Birzeit University

Master Thesis

Improving Multipath TCP Performance in SDN Networks

Ahmad Abu Mohsen
Department of Electrical and Computer Engineering

Supervisor: Dr. Abdalkarim Awad
Co-supervisor: Dr. Mohammad Jubran

This Master Thesis is prepared as part fulfillment of the degree requirements for the Joint Master in Electrical Engineering, JMEE Program.

February 9, 2021

MASTER THESIS

Improving Multipath TCP Performance in SDN Networks

Birzeit University

By

Ahmad Abu Mohsen

This thesis was successfully defended On February 9, 2021

Supervisor: Dr. Abdalkarim Awad

Co-supervisor: Dr. Mohammad Jubran

Examiners:

Dr. Ahmad Alsadeh

Dr. Iyad Tumar

Declaration of Authorship

I, Ahmad Abu Mohsen, declare that this thesis titled,"Improving Multipath TCP Performance In SDN Networks" and the work presented in it are my own. I confirm that:

- **This work was done wholly or mainly while in candidature for a research degree at Birzeit University.**
- **Where any part of this thesis has previously been submitted for a degree or any other qualification at Birzeit University or any other institution, this has been clearly stated.**
- **Where I have consulted the published work of others, this is always clearly attributed.**
- **Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.**
- **I have acknowledged all main sources of help.**
- **Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.**

Signed: _____

Date: _____

Abstract

Nowadays, the end hosts are equipped with multiple interfaces such as smartphones and the data center servers, where the smartphone can access the Internet through either WiFi interface or 3G/4G interface but not both simultaneously. Multipath TCP (MPTCP) is a promising technology that allows the devices that have more than an interface to utilize them at the same time. One of the recent achievements of the Internet Engineering Task Force (IETF) is MPTCP. It proposed an extension of regular TCP and its major purposes are offering better performance, throughput, and resilience to failures by dividing the flow into multiple subflows which will be then sent over many paths. The management of traditional networks is complicated. This decreases the development of the networking infrastructure where network operators need to configure each network device individually. Software-defined networking (SDN) decouples the control plane (CP) from the data plane (DP) and shift network devices to be just simple forwarding devices. Moreover, it shifts the control logic to a logically centralized controller. The controller guides the data plane components through an application programming interface (API), as well as maintains a global view of the network.

MPTCP has a positive impact on long flows in terms of throughput like transferring data with big sizes. However, MPTCP may degrade the efficiency of short flows which are sensitive to latency as web transfer applications. Moreover, MPTCP is an end-to-end protocol that cannot observe the state of lower layers in the network. This thesis investigates the impact of MPTCP on long flows and short flows in homogenous and heterogeneous networks, and it shows that a bandwidth gap between paths is a critical factor influencing the performance of the short flows and must be considered when routing the MPTCP subflows. To address the problems introduced above, this thesis propose a new architecture supported by SDN to improve the performance of MPTCP for short flows. The proposed architecture includes two modules: The topology module and the forwarding module, where the function of the topology module is to consider the bandwidth of all disjoint paths between hosts. While in forwarding module the best disjoint paths are selected based on the least bandwidth gap between paths. Our performance results showed an improvement of MPTCP performance for short flows compared to the disjoint approach.

المستخلص

في الوقت الحاضر، أجهزة المضيفين النهائيين مجهزة بواجهات متعددة مثل الهواتف الذكية وخوادم مركز البيانات، حيث يمكن للهاتف الذكي الاتصال بالانترنت من خلال واجهة الواي فاي او من خلال واجهة الجيل الثالث اوالرابع ولكن ليس كلاهما في وقت واحد. يعد البروتوكل الناقل متعدد المسارات تقنية واعدة تتيح للأجهزة التي لديها أكثر من واجهة استخدامهم في نفس الوقت، كما يعتبرالبروتوكل الناقل متعدد المسارات أحد الإنجازات الأخيرة لفريق عمل هندسة الإنترنت IETF ، حيث تم اقترحه كامتداد لـلبروتوكول الناقل العادي TCP وتتمثل اهدافه الرئيسية في تقدم أداءً أفضل، وإنتاجية أفضل، وتقليل انهيار الشبكة وذلك من خلال تقسيم التدفق إلى تدفقات فرعية متعددة ليتم إرسالها بعد ذلك عبر العديد من المسارات. تعد إدارة الشبكات التقليدية معقدة والذي يؤدي الى تقلل تطورالبنية التحتية للشبكات حيث يحتاج مشغلوا الشبكة إلى تكوين كل جهاز شبكة على حدة. الشبكات المعرفة بالبرمجيات SDN تفصل مستوى التحكم CP عن مستوى البيانات DP وتحول أجهزة الشبكة الى أجهزة بسيطة تقوم بإعادة توجيه البيانات، اضافة الى ذلك نقلت منطق التحكم إلى وحدة تحكم مركزية منطقيًا، تقوم وحدة التحكم بتوجيه أجهزة مستوى البيانات من خلال واجهة برمجة التطبيقات API والحفاظ على رؤية للشبكة.

البروتوكل الناقل متعدد المسارات له تأثير إيجابي على التدفقات الطويلة مثل نقل البيانات ذات الأحجام الكبيرة وذلك من حيث الإنتاجية، ولكن قد يقلل من كفاءة التدفقات القصيرة مثل تطبيقات الويب والتي تعتبر حساسة للكمون، اضافة الى ان البروتوكل الناقل متعدد المسارات هو بروتوكول طرف إلى طرف حيث لا يمكنه مراقبة حالة الطبقات الدنيا في الشبكة. بحثت هذه الأطروحة في تأثير البروتوكل الناقل متعدد المسارات على التدفقات الطويلة والتدفقات القصيرة في الشبكات المتجانسة وغير المتجانسة، وأثبتت أن فجوة النطاق الترددي بين المسارات هو عامل مهم يؤثر على أداء التدفقات القصيرة ويجب مراعاته عند توجيه التدفقات الفرعية الخاصة في البروتوكل الناقل متعدد المسارات. لمعالجة المشاكل التي تم ذكرها أعلاه، اقترحت هذه الأطروحة

بنية جديدة مدعومة من قبل الشبكات المعرفة بالبرمجيات بهدف تحسين أداء البروتوكل الناقل متعدد المسارات على التدفقات القصيرة. تتكون البنية المقترحة من وحدتين: وحدة الطبولوجيا ووحدة التوجيه حيث ان وظيفة وحدة الطبولوجيا هي حساب جميع المسارات المنفصلة بين المضيفين وحساب النطاق الترددي للمسارات ويكمن دور وحدة التوجيه هي اختيار أفضل المسارات المنفصلة بناءً على أقل فجوة في عرض النطاق الترددي بين المسارات. أظهرت نتائج نهجنا تحسين أداء البروتوكل الناقل متعدد المسارات على التدفقات القصيرة بالمقارنة مع الطريقة المنفصلة
.

Acknowledgements

For everyone who contributed to the success of this work, My supervisors Dr. Abdalkarim Awad and Dr. Mohammad Jubran, my mother, my father, brothers, and sisters, examining committee members, my dear friends, to Oqab, With sincere love and appreciation.

ahmad mohsen
February 9, 2021

Contents

List of Figures

List of Tables

List of Abbreviations

3G Third Generation

4G Fourth Generation

AMTCP Adaptive Multipath Transmission Control Protocol

API Application Programming Interface

BALIA Balanced Linked Adaptation Algorithm

BW Bandwidth

CP Control Plane

CWND Congestion Window

DMPTCP Dynamic Multipath TCP

DP Data Plane

HMAC Hash-based Message Authentication Code

HTTP Hypertext Transfer Protocol

IaaS Infrastructure as a Service

IETF Internet Engineering Task Force

IT Information Technology

LIA Linked Increases Algorithm

LLDP Link Layer Discovery Protocol

MMPTCP Maximum MultiPath TCP

MP Management Plane

MPTCP Multipath TCP

MPTCP-SF Multipath TCP for short flow

NOS Network Operation System

OF OpenFlow

OLIA Opportunistic Linked-Increases Algorithm

QOE Quality of Experience

QOS Quality of Service

RTO Retransmission Timeout

SDN SOFTWARE-Defined Network

SR Segment Routing

TCP Transmission Control Protocol

WiFi Wireless Fidelity

WVegas Weighted Vegas algorithm

Chapter 1

Introduction

Contents

1.1 General Overview

With the advancements in manufacturing technology, there is an increase in the number of Wi-Fi and mobile network interfaces installed in end-host devices (laptops, PCs, tablets, mobiles). For example, a laptop has more than one interface (Wi-Fi and Ethernet interfaces) through which it can connect to the Internet. Another example is smartphones, they are equipped with more than one interface (Wi-Fi interface and 3G, 4G, and 5G cellular network interfaces) to connect to the Internet as shown in Figure 1.1.

But, a user who has a device that is equipped with more than one interface can use only one technology to access the Internet; Wi-Fi or a mobile network (3G,4G). And so, the Internet connection at this device will be constraints to throughput and load balancing. Accordingly, there is a need for a technique that allows utilizing multipath where the data can be sent through more than one interface to increase throughput and quality of services [6]. So a user who has a smartphone can utilize more than interface at the same time like Wi-Fi and 3G to access the Internet. Likewise, the servers that exist in data centers are boosted with more than one interface, thus any two servers can communicate with each other through more than a path to increase the redundancy. Figure 1.2 shows a data center topology named fat tree. It shows also the servers/hosts connect and communicate with each other through more than a single path.

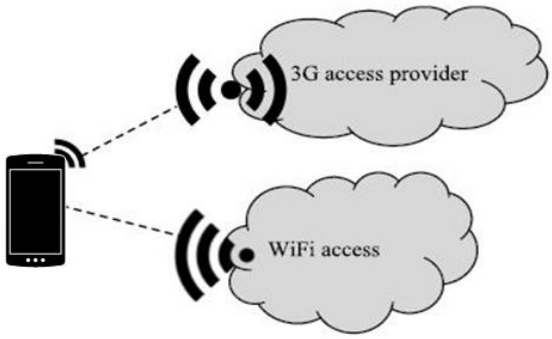


Figure 1.1: Todays smartphones can access the Internet via 3G or WiFi.

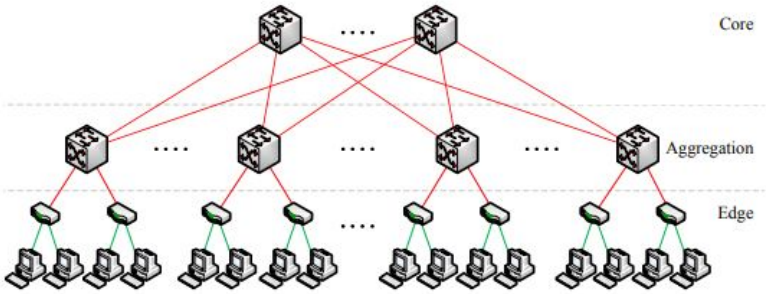


Figure 1.2: fat tree topology [1]

The demand for dedicated heavy computing resource is increasing rapidly, this leads to the development of data-centers that carry out distributed services like massive data processing, analysis, and storage. Most of the current IT systems and applications are massively relying on highly effective data centers. In modern data centers, large flows are generated by computation intensive applications and data exchange. These applications require high throughput as well as eliminating degradation in performance due to bottlenecks found in the network [7].

Multipath TCP (MPTCP) is one of the accomplishments of the Internet Engineering Task Force (IETF) [4]. MPTCP is a collection of extensions to allow TCP to offer a Multipath TCP service, which enables a transport connection to work through multiple paths together, the major point of MPTCP is splitting of flow into subflows to be sent over many paths. The increase in throughput and the network robustness against network failure are considered the main goals for the MPTCP [3].

Several large companies and researchers have been attracted to use Software-defined network (SDN), where the architecture of SDN depends on the separation of the control and data plane. SDN separates networks control logic (CP) from the forwarding process (DP). Furthermore, with the disconnection between CP and DP, the network switches will work as normal forwarding devices that receive the instructions from CP. A con-

troller in the Software-defined network maintains a global view of the network and it can be used to monitor the network state and periodically replaces congested flow paths with less congested ones for data transfer. SDN supports data center networks to solve a lot of problems like live network migration, update network management, and QoS support [2].

The latency is considered one of the critical metric for live applications as video streaming and web transfers [8]. In this light, when there is a loss in data, there will be retransmission for lost data. Therefore the users experience will be influenced harmfully because the data delivery not completed before it is deadlines [9].

MPTCP has been proposed to increase the throughput of the network by pooling bandwidth where the MPTCP divides the flow into subflows to be transmitted over multiple paths. MPTCP will be more efficient for applications that are sensitive to the throughput like transferring data with big sizes (Long Flow). However, MPTCP has a negative impact on short flows when sent over heterogeneous paths for the reasons shown below [10]:

- Paths heterogeneity in a network produces an increase in delays and causes packet reordering.
- MPTCP cannot recover the lost packet through fast retransmission, this will lead to increasing the number of retransmission Timeout (RTO).

1.2 Motivation

We are interested in pursuing research about this topic for the following reasons:

- The number of devices that have multiple interfaces is in a growing manner [11], like Smartphones, data center servers.
- There is a need for a technique that allows utilizing multiple paths to increase available bandwidth as well as redundancy against any network failure.
- The main feature of SDN technology is the ability to have a global view of the network.

1.3 Problem Statement

This thesis discusses two major problems:

- MPTCP is more effective for applications that are sensitive to the throughput like transferring data with big sizes (Long Flows), but how does the MPTCP affect the short flows when they sent over homogenous paths and heterogeneous paths in terms of bandwidth?
- MPTCP is an end-to-end protocol that cannot observe the state of lower layers in the network. For instance, different subflows of the same connection could be

distributed to the same route, a disjoint method [7] has been proposed to solve the mentioned problem to enhance the MPTCP performance for long flows in terms of throughput, but the disjoint method ignored the effect of MPTCP performance on short flows especially when those flows are distributed over heterogeneous routes in terms of paths bandwidth, so how we can improve the disjoint method to enhance the performance of MPTCP on short flows?

1.4 Thesis Contribution

We proposed an architecture supported by SDN (MPSSHetN) that includes two modules: the topology module and the forwarding module to improve MPTCP efficiency on short flows, where the proposed architecture considers the gap in terms of bandwidth between the disjoint paths when routing the subflows of MPTCP, as well as we implemented the proposed architecture with its algorithms using the Mininet [12] emulator, MPTCP Linux kernel [13], and the POX [14] as an SDN controller.

1.5 Thesis Objectives

The objectives of this thesis are illustrated below:

- This work studies the impact of MPTCP on both long and short flows in many scenarios.
- This work proves that dissimilarity between paths in terms of bandwidth is a significant factor influencing the performance of MPTCP on short flows.

1.6 Thesis Organization

This thesis is structured as follows.

Chapter 2 discusses the Software-Defined Networking and Multipath TCP.

Chapter 3 presents related work, discusses and compares between the researchers' works.

Chapter 4 experiments to evaluate the impact of MPTCP on both types of flows.

Chapter 5 discusses the proposed architecture and its modules as well as the function for each module. Moreover, presents the emulation of the proposed architecture and discusses the results of the experiments.

Chapter 6 summarises the thesis and research contributions and makes suggestions for further research and development.

Chapter 2

Background

Contents

This chapter explains the mechanism of the work of traditional networks and the definition of SDN. It also discusses the main component of SDN and the job of the control, data, and management planes. Moreover, it presents the benefits of the SDN controller and the advantages of the separation between the control and data plane. It also presents the architecture of SDN and SDN applications. The comparison between SDN and conventional networks is also discussed in this chapter. Moreover, This chapter studies MPTCP protocol, the main goals for MPTCP, MPTCP idea, and compatibility aims as well as the mechanism of MPTCP working.

2.1 Software-Defined Networking

In traditional networks, network operators configure each network device individually, but the manual configuration is time-consuming and increases the probability of error. The control plane determines how to deal with network traffic, the data plane sends traffic depending on the orders made by the control plane [2], these planes are coupled in network devices thus decreasing the development of the networking infrastructure.
Based on the original definition, [15], SDN is known as network architecture where the forwarding state in the data plane is controlled by a remotely controlled plane decoupled from the former. SDN separates the networks control logic (CP) from the forwarding process (DP). Furthermore, with the disconnection between CP and DP, the network switches will work as normal forwarding devices that receive the instructions from CP. The centralized controller (CP) is network operating system that can be installed in a physical or virtual server as shown in Figure 2.1.

The disconnection between CP and the DP can be achieved by using a well-known programming interface between the switches and the SDN controller. The controller sends the instructions to the data plane through an application programming interface (API), as shown in Figure 2.1 above. The Openflow is an example of API, the instructions are rules that will be installed on an OpenFlow-enabled switch to perform certain actions as dropping, forwarding, modifying, etc. The separation between control plane and data plane creates new abstractions in networking, facilitating network management and network development and creation. Based on functionality, the conventional network can be split into three planes: DP, CP, and MP as shown in Figure 2.2, where the DP (networking devices) responsible for forwarding data. CP employed to build the forwarding tables of the data plane components. The function of the last is to remotely check and configure the control functionality. After defining the network policy in the last plane, the policy will be enforced by the control plane, and the data plane performs it by forwarding data accordingly.

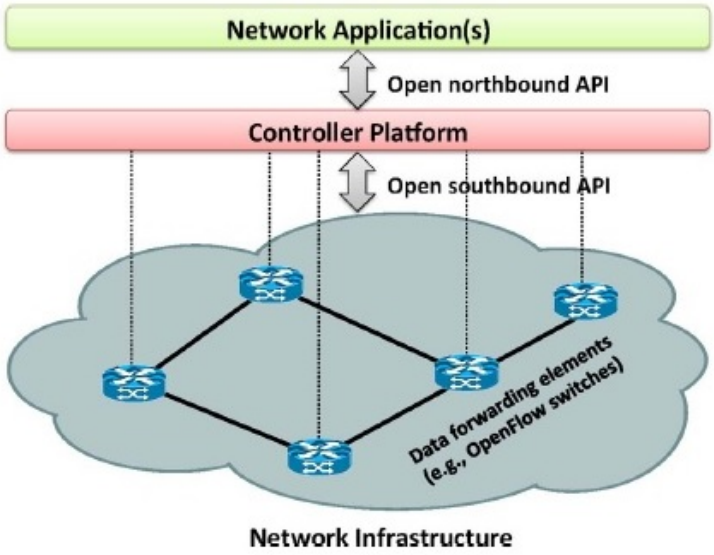


Figure 2.1: Simplified seeing of an SDN components. [2]

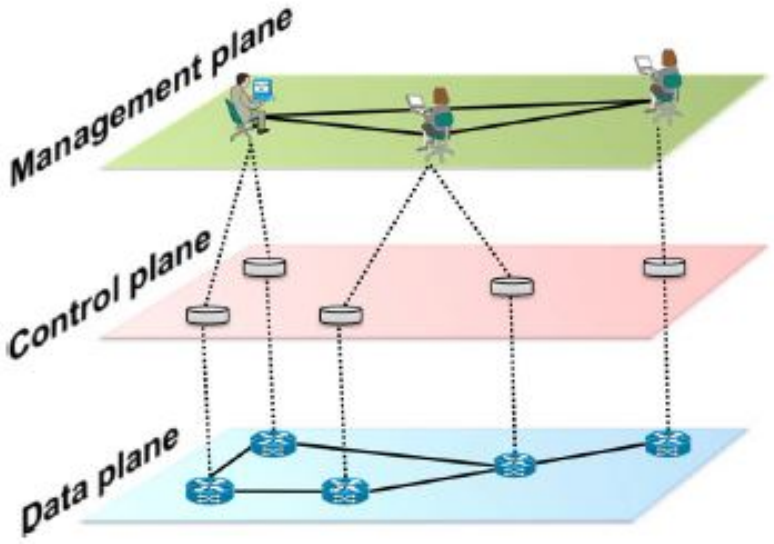


Figure 2.2: Layered view of networking functionality. [2]

There are many benefits to the controller. First, modifying network policies is simple that can be done using high-level languages and software components compared to vendor configurations. Second, centralization facilitates the progress of advanced networking functions, services, and applications since the controller has a global view of the network state. As mention above, in conventional networks the CP and DP are strongly jointed

and this coupling made the development and deployment of new networking feature more hard and complex, as well as the management of network is not efficient since if we want to add new network policy this policy must be configured on all CP of network devices individually and it may need to install new firmware and hardware upgrades. In this light, the traditional network is not cost effective, specialized, and hard-to-configure. But, in SDN the control plane is separated from the network devices and the control plane (controller) becomes an external entity. Figure 2.3 shows the differences between the traditional networking and SDN, where middleboxes are firewalls, and intrusion detection systems (IDSs), etc.

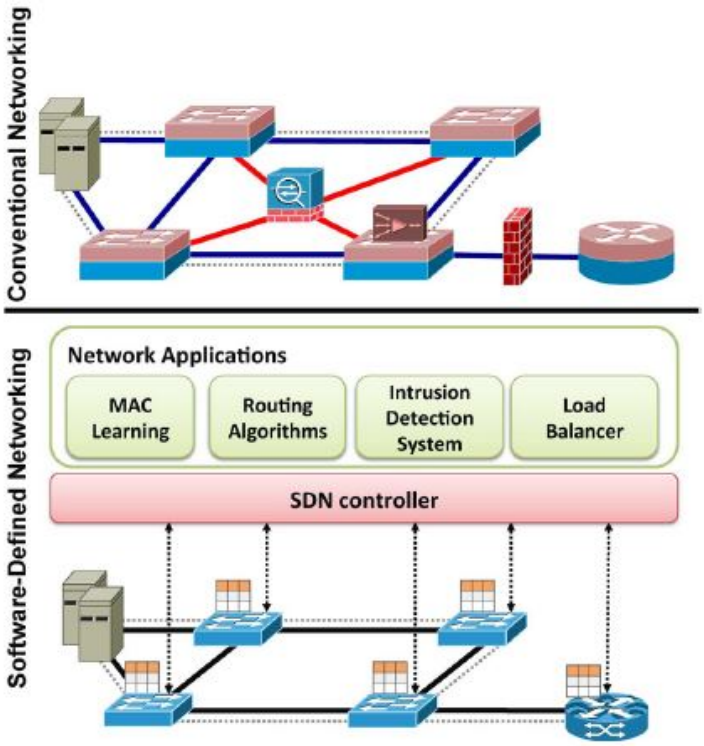


Figure 2.3: Traditional networking versus SDN. [2]

.

The separation between the control and data plane has several advantages:

- The policy decisions will be consistent and effective. Since all applications will apply based on network status.
- The applications programming become easy because of the abstractions offered by the control platform.
- Straightforward integration between the different applications, for example, routing and load balancing applications can be executed sequentially based on their priorities (the application which has the highest priority will be executed first).

2.2 SOFTWARE-Defined Networks Architecture:

An SDN architecture can be viewed as a set of different layers and each layer is responsible for certain functions as shown in Figure 2.4. The SDN can be seen from three perspectives (planes, layers, and system design architecture) as shown in Figure 2.4 below. This section will illustrate the most important layers in SDN architecture.

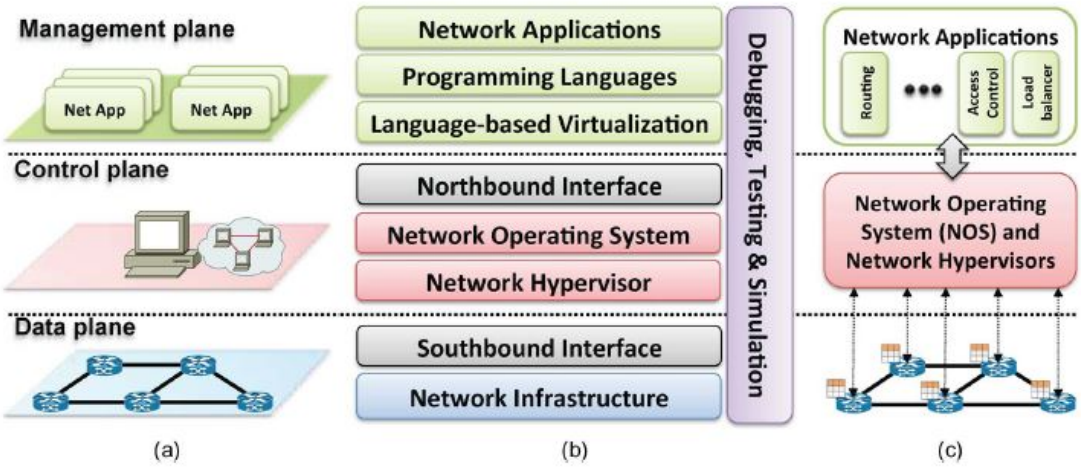


Figure 2.4: Software-Defined Networks in (a) planes, (b) layers, and (c) system design architecture. [2]

2.2.1 Infrastructure

The SDN infrastructure is a collection of network devices like routers, switches, and middlebox appliances, and those devices are similar to a conventional network. But, the major difference is the networking devices in SDN infrastructure are simple devices without included control or software to take self-actions. The brain of network devices is isolated from the data plane (DP) and it is shifted to a centralized controller (Network operating system and applications as shown in Figure 2.4).

These new networks are created on top of open and standard interfaces as OpenFlow, the main function of the open interface is to configure the heterogeneous forwarding devices dynamically. The controller and the forwarding devices are the two major elements in SDN architecture as depicted in Figure 2.4, the DP device is a hardware or software element that forward the traffic based on the rules that are installed on it from the controller, while a controller is a software stack runs on physical hardware or virtual machine [2]. In an OpenFlow switch, the flow table entry has three parts:

1. a matching rule.
2. actions to be applied on matching packets.

3. counters that keep statistics of matching Packets.

In an OpenFlow device, through a series of flow tables a path explains how packets should be treated. When Openflow device receives new packets, it first searches on the tables and match them. Based on Figure 2.5, the flow rules are series of several matching fields and the common rule on DP is to transmit the packets to CP. However, if the default rule not installed on DP, the DP will ignore the packets, as well as each rule has a priority which follows the series number of tables.

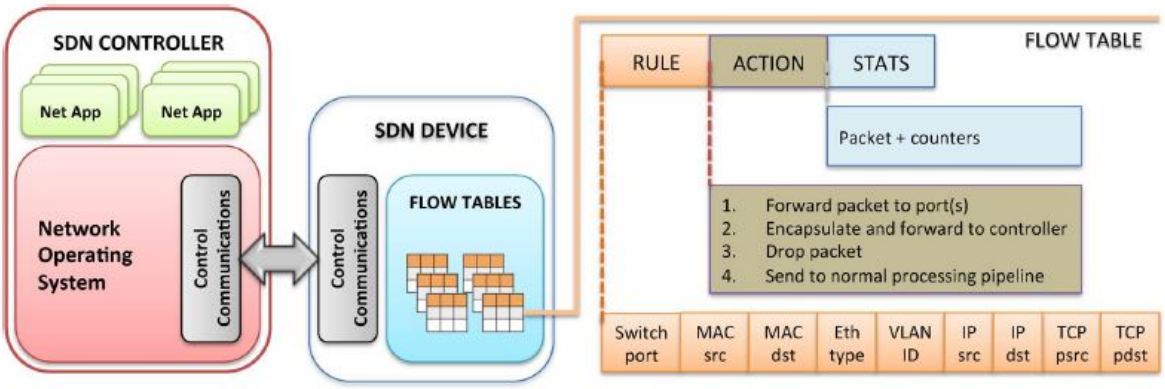


Figure 2.5: OpenFlow-enabled SDN devices. [2]

There are possible actions that involve the following:

1. Forward the packet to outgoing port(s).
2. Encapsulate it and forward it to the controller.
3. Drop it.
4. Send it to the normal processing pipeline.
5. Send it to the next flow table or to special tables.

The OpenFlow (OF) has been developed and now there are many version of it (V 1.0, V1.1, V1.2, V1.3, V1.4) and the OF is upgraded to support several match fields involving Ethernet, IPv4/v6, MPLS, TCP/UDP, etc.

2.2.2 Southbound Interfaces

Southbound interfaces (APIs) are bridges that connect control and forwarding devices, and through this layer, the CP is decoupled from the DP. OpenFlow protocol is widely deployed at SDN environment. It offers a communication channel between CP and DP. There are three information sources that are offered to Network Operation System (NOS) through Open Flow protocol:

1. Forwarding elements send an event based messages to the controller when a link status is changed.

2. Forwarding elements transmit packet-in messages to a controller when they received a new flow.

3. Flow statistics which are created by forwarding devices.

Where these messages are considered an important means to give the NOSs a flow information.

2.2.3 Network Hypervisors

Virtualization technology is widespread in the computer environment and the number of virtual servers is more than the number of physical servers. The hardware resources sharing feature is offered to the virtual machines through the Hypervisors, where each user in cloud infrastructure as a service (IaaS) can assign resources to his virtual machines on-demand from the shared resources. The machines migration feature is considered one of the important feature that provided through virtualization technology, where it provides the ability to transfer the machines from one physical server to another and thus providing flexible management. Based on the virtualization advantages, many commercial companies like VMWare and IBM started to propose a platform for SDN virtualization.

2.2.4 Network Operating Systems/Controllers

Facilitation of solving networking problems and network management are the aims of SDN that are promised and this will be done by using a logically centralized control provided by a NOS. NOS can provide many general network services like network status, network topology information, allocate the configuration for distributed devices. The controller is a crucial device in the SDN architecture, where it is the core part of the control logic that is responsible to create the network configuration according to the policies defined by the network engineers.
The controllers can be classified according to an architectural point of view, where they may be centralized or distributed. A centralized controller is a unique device that controls all forwarding elements. But, the main problems of a single controller that it forms a single point of failure in addition to that it is not quite sufficient to control a network that has a big number of DP devices. On other hands, the main advantage of distributed controllers is fault tolerance. That means if one node became down, another point must carry the responsibilities of the failed point. Figure 2.6 illustrates the SDN control platforms, SDN control platforms are divided into three layers: 1- the application and services 2- the core controller functions 3- the elements for southbound communications.

2.2.5 Programming Languages

Networks programming is beginning to convert from low-level machine languages (assembly) to high level programming language (Java, Python). The high-level languages are utilized in the SDN environment.

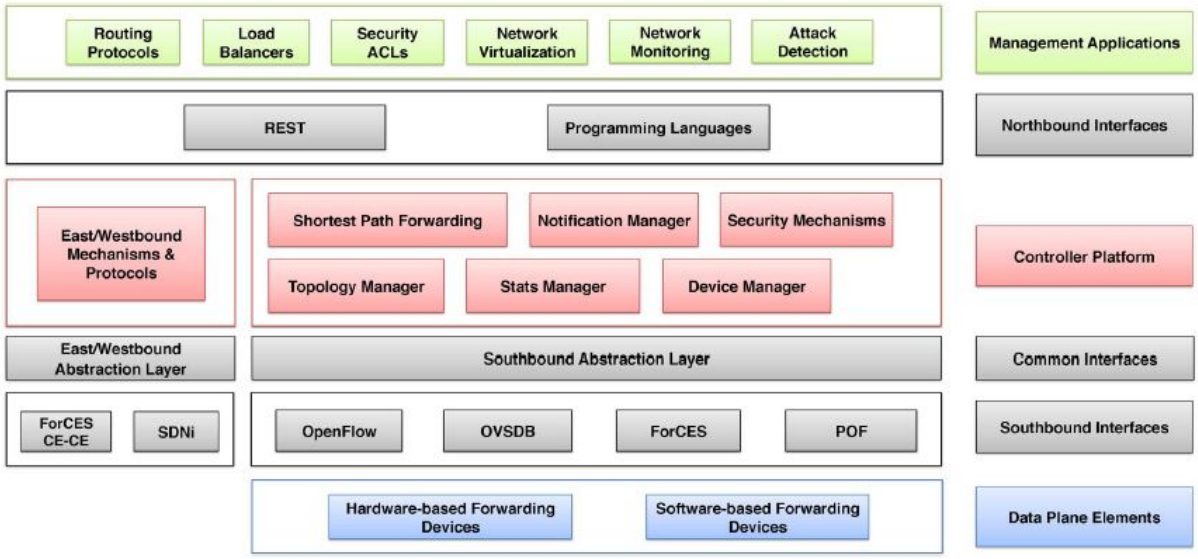


Figure 2.6: SDN control platforms: elements, services, and interfaces. [2]

2.2.6 Network Applications

Network applications execute the control logic that will be converted into commands to be uploaded in the data plane. For instance, building the routing application between two points (A, B), where the logic of the routing application is to determine a path between these points in order to carry the packets. According to the topology input (available paths), a routing application will determine the path to be used and command the controller to send the forwarding rules that carry the determined path to all forwarding devices. SDN can be used in any conventional (home, data center and Internet exchange point) regardless of network size and network applications types (routing and load balancing ..etc).

2.3 SDN Applications

The SDN is utilized to support many application as illustrated below [16]:

2.3.1 Internet Research

The SDN makes it possible to propose and implement ideas related to the future Internet without severely affecting the hardware of a running network. This is because SDN separates the control of traffic from data traffic. This means the separation between software and hardware, where this decoupling enables the examination of new ideas that aim to develop Internet architecture.

2.3.2 Rural Connections

SDN facilitates the complicated network such as the networks in data centers. So it can be used to facilitate Wi-Fi networks in rural area. In a rural area, the separation is done through using the infrastructures of different companies. The process starts by building the network on another company site in rural area. Then the party responsible to run the network (ISP) controls the devices remotely, hence the SDN make the administration of the rural network more adequate than the conventional.

2.3.3 Date Centers Upgrading

Data centers are considered the crucial part of many companies as Google data centers [17]. But these data centers need costly support maintenance. The programming and configuration of data centers have been decreased by using OpenFlow since it allows for remote control of switches from a central site.

2.3.4 Traffic Engineering

SDN facilitates the network management and boosts many network applications like load balancing that aims to divide the traffic between existing servers where SDN ease the allocation of available network services in a network.

2.4 SDN Benefits

SDN separates the control plane from the data plane and this separation give excellent control of network via programming, Where this decoupling will provide many advantages as illustrated below [18]:

2.4.1 Enhancing Configuration

Network configuration is considered one of the critical function in the network administration. When the network operators want to add a new network device to the current network that has divergent devices according to manufacturer companies, network operators have to do the manual configuration for the new device where network configuration depends on the manufacturing company. Assuming there is a problem in the network configuration, it requires from a network operator to make troubleshooting to the configuration and may do many reconfiguration attempts. Thus the traditional procedure (manual configuration) is boring and error-prone. In that vein, we can say the big challenge for the conventional network is the automatic and dynamic reconfiguration of a network. SDN unify the control plane of network elements regardless of the devices' type (switches, routers, firewalls) under one point (controller). Using a central point in SDN that is called the controller, network devices can be automatically configured as well as the ideal management can be achieved using the controller since all devices are seen under it.

2.4.2 Improving Performance

Maximum utilizing of the available network infrastructure is one of the important aims in network operation. In the traditional network, improving the performance of a complete network is complex and complicated, so the methods that are used in the conventional network depend on the local information of a part of the network to optimize that part. SDN gives a chance to improve the performance of a network as a whole. SDN utilizes the central control that manages all network elements and it has a global view of the network infrastructure.

2.4.3 Encouraging Innovation

SDN utilize the network programming where it boosts the research innovation that aims to develop and enhance network applications. Moreover, SDN produces pure decoupling between virtual networks, allowing the tests on a live environment.

2.5 Multipath TCP

According to Internet development, the need for Internet resources are growing and particularly the bandwidth. So the MPTCP will help the Internet development since the MPTCP work as resource pooling [3] by simultaneously using multiple disjoint paths. The two main benefits of multipath transport are:

- To increase the network connectivity by using more than one path.
- To increase the available network capacity by increasing the efficiency of the resource usage, and thus the user services will be available all time without any failure.

Multipath TCP (MPTCP) is a new protocol that has been proposed by the Internet Engineering Task Force (IETF). Multipath TCP, defined in [4], is a collection of extensions to allow TCP to offer a Multipath TCP service, which enables a transport connection to work through multiple paths together, the major point of MPTCP is the splitting of flow into subflows to be sent over many paths. According to Figure 2.7. Hosts A and B have two interfaces (multihomed) and each interface has a unique address and they communicate with each other through more than a single path. This Figure shows how MPTCP increases network connectivity and reliability. The addresses of hosts A and B are (A1, A2), (B1, B2) , it is notable the available paths are A1-B1, A1-B2, A2-B1, A2-B2. We conclude that the number of paths between the two host (A, B) equal num_addr(A) * num_addr(B).

.

2.6 Multipath TCP Functional Goals

The following functions are the aims of Multipath TCP [3]:

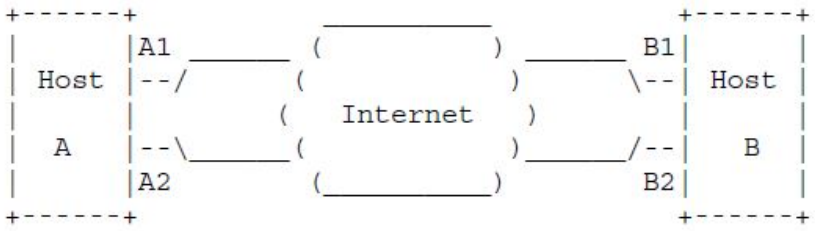


Figure 2.7: MPTCP Scenario. [3]

- Enhance Throughput: To achieve the minimum performance for multiple paths, the throughput of a Multipath TCP connection over multiple paths have to be better than a throughput of single TCP connection.
- Enhance Resilience: Multipath TCP should boost the employment of multiple paths interchangeably for resilience goals.

2.7 Compatibility Goals

A Multipath TCP should achieve a set of compatibility aims to boosts the deployment of the Internet. These Compatibility aims are presented next.

2.7.1 Application Compatibility

Multipath TCP has to be same as the TCP service model (in order, reliable, and byte-oriented delivery). Moreover, the throughput and resilience which are provided by a Multipath TCP connection should be higher than a single TCP connection.

2.7.2 Network Compatibility

In the conventional Internet architecture, network equipment run at the Internet layer and at the layers that are below the net layer, with the layers that are above the network layer (application, transport). Figure 2.8 shows the initial architecture of the Internet. But, this layering doesn't reflect the fact of Internet layering with the emergence of middleboxes.

.

The transport layer is interposed through a middlebox, sometimes it completely block the transport connections, as depicted in Figure 2.9. The network compatibility of MPTCP aims to keeping the compatibility with the Internet as it is today, as well as transport the traffic with existence middleboxes such as firewalls, NATs.

.

The modifications on Multipath TCP still exist at the transport layer. Multipath TCP must run at both versions of IPs (IPv4 and IPv6).

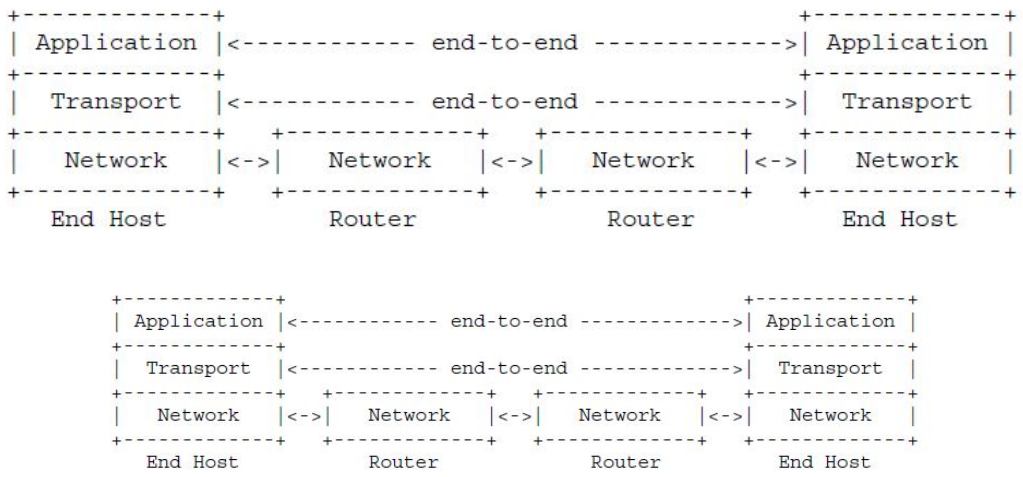


Figure 2.8: Traditional Internet Architecture. [3]

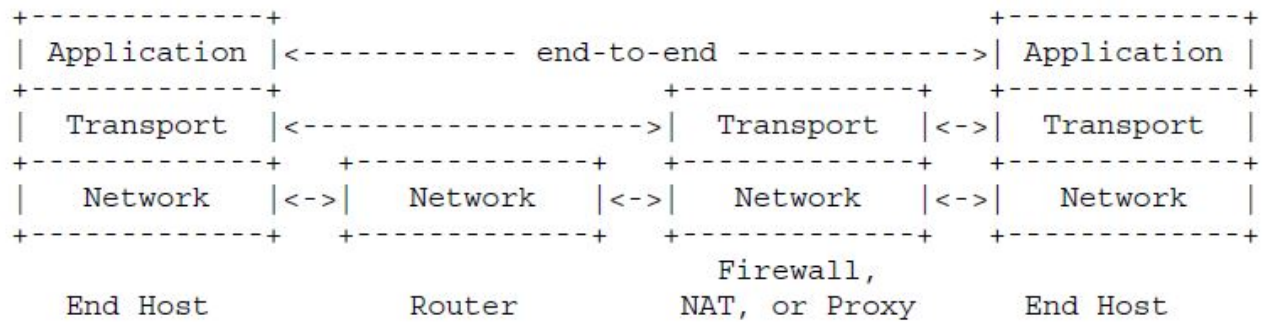


Figure 2.9: Internet Reality. [3]

2.7.3 Compatibility with Other Network Users

The architecture of MPTCP must allow the Multipath TCP flows to work with the flows of a single-path TCP (SPTCP), also it must be able to compete on the available bandwidth. The users who use SPTCP must not be harmed by the users who use multiple paths at shared bottlenecks. On a shared bottleneck, Multipath TCP flows have to share the bandwidth between each other with equal fairness and even with a SPTCP.

2.7.4 Security Goals

A number of new threats will be brought by multipath TCP, so the security of Multipath TCP should be higher than SPTCP (i.e Multipath TCP should be more secure than SPTCP) [3]. We will not discuss the security requirements and aspects of MPTCP because it out of the scope of this work.

2.8 Multipath TCP in the Networking Stack

MPTCP runs at the transport layer like TCP as well as it is clear to both higher and lower layers, it is a collection of extra features on top of standard TCP, Figure 2.10 describes this layering.

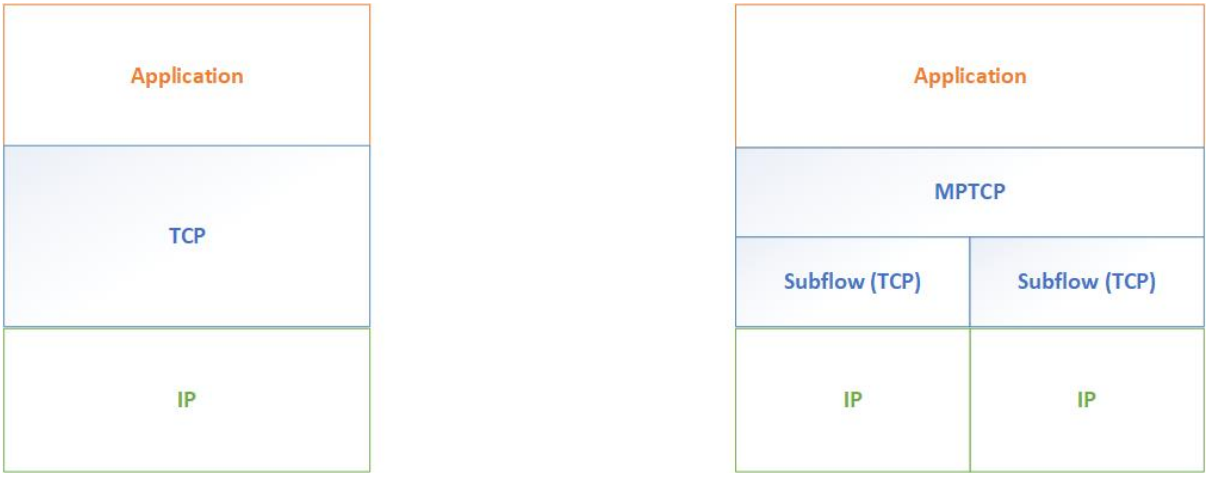


Figure 2.10: comparing of Standard TCP and MPTCP Protocol Stacks.

.

2.9 MPTCP Operation

A bidirectional byte stream connection is offered by MPTCP between two hosts which are communicating like usual TCP without any change in the application layer. But, MPTCP allows the hosts to utilize several paths and each path has an IP address to transfer the data that belongs to the MPTCP connection. In that vein, MPTCP connection seems like a usual TCP connection to the application layer. But, to the network layer, after the subflows are generated, each of them seems like a usual TCP flow whose segments carry a new TCP option type.

2.9.1 Initiating an MPTCP Connection

Initiating MPTCP connection starts like usual TCP (SYN, SYN/ACK, and ACK) packets and they exchange on each path of the multiple paths. But, these packets carry a new option called Multipath TCP Capable (MP_CAPABLE) where the aims of this option are [4]:

- It checks if the host that wishes to communicate with it is empowered with MPTCP or not.
- It is used for security purpose when there are new subflows that will be created.

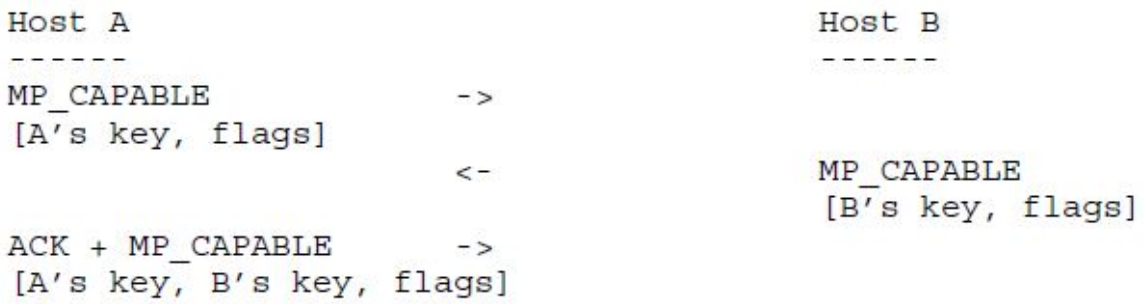


Figure 2.11: Initiating an MPTCP Connection. [4]

2.9.2 Starting a New Subflow

Assuming the MPTCP connection has been initiated between hosts through MP_CAPABLE, and additional subflows can be jointed to the current connection. Hosts have knowledge of their own addresses and via the signaling exchanges become aware of the other host's addresses. Utilizing this knowledge, a host can start a new subflow over a currently unused pair of addresses. In the MP_CAPABLE handshaking process, the sender and receiver are exchanging the keys between them and this offers an approach that can be utilized for authenticating purpose when a new subflow is created. In the connection among the initial subflows, just MP_CAPABLE option is used as well as the Join option will be used when new subflows want to join the current connection.

Assuming a new subflows will be created, it will initiate the connection like the method that is used in a usual TCP connection (SYN, SYN/ACK, and ACK). But, the handshaking packets carry a new option called MP_JOIN option. As presented in Figure 2.12, the sender transmits a random number, token, and address ID in the first MP_JOIN on the SYN packet, the receiver will reply with an SYN/ACK packet that too including an MP_JOIN option containing a random number and a truncated Hash-based Message Authentication Code (HMAC).

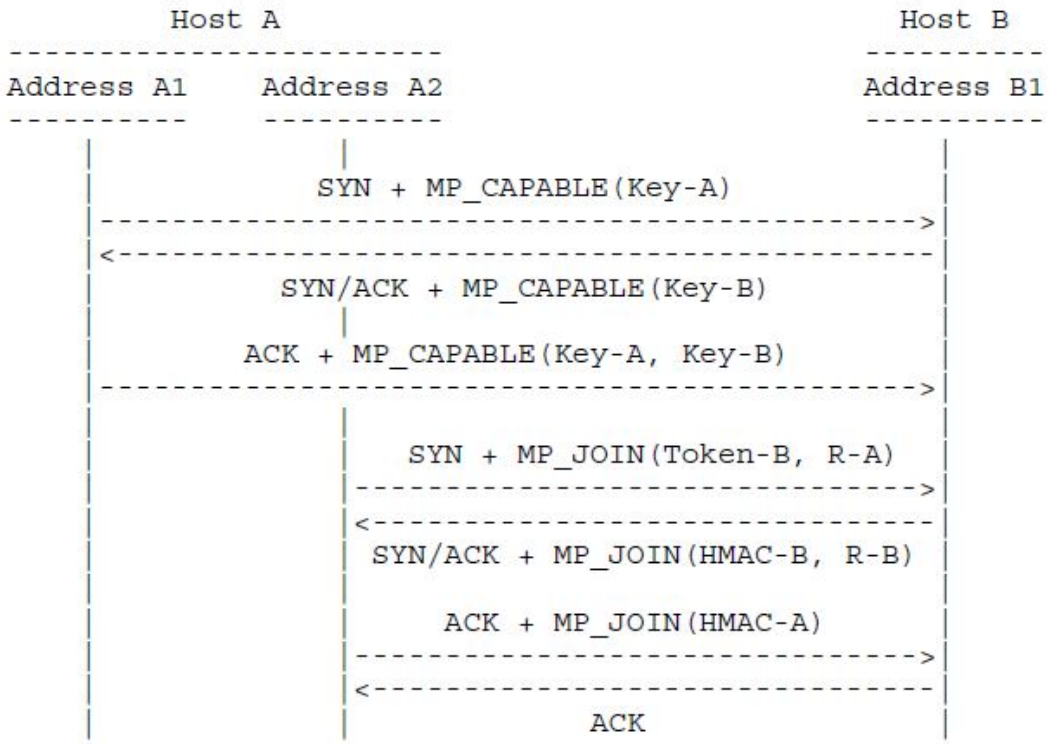


Figure 2.12: Example of Subflow establishment in MPTCP. [4]

2.9.3 Closing an MPTCP Connection

If there is no data to transmit from the sender side, the sender sends the receiver "Data FIN", where this signal gives the receiver an indication that the sender has no more data to transmit, once the receiver receives all data of MPTCP connection, it replies with "Data FIN".

Figure 2.13: Example of Closing MPTCP Connection. [4]

This chapter tackles the idea of SDN, where SDN separates CP from the DP and the DPs work as normal forwarding devices that receive the instructions from CP, CP has a global view of the network state. Moreover, it discussed the application field of SDN is like Rural Connections, Internet Research, Date Centers Upgrading, and Traffic Engineering. Moreover, This chapter presents the MPTCP protocol, MPTCP has been proposed as

an extension of regular TCP and its major purposes are offering better performance, throughput, and resilience to failures by dividing the flow into multiple subflows which will be then sent over many paths. MPTCP architecture must allow normal TCP flows to compete with MPTCP flows on available bandwidth.

Chapter 3

Related Works

Contents

3.1 MPTCP

MPTCP has attracted the attention of many researchers. MPTCP is one of the achievements of the Internet Engineering Task Force (IETF) [4]. The main advantages of MPTCP are to increase both throughput and robustness by utilizing more than one path [19]. MPTCP has a good performance with long flows because the data is divided into subflows then transmitted over multiple paths. The authors in [20] suggested an enhanced MPTCP congestion control algorithm over wireless networks to increase the throughput as well as to gain load balancing. The algorithm idea is to make full use of the congestion information of all the subflows belonging to a TCP connection to adaptively modify the transmission rate of each subflow. Moreover, the efficiency of MPTCP over long flows has been confirmed in [10, 21].

Many factors affect the performance of MPTCP [22]. Path scheduling and congestion control are the two main factors. The main job of the congestion control algorithm is controlling the transmission rate of each subflow, that aims to enhance throughput and link utilization by moving traffic from the path that is congested to less congested path. Many of MPTCP congestion control algorithms have been suggested such as the Linked Increases Algorithm (LIA) [23], Opportunistic Linked-Increases Algorithm (OLIA) [24], Balanced Linked Adaptation Algorithm (BALIA) [21], and Weighted Vegas algorithm (WVegas) [25]).

The second factor is path scheduling, it aims to utilize more than a single path to spread the data on them. The default Path scheduling in MPTCP is the lowest Round Trip Time (RTT). In the beginning, it utilizes the path that has the lowest RTT to send the packets until its congestion window becomes full. Then it moves to the next higher RTT path. The authors in [26] proposed an algorithm for the packet scheduler named

freezes packet. This algorithm aims to reduce the flow completion time for short flows. Their idea is based on measuring the delay difference between two paths, after that if the difference is significant then that packets will be sent over the fast path, and the other paths freezes.

Authors in [10] proposed Maximum MultiPath TCP (MMPTCP) algorithm, this algorithm aims to obtain low latency for short flows and high goodput for long flows. MMPTCP algorithm works in two phases. At first, it randomly scatters packets in the network under a single congestion window exploiting all available paths to handle the short flows and after a certain amount of data the MMPTCP switches to a regular MultiPath TCP mode for handling the long flows.

The authors in [27] proposed algorithm named adaptive multi-path transmission control protocol AMTCP, it aims to decrease resource overheads for short flows and obtaining a higher throughput for large flows. AMTCP idea is controlling the number of subflows based on application workloads. The explanations outlined below demonstrate why the MPTCP affects short flows negatively [10]:

- Paths heterogeneity in a network produces an increase in delays [8, 11], indeed this heterogeneity causes packet reordering. For instance, if a network has two paths and they were heterogeneous so when the data is transmitted over these paths, the packets that transmitted over the fast path will arrive before the packets that were transmitted over the slow path thus the receiver will fill its buffer whereas the other packets not received. Moreover, MPTCP may select the slowest path if the congestion window of the fast path is not available, so this makes a longer flow completion time.

- In general, the size of short flows are very small, as well as the short flows pose almost 40% from the web traffic with size less than 1MB [28]. In the case a single packet is lost from the short flows, it can compel whole connection to wait for Retransmission Timeout (RTO) to be triggered since this lost packet cannot be recovered using fast retransmission [10], this will lead to increasing the latency for the applications that consider delay is critical.

MPTCP has a negative impact on the behavior of the short flows that its size almost hundreds of KBs. Because of the Congestion Window (CWN) of the short flows may be small over its lifetime, so when a single packet is lost, MPTCP will not be able to recover this loss through fast retransmission which leads to a timeout. In [29], authors suggested Dynamic Multipath TCP (DMPTCP) algorithm, the objective of this algorithm is enhancing the performance of short flows in term of completion time, as well as long flows in term of throughput. This happens through evaluating the available subflows and automatically adjusting the number of subflows that meet the flows requirements. They proposed an analytical model aims to predict the amount of data that can be transferred over the faster subflows simultaneously before the packet arrival time over the slower subflow.

In [5], authors suggested an algorithm for the Internet traffic named Multipath TCP for short flow (MPTCP-SF), its idea has been built based on calculating the size of data being sent per flow to determine the flows classification and then allocate the number of MPTCP subflows based on flow size. They classified the web traffic into three categories, the thing that differentiates between these categories is the traffic size, where the size for each of them is 100KB, between 100KB to 400 KB, more than 400 KB. The algorithm selects the fastest path for the first category, a subset of available subflows for the second category as well as all available subflows for the last one.

3.2 SDN AND MPTCP

The SDN has drawn researchers ' interest. They benefited from the essential SDN function, which is decoupling the control plane (CP) from the data plane (DP) and the control plane became a logically centralized controller, resulting in the SDN controller has a global view of the network.

The Authors in [30] compared the throughput between MPTCP (PlanetLab Europe, which can be used by experimenters and researchers to test and validate their ideas relevant to multipath.) and the regular Internet, where they utilized the SDN controller to adjust the routes. The results confirmed the advantage of MPTCP in term of throughput, where the normal TCP offered 8.02 Mbps whereas MPTCP offered 14 Mbps throughput. But, they didnt take into account the effect of MPTCP on the short flows as well as the effect of path difference in terms of bandwidth on the performance of MPTCP.

Since the satellite networks are multilayered, so the MPTCP can be utilized in these networks. The work in [31] utilized the MPTCP in satellite network to increase the throughput and it propose the use of SDN to solve the shared bottleneck problem. Authors proposed an approach to solving the problem of the static number of subflows by making communication between SDN controller and an end host where they added a module at controller side as well as at end-host side that to dynamically adjust the number of subflow according to network status. Moreover, they proposed load and shared routing algorithms as well as they showed that the increase in number of subflows will increase throughput. But their approach was tested on file transferring and they didnt take short flow applications as well as he effect of path difference in terms of bandwidth on the performance of MPTCP.

In large-scale Layer 2 networks such as the networks that exist in data centers, the authors in [32] used SDN to boost the MPTCP efficiency to achieve high throughput for big size data exchange between servers. To use all the available bandwidth, they suggested modules located on an SDN controller and servers to control the number of subflows, as well as a routing algorithm to solve one of the MPTCP problems (MPTCP is an end-to-end protocol). But, they ignored the effect of MPTCP on short flows.

The authors in [33] proposed a routing approach by using the segment routing (SR) to mitigate the storage requirements of Open Flow switch. MPTCP divides large multime-

dia flows into subflows to be transmitted over multiple paths. Authors in [34] utilized the SDN to enhance the performance of MPTCP, they proposed a routing algorithm based on SR to adjust the subflows number, as well as they, showed that MPTCP is better than a normal TCP in term of throughput, link utilization, and end-users QoE. But, they tested their approach in the 5G data center with large flows and they didn't take the short flows into account as well as the effect of path difference in terms of bandwidth on the performance of MPTCP.

The authors in [7] proposed an MPTCP-aware SDN, where they allocated the MPTCP subflows to the disjoint paths to prevent conflict between the subflows that belong to the same connection. But, they didnt take into account the effect of MPTCP on short flows as well as the effect of path difference in terms of bandwidth on the performance of MPTCP. The authors in [35] proposed a scalable SDN-assisted architecture to solve the collision problem and they take paths dissimilarity in terms of delay into account. But, they ignored paths dissimilarity in terms of bandwidth.

Chapter 4

MPTCP Performance Evaluation

Contents

The aims of this chapter are:

- Study the effect of MPTCP on long flows which are sensitive to throughout.
- Investigate the effect of MPTCP on short flows which are sensitive to latency.
- Prove that the bandwidth gap between paths is a critical factor influencing the performance of the short flows.

4.1 Experimental Design

Figure 4.1 shows the proposed topology, we use the Mininet to simulate this network. Mininet [12] is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking. Furthermore, it yields a more effective use of time and resources compared to other workflows.
We use the IPerf tool to measure the long flow's throughput. IPerf is a tool used to evaluate the performance of a network, it can produce streams of data to test the throughput between the two ends. For short flows, we used a simple HTTP Server module, this module is existing in the mininet to measuring Average download time.

It is worth mentioning in our assumptions regarding the bandwidth value of the paths that our concern is the bandwidth gap (dissimilarity in terms of bandwidth) between paths not the absolute value of the path bandwidth.

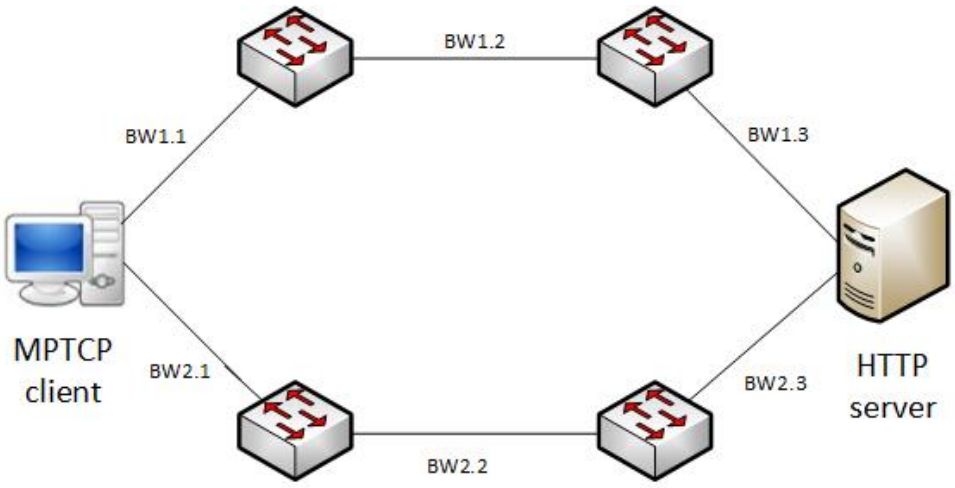


Figure 4.1: The experiment topology

4.1.1 MPTCP Configuration

In this subsection, we list and explain some of the MPTCP options:

* **net.mptcp_enabled**
  This option responsible for Disabling/Enabling MPTCP on the machine, 1 means enable, and 0 means disable.

* **net.mptcp.mptcp_syn retries**
  This option responsible for determining how often retransmission an SYN with the MP_CAPABLE option, and after that, the SYN will not carry the MP_CAPABLE option. This option is to handle middleboxes that deny SYNs with unknown TCP options, the default value is 3.

* **net.mptcp.mptcp_checksum**
  This option enables the use of MPTCP_checksum, it has two values(0,1). 0 for disable and 1 for enable.

* **net.mptcp.mptcp_path_manager**
  This is a module responsible for the management between the multiple paths efficiently, and this option has three values:

  - default: This choice will not advertise the IP addresses of the host and will not create new subflows. However, it will allow the passive creation of new subflows.
  - fullmesh: This choice will build a full-mesh of subflows among all available subflows.

- ndiffports: This choice creates X subflows between the same pair of IP-addresses.

* **net.mptcp.mptcp_scheduler**
  This a module responsible for utilizing more than a path to spread the data on them, and this option has three values:

  - default: It will send the data on the path that has the lowest RTT until its congestion window becomes full. Then it moves to the next higher RTT path.
  - roundrobin: This choice will send the data in a round-robin manner.
  - redundant: This choice will send the data on all available subflows in a redundant method.

We use MPTCP Release (v0.95) as shown in Figure 4.2.

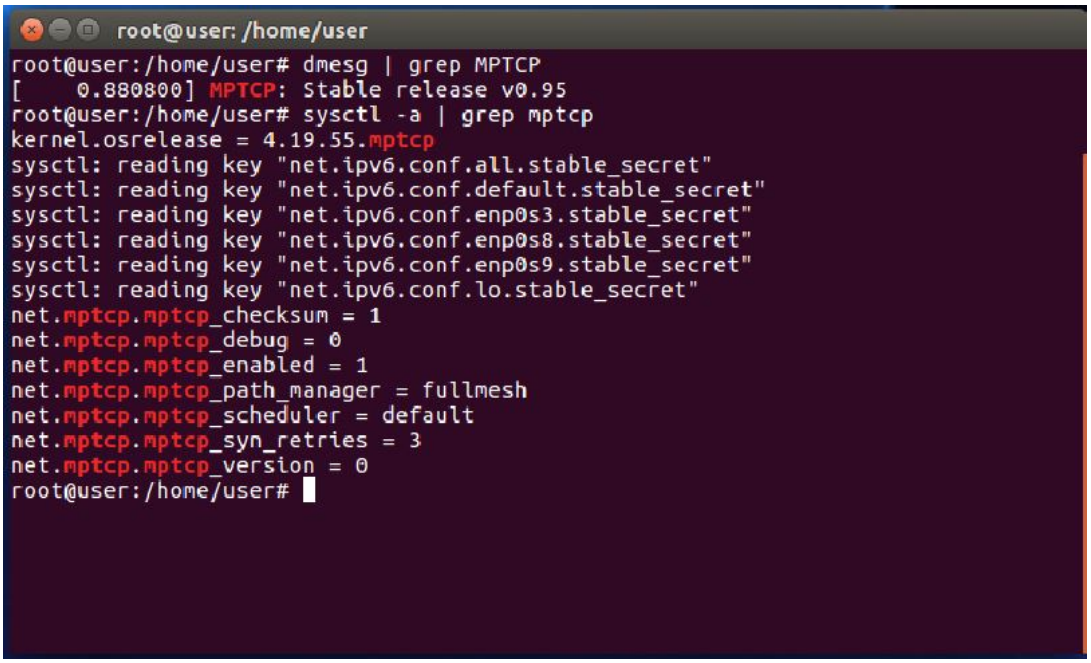


Figure 4.2: MPTCP Release (v9.95) and Setting

4.2 Impact of MPTCP on Long Flows

MPTCP has been proposed to increase the throughput of the network by pooling bandwidth and to increase the network connectivity, where MPTCP divides the flow into subflows to be transmitted over multiple paths.
MPTCP will be more efficient for the applications that are sensitive to the throughput like transferring data with big sizes (Long Flow). This section aims to prove that MPTCP has a positive impact on long flows.

4.2.1 Homogeneous Network

In this section, we study the impact of MPTCP on long flows when the network has homogeneous paths. We will do that by measuring the throughput between the server and the client. We assumed the bandwidth for each link is 10 Mbit/sec and the file size is 100MB.
Note: In all scenarios, we assumed the delay is zero, and all Figures' results are the average of 100 samples. It is worth mentioning that we used the IPerf tool to measure the efficiency of MPTCP on long flows in terms of throughput, so this section aims to prove that MPTCP has a positive impact on long flows regardless of the tool used in the measuring as confirmed in [10, 21].

Figure 4.3 illustrates the difference in the performance of SPTCP and MPTCP. It is significantly noted the MPTCP shows high performance comparing with SPTCP, where the average throughput for MPTCP is 15.133 Mbit/s while as for SPTCP is 7.14 Mbit/s.

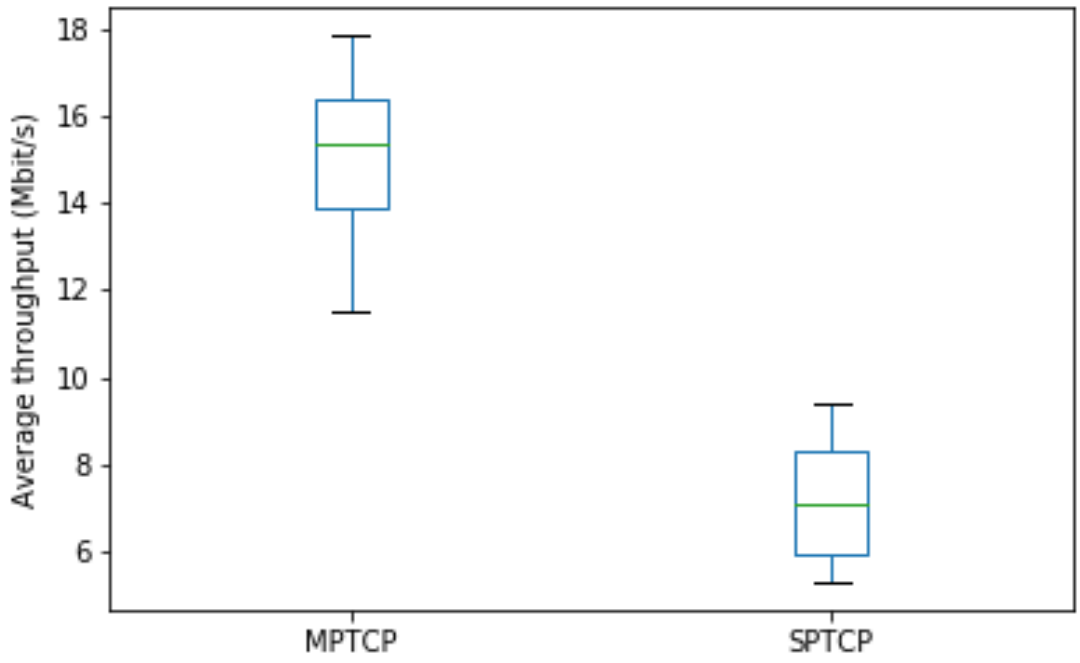


Figure 4.3: Comparing the performance of SPTCP and MPTCP in terms of throughput in bandwidth-homogeneous networks.

4.2.2 Heterogeneous Network

In this section, we study the impact of MPTCP on the long flows when the network has heterogeneous paths in terms of bandwidth. We assumed the bandwidth for the first path is fixed 10Mbit/sec (BW for each link of the first path = 10 Mbit/sec) and the bandwidth for the second path is varied as follows:

- 0.1Mbit/sec (BW for each link of the second path = 0.1Mbit/sec).
- 1Mbit/sec (BW for each link of the second path = 1Mbit/sec).
- 2Mbit/sec (BW for each link of the second path = 2Mbit/sec).
- 3Mbit/sec (BW for each link of the second path = 3Mbit/sec).
- 4Mbit/sec (BW for each link of the second path = 4Mbit/sec).

Equation 4.1 denotes the normalized MPTCP bandwidth gap, where BWp_1 is the first path bandwidth and BWp_2 is the second path bandwidth.

$$BWgap = \frac{|BWp_1 - BWp_2|}{Max\{BWp_1, BWp_2\}} \tag{4.1}$$

Figure 4.4 shows the performance of SPTCP and MPTCP when the paths are heterogeneous. The Figure shows that the decrease in heterogeneity in terms of bandwidth will lead to increasing the average throughput. Moreover, we noted there is no significant difference between the performance of SPTCP and MPTCP in the worst-case scenario ($BWgap = 0.99$).

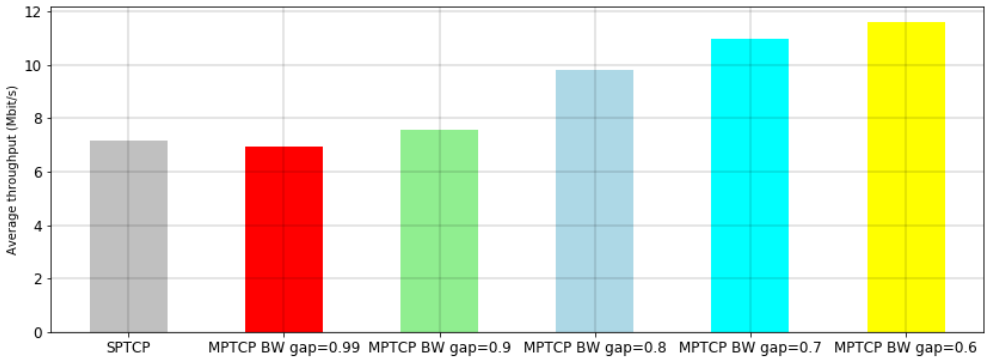


Figure 4.4: Comparing the performance of SPTCP and MPTCP in terms of throughput in heterogeneous network.

Summary:

- When the bandwidth paths are homogeneous, MPTCP performs better than SPTCP.
- When the bandwidth paths are heterogeneous, the decrease in heterogeneity in terms of bandwidth will lead to increasing the average throughput.
- As for MPTCP bandwidth gap = 0.99 (highest heterogeneity scenario), the performance of SPTCP and MPTCP almost is equal in terms of throughput.

4.3 Impact of MPTCP on Short Flows

This section studies the impact of MPTCP on short flows that are sensitive to latency. Moreover, it evaluates the performance of SPTCP and MPTCP in homogeneous and heterogeneous scenarios. We assumed that the client would download files from the HTTP server with different sizes (10 KB, 50 KB, 100 KB, 200 KB, 500 KB, 1 MB, 2 MB, and 5 MB). For the sizes of short flows, we attempted to cover the sizes as stated in [5] where the authors classified the web traffic into three categories, where the size for each of them is 100KB, between 100KB to 400 KB, more than 400 KB.

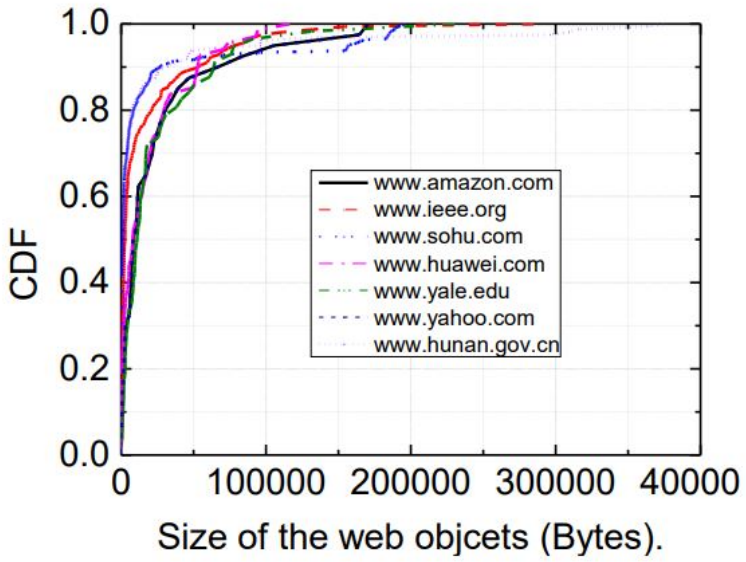


Figure 4.5: Distribution of web objects. [5]

4.3.1 Homogeneous Network

In this section, we study the impact of MPTCP on the short flows when the network has homogeneous paths. This is by measuring the average download time. We assumed the bandwidth for each link is 10 Mbit/sec (bandwidth for each path is 10 Mbit/sec).

Figure 4.6 shows that SPTCP and MPTCP performances are nearly equal in terms of average download time when file size = 10 KB.

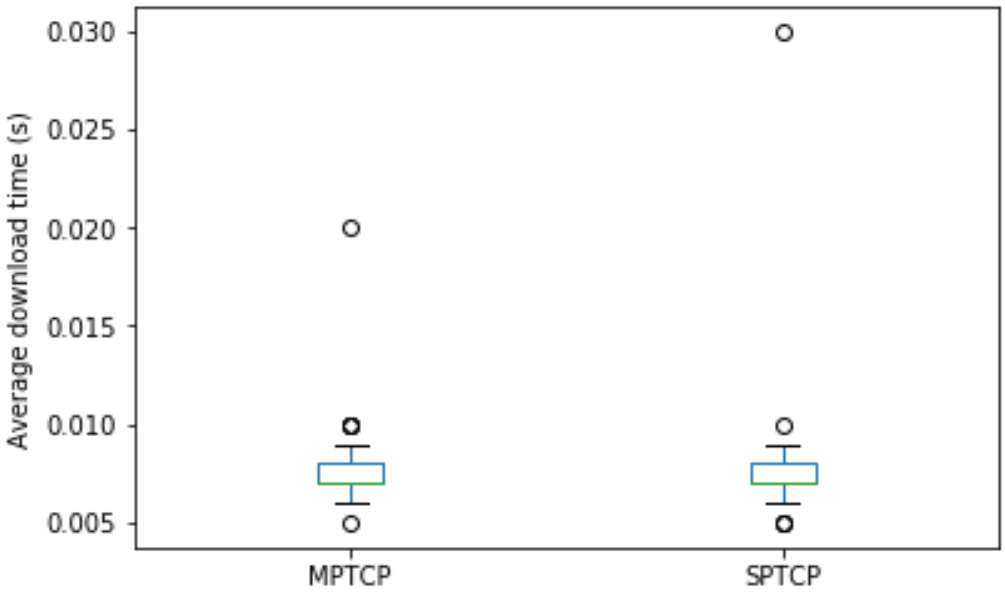


Figure 4.6: Average download complete time when file size equals 10 KB, the circles are the outliers.

Figure 4.7 shows that when the file size 50 KB, the performance of MPTCP is better than the SPTCP since the average download time for MPTCP is 0.0233 second while for SPTCP is 0.0379 second.

Figure 4.8a shows that when the file size 100 KB, the performance of MPTCP is better than the SPTCP since the average download time for MPTCP is 0.0542 second while for SPTCP is 0.0874 second. Figure 4.8b shows also that when the file size 200 KB, the performance of MPTCP is better than the SPTCP since the average download time for MPTCP is 0.0878 second while for SPTCP is 0.202 second. Similarly in Figure 4.8c where the file size 500 KB, the performance of MPTCP is better than the SPTCP since the average download time for MPTCP is 0.284 second while for SPTCP is 0.576 second. The same result can be observed in Figure 4.8d where the file size 1 MB, the performance of MPTCP is better than the SPTCP since the average download time for MPTCP is 0.568 second while for SPTCP is 1.157 seconds. However here, the difference between MPTCP and SPTC average download time is more than that when the file sizes were 200 KB and 500 KB. The difference between average download time of MPTCP and SPTCP increases further as the file size increases to 2 MB as shown in Figure 4.8e. In this Figure the performance of MPTCP is better than the SPTCP since the average download time for MPTCP is 1.132 seconds while for SPTCP is 2.544 seconds. Figure 4.8f shows that when the file size 5 MB, the performance of MPTCP is better than the SPTCP since the average download time for MPTCP is 2.904 seconds while for SPTCP is 6.558 seconds.

Figure 4.9 strongly shows that MPTCP performs better than SPTCP especially when

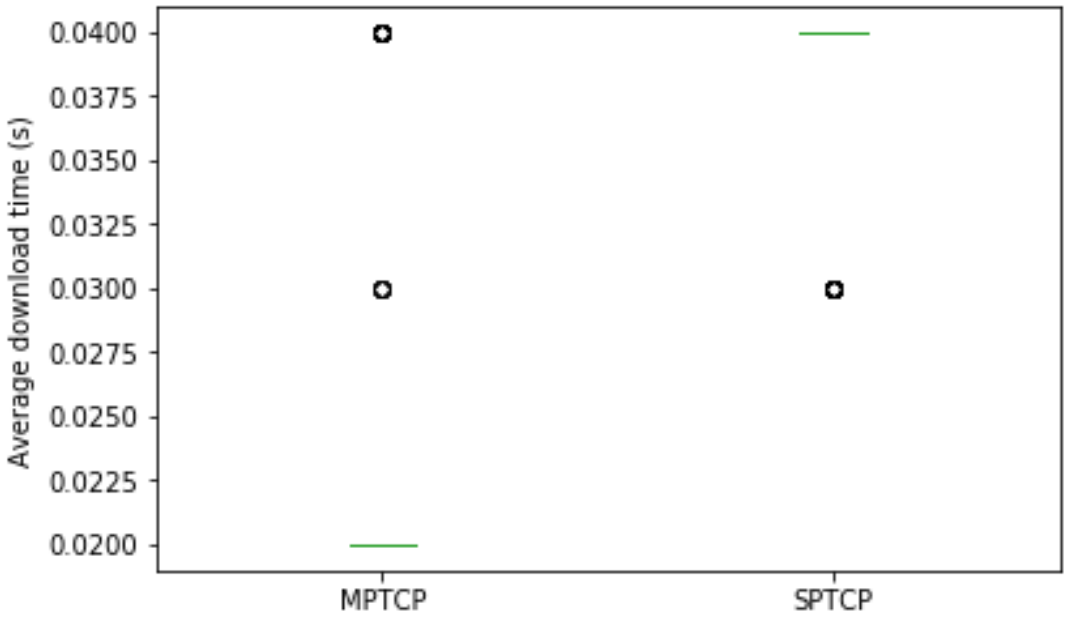
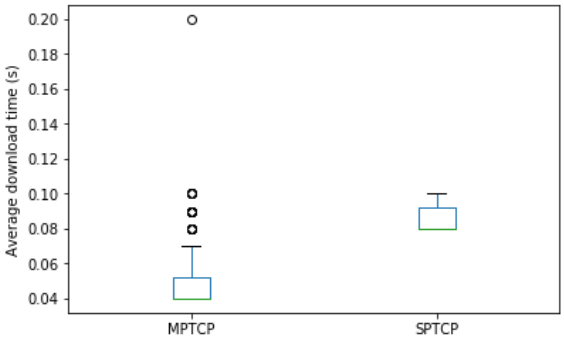
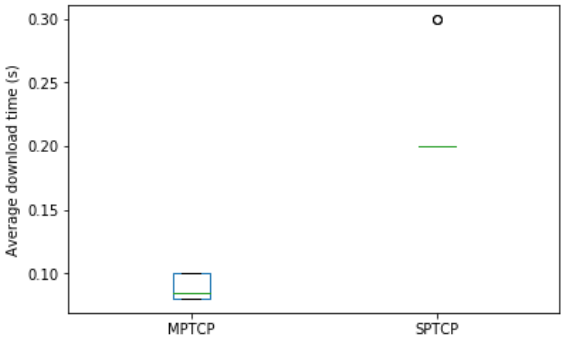


Figure 4.7: Average download complete time when file size equals 50 KB, the circles are the outliers.

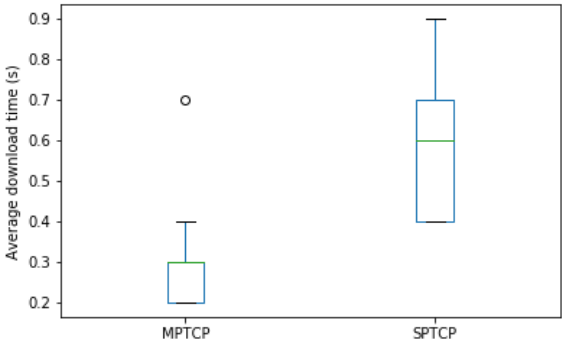
the file size is greater than 100 KB. Table 4.1 shows MPTCP and SPTCP output in terms of the average download time for different file sizes, The table strongly demonstrates that MPTCP performs better than SPTCP particularly when the file size exceeds 100 KB. We conclude that the performance of MPTCP is better than the SPTCP since in MPTCP the flows are split into subflows then sent over many paths while in SPTCP the flows are sent over a single path, so the average download time of these flows over SPTCP will be more than MPTCP.



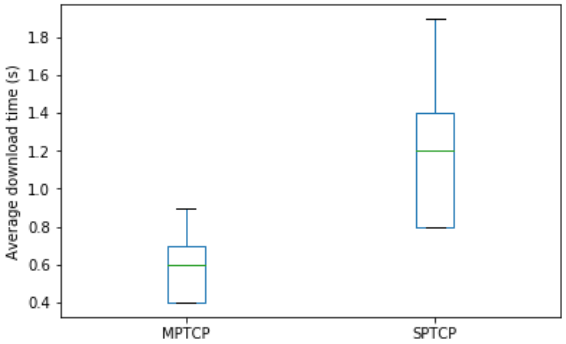
(a) File size equals 100 KB



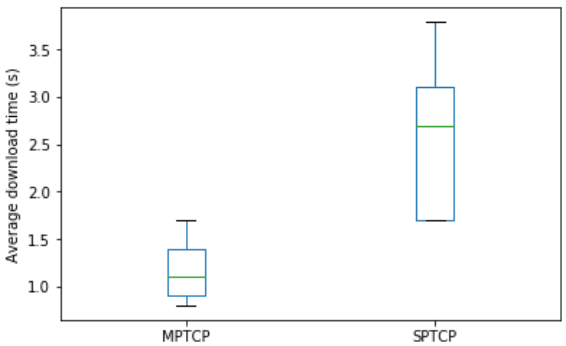
(b) File size equals 200 KB



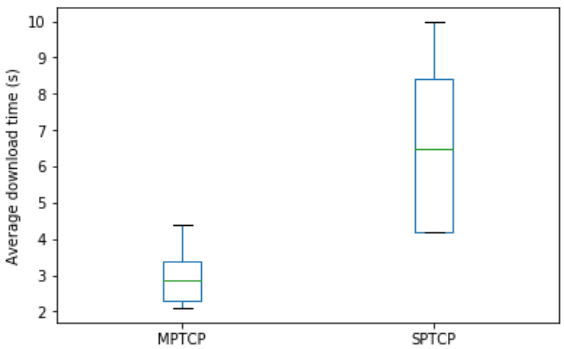
(c) File size equals 500 KB



(d) File size equals 1 MB



(e) File size equals 2 MB



(f) File size equals 5 MB

Figure 4.8: Average download complete time for Homogeneous Networks, the circles are the outliers

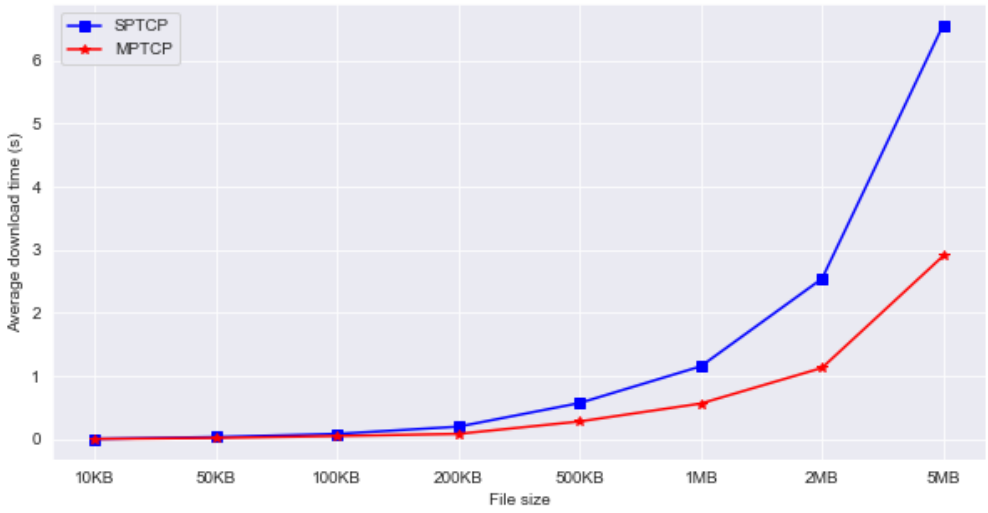


Figure 4.9: Average download complete time with different file sizes.

File Size	SPTCP	MPTCP
10 KB	0.00752 sec.	0.00767 sec.
50 KB	0.0379 sec.	0.0233 sec.
100 KB	0.0874 sec.	0.0542 sec.
200 KB	0.202 sec.	0.0878 sec.
500 KB	0.576 sec.	0.284 sec.
1 MB	1.157 sec.	0.568 sec.
2 MB	2.544 sec.	1.132 sec.
5 MB	6.558 sec.	2.904 sec.

Table 4.1: Average download complete time for different file sizes in the homogenous network.

4.3.2 Heterogeneous Network

In this section, we investigate the impact of MPTCP on short flows when the network has heterogeneous bandwidth paths.

Scenario 1

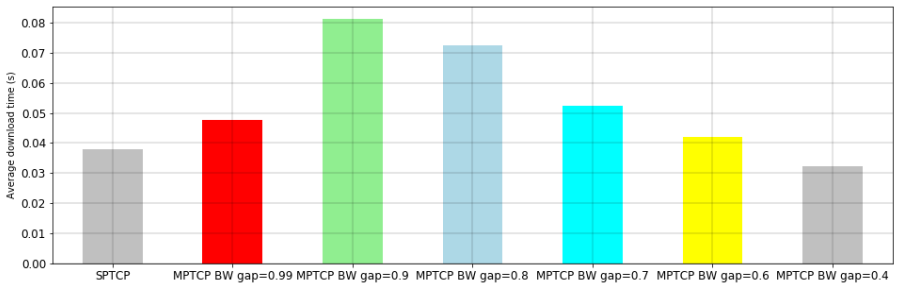
We assume the bandwidth for the first path is fixed 10Mbit/sec (BW for each link of the first path = 10 Mbit/sec) and the bandwidth for the second path is varied as follows:

- 0.1Mbit/sec (BW for each link of the second path = 0.1Mbit/sec).

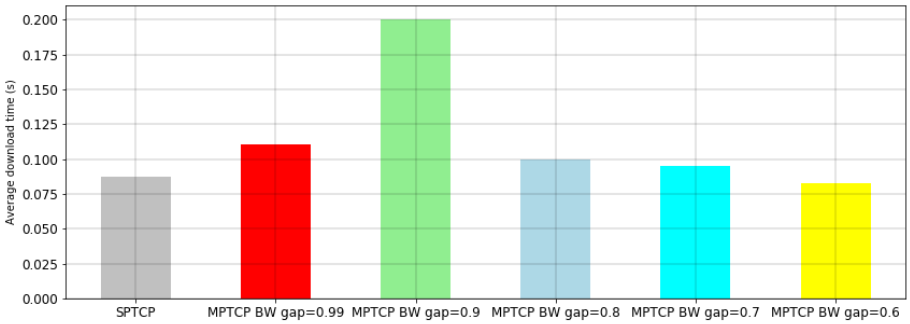
- 1Mbit/sec (BW for each link of the second path = 1Mbit/sec).
- 2Mbit/sec (BW for each link of the second path = 2Mbit/sec).
- 3Mbit/sec (BW for each link of the second path = 3Mbit/sec).
- 4Mbit/sec (BW for each link of the second path = 4Mbit/sec).

Figure 4.10a shows that when the file size 50 KB, the performance of SPTCP is better than the MPTCP when the bandwidth gap is 0.99, 0.9, 0.8, 0.7, and 0.6, as well as it illustrates that the performance of MPTCP and SPTCP is nearly similar when the gap is 0.4. Figure 4.10b also shows that the performance of SPTCP is higher than MPTCP when the bandwidth gap is 0.99,0.9,0.8, and 0.7 when the file size is 100 KB and illustrates that when the gap is 0.6, the performance of MPTCP and SPTCP is almost identical. MPTCP performs better than SPTCP when the bandwidth gap is equal to or less than 0.7 as depicted in Figure 4.10c where the file size equals 200 KB, also in this Figure, the performance of SPTCP is better than the MPTCP when the bandwidth gap is 0.99, 0.9, and 0.8. Figure 4.11a shows that when the file size is 500 KB, when the bandwidth gap is 0.99, the performance of SPTCP is better than MPTCP and illustrates that when the gap is 0.9, the performance of MPTCP and SPTCP is almost similar. In addition, when the bandwidth gap is equal to or less than 0.8, MPTCP performs better than SPTCP. The same result can be observed in Figure 4.11b where the file size 1 MB, the performance of SPTCP is better than the MPTCP when the bandwidth gap is 0.99, as well as it illustrates that the performance of MPTCP and SPTCP is nearly similar when the gap is 0.9. Moreover, the MPTCP performs better than SPTCP when the bandwidth gap is equal to or less than 0.8. Last Figure 4.11c indicates that when the file size is equal to 5 MB, while the bandwidth gap is 0.99, the SPTCP efficiency is higher than the MPTCP. Moreover, when the bandwidth gap is equal to or less than 0.9, the MPTCP performs better than SPTCP.

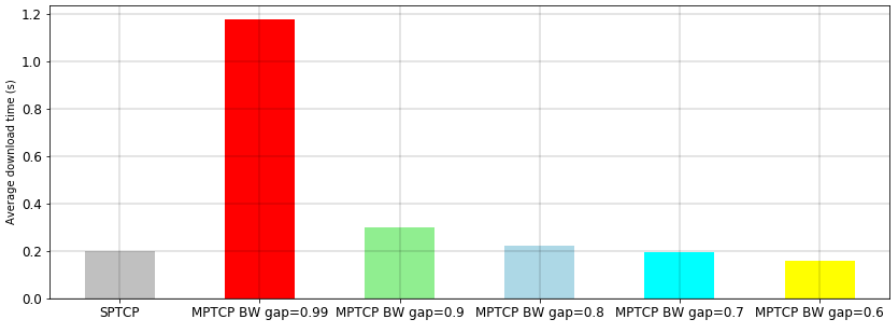
Table 4.2 shows the performance of SPTCP and MPTCP in terms of the average download time for different file sizes when the bandwidth of the paths is different, the table shows that decreasing the gap in terms of bandwidth will lead to enhance the performance of MPTCP.



(a) File size equals 50 KB

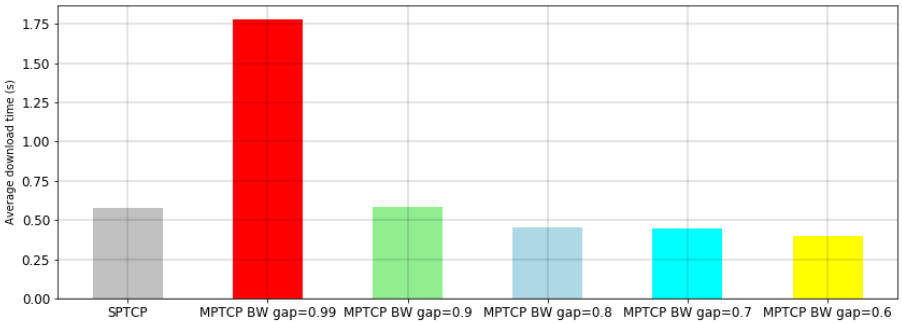


(b) File size equals 100 KB

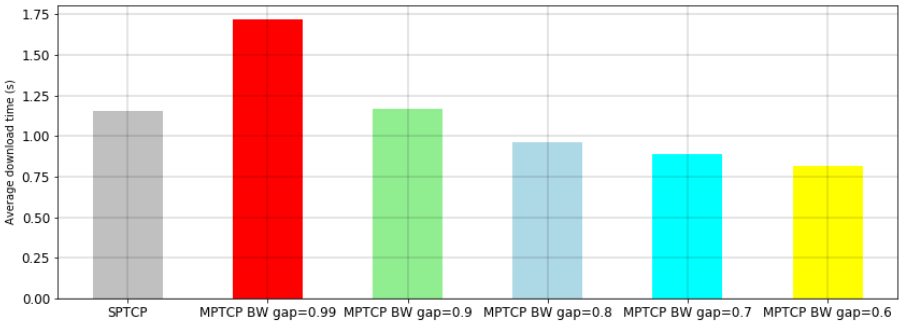


(c) File size equals 200 KB

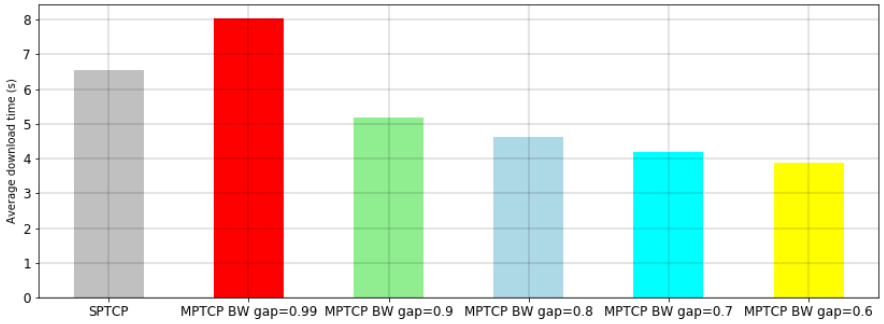
Figure 4.10: Average download complete time for Heterogeneous Networks



(a) File size equals 500 KB



(b) File size equals 1 MB



(c) File size equals 5 MB

Figure 4.11: Average download complete time for Heterogeneous Networks

File Size	SPTCP	gap 0.99	gap 0.9	gap 0.8	gap 0.7	gap 0.6	gap 0.4
50 KB	0.038 sec.	0.0476 sec.	0.081 sec.	0.073 sec.	0.052 sec.	0.0421 sec.	0.0322 sec.
100 KB	0.087 sec.	0.1101 sec.	0.2 sec.	0.1 sec.	0.095 sec.	0.0823 sec.	0.0607 sec.
200 KB	0.202 sec.	1.177 sec.	0.298 sec.	0.221 sec.	0.197 sec.	0.157 sec.	0.112 sec.
500 KB	0.576 sec.	1.778 sec.	0.584 sec.	0.456 sec.	0.444 sec.	0.395 sec.	0.349 sec.
1 MB	1.157 sec.	1.718 sec.	1.168 sec.	0.965 sec.	0.892 sec.	0.817 sec.	0.589 sec.
5 MB	6.558 sec.	8.035 sec.	5.169 sec.	4.636 sec.	4.183 sec.	3.869 sec.	3.418 sec.

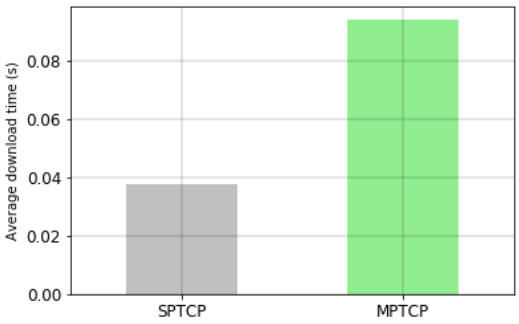
Table 4.2: Average download complete time for different file sizes in the heterogeneous network.

Scenario 2

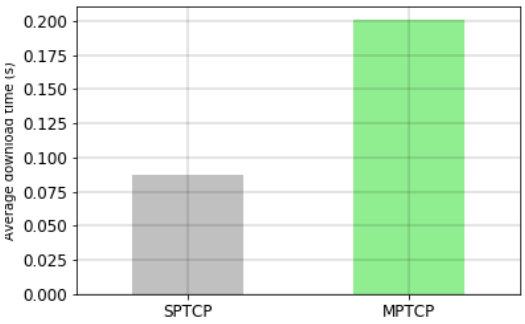
We assumed the bandwidth for each link of the first path is fixed 10Mbit/sec (BW of the first path = 10 Mbit/sec), and the bandwidths of the second path links are varied (BW2.1=10 Mbit/sec, BW2.2=1Mbit/sec, BW2.3=10Mbit/sec).

Figure 4.12a shows that when the file size 50 KB, the performance of MPTCP almost as the performance of MPTCP in Figure4.10a when the MPTCP BW gap = 0.9 second. Figure 4.12b shows also the performance of MPTCP is nearly equal to MPTCP performance in Figure 4.10b when the MPTCP BW gap is = 0.9. Similarly in Figure 4.12c where the file size 200 KB, the MPTCP output is almost the same as the MPTCP output in Figure 4.10c when the MPTCP BW gap is = 0.9. The same result can be observed in Figure 4.12d where the file size 500 KB, the MPTCP performance is approximately the same as the MPTCP performance in Figure4.11a when the MPTCP BW difference is 0.9. MPTCP performance in Figure 4.12e is nearly equal to MPTCP performance in Figure 4.11b when the MPTCP BW gap is = 0.9.

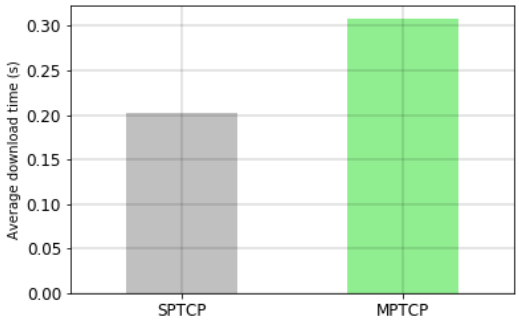
Table 4.3 and Figure 4.13 show the performance of SPTCP and MPTCP in terms of the average download time for different file sizes when the bandwidth of the second path links is heterogeneous, the table shows that MPTCP outputs almost equal MPTCP outputs that are illustrated in the fourth column of the previous table. This scenario aims to prove that the bandwidth for a path equals the minimum path links bandwidth.



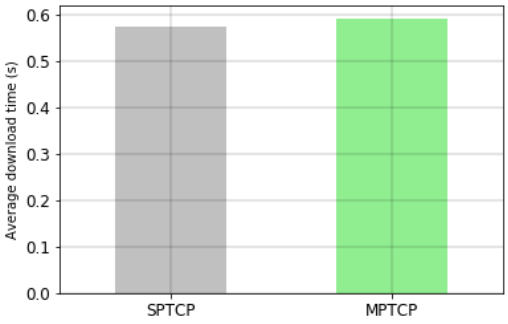
(a) File size equals 50 KB



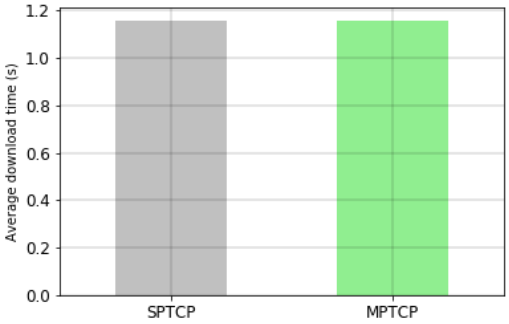
(b) File size equals 100 KB



(c) File size equals 200 KB



(d) File size equals 500 KB



(e) File size equals 1 MB

Figure 4.12: Average download complete time for different file sizes when the bandwidth of the second path links is heterogeneous.

File Size	SPTCP	MPTCP
50 KB	0.0379 sec.	0.0941 sec.
100 KB	0.0874 sec.	0.201 sec.
200 KB	0.202 sec.	0.308 sec.
500 KB	0.576 sec.	0.592 sec.
1 MB	1.157 sec.	1.155 sec.

Table 4.3: Average download complete time for different file sizes when the bandwidth of the second path links is heterogeneous.

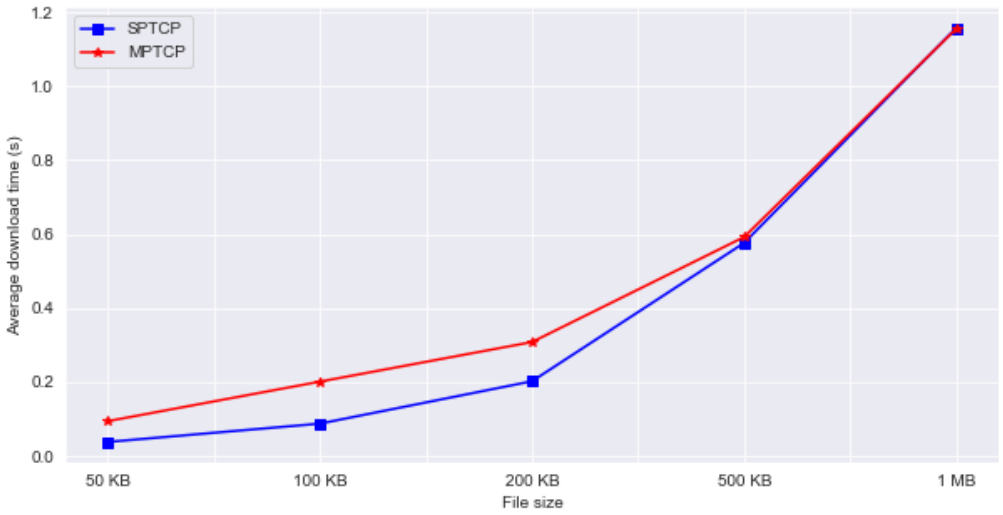


Figure 4.13: Average download complete time for different file sizes when the bandwidth of the second path links is heterogeneous.

Summary:

- MPTCP works better than SPTCP when the bandwidth paths are homogeneous.
- SPTCP performance was better than MPTCP especially when the MPTCP bandwidth gap = 0.99.
- There is a need for a technology that has the capability to select the best paths based on the network condition.
- The bandwidth for a path is equal to the minimum path link bandwidth.
- A significant consideration that must be considered when the routing of MPTCP subflows is the bandwidth gap between paths.

Chapter 5

Proposed Architecture and Experimental Results

Contents

In the previous chapter, we showed that the use of MPTCP will affect badly the transmission of short flows in heterogeneous networks. In this chapter, we will propose an architecture to enhance the impact of MPTCP on short flows. Moreover, This chapter evaluates the proposed architecture and compares the performance of single path TCP (SPTCP), disjoint method (Disjoint), and the proposed architecture (MPSSHetN) in terms of average download complete time with different file sizes.

5.1 Research Methodology

The main objective of this study is to propose a new architecture to improve MPTCP efficiency on short flows, where the proposed architecture considers the gap in terms of bandwidth between the disjoint paths when routing the subflows of MPTCP. We utilize the Software-defined Network (SDN) to select the optimal disjoint paths between the available paths. This work follows the procedure shown below:

- At first, we evaluate the performance of MPTCP on both types of flows (long, short) in the homogeneous and heterogeneous networks as shown in the previous chapter.

- Then we prove that bandwidth dissimilarity between paths is a significant factor and affects MPTCP output on short flows as illustrated in the previous chapter.
- Since the SDN controller has a global view of the topology, we employ the SDN controller to calculate all disjoint paths between the source and destination as well as calculate the bandwidth of the disjoint paths.
- The SDN controller will select the best disjoint paths based on bandwidth dissimilarity of all disjoint paths.
- Utilize the SDN controller to inspect the packet to provide deterministic subflow assignment to paths.

Mininet [12] is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking. Furthermore, it yields a more effective use of time and resources compared to other workflows.

5.2 Proposed Approach (MPSSHetN)

The aim of the suggested architecture, Multipath TCP for Short Flows Supported by SDN in Heterogeneous Networks (MPSSHetN), is to improve the impact of MPTCP on short flows by finding the best disjoint routes and taking into account the paths dissimilarity in term of bandwidth. The suggested architecture consists of three parts as illustrated below and in the Figure 5.1 :

1. Forwarding plane:
   This part presents the SDN infrastructure (Openflow switches), where these devices are simple devices without included control or software to take self-actions. The SDN switches forward the traffic based on the flow tables that are installed on them by the controller. The flow table entry has three parts: matching rule, actions, and counters. The flow rules are a set of several matching fields, and the popular rule for SDN switches is to send the packets to the SDN controller. When the SDN switches receive new packets, they first check for the tables they have to match. Potential actions include forwarding the packet to the outgoing port(s), forwarding the packet to the handler, or dropping the packet. Based on Figure 5.1, when the ingress switch receives the first packet, will send the packet to the SDN controller via Packet in message.

2. OpenFlow protocol:
   It is a bridge linking the control and forwarding devices, and it offers a communication channel between the SDN controller and the SDN switches.

3. SDN controller:
   This part is the most critical part of the proposed architecture. Many general network services can be supported by the SDN controller, such as network status monitoring, network topology data collection, allocation of the configuration for

distributed devices. The SDN controller is responsible for creating the network configuration according to the policies defined by the network engineers. The control plane determines how to handle network traffic, and the data planes send the traffic according to the controller's rules. The main job of the SDN controller is to find the best disjoint routes as well as assigning the suitable OpenFlow rules to the switches. This part contains two main modules:

- Topology module:
  Provides the forwarding module with a collection of disjoint paths between hosts and their bandwidth.
- Forwarding module:
  The purpose of this module is to select the optimal disjoint paths from the disjoint paths set and to provide the switches with the suitable OpenFlow rules.

Figure 5.1 shows the interaction between the three parts when the SDN switch receives the first packet and doesn't have the rules, the packet will be forwarded via packet-in message to the controller, where the function of the rules is to inform the switch where the packet should be sent. Then the SDN controller will extract the MPTCP options from the MPTCP header. Once the controller sees the MP_CAPABLE option in the header. It will know the switch is asking about the first path. The controller will send the flow tables to the switch via packet-out message.

Once the MPTCP client starts to add a new subflow, in the same way, the packet will be forward to the controller by the switch through packet-in message and the controller will parse the MPTCP header, once it sees the MP_JOIN option it will evaluate the next disjoint path based on the value of the difference between the MPTCP BW gap for disjoint path and threshold gap and based on the result the controller will determine whether to select this path or not. In case the path is selected, the controller will send the flow tables to the switch via packet out-message.

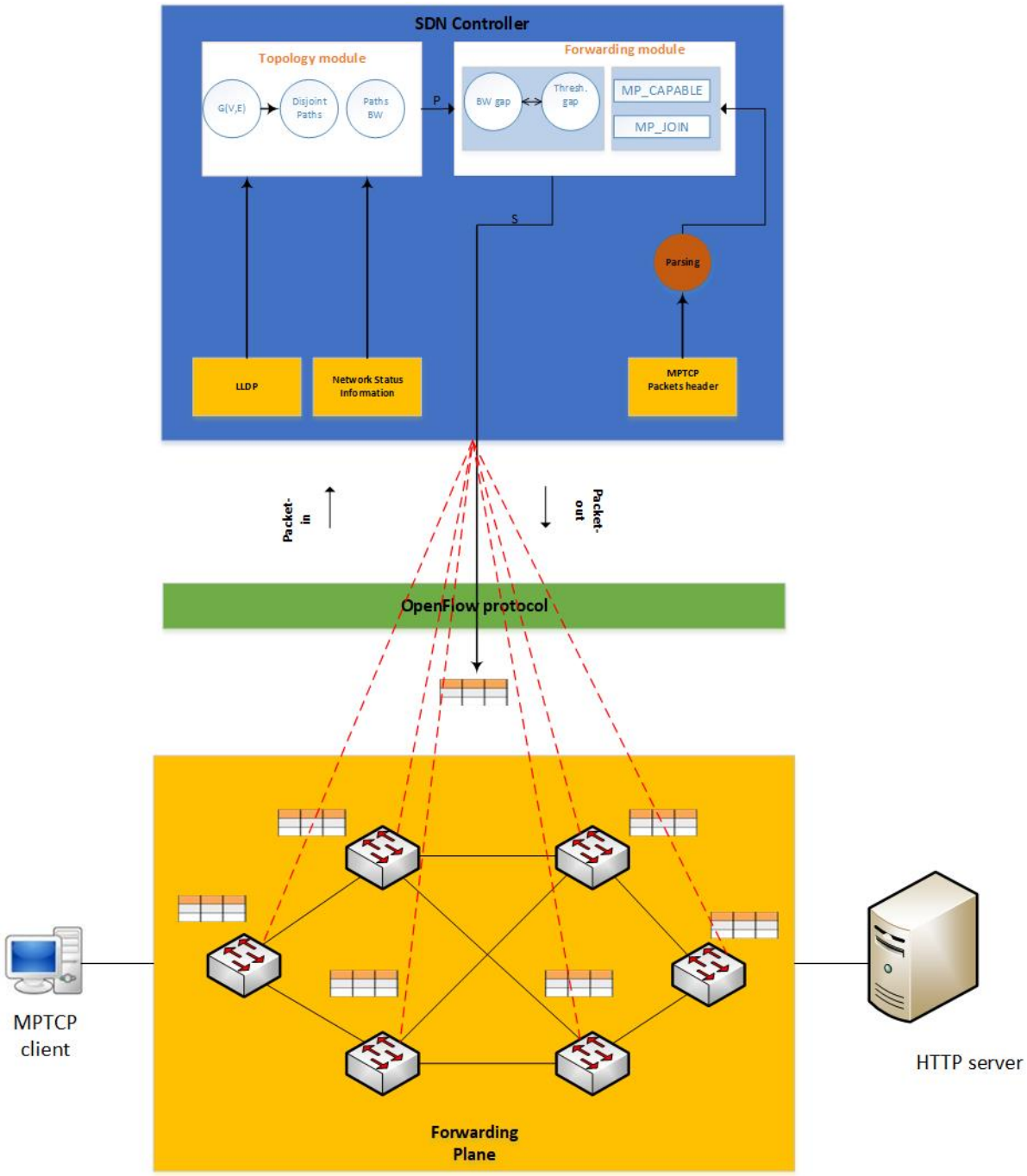


Figure 5.1: Proposed MPSSHetN architecture .

5.2.1 Topology module

This module has an up-to-date global view of the topology. This is done by messages being transferred between the SDN controller and the switches. Pox controller uses openflow.discovery component [36], this component transmits LLDP (Link Layer Discovery Protocol) messages to switches to get the information about the links that can be used to create a topology graph. To save the topology graph G(V, E) and to find all the disjoint paths, we use networkx tool [37].

Algorithm 1: Disjoint paths set

Input: Network topology G(V,E), Source node S, Destination node D
Output: Disjoint Pathlist P
for *i in nx.edgedisjointpaths (G,S,D):* **do**
  Add $path_i$ to Pathlist P
end
return Pathlist P

In the algorithm 1, the network topology, source node, and the destination node are the inputs, the disjoint pathlist P is the output. The objective of this algorithm is to find all disjoint paths between S and D and save them in the pathlist P.

The bandwidth available for a path is measured as illustrated in equation 5.1:

$$BW_j = Min\{BW_e, e \in j\} \tag{5.1}$$

where:

j is the path from source node S to destination node D.

e is every link of the path j.

The objectives of the topology module are:

- Provides the forwarding module with a collection of disjoint paths (P).
- Provides the forwarding module with the bandwidth of the disjoint paths.

5.2.2 Forwarding module

This module aims to select the optimal disjoint paths based on the network condition to improve the performance of MPTCP over short flows, as well as make the deterministic assignment of subflows to paths. The forwarding module will install the suitable OpenFlow rules to the switches after the selection process of optimal paths.

When the ingress switch forwards the first packet to the forwarding module, it will extract MPTCP options from the MPTCP header to determine whether the type is

MP_CAPABLE or MP_JOIN. If the type is MP_CAPABLE, it will select the first disjoint path. When the type is MP_JOIN, for every MP_JOIN packet, the forwarding module will calculate the MPTCP BW gap for every disjoint path and compare that value with the threshold gap, and then it will decide to choose this disjoint path or not based on that value.

Algorithm 2 offers a brief overview of the all processes of the forwarding module. The purpose of the algorithm is to find a set of paths that will boost MPTCP's output on short flows. Pathlist P, MPTCP options, Bandwidth BW_j of path j in Pathlist P, and $Threshold_{gap}$ are the inputs, where the Pathlist P is provided to the forwarding module by the topology module.
Once the forwarding module receives the first packet from the switch, the forwarding module will inspect the MPTCP options to identify if the option is MP_CAPABLE or MP_JOIN. In case the option is MP_CAPABLE, the forwarding module will select the first path (s1) and send the rules regarding the first path (s_1) to the switches, and then update the pathlist p (removing the first path from pathlist p)

Note: we assume the first path is the reference path in terms of bandwidth and the disjoint of the other paths will compare with it to calculate the BW gap.

Algorithm 2: Forwarding Module

Input: Pathlist P = p_1, p_2,.... , p_n, MPTCP packet option, Bandwidth BW_j of path j in Pathlist P, $Threshold_{gap}$
Output: A set of paths S = s_1, s_2,.... , s_n
parsing MPTCP packet header
if *option == MP_CAPABLE* **then**
  select the first path (s_1)
  send OpenFlow rules regarding the first path (s_1) to the switches
end
update Pathlist P
if *option == MP_JOIN* **then**
  for $path_j$ *in P* **do**
    calculate MPTCP BW gap_j
    if *MPTCP BW* $gap_j \leq Threshold_{gap}$ **then**
      select $path_j$ (s_i)
      send OpenFlow rules regarding $path_j$ (s_j) to the switches
    end
  end
end

If the option is MP_JOIN, the forwarding module will calculate MPTCP BW *gap* (as illustrated in equation 5.2) for all paths existed in the path list, and then compares MPTCP BW *gap* for each path with $Threshold_{gap}$. If MPTCP BW *gap* for path j is less than or equal $Threshold_{gap}$, the forwarding module will choose this (s_j) path and will send OpenFlow rules regarding this path to the switches.

$$BWgap = \frac{|BWp_1 - BWp_j|}{BWp_1} \tag{5.2}$$

where:

BWp_1 is the first path bandwidth (reference path).

BWp_j is the disjoint path bandwidth.

We assume the application size (Fi) is known in the proposed architecture, table 5.1 illustrates the relationship between the BW threshold and the application size approximately:

Application Size	BW threshold
Fi < 50 KB	0
50 KB ≥ Fi < 100 KB	0.4
100 KB ≥ Fi < 200 KB	0.6
200 KB ≥ Fi < 500 KB	0.7
Fi ≥ 500 KB	0.9

Table 5.1: The relationship between the BW threshold and the file size (Fi).

Based on the experiment results, we found that when the application size is less than 50KB, the disjoint paths must be identical in terms of bandwidth, otherwise the MPTCP will be affect negatively. As for the other application sizes, when selecting the disjoint paths, the table defines the threshold that the disjoint paths must not exceed between them in terms of variations in bandwidth.

5.3 Exchanging MPTCP packets in the proposed architecture

This section describes how MPTCP packets are handled through the proposed architecture (MPSSHetN). As illustrated in Figure 5.2 there are mainly two stages:

1. Establishing a MPTCP connection:
   Assuming the MPTCP client is willing to download a file from the MPTCP server and the OpenFlow switch has no knowledge about the network.

   - The MPTCP client transmits a SYN packet which carries the MP_CAPABLE option.

- Once the SDN switch has received the packet, the packet will be forwarded via packet-in message to the controller.
- Once the controller received the packet, the packet will be parsed to ensure the MP_CAPABLE option is included in the packet, where the aim of this option is to checks if the client is boosted with MPTCP.
- The controller will select the first disjoint path, after that the controller will send the flow tables to the switches.
- After completing the three handshaking process over the first path, the initiation of the MPTCP connection process will finish.

2. Add a new subflow:

   When the MPTCP client made sure the MPTCP server is MPTCP-enabled, in this step the MPTCP client is willing to add a new subflow.

   - The MPTCP client transmits a SYN packet which carries the MP_JOIN option.
   - Once the SDN switch has received the packet, the packet will be forwarded via packet-in message to the controller.
   - Once the controller has received the packet, the packet will be parsed to ensure the packet includes MP_ JOIN option.
   - The controller will evaluate the MPTCP BW gap between the first path (Reference path) and the next path, while the gap is less than or equal to the threshold, the controller will send the flow tables to the switches regarding the next path.
   - After that, the three handshaking process for adding a new subflow will finish, and this process will be repeated once there are adding a new subflow on the other disjoint paths.

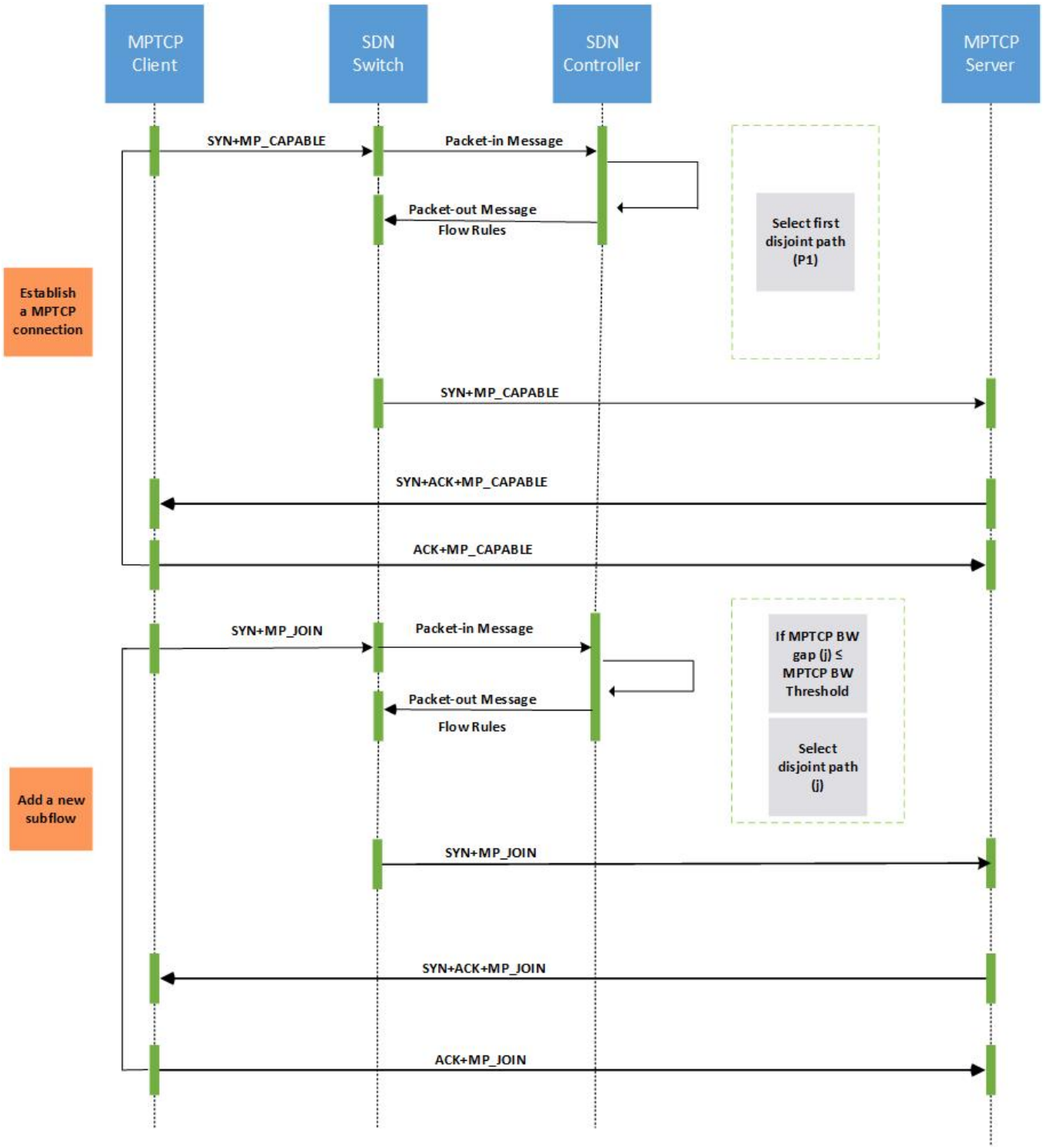


Figure 5.2: Exchanging MPTCP packets in MPSSHetN.

5.4 Experimental Setup

Figure 5.3 presents the testbed for evaluating the proposed architecture, the bandwidth between the switches is illustrated in this Figure. We assume the MPTCP client and MPTCP server are connected to the switch through three links. The testbed has been carried on Ubuntu 16.04 and the MPTCP release is v0.95, the testbed involves the following:

- Mininet [38]: to emulate the network topology.
- Open vSwitch [39] (OvS) 2.5.5: used to emulate the SDN switches.
- POX [14] : used as SDN controller.

The below Figures 5.4,5.5 show the performance of the regular TCP, Disjoint method [7], and the proposed architecture (MPSSHetN) in terms of average download complete time with different application sizes, all Figures results are the average of 100 samples.

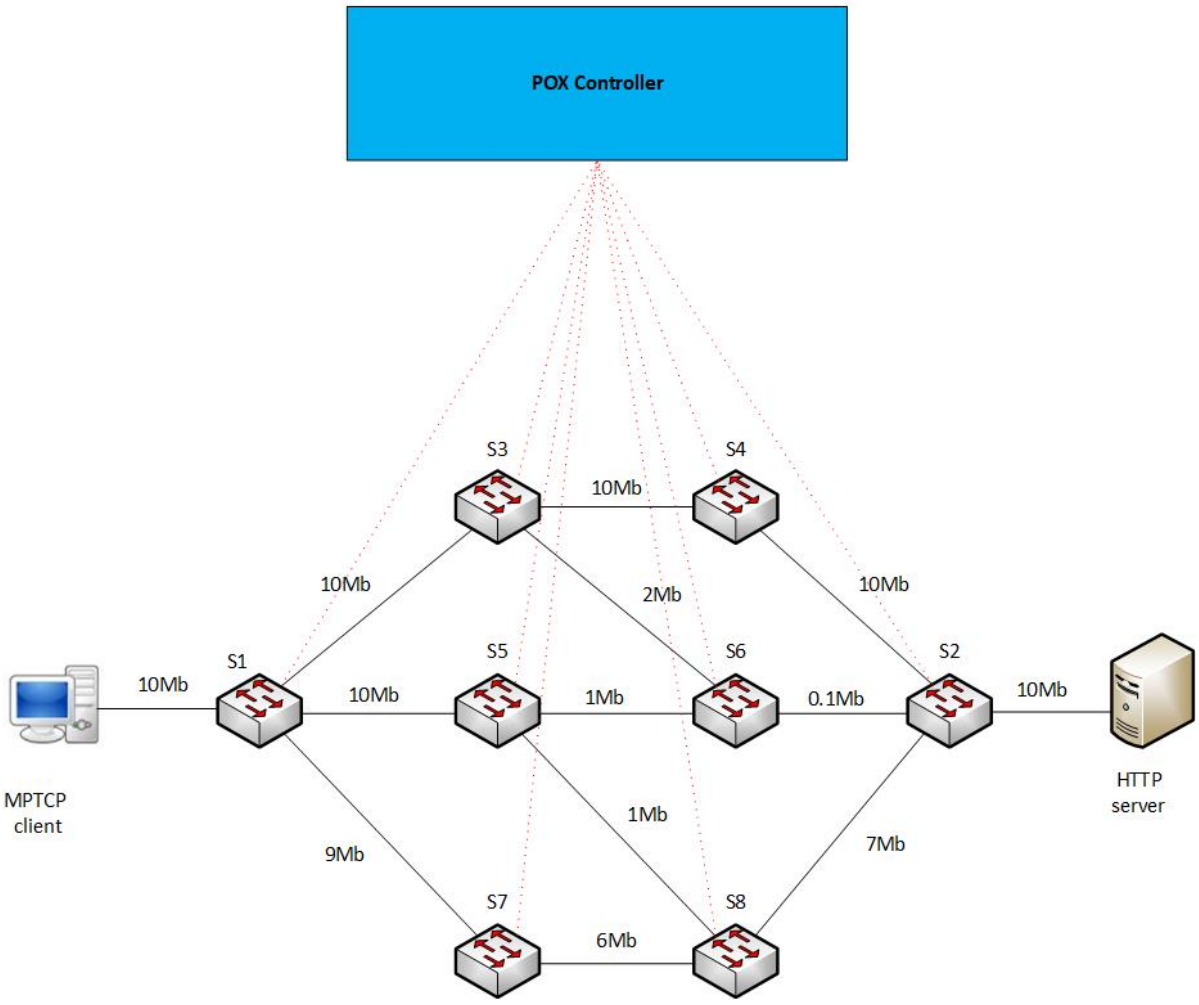
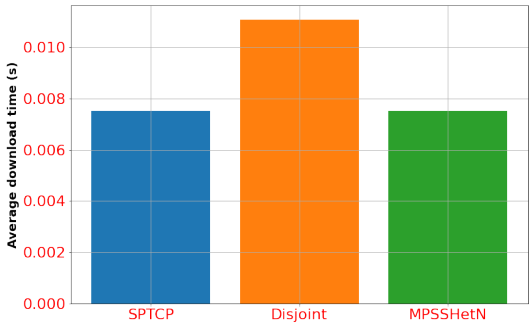


Figure 5.3: Experimental Testbed

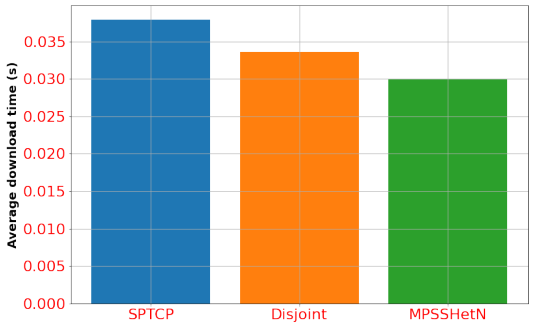
Figure 5.4a shows that when the application size 10 KB, the performance of SPTCP and MPSSHetN are equal and they are better than the disjoint method since the average download time for SPTCP and MPSSHetN is 0.00752 second while for the disjoint method is 0.01108 second. Figure 5.4b shows also that when the application size 50 KB, the performance of MPSSHetN is best than both SPTCP and disjoint method since the average download time for MPSSHetN is 0.0299 second while for SPTCP is 0.0379 second and for the disjoint method is 0.0336 second. Similarly in Figure 5.4c where the application size 100KB, the performance of MPSSHetN is best than both SPTCP and disjoint method since the average download time for MPSSHetN is 0.0504 second while for SPTCP is 0.0874 second and 0.0712 second for the disjoint method. The same result can be observed in Figure 5.4d where the application size 200 KB, the performance of MPSSHetN is best than both SPTCP and disjoint method since the average download time for MPSSHetN is 0.1 second while for SPTCP is 0.202 second and for the disjoint method is 0.111 second.

Figure 5.5a shows that when the application size 500 KB, the performance of MPSSHetN is best than both SPTCP and disjoint method, as well as normal TCP performs better than the disjoint method (The average download time for MPSSHetN is 0.305 second while for SPTCP is 0.576 second, for the disjoint method is 1.406 second). The same result can be seen in Figure 5.5b where the application size is 1 MB, the performance of MPSSHetN is higher than both the SPTCP and disjoint method since the average download time for MPSSHetN is 0.624 second while for SPTCP is 1.157 second, for the disjoint method is 1.649 second. Figure 5.5c shows that when the application size 2MB, the performance of MPSSHetN is best than both SPTCP and disjoint method, as well as the disjoint method performs better than SPTCP since the average download time for MPSSHetN is 1.274 second while for SPTCP is 2.544 second, for the disjoint method is 1.749. Likewise in Figure 5.5d where the application size 5 MB, the performance of MPSSHetN is best than both SPTCP and disjoint method, as well as the disjoint method performs better than SPTCP since the average download time for MPSSHetN is 3.246 second while for SPTCP is 6.558 second and for the disjoint method is 3.741 second.

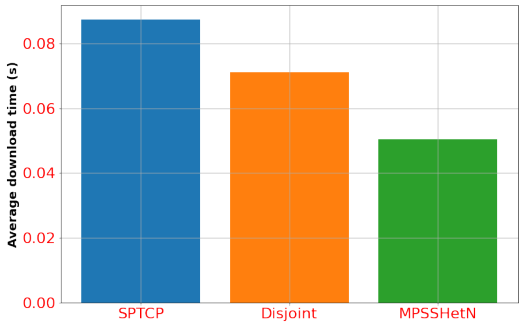
Table 5.2 shows the Comparison between the performance of normal TCP, Disjoint method, and MPSSHetN in terms of average download time for various application sizes.



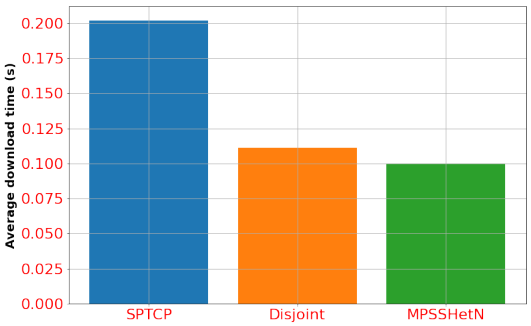
(a) Application size equals 10KB



(b) Application size equals 50 KB

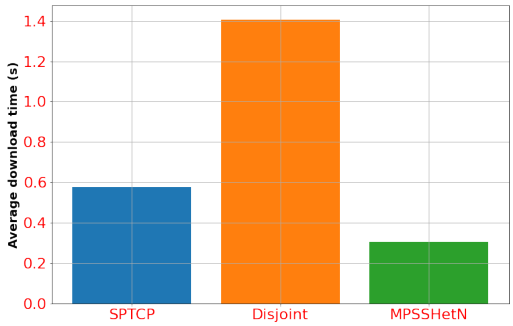


(c) Application size equals 100 KB

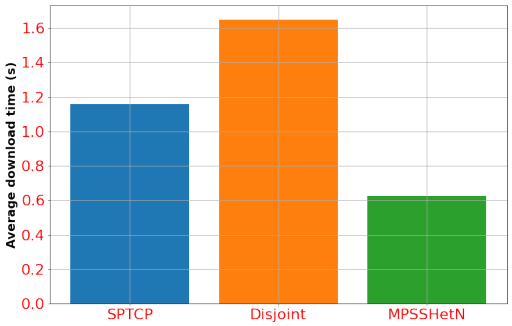


(d) Application size equals 200 KB

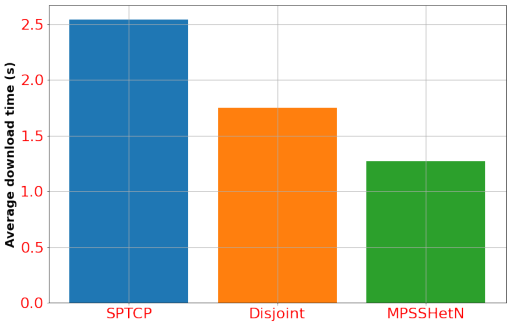
Figure 5.4: Average download complete time when application sizes equal 10 KB, 50 KB, 100 KB, and 200 KB.



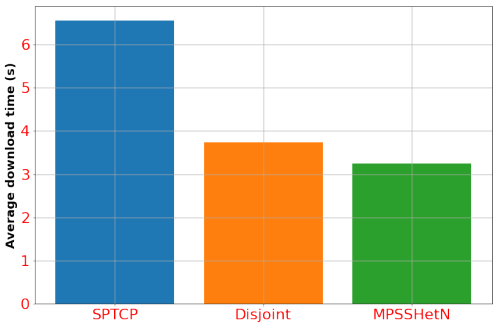
(a) Application size equals 500 KB



(b) Application size equals 1 MB



(c) Application size equals 2 MB



(d) Application size equals 5 MB

Figure 5.5: Average download complete time when application sizes equal 500 KB, 1 MB, 2 MB, and 5 MB.

File Size	SPTCP	Disjoint	MPSSHetN
10 KB	0.00752 sec.	0.01108 sec.	0.00752 sec.
50 KB	0.0379 sec.	0.0336 sec.	0.0299 sec.
100 KB	0.0874 sec.	0.0712 sec.	0.0504 sec.
200 KB	0.202 sec.	0.111 sec.	0.1 sec.
500 KB	0.576 sec.	1.406 sec.	0.305 sec.
1 MB	1.157 sec.	1.649 sec.	0.624 sec.
2 MB	2.544 sec.	1.749 sec.	1.274 sec.
5 MB	6.558 sec.	3.741 sec.	3.246 sec.

Table 5.2: Comparison between the performance of normal TCP, Disjoint method, and MPSSHetN in terms of average download time for various application sizes.

5.5 Summary and Discussion

As per the topology that is shown in Figure 5.3, there are three disjoint paths between the MPTCP client and the MPTCP server. The disjoint paths and their bandwidth are presented in table 5.3.
The topology module will provide the forwarding module with the disjoint paths and their bandwidth. The forwarding module will select the best paths based in the BW gap and application size.

Path	Path Distribution	Path Bandwidth	Bandwidth gap
P1	S1, S3, S4, S2	10 Mb	0
P2	S1, S5, S6, S2	0.1 Mb	0.99
P3	S1, S7, S8, S2	6 Mb	0.4

Table 5.3: The disjoint paths between the MPTCP client and the MPTCP server and paths bandwidth

As per Figure 5.4a, when application size equals 10 KB, the proposed method (MPSSHetN) will select just the first path (P1), this means when the forwarding module extracts the MPTCP header and once the option is MP_CAPABLE option it selects the first path. Once the option is MP_JOIN, the forwarding module will calculate the BW gap for each disjoint path as depicted in the table, and because there is no disjoint path that has zero dissimilarity compared with the first path in terms of bandwidth, the forwarding module will not select any of the rest of the disjoint paths. Moreover, it is noted the bad impact

of the disjoint method because it neglected the BW gap factor between the disjoint paths.

According to the Figures (5.4b, 5.4c, 5.4d, 5.5a, 5.5b, 5.5c, and 5.5d), for the rest of the application sizes, the proposed method selects the first and third path (P1 and P3). In the same way, when the forwarding module extracting the MPTCP header and once the option is MP_CAPABLE, it selects the first path (P1). Similarly, the first path (P1) is selected when the forwarding module extracts the MPTCP header and the option is MP_CAPABLE. Once the option is MP_JOIN, the forwarding module will calculate the BW gap for each disjoint path and will compare it with the threshold gap. Since the BW gap for P3 is less than the threshold gap, the forwarding module will select P3 and omit P2. Furthermore, it is noted in the Figures the negative effect of the disjoint method because it omitted the BW gap factor between the disjoint paths.

Chapter 6

Conclusion and Perspective

MPTCP is a promising technology and its main goals are to provide better performance, throughput, and resilience to failures by splitting the flow into several subflows which will be then sent over many paths. MPTCP has a positive effect on long flows in terms of throughput such as transmitting large-sized files. However, MPTCP may degrade the efficiency of short flows which are sensitive to latency as web transfer applications especially when the short flows are transferring over heterogeneous paths, that because the paths heterogeneity causes packet reordering. Moreover, MPTCP is an end-to-end protocol that cannot observe the state of lower layers in the network.
The administration of conventional networks is complex. This reduces network infrastructure growth, where network operators need to manually configure each network unit. Software-defined networking (SDN) decouples the control plane from the data plane and transfer the network devices to be just normal forwarding devices receive the instructions from the control plane. Furthermore, SDN shifts the control logic to a logically centralized controller. The controller guides the data plane components through an application programming interface, as well as the controller, has a global view of the network. SDN has many applications like rural connections, Internet research, data centers upgrading, and traffic engineering.

This thesis investigated the impact of MPTCP on long flows and short flows in homogenous and heterogeneous networks. For long flows, the experiment results showed that the performance of MPTCP is better than SPTCP when the bandwidth paths are homogeneous. In the majority of heterogeneous scenarios, the MPTCP throughput is often higher than the SPTCP throughput. Moreover, the results proved that reducing the heterogeneity in terms of paths bandwidth will lead to increasing the average throughput. As for short flows, the experiment results proved that MPTCP works better than SPTCP when the bandwidth paths are homogeneous. However, when the bandwidth of the paths is heterogeneous, the SPTCP performance was better than MPTCP in some scenarios. This is determined based on the bandwidth gap value and the application size. In addition to that, the experiments proved that the bandwidth gap between paths is a critical factor influencing the performance of the short flows and must be considered when routing the MPTCP subflows.

This thesis proposed a new architecture supported by SDN (MPSSHetN) to improve

the performance of MPTCP for short flows in the heterogeneous networks. MPSSHetN takes into account the bandwidth dissimilarity among paths for routing MPTCP subflows. MPSSHetN includes two modules: the topology module and the forwarding module, where the function of the topology module is to calculate all disjoint paths between hosts and their bandwidth and the role of the forwarding module is to select the best disjoint routes based on the least bandwidth gap. Our approach results showed the improvement of MPTCP performance for short flows compared to the disjoint approach.
Studying the effect of the dissimilarity of the paths in terms of delay and modifying our proposed architecture to select the best routes based on the bandwidth gap and delay gap will be part of our future work. Moreover, we plan to implement our approach over real-time video applications.

Bibliography

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 63–74, ACM, 2008.

[2] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] A. Ford, C. Raiciu, M. Handley, S. Barre, J. Iyengar, *et al.*, "Architectural guidelines for multipath tcp development," *IETF, Informational RFC*, vol. 6182, pp. 2070–1721, 2011.

[4] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "Tcp extensions for multipath operation with multiple addresses," tech. rep., 2013.

[5] W. Yang, P. Dong, W. Tang, X. Lou, H. Zhou, K. Gao, and H. Wang, "A mptcp scheduler for web transfer," *CMC-COMPUTERS MATERIALS & CONTINUA*, vol. 57, no. 2, pp. 205–222, 2018.

[6] Q. Wang, G. Shou, Y. Liu, Y. Hu, Z. Guo, and W. Chang, "Implementation of multipath network virtualization with sdn and nfv," *IEEE Access*, vol. 6, pp. 32460–32470, 2018.

[7] S. Zannettou, M. Sirivianos, and F. Papadopoulos, "Exploiting path diversity in datacenters using mptcp-aware sdn," in *Computers and Communication (ISCC), 2016 IEEE Symposium on*, pp. 539–546, IEEE, 2016.

[8] K. Yedugundla, S. Ferlin, T. Dreibholz, Ö. Alay, N. Kuhn, P. Hurtig, and A. Brunstrom, "Is multi-path transport suitable for latency sensitive traffic?," *Computer Networks*, vol. 105, pp. 1–21, 2016.

[9] S. Habib, J. Qadir, A. Ali, D. Habib, M. Li, and A. Sathiaseelan, "The past, present, and future of transport-layer multipath," *Journal of Network and Computer Applications*, vol. 75, pp. 236–258, 2016.

[10] M. Kheirkhah, I. Wakeman, and G. Parisis, "Mmptcp: A multipath transport protocol for data centers," in *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*, pp. 1–9, IEEE, 2016.

[11] J. A. Cordero, "Multi-path tcp performance evaluation in dual-homed (wired/wireless) devices," *Journal of Network and Computer Applications*, vol. 70, pp. 131–139, 2016.

[12] "Mininet." `http://http://mininet.org`. Accessed on 2019-01-02.

[13] "multipath tcp-linux kernel." `http://multipath-tcp.org/pmwiki.php/Main/HomePage`. Accessed on 2019-01-10.

[14] "Pox controller." `https://noxrepo.github.io/pox-doc/html/`. Accessed on 2019-02-05.

[15] K. Greene, "10 breakthrough technologies: Software-defined networking mit technol., Online. Available:," 2009. `http://www2.technologyreview.com/article/412194/tr10-software-defined-networking/`,.

[16] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[17] B. Boughzala, R. B. Ali, M. Lemay, Y. Lemieux, and O. Cherkaoui, "Openflow supporting inter-domain virtual machine migration," in *Wireless and Optical Communications Networks (WOCN), 2011 Eighth International Conference on*, pp. 1–7, IEEE, 2011.

[18] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[19] Q. Peng, A. Walid, and S. H. Low, "Multipath tcp algorithms: theory and design," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, pp. 305–316, ACM, 2013.

[20] P. Dong, J. Wang, J. Huang, H. Wang, and G. Min, "Performance enhancement of multipath tcp for wireless communications with multiple radio interfaces," *IEEE Transactions on Communications*, vol. 64, no. 8, pp. 3456–3466, 2016.

[21] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath tcp: Analysis, design, and implementation," *IEEE/ACM Transactions on networking*, vol. 24, no. 1, pp. 596–609, 2014.

[22] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath tcp schedulers," in *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, pp. 27–32, ACM, 2014.

[23] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp.," in *NSDI*, vol. 11, pp. 8–8, 2011.

[24] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "Mptcp is not pareto-optimal: performance issues and a possible solution," *IEEE/ACM Transactions on Networking (ToN)*, vol. 21, no. 5, pp. 1651–1665, 2013.

[25] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath tcp," in *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, IEEE, 2012.

[26] J. Hwang and J. Yoo, "Packet scheduling for multipath tcp," in *2015 Seventh International Conference on Ubiquitous and Future Networks*, pp. 177–179, IEEE, 2015.

[27] L. Li, N. Hu, K. Liu, B. Fu, M. Chen, and L. Zhang, "Amtcp: an adaptive multipath transmission control protocol," in *Proceedings of the 12th ACM International Conference on Computing Frontiers*, p. 29, ACM, 2015.

[28] R. Barik, M. Welzl, S. Ferlin, and O. Alay, "Lisa: A linked slow-start algorithm for mptcp," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2016.

[29] P. Dong, W. Yang, W. Tang, J. Huang, H. Wang, Y. Pan, and J. Wang, "Reducing transport latency for short flows with multipath tcp," *Journal of Network and Computer Applications*, vol. 108, pp. 20–36, 2018.

[30] F. Németh, B. Sonkoly, L. Csikor, and A. Gulyás, "A large-scale multipath playground for experimenters and early adopters," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 481–482, ACM, 2013.

[31] Z. Jiang, Q. Wu, H. Li, and J. Wu, "scmptcp: Sdn cooperated multipath transfer for satellite network with load awareness," *IEEE Access*, vol. 6, pp. 19823–19832, 2018.

[32] A. Hussein, I. H. Elhajj, A. Chehab, and A. Kayssi, "Sdn for mptcp: An enhanced architecture for large data transfers in datacenters," in *Communications (ICC), 2017 IEEE International Conference on*, pp. 1–7, IEEE, 2017.

[33] J. Pang, G. Xu, and X. Fu, "Sdn-based data center networking with collaboration of multipath tcp and segment routing," *IEEE Access*, vol. 5, pp. 9764–9773, 2017.

[34] A. A. Barakabitze, L. Sun, I.-H. Mkwawa, and E. Ifeachor, "A novel qoe-centric sdn-based multipath routing approach for multimedia services over 5g networks," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2018.

[35] K. Gao, C. Xu, J. Qin, S. Yang, L. Zhong, and G.-M. Muntean, "Qos-driven path selection for mptcp: A scalable sdn-assisted approach," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2019.

[36] "Pox controller manual." `https://openflow.stanford.edu/display/ONL/POX+Wiki.html`. Accessed on 2019-02-09.

[37] "networkx tutorial." `https://networkx.github.io/documentation/stable/`. Accessed on 2020-01-07.

[38] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19, ACM, 2010.

[39] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Open vswitch: Extending networking into the virtualization layer," *Proc. of ACM SIGCOMM HotNets*, 2009.