Multipath TCP: A Comparative Analysis for Linux kernel implementation

Agneev Ghosh, Jayant Malani, Heli Utpal Modi
University of California- San Diego
{a4ghosh,jmalani,hmodi}@eng.ucsd.edu

ABSTRACT

As the name suggests, Multipath TCP is an extension to the existing TCP protocol, having the added capability of allowing network data to be distributed across multiple existing network connections. Data is transmitted over the aforementioned network paths in a simultaneous fashion, ensuring complete utilization of the available bandwidth, thereby rendering the user network interface reliable. Currently MPTCP implementation in Linux Kernel is in experimental stage. An in depth hardware validation of MPTCP protocol was thus conducted to verify the implementation with respect to its proposed design goals. Furthermore, a comprehensive analysis was conducted to verify its performance with respect to TCP, UDP protocols under different congestion control algorithms and vacillating real time scenarios.

1. INTRODUCTION

Proliferation of multi-homed hosts render the concept of MPTCP as extremely imperative and riveting. Currently this proposed protocol is in the experimental stage thus requiring extensive research and commercial implementations in the same. Hence, we have been motivated to develop a hardware framework incorporating a comparative study of MPTCP protocol with respect to TCP and UDP protocols using existing MPTCP principles for Linux kernel implementation. We were also motivated to validate the current Linux Kernel MPTCP implementation with design goals proposed in [2].

Through our experimental efforts we have been able to study and analyze the usability and efficaciousness of MPTCP under vacillating conditions, thus validating the circumstances under which the use of MPTCP protocol ensures better utilization of the total available network bandwidth as compared to standard UDP and TCP protocols. This paper aims at validating and invalidating the behavior of MPTCP protocol as proposed in [1] along with enunciating the network conditions under which the MPTCP protocol achieves optimal performance. The primary contributions of our experiments include:

- Validation of key goals of MPTCP described in [2] i.e.
- Fair share with TCP during congestion
- Perform at least as well as TCP
- Should always use efficient path
- Efficient Resource Utilization
- Depiction of a threshold point (i.e. number of sub-flows) on and above which MPTCP always out performs TCP and UDP.
- Ethernet was always preferred over Wi-Fi in spite of creating conditions under which Wi-Fi should have been opted as an efficient path for data transmission.
- Performed a comparative analysis of several congestion control algorithms such as LIA, OLIA, BALIA and wVegas.

2. RELATED WORK

There has been a good deal of work in the field of MPTCP protocol. This work includes the motivation for MPTCP algorithm, its usability over TCP, design principles and some proposed fundamental behavior (such as the concept of fair share, performance as good as TCP, opting for efficient paths for transmission and optimal resource allocation). Also, there's been a lot of work on building multipath transport protocols [2]. Most of this work focuses on evaluating the congestion control protocols which fits best for MPTCP rather than analyzing the real time conditions under which MPTCP can actually perform better than TCP and UDP. The paper [2] discusses the real time working of MPTCP, COUPLED, SEMI-COUPLED and EWTCP algorithms. The main focus in that was congestion control in MPTCP.

There has been work describing the design of a multipath congestion control algorithm, its Linux multi-homed servers, data centers and mobile clients. It has been shown that some 'obvious' solutions for multipath congestion control might not function optimally. However, [2] proposes an algorithm, which is a replacement of TCP, that improves throughput and fairness compared to single-path TCP.

Additionally, [3] talks about Opportunistic Linked Increases Algorithm (OLIA) and LIA congestion controls for MPTCP and their trade offs. It talks about the current congestion control algorithm of MPTCP, LIA, forcing a tradeoff between optimal congestion balancing and responsiveness. OLIA's algorithm is similar to that of LIA except the increase part where the following formula is followed:

For each ACK on path r, increase w_r by

$$\frac{w_{\_r}/rtt_{\_r}{}^2}{(\sum(w_{\_p}/rtt_{\_p}))^2} + \frac{alpha_{\_r}}{w_{\_r}}$$

multiplied by MSS_r * bytes_acked.

Hence, it shows how OLIA's design departs from this tradeoff and solves the identified performance problems with LIA while retaining non-flappiness and responsiveness behavior of LIA.

Next, Balanced Linked Adaption (BALIA) congestion control for MPTCP has been described in [4]. It states the motivation behind BALIA which is that LIA and OLIA, suffer from either unfriendliness to Single Path TCP (SPTCP) or unresponsiveness to network changes under certain conditions. BALIA uses a new design framework that allows one to systematically explore the design space, thereby judiciously balancing the tradeoff between friendliness and responsiveness. BALIA's algorithm differs only in the AIMD part of the congestion avoidance phase and the window size is modified as follows:

For each ACK on path r, increase w_r by:

$$\frac{x_{\_r}}{rtt_{\_r} * (\sum(x_{\_k}))^2} * \left(\frac{1+alpha_{\_r}}{2}\right) * \left(\frac{4+alpha_{\_r}}{5}\right)$$

For each packet loss on path r, decrease w_r as:

$$\frac{w_r}{2} * \min(a_r)$$

$x_{\_r} = w_{\_r}/rtt_{\_r}$ and $alpha_{\_r} = \max(x_{\_r})/x_{\_r}$.

Furthermore, [5] delineates Weighted Vegas (wVegas) a delay-based congestion control scheme for MPTCP. It adopts packet queuing delay as congestion signals, thus achieving fine-grained load balancing. Also, it is more sensitive to changes of network congestion, thereby achieves more timely traffic shifting and quicker convergence. The algorithm mainly applies to the congestion avoidance phase and, in slow start phase, it also adds a chance to enter the congestion avoidance phase earlier, which is similar with the implementation of the TCP-Vegas. The decrease of the congestion avoidance phase, the fast retransmit and fast recovery algorithms are the same as TCP.

Finally, all the above discussed work has been aimed at the preliminary design of MPTCP along with proposing its expected behavior. The work related to different congestion controls also just gives the insight of the algorithm.

3. DETAILED DESIGN

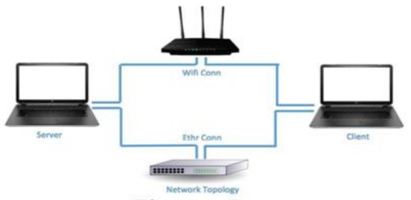


Figure A

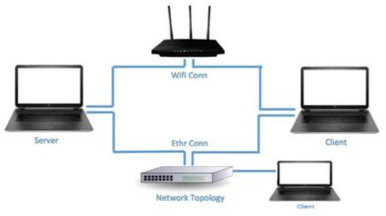
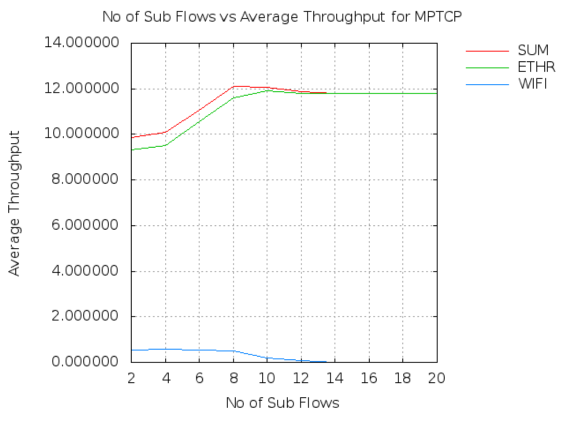


Figure B

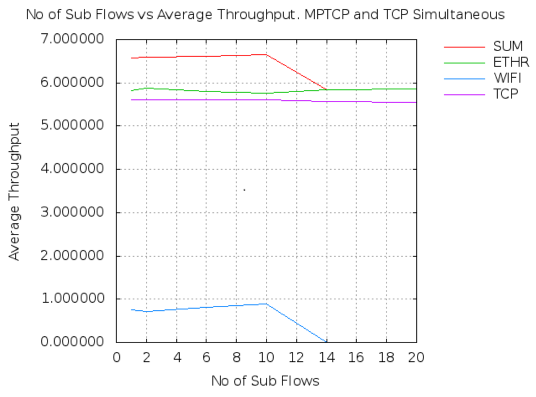
- Figure A depicts our primary setup which has been used primarily for our experimental purposes.
- It comprises of a MPTCP enabled client and server (on Linux kernel) having multiple communication paths viz. Ethernet (5-port 12.5MBps Unmanaged Gigabit Switch) and Wi-Fi (1MBps).
- The primary purpose of this setup was to initially emulate a scenario wherein which a Client host establishes a connection with a Server in order to exchange information by setting up a reliable TCP connection. Later, we modified the kernel at both the client and the server with the purpose of enabling MPTCP exchanges between the two hosts.
- Following the Linux kernel modifications, there were additional design decisions that needed to be made with respect to efficiently being able to send data using MPTCP connection.
- These decisions pertained to selecting the apposite congestion control algorithm which provides the optimal performance with respect to MPTCP.
- In addition, we also needed to fix the number of flows for each MPTCP connection which would potentially render the MPTCP application extremely efficacious.
- Once the basic setup was in place, we carried out a string of experiments as elaborated in the Evaluation section. In general, for the MPTCP experiments, the hosts were provided with 2 active network interfaces. The first interface provided an Ethernet connection between the hosts as managed by the aforementioned Gigabit Switch. The secondary network path in place was a commercial Wi-Fi connection.
- Figure B lays out our secondary architectural setup which has an additional (Non MPTCP) client (connected via Ethernet) introduced to compete with the MPTCP client, to emulate pragmatic real world scenarios.
- This setup was adopted to study the behavior of MPTCP protocol with respect to a number of key principles such as fair-share and to evaluate if MPTCP works at least as well as a single TCP connection in a shared network.
- In addition, the setup as depicted in Figure B also assists us in carrying out experiments to test how robust MPTCP protocol is when faced with a scenario wherein, there is active congestion in the network path due to the presence of a competing TCP/UDP connection.

4. EVALUATION

As mentioned in the previous section, we use MPTCP enabled client and server (on Linux kernel) having multiple communication paths viz. Ethernet (5-port 12.5MBps Unmanaged Gigabit Switch) and Wi-Fi (1MBps) for our experimental purposes.



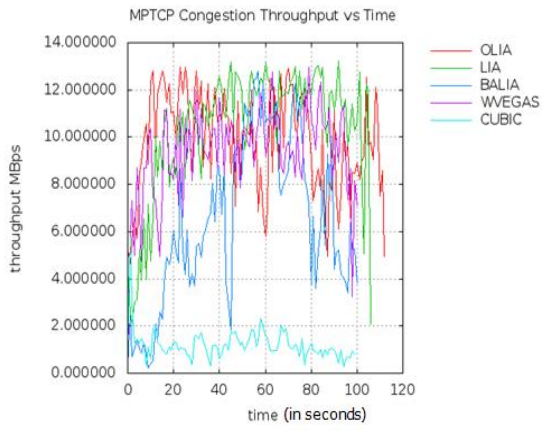
GRAPH 1



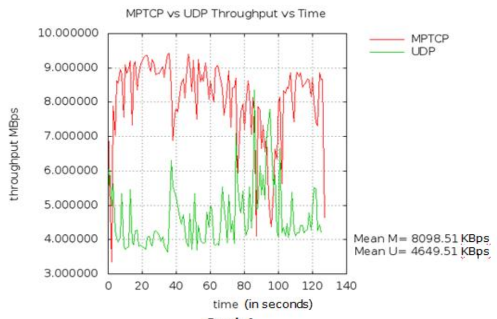
GRAPH 2

The first study that we conducted was to determine the optimal number of flows that ensure maximum utilization of network resources efficiently. For this purpose we varied the number of flows on each network path until we found the convergence point at which MPTCP performed optimally and beyond which the protocol does not meet its fundamental requirement, i.e., it ceases to transmit data over multiple network paths. *Graph 1* shows a study of the same wherein the number of flows (X-Axis) is uniformly increased starting from 2. We measured the respective throughput got for each flow count and observed that when flow count equals 10 for each network path, performance of MPTCP is optimal. The study was orchestrated to analyse this behaviour with respect to all the 3 scenarios, i.e., when MPTCP uses only Ethernet, when it uses only Wi-Fi and finally when it makes use of both the connections. *Graph 2* portrays a similar experimental analysis, with the addition of a simultaneous TCP flow, thus analysing the throughput of MPTCP under congestion.

Our next analysis was performed to find the best performing Congestion Control algorithm among existing algorithms designed for MPTCP protocol, namely, OLIA, LIA, BALIA, WVegas. *Graph 3* provides a comparative analysis among the algorithms representing an instantaneous throughput value with respect to time over a span of 120 seconds. This analysis lucidly underlines the fact that OLIA outperforms all the other congestion control algorithms in real-time, thus justifying our usage of OLIA in all our MPTCP related work and experiments.

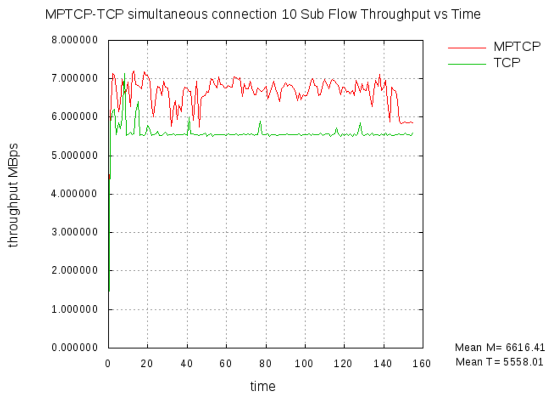


Graph 3

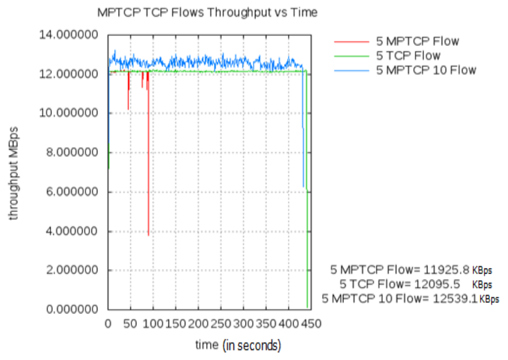


Graph 4

Using the above parameters, specifically fixing the number of sub-flows to 10 and using OLIA as our congestion control algorithm, we conduct performance analysis of MPTCP with respect to both TCP and UDP so as to give a very clear picture of MPTCP's usability and justifies why the protocol should be commercialised. *Graph 4* studies the performance of MPTCP with respect to UDP, and as it can be seen, MPTCP does seem to outperform UDP on an average, however, during certain short-lived time spans UDP performs almost as well as MPTCP. In *Graph 5* we compare the performance of MPTCP with respect to TCP and as expected MPTCP outperforms TCP throughout our study, thus strengthening our claim that using MPTCP indeed ameliorates the user experience with respect complete network utilization.



Graph 5



Graph 6

Further, we studied MPTCP's performance with respect to TCP by first equating the number flows in a single MPTCP connection to the number of logically separate TCP connections. Secondly, this study is analysed against n number of MPTCP connections where n equals the number of logically separate TCP connections. In this case, the number of sub-flows for each MPTCP connection equals 10 which as earlier mentioned, is optimal. According to the study conducted, it makes it quite evident of the fact that n number of MPTCP connections with sub-flows=10 out rightly outperforms the logically independent TCP connections which in-turn outperforms the single MPTCP connection having n number of sub-flows. This observation leads to the conclusion of the fact that having multiple MPTCP connection ensures the maximum resource utilization as compared to having multiple TCP connections. However, this study also brings out an interesting finding, which being that for equal number of TCP connections as the number of sub-flows in a single MPTCP connection, TCP does perform better that MPTCP.

A major benefit we while using MPTCP is it helps in maximum utilization of the resources by creating flows in under - utilized paths. For example, consider the topology as shown in figure C. For the below experiment we configured 4 laptops as routers who are connected to Wi-Fi on one side and Ethernet on the other. Let us name the laptop as Src A, Des A, Src B, Des B.

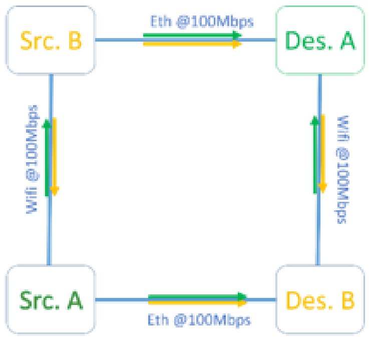


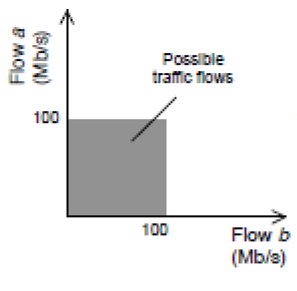
Figure C

In scenario 1: A TCP flow has been started from Src A to Des A using a path via Des B. Another TCP flow has been added from Src B to Des B which chooses path via Des A. As it can be clearly seen the Wi-Fi link Src B to Src A is never utilized, thus concluding TCP is not able to utilize the resources to its full capacity.

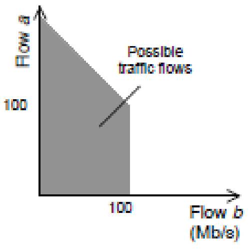
In scenario 2: An MPTCP flow is started from Src A to Des A and Src B to Des B. The first flow is able to utilize full bandwidth from both paths. Similar is the case for MPTCP flow directed from

Src B to Des B. Thus it can be concluded that MPTCP is able to utilize the resources to its full capacity.

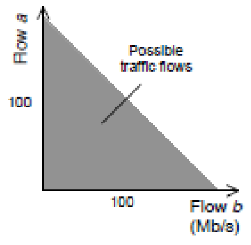
The resource pooling further allows a wider range of traffic matrices on the network as shown in graph 7. When you are only using TCP Flows you get a traffic matrix shown in *Graph 7a*. It can be clearly seen that the you only get 100 Mbps throughput for both Flow A and Flow B. When you enable an MPTCP on one of the flow on one path you get higher output, but only able to utilize full bandwidth of Network when TCP throughput = 0 as shown in *Graph 7b*. When you enable MPTCP flow on both the path you are able to utilize full bandwidth of both the paths thus gets a much wider traffic matrix as shown in *Graph 7c.*



Graph 7 a



Graph 7 b



Graph 7 c

5. DISCUSSION & CONCLUSION

We have presented a real time hardware verification of MPTCP protocol along with validating performance of multiple distinct MPTCP congestion control algorithms. Our three experimental design setups depicted three different network topologies and the tweaking of several network properties such as RTT delay, packet loss rate and MPTCP's fundamental parameters such as number of sub-flows resulted into the following key findings:

- MPTCP follows the fair share property while increasing the number of sub-flows.
- It performs as well as TCP when there is competing traffic.
- MPTCP utilized full capacity of the resources (i.e. throughput) on available paths as compared to TCP connection.
- From the comparative analysis among the MPTCP congestion control protocols, it can be concluded that OLIA performs the best by consuming least time for transmission.
- Also, it was observed a N MPTCP connection with (*N) sub-flows perform better than N TCP connections.

Some unexpected observations from our real time analysis were:

- In the experimental setup, MPTCP always preferred Ethernet in spite of increasing RTT delay or packet loss on it.
- We found that the optimal number of sub-flows count (*N) is 10, for which MPTCP outperforms TCP and UDP, always maintaining the property of fair share in both the cases.
- Finally, when the number of subflows were increased beyond 10, MPTCP ceases to send data over multiple paths owing to added processing overhead.

From the experience attained and corner cases observed due to some aspects of the experimental setup, the things that we would do differently, the next time we attempt research regarding MPTCP protocol are as follows:

- We would attempt to get a clean Wi-Fi connection so that we get a higher bandwidth on this path, thereby doubly validating the achieved result of MPTCP not opting for efficient Wi-Fi.
- Also, it would have been better to have more than two MPTCP enabled laptops. This would have helped in creating newer and complex network topologies for the simulation.
- Next, the analysis could have also been conducted on the external network

topology rather than a local network setup, for getting an idea of large-scale real time functioning of MPTCP.

As future work, evaluating the scalability with respect to deploying it over the internet , energy considerations for mobile devices, implementing better window management schemes are some the avenues of interest.

6. REFERENCES

[1] The Multipath-TCP
Available at: http://multipath-tcp.org (Accessed: 05th October, 2015)

[2] D. Wischik, C. Raiciu, A. Greenhalgh and M. Handley. Design, implementation and evaluation of congestion control for multipath TCP.

[3] R. Khalili. Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP.

[4] A. Walid. Balanced Linked Adaptation Congestion Control Algorithm for MPTCP.

[5] M. Xu. Delay-based Congestion Control for MPTCP.

[6] O. Bonaventure, M. Handley, and C. Raiciu. An Owerview of Multipath TCP.

[7] M. Zubair Shafiq, F. Le, M. Srivatsa, A. Liu. Cross-Path Inference Attacks on Multipath TCP.

[8] Y. Chen, Y. Lim, et. al. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks.

[9] C. Raiciu, S. Barre, et. al. Improving Datacenter Performance and Robustness with Multipath TCP.

[10] D. Gucea, O. Purdila. Shaping the Linux kernel MPTCP implementation towards upstream acceptance.

[11] C. Raiciu, C. Paasch, et. al. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP.

[12] M. Scharf. Multipath TCP (MPTCP) Application Interface Considerations.