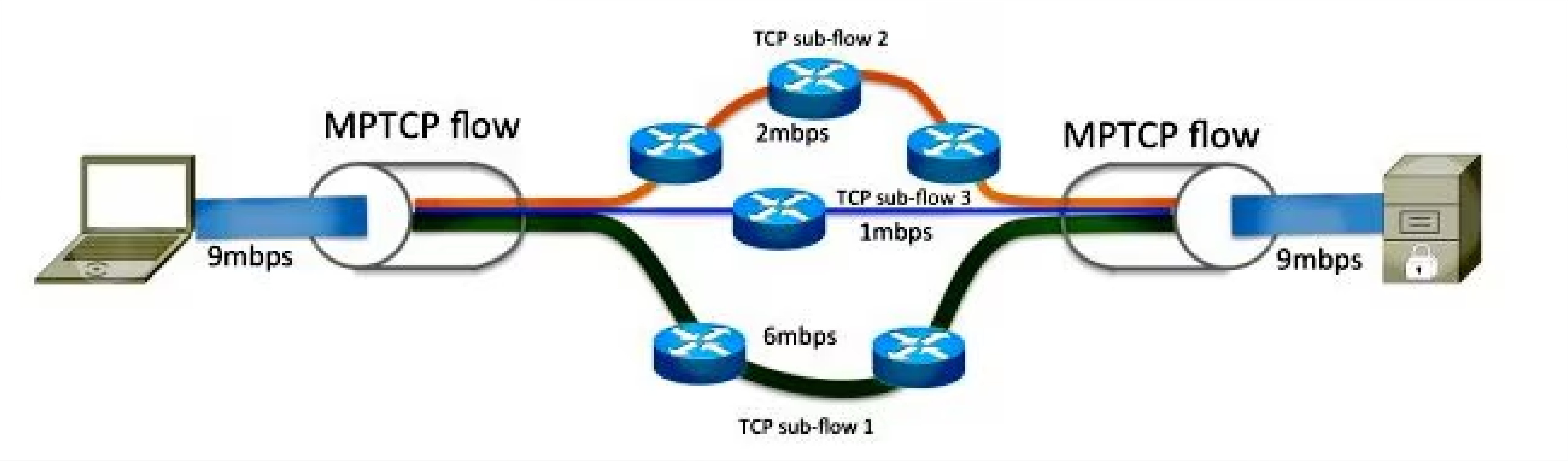
Investigating the Effect of Different Subflow MTUs on MPTCP Throughput

Presented By Shamalka Manorathne

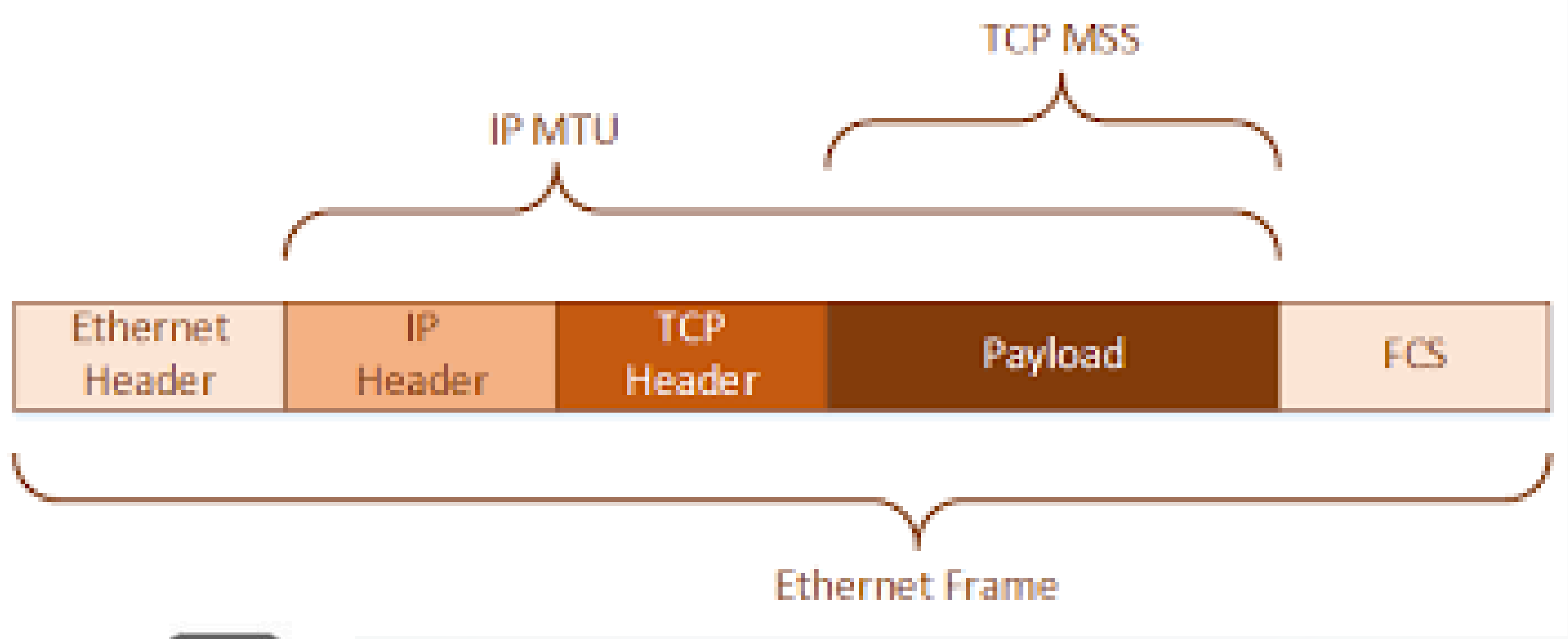
Multipath TCP

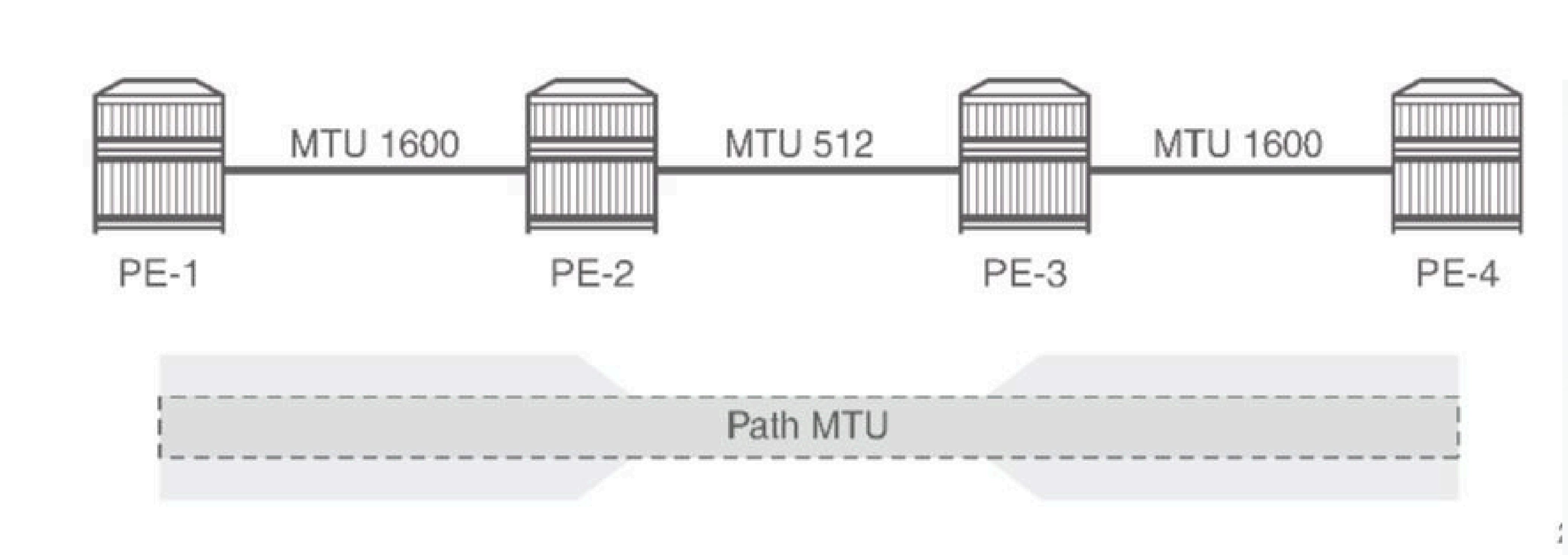


Understanding the Concepts

- Maximum Segment Size (MSS)
- Maximum Transmission Unit (MTU)

  MTU = MSS + IP header + TCP header
- Path MTU
- Path MTU Discovery





Research Gap

- It remains unclear how different path MTU subflows should take into account for scheduling packets across them.
- There is no analysis of throughput in such an environment.
- Current techniques do not emphasize the involvement of managing subflows with different path MTUs and the selection of appropriate MSS values for sending data in each subflow.
- Unlike RTT and bandwidth, the integration of MTU considerations has not been thoroughly explored in the literature.

Research Gap

- If the MSS is same for each segment, fragmentation needs to be done to send data through subflows with both large and small path MTUs.
- Fragmentation Overhead: Fragmentation adds additional processing overhead at both the sender and receiver.
- Suboptimal Resource Utilization: Some paths are to be underutilized if their actual MTU is higher than the used one, leading to inefficiencies in bandwidth usage.

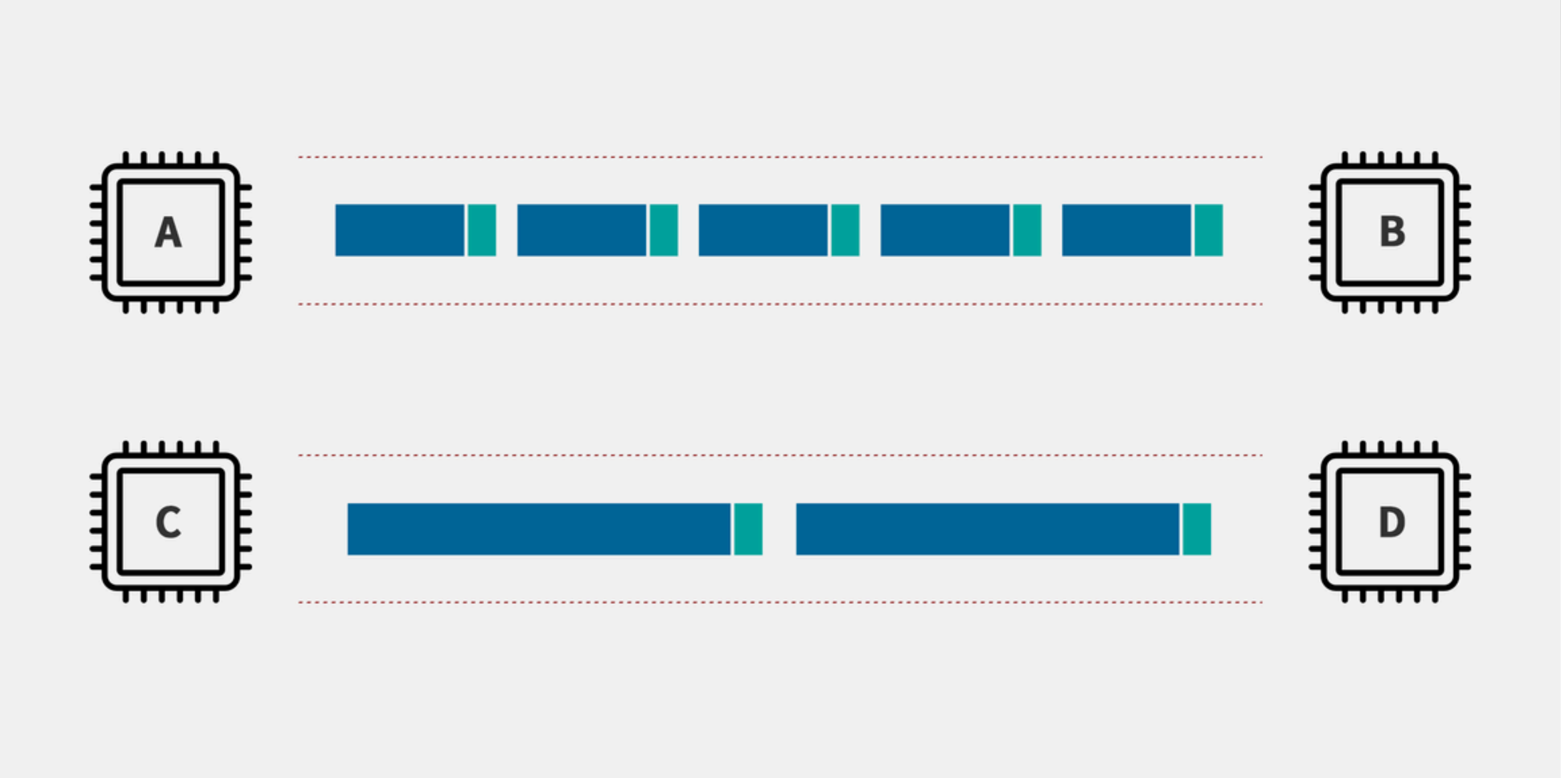
MTU 1500 bytes

IP	TCP	DATA
20	20	1460

Wich MTU size will use ?

MTU 9000 bytes

IP	TCP	DATA
20	20	8960



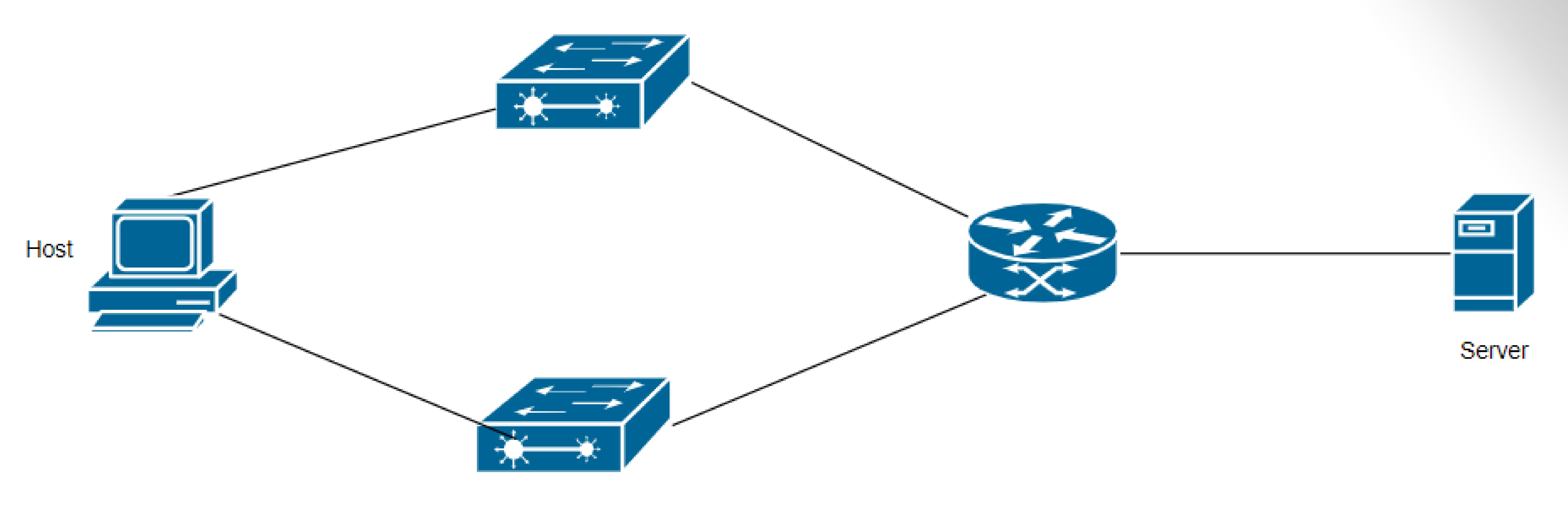
Jambo frames :

- data centers
- cloud environments

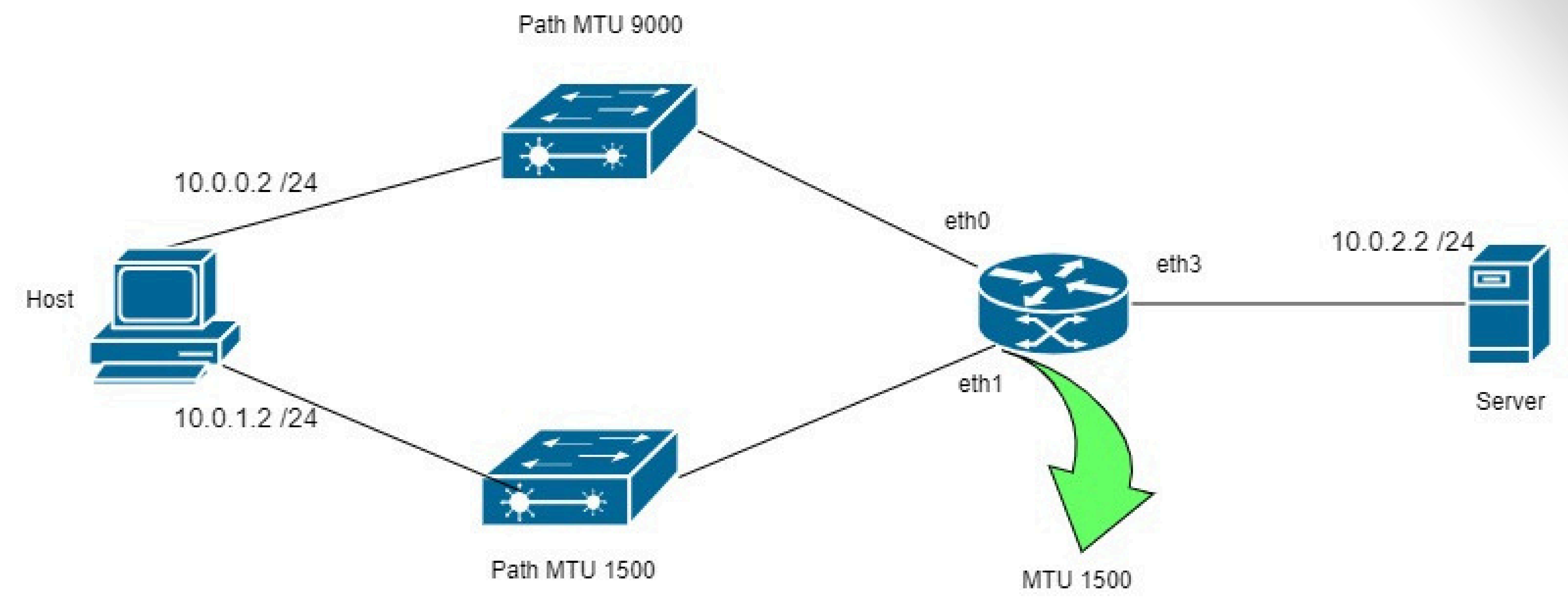
Existing Schedulers for Heterogeneous Networks

Schedular Name	Core Idea
minRTT	Assigns packets to the subflow with the smallest RTT among the subflows that are not CWND-limited.
BLEST (Blocking Estimation)	Optimizes MPTCP send window usage to prevent head-of-line blocking. It may skip sending on slower subflows to ensure faster subflows can send as soon as they are available.
ECF (Earliest completion first)	Assigns packets to the subflow that will deliver them the fastest. It may avoid using slower subflows, waiting instead for faster subflows to become available.
STTF (Shortest Transfer Time First)	Assigns unsent segments to subflows based on the shortest predicted transfer time. This rescheduling occurs at every interruption, such as each ACK arrival.

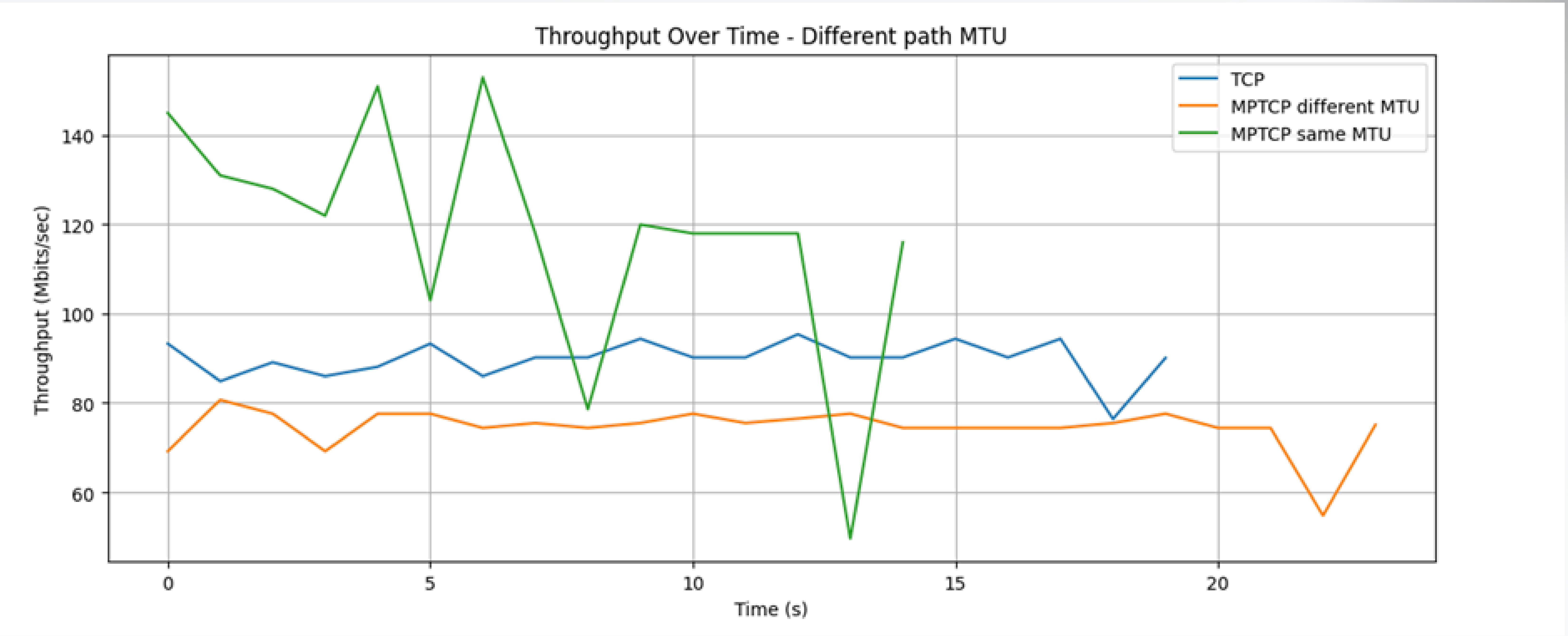
Topology



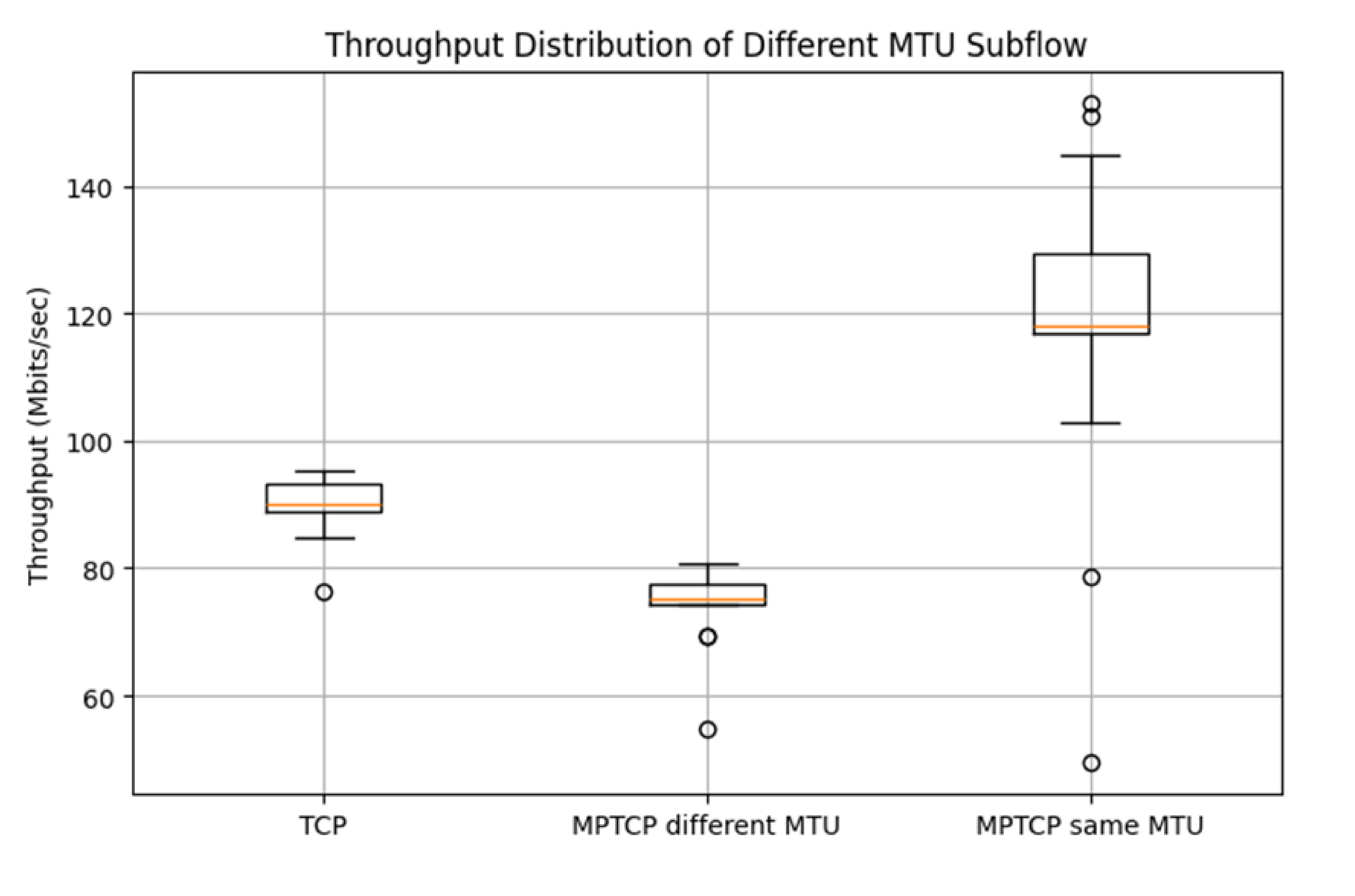
Lab Setup and Experiments



Test Results



Test Results



Multipath TCP
Different MTU

iperf -s -i 1

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.2.2 port 5001 connected with 10.0.0.2 port 55316
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec  7.35 MBytes  61.6 Mbits/sec
[  1] 1.0000-2.0000 sec  9.66 MBytes  81.1 Mbits/sec
[  1] 2.0000-3.0000 sec  9.19 MBytes  77.1 Mbits/sec
[  1] 3.0000-4.0000 sec  8.44 MBytes  70.8 Mbits/sec
[  1] 4.0000-5.0000 sec  9.26 MBytes  77.7 Mbits/sec
[  1] 5.0000-6.0000 sec  9.13 MBytes  76.6 Mbits/sec
[  1] 6.0000-7.0000 sec  9.04 MBytes  75.9 Mbits/sec
[  1] 7.0000-8.0000 sec  8.88 MBytes  74.5 Mbits/sec
[  1] 8.0000-9.0000 sec  8.95 MBytes  75.1 Mbits/sec
[  1] 9.0000-10.0000 sec  9.05 MBytes  75.9 Mbits/sec
[  1] 10.0000-11.0000 sec  8.95 MBytes  75.1 Mbits/sec
[  1] 11.0000-12.0000 sec  9.20 MBytes  77.1 Mbits/sec
[  1] 12.0000-13.0000 sec  9.02 MBytes  75.6 Mbits/sec
[  1] 13.0000-14.0000 sec  9.28 MBytes  77.9 Mbits/sec
[  1] 14.0000-15.0000 sec  9.01 MBytes  75.6 Mbits/sec
[  1] 15.0000-16.0000 sec  8.93 MBytes  74.9 Mbits/sec
[  1] 16.0000-17.0000 sec  8.85 MBytes  74.2 Mbits/sec
[  1] 17.0000-18.0000 sec  8.81 MBytes  73.9 Mbits/sec
[  1] 18.0000-19.0000 sec  9.03 MBytes  75.8 Mbits/sec
[  1] 19.0000-20.0000 sec  9.07 MBytes  76.1 Mbits/sec
[  1] 20.0000-21.0000 sec  8.99 MBytes  75.4 Mbits/sec
[  1] 21.0000-22.0000 sec  8.82 MBytes  74.0 Mbits/sec
[  1] 22.0000-22.3424 sec  3.10 MBytes  75.9 Mbits/sec
[  1] 0.0000-22.3424 sec   200 MBytes  75.1 Mbits/sec
```

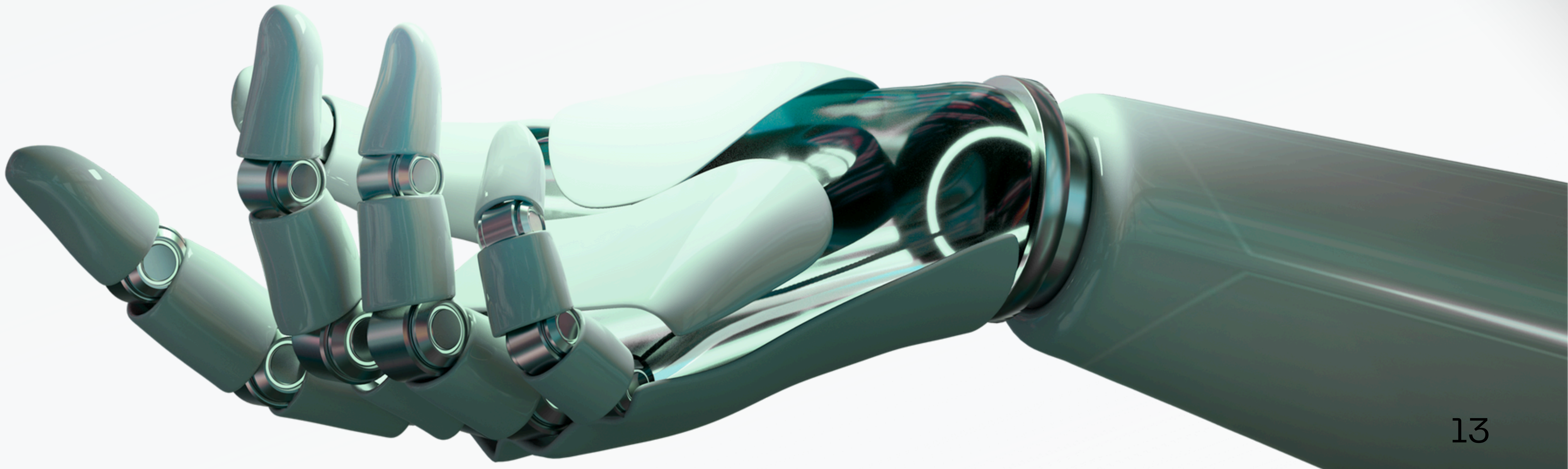
TCP

iperf -s -i 1

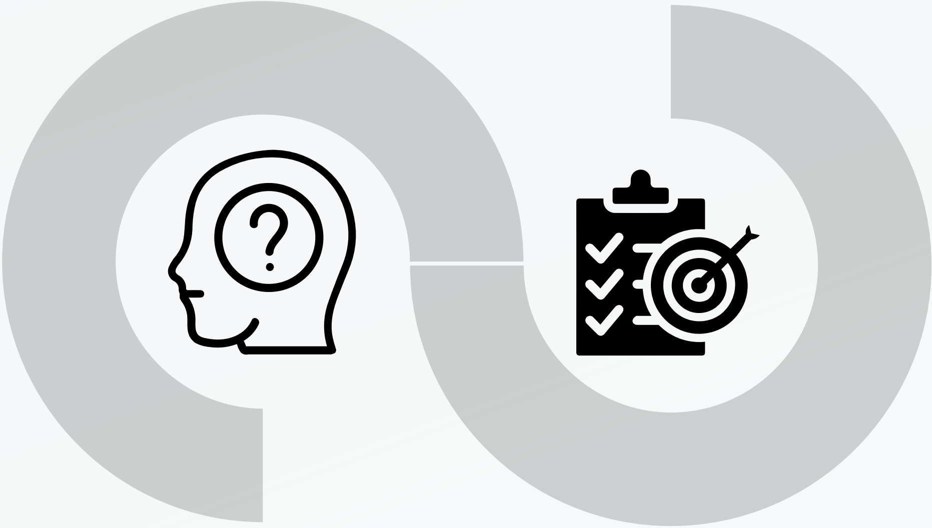
```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.2.2 port 5001 connected with 10.0.0.2 port 48584
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec  10.6 MBytes  88.6 Mbits/sec
[  1] 1.0000-2.0000 sec  10.2 MBytes  85.7 Mbits/sec
[  1] 2.0000-3.0000 sec  10.5 MBytes  88.4 Mbits/sec
[  1] 3.0000-4.0000 sec  10.4 MBytes  87.2 Mbits/sec
[  1] 4.0000-5.0000 sec  10.4 MBytes  87.2 Mbits/sec
[  1] 5.0000-6.0000 sec  10.3 MBytes  86.4 Mbits/sec
[  1] 6.0000-7.0000 sec  10.6 MBytes  88.8 Mbits/sec
[  1] 7.0000-8.0000 sec  10.6 MBytes  89.2 Mbits/sec
[  1] 8.0000-9.0000 sec  11.0 MBytes  92.2 Mbits/sec
[  1] 9.0000-10.0000 sec  11.0 MBytes  92.1 Mbits/sec
[  1] 10.0000-11.0000 sec  10.9 MBytes  91.8 Mbits/sec
[  1] 11.0000-12.0000 sec  10.9 MBytes  91.1 Mbits/sec
[  1] 12.0000-13.0000 sec  11.0 MBytes  92.2 Mbits/sec
[  1] 13.0000-14.0000 sec  10.9 MBytes  91.3 Mbits/sec
[  1] 14.0000-15.0000 sec  10.9 MBytes  91.7 Mbits/sec
[  1] 15.0000-16.0000 sec  11.0 MBytes  92.2 Mbits/sec
[  1] 16.0000-17.0000 sec  11.1 MBytes  92.8 Mbits/sec
[  1] 17.0000-18.0000 sec  11.0 MBytes  92.2 Mbits/sec
[  1] 18.0000-18.6163 sec  6.77 MBytes  92.1 Mbits/sec
[  1] 0.0000-18.6163 sec   200 MBytes  90.1 Mbits/sec
```

Research Aims

Exploring the behavior of Multipath TCP on subflows with different Path MTU values to ensure continuous network connectivity, evaluating the throughput and conducting a comparative analysis with standard TCP.



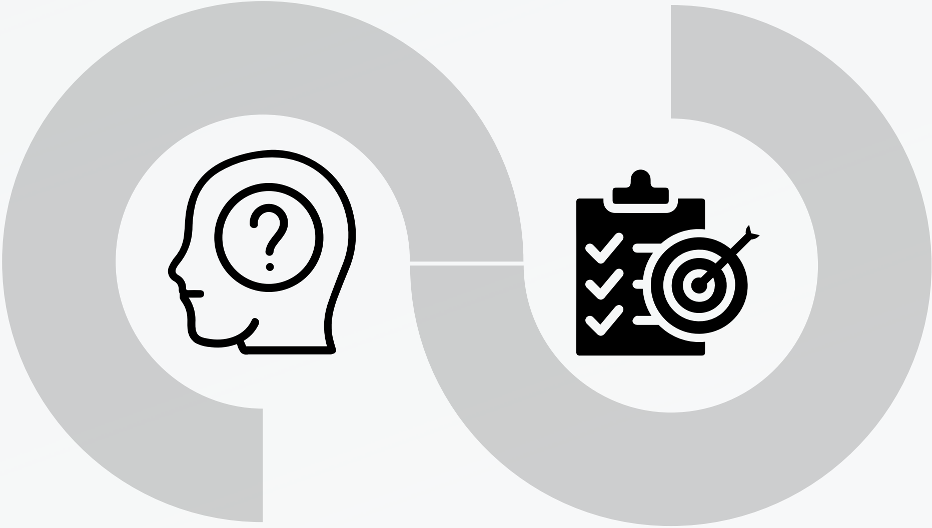
Objectives



- Analyze the behaviour of MPTCP on subflows with different path MTU values in current implementations using emulation.
- Investigate the variations in path MTU across different subflows and how these differences impact MPTCP performance.
- Evaluate the throughput of MPTCP when operating over subflows with different path MTU values compared to standard TCP.
- Analyze the impact of fragmentation on MPTCP performance when subflows have differing path MTUs.

Objectives

- Investigate how the MPTCP scheduler handles segments of different sizes and its effect on overall network performance and connectivity.
- Analyze and identify any performance bottlenecks/inefficiencies in MPTCP with different path MTU subflows.
- Suggest an improved technique(s) to enhance the MPTCP performance with different path MTU values.



Research Questions

How does Multipath TCP adjust to subflows with different Path MTU values compared to standard TCP?

Additionally, what impact does fragmentation overhead have on subflow packet scheduling in the context of different Path MTU?

How can we improve subflow packet scheduling In MPTCP to handle different Path MTU values effectively?

Research Methods

- Setup Testing Environment
- Run Tests Iterativly and Analyze the Results
- Prototype Model Implementation
- Iterative Improvement Based on Findings
- Performance Evaluation

Evaluation plan

STEP 01

Sending traffic over different path MTU subflows

STEP 02

Data Collection

STEP 03

Analyze Throughput

STEP 04

Improve the scheduler

STEP 05

Performance Evaluation

Scope

In Scope

- Analyze the existing MPTCP scheduling algorithms.
- Evaluate and contrast the efficiency of Multipath TCP with different path MTU subflows and standard TCP.
- Conduct emulations and real-world tests to model different Path MTU configurations.
- A detailed evaluation of MPTCP's performance across subflows with different path MTU.
- Test Using the default scheduling algorithm and related implementations.
- Experiments and all related works are done on Linux.

Scope

Out Scope

- This research will not thoroughly explore other aspects that affect scheduling, unless they are directly related to MTU challenges.
- This research will not focus on developing new congestion control algorithms.
- Improving each and every possible scheduling algorithm.
- Not address the specific security or encryption issues related to MPTCP.

Availability of the resources

- Linux
- MPTCP Kernel
- MPTCP documentation - RFC 8684
- Mininet emulation tool
- Wireshark

THANK YOU VERY MUCH