Improving Data Center Network Performance through Path Diversity

Undergraduate Project Proposal

by

Charles Joshua Alba
2013-06878
B.S. Computer Engineering

Kyle Dominic Gomez
2013-25650
B.S. Computer Engineering

Rene Josiah Quinto
2013-14854
B.S. Computer Engineering

Adviser:

Professor Roel Ocampo
Professor Isabel Montes

University of the Philippines, Diliman
March 2018

Abstract

Improving Data Center Network Performance through Path Diversity

Datacenter networks allow the operation of Internet services by partitioning requests to multiple servers, and aggregating results back to form the response sent to the users. This requires servers within the datacenter to communicate efficiently and rapidly. Services, differentiated by their workload profiles, require a network that can guarantee high throughput, and low flow completion time. To achieve these, this paper focuses on Multipath TCP (MPTCP) as a transport protocol in the context of datacenters, by making better use of the network topology. In addition, changes in the packet forwarding protocol within network switches and the Generic Offload Receiver (GRO) layer within network servers will be employed to make up for MPTCP's undesired behavior within datacenters such as network link hotspots. With this, we hypothesize an increase in throughput and decrease in the flow completion time. To test this, we created a simulated datacenter environment wherein various tests were done to measure throughput, goodput, flow completion time, and mean network utilization, tweaking flow types, network traffic conditions, switch forwarding protocol, and reorder resiliency in hosts in the process. Analyzing the data, the study found that packet spraying turned out to be beneficial for the network in terms of throughput, goodput, and mean network utilization. Reorder resiliency, on the other hand, resulted in a decrease in the performance of the network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To keep up with increasing demand for online services, requests and operations are usually serviced by partitioning tasks into multiple servers within a datacenter. These servers are then arranged in a special topology to ensure quick intercommunication between each other, regardless of packet stream length or size. As a consequence, servers have multiple paths between each other. There lies a promising performance benefit by taking advantage of this path diversity, in which the traditional transport protocol, TCP, cannot provide.

1.1 Servicing Increasing Demand with Datacenter Networks

Several companies and institutions (Google, Amazon, and Microsoft, to name a few) provide a wide variety of online services such as video streaming, online gaming, VoIP, file sharing, and simple web searching. These online services usually differ in their workload profiles [1, 2]—faster connections result to better performance for file sharing services, while connection stability is the main concern for video streaming services.

The demand for online services has been increasing steadily due to their popularity. Moreover, the competitive market have pushed for innovations to improve services, resulting in increased computational load [3]. To keep up with demand that increases in both quantity and complexity, companies need to expand and upgrade their resources, increasing costs [4, 5]. One way to meet this demand is to distribute service requests to multiple servers, then aggregating responses to form the final result (also known as the partition/aggregate design pattern) [6]. Infrastructures built with this demand distribution property are called datacenter networks (DCNs).

Apart from meeting the increasing demand, this provides several benefits. Firstly, by having redundant servers, services can be maintained even during partial network maintenance

[3]. Secondly, an organized and redundant network infrastructure eases management [3]. Finally, distributing requests to different servers increases server utilization, proving to be more cost-effective and scalable [7].

1.2 Connection-Rich Nature of Datacenter Network Topologies

Partitioned requests are distributed among servers within a datacenter (end-hosts) through packet streams (flows). Since majority of the flows stay within a datacenter [8], the network topology must guarantee a speedy and reliable connection between local end-hosts. These properties can be characterized with sufficient interconnection bandwidth [9, 10] and fault-tolerance [11].

Requests must be serviced quickly and aggregated back to the user, which means that the server response delays also add up and affect the performance of the service [6]. End-hosts also need to keep their internal data structures fresh [6] through frequent large transfers of data. Therefore, datacenters must also guarantee low latency for short flows, and high throughput to service large flows [6, 2].

Current datacenter network topologies guarantee the previous characteristics through different approaches. Switch-centric topologies [10, 12] rely solely on switches and routers to forward traffic across the network, which ensures speedy delivery due to specialized hardware at the expense of scalability. Server-centric topologies [13, 14] allow servers to forward packets as well, however software routing may be slow and can affect host-based applications [15].

Topologies can be also designed to be modular [14] or hierarchical [12] to increase network capacity through scalability [9]. This introduces multiple redundant links and paths between servers within the network, establishing a connection-rich nature [2, 16].

1.3 Better Network Utilization through Multiple Paths

Host-to-host connectivity within a datacenter was originally facilitated using the Transmission Control Protocol (TCP) [6, 17]. TCP ensures robustness in the delivery of packets between end-hosts in the presence of communication unreliability [18].

TCP connections make use of a single network path between end-hosts [19]. Since it is typical for two end-hosts in a datacenter network to have multiple paths, TCP is unable to fully utilize the available paths between end hosts [20], and may not be able to achieve optimal network utilization.

To make use of the available resources in a path-diverse network, Multipath TCP (MPTCP)

was proposed as an experimental protocol by the IETF. MPTCP uses parallel TCP connections [21, 22], which increases throughput and resiliency of the network [20].

1.4 Outline of the Document

The rest of the document is presented as follows. The next chapter will discuss different improvements suggested for MPTCP based on different metrics such as reliability, and a global view of the network. Additionally, some issues in unmodified MPTCP are also discussed and studied in depth. Chapter 3 discusses the problem statement, objectives, and scope and limitations of the project. Section 4 and 5 discuss the steps taken, results, discussion, and conclusions for the following experiments.

Chapter 2

Review of Related Work

While MPTCP promises performance benefits such as increased throughput between two hosts, it suffers from problems which hinder or negate its advantages. The nature of datacenter network traffic and topology presents two potential setbacks for MPTCP. First, short packet streams (short flows) might experience a higher latency due to MPTCP's connection establishment mechanism. Second, large packet streams (large flows) might suffer from lower throughput due to MPTCP's congestion and subflow mechanism. In addition to the presented solutions to these problems, a closer look is given to switch-based packet spraying, combined with host-based reordering resiliency, as a potential improvement over MPTCP.

2.1 Connection Establishment of MPTCP Penalizes Short Flows

MPTCP, being designed as an extension for TCP [21], inherits its congestion control mechanisms. One of which is Slow Start [23], where each subflow starts sending a limited amount of packets, then exponentially increases its sending rate until a link capacity/threshold is reached. Exceeding link capacity can result to packet loss due to switch overflows, making the receiver fail to acknowledge the packet. This subsequent action will cause the sender to retransmit the packet due to its internal retransmission timeout, or due to receiving three duplicate acknowledgements (TCP Fast Retransmit) for a previously sent packet, in addition to reducing the sending rate of the sender.

Since majority of the flows on a datacenter are short flows [24] and are of a bursty nature [6], MPTCP would cause these flows to stay in the Slow Start phase [25], where the sending rate is initially limited, then is increased drastically. Therefore, when short flows lose packets due to congestion, retransmission is necessary, and this increases the flow completion time [26].

In addition, MPTCP also inherits TCP's handshake protocol, in order to create and establish subflows between end-hosts. Since MPTCP cannot discern between short and long flows, it cannot decide as to how many subflows should be established to optimally send packets across the network. Therefore, a mismatch in the number of subflows, specifically for short flows would further increase the flow completion time due to the extra packets required to establish a connection.

To minimize the flow completion time for short flows, connections can start by spraying packets over multiple paths. Connections then can switch back to MPTCP upon reaching a certain amount of packets [26], to make use of MPTCP's benefits over long flows such as increased throughput and better congestion control. However, the introduction of packet spraying in addition to using MPTCP would open this specific implementation to both problems associated with each.

2.2 Decreased Throughput due to Redundant Retransmissions

MPTCP uses Equal-Cost Multipath (ECMP), a switch-based algorithm, to forward packets through the network [17]. ECMP bases its forwarding on a 5-tuple (source IP, destination IP, source port, destination port, protocol number) [27] which is hashed and used to forward packets to paths. This means that all data belonging to one subflow will strictly follow one network path. Selected network paths by ECMP may traverse links that have already been chosen by another subflow, which will effectively decrease the maximum available capacity of the link. Areas in the network that are close to exceeding link capacity are called hotspots. Colliding subflows may exceed the link capacity for any given link in a datacenter network, resulting in full switch queues. Full switch queues will then lead to dropped packets, which in turn, decrease mean network utilization [9].

Duplicating packets across multiple paths (bicasting/multicasting) in an MPTCP connection can be used to mitigate the decrease in throughput due to dropped packets [28]. In full bicasting, each connection transmits a copy of all packets sent across multiple subflows, while selective bicasting only duplicates packets that are sent as a retransmission of lost packets.

However, the increase in throughput for individual subflows brought about in both full and selective bicasting may not be optimal in a datacenter. This is, in part, attributed to the duplicating nature of bicasting wherein redundant data is sent through the network's links and consumes resources for other subflows. In addition to redundant data packets, control packets such as the ACK packets in TCP add to network traffic. This added traffic to the network can be perceived as a loss in throughput.

2.3 Diffusing Network Hotspots through MPTCP with Packet Spraying

As previously mentioned, network hotspots create areas of congestion in datacenter networks, throttling the throughput of the connections that pass through these hotspots. One way to avoid this is to use packet spraying [29]. Packet spraying spreads a sequence of packets through different candidate ports instead of forwarding packets of a single flow through a chosen port according to its hashed flow identifier as seen in ECMP. This scheme prevents the collision of multiple paths onto specific links (hash collision), and spreads the packets in the hopes of evenly distributing the load through all relevant links of the network.

However, distributing packets of a flow into different paths can lead to out-of-order packets if the paths have significantly different end-to-end delays [29]. End-hosts receiving packets that do not arrive in the order that they were sent may falsely perceive that the network is lossy and request for retransmissions [18]. The sending end-host will receive these retransmission requests and will falsely perceive that there is congestion in the network, cutting the congestion window in half which will then result in the reduction of throughput.

One way to avoid false congestion due to packet reordering is to adjust the time threshold of the end-hosts before they request for retransmission [26], effectively making end-hosts more tolerant to delays in the network. However, an improper configuration (mismatch) of this threshold will result in an increased delay before sending retransmission requests in actual congestion experienced in the network, greatly decreasing the flow completion time and penalizing short flows.

To increase the resiliency of the end-hosts against out-of-order packets while avoiding a threshold mismatch, one solution would be to increase the buffer size of the Generic Receive Offload (GRO) layer of the end-hosts [30]. The GRO layer works underneath the transport layer, and is responsible for bundling packets to decrease computational overhead versus individual per-packet processing. Increasing the buffer size enables it to accept packets that arrive earlier than those preceding it thereby increasing its resilience towards out-of-order packets. This is based on the assumption that, in a datacenter, delay differences between paths are significantly less than in non-datacenter networks.

2.4 Drawbacks Caused by MPTCP with a Global View

With Software Defined Networking (SDN), a global view of the network state can be provided to a central controller [31, 32, 33]. This enables network protocols to make use of global

network information, such as link utilization and end-to-end delays [34, 35], to create more informed decisions. SDN approaches have introduced several benefits to MPTCP, such as increased throughput and network resiliency, by controlling the number of subflows and scheduling the routes of different connections to avoid both network hotspots and out-of-order packets [34].

These benefits, however, comes at the price of controller overhead [34]. SDN controllers need to communicate regularly with the network switches in order to issue forwarding rules to adapt with the situation, causing delays. Since most congestion events that happen in datacenter networks are because of traffic bursts that only lasts for a few microseconds (microbursts) [36], controller overhead becomes significant, and so SDN implementations cannot respond fast enough. Due to controller overhead, SDN implementations can also increase the flow completion time of connections, penalizing short flows. Moreover, SDN implementations generally do not scale with a large number of connections as the controller overhead also increases with the number of active connections [34].

Chapter 3

Problem Statement and Objectives

3.1 Problem Statement

Common topologies in datacenter networks exhibit symmetry and path diversity [2]. This ensures that multiple paths of equal costs exist between two end-hosts [29]. In addition, an ideal datacenter network must guarantee high throughput for large flows, and low latency for short flows [6, 2].

Multipath TCP (MPTCP), an experimental protocol by the IETF, is a TCP extension that uses multiple paths over many connections [21]. By using multiple paths simultaneously, MPTCP aims to increase network utilization, specifically throughput, and increase "resilience of connectivity" [22]. This is done by employing Equal Cost Multi-Path (ECMP) [17], a switch-based forwarding algorithm. ECMP hashes subflows into separate paths using the packet's header information. However, hotspots in the network may occur when flows are hashed to paths with one or more links overlapping with each other [9], usually exceeding the link capacity. Because of this, the network experiences a drop in network utilization due to sender backoff [29].

Random packet spraying, an alternative to ECMP, can be used to resolve hotspots as it allows for a greater granularity for load balancing [2]. However, this can result in reordered packets at the receiver, triggering false retransmissions upon receipt of three duplicated packet acknowledgements [20], in addition to drastically reducing sender throughput [37]. This can be minimized by dynamically changing the retransmission threshold. However, a positive mismatch on the threshold may mean a slower response time for actual packet drops [37].

Minimizing retransmission due to out-of-order packets can be done by supplementing the function of the Generic Receive Offload (GRO) layer to fix the order of packets [1] in addition to merging packets together. This not only reduces computational overhead compared to per-

packet processing, but makes the end-hosts more tolerant to packet reordering, and thus reduces retransmissions. However, reordered packets must arrive within a short delay of each other since the switch queues are limited and timeouts are smaller.

Also, to maintain backwards compatibility with TCP, MPTCP also suffers from the complexity in connection establishment [29] and slow start [23], which in turn penalizes short flows through higher flow completion times. To maintain lower flow completion times and lower latency for short flows, a specified amount of packets are sprayed through multiple paths initially before switching to MPTCP [26]. However, this means that it inherits problems from both implementations altogether.

Better network utilization can be also achieved using Software-Defined Networking [34, 35] with MPTCP. With a global view of the network, it can better utilize path diversity, have a better gauge on the number of subflows per connection, and ideally minimize the receipt of out-of-order packets. This solution benefits large data flows, but due to the added controller overhead, it may penalize short flows. In addition, it may also have some scalability issues [35].

To combat delay and lower throughput caused by multiple timeouts on wireless networks, retransmission redundancy was implemented using full and partial bicasting over MPTCP [28]. This may not necessarily be effective in the datacenter setup as redundant packets may cause more false retransmissions.

We hypothesize that an increase in overall throughput and network utilization in MPTCP can be achieved through implementing packet spraying in the network instead of ECMP. Since packet spraying can introduce an eventual decrease in throughput, we hypothesize that this can be mitigated by creating reorder-resilient end hosts. By comparing the results we see from different experiments, we strengthen the basis for considering a reorder-resilient network for future datacenter networks, as well as potentially contribute to the growth of MPTCP as an experimental protocol.

3.2 Objectives

The objectives of this work are as follows: First is to experimentally prove that MPTCP benefits large flows, but penalizes short flows. Next, understand and prove that network hotspots occur due to ECMP-based switches. Lastly, observe and analyze the effects of packet spraying switches, as well as reorder-resilient hosts, to minimize network hotspots, and in turn benefit both short and large flows.

3.3 Scope and Limitation

The project will focus on datacenter networks, and assume ideal working conditions. More specifically, this project does not consider the possibility of switch failures, host failures, link damages that could cause degradation of performance or even disconnections. Among datacenter topologies, only fat tree topologies and possible variants will be considered.

While the nature and topology of datacenter networks are highly distinct from the vast majority of the Internet, or wide area networks (WANs), the results and observations presented in this paper may potentially apply to WANs as well.

As this paper relies heavily on experiments done through network simulations, this project cannot guarantee the realization of actual or realistic datacenter network traffic, but tests will be made to mimic certain datacenter network conditions such as worst-case scenarios.

Chapter 4

Methodology

Taking into consideration the penalties incurred on short flows from MPTCP connection establishment complexity, as well as the penalties incurred on long flows due to the network hotspots that arise as a consequence of ECMP routing, we chose to measure flow completion time, and end-host throughput and goodput, respectively. Mean network utilization was also measured to gauge the effectivity of each protocol in terms of how well the available network routes were used.

To properly attribute each change in the above metrics, several possible permutations of the routing behavior, transport protocol, flow types (or payload sizes) end-host pairs, and network conditions were tested. These metrics were extracted and compared against each other considering the varying parameters between each test.

4.1 Preliminary Work

Smaller tests using MPTCP were done to further increase the researcher's understanding of the protocol and its interaction with Mininet. MPTCP was confirmed to be working as indicated in the laboratory experiment of an SDN class [38]. An overview of the topology can be found in Figure 4.1. To simplify things, router `r2` is configured to forward packets, acting as a router[1]. All links are capped at a speed of 10 Mbps (i.e., the minimum supported transmission rate of both devices connected to the link is 10 Mbps).

[1]From this point onwards, we use the terms switch and router interchangeably to mean devices that do layer 2 forwarding and layer 3 routing.

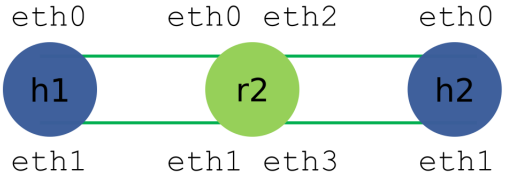


Figure 4.1: Mininet topology for MPTCP tests. Links are of 10 Mbps speed. Topology copied from [38]

Routing Protocol	Path Manager	Initiates subflow creation	Number of used IP address pairs	Number of used port number pairs	Number of subflows
TCP	None	N/A	1	1	1
MPTCP	default	No	As sender: 1 As receiver: based on sender	As sender: 1 As receiver: based on sender	As sender: 1 As receiver: based on sender
MPTCP	fullmesh	Yes	All, combination	Configurable, default is 1	`No. of source IP-port pairs * No. of destination IP-port pairs`
MPTCP	ndiffports	Yes	1	Corresponds to number of subflows	Configurable, default is 2

Table 4.1: Comparison of MPTCP path managers and TCP.

4.1.1 Managing Multipath TCP Subflows

In the preliminary setup and tests, it was found that MPTCP can be configured using different parameters. MPTCP handles the discovery, setup, and termination of subflows through the heuristics of a path manager [18]. We considered three of the four available path managers that MPTCP provides [39].

First, the default path manager does not initiate nor announce different IP addresses, but will still accept subflows from end-hosts that do. Next, the full mesh path manager creates flows using all combinations of interfaces (device within end-host to connect to links [40]) of both end-hosts (i.e. assuming all interfaces correspond to unique IP addresses, two end-hosts with three interfaces each will have 9 subflows). Finally, the n-different-ports path manager allows the control over the number of subflows in a connection through the use of different ports. The fourth path manager called binder, isn't considered as it was designed for mobility of devices, a trait not present in datacenter networks.

In summary, we see the comparisons of MPTCP path managers in Table 4.1. To serve as a control group, TCP was characterized as well.

Routing Protocol/Path Manager	Sender throughput (in Mbits/sec)
TCP	9.78
Default MPTCP	9.43
Ndiffports MPTCP	9.46
Full mesh MPTCP	19.1

Table 4.2: Comparison of sender throughput between MPTCP path managers and TCP. The topology used was described in Figure 4.1. Throughput values were taken using command line tool *iperf*.

4.1.2 Validating the Behavior of MPTCP Path Managers

Considering the topology described in Figure 4.1, we expect to see TCP, default MPTCP, and ndiffports MPTCP to have a sender rate of 10 Mbps, whereas Full Mesh MPTCP can have a sender rate of up to 20 Mbps. This is because the first three would only use 1 IP address pair, and thus is limited to 1 subflow. But, the Full Mesh MPTCP establishes four subflows, because both hosts have two available interfaces, fully using all possible links. This was validated experimentally and the results are shown in Table 4.2.

However, using more subflows may sometimes come at a cost. For short flows, opening multiple subflows may not be necessary, as the packets may have already been sent even before a new subflow has been established. To prove that MPTCP does indeed introduce some overhead for short flows, the flow completion time was measured between TCP, Default MPTCP, and Full mesh MPTCP.

This experiment once again uses the topology in Figure 4.1. The left host (`h1`) will request for the web page from the right host (`h2`), which has a web server running. The web page to be fetched is a default directory index page, and because the web server sends a stream of small packets to complete the request, it counts as a short flow. The tests were ran 10 times with TCP, default MPTCP, and full mesh MPTCP, and the flow completion time (FCT) were noted. We defined the flow completion time as the time difference between the first `SYN` packet up to the last `ACK` packet for the last `FIN` packet.

A summary of results are shown on Table 4.3. Here, we see that TCP has the fastest flow completion time. As default MPTCP works much like TCP, but has an added establishment overhead over TCP, it has a slower flow completion time. Lastly, since full mesh MPTCP opens multiple subflows while transmitting data, packets for connection establishment of other subflows compete for the same available links, resulting in the slowest flow completion time. In addition,

Routing protocol/Path Manager	FCT (Mean, in ms)	FCT (Standard Deviation, in ms; Relative Standard Deviation)
TCP	7.86	1.53 (19.51%)
default MPTCP	8.39	1.10 (13.11%)
full mesh MPTCP	10.5	3.86 (36.70%)

Table 4.3: Mean, standard deviation, and relative standard deviation of flow completion time for different MPTCP path managers and TCP.

since flows are terminated like TCP connections, extra packets are sent to terminate all open subflows before finishing.

As the topology considered for the previous experiments is relatively simpler compared to actual datacenter topologies, we hypothesize that the observed behavior will still hold true, and the effects are amplified to a certain extent.

4.2 Building the Test Environment

Mininet was chosen to orchestrate the virtual network, its switches, and its hosts. The network topology is generated programmatically, and switch behavior can be swapped through configuration files. It also ensures that all virtual hosts are running on the same configuration as the server it is running in. This ease of use allows the researchers to implement multiple changes at once.

MPTCP is available as an ns-3 model [41] or a kernel patch [42]. Juggler [1], which modifies the GRO function to fix packet ordering, is only readily available as a kernel patch. Since the experiments require hosts that require both features to be present, it was decided that the kernel patches were to be used. The MPTCP and Juggler custom kernels were merged together and applied to the server to virtualize certain hosts.

Switches, on the other hand, can be described with its normal forwarding algorithm, which stores only one next-hop for all destinations in its routing table. However, the switches must also be capable of Equal Cost Multi-Path (ECMP), and Packet Spraying (PS), which require a routing table storing multiple next-hops. Since these forwarding behavior require inspection into packet headers and metadata, we utilized and modified software switches enabled by P4, a programming language used to process packets [43].

Technical Specifications	Value
Processor	Intel(R) Core i7-3612QE
Processor Speed	2.10 GHz
Number of Cores	4
RAM	16 GB
Operating System	`Ubuntu 16.04.3 LTS`
Linux Kernel	`Linux version 4.9.60.mptcp`

Table 4.4: Testbed Technical Specifications.

4.2.1 Testbed Specifications

Experiments and tests were ran on two SuperMicro bare metal servers with the specifications listed below. One server will be patched with the MPTCP custom kernel only, and the other with the merged MPTCP and Juggler kernel.

4.2.2 Setting up the Network Topology

For this experiment, the fat tree topology was chosen, a common and scalable data center network topology. Like all DCN topologies, the fat tree topology provides several equal cost paths between any two end-hosts. Moreover, because of the symmetry of the topology even with scaling, a Mininet topology can be easily set up with unique addressing [10]. Mininet was used for the simulation of the nodes, and Python was used to construct the topology (See Figure **??**) A $K = 4$ fat tree topology can be seen in 4.2, complete with the conventions used.

The following are the definitions and conventions followed in generating the topology.

4.2.2.1 Standard Topology Structure

For a given scaling parameter K, the topology is composed of K pods. Each pod is composed of $\frac{K}{2}$ aggregate routers and$\frac{K}{2}$ edge routers, with each aggregate router connected to all edge routers, and vice versa. Each edge router is connected to $\frac{K}{2}$ hosts, for a total of $(\frac{K}{2})^2$ hosts per pod. There are a total of $(\frac{K}{2})^2$ core routers, each connected to one aggregate router per pod.

4.2.2.2 Indexes and Names

The following numbering and naming conventions are used.

- The pods are numbered for 0 to $K - 1$.

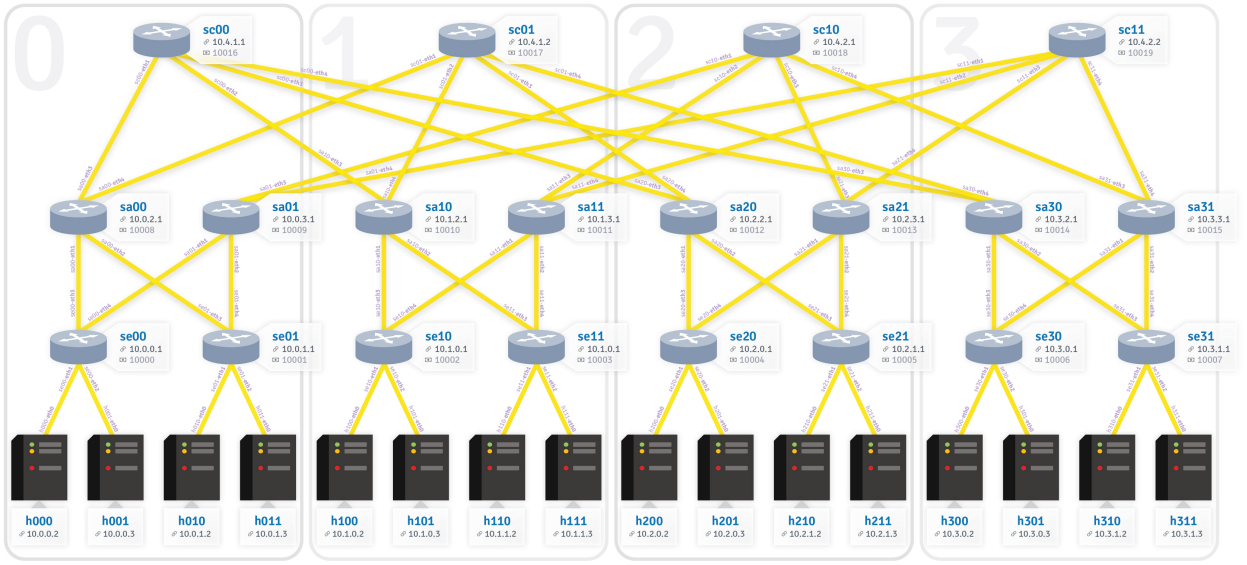


Figure 4.2: Fat tree (`k = 4`) topology in Mininet, including naming and addressing conventions.

- Edge routers are named `se<pod><i>`, where `<pod>` is the pod id, and `<i>` is the edge id within the pod, ranging from 0 to $\frac{K}{2} - 1$.
- Aggregate routers are named `sa<pod><i>`, where `<pod>` is the pod id, and `<i>` is the aggregate id within the pod, ranging from 0 to $\frac{K}{2} - 1$.
- Core routers are named `sc<i><j>`, where `<i>` and `<j>` ranges from 0 to $\frac{K}{2} - 1$. `<j>` is an identifier for all cores of the same `<i>`, and `<i>` determines which aggregate core it connects to for each pod. The significance of `<i>` and `<j>` are explained in the Links subsection.
- Hosts are named `h<pod><i><j>`, where `<pod>` is the pod id, `<i>` is the edge id of the edge router it is connected to, and `<j>` is the host id for all hosts connected to the i'th edge router.

4.2.2.3 IP Addresses

From the node names, we can map it directly to unique IP addresses [10].

- For hosts with names `h<pod><i><j>`, the IP address is `10.<pod>.<i>.<j+2>`.
- For edge routers with names `se<pod><i>`, the IP address is `10.<pod>.<i>.1`.
- For aggregate routers with names `sa<pod><i>`, the IP address is `10.<pod>.<i+(K/2)>.1`.

- For core routers with names `sc<i><j>`, the IP address is `10.<K>.<i+1>.<j+1>`.

4.2.2.4 Links

The following describes more precisely the connections between nodes[10].

- An edge router `se<POD><I>` is connected to hosts `h<POD><I><j>` for all $0 \leq j \leq \frac{K}{2} - 1$.
- All aggregate routers are connected to edge routers in the same pod, and vice versa. More precisely, an aggregate router `sa<POD><I>` is connected to edge routers `se<POD><j>` for all $0 \leq j \leq \frac{K}{2} - 1$.
- A core router `sc<I><J>` is connected to aggregate routers `sa<pod><I>` for $0 \leq pod \leq K - 1$.

4.2.2.5 Port Assignment

Each link connected to a router is assigned a port for that router. For the following descriptions, the ports are numbered from 0 to $K - 1$ (though in reality, it is usually numbered from 1 to K). The following describes the assignment for the p'th port for each type of router.

- For an edge router `se<POD><I>`, the first $\frac{K}{2}$ ports is assigned to host `h<POD><I><p>`, and the last $\frac{K}{2}$ ports is assigned to aggregate router `sa<POD><p-(K/2)>`.
- For an aggregate router `sa<POD><I>`, the first $\frac{K}{2}$ ports is assigned to edge router `se<POD><p>`, and the last $\frac{K}{2}$ ports is assigned to core router `sc<I><p-(K/2)>`.
- For a core router `sc<I><J>`, the ports are assigned to aggregate router `sa<p><I>`.

The interfaces are similarly assigned, but from `eth1` to `eth<K>` instead of 0 to $K - 1$.

4.2.2.6 Thrift Port

Thrift ports are assigned for each router, and can be used to communicate and debug with the router. The following describes how the thrift ports are assigned.

- The first thrift port is 10000, and is assigned to `se00`. Succeeding ports are increasing in increments of 1.
- The first $K * \frac{K}{2}$ thrift ports are assigned to edge routers `se00, ..., se0<(K/2)-1>, se10, ..., se<K-1><(K/2)-1>`.

- The next $K * \frac{K}{2}$ thrift ports are assigned to aggregate routers `sa00, ..., sa0<(K/2)-1>, sa10, ..., sa<K-1><(K/2)-1>`.
- The next (and last) $(\frac{K}{2})^2$ thrift ports are assigned to core routers `sc00, ..., sc0<(K/2)-1>, sc10, ..., sc<(K/2)-1><(K/2)-1>`.

4.2.3 Switch Behavior

The P4 `behavioral-model` repository [44] was forked to get its target executables (e.g., `simple_router`) and modified the source to extend their functionalities. Each behavior is defined by tables and actions coded in C++, and compiled to a JSON using the `p4c-bm` tool [45]. The JSON is fed to the target executable, and the tables are filled with entries during initialization.

4.2.3.1 Downstream Packets

For the following discussions, packets that are forwarded towards the core (host to edge, edge to aggregate, or aggregate to core) are considered to be upstream, and otherwise downstream. Since downstream packets in the fat tree topology have a unique path towards their destination, each router has only a single correct port to forward to per destination in the downstream path. Thus, for any switch behavior, the packet's destination IP Address is matched with a longest prefix match to check if the packet is headed downstream, and if so forward it to the appropriate port. All bits are matched for edge routers, the first 24 bits for aggregate routers, and the first 16 bits for core routers.

For any router forwarding upstream, each of the $\frac{K}{2}$ upstream ports to choose from is a valid path. Thus, different switch behaviors may differ in how packets are forwarded upstream. For this experiment, three switch behaviors have been implemented - `Static`, `ECMP`, and `PS` (for packet spraying), with `Static` serving as the control. The packet forwarding scheme for each behavior is discussed in detail in the next subsections.

4.2.3.2 Static

For Static behavior, every destination is assigned to a random port with equal probability during initialization, and will not change during runtime (thus the name `Static`). Thus, all upstream paths are matched in the same table as downstream packets, but using all 32 bits.

4.2.3.3 ECMP

For `ECMP` behavior, the flow metadata is hashed to determine the forwarding port. The flow metadata consists of the `source IP Address`, `source port`, `destination IP address`, `destination port`, and `protocol number`. The hash used is the CRC16 hash function, and outputs a 3-bit integer.

A 2-parameter matching is done on the table, the first parameter being the `destination IP Address`, and the second parameter being the output of the hash function. The first parameter is matched using a longest prefix match, and the second parameter is matched exactly.

For each downstream port, 8 entries are inserted into the table, one for each 3-bit integer for the second parameter. Another 8 entries are inserted for upstream packets, one for each 3-bit integer for the second parameter, and with `0.0.0.0/0` as the first parameter (match anything). With this setup, downstream ports are matched with the appropriate entry regardless of the hash value, and those that aren't matched downstream are matched according to the hash value. Each 3-bit integer is then assigned to an upstream port in a random but fair manner i.e. all upstream ports have the same number of hash values assigned to it, but randomly shuffled. Since we can have at most 8 hash values, K is limited to at most 16, though this can be increased when necessary.

4.2.3.4 PS

For `PS` behavior, the chosen upstream forwarding port is chosen uniformly at random, also known as Random Packet Spraying. According to the P4 Specifications Document [46], uniform random assignment on a field is a primitive operation, however it was found that the `simple_router` target in the P4 `behavioral-model` repository [44] did not support said operation. Thus, the source code was modified to extend support, and the target was recompiled.

The `destination IP Address` is matched with a longest prefix match to forward downstream packets, similar to Static behavior. In addition, upstream packets are matched with the entry `0.0.0.0/0` (match all) and corresponds to a special action that assigns a uniformly random upstream port as the forwarding port.

4.3 Test Design

To review, the goals of the tests are to measure the network performance given varying configurations as previously discussed. In particular, the study will test mainly throughput and flow completion time, with goodput and mean network utilization as extra data points.

For the purposes of this section, a topology with $K = 4$ will be considered. This means that there are 16 hosts, 16 switches in the edge and aggregate layers, and 4 switches in the core (see Figure 4.2).

With this in mind, this allows for 4 unique paths between two hosts.

$$Given\ k = 4,\ paths_{max} = (\frac{K}{2})^2 = (\frac{4}{2})^2 = 4 = n \tag{4.1}$$

Considering that each host uses only one interface to communicate to the entire network, preference was given to using `ndiffports` (with $n = 4$) as MPTCP's path manager in this experiment. This can be done through the `sysctl` feature. The transport protocol can be changed through `net.mptcp.mptcp_enabled`. Furthermore, we can control the MPTCP path manager using `net.mptcp.mptcp_path_manager`. In this case, we set the path manager of MPTCP-enabled hosts to `ndiffports`.

4.3.1 Network Traffic Conditions

Tests were done in two simulated network conditions. The first condition assumed a network without any activity, allowing for two hosts to communicate using all of the available resources of the network (hereinafter referred to as a *silent network*). The second condition assumed some form of simulated traffic, comparable to the performance in a live network (hereinafter referred to as a *noisy network*).

These were done to observe the changes in the design metrics in different network conditions. Note that the simulation of traffic was an approximation and may not be representative of the actual datacenter network traffic.

In the *silent network*, host pairs took turns at sending data to each other, without any other activity in the network. Pairs were chosen by random, ensuring that all hosts became a server or client at some point. The clients requested a payload of certain size from the server, one after another. This request was then repeated a certain number of times. The designations were fixed for each iteration of the test. This was done through a network bandwidth measurement tool named *iperf*.

In the *noisy network*, two hosts requested data from multiple hosts at the same time, polluting the network with activity. The network was split between two groups with one client each, and the rest acting as servers. Considering again a fat tree topology with $K = 4$, one client was chosen at random at the start of the test and seven were chosen to be file servers. Like the *silent network*, designations were fixed for each iteration of the test. This was done through an

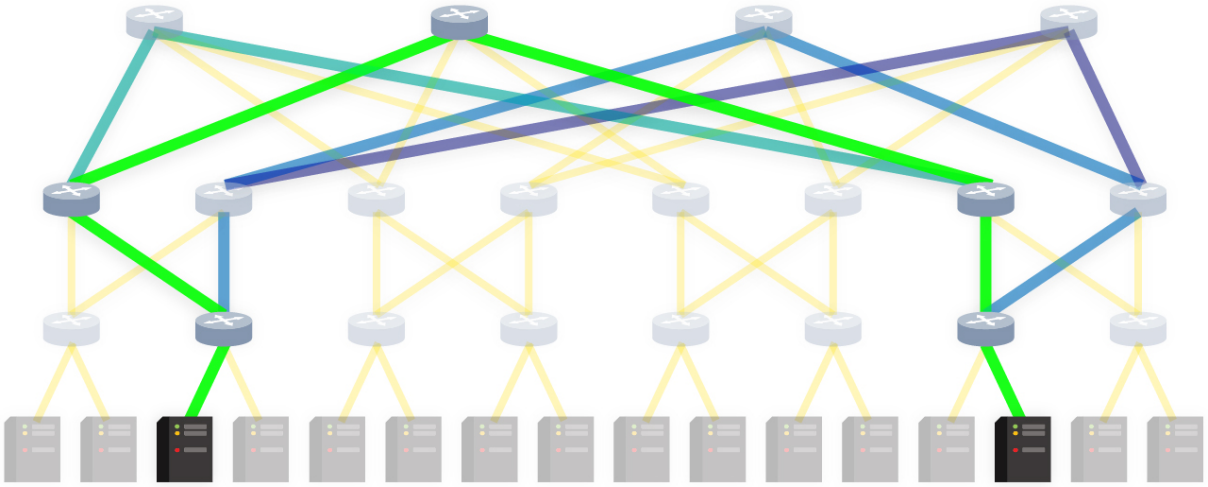


Figure 4.3: Two hosts communicating in a *silent network* setup. Note that there are no other network activity, and that there are four paths between the two hosts (indicated by the green ang bluish lines).

HTTP server in Python, and a tool to transfer data called *curl*.

In both network conditions, multiple server-client pairs were used to obtain a thorough assessment of the different links in the network. Analysis was done through inspection of the packet capture output of Mininet. Each test is also repeated to obtain a sufficient number of data points.

4.3.2 Test Parameters

The testbed can have varying permutations of network and host configurations. To aid in this, the test can be run using different testing parameters namely, host reorder resiliency, switch behavior, transport protocol, and payload sizes.

Datacenters cater to requests of varying sizes. To have an approximation of these requests and how they affect the performance of the changes to the network stack, we included the option to test with different payload sizes. Three flow types are to be used and tested for these experiments: `query`, `short`, and `long` flows [6].

Note that for the silent network, a query flow is defined with a flow size of 128KB as the tool used did not allow for the payload size to go below 128KB.

As a baseline comparison for the improvements of MPTCP, a TCP option was included in the testing environment. These results served as a form of control for the performance of MPTCP as a protocol. In addition, the performance of the changes in the network stack with TCP as its

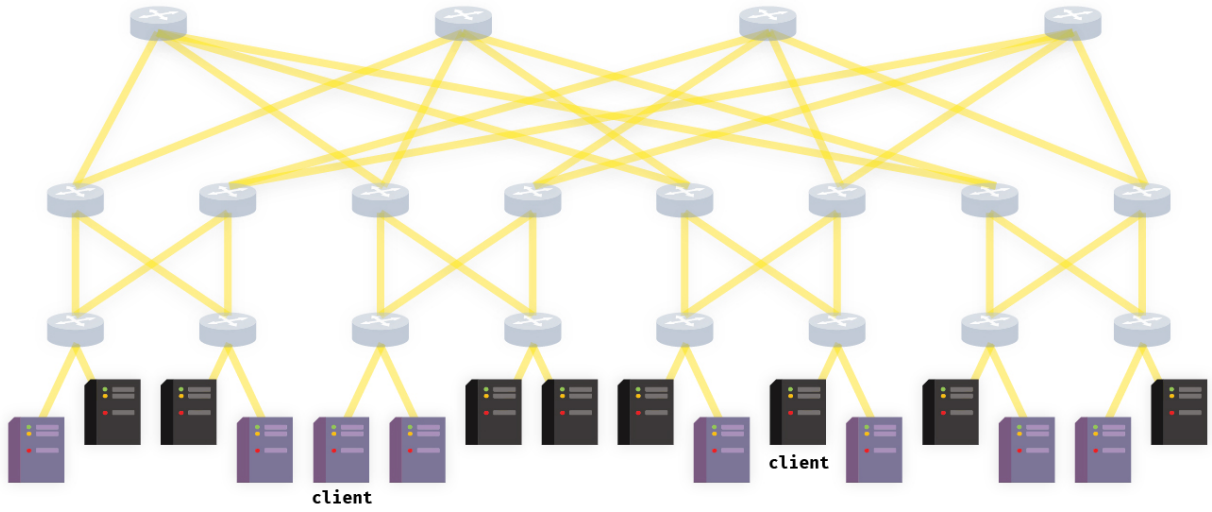


Figure 4.4: Two host groups in a *noisy network* setup, colored in black and purple. Note that one host in each group is designated as the client, to communicate with the rest of the group.

Flow	Examples	Silent Network	Noisy Network
query	Quick, bursty connections	128 kiB	10 kiB
short	Web page requests	500 kiB	500 kiB
long	Streaming, VoIP, file hosting and transfer	25 MiB	25 MiB

Table 4.5: Flow definitions, examples, and their corresponding sizes on both simulated network traffic conditions.

transport protocol was also measured and analyzed.

4.3.3 Performance Metrics

To review, throughput, goodput, flow completion time, and mean network utilization between end-hosts were measured to assess the effects of reorder resiliency in end-hosts and packet spraying in routers.

4.3.3.1 Throughput and Goodput

We define throughput to be the rate of the bytes transferred between two hosts in the network (i.e., total bytes transferred / flow completion time). In contrast, goodput is the rate of the actual data (only the size of the payload) transferred over a period of time (i.e., payload size in bytes / flow completion time).

A comparison between goodput and throughput can be used to indicate how much overhead affects the rate of transmission of actual data. If throughput is higher than goodput, this can be an indication to the presence of added MPTCP headers or packet retransmissions. This translates to no perceptible improvement from the perspective of the hosts.

4.3.3.2 Flow Completion Time

We define flow completion time to be the amount of time that it takes for a request between two hosts to complete. This includes the first request packet sent from the client to the server and the last `FIN` packet sent from server to the client.

An increase in flow completion time can be attributed to the overhead of connection establishment (for MPTCP), packet retransmissions, and/or `ACK` timeouts.

4.3.3.3 Mean Network Utilization

For this paper, we consider mean network utilization to be the ratio of used links between two hosts. For a fat tree network topology with $K = 4$, there are at most 4 paths that connect two hosts. Ideally, mean network utilization is maximized if all of paths have the near-equal (if not equal) minimum amount of bytes transferred in them (i.e., all paths have a balanced load with regards to the flow).

			Test Clusters					
			Silent Network (1-1)			Noisy Network (1-many)		
Independent Variables	ecmp-mptcp	vanilla	query	short	long	query	short	long
		juggler	query	short	long	query	short	long
	ps-mptcp	vanilla	query	short	long	query	short	long
		juggler	query	short	long	query	short	long

Table 4.6: Test clusters and independent variables used in this experiment.

4.3.4 Statistical Tests

All in all, the test clusters and independent variables can be summarized with the following table.

There are two independent variables: the switch behavior combined with the underlying transport protocol (`ecmp-mptcp`, `ps-mptcp`), as well as the inclusion/non-inclusion of Juggler (`juggler`, `vanilla` respectively). There are six test clusters, grouped by network traffic conditions (*silent network*, *noisy network*), permuted with the three payload sizes (`query`, `short`, `long`).

There are four dependent variables pertaining to the performance metrics discussed previously: throughput, goodput, flow completion time, and mean network utilization.

Each test cluster was subjected to a two-way MANOVA to verify the significance of the differences, and ranked via comparison of means. In addition, a comparison of percent differences of goodput and throughput was executed to inspect the effects of hotspots in the network.

Another experiment was also executed that includes TCP alongside MPTCP (the addition of `static-tcp` and `ps-tcp`), to serve as control. For the majority of the next chapter, we will be discussing the experiment concerning MPTCP tests only.

Chapter 5

Results and Analysis

A proper multivariate analysis of variance (MANOVA) shall be executed by meeting the following assumptions [47]: dependent variables should be continuous, independent variables should consist of two or more categorical groups, there should be independence between observations, and there should be a sufficient number of data points. These four assumptions were met by virtue of the experimental setup.

Other assumptions for MANOVA, if unmet, reduce the validity of the results. Some tests show that our data set does not meet some of these assumptions, which means that the following results should be interpreted with reservations.

5.1 Tests of Between-Subjects Effects and Best Performing Independent Variables

We chose a confidence interval of 95% ($\alpha = 0.05$) to determine the significance of the various test groups. For this chapter, we will be focusing solely on the tests of between-subjects effects. Table 5.1 shows the values for each test group. We then listed the independent variables (Table 5.2) with the highest mean for outcomes that have a significant result.

For a given link in the network, bandwidth is limited to 20 Mbps. Tests within a silent network, specifically for throughput, goodput, and flow completion time are inconclusive as they can maximize the bandwidth regardless of switch behavior and transport protocol. The throughput and goodput for a given flow naturally reach the bandwidth cap in a silent network as there are no other flows competing for resources.

Test Clusters		Independent Variables	Dependent Variables			
Network Traffic Conditions	Flow Type		Goodput	Throughput	Flow Completion Time	Mean Network Utilization
Silent Network (1-1)	query	Switch Behavior + Transport Protocol	0.640	0.888	0.310	**0.000**
		Inclusion of Juggler	**0.000**	**0.000**	**0.000**	0.675
		Combined Interaction	**0.010**	0.110	**0.009**	0.230
	short	Switch Behavior + Transport Protocol	**0.007**	0.571	0.057	**0.000**
		Inclusion of Juggler	**0.000**	**0.040**	**0.000**	0.133
		Combined Interaction	0.259	0.623	0.748	0.600
	long	Switch Behavior + Transport Protocol	**0.015**	0.895	0.479	**0.000**
		Inclusion of Juggler	**0.000**	0.238	0.171	**0.001**
		Combined Interaction	**0.000**	**0.000**	0.541	**0.001**
Noisy Network (1-many)	query	Switch Behavior + Transport Protocol	**0.023**	**0.008**	0.127	**0.000**
		Inclusion of Juggler	**0.019**	0.139	0.821	0.805
		Combined Interaction	0.061	0.249	0.063	0.843
	short	Switch Behavior + Transport Protocol	**0.000**	**0.000**	**0.000**	**0.000**
		Inclusion of Juggler	**0.001**	**0.000**	**0.000**	0.126
		Combined Interaction	**0.005**	**0.000**	**0.000**	0.150
	long	Switch Behavior + Transport Protocol	0.113	**0.000**	**0.008**	**0.000**
		Inclusion of Juggler	0.374	**0.014**	0.716	0.137
		Combined Interaction	0.545	0.309	0.589	0.232

Table 5.1: Significance values of each test group in the two-way MANOVA. Significant tests are marked in bold.

Test Clusters		Independent Variables	Dependent Variables			
Network Traffic Conditions	Flow Type		Goodput	Throughput	Flow Completion Time	Mean Network Utilization
Silent Network (1-1)	query	Switch Behavior + Transport Protocol				ps-mptcp
		Inclusion of Juggler	vanilla	vanilla	vanilla	
		Combined Interaction	ps-mptcp-vanilla		ps-mptcp-vanilla	
	short	Switch Behavior + Transport Protocol	ps-mptcp			ps-mptcp
		Inclusion of Juggler	vanilla	juggler	vanilla	
		Combined Interaction				
	long	Switch Behavior + Transport Protocol	ps-mptcp			ps-mptcp
		Inclusion of Juggler	juggler			juggler
		Combined Interaction	ecmp-mptcp-juggler	ps-mptcp-juggler		ps-mptcp
Noisy Network (1-many)	query	Switch Behavior + Transport Protocol	ps-mptcp	ps-mptcp		ps-mptcp
		Inclusion of Juggler	juggler			
		Combined Interaction				
	short	Switch Behavior + Transport Protocol	ps-mptcp	ps-mptcp	ps-mptcp	ps-mptcp
		Inclusion of Juggler	vanilla	juggler	vanilla	
		Combined Interaction	ps-mptcp-vanilla	ps-mptcp-juggler	ps-mptcp-vanilla	
	long	Switch Behavior + Transport Protocol		ps-mptcp	ps-mptcp	ps-mptcp
		Inclusion of Juggler	vanilla		vanilla	
		Combined Interaction				

Table 5.2: Best performing configurations, with non-significant results filtered out.

Network Traffic Conditions	Flow Type	Switch Behavior and Transport Protocol			
		Statically-configured switches (TCP)	Packet Spraying-based switches (TCP)	ECMP-based switches (MPTCP)	Packet Spraying-based switches (MPTCP)
Silent Network (1-1)	query	8.99%	10.54%	12.96%	13.39%
	short	9.68%	11.41%	18.62%	17.50%
	long	31.69%	30.18%	35.44%	34.47%
Noisy Network (1-many)	query	16.29%	23.61%	31.25%	31.65%
	short	32.05%	43.71%	47.69%	47.71%
	long	51.16%	54.37%	57.80%	58.10%

Table 5.3: Percentage of traffic in excess of "good" traffic (i.e., actual payload size).

5.2 Gap Between Goodput and Throughput as an Indicator of Network Hotspots

Network hotspots, defined as switches that have links that are near to exceed their capacity, result in an increase of dropped packets and increased packet retransmissions. This results in an increase in flow completion time, as well as the total amount of bytes transferred. Therefore, a significant gap between throughput and goodput may be noticed if network hotspots are in the network. In this experiment, we took the observed means of goodput and throughput for statically-configured switches (with hosts running TCP), ECMP-based switches (with hosts running MPTCP), and PS-based switches (with hosts running TCP and hosts running MPTCP) and calculated the percentage of traffic in excess of the actual flow size.

If we assume that the percentage of excess traffic is caused by retransmission, then we can see that statically-configured switches has significantly less retransmissions over both ECMP-based switches and PS-based switches for query and short flows. However, during long flows, the difference between all the switch behaviors seem insignificant. In all cases, ECMP-based switches and PS-based switches (with hosts running MPTCP) performed comparably equally, implying that packet spraying did not mitigate the hotspot problem.

5.3 Flow Completion Time Performed Worse with Reorder-Resilient Hosts; Inconclusive Effects on Throughput

Since Juggler, the tool to enable reorder-resiliency in the hosts, was created to increase tolerance between TCP packet arrival times, it might not perform as well with MPTCP. Based on

our results, Juggler had a negative effect on flow completion time for a silent network with query and short flows, as well as for a noisy network for short flows.

This can also explain the inconclusive results for throughput. Results are mixed, with Juggler increasing throughput for short flows (on both silent and noisy networks), but failing to increase throughput for query flows on a silent network and long flows on a noisy network.

This leads us to believe that Juggler cannot be simply placed in a network without negatively affecting its performance.

5.4 Packet Spraying Generally Performed Better than ECMP

As for the switch behavior, packet spraying increased mean network utilization for all test groups compared to ECMP. In addition, packet spraying performed better than ECMP for goodput and throughput tests in a noisy environment.

This behavior may be preferred as most datacenter networks are high in network activity.

5.5 Effects of Combining Packet Spraying-based Switches with Reorder-Resilient Hosts

A combination of packet spraying and reorder resiliency improves throughput for both silent and noisy networks. However flow completion time increases (due to the presence of Juggler as seen in the Juggler-only analysis).

Goodput results turn out to be inconclusive as tests performed better with PS-based switches, but some preferred the inclusion of Juggler while others don't. Mean network utilization remains significantly higher when compared to the rest which can be attributed to the packet spraying behavior.

Chapter 6

Conclusion and Recommendations

Based on preliminary tests, we found that MPTCP increases throughput significantly compared to TCP, especially with using multiple paths. However the connection establishment overhead needed to create multiple subflows penalizes the speed at which flows are created and consequently completed.

In addition, the paper promised to experimentally prove that ECMP-based switches causes hotspots in the network. By comparing the throughput and goodput of ECMP-based switches without reorder resiliency, we saw that there are was no improvements to the hotspots in the network. However hotspots in the network may be a consequence of the experiment setup. This could either be attributed to the amount of traffic in the noisy network or due to the limited bandwidth at the single interface of the host for the silent network.

It was also found that packet spraying improves throughput, goodput, and mean network utilization for a noisy environment. Considering that most datacenters will, more often than not, operate with a lot of concurrent flows in the network, packet spraying may be beneficial.

Packet spraying and reorder resiliency show a significant difference between groups (except for a noisy network with long and query-length flows and silent network with short flows). Looking at a noisy network with short flows, throughput is improved and flow completion time worsens.

From this, we can say that packet spraying is better in terms of throughput, goodput, and mean network utilization. Flow completion time for packet spraying turned out inconclusive data.

6.1 Recommendations and Future Work

Further analysis may be done on TCP combined with packet spraying and reorder resilient hosts. According to initial testing, TCP with reorder resilient hosts and packet spraying showed an increase in throughput and goodput. This may be promising as there would be no need to implement a new transport protocol.

Network hotspots may also be studied further by creating a special testbed specifically built to measure the amount of traffic going through specific nodes in the network. We hypothesize that packet spraying may still improve network hotspots caused by ECMP.

Data collected in an datacenter network with real traffic would be beneficial for the validity of the tests since the simulated traffic in this experiment is merely an approximation.

Bibliography

[1] Y. Geng, V. Jeyakumar, A. Kabbani, and M. Alizadeh, "Juggler: a practical reordering resilient network stack for datacenters," in *Proceedings of the Eleventh European Conference on Computer Systems*, p. 20, ACM, 2016.

[2] J. He and J. Rexford, "Toward internet-wide multipath routing," *IEEE network*, vol. 22, no. 2, 2008.

[3] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[4] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 50–61, ACM, 2011.

[5] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *ACM SIGOPS operating systems review*, vol. 41, no. 6, pp. 205–220, 2007.

[6] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *ACM SIGCOMM computer communication review*, vol. 40, pp. 63–74, ACM, 2010.

[7] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[8] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 267–280, ACM, 2010.

[9] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks.," in *NSDI*, vol. 10, pp. 19–19, 2010.

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

[11] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 39–50, ACM, 2009.

[12] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE transactions on Computers*, vol. 100, no. 10, pp. 892–901, 1985.

[13] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 75–86, ACM, 2008.

[14] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[15] B. Lebiednik, A. Mangal, and N. Tiwari, "A survey and evaluation of data center network topologies," *arXiv preprint arXiv:1605.01701*, 2016.

[16] S. Rost and H. Balakrishnan, "Rate-aware splitting of aggregate traffic," tech. rep., Technical report, MIT, 2003.

[17] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," in *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 266–277, ACM, 2011.

[18] J. Postel *et al.*, "Transmission control protocol rfc 793," 1981.

[19] J. F. Kurose, *Computer networking: A top-down approach featuring the internet, 3/E*. Pearson Education India, 2005.

[20] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath tcp development," tech. rep., 2011.

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

[22] A. Ford, "Multipath tcp architecture: Towards consensus."

[23] W. R. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," 1997.

[24] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *ACM SIGCOMM computer communication review*, vol. 39, pp. 51–62, ACM, 2009.

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

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

[27] C. E. Hopps and D. Thaler, "Multipath issues in unicast and multicast next-hop selection," 2000.

[28] M. Fukuyama, N. Yamai, and N. Kitagawa, "Throughput improvement of mptcp communication by bicasting on multihomed network," in *Student Project Conference (ICT-ISPC), 2016 Fifth ICT International*, pp. 9–12, IEEE, 2016.

[29] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *INFOCOM, 2013 Proceedings IEEE*, pp. 2130–2138, IEEE, 2013.

[30] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient datacenter load balancing in the wild," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 253–266, ACM, 2017.

[31] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[32] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[33] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.

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

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

[36] S. Ghorbani, Z. Yang, P. Godfrey, Y. Ganjali, and A. Firoozshahian, "Drill: Micro load balancing for low-latency data center networks," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 225–238, ACM, 2017.

[37] M. Zhang, B. Karp, S. Floyd, and L. Peterson, "Rr-tcp: a reordering-robust tcp with dsack," in *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on*, pp. 95–106, IEEE, 2003.

[38] C.-H. Ke, "Multipath tcp test." Website, Dec. 2017.

[39] U. catholique de Louvain, "Multipath tcp - linux kernel implementation : Users - configure mptcp browse." Website, Aug. 2017.

[40] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, "Rfc 5340, ospf for ipv6," *IETF. July*, vol. 24, 2008.

[41] M. Kheirkhah, "Multipath tcp (mptcp) implementation in ns-3." Github, May 2014.

[42] U. catholique de Louvain, "Linux kernel implementation of multipath (mptcp)." Github, Nov. 2017.

[43] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[44] p4lang, "behavioral-model: Rewrite of the behavioral model as a C++ project without auto-generated code (except for the PD interface)," Mar. 2018. original-date: 2015-01-26T21:43:23Z.

[45] p4lang, "p4c-bm: Generates the JSON configuration for the behavioral-model (bmv2), as well as the C/C++ PD code," Mar. 2018. original-date: 2015-08-07T15:48:22Z.

[46] p4lang, "p4lang/p4-spec," Feb. 2018. original-date: 2015-08-14T08:45:53Z.

[47] "How to perform a two-way MANOVA in SPSS Statistics | Laerd Statistics."

Appendix A

Code Snippets

These code snippets come from the researchers' git repository available at `https://github.com/MMfSDT/`

A.1 Topology Generator

These code snippets come from the researchers' git repository available at `https://github.com/MMfSDT/mininet-topo-generator`

A.2 Network Tests

These code snippets come from the researchers' git repository available at `https://github.com/MMfSDT/network-tests`

A.3 MPTCP-Juggler Modified Kernel

These code snippets come from the researchers' git repository available at `https://github.com/MMfSDT/mptcp-juggler`

Appendix B

Experimental Results

B.1 Raw Results

The test sets were defined at 4.6. All results (including those that included TCP end-hosts), SPSS outputs, and SPSS-ready `.csv` files is available at `https://github.com/MMfSDT/thesis-results`.

B.2 Statistical Data

We took the raw results and applied a two-way MANOVA in SPSS, including post-hoc analysis where applicable.

B.2.1 ECMP-MPTCP vs. PS-MPTCP

B.2.1.1 Silent Network (1-1): Query Flows

Between-Subjects Factors

		Value Label	N
Router Type | Protocol	1	ecmp-mptcp	160
	2	ps-mptcp	160
Juggler Enabled	1	false	160
	2	true	160

Descriptive Statistics

	Router Type | Protocol	Juggler Enabled	Mean	Std. Deviation	N
Goodput	ecmp-mptcp	false	1833533.179112430000	99210.6519444564500	80
		true	1751989.144443832000	115824.6920164541400	80
		Total	1792761.161778131300	115016.2167963604000	160
	ps-mptcp	false	1863812.663847402000	89767.9853446404300	80
		true	1708404.238500124000	182712.6659171651000	80
		Total	1786108.451173763300	163299.4258662041000	160
	Total	false	1848672.921479916200	95524.2540687409200	160
		true	1730196.691471977400	154046.6458976601000	160
		Total	1789434.806475947000	141054.1867727976400	320
Throughput	ecmp-mptcp	false	2079808.270281898600	103632.0127256187300	80
		true	2039500.380635019400	270268.5682264126400	80
		Total	2059654.325458458600	205030.7506971138000	160
	ps-mptcp	false	2113147.859705757300	95442.9697405130900	80
		true	2011535.360597368600	155473.9054193348600	80
		Total	2062341.610151563000	138324.2513287808400	160
	Total	false	2096478.064993828300	100705.9527179419500	160
		true	2025517.870616194300	220226.3443787552000	160
		Total	2060997.967805010700	174618.2551098666700	320
Flow Completion Time	ecmp-mptcp	false	.071708	.0041731	80
		true	.075141	.0050568	80
		Total	.073424	.0049319	160
	ps-mptcp	false	.070491	.0035242	80
		true	.077876	.0110814	80
		Total	.074184	.0089947	160
	Total	false	.071099	.0038982	160
		true	.076509	.0086948	160
		Total	.073804	.0072522	320
Mean Network Utilization	ecmp-mptcp	false	.638272603042754	.170691420939118	80
		true	.626292446115412	.182404149943830	80
		Total	.632282524579083	.176191022323148	160
	ps-mptcp	false	.843198173309977	.089877819775924	80
		true	.867970186058475	.063791373105074	80
		Total	.855584179684226	.078675661108174	160
	Total	false	.740735388176366	.170453569618141	160
		true	.747131316086943	.182336920194718	160
		Total	.743933352131654	.176247536788951	320

Box's Test of Equality of Covariance Matrices[a]

Box's M	1527.674
F	49.762
df1	30
df2	274544.420
Sig.	.000

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	1.000	1040211.359[b]	4.000	313.000	.000	1.000
	Wilks' Lambda	.000	1040211.359[b]	4.000	313.000	.000	1.000
	Hotelling's Trace	13293.436	1040211.359[b]	4.000	313.000	.000	1.000
	Roy's Largest Root	13293.436	1040211.359[b]	4.000	313.000	.000	1.000
router_proto	Pillai's Trace	.441	61.778[b]	4.000	313.000	.000	.441
	Wilks' Lambda	.559	61.778[b]	4.000	313.000	.000	.441
	Hotelling's Trace	.790	61.778[b]	4.000	313.000	.000	.441
	Roy's Largest Root	.790	61.778[b]	4.000	313.000	.000	.441
juggler_enabled	Pillai's Trace	.253	26.451[b]	4.000	313.000	.000	.253
	Wilks' Lambda	.747	26.451[b]	4.000	313.000	.000	.253
	Hotelling's Trace	.338	26.451[b]	4.000	313.000	.000	.253
	Roy's Largest Root	.338	26.451[b]	4.000	313.000	.000	.253
router_proto * juggler_enabled	Pillai's Trace	.034	2.789[b]	4.000	313.000	.027	.034
	Wilks' Lambda	.966	2.789[b]	4.000	313.000	.027	.034
	Hotelling's Trace	.036	2.789[b]	4.000	313.000	.027	.034
	Roy's Largest Root	.036	2.789[b]	4.000	313.000	.027	.034

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

b. Exact statistic

Levene's Test of Equality of Error Variances[a]

		Levene Statistic	df1	df2	Sig.
Goodput	Based on Mean	9.239	3	316	.000
	Based on Median	2.750	3	316	.043
	Based on Median and with adjusted df	2.750	3	191.913	.044
	Based on trimmed mean	5.680	3	316	.001
Throughput	Based on Mean	3.547	3	316	.015
	Based on Median	2.810	3	316	.040
	Based on Median and with adjusted df	2.810	3	158.824	.041
	Based on trimmed mean	2.741	3	316	.043
Flow Completion Time	Based on Mean	16.940	3	316	.000
	Based on Median	4.782	3	316	.003
	Based on Median and with adjusted df	4.782	3	130.635	.003
	Based on trimmed mean	9.623	3	316	.000
Mean Network Utilization	Based on Mean	22.752	3	316	.000
	Based on Median	21.281	3	316	.000
	Based on Median and with adjusted df	21.281	3	206.893	.000
	Based on trimmed mean	22.263	3	316	.000

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Tests of Between-Subjects Effects

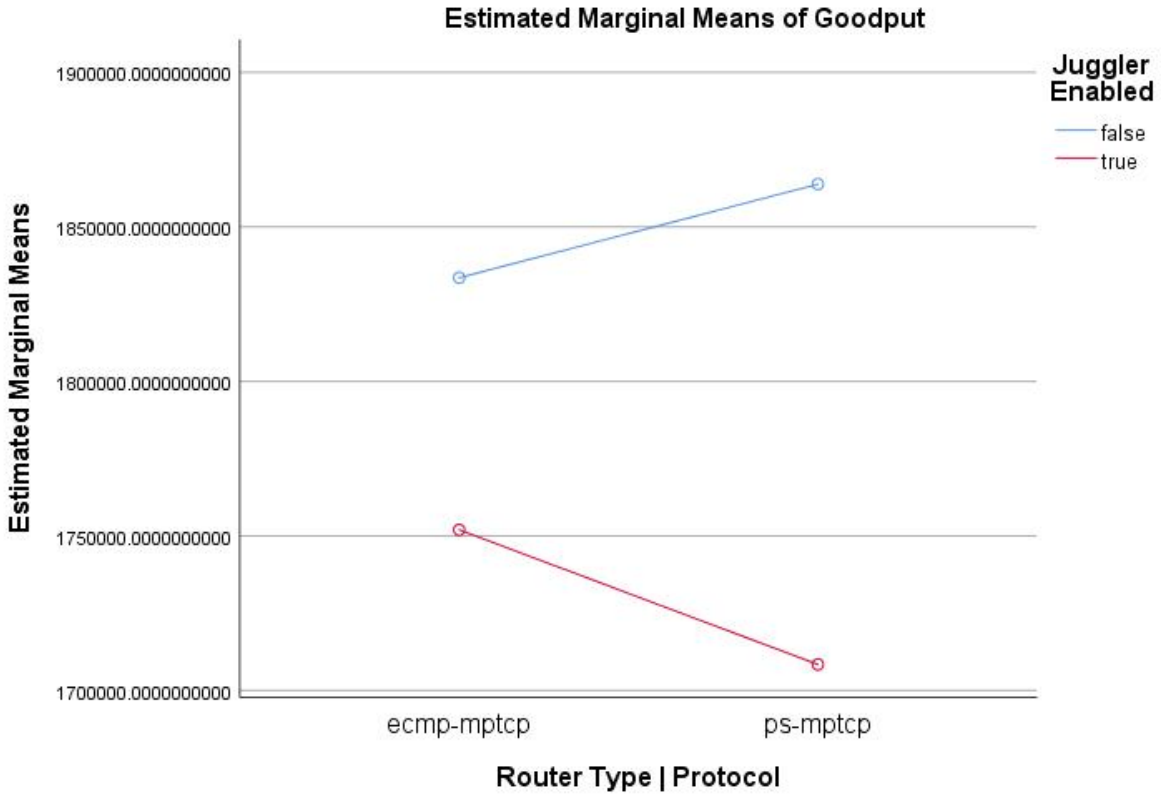
Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Goodput	1235589015029.002[a]	3	411863005009.667	25.463	.000	.195
	Throughput	478570757635.229[b]	3	159523585878.410	5.451	.001	.049
	Flow Completion Time	.003[c]	3	.001	20.196	.000	.161
	Mean Network Utilization	4.019[d]	3	1.340	71.883	.000	.406
Intercept	Goodput	1024664616520836.200	1	1024664616520836.200	63348.347	.000	.995
	Throughput	1359268039454845.000	1	1359268039454845.000	46444.428	.000	.993
	Flow Completion Time	1.743	1	1.743	39124.878	.000	.992
	Mean Network Utilization	177.100	1	177.100	9501.801	.000	.968
router_proto	Goodput	3540684670.834	1	3540684670.834	.219	.640	.001
	Throughput	577719921.744	1	577719921.744	.020	.888	.000
	Flow Completion Time	4.613E-5	1	4.613E-5	1.035	.310	.003
	Mean Network Utilization	3.989	1	3.989	214.024	.000	.404
juggler_enabled	Goodput	1122929366151.441	1	1122929366151.441	69.423	.000	.180
	Throughput	402827934888.893	1	402827934888.893	13.764	.000	.042
	Flow Completion Time	.002	1	.002	52.544	.000	.143
	Mean Network Utilization	.003	1	.003	.176	.675	.001
router_proto * juggler_enabled	Goodput	109118964206.618	1	109118964206.618	6.746	.010	.021
	Throughput	75165102824.562	1	75165102824.562	2.568	.110	.008
	Flow Completion Time	.000	1	.000	7.009	.009	.022
	Mean Network Utilization	.027	1	.027	1.449	.230	.005
Error	Goodput	5111325455328.149	316	16175080554.836			
	Throughput	9248228912983.785	316	29266547192.987			
	Flow Completion Time	.014	316	4.455E-5			
	Mean Network Utilization	5.890	316	.019			
Total	Goodput	1031011530991192.900	320				
	Throughput	1368994839125461.800	320				
	Flow Completion Time	1.760	320				
	Mean Network Utilization	187.009	320				
Corrected Total	Goodput	6346914470357.151	319				
	Throughput	9726799670619.014	319				
	Flow Completion Time	.017	319				
	Mean Network Utilization	9.909	319				

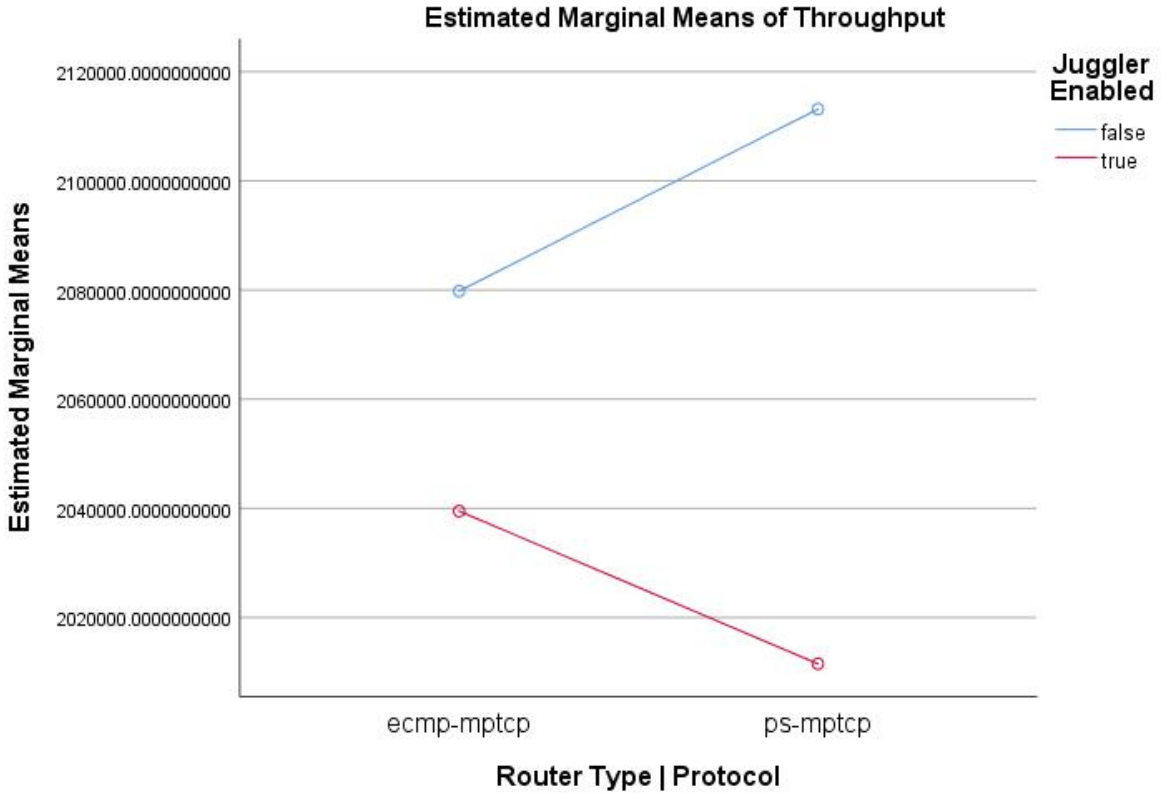
a. R Squared = .195 (Adjusted R Squared = .187)

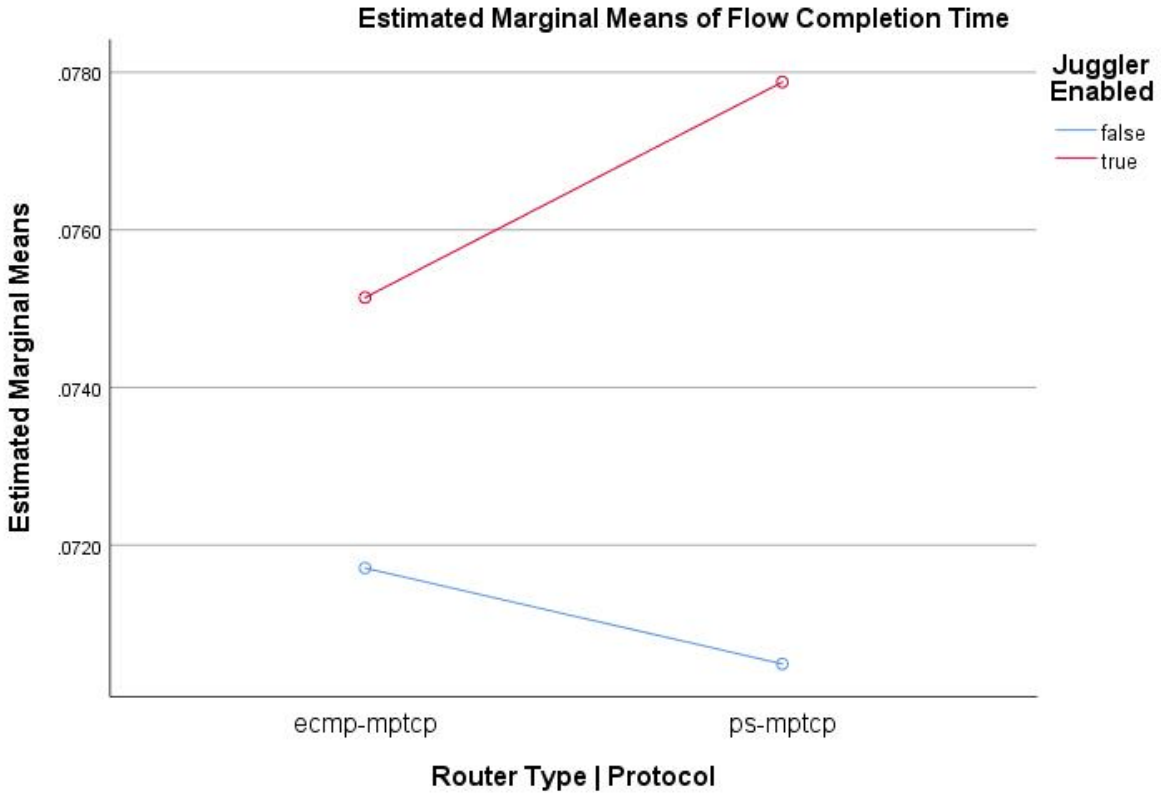
b. R Squared = .049 (Adjusted R Squared = .040)

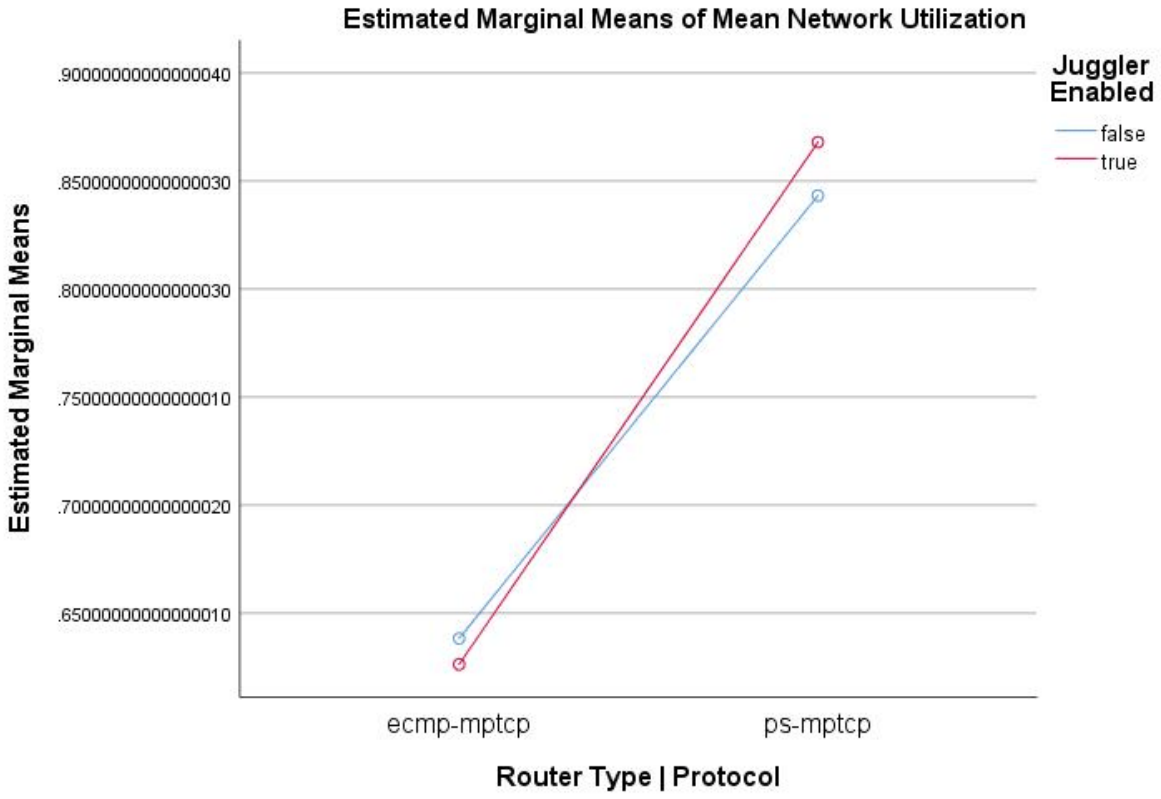
c. R Squared = .161 (Adjusted R Squared = .153)

d. R Squared = .406 (Adjusted R Squared = .400)


Estimated Marginal Means of Goodput
Estimated Marginal Means
1900000.0000000000
1850000.0000000000
1800000.0000000000
1750000.0000000000
1700000.0000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Throughput
Estimated Marginal Means
2120000.0000000000
2100000.0000000000
2080000.0000000000
2060000.0000000000
2040000.0000000000
2020000.0000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Flow Completion Time
Estimated Marginal Means
.0780
.0760
.0740
.0720
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Mean Network Utilization
Estimated Marginal Means
.90000000000000040
.85000000000000030
.80000000000000030
.75000000000000010
.70000000000000020
.65000000000000010
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true

B.2.1.2 Silent Network (1-1): Short Flows

Between-Subjects Factors

		Value Label	N
Router Type | Protocol	1	ecmp-mptcp	160
	2	ps-mptcp	160
Juggler Enabled	1	false	160
	2	true	160

Descriptive Statistics

	Router Type | Protocol	Juggler Enabled	Mean	Std. Deviation	N
Goodput	ecmp-mptcp	false	2025190.410865331300	89072.3742739332100	80
		true	1940790.770380375700	198167.8901936360300	80
		Total	1982990.590622854000	158889.2184974950200	160
	ps-mptcp	false	2090586.413623892700	102010.4433730903000	80
		true	1967948.507911314300	183618.6978513579200	80
		Total	2029267.460767603500	160330.5708630291300	160
	Total	false	2057888.412244611200	100936.7116592786300	160
		true	1954369.639145846200	190916.7305358554600	160
		Total	2006129.025695229000	161037.3925541599700	320
Throughput	ecmp-mptcp	false	2404914.233449672000	322073.5555848654500	80
		true	2468607.477738653000	443296.8241896250000	80
		Total	2436760.855594161400	387554.0594360570500	160
	ps-mptcp	false	2408021.823504277000	141763.3309359557800	80
		true	2511661.629609386000	456341.2580908186500	80
		Total	2459841.726556832000	340816.9897260628000	160
	Total	false	2406468.028476972600	248046.6511691877400	160
		true	2490134.553674020300	448969.0527032917000	160
		Total	2448301.291075494600	364545.3340071155000	320
Flow Completion Time	ecmp-mptcp	false	.253312	.0114504	80
		true	.268528	.0507983	80
		Total	.260920	.0374900	160
	ps-mptcp	false	.245550	.0133578	80
		true	.263003	.0312521	80
		Total	.254276	.0255060	160
	Total	false	.249431	.0129983	160
		true	.265765	.0421316	160
		Total	.257598	.0321850	320
Mean Network Utilization	ecmp-mptcp	false	.688533495347942	.146042722124866	80
		true	.713339011624170	.150439500313646	80
		Total	.700936253486056	.148313233479325	160
	ps-mptcp	false	.919606435286936	.044290137702911	80
		true	.931582043755834	.041829911178551	80
		Total	.925594239521385	.043359972811855	160
	Total	false	.804069965317439	.158127913522651	160
		true	.822460527690002	.155230845264898	160
		Total	.813265246503720	.156711146809555	320

Box's Test of Equality of Covariance Matrices[a]

Box's M	1436.832
F	46.803
df1	30
df2	274544.420
Sig.	.000

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	1.000	251177.629[b]	4.000	313.000	.000	1.000
	Wilks' Lambda	.000	251177.629[b]	4.000	313.000	.000	1.000
	Hotelling's Trace	3209.938	251177.629[b]	4.000	313.000	.000	1.000
	Roy's Largest Root	3209.938	251177.629[b]	4.000	313.000	.000	1.000
router_proto	Pillai's Trace	.518	84.104[b]	4.000	313.000	.000	.518
	Wilks' Lambda	.482	84.104[b]	4.000	313.000	.000	.518
	Hotelling's Trace	1.075	84.104[b]	4.000	313.000	.000	.518
	Roy's Largest Root	1.075	84.104[b]	4.000	313.000	.000	.518
juggler_enabled	Pillai's Trace	.145	13.288[b]	4.000	313.000	.000	.145
	Wilks' Lambda	.855	13.288[b]	4.000	313.000	.000	.145
	Hotelling's Trace	.170	13.288[b]	4.000	313.000	.000	.145
	Roy's Largest Root	.170	13.288[b]	4.000	313.000	.000	.145
router_proto * juggler_enabled	Pillai's Trace	.015	1.173[b]	4.000	313.000	.323	.015
	Wilks' Lambda	.985	1.173[b]	4.000	313.000	.323	.015
	Hotelling's Trace	.015	1.173[b]	4.000	313.000	.323	.015
	Roy's Largest Root	.015	1.173[b]	4.000	313.000	.323	.015

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

b. Exact statistic

Levene's Test of Equality of Error Variances[a]

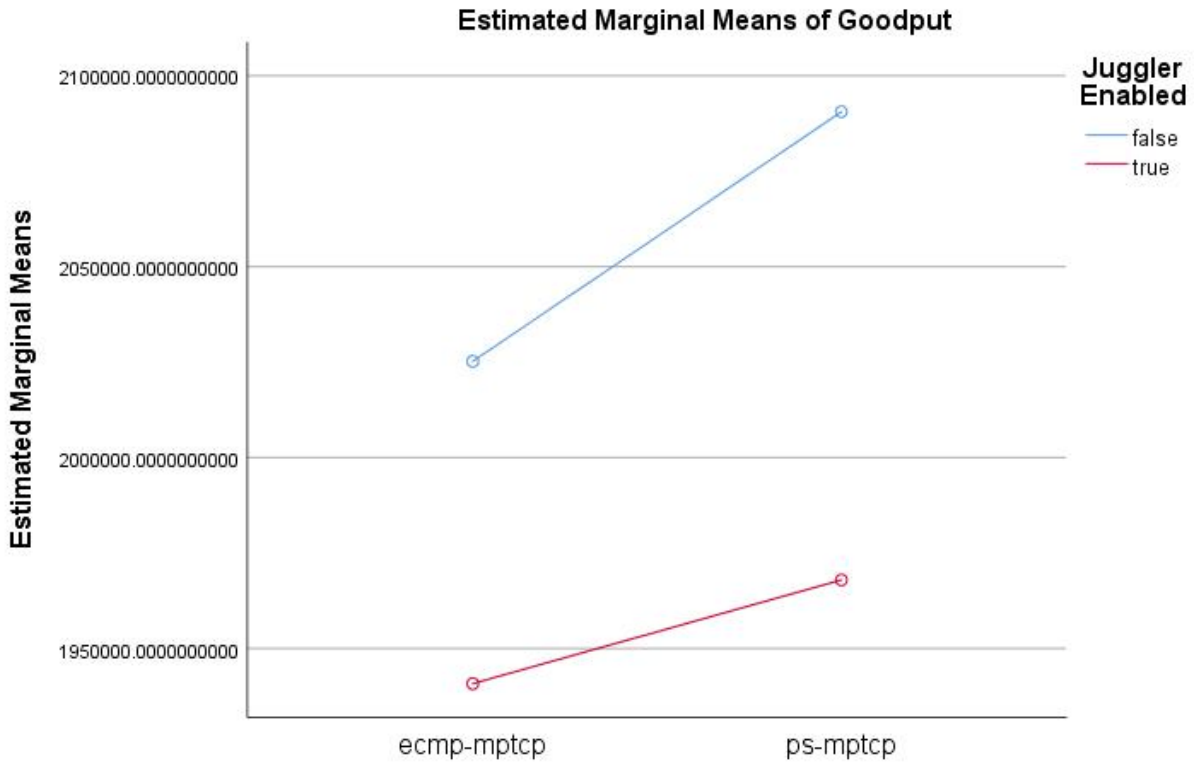
		Levene Statistic	df1	df2	Sig.
Goodput	Based on Mean	6.122	3	316	.000
	Based on Median	6.135	3	316	.000
	Based on Median and with adjusted df	6.135	3	223.718	.001
	Based on trimmed mean	5.947	3	316	.001
Throughput	Based on Mean	11.868	3	316	.000
	Based on Median	6.277	3	316	.000
	Based on Median and with adjusted df	6.277	3	237.787	.000
	Based on trimmed mean	8.750	3	316	.000
Flow Completion Time	Based on Mean	4.709	3	316	.003
	Based on Median	4.057	3	316	.008
	Based on Median and with adjusted df	4.057	3	144.499	.008
	Based on trimmed mean	3.977	3	316	.008
Mean Network Utilization	Based on Mean	37.524	3	316	.000
	Based on Median	37.522	3	316	.000
	Based on Median and with adjusted df	37.522	3	184.554	.000
	Based on trimmed mean	37.519	3	316	.000

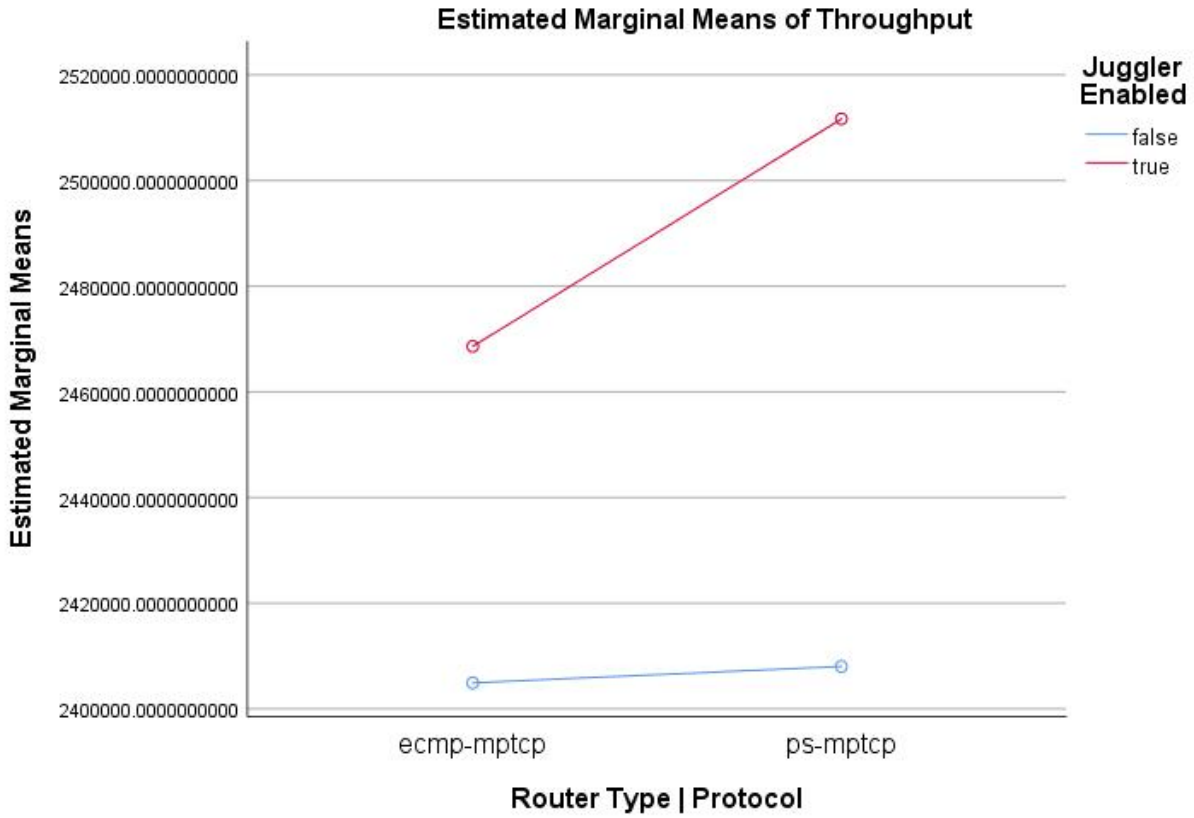
Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

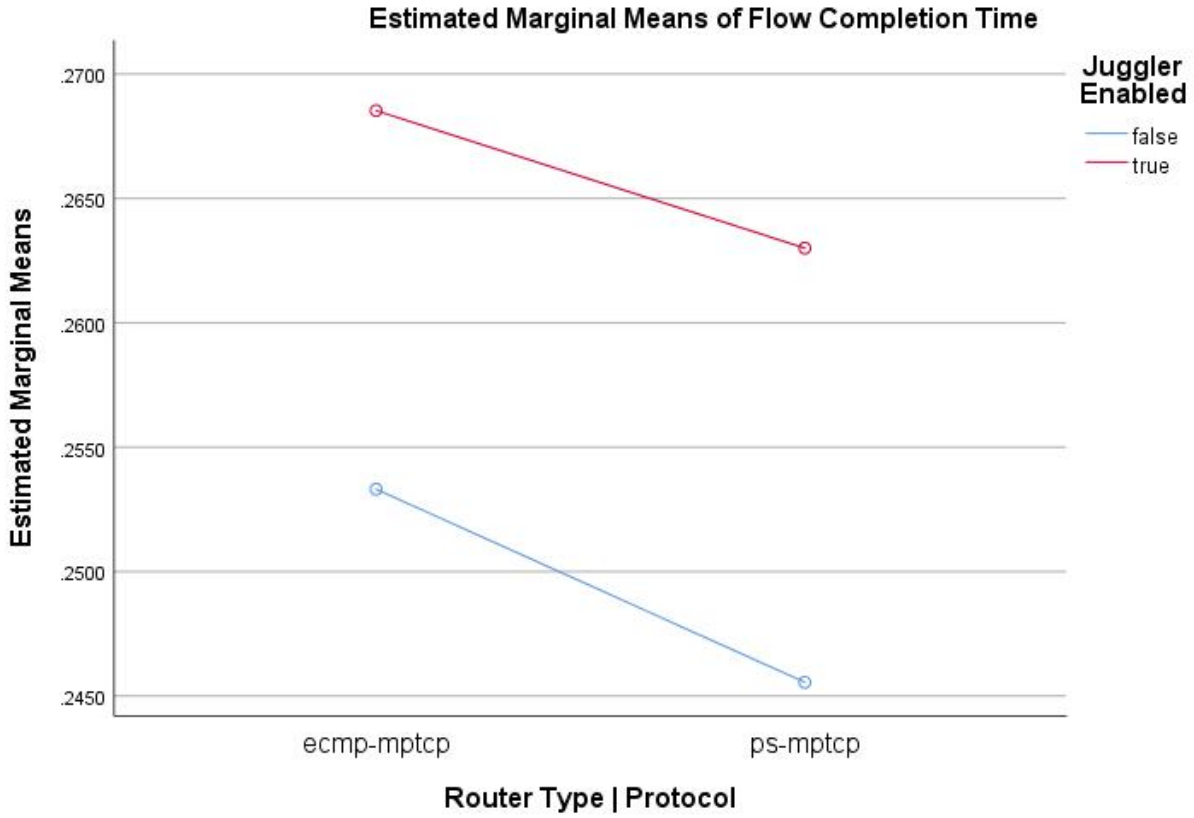
a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

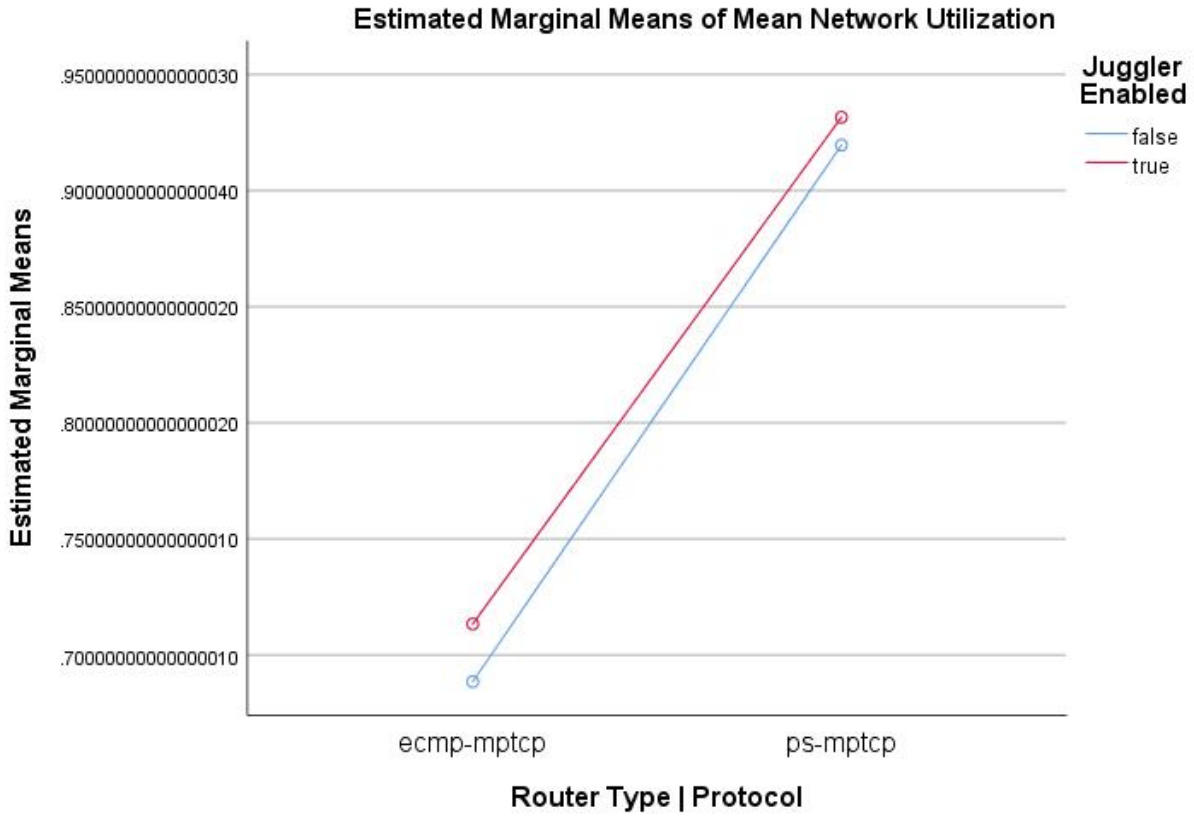
Tests of Between-Subjects Effects

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Goodput	1057858106093.778[a]	3	352619368697.926	15.444	.000	.128
	Throughput	634539679454.016[b]	3	211513226484.672	1.601	.189	.015
	Flow Completion Time	.025[c]	3	.008	8.612	.000	.076
	Mean Network Utilization	4.068[d]	3	1.356	113.779	.000	.519
Intercept	Goodput	1287857173675805.000	1	1287857173675805.000	56406.812	.000	.994
	Throughput	1918137347802223.500	1	1918137347802223.500	14515.189	.000	.979
	Flow Completion Time	21.234	1	21.234	21966.167	.000	.986
	Mean Network Utilization	211.648	1	211.648	17758.740	.000	.983
router_proto	Goodput	171323896831.542	1	171323896831.542	7.504	.007	.023
	Throughput	42618128351.622	1	42618128351.622	.323	.571	.001
	Flow Completion Time	.004	1	.004	3.653	.057	.011
	Mean Network Utilization	4.038	1	4.038	338.791	.000	.517
juggler_enabled	Goodput	857290910709.880	1	857290910709.880	37.548	.000	.106
	Throughput	560006995083.858	1	560006995083.858	4.238	.040	.013
	Flow Completion Time	.021	1	.021	22.079	.000	.065
	Mean Network Utilization	.027	1	.027	2.270	.133	.007
router_proto * juggler_enabled	Goodput	29243298552.345	1	29243298552.345	1.281	.259	.004
	Throughput	31914556018.585	1	31914556018.585	.242	.623	.001
	Flow Completion Time	.000	1	.000	.104	.748	.000
	Mean Network Utilization	.003	1	.003	.276	.600	.001
Error	Goodput	7214782228311.225	316	22831589330.099			
	Throughput	41758423194834.620	316	132146908844.413			
	Flow Completion Time	.305	316	.001			
	Mean Network Utilization	3.766	316	.012			
Total	Goodput	1296129814010209.500	320				
	Throughput	1960530310676510.800	320				
	Flow Completion Time	21.565	320				
	Mean Network Utilization	219.482	320				
Corrected Total	Goodput	8272640334405.003	319				
	Throughput	42392962874288.630	319				
	Flow Completion Time	.330	319				
	Mean Network Utilization	7.834	319				


Estimated Marginal Means of Goodput
Estimated Marginal Means
2100000.0000000000
2050000.0000000000
2000000.0000000000
1950000.0000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Throughput
Estimated Marginal Means
2520000.0000000000
2500000.0000000000
2480000.0000000000
2460000.0000000000
2440000.0000000000
2420000.0000000000
2400000.0000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Flow Completion Time
Estimated Marginal Means
.2700
.2650
.2600
.2550
.2500
.2450
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Mean Network Utilization
Estimated Marginal Means
.95000000000000030
.90000000000000040
.85000000000000020
.80000000000000020
.75000000000000010
.70000000000000010
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true

B.2.1.3 Silent Network (1-1): Long Flows

Between-Subjects Factors

		Value Label	N
Router Type | Protocol	1	ecmp-mptcp	160
	2	ps-mptcp	160
Juggler Enabled	1	false	160
	2	true	160

Descriptive Statistics

	Router Type | Protocol	Juggler Enabled	Mean	Std. Deviation	N
Goodput	ecmp-mptcp	false	2163610.9436017573000	187810.21901629394000	80
		true	2322324.3511051247000	45669.04506876159000	80
		Total	2242967.6473534430000	157793.48601480536000	160
	ps-mptcp	false	2274885.0683101940000	251914.09787437078000	80
		true	2299065.1028885120000	58734.82016634039600	80
		Total	2286975.0855993545000	182734.57081030402000	160
	Total	false	2219248.0059559770000	228410.07930395854000	160
		true	2310694.7269968190000	53725.33961271857000	160
		Total	2264971.3664763966000	171871.08675737496000	320
Throughput	ecmp-mptcp	false	3677803.448847078700	987857.5594465388000	80
		true	3270438.850123586600	1035782.3273524552000	80
		Total	3474121.149485332400	1029397.2592449515000	160
	ps-mptcp	false	3144856.988682886000	1088379.7923328874000	80
		true	3834824.351743714400	1154062.3878532800000	80
		Total	3489840.670213300300	1170497.4304331432000	160
	Total	false	3411330.218764981700	1069989.4035068306000	160
		true	3552631.600933649000	1129125.3829299868000	160
		Total	3481980.909849315400	1100506.7479568190000	320
Flow Completion Time	ecmp-mptcp	false	12.214184	1.1520984	80
		true	11.292455	.2293893	80
		Total	11.753319	.9483499	160
	ps-mptcp	false	13.815303	21.6318882	80
		true	11.409761	.2998489	80
		Total	12.612532	15.2970002	160
	Total	false	13.014743	15.2905921	160
		true	11.351108	.2725399	160
		Total	12.182926	10.8289313	320
Mean Network Utilization	ecmp-mptcp	false	.768107136346327	.112764662004438	80
		true	.826072209040875	.107861317729378	80
		Total	.797089672693601	.113770227505103	160
	ps-mptcp	false	.987752794100423	.006948465752649	80
		true	.986286818261603	.006368029143416	80
		Total	.987019806181012	.006684144457696	160
	Total	false	.877929965223375	.135936918002783	160
		true	.906179513651239	.110716462538570	160
		Total	.892054739437307	.124581089890020	320

Box's Test of Equality of Covariance Matrices[a]

Box's M	3870.694
F	126.083
df1	30
df2	274544.420
Sig.	.000

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	.999	71860.750[b]	4.000	313.000	.000	.999
	Wilks' Lambda	.001	71860.750[b]	4.000	313.000	.000	.999
	Hotelling's Trace	918.348	71860.750[b]	4.000	313.000	.000	.999
	Roy's Largest Root	918.348	71860.750[b]	4.000	313.000	.000	.999
router_proto	Pillai's Trace	.624	129.960[b]	4.000	313.000	.000	.624
	Wilks' Lambda	.376	129.960[b]	4.000	313.000	.000	.624
	Hotelling's Trace	1.661	129.960[b]	4.000	313.000	.000	.624
	Roy's Largest Root	1.661	129.960[b]	4.000	313.000	.000	.624
juggler_enabled	Pillai's Trace	.178	16.926[b]	4.000	313.000	.000	.178
	Wilks' Lambda	.822	16.926[b]	4.000	313.000	.000	.178
	Hotelling's Trace	.216	16.926[b]	4.000	313.000	.000	.178
	Roy's Largest Root	.216	16.926[b]	4.000	313.000	.000	.178
router_proto * juggler_enabled	Pillai's Trace	.158	14.668[b]	4.000	313.000	.000	.158
	Wilks' Lambda	.842	14.668[b]	4.000	313.000	.000	.158
	Hotelling's Trace	.187	14.668[b]	4.000	313.000	.000	.158
	Roy's Largest Root	.187	14.668[b]	4.000	313.000	.000	.158

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

b. Exact statistic

Levene's Test of Equality of Error Variances[a]

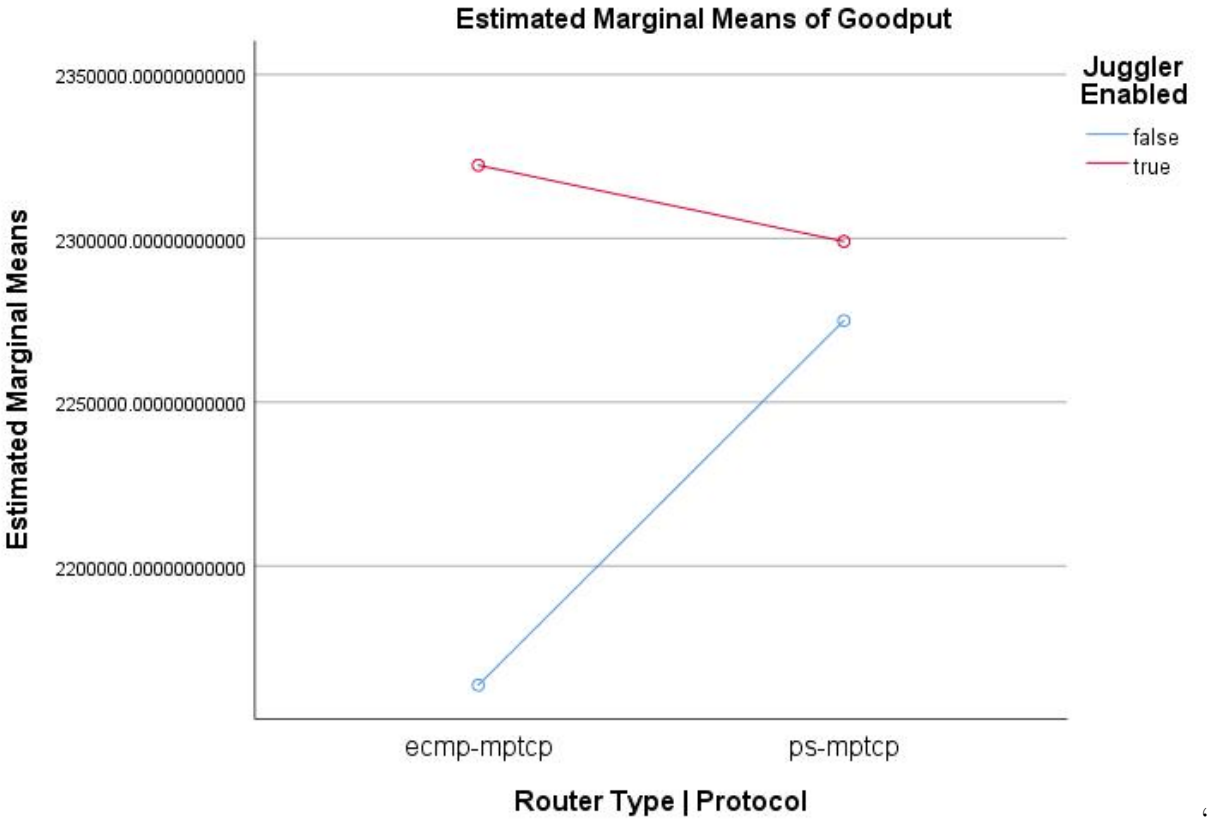
		Levene Statistic	df1	df2	Sig.
Goodput	Based on Mean	14.398	3	316	.000
	Based on Median	13.362	3	316	.000
	Based on Median and with adjusted df	13.362	3	117.464	.000
	Based on trimmed mean	13.644	3	316	.000
Throughput	Based on Mean	3.445	3	316	.017
	Based on Median	3.768	3	316	.011
	Based on Median and with adjusted df	3.768	3	253.644	.011
	Based on trimmed mean	3.537	3	316	.015
Flow Completion Time	Based on Mean	3.444	3	316	.017
	Based on Median	.940	3	316	.422
	Based on Median and with adjusted df	.940	3	79.213	.425
	Based on trimmed mean	.950	3	316	.417
Mean Network Utilization	Based on Mean	72.620	3	316	.000
	Based on Median	70.797	3	316	.000
	Based on Median and with adjusted df	70.797	3	158.938	.000
	Based on trimmed mean	72.991	3	316	.000

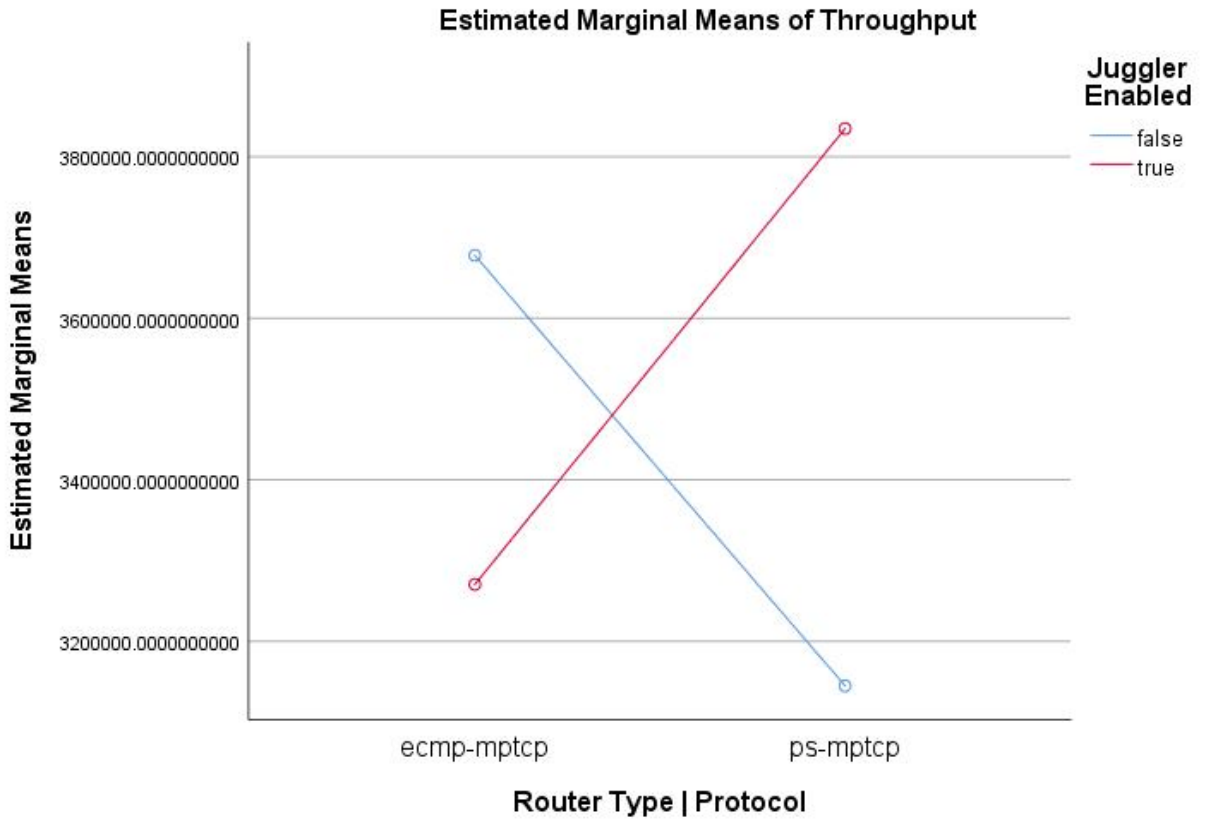
Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

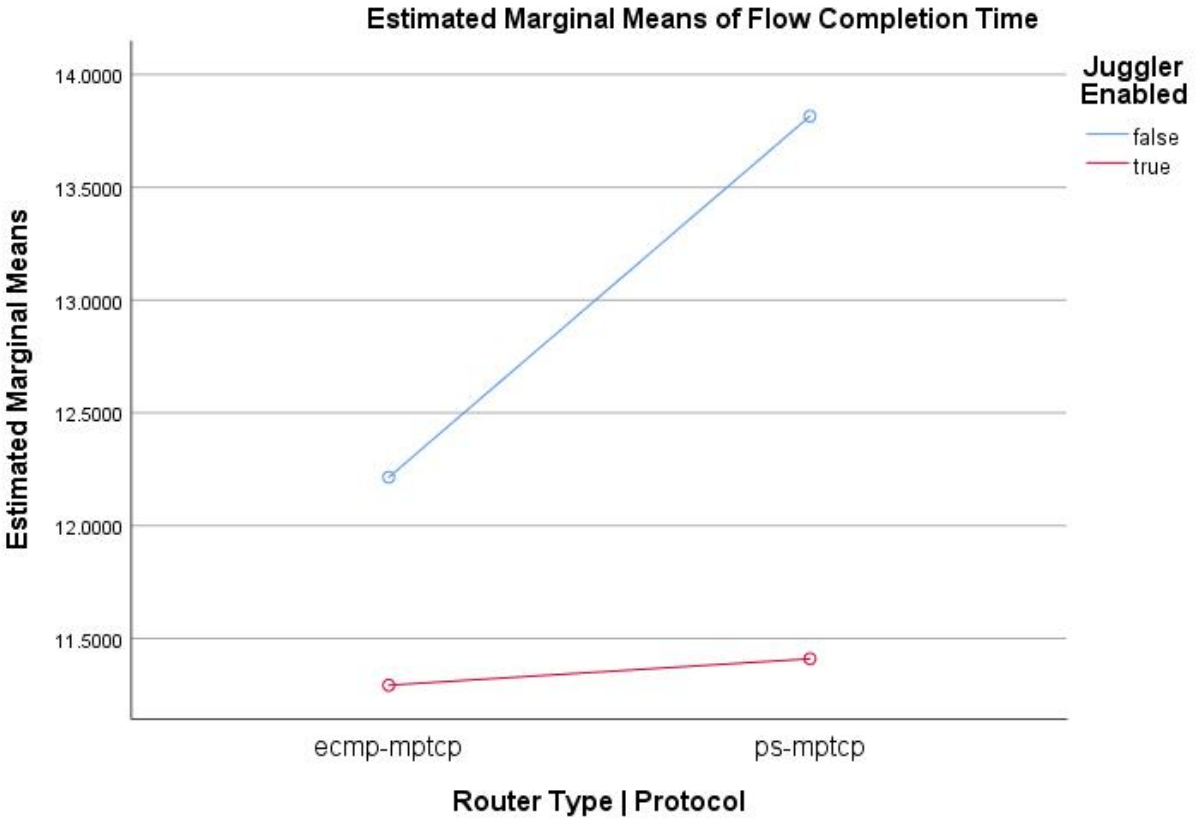
a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

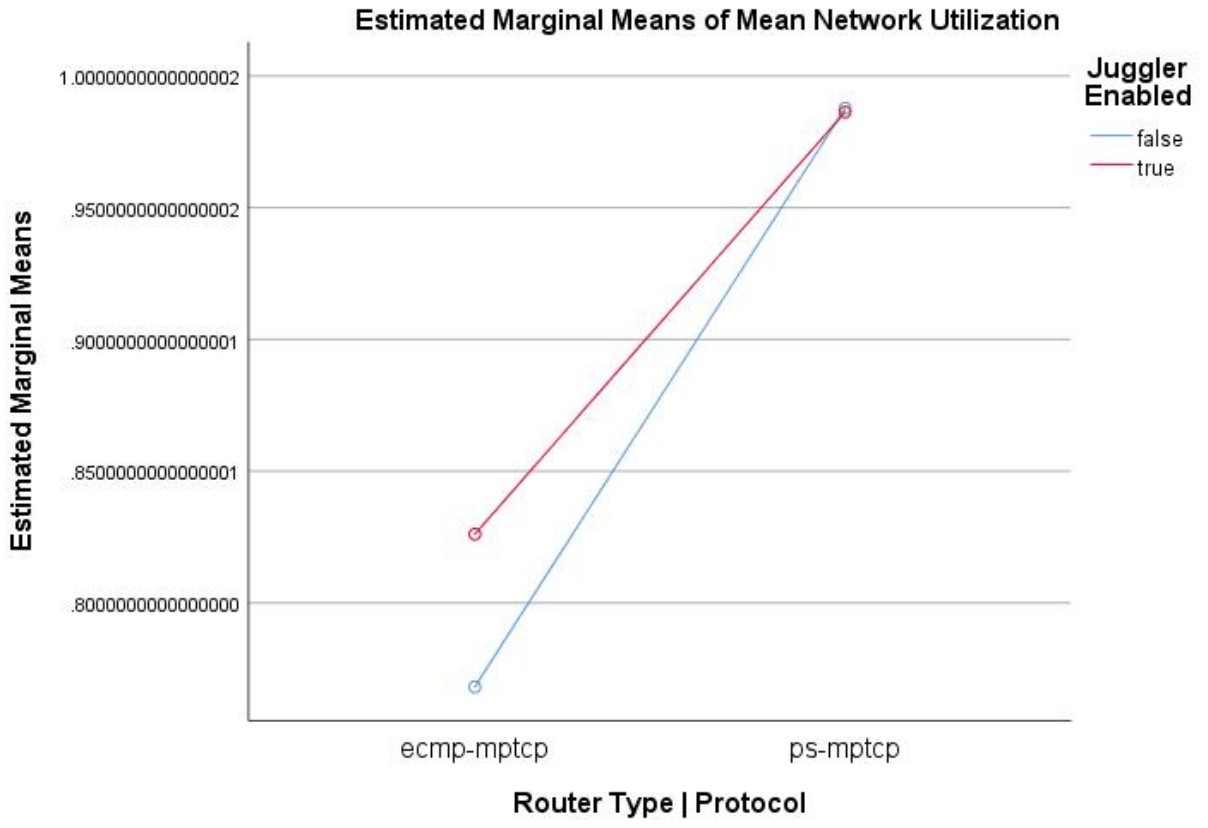
Tests of Between-Subjects Effects

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Goodput	1185917161418.956[a]	3	395305720472.985	15.165	.000	.126
	Throughput	25699803401844.000[b]	3	8566601133948.000	7.506	.000	.067
	Flow Completion Time	324.508[c]	3	108.169	.922	.431	.009
	Mean Network Utilization	3.020[d]	3	1.007	164.785	.000	.610
Intercept	Goodput	1641630493106549.200	1	1641630493106549.200	62976.844	.000	.995
	Throughput	3879741138097618.000	1	3879741138097618.000	3399.451	.000	.915
	Flow Completion Time	47495.577	1	47495.577	404.727	.000	.562
	Mean Network Utilization	254.644	1	254.644	41678.656	.000	.992
router_proto	Goodput	154932369677.420	1	154932369677.420	5.944	.015	.018
	Throughput	19768266553.347	1	19768266553.347	.017	.895	.000
	Flow Completion Time	59.060	1	59.060	.503	.479	.002
	Mean Network Utilization	2.886	1	2.886	472.344	.000	.599
juggler_enabled	Goodput	669000223129.758	1	669000223129.758	25.664	.000	.075
	Throughput	1597286448221.953	1	1597286448221.953	1.400	.238	.004
	Flow Completion Time	221.415	1	221.415	1.887	.171	.006
	Mean Network Utilization	.064	1	.064	10.449	.001	.032
router_proto * juggler_enabled	Goodput	361984568611.831	1	361984568611.831	13.887	.000	.042
	Throughput	24082748687068.010	1	24082748687068.010	21.101	.000	.063
	Flow Completion Time	44.034	1	44.034	.375	.541	.001
	Mean Network Utilization	.071	1	.071	11.562	.001	.035
Error	Goodput	8237237716329.423	316	26067207963.068			
	Throughput	360645914231374.800	316	1141284538706.882			
	Flow Completion Time	37083.267	316	117.352			
	Mean Network Utilization	1.931	316	.006			
Total	Goodput	1651053647984294.200	320				
	Throughput	4266086855730841.000	320				
	Flow Completion Time	84903.352	320				
	Mean Network Utilization	259.595	320				
Corrected Total	Goodput	9423154877748.379	319				
	Throughput	386345717633218.800	319				
	Flow Completion Time	37407.775	319				
	Mean Network Utilization	4.951	319				


Estimated Marginal Means of Goodput
Estimated Marginal Means
2350000.00000000000
2300000.00000000000
2250000.00000000000
2200000.00000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Throughput
Juggler Enabled
false
true
Estimated Marginal Means
3800000.0000000000
3600000.0000000000
3400000.0000000000
3200000.0000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol


Estimated Marginal Means of Flow Completion Time
Estimated Marginal Means
14.0000
13.5000
13.0000
12.5000
12.0000
11.5000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true



B.2.1.4 Noisy Network (1-many): Query Flows

Between-Subjects Factors

		Value Label	N
Router Type | Protocol	1	ecmp-mptcp	140
	2	ps-mptcp	140
Juggler Enabled	1	false	140
	2	true	140

Descriptive Statistics

	Router Type | Protocol	Juggler Enabled	Mean	Std. Deviation	N
Goodput	ecmp-mptcp	false	130081.98664975523000	72772.810126721150000	70
		true	135277.65198086870000	53008.797152710205000	70
		Total	132679.81931531190000	63486.588467637540000	140
	ps-mptcp	false	134492.80995088053000	77517.400335168930000	70
		true	180710.45075467100000	138465.781015943000000	70
		Total	157601.63035277577000	114184.664648373890000	140
	Total	false	132287.39830031793000	74944.303339390970000	140
		true	157994.05136776983000	106920.649768514060000	140
		Total	145140.72483404374000	93056.885517486720000	280
Throughput	ecmp-mptcp	false	190642.94739767036000	97421.231508653070000	70
		true	195316.38902020728000	73956.042334802190000	70
		Total	192979.66820893880000	86208.360720944630000	140
	ps-mptcp	false	211919.80978495884000	99482.685924615840000	70
		true	249231.59533545720000	176311.925156181620000	70
		Total	230575.70256020800000	143855.748335560700000	140
	Total	false	201281.37859131470000	98681.817358587430000	140
		true	222273.99217783235000	137397.750625034820000	140
		Total	211777.68538457356000	119864.111506060200000	280
Flow Completion Time	ecmp-mptcp	false	.088426	.0252672	70
		true	.110597	.1640139	70
		Total	.099511	.1174488	140
	ps-mptcp	false	.091969	.0434341	70
		true	.074584	.0446935	70
		Total	.083276	.0447676	140
	Total	false	.090197	.0354479	140
		true	.092591	.1211266	140
		Total	.091394	.0890898	280
Mean Network Utilization	ecmp-mptcp	false	.512515936142114	.179398217637920	70
		true	.513437301093449	.169870611429815	70
		Total	.512976618617781	.174070442310808	140
	ps-mptcp	false	.689051071752206	.142344850413757	70
		true	.697602275953379	.148075006436585	70
		Total	.693326673852792	.144778409749915	140
	Total	false	.600783503947160	.184069143677082	140
		true	.605519788523414	.183708007441376	140
		Total	.603151646235287	.183574148946625	280

Box's Test of Equality of Covariance Matrices[a]

Box's M	511.318
F	16.599
df1	30
df2	209438.549
Sig.	.000

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	.945	1178.895[b]	4.000	273.000	.000	.945
	Wilks' Lambda	.055	1178.895[b]	4.000	273.000	.000	.945
	Hotelling's Trace	17.273	1178.895[b]	4.000	273.000	.000	.945
	Roy's Largest Root	17.273	1178.895[b]	4.000	273.000	.000	.945
router_proto	Pillai's Trace	.250	22.760[b]	4.000	273.000	.000	.250
	Wilks' Lambda	.750	22.760[b]	4.000	273.000	.000	.250
	Hotelling's Trace	.333	22.760[b]	4.000	273.000	.000	.250
	Roy's Largest Root	.333	22.760[b]	4.000	273.000	.000	.250
juggler_enabled	Pillai's Trace	.045	3.251[b]	4.000	273.000	.013	.045
	Wilks' Lambda	.955	3.251[b]	4.000	273.000	.013	.045
	Hotelling's Trace	.048	3.251[b]	4.000	273.000	.013	.045
	Roy's Largest Root	.048	3.251[b]	4.000	273.000	.013	.045
router_proto * juggler_enabled	Pillai's Trace	.035	2.466[b]	4.000	273.000	.045	.035
	Wilks' Lambda	.965	2.466[b]	4.000	273.000	.045	.035
	Hotelling's Trace	.036	2.466[b]	4.000	273.000	.045	.035
	Roy's Largest Root	.036	2.466[b]	4.000	273.000	.045	.035

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

b. Exact statistic

Levene's Test of Equality of Error Variances[a]

		Levene Statistic	df1	df2	Sig.
Goodput	Based on Mean	3.070	3	276	.028
	Based on Median	2.366	3	276	.071
	Based on Median and with adjusted df	2.366	3	167.873	.073
	Based on trimmed mean	2.410	3	276	.067
Throughput	Based on Mean	2.428	3	276	.066
	Based on Median	2.010	3	276	.113
	Based on Median and with adjusted df	2.010	3	170.040	.114
	Based on trimmed mean	1.931	3	276	.125
Flow Completion Time	Based on Mean	4.726	3	276	.003
	Based on Median	1.654	3	276	.177
	Based on Median and with adjusted df	1.654	3	86.973	.183
	Based on trimmed mean	1.656	3	276	.177
Mean Network Utilization	Based on Mean	.749	3	276	.524
	Based on Median	.747	3	276	.525
	Based on Median and with adjusted df	.747	3	265.266	.525
	Based on trimmed mean	.685	3	276	.562

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Tests of Between-Subjects Effects

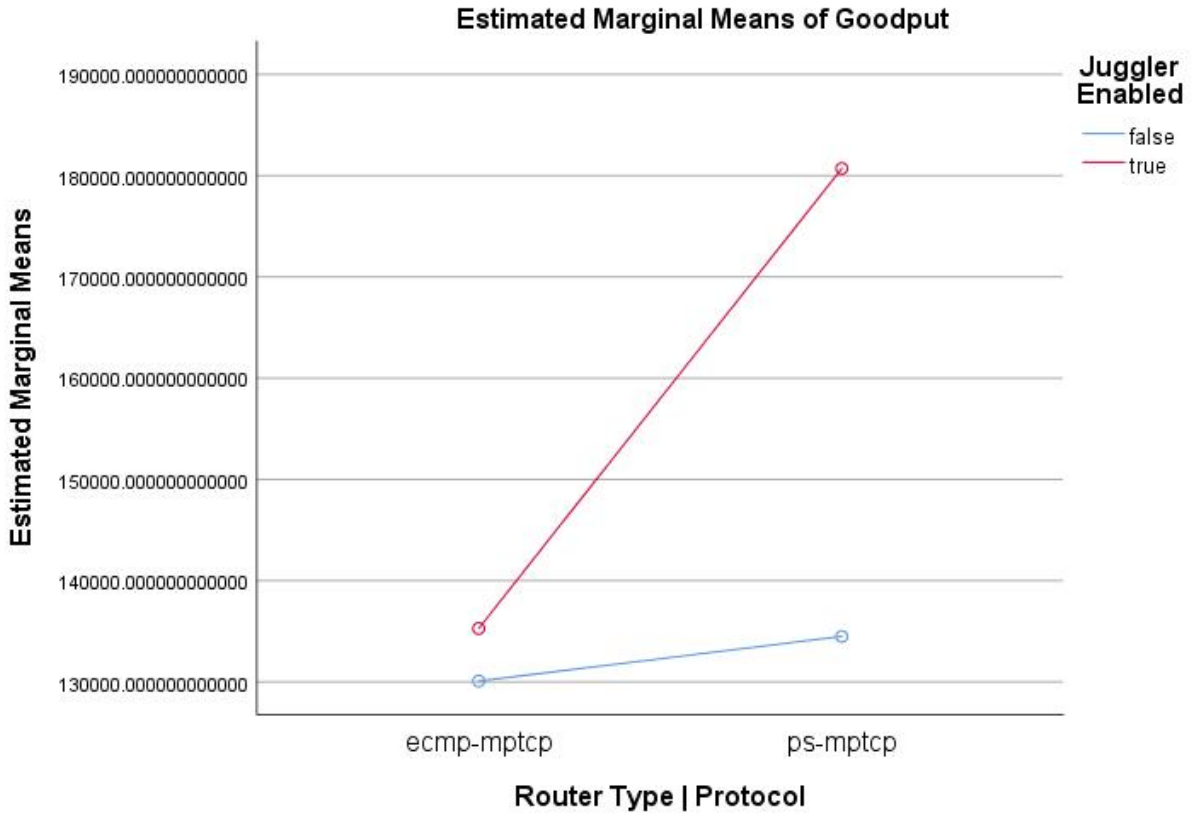
Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Goodput	119184050666.632[a]	3	39728016888.877	4.774	.003	.049
	Throughput	148432689840.725[b]	3	49477563280.242	3.538	.015	.037
	Flow Completion Time	.046[c]	3	.015	1.962	.120	.021
	Mean Network Utilization	2.279[d]	3	.760	29.442	.000	.242
Intercept	Goodput	5898432401498.456	1	5898432401498.456	708.786	.000	.720
	Throughput	12557940647517.260	1	12557940647517.260	897.908	.000	.765
	Flow Completion Time	2.339	1	2.339	297.718	.000	.519
	Mean Network Utilization	101.862	1	101.862	3947.058	.000	.935
router_proto	Goodput	43476766577.093	1	43476766577.093	5.224	.023	.019
	Throughput	98942325925.927	1	98942325925.927	7.074	.008	.025
	Flow Completion Time	.018	1	.018	2.349	.127	.008
	Mean Network Utilization	2.277	1	2.277	88.225	.000	.242
juggler_enabled	Goodput	46258240835.123	1	46258240835.123	5.559	.019	.020
	Throughput	30848287763.500	1	30848287763.500	2.206	.139	.008
	Flow Completion Time	.000	1	.000	.051	.821	.000
	Mean Network Utilization	.002	1	.002	.061	.805	.000
router_proto * juggler_enabled	Goodput	29449043254.416	1	29449043254.416	3.539	.061	.013
	Throughput	18642076151.299	1	18642076151.299	1.333	.249	.005
	Flow Completion Time	.027	1	.027	3.486	.063	.012
	Mean Network Utilization	.001	1	.001	.039	.843	.000
Error	Goodput	2296839869211.251	276	8321883584.099			
	Throughput	3860073368530.561	276	13985773074.386			
	Flow Completion Time	2.168	276	.008			
	Mean Network Utilization	7.123	276	.026			
Total	Goodput	8314456321376.337	280				
	Throughput	16566446705888.550	280				
	Flow Completion Time	4.553	280				
	Mean Network Utilization	111.264	280				
Corrected Total	Goodput	2416023919877.883	279				
	Throughput	4008506058371.286	279				
	Flow Completion Time	2.214	279				
	Mean Network Utilization	9.402	279				

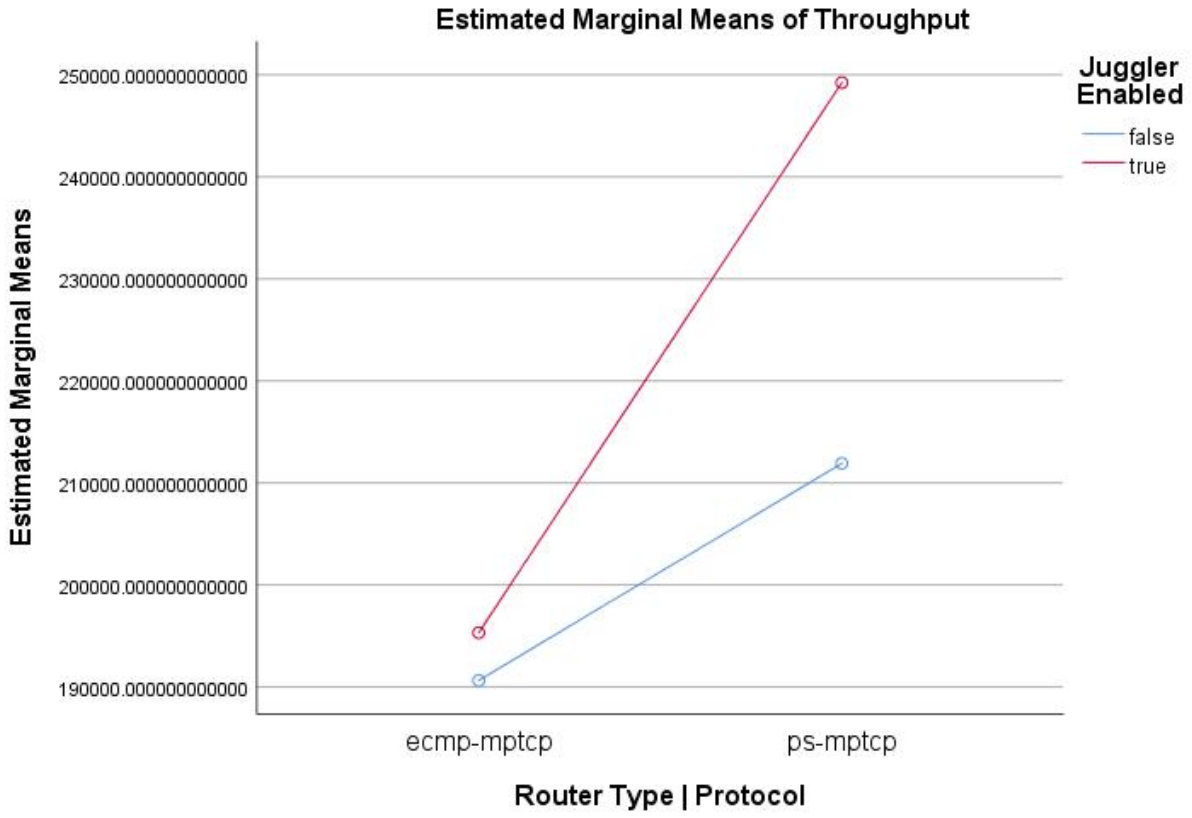
a. R Squared = .049 (Adjusted R Squared = .039)

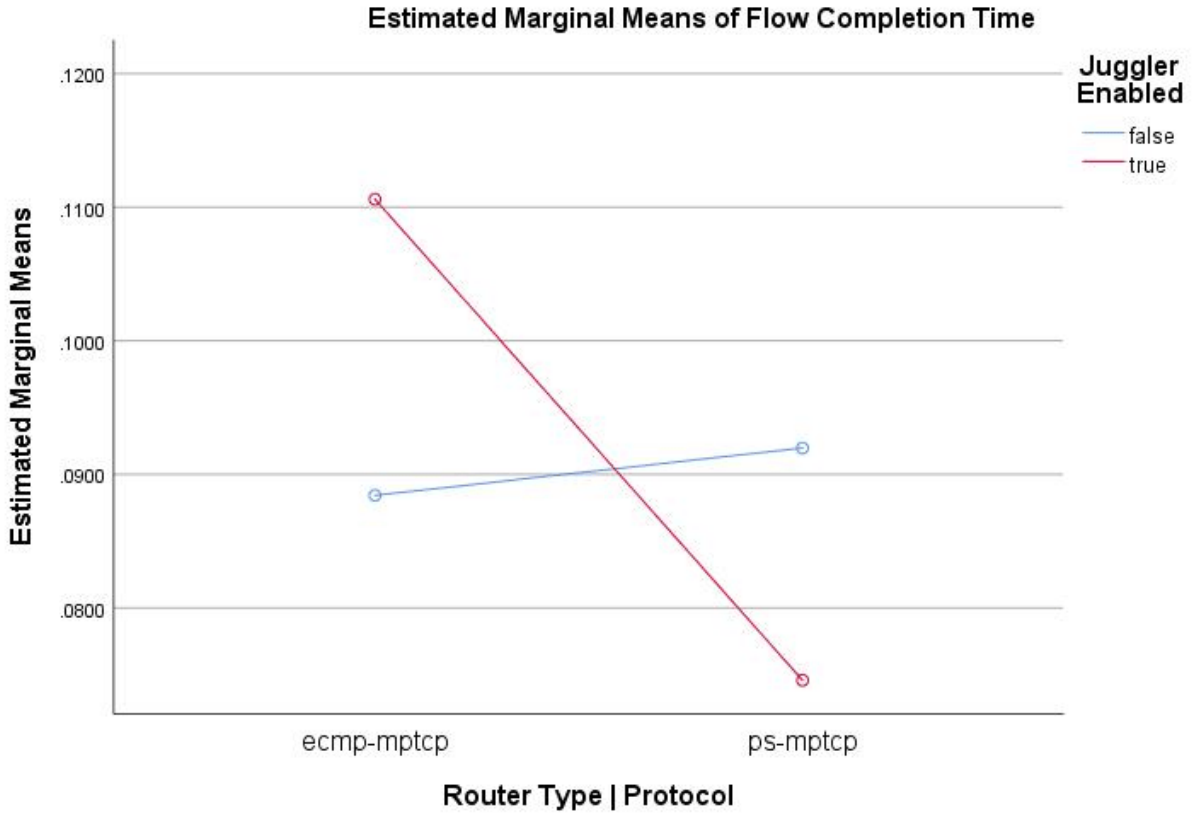
b. R Squared = .037 (Adjusted R Squared = .027)

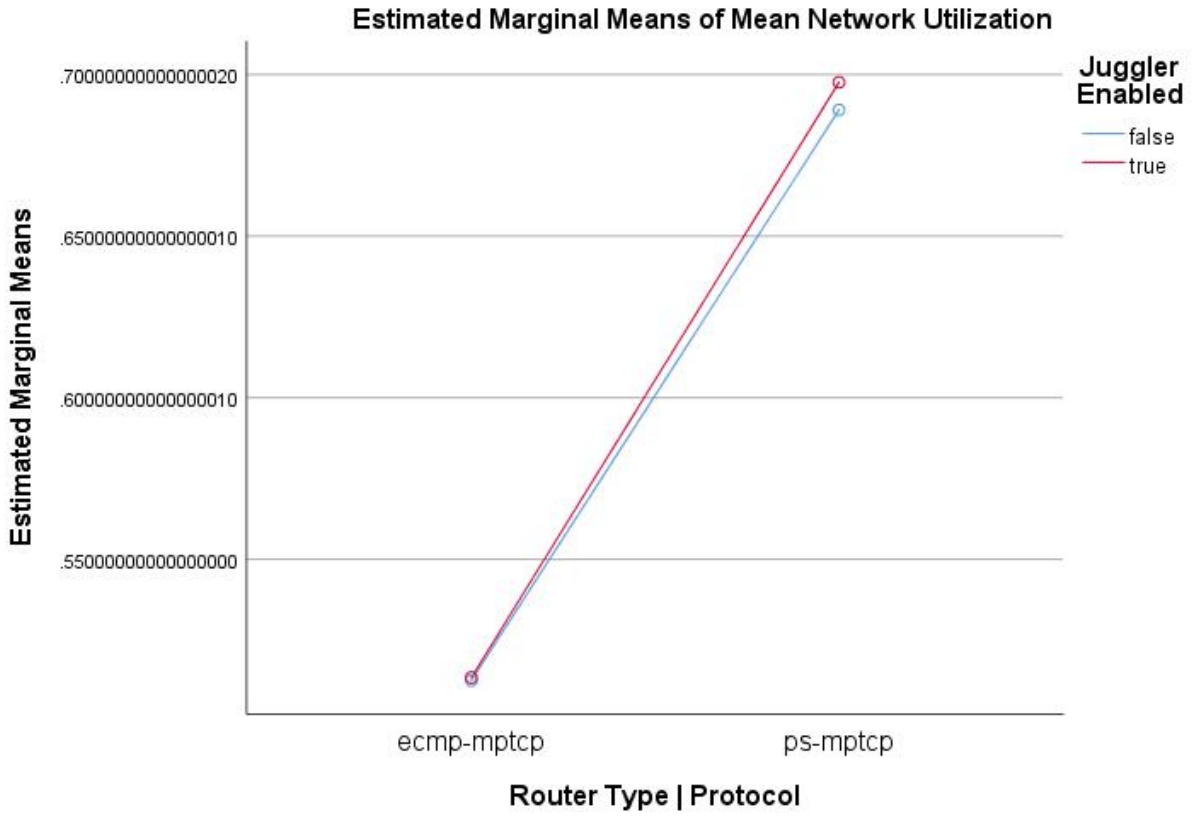
c. R Squared = .021 (Adjusted R Squared = .010)

d. R Squared = .242 (Adjusted R Squared = .234)


Estimated Marginal Means of Goodput
Estimated Marginal Means
190000.000000000000
180000.000000000000
170000.000000000000
160000.000000000000
150000.000000000000
140000.000000000000
130000.000000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Throughput
Estimated Marginal Means
250000.000000000000
240000.000000000000
230000.000000000000
220000.000000000000
210000.000000000000
200000.000000000000
190000.000000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Flow Completion Time
Estimated Marginal Means
.1200
.1100
.1000
.0900
.0800
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Mean Network Utilization
Estimated Marginal Means
.70000000000000020
.65000000000000010
.60000000000000010
.55000000000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true

B.2.1.5 Noisy Network (1-many): Short Flows

Between-Subjects Factors

		Value Label	N
Router Type | Protocol	1	ecmp-mptcp	140
	2	ps-mptcp	140
Juggler Enabled	1	false	140
	2	true	140

Descriptive Statistics

	Router Type | Protocol	Juggler Enabled	Mean	Std. Deviation	N
Goodput	ecmp-mptcp	false	301659.4835088374700	44046.62460222136000	70
		true	249797.5678622280800	50415.79013050923000	70
		Total	275728.5256855328300	53870.76031883011000	140
	ps-mptcp	false	307845.5648762539500	45154.04286551144500	70
		true	303641.8948754358600	114422.56513533687000	70
		Total	305743.7298758447000	86693.28380050460000	140
	Total	false	304752.5241925456000	44551.30937239330000	140
		true	276719.7313688321000	92146.16948129212000	140
		Total	290736.1277806887500	73595.22852963130000	280
Throughput	ecmp-mptcp	false	483559.1958502109000	124157.17239425132000	70
		true	570648.7034776817000	139848.49337941714000	70
		Total	527103.9496639460000	138817.46454512418000	140
	ps-mptcp	false	474211.7405770042700	103859.37944501467000	70
		true	695273.4408199589000	128847.58436991475000	70
		Total	584742.5906984812000	160936.87296466570000	140
	Total	false	478885.4682136077000	114143.05485138515000	140
		true	632961.0721488203000	147852.44823227334000	140
		Total	555923.2701812141000	152767.99844349723000	280
Flow Completion Time	ecmp-mptcp	false	1.733577	.2586301	70
		true	2.115506	.3409796	70
		Total	1.924541	.3572802	140
	ps-mptcp	false	1.701321	.2714573	70
		true	1.812667	.3892399	70
		Total	1.756994	.3389839	140
	Total	false	1.717449	.2646613	140
		true	1.964086	.3949898	140
		Total	1.840768	.3576145	280
Mean Network Utilization	ecmp-mptcp	false	.737176175514362	.120499870844079	70
		true	.769186526381652	.119589090349368	70
		Total	.753181350948006	.120686444273761	140
	ps-mptcp	false	.932168862844109	.044471631643217	70
		true	.933129572385321	.039543976498282	70
		Total	.932649217614715	.041931127024434	140
	Total	false	.834672519179235	.133279966642400	140
		true	.851158049383486	.121009256070188	140
		Total	.842915284281360	.127332261419031	280

Box's Test of Equality of Covariance Matrices[a]

Box's M	640.417
F	20.790
df1	30
df2	209438.549
Sig.	.000

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	.998	35133.396[b]	4.000	273.000	.000	.998
	Wilks' Lambda	.002	35133.396[b]	4.000	273.000	.000	.998
	Hotelling's Trace	514.775	35133.396[b]	4.000	273.000	.000	.998
	Roy's Largest Root	514.775	35133.396[b]	4.000	273.000	.000	.998
router_proto	Pillai's Trace	.534	78.204[b]	4.000	273.000	.000	.534
	Wilks' Lambda	.466	78.204[b]	4.000	273.000	.000	.534
	Hotelling's Trace	1.146	78.204[b]	4.000	273.000	.000	.534
	Roy's Largest Root	1.146	78.204[b]	4.000	273.000	.000	.534
juggler_enabled	Pillai's Trace	.430	51.435[b]	4.000	273.000	.000	.430
	Wilks' Lambda	.570	51.435[b]	4.000	273.000	.000	.430
	Hotelling's Trace	.754	51.435[b]	4.000	273.000	.000	.430
	Roy's Largest Root	.754	51.435[b]	4.000	273.000	.000	.430
router_proto * juggler_enabled	Pillai's Trace	.099	7.487[b]	4.000	273.000	.000	.099
	Wilks' Lambda	.901	7.487[b]	4.000	273.000	.000	.099
	Hotelling's Trace	.110	7.487[b]	4.000	273.000	.000	.099
	Roy's Largest Root	.110	7.487[b]	4.000	273.000	.000	.099

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

b. Exact statistic

Levene's Test of Equality of Error Variances[a]

		Levene Statistic	df1	df2	Sig.
Goodput	Based on Mean	5.427	3	276	.001
	Based on Median	3.145	3	276	.026
	Based on Median and with adjusted df	3.145	3	123.588	.028
	Based on trimmed mean	3.597	3	276	.014
Throughput	Based on Mean	1.308	3	276	.272
	Based on Median	.973	3	276	.406
	Based on Median and with adjusted df	.973	3	237.644	.406
	Based on trimmed mean	1.297	3	276	.276
Flow Completion Time	Based on Mean	2.982	3	276	.032
	Based on Median	3.134	3	276	.026
	Based on Median and with adjusted df	3.134	3	254.592	.026
	Based on trimmed mean	2.807	3	276	.040
Mean Network Utilization	Based on Mean	36.306	3	276	.000
	Based on Median	34.137	3	276	.000
	Based on Median and with adjusted df	34.137	3	174.861	.000
	Based on trimmed mean	36.151	3	276	.000

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Tests of Between-Subjects Effects

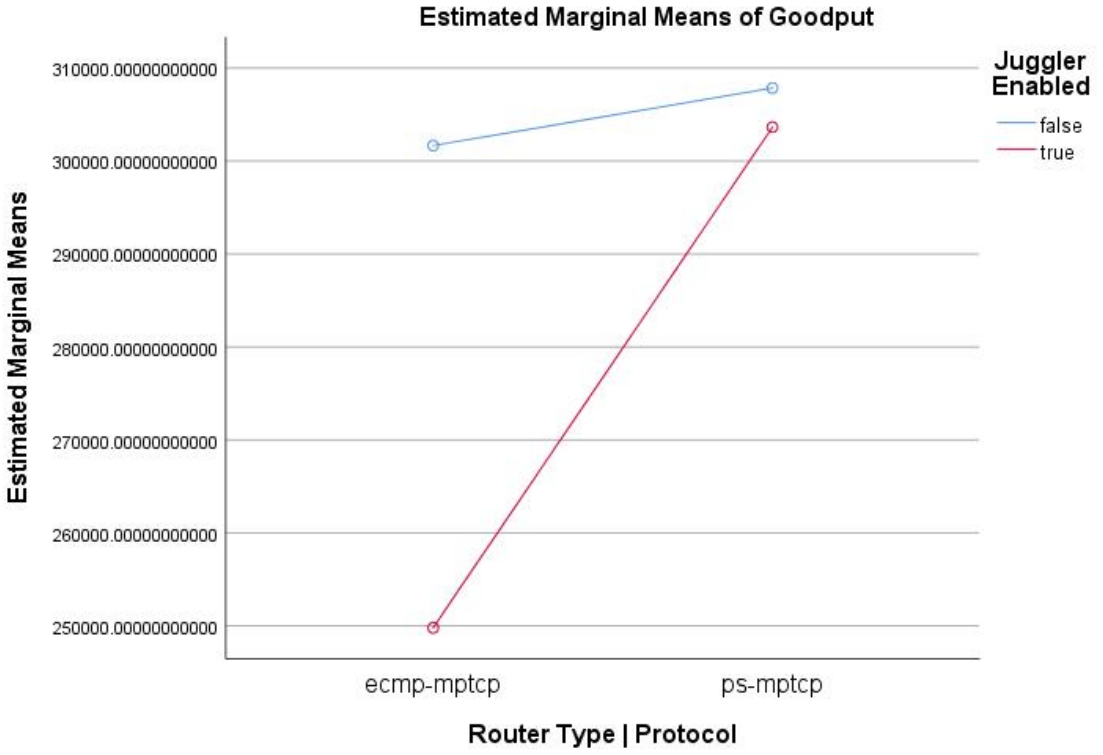
Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Goodput	157820393541.443[a]	3	52606797847.148	10.729	.000	.104
	Throughput	2208404923680.102[b]	3	736134974560.034	47.218	.000	.339
	Flow Completion Time	7.504[c]	3	2.501	24.503	.000	.210
	Mean Network Utilization	2.291[d]	3	.764	94.367	.000	.506
Intercept	Goodput	23667698879134.523	1	23667698879134.523	4826.875	.000	.946
	Throughput	86534191052113.020	1	86534191052113.020	5550.526	.000	.953
	Flow Completion Time	948.759	1	948.759	9293.514	.000	.971
	Mean Network Utilization	198.942	1	198.942	24588.619	.000	.989
router_proto	Goodput	63063873781.029	1	63063873781.029	12.861	.000	.045
	Throughput	232554905821.561	1	232554905821.561	14.917	.000	.051
	Flow Completion Time	1.965	1	1.965	19.248	.000	.065
	Mean Network Utilization	2.255	1	2.255	278.663	.000	.502
juggler_enabled	Goodput	55008623144.807	1	55008623144.807	11.219	.001	.039
	Throughput	1661750420960.038	1	1661750420960.038	106.589	.000	.279
	Flow Completion Time	4.258	1	4.258	41.710	.000	.131
	Mean Network Utilization	.019	1	.019	2.351	.126	.008
router_proto * juggler_enabled	Goodput	39747896615.605	1	39747896615.605	8.106	.005	.029
	Throughput	314099596898.504	1	314099596898.504	20.147	.000	.068
	Flow Completion Time	1.281	1	1.281	12.551	.000	.043
	Mean Network Utilization	.017	1	.017	2.085	.150	.007
Error	Goodput	1353315494248.255	276	4903317008.146			
	Throughput	4302914192532.523	276	15590268813.524			
	Flow Completion Time	28.176	276	.102			
	Mean Network Utilization	2.233	276	.008			
Total	Goodput	25178834766924.223	280				
	Throughput	93045510168325.560	280				
	Flow Completion Time	984.440	280				
	Mean Network Utilization	203.465	280				
Corrected Total	Goodput	1511135887789.698	279				
	Throughput	6511319116212.625	279				
	Flow Completion Time	35.681	279				
	Mean Network Utilization	4.524	279				

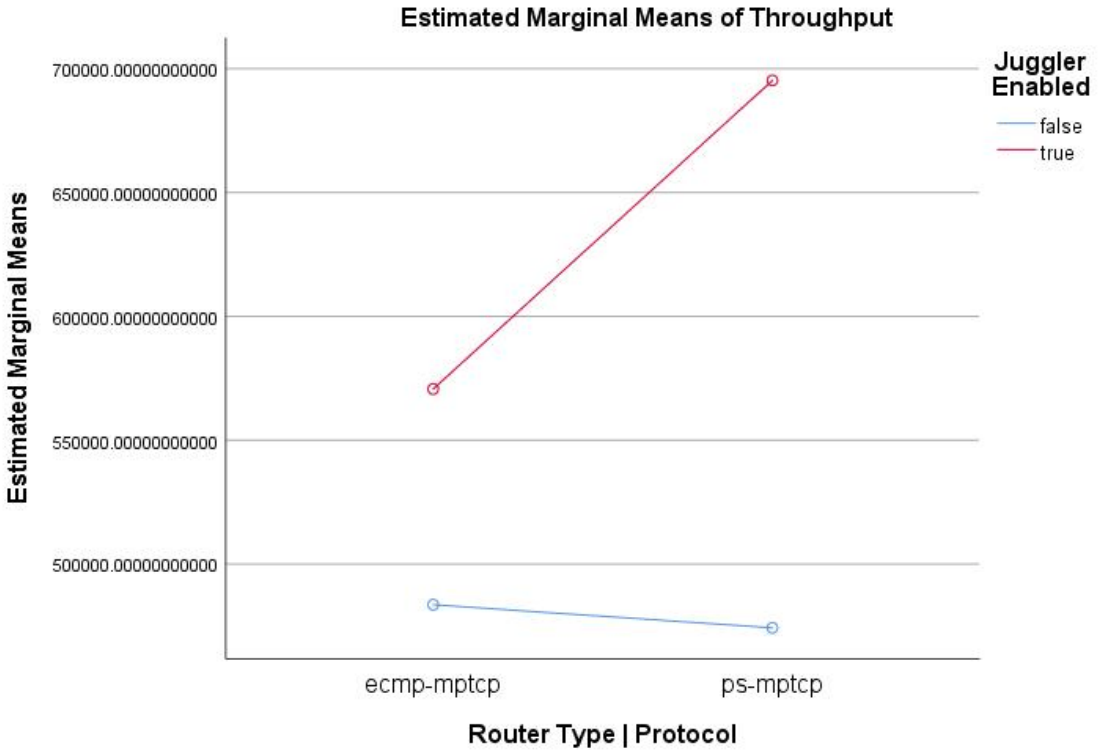
a. R Squared = .104 (Adjusted R Squared = .095)

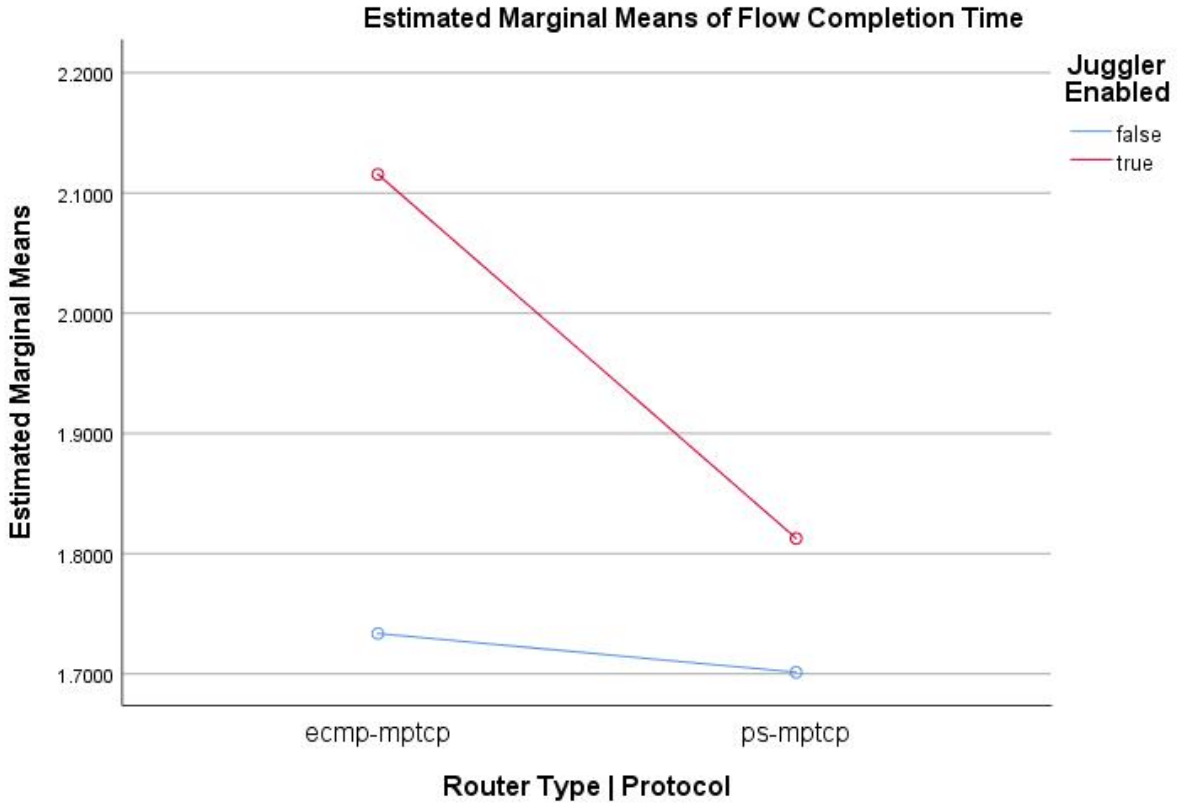
b. R Squared = .339 (Adjusted R Squared = .332)

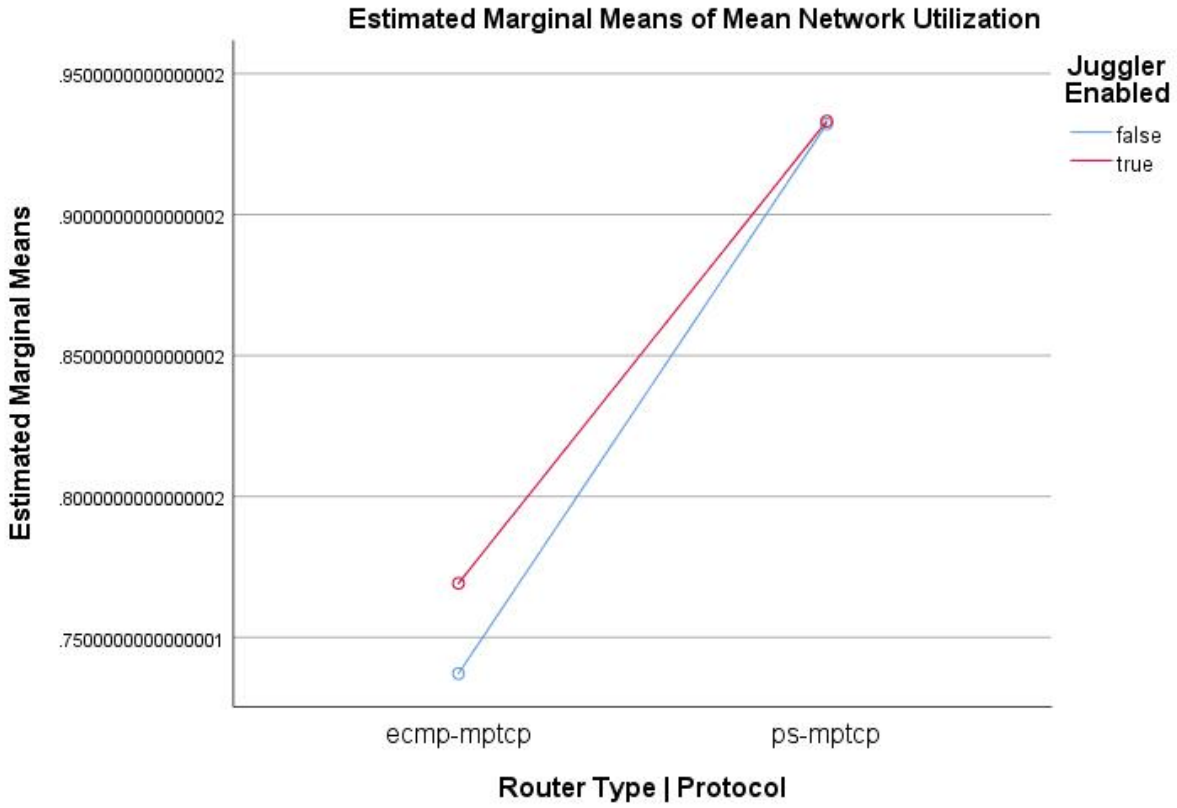
c. R Squared = .210 (Adjusted R Squared = .202)

d. R Squared = .506 (Adjusted R Squared = .501)


Estimated Marginal Means of Goodput
Estimated Marginal Means
310000.00000000000
300000.00000000000
290000.00000000000
280000.00000000000
270000.00000000000
260000.00000000000
250000.00000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Throughput
Estimated Marginal Means
700000.00000000000
650000.00000000000
600000.00000000000
550000.00000000000
500000.00000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Flow Completion Time
Estimated Marginal Means
2.2000
2.1000
2.0000
1.9000
1.8000
1.7000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Mean Network Utilization
Estimated Marginal Means
.9500000000000002
.9000000000000002
.8500000000000002
.8000000000000002
.7500000000000001
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true

B.2.1.6 Noisy Network (1-many): Long Flows

Between-Subjects Factors

		Value Label	N
Router Type | Protocol	1	ecmp-mptcp	140
	2	ps-mptcp	140
Juggler Enabled	1	false	140
	2	true	140

Descriptive Statistics

	Router Type | Protocol	Juggler Enabled	Mean	Std. Deviation	N
Goodput	ecmp-mptcp	false	309335.1853039946000	62642.33135123693000	70
		true	312189.9091078949000	94032.96759999933000	70
		Total	310762.5472059449000	79619.47346402657000	140
	ps-mptcp	false	319248.1129733109000	49958.05183836121400	70
		true	334333.9064675750000	114823.20284165119000	70
		Total	326791.0097204427600	88549.35646553370000	140
	Total	false	314291.6491386527000	56670.84373031947000	140
		true	323261.9077877348000	105154.75400242770000	140
		Total	318776.7784631940000	84434.41253414651000	280
Throughput	ecmp-mptcp	false	756729.7156695098000	96954.55016255463000	70
		true	715990.9139932843000	91457.84029395480000	70
		Total	736360.3148313971000	96105.96376565058000	140
	ps-mptcp	false	788454.4972635315000	101992.79649208076000	70
		true	771451.4704736958000	98913.43585700956000	70
		Total	779952.9838686133000	100465.82463858456000	140
	Total	false	772592.1064665208000	100416.88355357054000	140
		true	743721.1922334898000	98911.17540333194000	140
		Total	758156.6493500053000	100533.63110325145000	280
Flow Completion Time	ecmp-mptcp	false	87.077251	12.2116932	70
		true	88.556587	16.3269302	70
		Total	87.816919	14.3841160	140
	ps-mptcp	false	83.612153	9.9048116	70
		true	83.324580	15.2975920	70
		Total	83.468366	12.8408313	140
	Total	false	85.344702	11.2138029	140
		true	85.940584	15.9807549	140
		Total	85.642643	13.7830694	280
Mean Network Utilization	ecmp-mptcp	false	.773969743488988	.118010882661553	70
		true	.801092703004221	.120282625212428	70
		Total	.787531223246605	.119500366608341	140
	ps-mptcp	false	.985975132940220	.006990405926231	70
		true	.988906678415254	.007043921459653	70
		Total	.987440905677737	.007144997885028	140
	Total	false	.879972438214604	.135110531366719	140
		true	.894999690709737	.126840478579103	140
		Total	.887486064462171	.131022101291228	280

Box's Test of Equality of Covariance Matrices[a]

Box's M	928.432
F	30.140
df1	30
df2	209438.549
Sig.	.000

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	.999	123425.309[b]	4.000	273.000	.000	.999
	Wilks' Lambda	.001	123425.309[b]	4.000	273.000	.000	.999
	Hotelling's Trace	1808.429	123425.309[b]	4.000	273.000	.000	.999
	Roy's Largest Root	1808.429	123425.309[b]	4.000	273.000	.000	.999
router_proto	Pillai's Trace	.624	113.129[b]	4.000	273.000	.000	.624
	Wilks' Lambda	.376	113.129[b]	4.000	273.000	.000	.624
	Hotelling's Trace	1.658	113.129[b]	4.000	273.000	.000	.624
	Roy's Largest Root	1.658	113.129[b]	4.000	273.000	.000	.624
juggler_enabled	Pillai's Trace	.054	3.889[b]	4.000	273.000	.004	.054
	Wilks' Lambda	.946	3.889[b]	4.000	273.000	.004	.054
	Hotelling's Trace	.057	3.889[b]	4.000	273.000	.004	.054
	Roy's Largest Root	.057	3.889[b]	4.000	273.000	.004	.054
router_proto * juggler_enabled	Pillai's Trace	.011	.746[b]	4.000	273.000	.561	.011
	Wilks' Lambda	.989	.746[b]	4.000	273.000	.561	.011
	Hotelling's Trace	.011	.746[b]	4.000	273.000	.561	.011
	Roy's Largest Root	.011	.746[b]	4.000	273.000	.561	.011

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

b. Exact statistic

Levene's Test of Equality of Error Variances[a]

		Levene Statistic	df1	df2	Sig.
Goodput	Based on Mean	4.232	3	276	.006
	Based on Median	1.619	3	276	.185
	Based on Median and with adjusted df	1.619	3	191.518	.186
	Based on trimmed mean	2.125	3	276	.097
Throughput	Based on Mean	.030	3	276	.993
	Based on Median	.021	3	276	.996
	Based on Median and with adjusted df	.021	3	267.662	.996
	Based on trimmed mean	.019	3	276	.996
Flow Completion Time	Based on Mean	4.008	3	276	.008
	Based on Median	2.730	3	276	.044
	Based on Median and with adjusted df	2.730	3	238.638	.045
	Based on trimmed mean	3.199	3	276	.024
Mean Network Utilization	Based on Mean	92.154	3	276	.000
	Based on Median	90.314	3	276	.000
	Based on Median and with adjusted df	90.314	3	138.532	.000
	Based on trimmed mean	92.744	3	276	.000

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + router_proto + juggler_enabled + router_proto * juggler_enabled

Tests of Between-Subjects Effects

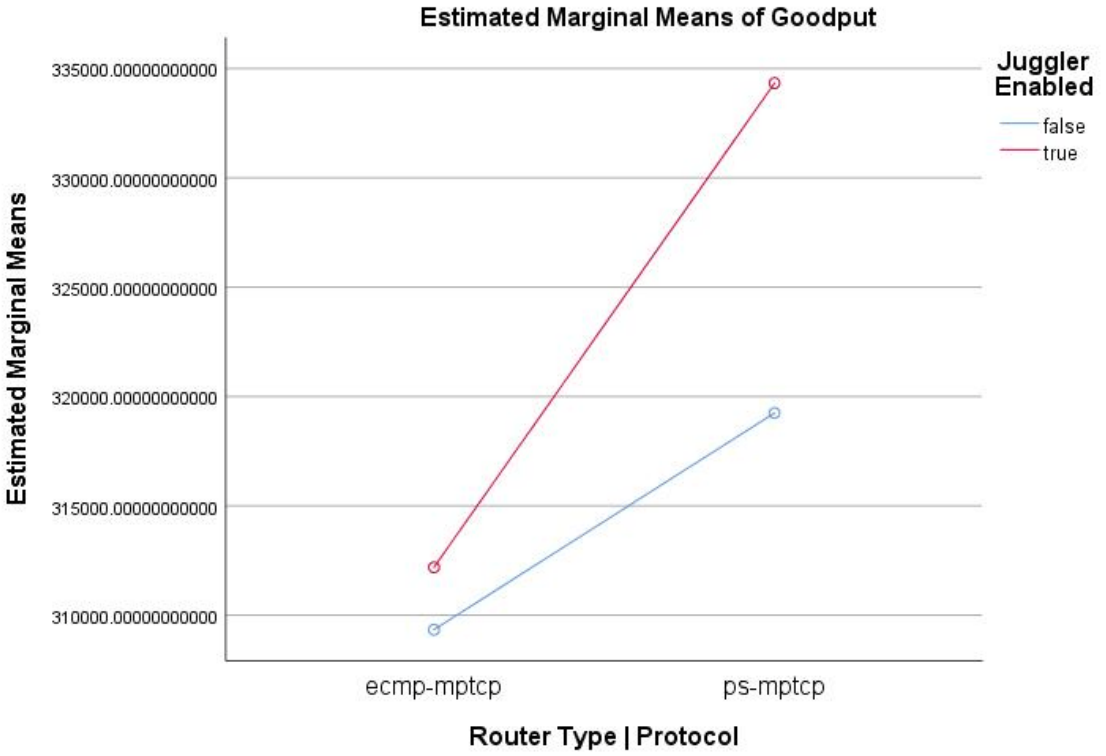
Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Goodput	26234384207.693[a]	3	8744794735.898	1.230	.299	.013
	Throughput	201228806436.253[b]	3	67076268812.084	7.070	.000	.071
	Flow Completion Time	1403.183[c]	3	467.728	2.502	.060	.026
	Mean Network Utilization	2.824[d]	3	.941	132.127	.000	.590
Intercept	Goodput	28453217656464.210	1	28453217656464.210	4000.954	.000	.935
	Throughput	160944421387015.700	1	160944421387015.700	16963.338	.000	.984
	Flow Completion Time	2053705.437	1	2053705.437	10985.088	.000	.975
	Mean Network Utilization	220.537	1	220.537	30960.183	.000	.991
router_proto	Goodput	17983812740.507	1	17983812740.507	2.529	.113	.009
	Throughput	133022455565.184	1	133022455565.184	14.020	.000	.048
	Flow Completion Time	1323.694	1	1323.694	7.080	.008	.025
	Mean Network Utilization	2.797	1	2.797	392.725	.000	.587
juggler_enabled	Goodput	5632587816.201	1	5632587816.201	.792	.374	.003
	Throughput	58347078205.568	1	58347078205.568	6.150	.014	.022
	Flow Completion Time	24.855	1	24.855	.133	.716	.000
	Mean Network Utilization	.016	1	.016	2.219	.137	.008
router_proto * juggler_enabled	Goodput	2617983650.984	1	2617983650.984	.368	.545	.001
	Throughput	9859272665.504	1	9859272665.504	1.039	.309	.004
	Flow Completion Time	54.634	1	54.634	.292	.589	.001
	Mean Network Utilization	.010	1	.010	1.438	.232	.005
Error	Goodput	1962804051368.522	276	7111608881.770			
	Throughput	2618627257766.244	276	9487779919.443			
	Flow Completion Time	51599.284	276	186.954			
	Mean Network Utilization	1.966	276	.007			
Total	Goodput	30442256092040.430	280				
	Throughput	163764277451218.200	280				
	Flow Completion Time	2106707.905	280				
	Mean Network Utilization	225.326	280				
Corrected Total	Goodput	1989038435576.215	279				
	Throughput	2819856064202.497	279				
	Flow Completion Time	53002.468	279				
	Mean Network Utilization	4.790	279				

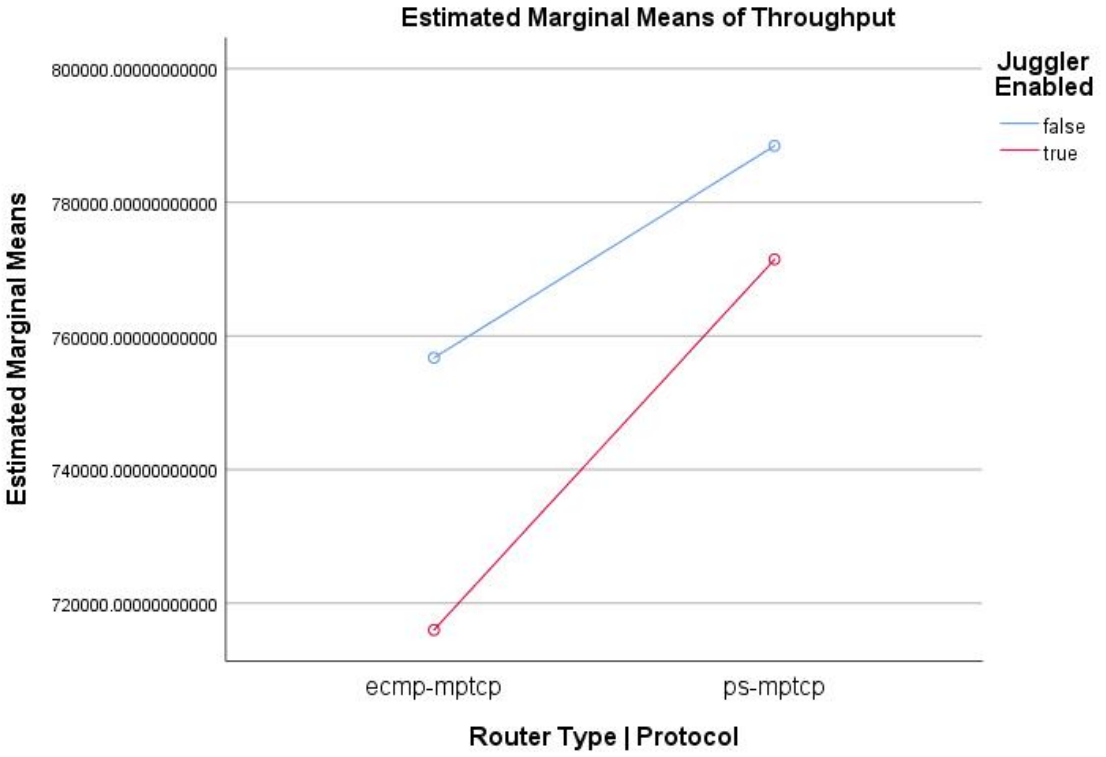
a. R Squared = .013 (Adjusted R Squared = .002)

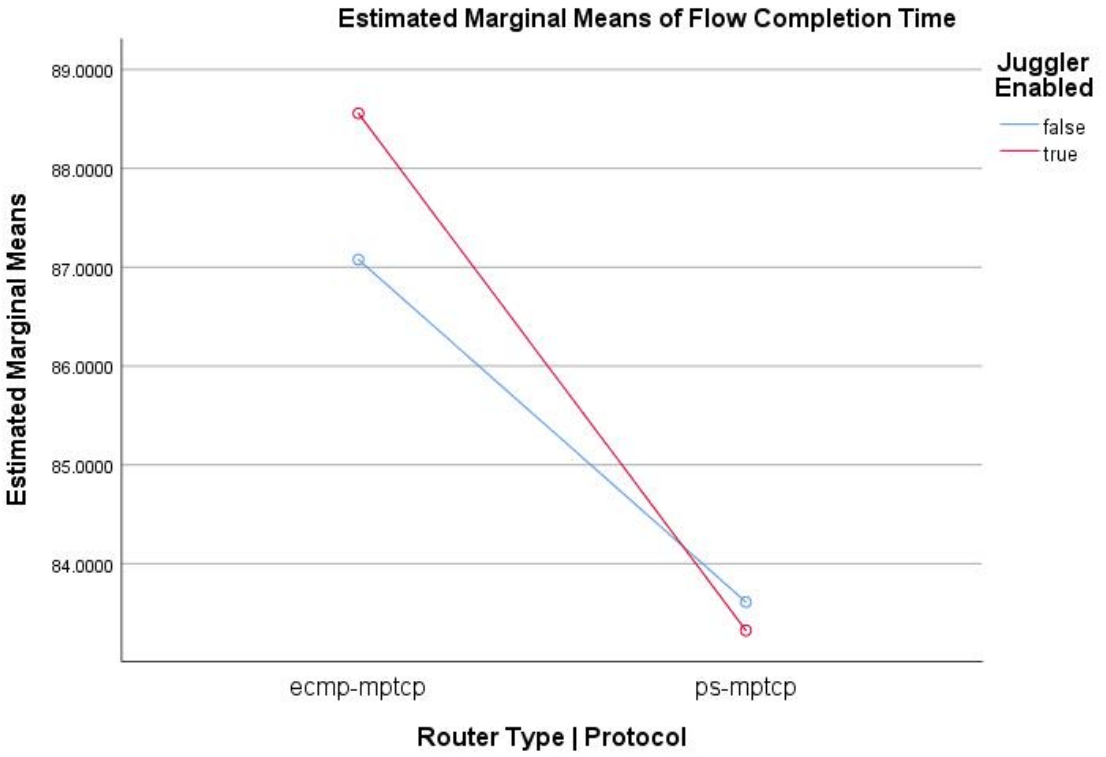
b. R Squared = .071 (Adjusted R Squared = .061)

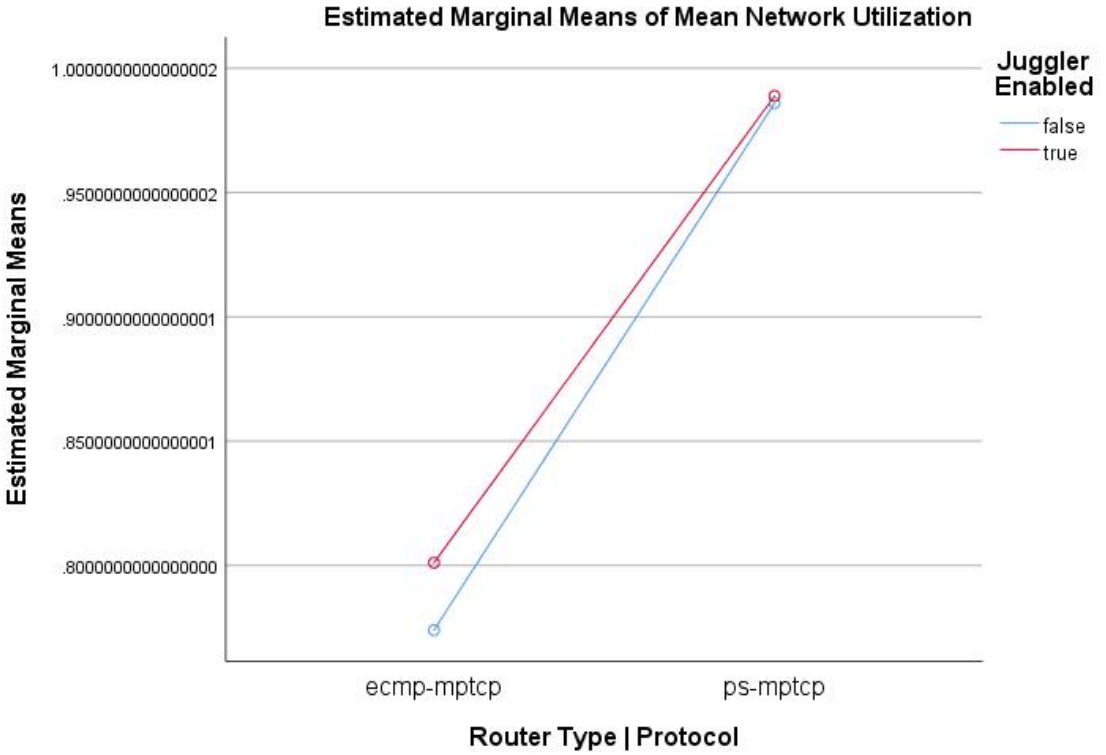
c. R Squared = .026 (Adjusted R Squared = .016)

d. R Squared = .590 (Adjusted R Squared = .585)


Estimated Marginal Means of Goodput
Estimated Marginal Means
335000.00000000000
330000.00000000000
325000.00000000000
320000.00000000000
315000.00000000000
310000.00000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Throughput
Estimated Marginal Means
800000.00000000000
780000.00000000000
760000.00000000000
740000.00000000000
720000.00000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Flow Completion Time
Estimated Marginal Means
89.0000
88.0000
87.0000
86.0000
85.0000
84.0000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true


Estimated Marginal Means of Mean Network Utilization
Estimated Marginal Means
1.0000000000000002
.9500000000000002
.9000000000000001
.8500000000000001
.8000000000000000
ecmp-mptcp
ps-mptcp
Router Type | Protocol
Juggler Enabled
false
true