Arab American University – Jenin

Faculty of Graduate Studies

Performance evaluation of multipath TCP congestion control algorithms

By

Sameer Suleiman Zuhairi

Supervisor

Dr. Mohammad M. N. Hammarsheh

This thesis was submitted in partial fulfillment of the requirements for the master in computer science

© Arab American University – Jenin 2022. All rights reserved.

THESIS APPROVAL

PERFOMANCE EVALUATION OF MULTIPATH TCP CONGESTION CONTROL ALGORITHMS

By
Sameer Suleiman Zuhairi

This thesis was defended successfully on June 26th 2022 and approved by:

Examination Committee Members	**Signature**
Dr. Mohammad M. N. Hammarsheh *Chairperson of Committee*	
Dr. Adwan Yasin *Internal Examiner*	
Dr. Saeed Salah *External Examiner*	

DECLERATION

I, the undersigned, declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

Candidate Name: Sameer Suleiman Zuhairi

Signature: 

Date: 08-11-2022

Dedication

Praise and appreciation to God for all of his blessings and for the production of knowledge that would not have been possible without this thesis. This is for my father, Suleiman Zuhairi, who opened the way for me to get to where I am now. Thank you for every second you battle and strive for us. This thesis is the consequence of our pledge to you that we would follow your example in becoming a good man. To my mother, the lady of dignity, honesty, and kindness, your hug served as a haven for me; I cherish every word you taught us and we learn from you. This is my recompense for the long hours you endured caring for me when I was small. To my dearest wife, who has brought me comfort and happiness by providing me with hope and support. To my brothers, sisters, and all my friends who never cease to support me, I wish God would continue to bless them and grant them their heart's desires.

Acknowledgments

I would like to express my sincerest appreciation to my supervisor, Dr. Mohammad Hammarsheh, for his constant encouragement, support, and direction throughout the completion of my thesis.

Additionally, I would like to express my profound gratitude and admiration to the Faculty of Engineering and Information Technology. I would like to express my gratitude to my friends for their encouragement and assistance throughout this process.

I would like to express my gratitude to my family for their unwavering support and encouragement, as well as for always being there when I needed them. Finally, but certainly not least, I would like to express my gratitude to my friend Khaled Al- Khatib, who began this trip alongside me and has consistently supported me, I wish him success.

Contents

Abbreviations

5G	**Fifth wireless mobile generation**
IETF	**The Internet Engineering Task Force**
WIFI	**Wi-Fi is a family of wireless network protocols, based on the IEEE 802.11**
IP	**Internet Protocol**
MPATH	**Multipath TCP**
WLAN	**Wireless Local Area Networks**
TCP	**Transport Control Protocol**
Taho	**One of Loss based congestion control algorithms.**
CSMA/CA	**Carrier Sense Multiple Access with Collision Avoidance**
ACK	**Acknowledgment**
Vegas	**Vegas is a TCP congestion avoidance algorithm that emphasizes packet delay, rather than packet loss**
PING	**a computer network administration tool used to check whether a host is reachable**
OLIA	**One of multipath TCP congestion control algorithm**
BALIA	**One of multipath TCP congestion control algorithm**
WVEGAS	**One of multipath TCP congestion control algorithm**
Regular	**Single Path TCP**
Reno	**One of Loss based congestion control algorithms**
OS	**Operating system**
QoS	**Quality of Service**
IPv4	**Internet Protocol Version 4**
IEEE	**Institute of Electrical and Electronics Engineers**

Abstract

The congestion control (CC) of the network has a considerable importance over all network applications. It is proved that Internet Service Providers (ISPs) need to develop a better understanding of modern concepts and strategies of the CC. For instance, whenever network design complexity increases the variety of congestion control algorithms (CCA) designs rises. These algorithms are important to enhance the network performance.

The Internet Engineering Task Force (IETF) developed Multipath TCP CC. It enables the distribution of a single data stream across many paths. This strategy can improve the network's availability and bandwidth. It can also lead to efficient network usage and high utilization without being aggressive with competing flows.

The behavior of a few of the best multi-path congestion control algorithms (MPTCP CCA) was investigated in this research. The purpose of this research is to describe the optimal performance of MPTCP CCA in a variety of situations, followed by the optimal implementation of the chosen algorithm. The optimal algorithm must perform better than single-path TCP in terms of throughput and fairness, and it must be capable of being used in place of TCP. Furthermore, we demonstrate that some multipath CCA might be damaging to the network's overall performance.

Depending on a deep analysis of literature review, a performance analysis scenarios were built to investigate which is MPTCP CCA design has a better behavior on most situations. The study finding hypotheses were tested by using six MPTCP CCA in many situations. The study findings prove that none of the considered multipath algorithms is ideal. The MPTCP CCA is widely applicable and can be used to determine the behavior of a multipath variation of an improved CCA.

We have demonstrated that MP transmission through MPTCP provides a significant performance improvement in certain scenarios. Furthermore, we found that the cubic CCA outperforms MPTCP OLIA, MPTCP BALIA, MPTCP, wVegas and regular TCP in all circumstances except when MPTCP is used without competing with other algorithms. Thus, even if we use many ISPs in our scenario, there will always be some costs associated with using MP techniques. However, the total performance benefits associated with MP transport remains significant. Each of these methods is implemented and evaluated on a Linux environment. Additionally, we evaluate it for multi-homed servers.

In this study, the researchers used throughput as main metric, because it is a clear goal of most CCA mechanisms to maximize throughput, throughput and goodput are sometimes distinguished from one another. While throughput is the link utilization or flow rate in bytes per second, goodput is the subset of throughput made up of useful traffic and is also measured in bytes per second. The increasing throughput is important in a variety of environments, including overloaded and underused networks as well as long-lived and short-lived flows. For instance, throughput has been assessed in terms of the transfer times for connections with a variety of transfer sizes in order to evaluate Quick-Start, a proposal to enable flows to start up more quickly than slow-start.

The researchers used ifstate tool to calculate the throughput in this study, the ifstat tool displays statistics for network interfaces. The interface maintains copies of the previous information shown in history files. It shows the difference between the most recent and current calls by default.

Except the fourth scenario, MPTCP Cubic outperforms all other algorithms, resulting in an average throughput of 1908 (kb/s) in the first scenario, up 43% from the next-closest algorithm. In the second case, MPTCP Cubic performs better than every other algorithm and achieves a throughput of 1544 (kb/s), an increase of 18.6% over the next-closest algorithm. The third scenario

experienced the same results, but MPTCP Cubic average throughput was slightly higher than the closest algorithm at 1247 (kb/s), an increase of just 14%. In the fourth scenario, all MPTCP algorithms outperform the cubic, and the MPTCP Cubic throughput was 60% less than the best while producing results that were similar to those of the regular algorithm.

Chapter 1: Introduction

1.1 Research Background

The MPTCP is one of the most popular and trusted network algorithms for controlling congestion in multi-paths (MP) networks. This algorithm distributes single traffic into multiple flows and returns to multiple benefits because it distributes traffic into multiple sub-flows, which reduces the possibility of high congestion, packet loss, and time delay (Hijawi & Hamarsheh, 2016). This algorithm is used with multi-homed servers. And MP techniques raise important questions, including how bandwidth would be shared between competing traffic flows if MPTCP is used as CCA protocol. To answer these questions, we need to understand the key fundamental idea of the CCA, which can also be categorized as follows:

When there is no loss, the congestion window will grow in an additive way, but when there is a loss, it will decrease in a multiplicative way. At the start of the connection, its increase will be exponential. This is the basic principle of congestion management algorithms.

MPTCP is made up of many sub-flows, all of which make up the main flow, so each sub-flow has its own congestion window. MPTCP separates packets and sends them in sub-flows, and the congestion window grows when each sub-flow acknowledgment received (Abed, Ismail, & Jumari, 2012). So, this brings up the question of why we use a special algorithm to control the congestion window in a multipath network. As it will be discussed in the coming sections, each flow in MP network has its own congestion window. However, if we use the regular CCA protocol, each sub-flow will be treated as a single flow,

resulting in a doubled congestion window, which will have an impact on the other single flows that share the congestion window. Therefore, in MP network, MPTCP CCA is used to regulate the congestion window on each sub-flow, and this protocol relies on weights in order to calculate the congestion window for each sub-flow. The weight to keep the in taking a percentage of bandwidth granting it to a sub-flow, and the sum of all congestion windows for all sub-flows must not exceed the congestion window on the single path (Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

1.2. Problem Statement

CC is critical for improving network performance. Using various methods of network resource allocation improves throughput. This is accomplished via CCA that employ a variety of techniques and policies. The demands and resource availability are balanced in these CC strategies. It also tries to keep the Quality of Service (QoS) as high as possible. CC for applications within the same LAN, and other integrated networks is complex, making it difficult to find a reasonable solution that does not compromise other criteria such as delay, increasing or decreasing the number of packets sent, preventing new sessions, and service degradation. Because network design can differ from one internet vendor to another, the congestion network problem will be different from one network to another. As a result, there is no universal solution that can be applied to all integrated networks. Thus, the tradeoff is a possible solution to reduce the issue as much as possible in the short term.

Making certain that the fairly use of the available resources is likewise a difficult task. On the other hand, while several experiments and testing measures have been conducted within data centers using MPTCP CCA - which may be utilized in a variety of production systems within data centers - there are still several limitations to its use in large-scale networks.

Based on what was mentioned before, the researcher puts forward the main questions as follows:

- How the network is becomes congested?
- What are the relevant metrics of a CC?
- How do the various metrics of a CC influence the network throughput?
- What is the difference between Regular and MPTCP CCA?
- What are the strategies used in both protocols? What is the best strategy used to control network congestion?
- In MPTCP CCA, how network capacity should be shared efficiently and fairly between competing flows?

To answer the research questions, the researchers used multiple tools to investigate the myths relationships described in the theoretical model after reviewing literature for regular and MPTCP CCA.

The myths related to Regular CC TCP algorithms:

a- Congestion is caused by a shortage of buffer space and will be solved when memory becomes cheap enough to allow infinitely large memories.

b- Congestion is caused by slow links. The problem will be solved when high-speed links become available.

c- Congestion is caused by slow processors. The problem will be solved when the speed of the processors is improved.

The goals of the MPTCP CCA:

a- Multipath-capable flows should be designed so that they shift their traffic from congested paths to uncongested paths (Lubna, Mahmud & Cho, 2018).

b- A multipath sender stripes packets across the available sub-flows (Lubna, Mahmud & Cho, 2018).

c- A multipath flow should perform at least as well as a single-path flow would on the best of the paths available to it (Lubna, Mahmud & Cho, 2018).

d- A multipath flow should not take up any more capacity on any one of its paths than if it was a single path flow using only that route. This guarantees that it will not unduly harm other flows.

1.3. Purpose of the Study

The goal of this thesis is to evaluate the performance of the common MPTCP CCA in a number of situations before recommending the best situations in which this algorithm can be run efficiently, ensuring that the algorithm is at fairest and least aggressive point.

The multi-home internet server is a server with multiple-path network cards that has become very important in the last ten years; and today no organization works with a single-interface server. On the other hand, balancing MP in a multi-homed server is a big issue, so in our research, we compare the performance before and after the balancing.

1.4. Significance of the Study

CC has become increasingly important in order to improve network performance and throughput by allocating network resources. By employing a variety of schemes and rules, these solutions achieve a balance between demand and available resources. While maintaining QoS to the maximum degree possible by reducing end-to-end delay time that may occur after implementing some of the schemes such as Service Denial Schemes, Degradation Schemes, Scheduling Schemes, etc.

The goal of this thesis is to apply CC schemes, then compare regular TCP with the most widely used MPTCP CCA, to help the researchers in their efforts to improve these algorithms.

Numerous studies have been conducted on this topic, and numerous solutions have been proposed to alleviate network congestion. Previous research has revealed that one of the most significant causes of network congestion is memory space consumption, which is initially very limited and can be solved by memory upgrades. Also, the problem may be caused by link speed limitations, which may be resolved by replacing them with faster connections, or processor limitations, which can be resolved by replacing them with faster processors. However, all of these solutions are classified as "static" solutions, which may worsen rather than eliminate the problem. For example, if we assume that switches have infinite memory, the queues and delays will be extremely long, causing packets to wait a very long time in the queue before being sent via links to their destination. Moreover, packets may be lost or resent due to their own expiration. Therefore, static solutions might exacerbate the problem rather than solve it, resulting in lower performance. As a result, the researchers began to focus on solutions that are more dynamic.

Some of these solutions revolve around the implementation of CC strategies and policies, such as the ones listed below:

1. Research resource creation schemes: Such schemes dynamically reconfigure the resource to improve its capacity. Examples of such schemes include:

- Dial-up links that can only be added when there is a lot of usage
- On satellite links, power is increased to increase bandwidth.

Path splitting to send extra traffic along routes that might not be suitable when there is little traffic.

2. Demand reduction schemes: these schemes attempt to lower demand to the available resource level; most of these schemes require the sender to be aware of the network's status condition in order to raise and decrease demand based on the available resource. Some of the operations that this type of scheme can do are summarized below:

- As it is with service denial schemes, do not allow the creation of new sessions.

- Require senders to lower their traffic, adjusts the quantity of transmitted packets based on the current network load.
- This type of operation enforces the senders to schedule their sending packets based on the network load as in scheduling schemes.

1.5. Research Methodology

This study used experimental methods to target the problem statement, shape the hypothesis, draw conclusions and recommendations, and collect and analyze data, as well as allowing for a critical evaluation of the study's overall validity and reliability. We used the ifstat tool for our evaluations. In comparison to other network measuring tools, it has a lot of advantages, ifstat main purpose is to provide a tool for comparing the performance of several transport connections and protocols. That is, by adjusting parameters, it is possible to establish alternative flows between two systems, in order to execute a measurement that has been configured, capture the data and put it through a post-processing process so it can be analyzed statistically. MPTCP, in particular, is supported by IfState. In this research, we use four different kinds of experiments for analyzing the throughput behavior of different CCA, each measurement has a duration of 15 seconds and has been repeated 15 times in order to avoid outliers from background traffic noises.

1.6. Research Structure

This thesis is divided into five chapters:

- The first chapter is an introduction that provides background, problem statement, main hypotheses, purpose, significance, methodology, and structure of the research.
- The second is the literature review having three sections: Section (A): Overview, Section (B): Congestion Control Performance Measurement Metrics, Section (C): Literature Reviews

- The third chapter discusses the method used to meet the objectives.
- The fourth chapter is the empirical section that focuses on the results of the experiments.
- The fifth chapter contains all recommendations, limitations, and conclusions.

Chapter 2: Background and Literature Review

2.1. Regular TCP

2.1.1. Regular TCP CC

The IP layer can make its best effort to deliver the data without loss, but it cannot guarantee that the data is delivered to the right application in the correct order without any errors. However, the TCP layer can guarantee this, and as a result, every segment in the TCP layer has a unique sequence number that can be used later in the application delivery process. Then, TCP will partition the data into segments and placed in the buffer, with each segment receiving a unique sequence number. This sequence number of the segment will be used later by flow control and CC, which employs a sliding window to control the number of segments that must be sent to the receiver in a single transmission.

TCP CC has a major and significant influence on the TCP protocol, which is responsible for byte streaming, and checking the reliability of the data. One of the most important CC goals is network resource management efficiently, which causes less wasting of resources, especially when the resource is competing with other flows at the same time as allowing resource sharing with other flows (Scharf & Ford, 2013). In addition, the delay will have a negative impact on applications like video streaming. The CCA will mitigate this negative impact by reducing the delay and ensuring low latency. Without CC, data loss will increase and throughput will be significantly reduced, which should eventually result in congestion collapse, resulting in extremely low throughput and extremely long delays (Scharf & Ford, 2013).

TCP CC continuously monitors the congestion state and aims to determine the current congestion state; it then uses this information to adjust the sending rate to maximize link utilization without causing data loss, and it distributes available bandwidth fairly among all current flows (Ahmad, Ngadi, & Mohamad, 2015). As a result, the TCP CC always attempts to fully utilize the link without reaching 100 percent utilization, because doing so will result in a congestion state. Also, the TCP CC will increase the congestion window (cwnd) when no congestion event has occurred and decrease the cwnd when a congestion event has occurred (Khalil, 2012). When the new connection is established, the slow start phase begins. During this phase, the cwnd is incremented for each recipient of the Acknowledgement, and when congestion is detected, the connection is entered into a new state called the congestion avoidance state. In this state, the cwnd is increased more slowly than it was in the previous state, increasing by only one segment per round trip time (RTT) (Abed, Ismail, & Jumari, 2012). Section 2.1.1.3 gives a description of this process.

The majority of CCA have been developed with several goals, including minimizing delays and avoiding underutilization, maximizing throughput and ensuring a fair share of bandwidth among all flows. CC will reduce the amount of bandwidth available for all current flows and increase it for new ones until the desired level of fairness is achieved (Al-Saadi, 2019). While there are a variety of TCP CCA available, some of them are loss-based algorithms, others are delay-based algorithms, and still others are hyper-algorithms. The loss-based algorithms will increase the CC until a loss has occurred. At this point, the congestion avoidance state will be activated. However, delay-based algorithms continue to increase the congestion window until a loss packet is about to occur. At this point, the congestion avoidance state begins, causing a delay in processing the queued packets. Hyper algorithms operate, in some conditions, as loss-based algorithms, and in others, as delay-based algorithms. TCP Tahoe and Fast Retransmit are two examples of loss-based CCA (Al-Saadi, 2019).

2.1.2. Additive Increase/Multiplicative Decrease (AIMD)

Per RTT, the TCP sender expands its congestion window by one packet at most. The TCP sender reduces its congestion window by half in response to a congestion alert. Also the TCP sender expand the congestion window in order to take advantage of newly available network capacity. This is referred to as an "additive increase multiplicative decrease", (Khalil, 2012).

2.1.3. Slow Start Algorithm

The slow-start phase is the first stage of TCP software implementation. The TCP sender uses the slow-start method to change the data flow rate to the receiver. The new slow-start period begins with every acknowledgement (ACK) received from the TCP receiver. This means that the TCP sender's transmission rate is entirely dependent on the acknowledgements (ACK) received by the TCP receiver.

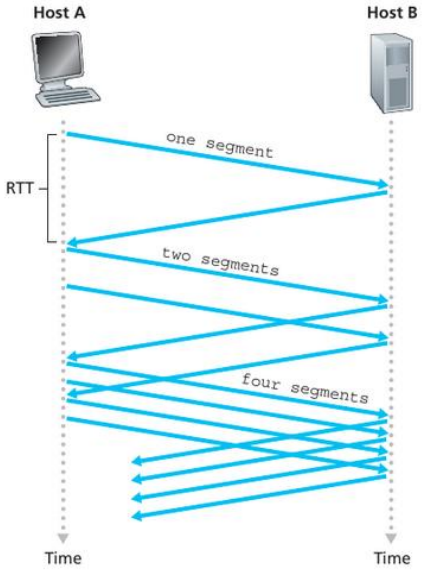

Figure 2.1: Slow Start Phase (Source: Abed, Ismail, & Jumari, 2012).

Referring to Figure 2.1, the slow start begins with a single segment for cwnd (cwnd=1 segment). The cwnd increases by one with each ACK received (new cwnd = previous cwnd + 1). Until cwnd hits the link's congestion point, the exponential growth of cwnd with each RTT, will duplicate cwnd size (new cwnd = previous cwnd × 2), (Ivan Petrov, 2013).

2.1.4. Congestion Avoidance

When the threshold value is reached (ssthresh), TCP reduces the rate at which window sizes grow (from exponential in the slow-start state to linear in the congestion avoidance state). TCP transmission rate exceeds network link capacity after a period of time, resulting in packet loss.

TCP detects packet loss instantly and reduces the congestion window size to about half of its original size. (Chia-Tai Chan, 2004).

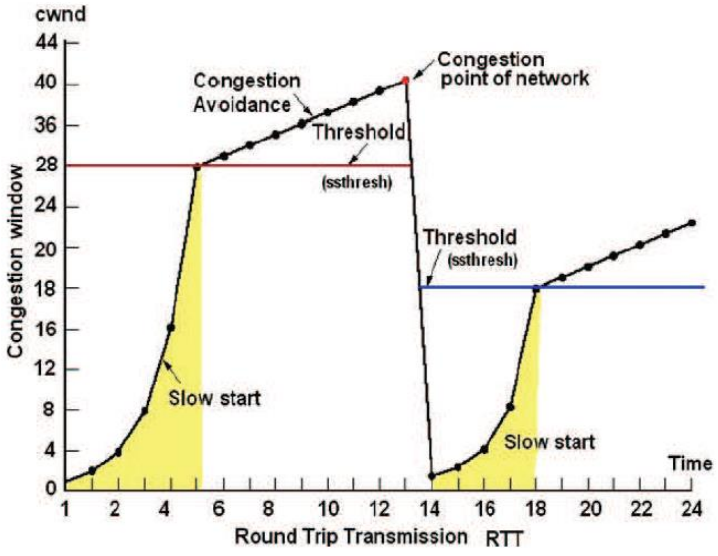

Figure 2.2: Congestion avoidance (Source: Source: Abed, Ismail & Jumari, 2012).

The combined operations and function link between the slow-start and congestion avoidance phases are explained in Figure 2.2. When the congestion window size exceeds the recent slow-start threshold, the

congestion avoidance mechanism is activated, and then it is reduced to half of its earlier dimension when a TCP sender senses congestion (packet loss). The slow-start in some TCP implementations, such as Reno, begins at half its prior size. Other TCP variations, such as Tahoe, start the slow-start from cwnd = 1 (rather than cwnd = half of its previous size).

The slow-start phase should theoretically result in smooth exponential growth, but in practice, the rise is less rapid (Abed, Ismail, & Jumari, 2012). To keep the transmission flow rate from oscillating, the congestion avoidance mechanism's stability is applied. The transfer amount is increased reliably in a linear way throughout the duration of congestion avoidance, (Eddy & Yogesh, 2005). TCP initiates the congestion avoidance mechanism if cwnd > ssthresh, However, if cwnd = ssthresh, either slow-start or congestion can be used (Dushyant, 2005). This method will force the sender to gradually increase its flow rate as it approaches the section where earlier congestion had been detected, (Qian and Dongfeng, 2010). The source periodically transmits two segments for each ACK received during the slow-start phase of the window expansion method. The source node normally sends one segment for each ACK received during the congestion avoidance phase (Ha, Rhee & Xu, 2008).

2.1.5. Fast Recovery (TCP Reno)

Slow-start of TCP CC, significantly decreases throughput after segment damage is detected. Fast retransmission and recovery provide a mechanism to speed up connection restoration. This mechanism detects segment losses through duplicate acknowledgements. This mechanism is a TCP's recovery method for avoiding the retransmission time-out of the waiting time for each loss segment (Jiang, Zhang & Guan, 2014). The fast recovery mechanism adjusts the sending of new segments until the sender receives a non-duplicate acknowledgement. Assume you receive a packet (sequence number 1), the receiver sends an acknowledgement with a value of one added to the sequence number (sequence number = 1+ 1 = 2). This

means that the receiver has received packet number one and is anticipating packet number two from the sender.

If three subsequent packets are lost, the receiver receives packet number five. The receiver sends an acknowledgement with the sequence numbers 2 and 6 after receiving packet number 5. When the receiver receives packet number 6, it sends an acknowledgement containing the numbers 2 and 7. As a result, the sender receives multiple acknowledgements with the same sequence number 2. This is known as duplicate acknowledgement. The sender can then be reasonably certain that the segment with the following higher sequence number was dropped, and will not be delivered out of order. This process conserves acknowledgement clocking, and when a non-duplicate ACK arrives, the fast recovery is completed. And cwnd has been reduced (Henderson, Floyd, Gurtov, Nishida & University of Oulu, 2012).

With reference to Figure. 2.3, when a segment is lost, The TCP receiver will send ACK segments containing the next predictable order number. The sequence number will be associated with the lost segment. When an individual segment is lost, TCP will continue to generate ACKs for the subsequent segments. The sender will then receive duplicate ACKs as a result of this. The duplicate ACK represents a lost packet (Chauvenne & Libiolle, 2016).

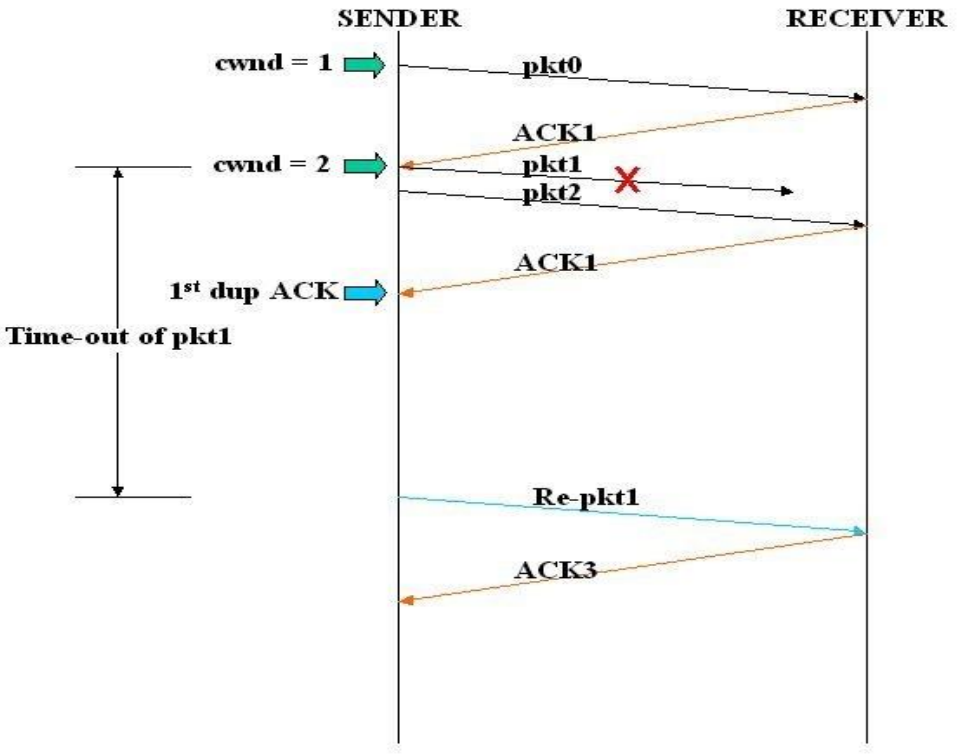

Figure 2.3: Fast retransmit mechanism of Tahoe (Source: Abed, Ismail, & Jumari, 2012)

2.1.6. The Fast Retransmit

During the fast retransmit stage, when TCP receives duplicate ACKs, it decides to resend the segment, where there is no waiting time required for the segment timer to expire. This method will help accelerate the recovery of segment losses. When a segment is lost in fast recovery, the TCP attempts to maintain the current flow rate rather than returning to slow-start. TCP Tahoe was the first protocol to use the fast retransmit mechanism (Tomar & Panse, 2012). based on the idea of resending the unacknowledged segment after receiving three duplicate ACKs. When this occurs, the cwnd size is reset to one packet, and the slow-start process is initiated. Figure 2.4 depicts the TCP Tahoe fast retransmit mechanism. The fast recovery mechanism first appeared in Reno (Abrahamsson, Hagsand & Marsh, 2002). When fast

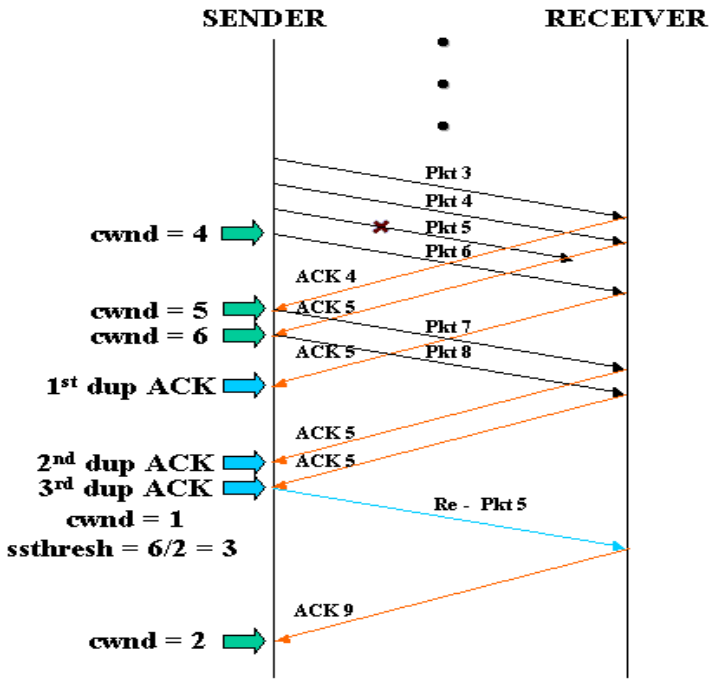

Figure 2.4: Fast retransmit mechanism of Tahoe (Source: Abed, Ismail, & Jumari, 2012)

retransmit is used, fast recovery takes the place of slow-start. While duplicate ACKs indicate the loss of a segment, it indicates that packets are still being transmitted because the source received a packet with a sequence number greater than the missing packet. The assumption in this scenario is that a single packet has been dropped and the network is not fully congested. As a result, the TCP sender does not need to fully return to slow-start mode, but only to half of the previous rate The first two stages are used by the TCP sender to control the number of packets inserted into the connection. Whereas fast retransmission and fast recovery are used to recover from packet losses without the need to resend retransmission timeout packets RTO Reno is an improved version of Tahoe in which the fast retransmit process has been enhanced the process is being improved further to include fast recovery (Source: Abed, Ismail, & Jumari, 2012).

Reno frequently promotes fast retransmit recovery, which is followed by congestion avoidance rather than slow-start (Abrahamsson, Hagsand & Marsh, 2002). After a quick retransmit, the mechanism causes the connection to remain unfilled for a short instant. As a result, there is no need to slow-start to fill-up

for every single packet loss. Fast retransmit reduces the cwnd to half while continuing to send segments at this reduced level. When a primary threshold of duplicate ACK is received, the fast recovery function is inserted into the function of TCP sender (This equates to four ACK with the same sequence number). When the threshold value is reached, TCP sender reduces cwnd by half and sends a single packet again. Unlike the Tahoe sender, when a single packet is lost within a window, TCP Reno performs better than Tahoe. The Reno falls when multiple packets are lost in the same window (Mishra, PratapVerma, KumarSrivastava, & Gupta, 2018). If any losses occur, the sequences' holes will be rearranged and the receiver will be prepared to accept new segments that are compatible with its window, over time, the receiver will send ACKs indicating the sequence hole for each segment. As a result, duplicated ACKs will be generated. If three or more duplicate Acknowledgements for the same segment are sent from receiver to sender, the sender immediately recognizes that this segment has been missed and will resend it before the RTO expires (HO Cheng-Yuan, Yaw-Chung, Chan & Cheng-Yun, 2008).

2.1.7. Other implementation of fast recovery algorithm (TCP NewReno)

TCP NewReno has another implementation of the fast recovery algorithm. In NewReno, the fast recovery algorithm is created by interpreting a fractional acknowledgement as an indicator of an additional lost packet at this sequence number (Reiher & Kleinrock's, 2010). Where fractional ACK is acknowledged after the earlier point and while still in the recovery process window. When many packets are lost for data from a single window, this will improve connection throughput. TCP Reno is waiting RTO for each packet in this situation. Followed by a slow start, so although TCP NewReno addresses this issue through improved recovery (HO Cheng-Yuan, Yaw-Chung, Chan & Cheng-Yun, 2008). When many packet losses occur in the window, Reno's performance suffers. If the segments arriving at the destination are not in the correct order, the host cannot distribute these segments to the end request because they must be buffered

until the correct sequence is obtained. To notify the sender of the lost packet, the receiver must send an instant duplicate ACK (based on the out-of-sequence segments). As well as the sequence's expected number after receiving three duplicate ACKs, reduce cwnd to a single segment. Regardless of the congestion situation, the network can still send segments (Hassan, 2016). The goal of TCP differences is that each type has some distinguishing features, such as the base TCP is now known as TCP Tahoe. TCP Reno enhances TCP Tahoe with a new mechanism called fast recovery. TCP Reno's newest retransmission mechanism is used by Newreno. The use of Sacks allows the receiver to specify several additional data packets received out of order within a single duplicate acknowledgement. TCP Vegas has developed its own retransmission and CC strategies. Figure 2.5 depicts the functional interference between the slow-start, congestion avoidance, and fast recovery phases with all expected probabilities.

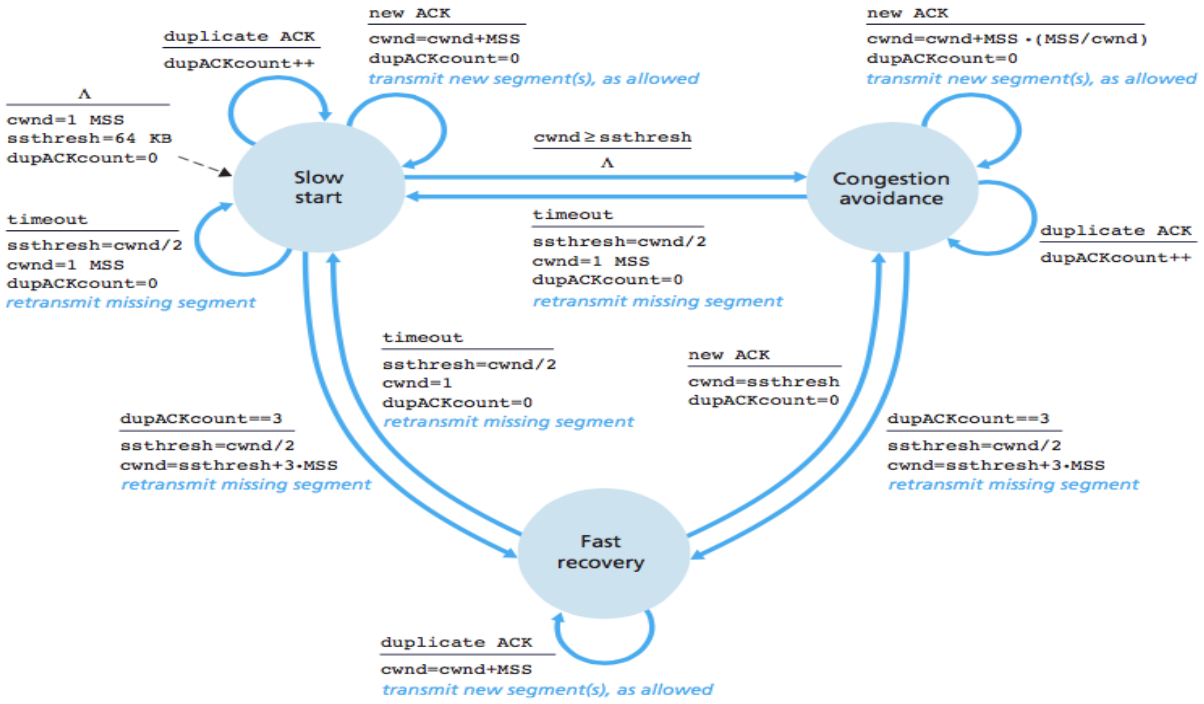

Figure 2.5: Functional interference of recovery phases (Source: Abed, Ismail, & Jumari, 2012).

2.1.8. Delay based algorithms

The TCP Vegas algorithm is an example of a delay-based algorithm. TCP Vegas is an algorithm that relies on the packet delay without packet loss to control the congestion at the bottleneck. The TCP Vegas inspects the congestion in the first stage as a result of an increase in the round-trip time value, as opposed to the loss-based algorithms, which inspect the congestion after the loss has occurred. This type of algorithm is primarily dependent on the calculation of RTT; if the value is very small, the throughput is less than the bandwidth. If the value is very high, the bandwidth will be full soon, and the loss will occur if no action is taken. This is the main principle of which similar algorithms use it as a control point when it enters into the congestion avoidance state (Pérez, Alonso, Veiga & García, 2014).

2.1.9. Loss based algorithms

Loss-based CC algorithms are simple to implement. When competing with latency-sensitive flows, flows using loss-based CC become problematic. When competing with latency-sensitive flows, TCP based on loss pushes the bottleneck queue to a high threshold until a buffer overflow occurs, resulting in a longer queuing delay. The additional latency that results can have a negative impact on latency sensitive applications (for example, live video streaming and online gaming) and generally degrades network QoS. Using packet loss as a signal wastes network resource as well (Leith & Andrew, 2009).

2.2. MPTCP CCA

Using multi-path, a single stream of data can be divided into multiple paths. This method has significant advantages in terms of data reliability, and it can improve the way network resources are utilized. Additionally, the connection will not be lost even if one of the paths fails.

MP links may also be advantageous for load balancing purposes in servers that contain multiple network cards. The main question that arises in MPTCP CCA is how to share the available bandwidth between flows in a fairway, which is addressed in the following section (Pokhrel, Panda & Vu, 2019).

Each connection consists of multiple flows, each of which may take a different route to reach the destination, as well as each of which has its own cwind. To see how to share available bandwidth between flows in a fair way, we must consider the following scenario: if a sender sends its packets through two available paths, and this sender achieves twice the throughput that it achieved in the regular TCP, we can conclude that the sender got twice the throughput in the regular TCP (Pokhrel, Panda & Vu, 2019).

When the regular CCA is used, the share of available paths is not fair, especially when there is other competing traffic on the same paths, because the regular CCA has a negative effect on other flows in the same path. This will cause the other flow to stop sending any traffic through these paths. Moreover, one of the solutions offered by MPTCP is a weighted version, in which every flow is given a dedicated weight based on the amount of bandwidth available on the regular TCP connection. Another challenge is to make very efficient use of available bandwidth. To address this issue, one of the MPTCP principles is to shift the data into a less congested path. There is also a problem with RTT mismatch, which affects most MPTCP algorithms. We can see the RTT mismatch between 3G and wireless networks because 3G has a large buffer, which results in a long delay and few dropped packets, whereas wireless has a small buffer, which results in a short delay but many dropped packets (Shiva Raj, 2018). So the proposed solution to avoid this shortage was to use a semi-coupled MPTCP CCA. It was implemented (Zhou, Dreibholz, Zhou, Tan, & Gan, 2017). To achieve a number of objectives, including selecting the best path, ensuring fairness with regular TCP, and balancing congestion.

The following three rules are used by most CCA:

1) Increased throughput, and it could achieve throughput levels comparable to the best available regular TCP.

2) The MP flow should not allocate the shared link resource in the same way that a regular TCP flow would.

3) The algorithm for multipath CC should shift traffic from the most congested path to the least congested path while adhering to the prior two rules (Zhou, Dreibholz, Zhou, Tan, & Gan, 2017).

2.2.1. EWTCP TCP CC

EWTCP is a multipath technique that was proposed to improve the current MPTCP algorithm, every sub-flow r ∈ R has its own congestion window w_r, so that the congestion window increases by $\frac{a}{wr}$, every acknowledgment on the sub-flow, and decreases by w_r / 2 for each loss on the sub-flow, where $\alpha = \frac{1}{\sqrt{n}}$, and n is the number of paths. This very basic strategy does not require any explicit shared bottleneck identification. Although EWTCP can compete fairly with regular flows but it cannot use the network efficiently (Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011). Some traffic control algorithms were created to transfer traffic from a high-congestion state to a low-congestion state.

2.2.2. Coupled TCP CC

The basic idea behind coupled algorithms is that a multipath flow should send all of its traffic to the least congested path. The coupled flow increases the congestion window by $\frac{1}{W_{total}}$ with each acknowledgement in each flow, and decreases the congestion window by $W_{\frac{total}{2}}$ with each loss on the flow, where "W_{total}" is the total congestion window size across all sub-flows, (W_{total}= $\frac{2}{P}$, and P = total loss), so it appears that the underutilization issue that caused the congestion window to be computed based on MPTCP EWTCP -flows is solved, by shifting the traffic from most congested paths to the lease congested paths (Guo, Huang & Zhang, 2014).

2.2.3. Semi-Coupled TCP CC

Semi-coupled always tries to keep the same amount of traffic on each link, but with a bias in favor of the less congested link. For example, assume semi coupled MPTCP CCA uses three links, two of them with a 1% drop possibility and the third with a 5% drop possibility. In this example, the two less congested links will each take 45% of the total load, while the most congested link will only take 10% of the total. To work on MP CC methods, the researchers try harder to fulfill the aims of fairness. The major role to play in doing that is to make sure that the MP throughput does not exceed the throughput if we use a single path instead of multiple paths at all times. If there are competing flows in the links, the MPTCP final throughput will be the sum of all the links' bandwidth. But if there are competing flows on the access links, the answer will be different and will depend on our understanding of the main fairness role, which says that the bandwidth on sub-flows must not exceed the bandwidth if we use only one single path.

If we have two network interfaces with the following capacities: 14.4 MB and 2 MB, and there is no competition, the user will only use 14.4 MB, which indicates that one of the access links will be underutilized, which is unacceptable because there is no competition and utilization must be at maximum, this can only be achieved if all of the links are fully utilized (Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

2.2.4. Opportunistic Linked Increases Algorithm (OLIA)

OLIA algorithm is an enhancement from LIA algorithm and that focuses on a tradeoff between optimal congestion balance and responsiveness, OLIA is an alternative that satisfies both of these requirements. The congestion window w_r is raised by two terms when an ACK is received on the sub-flow (Khalil, 2012). While most CCA work by trading between the optimal congestion balance and the

ability to respond, OLIA offers an alternative solution that uses the following methods: The cwind will increase when acknowledgment is received by any of the next two conditions:

a. This first condition is dependent on the optimal congestion balancing strategy, which is based on the equation below:

$$(w_r/\tau_r)/\ [\Sigma_{k\in R}(w_k/\tau_k)]^2,$$

A connection R is comprised of a number of sub-flows R, each of which can connect the Internet through a different path., $k\ is\ the\ subset\ of\ all\ subflows$ in the same connection R, where w_r the congestion window of the sub-flow r and τ is the round-trip time.

b. The second condition depends on the $\alpha_r\ /\ w_r$ guarantee, which ensures that the congestion window is responsive to the cur$r$rent change in the current w_r window.

Unlike other multipath CC methods, the parameter r is unique to each sub-flow and is not shared by all sub-flows at the same time. This parameter is responsible for traffic shifts amongst all sub-flows, and the sub-flows are divided into three types (Bruno Yuji Lino Kimura, 2018):

1. The first type is defined by W – set of sub-flows with the largest of w_r.
2. The second type depends on the B – set of the best of the sub-flows on r which is generated between two losses.
3. The third type depends on the C – set of the sub-flows with no larger window.

When C is not empty, there is at least one best sub-flow. When w_r is small, r will be positive for all $r \in C$,and negative for all$r \in W$, causing w_r to increase quicker for$r \in B$, and slower for $r \in W$, this causes the sub-flow to travel forward from the fully utilized W to the less utilized channel C.

2.2.5. Balanced Linked Adaptation (BALIA)

When the paths have similar RTTs, OLIA can be unresponsive to changes in network conditions (Kim, Song, Mahmud, & Cho, 2021). BALIA is MPTCP CCA approach that allows w_r oscillation to reach an

optimal level to ensure a good balance of friendliness and responsiveness (Mingwei Xu, 2014). For MPTCP connection of a single path (|R|=1), we have α_r= 1. This makes both the increment and decrement of $w_r$to reduce to the same ones of TCP standard (Kim, Song, Mahmud, & Cho, 2021).

2.2.6. Weighted Vegas (wVegas)

While OLIA and BALIA are all packet loss algorithms, wVegas) is a delay-based CCA that estimates the link queueing delay q_r to detect path congestion and implement CC. The wVegas control adjusts α_r according to the weight w_r which is a ratio between the sending rate of the current sub-flow x_r and the aggregate multipath sending rate, whenever α_r is less than δ_r, δ_r is a difference between expected and actual x_r (Guo, Huang & Zhang, 2014).

w_r is adapted in two steps on the transmission round on r. The first behavior is Similar to TCP-Vegas (Guo, Huang & Zhang, 2014). By increasing or decreasing the value of w_r, according to $\delta_r \wedge \alpha_r$. The seconed behavior, by modifying $w_r$according to q_r. This is accomplished by tracking the variation of queue delay $q_r$as measured by the difference between the average observed RTT (τ_r) and the minimal RTT that has been measured (τ_r). The value of w_r is reduced by $\tau_r/2\,\tau_r$ when the variation in the link queue is more than $2q_r$. While minimizing packet loss and avoiding buffer bloat. Thus, wVegas provides a congestion avoidance phase that is more sensitive to network changes than traditional CC that are based on packet loss exclusivity.

2.2.7. Traffics Distributions on Sub-Flows

Assume that the coupled algorithm is used in Figure 2.6 b, then all traffic on the two-hop paths will be moved to the one-hop path due to high congestion on the two-hop path. In Figure 2.6 a, the EWTCP

throughput is estimated by splitting the throughput across all paths. However, the coupled compute the throughput based on the path with the least congestion, so they only use that path.

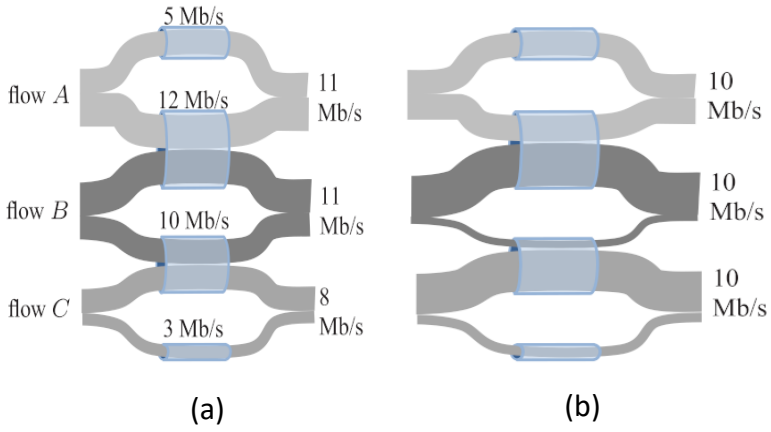

Figure 2.6: A scenario where EWTCP (a) does not equalize congestion or total throughput, whereas COUPLED (b) does (Source: Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

To understand Coupled issue with RTT mismatch, assume you have a wireless device with two interfaces, one connected to 3G and the other to the wireless router. The 3G flow uses a large buffer, which causes delay and small packet loss, whereas the wireless uses a small buffer, which causes small delay and large packet loss (Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

2.2.8. Efficient Path Selection

The following equation can be used to compute throughput in Figure 2.7:

$\sqrt{2 / p} / RTT$, So, the throughput for WIFI is 707 pkt/s, and for 3G is 141 pkt/s, and if we use EWTCP with two paths, the throughput will be calculated as (707 + 141) /2 = 424 pkt/s, and if we use the coupled,

the throughput will be calculated only on the less congested path, which is the path with the least packet loss. So, the coupled traffic will be 141 pkt/s. In addition to the abovementioned issues.

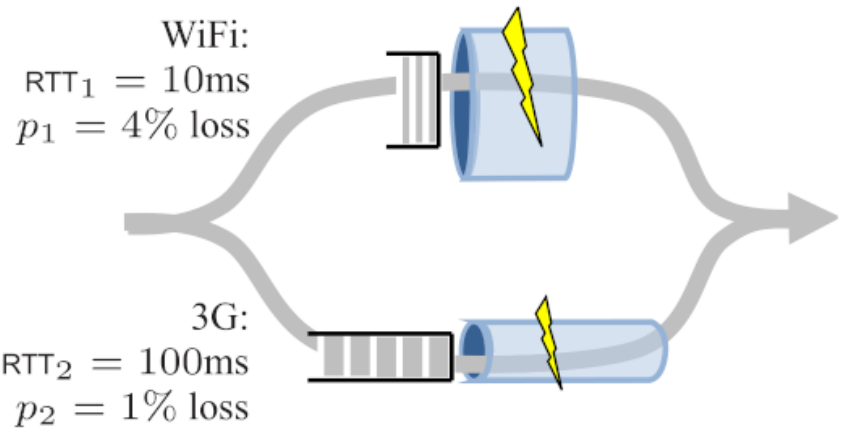

Figure 2.7: A scenario in which RTT and congestion mismatch can lead to low throughput (Source:Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

2.2.9. Static Environment

The coupled algorithm has additional problems, even if all sub-flows have the same RTT (Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011). If the performance of MPTCP in a static environment is compared versus a dynamic environment that the static environment can be shown in Figure 2.8 is the better environment for evaluating these three algorithms. This scenario has five bottlenecks. Each link has multipath flow. All paths have the same RTT, and all buffers have the same bandwidth delay. The capacity on link C has been reduced, causing all traffic on this link to shift to links B and D, causing these two links to become more congested. This situation will force the other flows to shift their traffic toward links A and E, resulting in traffic being evenly distributed across all links if the balancing was excellent.

Moreover, if the balancing does not occur because the MPTCP itself does not support the transition from a high congestion path to a low congestion path, we must investigate how we can improve the situation

by employing the best scheme available in the given situation (Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

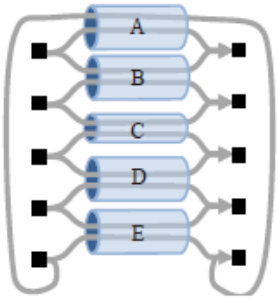

Figure 2.8: Five bottlenecks' topology is used to evaluate how well congestion is balanced. (Source:Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

Additionally, there is a comparison for MPTCP in a dynamic environment. In this case, there is an issue with variant load. Assume that there are two links (upper and bottom links), each with a capacity of 100 MB/s and a buffer of 50 packets/s, and that we are running burst traffic on the bottom link every 10 milliseconds. There are numerous issues that have been discovered in a diverse range of different scenarios. The good result will be affected by the fast congestion level variant, and the scenarios that were tested included a sudden variant to simulate the mobile environment. The results of a real experiment, which simulates the use of a Linux server and the MPTCP algorithm, are included in this research. The first experiment involves the use of two network interfaces, each of which has 100 MB of memory, and these interfaces communicate with a large number of client machines. When the results were obtained, the dummy net tool was utilized in order to simulate the WAN network. Then they connected five machines to the first link, followed by 15 machines that were connected to the second link, and the two groups of machines used long-lived traffic to communicate with one another (New Reno). After that, they must create a diagram that depicts the throughput on each link as well as the balancing that occurs after

congestion occurs on link 2 and flows shift to link 1. They repeated the experiment using the same topology but with increased and decreased traffic loads. On link 1, the heavy load ran every 60 seconds and the light load every 10 seconds. They ran a single long live flow on link 2. Additionally, they performed three multipath flows on the two links, one for each of the multipath algorithms. In this scenario, they needed to determine which algorithm performed the best and which algorithm performed the worst, as well as which algorithm performed the best in terms of fairness with respect to other flows. The result was as shown in Table 2.1 (Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

Algorithm	Top link	Bottom link
EWTCP	85	100
MPTCP	83	99.8
COUPLED	55	99.4

Table 2.1: Comparison results for multipath algorithms in a dynamic environment (Source:Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011)

Because the coupled algorithm works to shift traffic from the most congested path to the least congested path, it performed poorly on the bottom link and poorly on the top link. This is because the coupled algorithm works to shift traffic from the most congested path to the least congested path. While the EWTCP and MPTCP work on load balancing between all available paths, they performed poorly on the bottom link and poorly on the top link. The continuous growth of cloud applications from many IT companies such as Google and Microsoft create many huge data centers, and this means that there is a huge amount of traffic that can be shifted between the machines

2.3. Congestion Control Algorithms

2.3.1. ALGORITHM: REGULAR TCP

- for each loss reduce cwnd by:

$$\frac{1}{w} \qquad (1)$$

- For each ACK segment over sub-flow, increase cwnd by:

$$\frac{w}{2} \qquad (2)$$

Additionally, an exponential rise is applied at the beginning of a connection, as well as immediately after a retransmission timeout.

2.3.2. ALGORITHM: MPTCP CCA

- Each ACK on sub-flow r, for each subset S ⊆ R compute the following:

$$\frac{max_{s\in S} w_s/RTT_S^2}{\left(\varepsilon_{s\in S} w_{\frac{s}{RTT_S}}\right)^2} \qquad (3)$$

Then, for all such S, determine the minimum and increase w_r by that amount.

2.3.3. ALGORITHM: EWTCP

- For each ACK on path r increase window w_r by:

$$\frac{a}{w_r} \qquad (4)$$

- For each loss on path r reduce window w_r by:

$$\frac{w_r}{2} \qquad (5)$$

The size of each sub-flow's window is proportional to a^2, while a = $\frac{1}{\sqrt{n}}$ and n represents the number of paths.

2.3.4. ALGORITHM: COUPLED

- For each ACK on path r increase window w_r by:

$$\frac{1}{w_{total}} \quad (6)$$

- For each loss on path r Reduce window w_r by:

$$w_{\frac{total}{2}} \quad (7)$$

The total window size across all sub flows is given by w_{total}.

2.3.5. ALGORITHM: SEMI COUPLED

- For each ACK on path r increase window w_r by:

$$\frac{a}{w_{total}} \quad (8)$$

- For each loss on path r reduce window w_r by:

$$\frac{w_r}{2} \quad (9)$$

a Is constant which controls the aggressiveness.

2.3.6. ALGORITHM: OLIA

- For each ACK on path r increase window w_r by:

$$w_r = w_r + min\left(\frac{\alpha}{w_{total}}, \frac{1}{w_r}\right) \quad (10)$$

And $\alpha = w_{total} \frac{\max\left(\frac{w_r}{\tau_r^2}\right)}{\left[\sum k \in R\left(\frac{w_k}{\tau_k}\right)\right]^2}$ + (11)

- For each loss on path r reduce window w_r by:

$$w_r = w_r - \frac{w_r}{2} \quad (12)$$

2.3.7. ALGORITHM: BALIA

- For each ACK on path r Increasing window w_r by:

$$w_r = w_r \left[\left(\frac{1+\alpha w}{2}\right)\left(\frac{4+\alpha_r}{5}\right) \right] \quad (13)$$

$$\alpha_r = \frac{\{x\}}{x_r}, \text{ where } x_r = \frac{w_r}{\tau_r} \quad (14)$$

- For each loss on path r reduce window by:

$$w_r = w_r - \left[\frac{w_r}{2}\left(a_r, \frac{3}{2}\right)\right] \quad (15)$$

2.3.8. ALGORITHM: WVEGAS

- $For\ each\ ACK\ on\ path\ r$ Increasing window by w_r :

$$w_r - 1, \text{ If } \delta_r > \alpha_r \quad (16)$$

$$w_r + 1, \text{ If } \delta_r < \alpha_r \quad (17)$$

- For each loss on path r Reduce window w_r by :

$$w_r = w_r - \frac{w_r}{2} \qquad (18)$$

2.4. Congestion Control Performance Measurement Metrics

2.4.1 Flow Control

On the receiving end, if the data is received in the correct order, and without error, the receiver will send an acknowledgement to the sender, informing the sender that all bytes in the segment, with sequence numbers smaller than the segment sequence number, have been delivered correctly and without errors. By sending such acknowledgements, there is no need to send acknowledgement to each segment because the sequence acknowledge includes all segments with sequence numbers smaller than the sequence number of the segment. Thus, the process of sending a sliding window from the sender, and receiving acknowledgement for all segments sent within the window, is referred to as flow control. The buffer size on the sender side is determined by the sender's resources such as memory, availability, and application configurations, as well as the receiver advertised “rwind”. This mechanism prevents the fast sender from continuing to send without receiving any acknowledgment (Al-Saadi, 2019).

2.4.2 Round Trip Time (RTT)

RTT is defined as the length time required to send a data packet to a destination plus the time it takes until it is acknowledged, where the round-trip time is measured in milliseconds (ms). This distant trip can be determined using the ping tool, which is included with most operating systems, Figure 2.9 depicts round trip time.

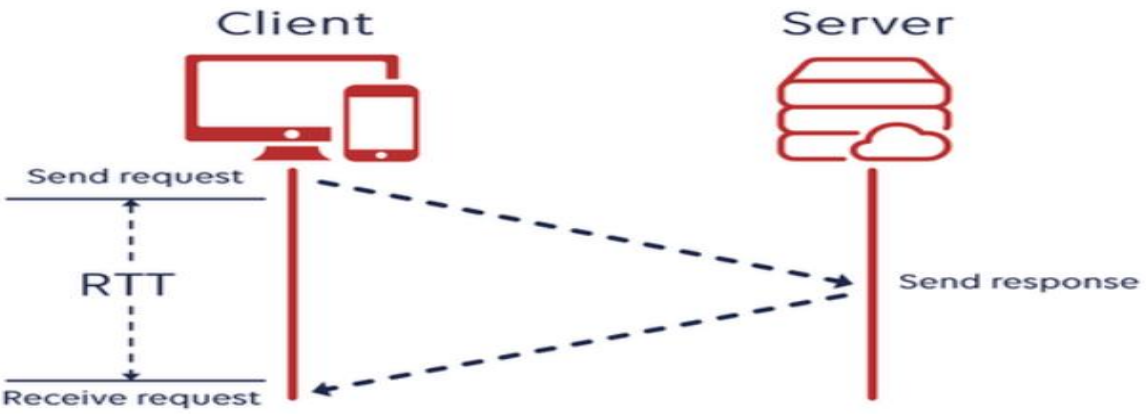

Figure 2.9: Round Trip Time (Source: Rasool Al-Saadi, 2019).

As an example, if a user is located in Palestine and sends a packet to Jordan, the network traffic is routed through numerous routers in numerous locations. When the destination in Jordan receives the packet and sends an acknowledgement to the sender, the acknowledgement is routed through numerous routers in various locations until it reaches the source, implying that the RTT is equal to the total destination time from sending the packet to receiving the acknowledgment at the source. RTT is one of the most important metrics because it tells us how well the network connections are performing. Additionally, ping is one of the most commonly used tools by network administrators, as decreasing the RTT, the overall performance of the network and the speed of the connections are improved (Al-Saadi, 2019). If the distance between the source and destination is large, this increases the RTT time, but this factor can also be eliminated by upgrading the medium's material. By upgrading the material of the medium to optical one and transmitting data at light speed, the distance factor will be reduced to its minimum and possibly eliminated, (Christiansen, 2021). If the data is transferred within the LAN, it will be faster and the RTT will be much smaller than when the data is transferred over a WAN connection. On the other hand, the request is transferred from the LAN to the WAN, the WAN has a negative impact, because the LAN can become a bottleneck in the network due to the large number of users within the LAN consuming the available bandwidth in this situation. The type of connection medium has a significant

impact on the speed of a request. For example, an optical connection medium can transmit data at the speed of light, whereas this cannot be accomplished by a traditional medium, (Malcolm Johnson, 2009). RTT mismatch is a problem that affects most MPTCP CCA. For example, when a laptop is connected to a 3G USB and WIFI, the flow is divided into two sub-flows, one sub-flow for each adapter.

Because of the low loss of the 3G adapter, most multipath TCP congestion algorithms, especially coupled algorithms, will prefer the 3G adapter and will therefore lose the high bit rate transition provided by wireless and will instead use the lower bit rate transition provided by 3G. Furthermore, these algorithms will not shift the traffic into less congested paths because they prefer the smallest RTT path; only possible if all paths have the same RTT. And these types of paths are frequently used in data centers, (Source:Wischik, Raiciu, Greenhalgh, Handley & University College London, 2011).

2.4.3 Response Time and Delay

Response time is the amount of time it takes for the destination to process a request and respond to the source; for example, if the server's resources are overloaded, it will take longer for the server to respond to the request (Sematext, 2022). Propagation delay is referred to as the time it takes for a packet into flight to arrive at its destination, and the maximum speed at which a packet can arrive at its destination is referred to as the speed of light. If the source and destination are in the same building at the distance of 200 m, the propagation delay will be ~ 1 μsec. If they are located in different countries at a distance of 20,000 km, however, the delay is in order of 0.1 sec; if they are in separate LANs and the distance between them is 100,000 kilometers, the propagation delay is 2 μsec. Certain interactive applications require a propagation delay of less than 200 μsec. If the propagation delay on the server hosting these applications exceeds this value, no network enhancement will resolve the issue; the only option for the user is to exit these applications and migrate to a higher performing application (Giuliano, Schonfeld & Khokhar, 2019).

Transmission delay is the time required by the sender to transmit a packet through the transition medium is referred to as the transmission delay (Giuliano, Schonfeld & Khokhar, 2019). Additional transmitting delays will occur if the server has a large number of active sessions and insufficient resources to schedule additional requests, or if the server operating system does not support real-time scheduling algorithms. If the packet travels through a network medium that is shared by a large number of network flows and these flows consume the network bandwidth, or if the network bandwidth is limited, such as on a traditional link that supports up to 10 MB/s, upgrading these links to 100 MB/s or 1 GB/s can reduce the transition delay by several orders of magnitude (Giuliano, Schonfeld & Khokhar, 2019). If packets pass through a shared medium that is used by multiple devices concurrently, the appropriate network shared algorithms that can be selected in this type of network can affect and reduce transmission (Giuliano, Schonfeld & Khokhar, 2019).

2.4.4 Congestion And Fairness

When the offered load in sharing resources exceeds the total network capacity, the effective load will go to zero as load increases (Reiher & Kleinrock's, 2010), and this might happen when the router's buffer becomes congested and send more packets. The destination indicates the size of the receiver window through the TCP header. This advertises the amount of data that it can receive and enables the congestion window to determine the amount of data it can reliably transmit without an ACK. Once received, the data is stored in the receiver buffer and the receiver sends an ACK or a set of ACKs depending on the amount of received data. The congestion window size keeps increasing up to the maximum receiver window, or until the network reaches its limit. However, the rate of sending data is bound to the congestion window, and even when the receiver window is large, the congestion window may be smaller – especially if the network does not support transmitting data equivalent to the maximum Receiver Window (RWND) size.

Congestion is detected either by receipt of a duplicate acknowledgement or timeout signal. Once this happens, the TCP sender decreases the sending rate by decreasing the congestion window size by a factor determined by the algorithm used. The maximum amount of unacknowledged data that the source can send is the lower of the two windows (stackpath, 2020). Fairness is the degree of equality in resource allocation that does not negatively impact the network's throughput is referred to as fairness (Go Hasegawa, 2010). The amount of time it takes for the network flow to reach a fair state is measured in seconds (Al-Saadi, 2019). Latency is referred to a measure of delay that refers to the amount of time that can measure the time between the time a packet is sent from its source and the time it is received at its destination (cloudflare, 2020).

2.4.5 Throughput and Bandwidth

Throughput is defined as the amount of data that can be transferred from the source to the destination within a specified time frame; it also refers to the number of packets that successfully arrive at the destination (Verma & Zhang, 2020). Throughput is measured in bits per second, and it is one of the performance keys in all networks; increasing this key will increase the performance of the network. Additionally, we can improve this key by lowering the latency, because latency has the effect of decreasing throughput by slowing down the network, which can result in a very bad or poor network, (Fan Zhang, 2020). While both throughput and bandwidth refer to the amount of data that can be received at a given time, the difference between them is that bandwidth refers to the capacity rather than the speed (Kassim, 2011).

Chapter 3: Methodology

3.1. Experimental Model Setup

For this experiment, we used a laptop running the Linux operating system Ubuntu 16.04 LTS and (MPTCP v0.93.4) deployed on a special kernel version (4.19.126.mptcp) on a real machine, which is a special kernel distribution designed specifically to handle MPTCP network management. In addition, we used a 3G USB interface (Huawei Mobile Wi-Fi E5576-855), a WIFI base station (Tplink TL-WA901ND 450 Mbps Wireless N Access Point), and two mobile phones (iOS and Android). The topology of the experiment is shown in Figure 3.1.

We used “ifstat” (Ronald J, 2021) for measuring throughput per flow. This experiment investigates the throughput of MPTCP CCA in the following four scenarios:

1. Static without competing,
2. Static with competing,
3. Moving without competing,
4. Moving with competing.

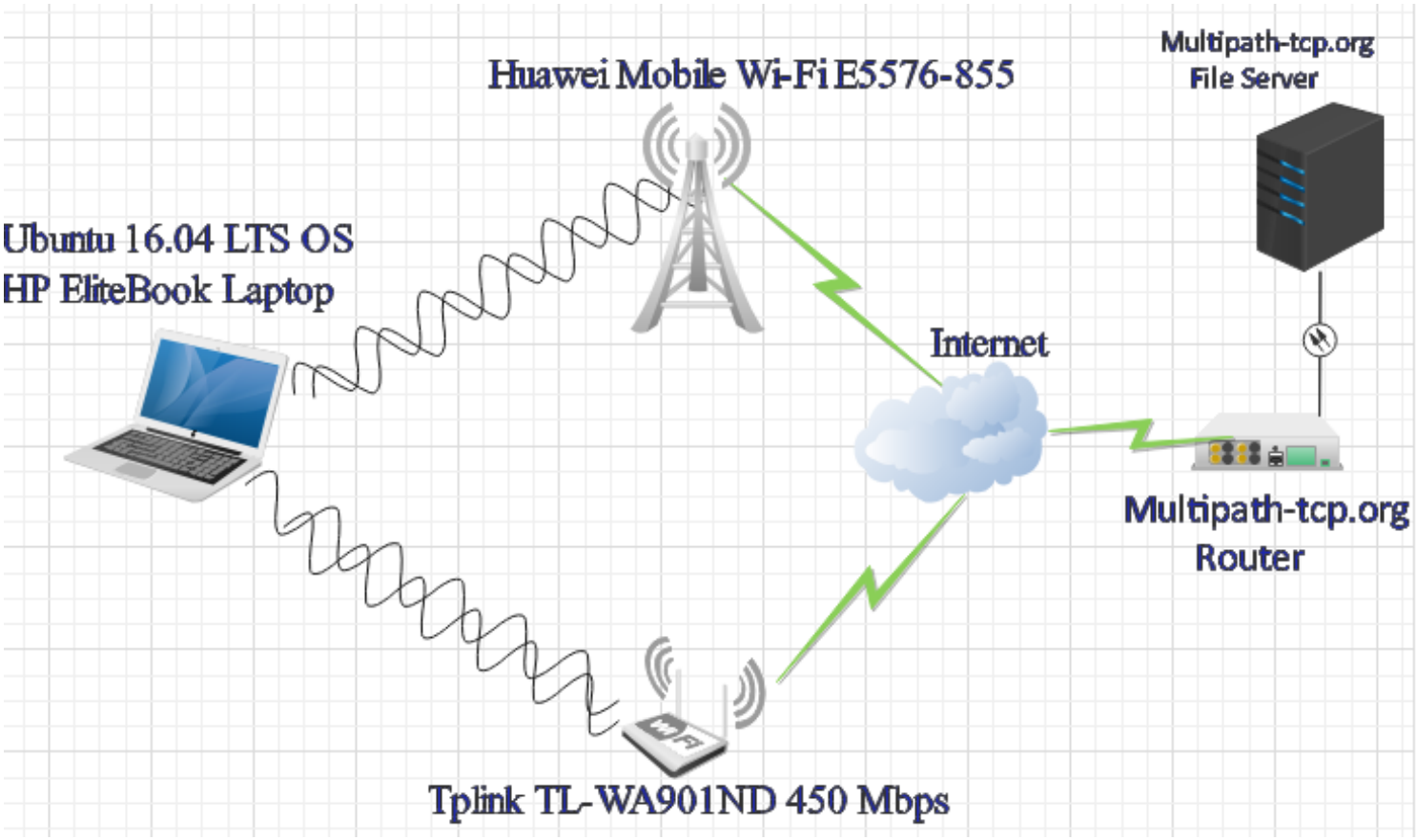

Figure 3.1 Experiment Topology

3.2. Scenarios

Scenario #1

Figure 3.1shows a simple scenario in which the transmitter and receiver are separated by two paths 3G and ADSL. The researchers in this scenario wants to investigate how the MPTCP CCA and regular TCP work while the client is in a static state without competing with other traffics. And without having to compete with other traffic. The experiment begins with one TCP running over 3G and the other on ADSL. Then the research stop the regular tcp and run MPTCP, then download files from the following site: http://multipath-tcp.org/snapshots , each downloaded file size is about 121 MB. In this state we run the

laptop to download in MPTCP, The 3G and WIFI signals were excellent, and the laptop was static, for each scenario was run 15 tests for 15 times, 5 using WIFI interface, 5 using 3G Interface and 5 using MPTCP Interface.

Scenario #2

In this scenario, we use the same model as in the first scenario, the difference is that the researcher added new traffic to compete with the existing regular TCP and MPTCP traffics, the competing traffic is video streaming from YouTube. The researcher in this scenario wants to investigate how the MPTCP CCA and regular TCP work when the laptop is on the move state with competing with other traffics.

Scenario #3

(Figure 3.2) shows a simple scenario in which the transmitter and receiver are separated by two paths 3G and ADSL. The researcher in this scenario wants to investigate how the MPTCP CCA and regular TCP work while the client is in a movement state. And without having to compete with other traffics. The experiment begins with one TCP running over 3G and the other on ADSL. Then the research stops the regular tcp and ran MPTCP, then downloading files from the following site: http://multipath-tcp.org/snapshots , each download file size is about 121 MB.

Scenario #4

In this scenario, we use the same model as in the third scenario, the difference is that the researcher added two new traffics to compete with the existing regular TCP and MPTCP traffics, the competing traffic is video streaming from YouTube. The researcher in this scenario wants to investigate how the MPTCP CCA and regular TCP work while the client in movement state with competing with other traffics.

Chapter 4 Simulation Results and Discussion

4.1. Introduction

In this chapter we discuss the results of our experiments and the relationship between these results and the main theory, which was discussed in the previous chapter, in Section 3.

All experiments have been conducted on the regulate TCP algorithms, including MPTCP CUBIC, OLIA, BLIA, and Wvegas. Some of these algorithms are categorized as loss-based algorithms, while others are categorized as delay-based algorithms. Because the loss-based algorithms cannot predict the loss, they do not enter the congestion avoidance state, and they achieve high throughput. Noticing that it increases the congestion window until the loss occurs, at every point it enters the slow start phase directly, allowing these algorithms to consume the maximum amount of the available bandwidth possible. Meanwhile, the congestion avoidance algorithm can predict the loss and enter the congestion avoidance phase before the loss occurs, allowing them to always work underutilization. The majority of network resources are not utilized, allowing the loss-based algorithm to consume the maximum amount of bandwidth possible. Furthermore, because modern operating systems now employ extremely fast CCA, such as cubic, it is possible that wVegas and other CCA will be overcome when compared to them, and this has been proven in the results of experiments. The second part of the experiment will demonstrate the standard TCP protocol on a single path, and compare it to multipath algorithms. In addition, we can see that only the MPTCP cubic algorithm outperforms the single path TCP algorithm (SPTCP). And this can occur only in the non-competing case, because in the competing case, the throughput equals the throughput on the best path, whereas in the non-competing case, the total throughput equals the throughput on all sub-flows, effectively doubling the throughput. Besides that, there are two types of routes: RTT routes and miss-RTT routes. The difference between the two types of routes is that if all sub-flows have a common

RTT, then the coupled algorithms will divide the traffic into all routes with a prevalent weight. If all routes do not have a common RTT, then the coupled algorithms will choose the least RTT routes.

4.2. Performance Analysis of Scenario 1

From Figure 4.1b, it is clearly observed that the MPTCP CUBIC outperforms all the considered MPTCP CCA. This is due to the fact that when there are competing sub-flows, the throughput is equal to the throughput for the better sub-flow, however, when there is no competition, the throughput equals the sum of all sub-flows, and the results show that the MPTCP Cubic throughput improves significantly as the number of sub-flows increases along the same path, so this algorithm perform better than regular TCP and this can be noticed by comparing results in Figure4.1a with results in Figure 4.1b. On the other hand, the rest of algorithms performs worse than regular TCP, except MPTCP OLIA Which performs better than regular TCP without clear winner as illustrated by the following graphs: Figure 4.1a to Figure4.1e. In addition, we can notice the throughput of wVegas is the lowest of the four CCA, this algorithm is a member of the delay-based family of algorithms, which can detect loss before it occurs, and enters congestion window state avoidance. This algorithm is extremely sensitive to this detection, as we can see that many decrease occurred, which can result in the network performance being very pure and unable to reach high utilization by wasting network utilization, in contrast to OLIA and BALIA, which improve network performance at the expense of increased packet losses due to a lack of network state awareness. As is clear from the Figure 4.2 which shows the average Payload Throughput [kbit/s].

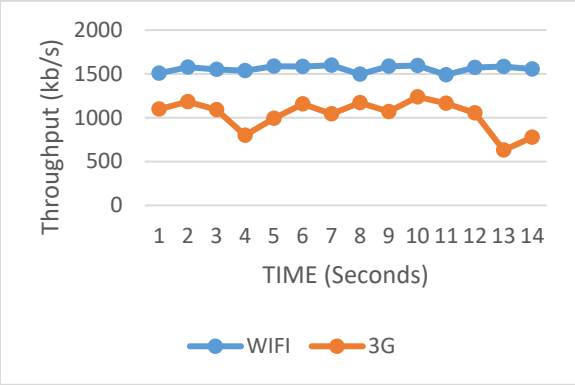

Figure 4.1 (a) Regular sub flows Throughput

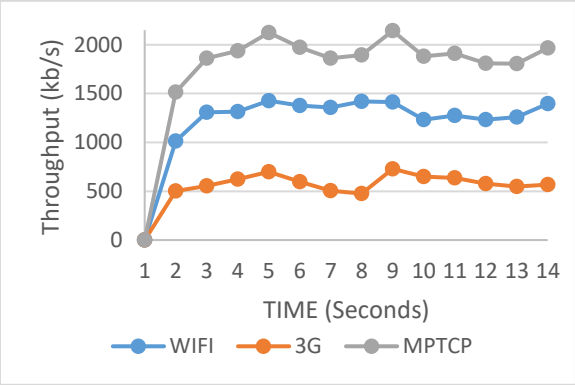

Figure 4.1 (b) Cubic sub flows Throughput

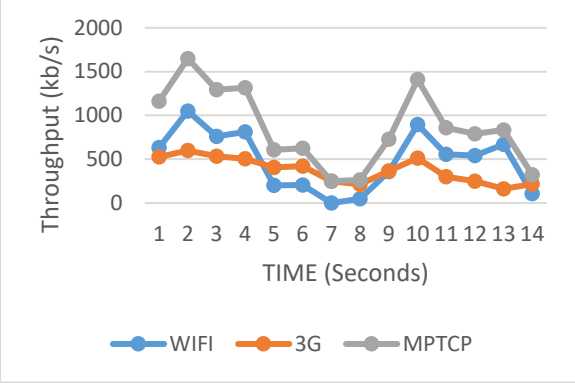

Figure 4.1 (c) Balia sub flows Throughput

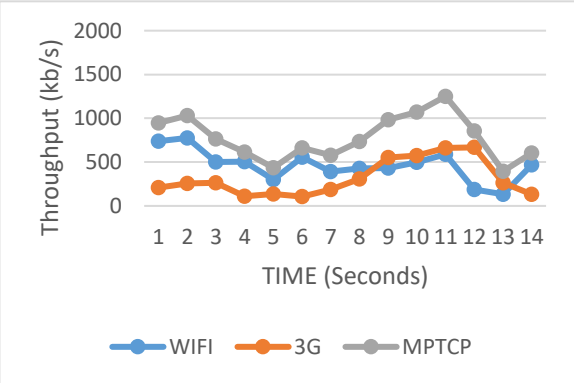

Figure 4.1 (d) Wvegas sub flows Throughput

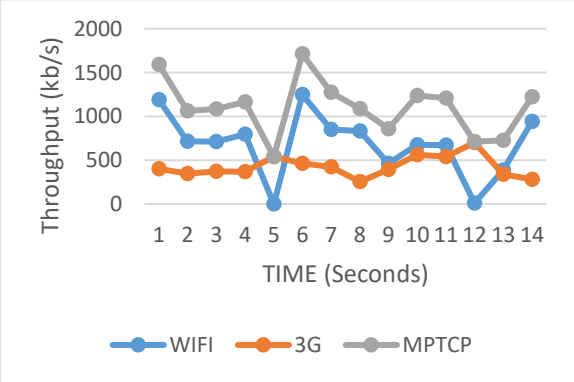

Figure 4.1 (e) Olia sub flows Throughput

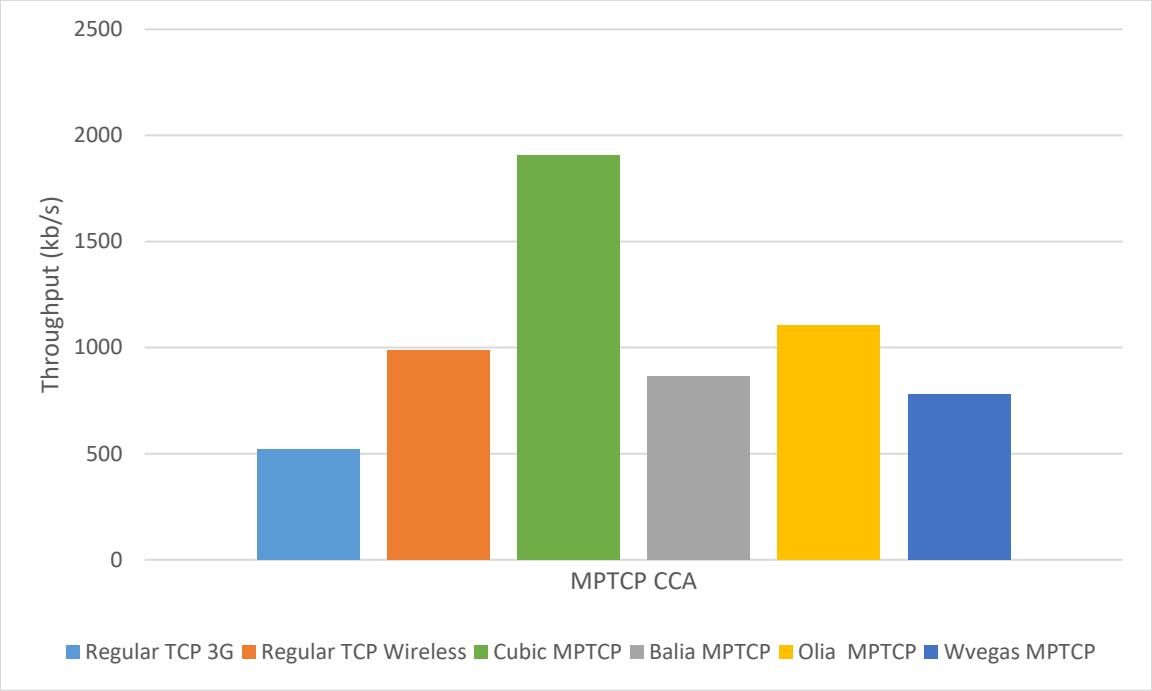

Figure 4.2 Average throughput per CCA for scenario #1

4.3. Performance Analysis of Scenario 2

From Figures 4.3 (a) to (e), show clearly that MPTCP Cubic has a higher throughput than Balia, wVegas , Olia and Regular TCP. Furthermore, Figure 4.4 shows clearly that wVegas outperforms other CCA. And this because the traffic was distributed approximately evenly between the interfaces by this algorithm, this algorithm has benefited from stability of 3G interface. Unlike the OLIA algorithm, which put a higher load on the Wi-Fi interface, which had a lot of Oscillation, sometimes a loss of packets occurs, and this is due to the fact that this algorithm is a loss based algorithm. On the other hand, we find that Balia has outperformed the Olia, and this due to its unique ability to shift traffic from congested interface (WIFI) to less congested interface (3G). This algorithm also had no packet losses, it relates to the wVegas CCA nature as a delay-based algorithm.

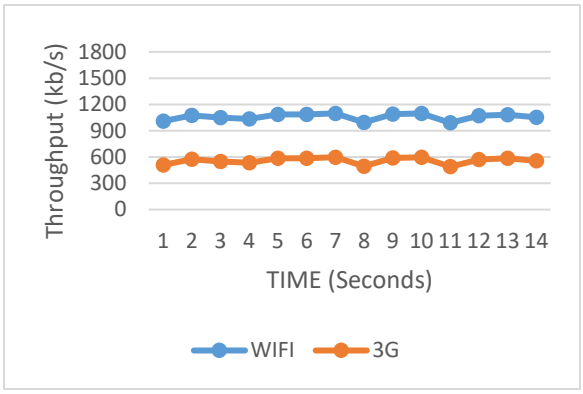

Figure 4.3 (a) Regular sub flows Throughput

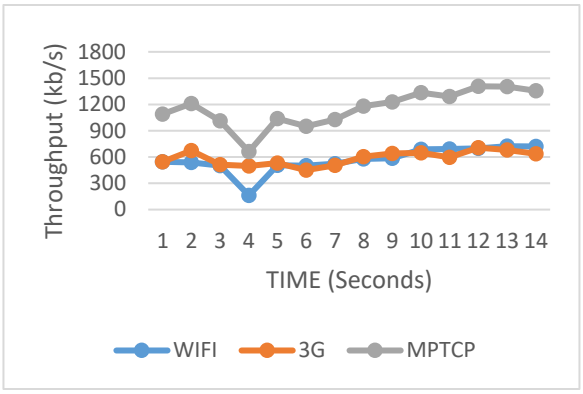

Figure 4.3 (b) Cubic sub flows Throughput

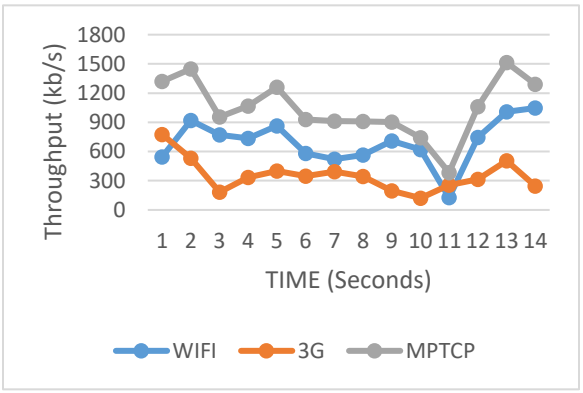

Figure 4.3 (c) Balia sub flows Throughput

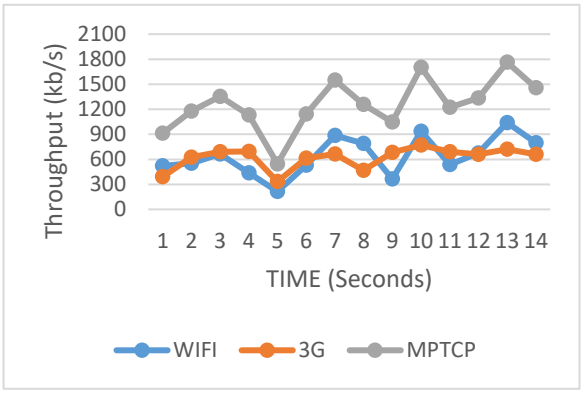

Figure 4.3 (d) Wvegas sub flows Throughput

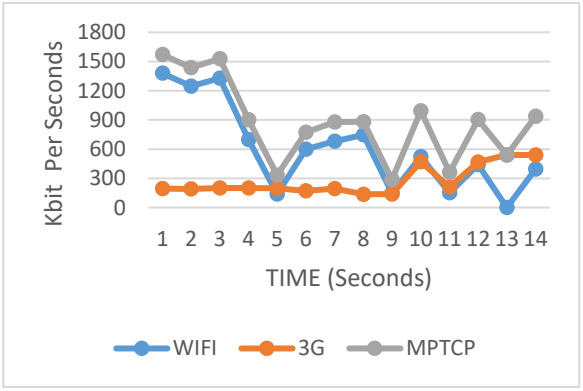

Figure 4.3 (e) Olia sub flows Throughput

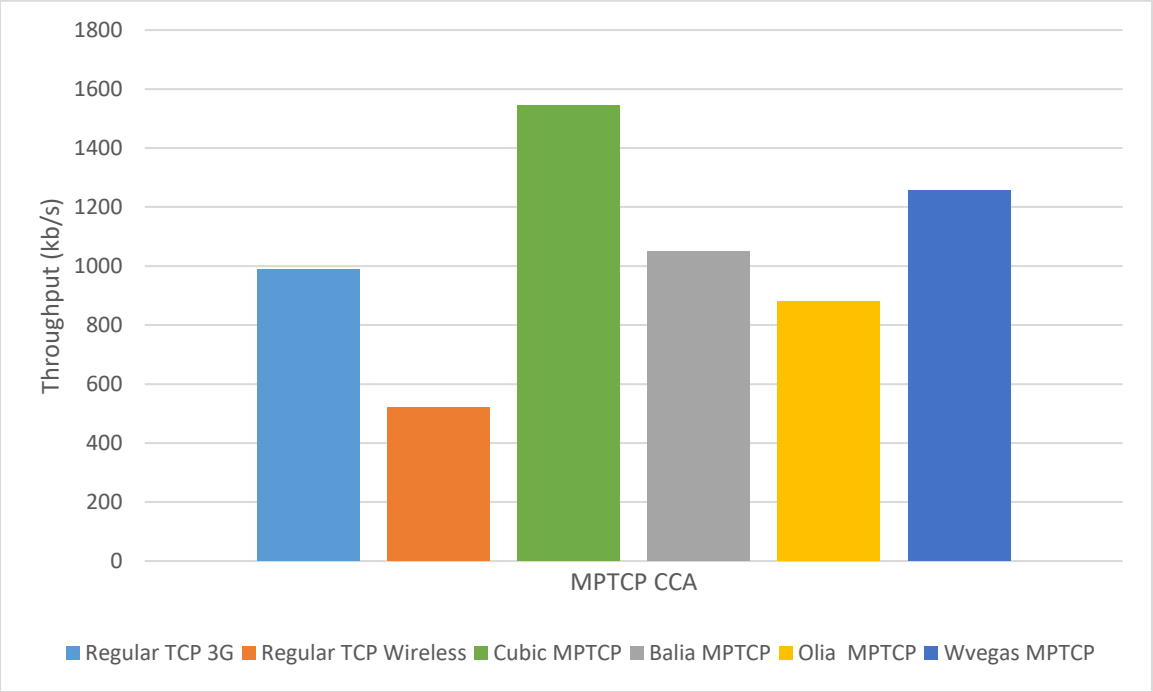

Figure 4.4 Average throughput per CCA for scenario #2

4.4. Performance Analysis of Scenario 3

From Figures 4.5 (a) to (e), show clearly that MPTCP Cubic has a higher throughput than Balia, wVegas, Olia and Regular TCP. Furthermore, Figure 4.6 shows clearly that Olia outperforms other MPTCP CCA, this occurred because this algorithm is a loss-based algorithm, which means that the slow-start algorithm will increase the transmission rate until a packet loss is detected, causing traffic to continue to increase. Although wVegas performed worse than Olia, it outperformed the Balia algorithm, because it distributed approximately evenly between the interfaces, this algorithm has benefited from stability of 3G interface. Unlike the Balia MPTCP CCA which put a low amount of traffic on the 3G interface, as a result, its ability to shift traffics was insufficient to defeat the wVegas algorithm.

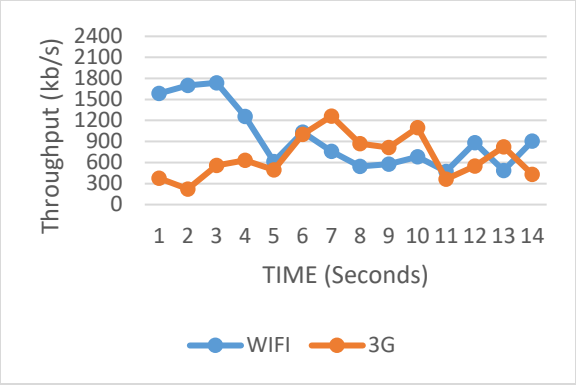

Figure 4.5 (a) Regular sub flows Throughput

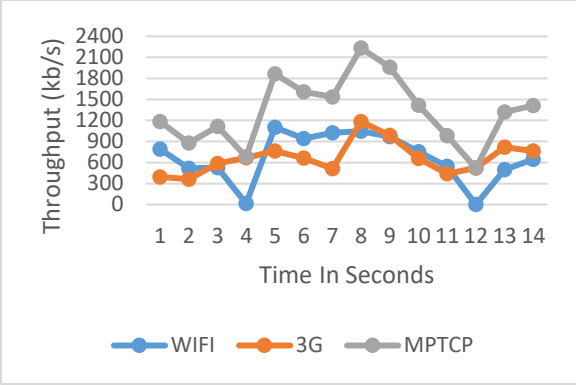

Figure 4.5 (b) Cubic sub flows Throughput

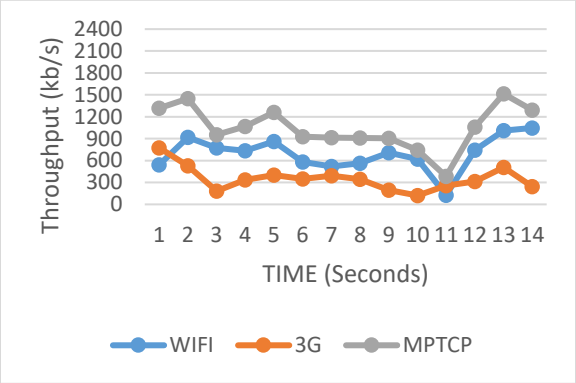

Figure 4.5 (c) Balia sub flows Throughput

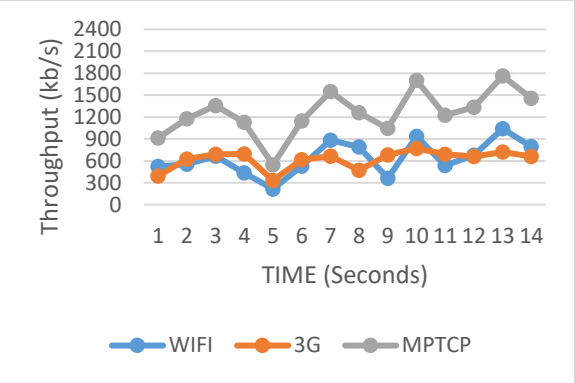

Figure 4.5 (d) Wvegas sub flows Throughput

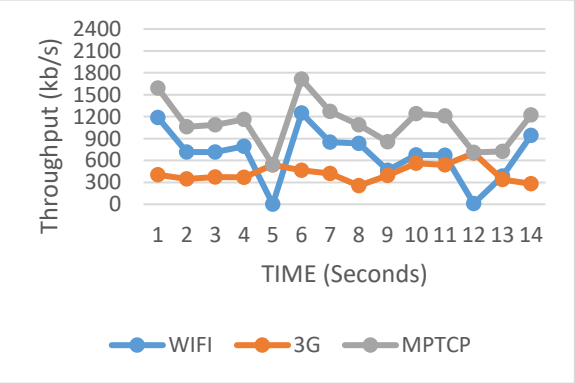

Figure 4.5 (e) Olia sub flows Throughput

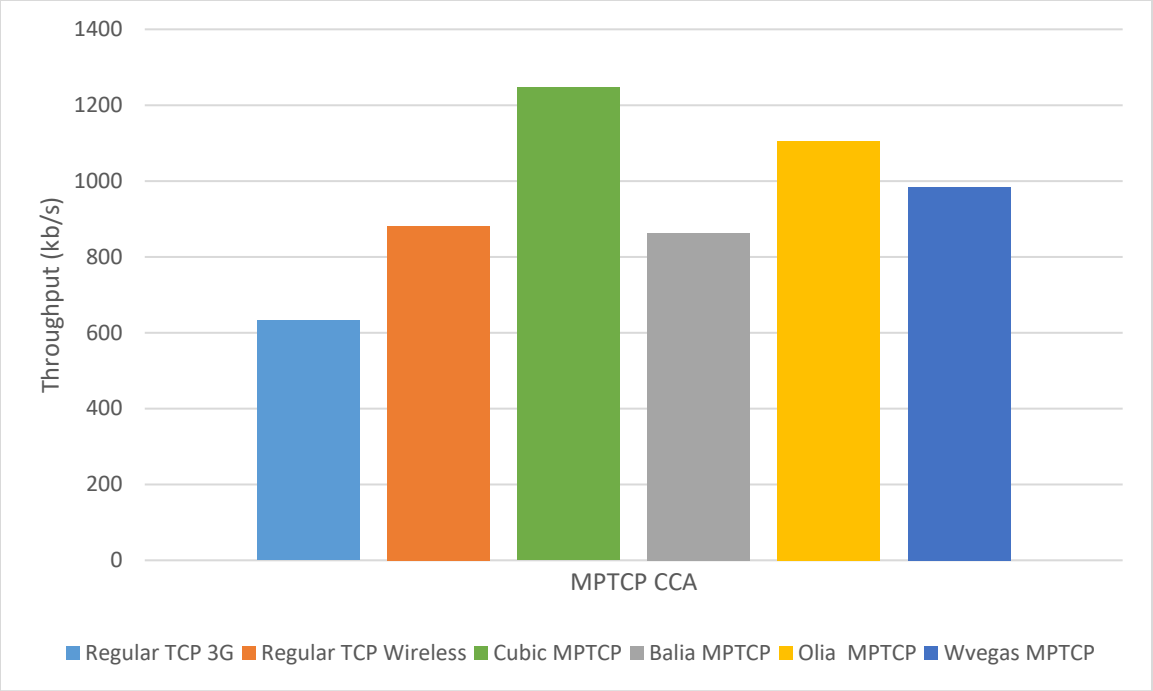

Figure 4.6 Average throughput per CCA for scenario #3

4.5. Performance Analysis of Scenario 4

From Figures 4.7 (a) to (e), show clearly that MPTCP Balia MPTCP Olia has a higher throughput than regular TCP, cubic MPTCP and wVegas, Olia and Regular TCP. Furthermore, Figure 4.8 show clearly the performance of Cubic MPTCP CCA and regular TCP was poor. This means that when there is a movement that causes a packet loss in addition to competition, the MPTCP CUBIC performs poorly, it performs nearly to the regular TCP algorithm. Figure 3.10 shows clearly that wVegas MPTCP defeat regulate TCP and Cubic MPTCP CCA. However, when compared to the algorithms Olia and Balia, it lost by a small margin. Because wVegas is a delay-based algorithm, whereas Olia and Balia are loss-based algorithms. Finally, this experiment showed that in the event of a packet loss and a competition, coupled algorithms are superior to those that do not belong to this family of algorithms.

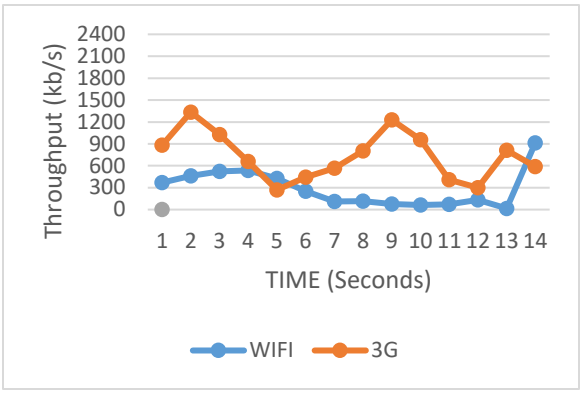

Figure 4.7 (a) Regular sub flows Throughput

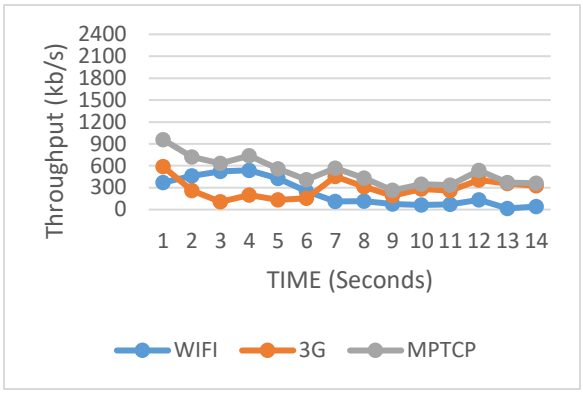

Figure 4.7 (b) Cubic sub flows Throughput

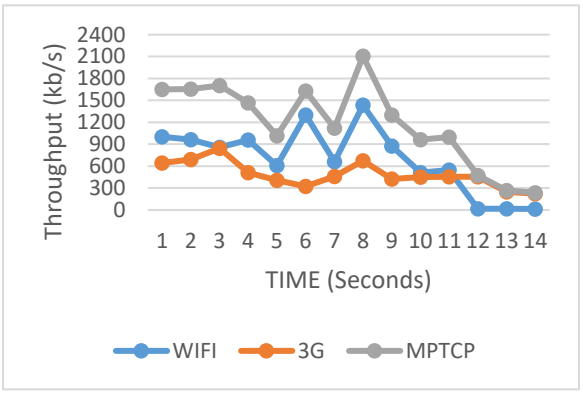

Figure 4.7 (c) Balia sub flows Throughput

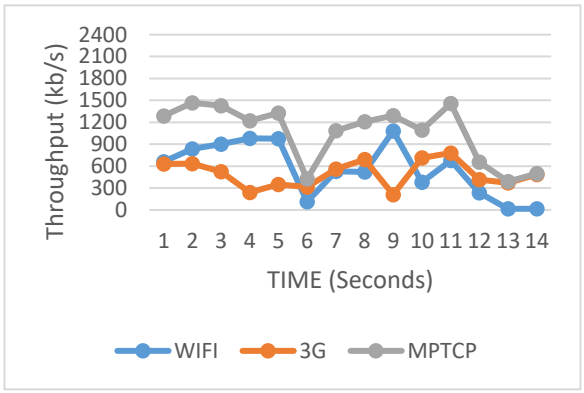

Figure 4.7 (d) Wvegas sub flows Throughput

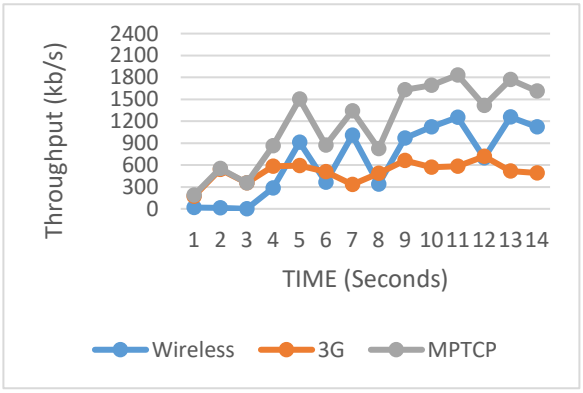

Figure 4.7 (e) Olia sub flows Throughput

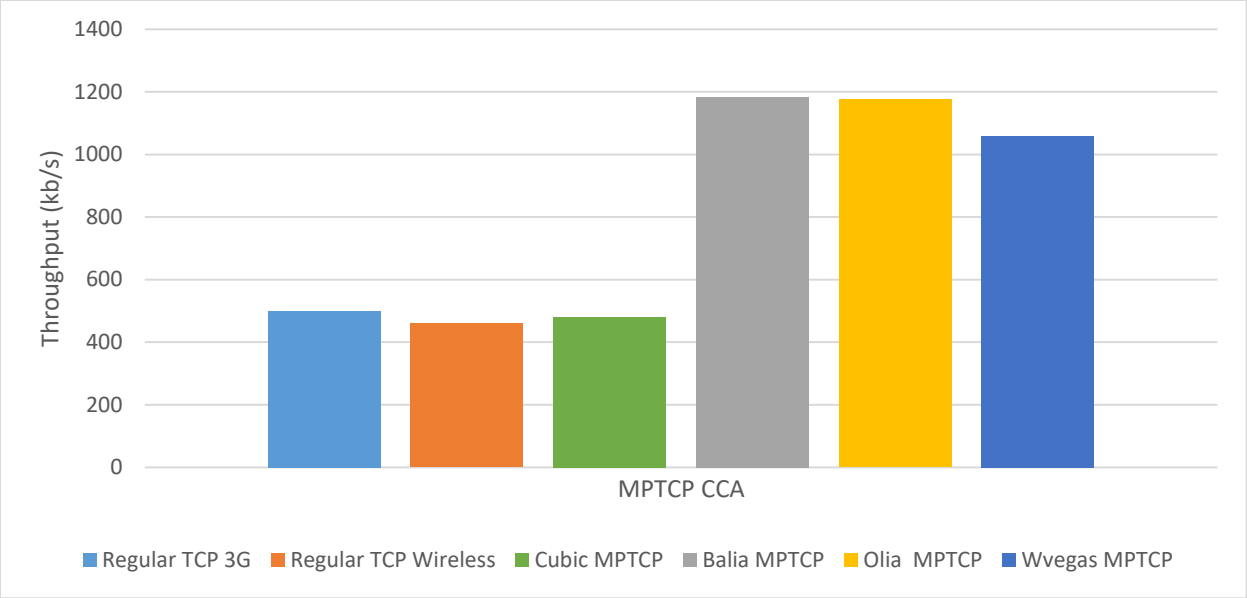

Figure 4.8 Average throughput per CCA for scenario #4

4.6. Discussion

The congestion management algorithm is one of the most significant components of the TCP algorithm. It is primarily responsible for adjusting throughput, so that it increases and decreases the throughput as needed. In this thesis, we evaluated a number of the most recent MPTCP CCA that have been developed. Each TCP connection also has a congestion window that controls the number of packets, which can be sent to the destination without waiting for an acknowledgement from the source. When using the MPTCP CCA, the same approach is used, so that each sub-flow has CC parameter. This parameter can be used to control the number of packets that are sent to the destination without waiting for an acknowledgment to arrive. As with TCP CCA, there are a variety of MPTCP CCA, including Cubic, Balia, Olia and Wvegas. Furthermore, many of the researches have been conducted within datacenters, but only a few have been done on a large scale over the internet, so the primary goal of this thesis is to evaluate some of the most recent CCA over single path, and multipath with download destinations from the internet. All of the results from this research are derived from the Linux operating system, which has

a specific kernel that is tailored for MPTCP CCA evaluation. In addition, the researcher examined each scenario separately, highlighting the algorithms' weaknesses and strengths.

Chapter 5: Conclusions and Future Work

5.1. Research Limitations

We chose the experiment's methodology, where the outcomes of this type of research are always subjected to human error since we always require control of metrics under specified conditions. Due to the RTT mismatch between 3G and Wi-Fi, the use of ADSL and 3G together restricted the ability of the MPTCP Coupled CCA to move the data from the congested paths to the least congested paths. The experiments consume a lot of time because each experiment is repeated numerous times, and then the average of all of these results is calculated. Furthermore, in each experiment, we must isolate the results and variables; after all testing is completed, we must combine the results and variables; this takes time and may result in some human errors. The internet connection used for this experiment, ADSL, is not a stable connection, it is possible that an error occurred as a result of a disconnection on the ADSL network, and this interruption in service had an impact on the research results. Experimental research always provides us with a yes or no answer, and also demonstrates whether the experiment works or not as the theory states. But, if you obtain a partial result, this result will be classified as a no answer because we do not receive all of the results, and sometimes do not have an explanation for why such results occurred.

5.2. Conclusions and Future Work

We introduced the operation of CCA, demonstrated algorithms that balanced traffic between sub-flows over WIFI and 3G, and coupled algorithms that shifted traffic based on the availability of sub-flows or the strength of the signal. Additionally, we demonstrated algorithms that used a mechanism of fairness to combat the aggressiveness of some algorithms that share the same path. As example. MPTCP algorithms prevents their sub-flow from being aggressive. This was shown when the MPTCP Cubic algorithm was tested with competing algorithms. In this case the sum of all sub-flows throughput of Cubic MPTCP CCA must be smaller or equal the optimum path bandwidth. This experiment shows that Cubic MPTCP CCA is the best algorithm for multipath communication because it works in all situations as the best algorithm, or very close to the best algorithm in some situations. Until now, there has been no algorithm that works as well as CUBIC at trading between utilized links, fairness, and network stability, and this algorithm works equally well in MPTCP and regular TCP. With the increased use of internet-connected devices such as smartphones, there is a need to develop an efficient interface. At this point, these devices use the switch method to determine the optimal network based on the network's heuristics. It is particularly important for smartphones to balance between all interfaces whenever there is more than one network available, particularly between 3G and WIFI, and to replace this method with the previous one in smartphones, which is based solely on binary decisions, and does not allow to use two interfaces concurrently, which can provide with double bandwidth, and also provide extremely high availability. By continuing to work on MPTCP algorithms. We can eventually reach a point where all internet devices can be used.

References

Hijawi, H., & Hamarsheh, M. (2016). PERFORMANCE ANALYSIS OF MULTI-PATH TCP NETWORK. International Journal of Computer Networks & Communications (IJCNC), 8(2), 145–157.

Abed, G., Ismail, M., & Jumari, K. (2012). Exploration and evaluation of traditional TCP congestion control techniques. Journal of King Saud University - Computer and Information Sciences, 24(2), 145–155.

Wischik, D., Raiciu, C., Greenhalgh, A., Handley, M., & University College London. (2011). [Review of Design, implementation and evaluation of congestion control for multipath TCP]. In Proceedings of the 8th USENIX conference on Networked systems design and implementation (pp. 99–112).

Lubna, T., Mahmud, I., & Cho, Y.-Z. (2018). Dynamic Congestion Control Algorithm for Multipath Transport Protocols. 2018 International Conference on Information and Communication Technology Convergence (ICTC).

Scharf, M., & Ford, A. (2013, March). Multipath TCP (MPTCP) Application Interface Considerations. RFC Editor. Retrieved from https://www.rfc-editor.org.

Ahmad, M., Ngadi, M., & Mohamad, M. (2015). EXPERIMENTAL EVALUATION OF TCP CONGESTION CONTORL MECHANISMS IN SHORT AND LONG DISTANCE NETWORKS. Journal of Theoretical and Applied Information Technology, 71(2), 145–155.

Khalil, E. (2012). A Modified Congestion Control Algorithm for Evaluating High BDP Networks. Journal of Theoretical and Applied Information Technology, 12(11).

Al-Saadi, R. (2019). A Hybrid Loss-Delay Gradient Congestion Control Algorithm for the Internet. PHD thesis. School of Software and Electrical Engineering / Swinburne University of Technology.

Eddy, W., & Yogesh, Y. (2005). Adapting End Host Congestion Control for Mobility. Contractor Report. Nokia Research Center Irving, TX, United States: Work of the US Gov. Public Use Permitted.

Dushyant, B., (2005). “Third-Party Tcp Rate Control”. Master thesis. University of Waterloo.

Ha, S., Rhee, I., & Xu, L. (2008). CUBIC: a new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating Systems Review, 42(5), 64–74.

Jiang, Y., Zhang, J., & Guan, Q. (2014). Improvement of TCP Reno Congestion Control Protocol. Sensors and Transducers, 163(1), 308–315.

Henderson, T., Floyd, S., Gurtov, A., Nishida, Y., & University of Oulu. (2012, April). The NewReno Modification to TCP’s Fast Recovery Algorithm. RFC Editor. Retrieved from https://www.rfc-editor.org.

Chauvenne, R., & Libiolle, T. (2016). MultiPath TCP Connections: Analysis and Models. Master thesis. Computer science and engineering department,

Tomar, P., & Panse, P. (2012). A Comprehensive Analysis and Comparison of TCP Tahoe, International Journal of Computer Science and Information Technologies, Vol. 2 (5), 2467-2471.

Abrahamsson, H., Hagsand, O., & Marsh, I. (2002). TCP over High-Speed Variable Capacity Links: A Simulation Study for Bandwidth Allocation. Protocols for High-Speed Networks, Vol. 2334, 117–129.

Mishra, N., PratapVerma, L., KumarSrivastava, P., & Gupta, A. (2018). An Analysis of IoT Congestion Control Policies. Procedia Computer Science, 132, 444–450.

Cheng-Yuan, HO, Yaw-Chung, C., Chan, Y.-C., & Cheng-Yun, H. (2008). Fast retransmit and fast recovery schemes of transport protocols: A survey and taxonomy. Computer Networks, 52(6), 1308–1327.

Reiher, P., & Kleinrock's, L. (2010). Host-to-Host Congestion Control for TCP. IEEE Communications Surveys & Tutorials, 12(3), 304–342.

Hassan, I. (2016). MultiPath TCP Connections: Analysis and Models. Master thesis. Department of Informatics, university of Oslo.

Pérez, M., Alonso, S., Veiga, M., & García, C. (2014). Common problems in delay-based congestion control algorithms: a gallery of solutions. European Transactions on Telecommunications, 24(1), 1.

Leith, D., & Andrew, L. (2009). Experimental Evaluation of Delay/Loss-based TCP Congestion Control Algorithms. Proceedings of the 6th International Workshop on Protocols for Fast Long-Distance Networks.

Pokhrel, S., Panda, M., & Vu, H. (2019). Fair Coexistence of Regular and Multipath TCP over Wireless Last-Miles. IEEE, 18(3), 574–587.

Zhou, F., Dreibholz, T., Zhou, X., Tan, Y., & Gan, Q. (2017). The Performance Impact of Buffer Sizes for Multi-path TCP in Internet Setups. 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA).

Guo, W., Huang, J., & Zhang, Y. (2014). Delay-based Congestion Control for Multipath TCP. International Journal of Future Generation Communication and Networking, 7(1), 97–104.

Peng, Q., Walid, A., Hwang, J., & Low, S. H. (2016). Multipath TCP: Analysis, Design, and Implementation. IEEE/ACM Transactions on Networking, 24(1), 596–609.

Kim, G. H., Song, Y. J., Mahmud, I., & Cho, Y. Z. (2021). Adaptive Decrease Window for BALIA (ADW-BALIA): Congestion Control Algorithm for Throughput Improvement in Nonshared Bottlenecks. Electronics, 10(3), 294.

Christiansen, P. (2021). How Fast Is Fiber. Retrieved from https://www.highspeedinternet.com.

Sematext. (2022). Response Time. Sematext. Retrieved from https://sematext.com.

Giuliano, S., Schonfeld, D., & Khokhar. (2019) “INTRODUCTION TO DIGITAL SYSTEMS AND COMPUTER ENGINEERING”, The Electrical Engineering Handbook. Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, Illinois, USA. Vol. 1, pp. 401–425.

stackpath. (2020). WHAT IS CWND AND RWND. Stackpath. Retrieved from https://www.stackpath.com.

https://www.cloudflare.com/learning/performance/glossary/what-is-latency/

Verma, P., & Zhang, F. (2020). Bandwidth and Throughput of Networks: Packet Switched Networks. Textbooks in Telecommunication Engineering, 63–70.

Kassim, M. (2011). An Analysis on Bandwidth Utilization and Traffic Pattern for Network Security Management. International Conference on Future Information Technology, 13.

APPENDIX
Part A
MPTCP download and installation

#wget −q −O − ht tp : *//multipa th−tcp .org /mptcp. gpg.key | sudo apt−key add − //The gpg key is added as a trusted source*

The kernel is installed by running the command below.

1. #sudo apt−get update

2. #sudo apt−get instal linux−mptcp

To boot the MPTCP kernel, the grub bootloader must be modified. This is done by using the apt-get command to install grub-customizer:

1. #sudo apt−get install grub−customizer // installs the program

2. #sudo grub−customizer // runs the program

In order for the client to be able to use numerous sub-flows at the same time, currently, the setup is done by running a script containing the commands shown below.

Part B
Network setup

#this creates two different routing tables that we use based on the source- address.
ip rule add from (ip of first interface)table 1
ip rule add from (ip of second interface)table 2

Configure the two different routing tables
ip route add (the first subnet)/24 dev eth0 scope link table 1
ip route add default via (the default gateway) dev eth0 table 1

ip route add (the second subnet)/24 dev eth1 scope link table 2
ip route add default via (the gateway for the second subnet) dev eth1 table2

Default route for the selection process of normal internet-traffic
ip route add default scope global nexthop via (the default gateway)dev eth0

The MPTCP must be enabled in the kernel before it can be used. This is done by selecting a kernel flag to enable the MPTCP option, using the following command.

#sysctl -w net.*mptcp.mptcp_enabled=1*

The path manager of the Multipath TCP is set to be default, using the following command.

sysctl -w net.mptcp.mptcp_path_manager = default

The scheduler of the Multipath TCP is set to be default, using the following command.

sysctl -w net.mptcp.mptcp_schedular = roundrobin

To print network interface statistics the below ifstat command was used.

#timeout 15 ifstat –I (the name of the first interface), (the name of the second interface)

ABSTRACT IN ARABIC

ملخص الدراسة

التحكم بالازدحام على الشبكة له أهمية كبيرة في جميع تطبيقات الشبكة، وقد ثبت أن مقدمي خدمات الإنترنت بحاجة إلى تطوير فهم أفضل للمفاهيم والاستراتيجيات الحديثة للتحكم بالازدحام، على سبيل المثال، كلما زاد تعقيد تصميم الشبكة يزداد تنوع تصميمات خوارزميات التحكم في الازدحام، وهذه الخوارزميات مهمة لتحسين أداء الشبكة.

قامت فرقة العمل المعنية بهندسة الإنترنت بتطوير بروتوكول نقل البيانات في المسارات المتعددة، مما أتاح توزيع تدفق البيانات عبر العديد من المسارات. يمكن لهذه الاستراتيجية تحسين كفائة عمل الشبكة، وتصبح متاحة للاستخدام العالي دون أن يكون هنالك عدوانية مع التدفقات المتنافسة.

في هذه الدراسة تم التحقق من سلوك عدد من أفضل خوارزميات التحكم بالازدحام في المسارات المتعددة، وذلك بغرض وصف الأداء الأمثل لتلك الخوارزميات في مجموعة متنوعة من المواقف. والخوارمية الجيدة هي الخوارزمية التي كان ادائها أفضل من خوارزمية نقل البيانات في المسار الواحد.

علاوة على ذلك، لقد أثبتنا أن بعض خوارزميات التحكم بالازدحام قد الحق ضرر بالأداء العام للشبكة في مواقف معينة. بالاعتماد على تحليل عميق لمراجعة الأعمال السابقة، تم بناء سيناريوهات تحليل الأداء لمعرفة اي خوارزمية لها سلوك أفضل في معظم المواقف، ثم تم مقارنة النتائج مع فرضيات الدراسة، وأثبتت نتائج الدراسة أن جميع الخوارزميات متعددة المسارات التي تم استخدامها في هذه الدراسة ليست مثالية، وأثبتت هذه الدراسة أن أحد هذه الخوارزميات من نوع كيوبيك قابل للتطبيق على نطاق واسع، وقد تفوق على باقي الخوارزميات التي تم استخدامها في هذه الدراسة، كما أن جميع التجارب تمت باستخدام نظام تشغيل لينكس مفتوح المصدر، و استخدامت الإنتاجية كمقياس رئيسي، وذلك لتحقيق أقصى قدر من الإنتاجية.

أظهرت التجارب العملية في هذه الدراسة ان الخوارزمية من نوع كيبويك حققت أفضل النتائج بإستثناء السيناريو الرابع، فقد حققت في السيناريو الأول انتاجية بمعدل 1908 كيلو بت في الثانية، متفوقة على أقرب خوارزمية لها بفارق 43 %. وفي السناريو الثاني حققت انتاجية بمعدل 1544 كيلو بت في الثانية، متفوقة بفارق 18.6 % غلى أقرب خوارزمية لها، وفي السيناريو الثالث حققت انتاجية بمعدل 1247 كيلو بت في الثانية متفوقة بفارق 14% عن أقرب خوارزمية لها .