International Journal of Information Communication Technologies and Human Development
Volume 14 • Issue 1

Exploring Multipath TCP Schedulers in Heterogeneous Networks

Vidya Sachin Kubde, Datta Meghe College of Engineering, India*
Sudhir Sawarkar, Datta Meghe College of Engineering, India

ABSTRACT

Multipath transmission control protocol (MPTCP) is a transport layer protocol that transmits TCP segments on more than one path in multihomed devices. It was designed with the aim of bandwidth aggregation and redundant connections. Currently, multihomed devices have wireless interfaces of heterogeneous nature. MPTCP is not able to give its optimal performance in heterogeneous networks. This paper presents an experimental performance study of four different schedulers, namely roundrobin, default, blest, and redundant. The testbed comprises ethernet, LTE, and wifi networks to connect multihomed devices. The authores have compared the scheduler performance in terms of throughput, download time, and path utilization rate in homogenous and heterogenous scenarios. Results showed that round robin provides optimal throughput in homogenous networks and also performs bandwidth aggregation by utilizing both the paths but fails to perform in heterogenous networks. Blest provides best throughput among the four schedulers but prefers fast path only.

KEYWORDS

MPTCP, Path Utilization, Scheduler, TCP

INTRODUCTION

Multipath TCP is (MPTCP) an extension of TCP, evolved for today ’s multihomed devices. MPTCP is an ongoing effort of the IETF’s Multipath TCP working group Ford et al. (2012). The TCP is a widely used single path protocol, if that fails for any reason then the connection has to be reestablished. MPTCP on the other hand establishes a single connection with all the available interfaces, to deal with the network failures. MPTCP also benefits resource utilization, and bandwidth aggregation Paasch and Barre (2014). As of today, the Linux Kernel MPTCP implementation Apple (2017) is one of the most widely used MPTCP implementations besides Apple’s implementation for the cloud-based assistant system Siri Postel (1981).

TCP is mainly designed for wireline networks. MPTCP is built over TCP and is designed for smart home devices, which mostly use wireless networks. A smartphone is having two wireless interfaces, WIFI, and Cellular networks. Thus, MPTCP has to deal with the wireless channel impairments. This is because packet loss, network delay, roundtrip time variation is very probable in wireless networks. Indeed, the MPTCP have to deal with more than one wireless networks with different network charac-

DOI: 10.4018/IJICTHD.295857 *Corresponding Author teristic. Thus, to aggregate, the throughput of the multiple paths in MPTCP of different characteristics is a challenging issue.

The performance of MPTCP depends upon packet Scheduler. A scheduler assigns the packets to the available paths. A wrong scheduling decision leads to decrease in performance of MPTCP in both the heterogenous networks and homogenous networks such as decrease in throughput, higher download time, poor path utilization. Heterogeneity of paths leads to increase in out of order packets which in turn causes Head of Line (HOL) blocking issue, due to receiver window limitation. An optimized packet scheduler, will use all the available paths, will reduce out of order packets in order to increase throughput and decrease download time.

Hence, our study aims to experimentally verify the behavior of MPTCP schedulers in different wire- less networks. We have examined MPTCP scheduler performance concerning Through- put, Download time, and Path utilization rate. The testbed comprises of two scenarios homogenous (WIFI WIFI) and heterogeneous (WIFI- LTE and WIFI -Ethernet). We have analyzed that Round robin per- forms best in homogenous networks by utilizing all the available paths, but is unable to perform in heterogenous net- works. Blest, is able to perform with heterogenous networks by preferring fast paths only which reduces out of order packets.

This paper makes the following contributions. First the experiment study for the comparison be- tween the throughput of MPTCP schedulers. Secondly, we have observed the MPTCP schedulers' download time for different file sizes. Lastly, we have analyzed the effect of scheduling policies on path utilization rate.

The rest of the paper is organized as follows, we provide a brief introduction, and the most relevant research work in the literature is provided in Section 2. Section 3 provides details about Experimental setup and study Finally we conclude our analysis in section 4.

BACKGROUND & MOTIVATION

In single path TCP if packets are not lost or not retransmitted, then they will arrive in order Paasch (2014). In MPTCP as packets are going to traverse through multiple paths, with different characteristics causing out of order at the receiver end Hurtig (2018). This results in Head of Line Blocking impacting the throughput.

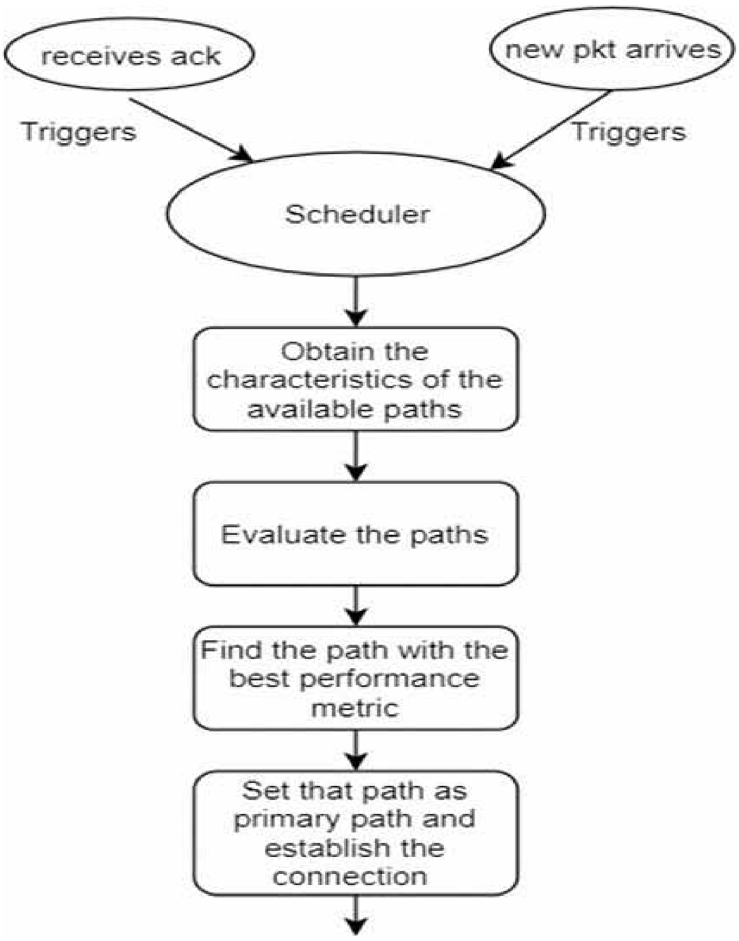
Analysis of Existing MPTCP Scheduling Algorithms

The MPTCP architecture is introduced in RFC Ford et al. (2012). Over the past years, there has been a lot of research on MPTCP implementation, design, and performance issues. The scheduler deals with the se- lection of paths, to increase the throughput compared to single path TCP. Fig 1, depicts the scheduling process, the scheduler is invoked, either when a new packet has arrived from the application layer or acknowledgment is received. The scheduler will acquire the path characteristics, round trip time (RTT), signal strength, and through- put, loss rate. The transmission performance of the paths is evaluated with these parameters. The best path is selected to establish the connection.

A wealth of research has been done to resolve the unsolved issues of packet scheduling in MPTCP. Some works implement Round Robin Hwang and Yoo (2015). This scheduler selects the paths one after another in turn. It could not perform in heterogeneous networks. To face the heterogeneity, MinRTT was evolved which selects the path with the lowest RTT. MinRTT Raiciu (2012), is the default scheduler of MPTCP to date. The amount of data on this selected path is decided by its congestion window. This scheduler worked well except on memory constrained devices that use a small receive window This problem was identified by Costin Raiciu Yang et al. (2014) to avoid the head-of-line blocking issue caused by a limited receiver window. Likewise, DAPS Lim (2017) replaced RTT by the forward delay that is sending time plus inflight time to estimate the time taken by the packet to reach the destination. Although the work added more precision, this delay aware scheduler is advantageous only when the values of RTT and congestion window remains stable for the whole duration of the schedule. Thus, DAPS is not able to deal with network failures.

Further, in a study by yang Ferlin (2016), the authors propose an OTIAS algorithm that tries to resolve this issue by using current data. Its work is based on the idea of scheduling more segments on a sub- flow than what it can currently send. Lim in his work ECF Oh and Lee (2015) believes that the fast path is not utilized to its full extent. It aims to minimize the periods where a fast sub-flow becomes idle. Blest [13] worked with this same idea that instead of using slow paths, It prioritizes only fast paths. CP [14] contributes by, blocking the slow path. If the slow path is causing performance issues then CP prefers to block that path. In STTF Hwang and Yoo (2015) hurting believes that in default scheduler unavailable paths are sometimes a better choice. The idea behind STTF is very simple; for each segment to schedule, calculate its transmission time considering data already in flight. STMS Shi et al. (2018) deals with out-of-order sending of packets for in order receiving

Figure 1. Working of MPTCP scheduler


receives ack
new pkt arrives
Triggers
Triggers
Scheduler
Obtain the characteristics of the available paths
Evaluate the paths
Find the path with the best performance metric
Set that path as primary path and establish the connection

Alternative Scheduling Decisions for Multipath TCP Kimura et al. (2017) pro- posed policies for different types of applications sorting the paths, one of the policies deals with sending rate of the flows, the other policy uses the highest available space in the congestion window. In Qaware

Sreedhar (2018) was motivated by the fact that the particular sub-flow which is used more frequently tends to increase its end- to- end delay gradually, making it less attractive to use. They have used end-to-end delay means local device driver queue occupancy plus end-to-end delay measurements for path selection.

Remp Frogmen (2016) is a different category of the schedulers, where all the paths are used but the same data is transferred on both the paths. It does not enhance the throughput but only is useful in latency-sensitive applications.

DEMS Guo and Ethan (2017) believes that by strategically planning the scheduling one can reduce the download time. It achieves this by decoupling both the paths mean sending the data in a forward direction on one path and in a reverse direction on the other path, it performs data reinjection, to a very small extent to reduce the download time. A detailed analysis of the schedulers is described in table1

MinRTT (Default Scheduler)

The MinRTT is the default scheduler, which selects the path with lowest RTT and sends the packet on it until its congestion window becomes full. Then it selects the path with the next higher RTT. The MinRTT algorithm has a load balancing effect by putting more data on the high-quality sub flow. This algorithm does not consider packet ordering.

Suppose there are two sub flows Rf and Rs having RTT of 5ms and 10ms respectively. Both the paths have same congestion win- dow size 5. The time required by slow sub flow is double the time required by fast sub flow. When there are 6 packets to transfer. Ac- cording to algorithm packets 1-5 will be sent on fast path and packet 6 will be sent on slow path. The receiver has to wait for packets of slow path for long time. Thus, the performance of the algorithm depends upon the path difference.

Blest

The Blest algorithm improves application performance by reducing out-of-order packets in heterogeneous networks. The algorithm estimates whether Head of Line Blocking problem will occur. First it finds the number of packets that can be sent on fast sub flow with- out Head of Line blocking. If the RTT difference between the two paths is very large, then the packets sent on the slow sub flow arrive relatively late. This leads to out-of-order packets at the receiver.

Consider, there are two sub flows Rf and Rs having round trip times of RTTf and RTTs. If slow sub flow is selected for packet transmission, the algorithm assumes that one segment will occupy space of one RTTs in the MPTCP send window. The Blest estimates the packet X that can be sent on fast sub flow without Head of Line Blocking during slow RTTs as

$rss = RTTs/RTTf$

$X = MSSf \cdot (CWND + (rtts - 1)/2) \cdot rtts$

Inaccuracy of X is adjusted by λ value. If $X + \lambda > |M|$-MSSs. (in flight+1), then the next segment will not be sent on slow sub flow. The scheduler prefers to wait for fast sub flow, in order to reduce Head of Line Blocking caused by out-of-order packets at the receiver side. Blest outperforms in heterogeneous networks however it can’t efficiently utilize all the available paths.

EXPERIMENTS IN CONTROLED LAB SETUP

In this section, we have studied the performance of four MPTCP schedulers based on Linux Kernel platform with test bed shown in fig 2 under different RTT, Bandwidth and file size to analyze the performance of each scheduler under different network scenarios.

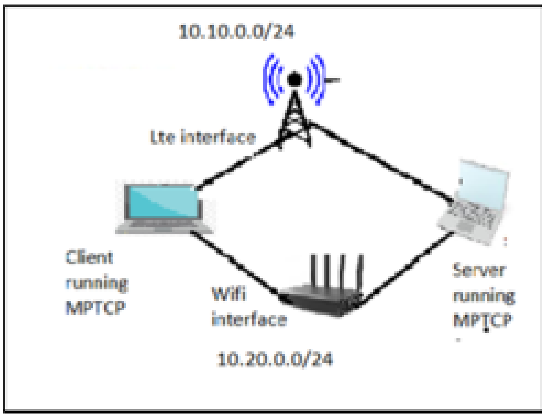
Testbed Setup

In this section we have studied the performance of MPTCP by manually setting RTT and Bandwidth parameters in our controlled environment. The test bed as shown in fig 2 consists of MPTCP client

Table 1. Scheduler goals and path utilization with constraints

Scheduler Goals		Path Utilization	Constraints
Round Robin [6]	Optimal load balancing	Simultaneous use of all the paths	Unknown fairness
Min-RTT [6]	To increase the throughput	One path at a time	It cannot work with asymmetric paths.
DAPS [10]	Tries to maximize the probability of in-order arrival	Simultaneous Path Utilization	Cannot deal with network failures. It builds schedule runs being unable to react to network changes. It does not apply schedule reinjections (retransmissions)
ECF [12]	Aims to minimize completion time	Prefers fast paths	Can be hurt by head-of-line blocking
OTIAS [11]	Schedules data to minimize transmission time	Simultaneous Path Utilization	1. It assumes symmetric forward delays (OWD = RTT/2) 2. It does not apply schedule reinjections (retransmissions)
DEMS [16]	Reduced download time	Simultaneous use of all the paths	Rely on exact knowledge of data chunk boundary for efficient scheduling
CP [14]	Out of order packets	Prefers fast paths	Requires network assistance
STMS [15]	Out of order packets	Simultaneous use of all the paths	RTT error sensitivity
Qaware [17]	Limits the HOL issue	Simultaneous use of all the paths	Requires network assistance
Blest [13]	Aims to reduce head-of-line blocking	Mostly fast paths	RTT error sensitivity

Figure 2. System setup



(Dell laptop equipped with Intel® core TM i-7 4790 CPU @ 3.6 GHz and 20 GB of memory connected to MPTCP server (Intel core i7-7700 processor @ 3.6 GHz and paired with 64 GB of DDR4 RAM. Both are MPTCP enabled with Linux Kernel and MTCP version of 4.19.105, mptcpv9 respectively. Three networks Ethernet, LTE and Wi-Fi are used for concurrent transmission. For traffic generation iperf3 client is used. Constant bit rate traffic is generated for 200 sec iterative for each scheduler respectively.

Methodology

The experiments are performed in controlled environment, in order to generate baseline results, which can be used as a reference for real world experiments. We have established MPTCP connection between the client and server, were in the server uses iperf3 to generate data traffic. The experiment is repeated 50 times for each scheduler and throughput is calculated with respect to different network scenarios.

For switching between TCP and MPTCP “MPTCPenable=0” and “MPTCPenable=1” is used respectively. Configuration of the schedulers is done using command “Scheduler=default/round robin/redundant/blest. For setting the network parameters like RTT and Bandwidth shown in table 2 **“**tcqdisc**”** is used. Next to study the effect of web traffic on MPTCP performance, we choose various file sizes like 64k, 128k, 256k, 1MB, 256MB etc. Basic command used for traffic generation at sender is: “iperf3 -c 10.30.3.13-n 64K -i”

“ss” is the tool used to generate live data set of connection established between sender to receiver. It gives connection’s different interface wise information, including receiver buffer size, sender buffer size, congestion window, average round trip time, byte acknowledged, retransmission time out, sender ip address, receiver ip address, unacknowledged packets and many more. This information is grabbed for every sec by running ss command in python script to generate csv file. “ss” command is executed as “ss - aiet4nm”. Generated data file of csv is analytically observed by using “R” Tool.

Evaluation

In this section, we investigated the impact of paths with different RTT’s over MPTCP performance.

MPTCP Performance comparison in different Network Scenarios

To study the performance of MPTCP scheduler with respect to different networks, three sets of experiments as shown in table 2 are conducted for each scheduler i) WIFI-Ethernet ii) Wi-Fi- LTE iii) WIFI-WIFI

Table 2. Network specifications

Interface	Rate	RTT
LTE	16 mbps	70ms
Wi-Fi	15mps	60ms
Ethernet	80mbps	10ms

It was noticed that as indicated in fig 3 WIFI-WIFI scenario Round robin performed the best having a throughput boost of 50 percent above other schedulers. The working of Round robin indicates that it uses all the paths one after the other, as both the paths are of equal characteristics, therefore packets over both the paths will reach in order at the receiver, resulting in good throughput performance. The Blest and Default preferred fast path only so bandwidth aggregation was not accomplished

Figure 3. Throughput comparison of schedulers in homogenous networks (WiFi-WiFi) |

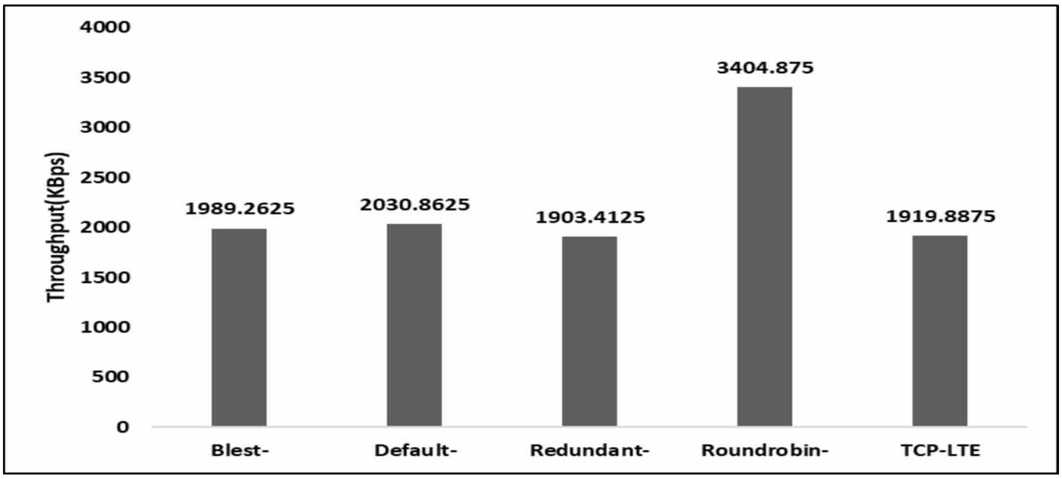
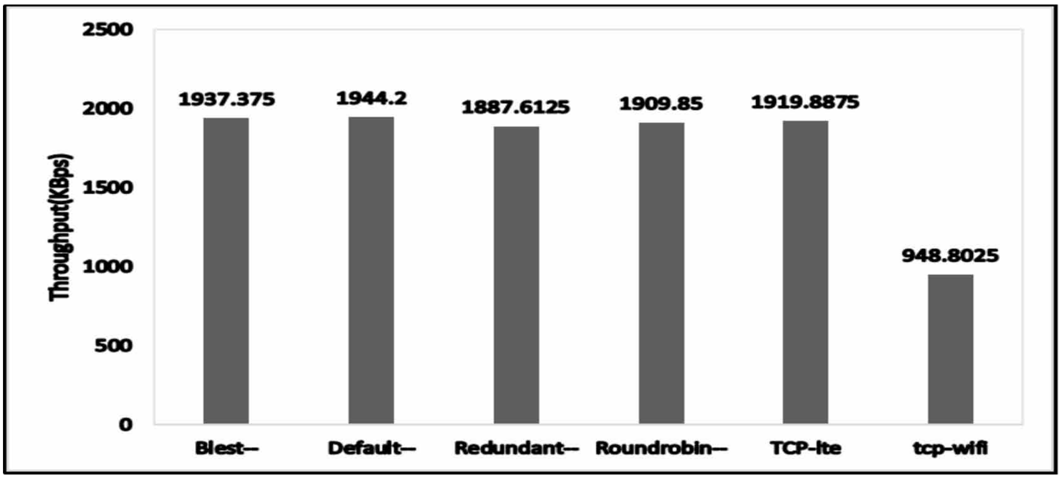


Figure 4. Throughput comparison of schedulers in heterogeneous networks (WIFI-LTE)



In fig 4, as there is minor variance in network characteristics, here all the schedulers are functioning nearly identical. In the heterogeneous networks presented in fig 5, only Blest was able to execute. As the packets travelling on heterogeneous pathways that is Ethernet and Wi-Fi will reach the receiver at different time producing out-of-order packets resulting in Head of Line Blocking. Thus, scheduler like Blest which handles with out-of-order packets, by prioritising fast pathways exclusively was able to execute over here. Blest gives an 8 percent boost over Default and 35 percent over Redundant and 50 percent over Round robin.Round robin employed both the paths, therefore the packets on Ethernet reached the receiver relatively fast than Wi-Fi path creating out-of-order packets resulting into lower performance.

Path Utilization in WIFI-Eth and WIFI-Lte scenarios

As shown in Fig 6-7 Default and Blest scheduler have used WIFI network for less than 10 percent in both the situations WIFI -Ethernet and WIFI -LTE. This is because these schedulers give higher

Figure 5. Throughput comparison of schedulers in heterogeneous networks (WIFI-Eth)

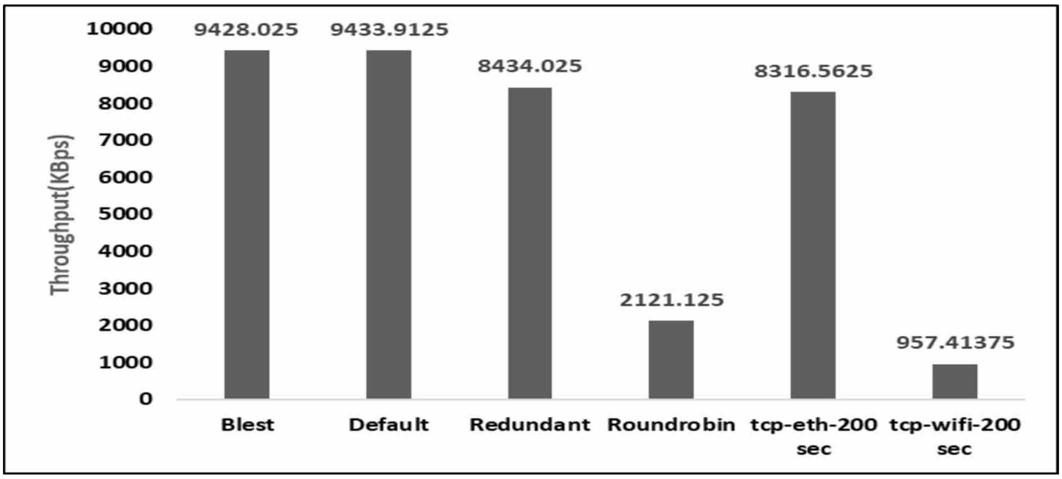


Figure 6. Path utilization in Wi-Fi-eth scenario

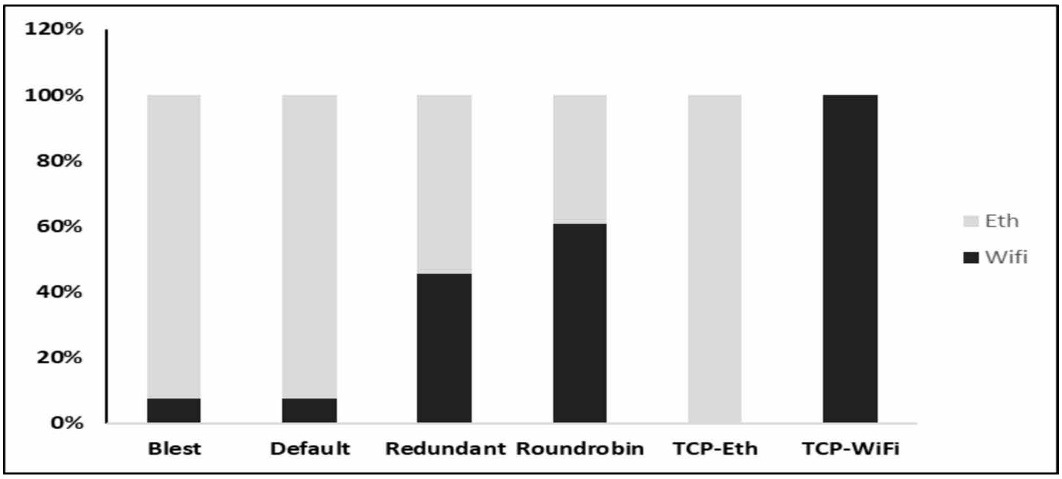


Table 3. traffic distribution in Wi-Fi-eth network

Schedulers	WIFI	Eth
Blest	7%	93%
Default	7%	93%
Redundant	46%	54%
Round robin	61%	39%
TCP-Eth	0%	100%
TCP-WIFI	100%	0%

Table 4. Traffic distribution in LTe-WiFi network

Schedulers	Lte	WiFi
Blest	93%	7%
Default	93%	7%
Redundant	50%	50%
Round robin	48%	52%
TCP LTE	100%	0%
TCP Wifi	0%	100%

Figure 7. Path utilization in WIFI-LTE scenario

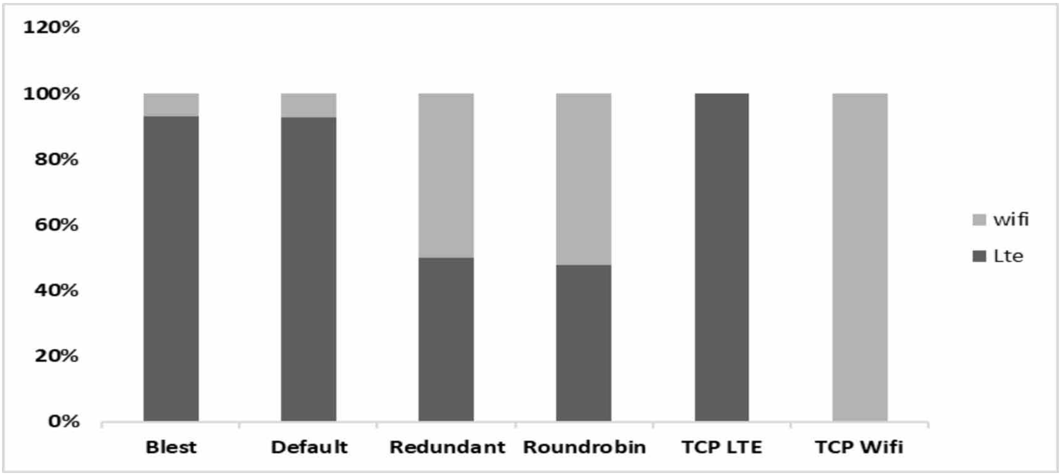
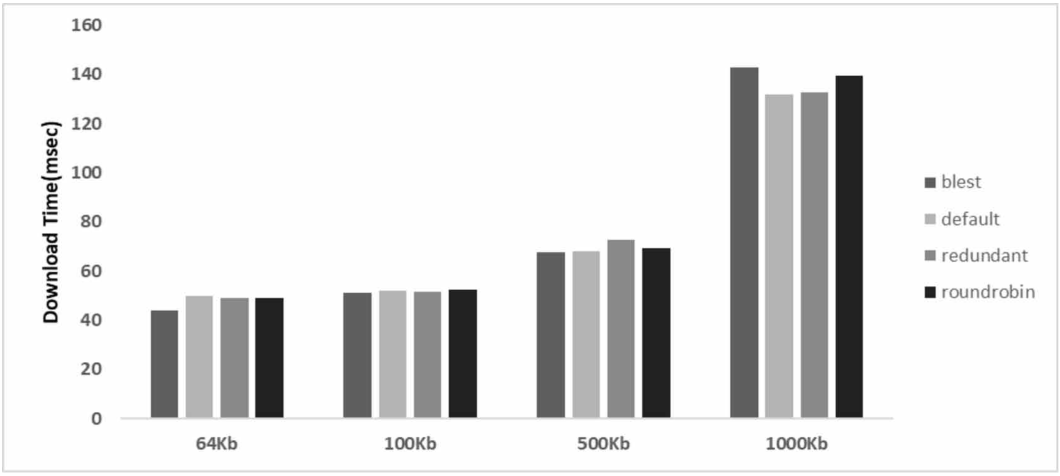


Figure 8. Download time of schedulers with different data volumes



priority to the path with lower RTT. In the case of Round robin and Redundant both the networks contributed equally.

Table 5. Download time of schedulers with respect to different data sizes

Data	Blest	Default	Redundant	Round robin
64Kb	44.158(ms)	49.7615 (ms)	49.0847 (ms)	49.2624 (ms)
100Kb	51.305 (ms)	52.1023 (ms)	51.657 (ms)	52.4682 (ms)
500Kb	67.646 (ms)	68.2763 (ms)	72.5718 (ms)	69.2926 (ms)
1000Kb	142.71 (ms)	131.795(ms)	132.6734 (ms)	139.5432 (ms)

MPTCP Scheduler Performance Over Different File Size:

Fig 8 indicates that for 64 Kb of data, Blest gives a 29 percent boost in download time, over other Schedulers. For 200Kb and 500Kb data segments, Blest gives 5 percent rapid download time as com- pared to other three schedulers. In 1000Kb of data, transfer results are reversed round robin delivers 7 percent speedier download time compared with Blest. For tiny data volumes (64Kb) practically all the schedulers behave the same as there is no need for a second sub-flow before the second connection is established the file transfer is done. Since the data volume goes above 100Kb the performance of the schedulers choosing fast paths will be higher as Blest and Default will select Ethernet first, the data transfer will conclude with fast path only. On the contrary Round robin employs both the pathways, hence demands significant download time. As the data size increases that is 1000Kb Blest have to use sluggish pathways generating excessive download times. This behaviour suggests that Blest can execute for small data sizes, up to 1000 Kb.

Findings and Analysis

In these tests, we have evaluated four schedulers behavior in diverse network conditions. The inves- tigation was done in terms of throughput, download time, and path utilization rate. It was observed that Round robin performed the best in homogeneous networks but failed in heterogeneous networks. On the other side, Blest and Default have outperformed in heterogeneous networks. Blest and Default employs slow path, when the fast path is not available. Thus, Bandwidth aggregation is not achieved in these schedulers, as they choose fast path exclusively. In terms of download time, it was noticed that for small data volumes performance of all the schedulers is nearly equal. For huge data volumes, in WIFI-LTE situation Blest adds additional delay while waiting for faster sub-flows, this results to increase in download time.

CONCLUSION

In this study we have investigated the available MPTCP schedulers and outline their properties. Then we have analysed the behaviour of four widely deployed schedulers over diverse network circumstances in real Linux platform. Path heterogeneity is the most impacting parameter in MPTCP performance. Mostly to deal with out-of-order packets, schedulers employ fast pathways alone, this does not achieve MPTCP's purpose of Bandwidth aggregation. In the future, we want to present a Scheduler which utilises all the pathways and also deals with out-of-order packets in heterogeneous networks.

REFERENCES

Apple. (2017). Advances in networking. *Proc. Worldwide Developers (WWDC).*

Ferlin, S. (2016). BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks. *IFIP Networking Conference (IFIP Networking) and Workshops*. doi:10.1109/IFIPNetworking.2016.7497206

Ford, A., Raiciu, C., Handley, M., Bonaventure, O. M., Handley, C., & Raiciu. (2012). *TCP Exten- sions for Multipath Operation with Multiple Addresses.* IETF.

Frogmen, A. (2016). ReMP TCP: Low latency multipath TCP. *IEEE International Conference on Communications (ICC).*

Guo, Y. & Ethan. (2017). DEMS: Decoupled multipath scheduler for accelerating multipath transport. *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. doi:10.1145/3117811.3119869

Hurtig, P., Grinnemo, K.-J., Brunstrom, A., Ferlin, S., Alay, O., & Kuhn, N. (2018). Low-latency scheduling in MPTCP. *IEEE/ACM Transactions on Networking*, *27*(1), 302–315. doi:10.1109/TNET.2018.2884791

Hwang, J., & Yoo, J. (2015). Packet scheduling for multipath TCP. *Seventh International Conference on Ubiquitous and Future Networks*. doi:10.1109/ICUFN.2015.7182529

Kimura, B. Y. L., Lima, D. C. S. F., & Loureiro, A. A. F. (2017). Alternative Scheduling Decisions for Multipath TCP. *IEEE Communications Letters*, *21*(11), 2412–2415. doi:10.1109/LCOMM.2017.2740918

Lim, Y. S. (2017). ECF: An MPTCP path scheduler to manage heterogeneous paths. *Proceedings of the 13th International Conference on emerging Networking Experiments and Technologies*. doi:10.1145/3143361.3143376

Oh, B. H., & Lee, J. (2015). Constraint-based proactive scheduling for MPTCP in wireless networks. *Computer Networks*, *91*, 548–563. doi:10.1016/j.comnet.2015.09.002

Paasch, C. (2014). Experimental evaluation of multipath TCP schedulers. *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*. doi:10.1145/2630088.2631977

Raiciu, C. (2012). How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. *9th USENIX Symposium on Networked Systems Design and Implementation*.

Shi, H., Shi, H., Cui, Y., Wang, X., Hu, Y., Dai, M., Wang, F., & Zheng, K. (2018). STMS: Improving MPTCP throughput under heterogeneous networks. *USENIX Annual Technical Conference*, 719–730.

Sreedhar, T. (2018). QAware: A cross-layer approach to MPTCP scheduling. *IFIP Networking Conference (IFIP Networking) and Workshops*. doi:10.23919/IFIPNetworking.2018.8696843

Yang, F., Wang, Q., & Amer, P. D. (2014). Out-of-order transmission for in-order arrival scheduling for multipath TCP. *28th International Conference on Advanced Information Networking and Applications Workshops.* doi:10.1109/WAINA.2014.122