 2016, pp. 286–294. doi: 10.1109/NETSOFT.2016.7502424.

[7] C. Raiciu, M. Handley, and D. Wischik, ‘Coupled Congestion Control for Multipath Transport Protocols’, RFC Editor, RFC6356, Oct. 2011. doi: 10.17487/rfc6356.

[8] B. Varghese and R. Buyya, ‘Next generation cloud computing: New trends and research directions’, *Future Gener. Comput. Syst.*, vol. 79, pp. 849–861, Feb. 2018, doi: 10.1016/j.future.2017.09.020.

[9] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, ‘MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution’, *IEEEACM Trans. Netw.*, vol. 21, no. 5, pp. 1651–1665, Oct. 2013, doi: 10.1109/TNET.2013.2274462.

[10] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, ‘Design, implementation and evaluation of congestion control for multipath TCP’.

[11] T. Lubna, I. Mahmud, G.-H. Kim, and Y.-Z. Cho, ‘D-OLIA: A Hybrid MPTCP Congestion Control Algorithm with Network Delay Estimation’, *Sensors*, vol. 21, no. 17, p. 5764, Aug. 2021, doi: 10.3390/s21175764.

[12] Q. Peng, A. Walid, J. Hwang, and S. H. Low, ‘Multipath TCP: Analysis, Design, and Implementation’, *IEEEACM Trans. Netw.*, vol. 24, no. 1, pp. 596–609, Feb. 2016, doi: 10.1109/TNET.2014.2379698.

[13] A. Walid, Q. Peng, J. Hwang, and S. H. Low, ‘Balanced Linked Adaptation Congestion Control Algorithm for MPTCP’, Internet Engineering Task Force, Internet-Draft draft-walid-mptcp-congestion-control-04, Jan. 2016. [Online]. Available: https://datatracker.ietf.org/doc/draft-walid-mptcp-congestion-control/04/

[14] B.-H. Oh and J. Lee, ‘Constraint-based proactive scheduling for MPTCP in wireless networks’, *Comput. Netw.*, vol. 91, pp. 548–563, Nov. 2015, doi: 10.1016/j.comnet.2015.09.002.

[15] S. Badotra and S. N. Panda, ‘Software-Defined Networking: A Novel Approach to Networks’, in *Handbook of Computer Networks and Cyber Security*, B. B. Gupta, G. M. Perez, D. P. Agrawal, and D. Gupta, Eds., Cham: Springer International Publishing, 2020, pp. 313–339. doi: 10.1007/978-3-030-22277-2_13.

[16] F. Kelly and T. Voice, ‘Stability of end-to-end algorithms for joint routing and rate control’, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 5–12, Apr. 2005, doi: 10.1145/1064413.1064415.

[17] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, ‘Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet’, *IEEEACM Trans. Netw.*, vol. 14, no. 6, pp. 1260–1271, Dec. 2006, doi: 10.1109/TNET.2006.886738.