PERFORMANCE ANALYSIS OF MULTI-PATH TCP NETWORK

Hamzah M A Hijawi and Mohammad M. N. Hamarsheh

Department of Computer Science, Arab American University, Jenin, Palestine

ABSTRACT

MPTCP is proposed by IETF working group, it allows a single TCP stream to be split across multiple paths. It has obvious benefits in performance and reliability. MPTCP has implemented in Linux-based distributions that can be compiled and installed to be used for both real and experimental scenarios. In this article, we provide performance analyses for MPTCP with a laptop connected to WiFi access point and 3G cellular network at the same time. We prove experimentally that MPTCP outperforms regular TCP for WiFi or 3G interfaces. We also compare four types of congestion control algorithms for MPTCP that are also implemented in the Linux Kernel. Results show that Alias Linked Increase Congestion Control algorithm outperforms the others in the normal traffic load while Balanced Linked Adaptation algorithm outperforms the rest when the paths are shared with heavy traffic, which is not supported by MPTCP.

KEYWORDS

TCP, MPTCP, Multi-homed, Congestion Controls

1. INTRODUCTION

Transmission control protocol TCP is widely used on the internet today. Most applications use TCP as a reliable transfer protocol to transmit data across the internet. TCP has been evolving since the first design from the 1970s. One of the early design decisions in TCP still frustrating users; the separation between transport and network layers is not completely transparent. However, in order to differentiate between individual data streams among incoming traffic, the receiver demultiplexes packets based on the five tuples; source IP, destination IP, source port, destination port and the protocol identifier, this is also called socket. Thus, any TCP connection is a single path and bounded to a unique socket between the source and destination [1]. This means that after the connection is established if for some reason the IP address is changed in the source or destination then the whole connection will fail.

Today networks are becoming multipath, most of the servers are multi-homed, mobile devices have multiple wireless interfaces like WiFi and GSM, and data centers have many redundant links. This type of redundancy evolves the need for a new TCP design to make it operational over multipath, this is called multipath TCP [2]. MPTCP is a major modification to TCP, which allows multiple paths to be used simultaneously by a single TCP transport connection. It was first proposed by IETF [3]. MPTCP has obvious benefits for reliability in case of link failures and for load balancing in case of multi-homed servers and data centers. MPTCP also can achieve better performance and it is more robust than a single path TCP while maintaining compatibility with existing applications. This is achieved by providing a regular TCP interface to applications and let the transport layer handles the multipath connections. MPTCP is presented as a sublayer under TCP layer, it spreads data over multi-paths which are called sub-flows [3].

DOI : 10.5121/ijcnc.2016.8213

The design of MPTCP has been influenced by many requirements. Mainly, application and network compatibilities [2]. Application compatibility means that applications that currently run over TCP should work over MPTCP without any change. Network compatibility means that MPTCP should work with any network on which TCP operates.

MPTCP connection is established by using three-way handshake with TCP options. MP_CAPABLE option in the SYN packet indicates that the source can perform multipath TCP. If the destination also supports multipath TCP, then it replies with SYNACK packet, which also contains MP_CAPABLE option, after that source replies with ACK packet which again contains MP_CAPABLE option to confirm that the connection will use MPTCP. Other options like using a random key in the three-way handshake can be used for security purposes. Once the connection established any of participants can create additional sub-flow if it finds a new path to the destination by sending a new SYN packet with MP_JOIN option and information on which MPTCP session to join. Once the connection has more than one sub-flow, it is up to the system on each end to decide how to split the traffic between sub-flows. Each sub-flow looks like a normal TCP connection; it has its own sequence numbers and congestion control which are different from other sub-flows. New sub-flow can come up if a new path is available and can vanish if one of the paths becomes unavailable [4].

One of the most important components of TCP is its congestion control mechanism, which allows it to adjust throughput according to the network congestions. Each TCP connection maintains a variable called congestion window. It governs the amount of traffic that the sender can send without waiting for an acknowledgment from the receiver. The same approach is applied in MPTCP in which each sub-flow has its own congestion window and it is updated dynamically according to the congestion in the path it takes to reach the destination. Few congestion control algorithms for MPTCP has been developed like coupled congestion control [5], opportunistic linked-increases congestion control [6, 7], balanced linked adaptation congestion control [8] and delay based congestion control [9].

The main goal of this paper is to evaluate the new emerging MPTCP protocol over single and multiple interfaces. The throughput of different scenarios is measured using the Linux operating system. Outputs from all scenarios are compared with each other and the performance of different MPTCP congestion control algorithms is compared.

2. RELATED WORK

MPTCP is a new approach towards efficient load balancing, it does balance at the end nodes as part of TCP process. MPTCP was described in RFC 6897 [4], an active working group in the IETF. Many attempts have been made in order to implement MPTCP in the Linux Kernel. An effort was made in [10]where the Kernel can be tuned and customized by system call variables.
With the possibility of using multiple paths for TCP, concerns have arisen about congestion controls on these paths. An effort was made to implement effective and robust congestion control mechanisms that effectively use the paths without harming the original TCP path. In [11], it was shown that implementing some congestion control algorithms for multipath TCP could be harmful. Equally-Weighted TCP EWTCP is an example of these harmful algorithms [12]. It is a weighted version of multipath TCP which distributes the data fairly compared to the regular MPTCP. It is not very efficient because it doesn't split traffic evenly across the available paths. COUPLED congestion control algorithm makes better decisions than EWTCP and solved some fairness problems [5]. The weakness of this algorithm is obvious when the available paths have different throughputs like the case in WiFi and GSM. It makes unequal round trip times (RTTs) because the COUPLED algorithm tends to send all its traffic on the less congested path, which makes the use of low throughput path inefficient. The proposed solution to these problems is

given in [12]. It is called SEMI-COUPLED congestion control algorithm. It is implemented based on some design goals like good path selection, balanced congestion, and compatibility with existing regular TCP.

A practical multipath congestion control algorithm is proposed in [13]. It focuses on improving throughput and balance the congestion. It also compares the proposed algorithm with other three congestion control algorithms: Fully coupled, linked increases and uncoupled TCP. The simulation was carried out using CWNDSim. It was shown that the proposed algorithm succeeded in keeping total throughput close to the target rate, and the traffic is pushed into the less congested path. An overview of fully functional MPTCP is provided in [2]. The results performed on the Linux machines that implemented MPTCP protocol in their Kernels. They studied the most used cases for MPTCP which are the mobile devices and data canters. The data transfer measurements of the mobile focus on the typical mode of operation in which the mobile device is connected to WiFi, then connection goes down and the device switched to 3G. The results were compared with the single path TCP and showed a smooth handover in the case of MPTCP, this is because data keeps flowing despite the interface changed. The other scenario is load balancing in the datacenter; the comparison was made with standard TCP, two flows MPTCP, and four flows MPTCP when tested on EC2 test bed with 40 instances. They showed that MPTCP with four paths outperforms both standard TCP and two paths MPTCP.

3. MPTCP CONGESTION CONTROL

MPTCP distributes loads through the creation of multiple sub-flows across potential paths between source and destination. Congestion control in MPTCP is different from regular TCP. The simplest way is to run the standard TCP congestion control on each sub-flow. However, this solution is not efficient as it will give the multi-path flow more than its fair share if more than one sub-flow across the same bottleneck link which is also shared with standard TCP flow.Furthermore, it is desirable that the source with multiple paths will transfer more traffic using the least congested path, achieving the resource pooling where a group of links behaves like one shared link with bigger capacity [5].

Any multipath congestion control algorithm must meet a set of requirements which are summarized in three different goals. The first goal is improving the throughput; a multipath connection should perform at least as well as single path TCP would on the best of available paths, this means that in the worst case the MPTCP would have throughput same as standard TCP. The second goal does not harm;a multipath sub-flow should not take capacity more than single path TCP would on the same link, this ensure that multipath sub-flow will not harm other flows. The third goal is balancing the congestion; the multipath flow should move as much as traffic to the least congested paths [3, 5, 11]. The first and second goals together ensure the fairness at bottlenecks while the third goal achieves the concept of the resource pooling. However, if each multipath flow sends more traffic through its least congested path, the network traffic will move away from congested paths resulting in improvements in robustness and overall network throughput.

Many congestion control algorithms were proposed, most of them were implemented in Linux Kernels. The following four congestion control algorithms were implemented in most of Linux distributions [10].

3.1 Alias Linked Increase Congestion Control

Alias Linked Increase Algorithm LIA couples the congested control algorithms that are running on different sub-flows by linking their increase functions and dynamically adapts the congestion window [5],the algorithm is only applied to the increase part of the congestion avoidance phase. The resultis an algorithm that is fair to standard TCP at bottlenecks and at the same time moving the traffic away from congested paths.

The additive increase and subtractive decrease behaviors can be described asfollows:

- For each ACK received on sub-flow *i*, the congestion window $cwnd_i$ is increased by:

$$Min\left\{\frac{\alpha.\mathrm{B_{ack}}.\mathrm{Mss_i}}{\sum_{i=0}^{n} cwnd_i}, \frac{B_{ack}.Mss_i}{\mathrm{cwnd_i}}\right\} \quad (1)$$

Where:

α: A parameter that describes the aggressiveness of the multipath flow.
B_{ack}: Number of acknowledged bytes.
Mss_i: Maximum segment size on sub-flow *i*.
n:Total number of sub-flows.

- For each loss on sub-flow i, decrease $cwnd_i$ by $cwnd_i/2$.

Formula (1) describes the increase behavior of the algorithm in Bytes, thefirst argument computes the window increase value for the multipath sub-flow and the second one computes the increasevalue for TCP in the same scenario. Taking the minimum of these values ensures that any multipath sub-flow will not take capacity more than single path TCP, hence achieving the second design goal. Alpha is a parameter used to adjust the aggressiveness of the multipath flow, its valueis chosen such that the total throughput of the multipath flow is equal to the throughput which TCP would get on the best path and this meets the first design goal.

According to the first term in formula (1), the total throughput for a multipath flow depends on alpha, maximum segment sizes, and round trip times of its paths. In order to meet the first design goal, it is impossible to choose a single value of alpha that achieves the desired throughput at each time. Hence, alpha is computed based on the observed behaviors of all paths as shown in formula (2), alpha is derived by equalizing the rate of multipath flow with TCP flow running on the same path.

$$\left(\sum_{i=0}^{n} cwnd_i\right) \frac{Max\left\{\frac{cwnd_i}{RTT_i^2}\right\}}{\left(\sum_{i=0}^{n} \frac{cwnd_i}{RTT_i}\right)^2} \quad (2)$$

Where:

$Max\left\{\frac{cwnd_i}{RTT_i^2}\right\}$:Maximim value of any possible path.

$\sum_{i=0}^{n} \frac{cwnd_i}{RTT_i}$: Summation of all possible values of all paths.

3.2 Opportunistic Alias Linked Increase Congestion Control

Through analysis and measurements provided in [7], it was proven that current LIA implementation forces a tradeoff between optimal resource pooling and the responsiveness, both goals can't be achieved at the same time, this leads to the fairness problem to TCP users. However, upgrading part of users from standard TCP to MPTCP can reduce the throughput for other users without any benefit to MPTCP users and can also violate the third design goal.

OLIA, the opportunistic linked increases algorithm wasintroduced as an alternative for LIA,it a window based congestion control algorithm. It couples the increase of congestion windows and uses the same behavior in regular TCP in case of loss. This algorithm is only applied to the increase part of the congestion avoidance phase, slow start algorithm is the same as the one used in regular TCP with a small modification in case of multiple paths are established [6, 7].The additive increase behavior can be described as follows:

- For each ACK received on path *i*, increase congestion window $cwnd_i$by:

$$\frac{\frac{\mathrm{cwnd_i}}{RTT^2}}{\left(\left(\sum_{i=0}^{n} cwnd_i\right).\left(\frac{cwnd_p}{RTT_p}\right)\right)^2} + \frac{\alpha_i}{cwnd_i} \tag{3}$$

  Where:

  $cwnd_p$:Window size of a path p with largest congestion window.
  RTT_p:Round trip time of a path p with largest congestion window.
  α_i: Adjust parameter for a path i.
  n: is the total number of sub-flows.

- For each loss on sub-flow i, decrease $cwnd_i$ by $cwnd_i/2$.

The first term in formula (3) provides the optimal resource pooling, it is a TCPcompatible version that compensates for different RTTs. The second term with alpha guarantees the responsiveness and non-flappiness of the algorithm.

3.3 Balanced Linked Adaptation Congestion Control

Recent MPTCP congestion protocols like LIA and OLIA suffer from either unfairness to the single path TCP or unresponsiveness to network changes under certain conditions especially when all paths used for MPTCP have the same round trip time. However, the trade-off between these two issues is inevitable. Balances linked adaptation algorithm BALIA judiciously balances this trade-off [8],it strikes a good balance between friendliness and responsiveness. The additive increase and subtractive decrease behaviors of BALIA can be described as follows:

- For each ACK on path *i*, increase $cwnd_i$ by:

$$\left(\frac{\mathrm{x_i}}{RTT_i.\left(\sum_{k=0}^{n} x_k\right)^2}\right).\left(\frac{1+a_i}{2}\right).\left(\frac{4+a_i}{5}\right) \tag{4}$$

  Where:

$$x_i = \frac{cwnd_i}{RTT_i}$$

$$a_i = \frac{Max\{x_k\}}{\mathrm{x_i}}$$

n: is the total number of sub-flows.

- For each loss on path *i*, decrease $cwnd_i$ by:

$$\left(\frac{\mathrm{cwnd_i}}{2}\right) . Min\{a_i, 1.5\} \quad (5)$$

If there is only a single path available then a_i will be one and the increment and decrement formulas will be reduced to those of TCP Reno algorithm.

3.4 Delay-Based Congestion Control

A Delay-based congestion control algorithm called wVegas is introduced in [9]. Unlike LIA which is based on packet loss events, wVegas uses packet queuing delay as a congestion signal. Compared with loss based congestion control algorithms described previously, wVegas is more sensitive to the changes of network congestions, and can achieve more timely traffic shifting and faster convergence. The following operations must be performed at the end of each transmission round:

- For a sub-flow*i*, calculate the difference between the expected sending rate and actual sending rate.

$$diff_i = \left(\frac{\mathrm{cwnd_i}}{base\_RTT_i} - \frac{\mathrm{cwnd_i}}{RTT_i}\right) . base\_RTT_i \quad (6)$$

  Where, RTT_i is the average *RTT* on the last round on sub-flow *i*, and $base\_RTT_i$ is the *RTT* of a sub-flow *i* when the path is not congested.

  If the sub-flow is in the slow start phase and the $diff_i$ is larger than a threshold called gamma, the algorithm must enter the congestion avoidance phase.

- In the congestion avoidance phase, if the $\mathrm{diff_i}$ is not less than unfairness a_i, then the rate must be updated.

$$rate_i = \frac{\mathrm{cwnd_i}}{RTT_i} \quad (7)$$

$$weight_i = \frac{\mathrm{rate_i}}{total\ rate\ of\ all\ i} \quad (8)$$

$$a_i = weight_i\ .total\_a \quad (9)$$

  If $diff_i$is larger than a_i, then $cwnd_i = \mathrm{cwnd_i} - 1$ else,$cwnd_i = \mathrm{cwnd_i} + 1$

- The last task that wVegas tries to do is improving the accuracy of $\mathrm{base\_RTT_i}$ by making the congestion window back off once detecting a queuing delay larger than some thresholds. By this task, bottleneck link can drain off the backlogged packets. However, all flows involved have a chance to obtain the most accurate propagation delay. wVegas first calculates the queuing delay as follows:

$$queue\_delay_i = RTT_i - base\_RTT_i \quad (10)$$

If the current queuing delay is less than the saved queue_delay$_i$ then the queue_delay$_i$ must be replaced by the current one. If the current queuing delay is twice queue_delay$_i$ then perform the following operation:

$$\text{cwnd}_i = (\text{cwnd}_i).(0.5).(\frac{base\_RTT_i}{RTT}) \quad (11)$$

4. EXPERIMENTAL RESULTS

Experimental evaluation of MPTCP is performed using Ubuntu Linux. The Kernel is replaced with Ubuntu Kernel, which supports MPTCP [10]. Multiple experimental scenarios are done using a laptop that had WiFi and 3G interfaces. The topology of the system used is described in figure 1. Ubuntu 14.04 is installed on a laptop which represents the FTP client, and the Linux Kernel is replaced with multipath TCP v0.90 Kernel.Each of the network interfaces has a speed of 4Mbps.

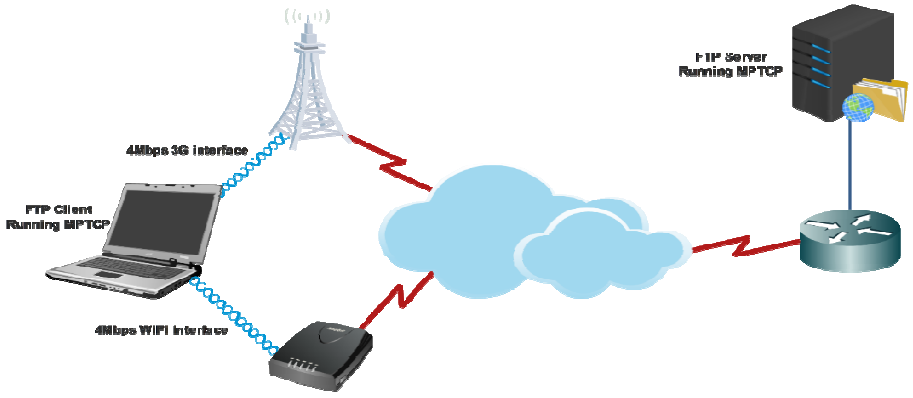

Figure 1: System Topology

4.1 Comparison between Regular TCP and MPTCP

In this experiment, we downloaded a 1GB file from a server through the internet in three different scenarios. The first scenario by using regular TCP over WiFi interface, thesecond scenario by using regular TCP over 3G interface, and the last scenario by using MPTCP over WiFi and 3G at the same time. The comparison of the throughput for the three scenarios is shown in Figure 2. The figure shows that using MPTCP over all available interfaces can enhance the throughput as expected. It also shows that the throughput using MPTCP is almost the sum of each path throughput.Sometimes the throughput is less than the sum because the experiments are done using a link with shared capacity to the internet. Throughput is monitored using a Linux tool called ifstat [14], the summation of download speed for each network interface is collected and drawn as an average value for each ten readings; this is to make the graph smoother. It is also clear that that the 3G link stability is not as good as the WiFi link and this is reflected on the performance of the MPTCP link. This explains the odd behaviour at 150, 300 and 550 s.

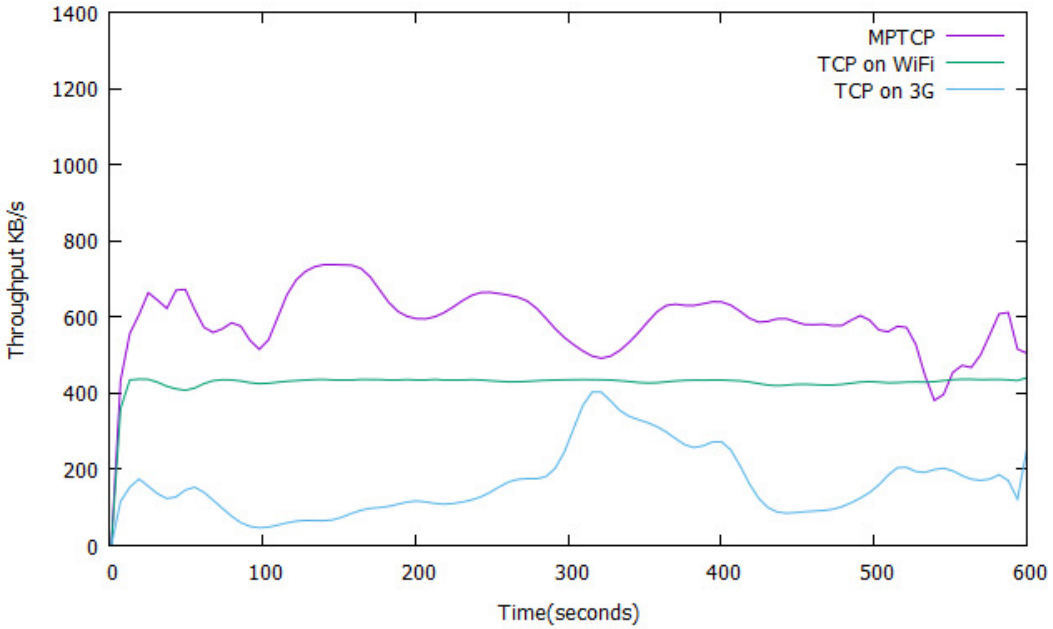

Figure 2: Throughput using MPTCP at normal traffic

4.2 Heavy Load Comparison

In this experiment, we use the FTP to download 1GB file using MPTCP over WiFi and 3G interfaces. While the file is downloading, YouTube video fileat time 300 seconds is started and consumes most of the bandwidth. The same scenario was repeated with standard TCP. Figure 3 shows the throughput for both cases. The figure shows that the file gets better throughput in case of MPTCP as expected. YouTube server does not support MPTCP and the video download was using WiFi interface only. Thus, the total capacity of the3G interface is used for the FTP download and the WiFi is shared between FTP and video download. Before time 150seconds, the through put of the MPTCP is not steady because of the shared link capacity.When YouTube video started at about 280seconds, the throughput dropped to 60 KB/s for standard TCP and to 170 KB/sfor MPTCP, this shows how using multi-paths can enhance MPTCP applications throughput without affecting the behavior of standard TCPat congestion bottlenecks.

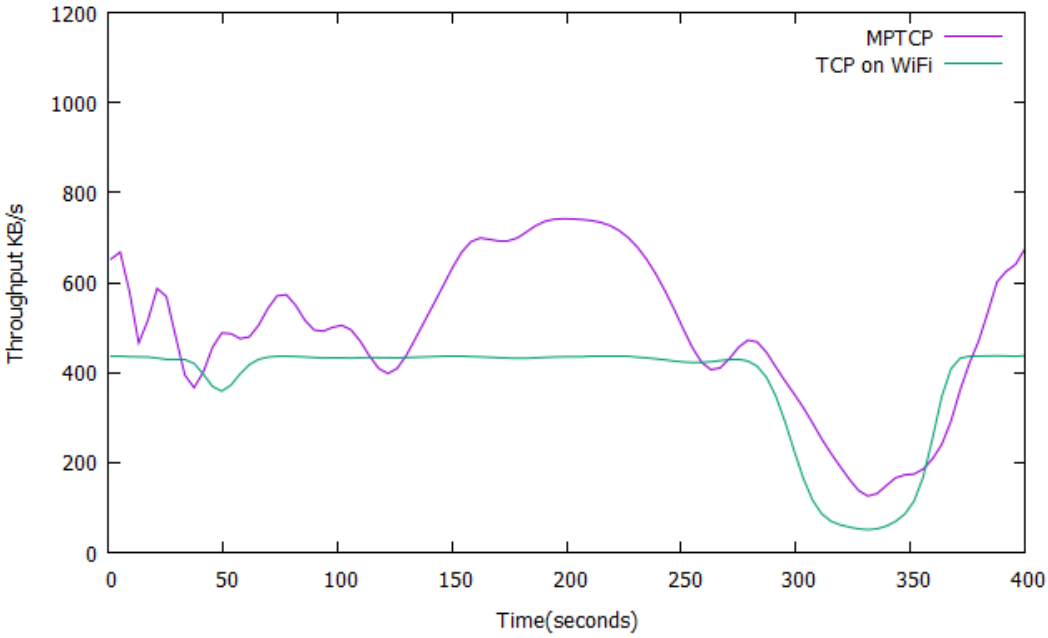

Figure 3: Throughput using MPTCP at heavy traffic

4.3 MPTCP sub-flows over Single Interface

In this experiment, the download throughput is monitored using MPTCP with one sub-flow over a WiFi interface. It is compared withdownload throughput of two sub-flows over the same interface. Results show a good enhancement on throughput as the number of sub-flows increase over the same path. Figure 4 shows that the download speed in case of MPTCP with two sub-flows is increased by 70 KB/s when it is compared with MPTCP with 1 sub-flow.

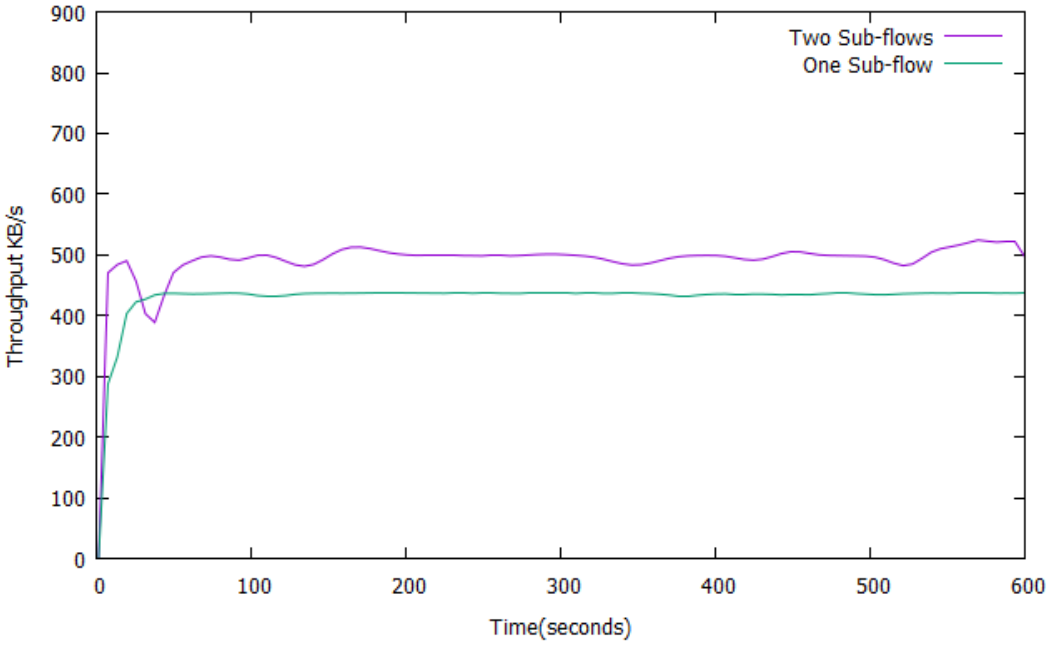

Figure 4: MPTCP sub-flows

4.4 Application Layer Handover

The throughput of TCP with application layer handover is monitored and compared with that of MPTCP. The throughput of TCP protocol is monitored while downloading a 1GB file from the server using a TCP connection on WiFi interface. The WiFi access point is switched off during the download. The application detected that WiFi interface went down and reconnected with a new TCP connection (switched) to the 3G interface. The same scenario is implemented again with MPTCP enabled. Figure 5 shows the results from both cases, the handover occurs between 100 – 130 seconds with MPTCP and data keeps flowing despite the lossof one of the interfaces. It is shown that at the handover time, the download speed does not go to zero in thecase of MPTCP.In thecase of TCP, the throughput goes to zero for all of the handover periodsuntil anew connection is established.

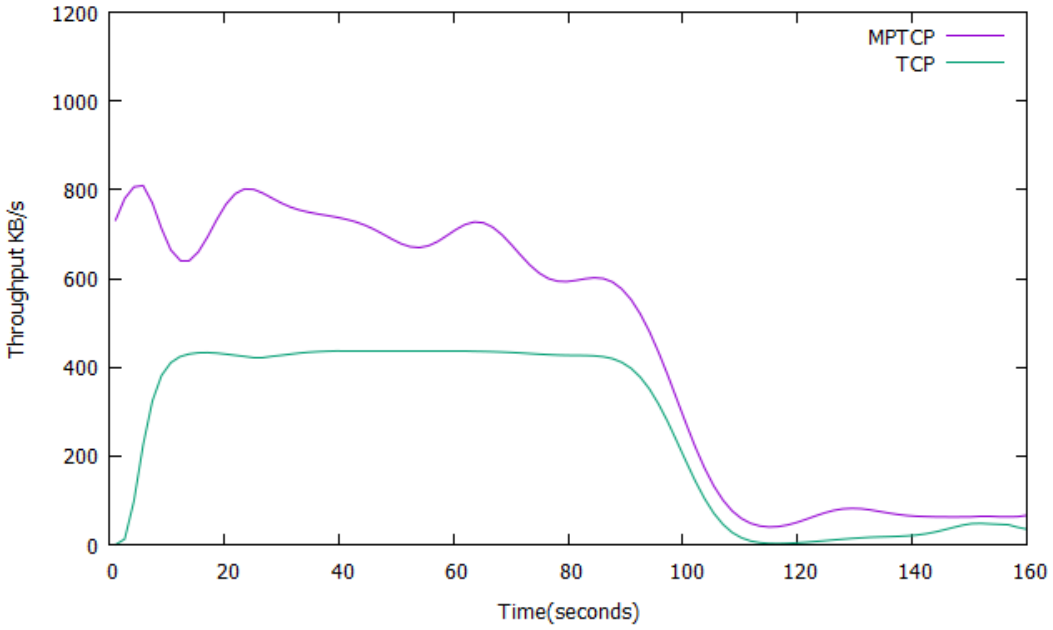

Figure 5: MPTCP handover

4.5 Congestion Controls Algorithms Comparison

In this experiment, a comparison of the throughput when downloading a 1 GB file is made between the four different congestion control algorithms discussed in section three. The file is downloaded 4 times, each time different congestion control algorithm is used. These are LIA, OLIA, BALIA and wVegas. WiFi and 3G interfaces are both utilized for MPTCP at normal traffic load. Experiment results show that LIA has thebest throughput most of the time (150s – 480s) as shown in figure 6. BALIA also has a good throughput for the time from 0s to 320s at this type of load. OLIA and wVegas did not show good throughput at this load and were not stable most of the time.

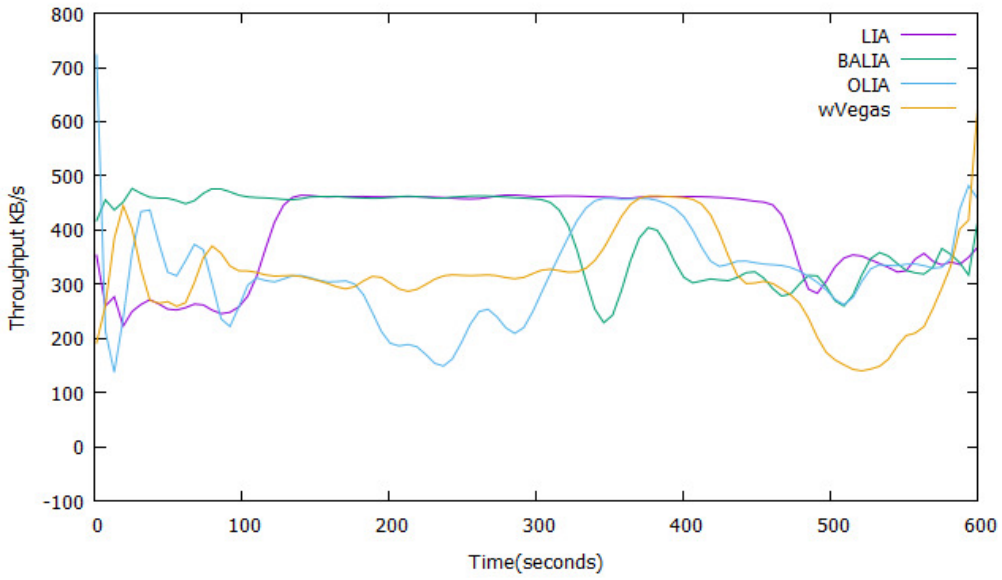

Figure 6: Throughput comparison between MPTCP congestion controls algorithms

4.6 Congestion Controls Algorithms Comparisonunder Heavy Traffic

The congestion control algorithms are designed to improve the throughput at heavy load. The throughput is monitored at heavy load while downloading the 1GB file four different times. Each time one of the congestion control algorithm is used and a YouTube video file is started at time 300 seconds to consume most of the bandwidth.The throughput in each case is compared with all other cases to show the behaviour of the four different congestion control algorithms under heavy traffic scenario to test how each of them reacts under extreme cases.

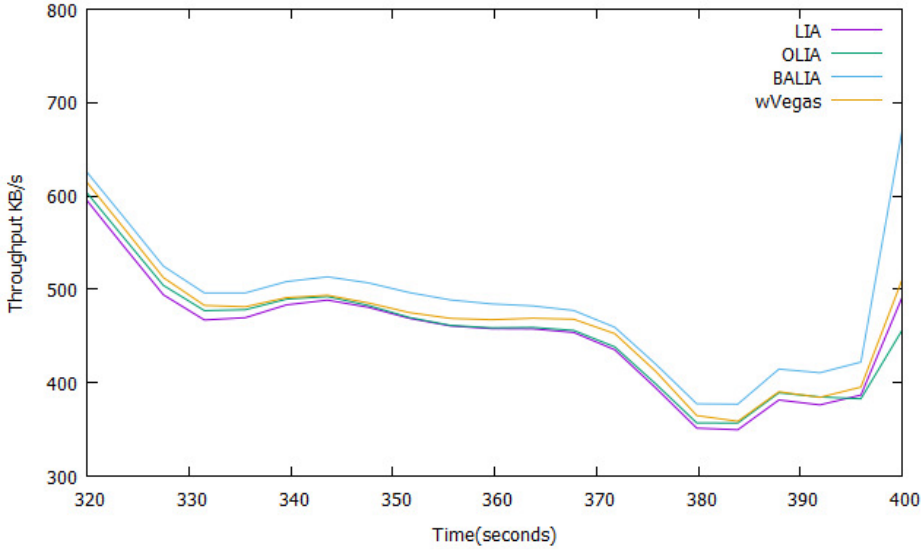

Figure 7: Throughput comparison between MPTCP congestion controls algorithms under Heavy Traffic Load

Figure 7 shows the comparison results, it clearly shows that BALIA outperforms other congestion algorithms when the paths are shared with heavy traffic that is not supported by MPTCP. This explains the trade-off between friendliness and responsiveness in the old congestion control implementations in LIA and OLIA. Wvegasoutperforms LIA and OLIA, this is because it uses packet questing delay as a congestion signal instead of loss based, and this makes wVegas more sensitive to the changes in the network congestions and can achieve more timely traffic shifting and faster convergence.

6. CONCLUSIONS

MPTCP is one of the most significant changes to TCP in the past years, it allows TCP streams to be split over multiple paths to improve the performance of TCP applications and increase throughput and reliability. MPTCP is implemented in the Linux Kernel and can be compiled and installed on most of Linux distributions. In this article, we did a performance analysis of MPTCP.It is shown experimentally that using MPTCP outperforms normal TCP and has many advantages in different scenarios. The throughput of different congestion MPTCP algorithms is also compared.

REFERENCES

[1] Postel, Jon. "RFC793: Transmission Control Protocol. USC." Information Sciences Institute 27 (1981): 123-150.

[2] Bonaventure, Olivier, Mark Handley, and Costin Raiciu. "An overview of Multipath TCP."; login: 37, no. 5 (2012): 17-23.

[3] Ford, C. Raiciu, and M. Handley. "TCP Extensions for Multipath Operation with Multiple Addresses" draft-ietf-mptcpmultiaddress (work in progress)." (2010).

[4] Scharf, Michael, and Alan Ford. Multipath TCP (MPTCP) application interface considerations. No. RFC 6897. 2013.

[5] Raiciu, C., M. Handley, and D. Wischik. "RFC 6356, Coupled Congestion Control for Multipath Transport Protocols." (2011).

[6] Khalili, Ramin, Nicolas Gast, and Miroslav Popovic. "Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP." (2013).

[7] Khalili, Ramin, Nicolas Gast, Miroslav Popovic, Utkarsh Upadhyay, and Jean-Yves Le Boudec. "MPTCP is not pareto-optimal: performance issues and a possible solution." In Proceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 1-12. ACM, 2012.

[8] Walid, A., Q. Peng, J. Hwang, and S. Low. "Balanced Linked Adaptation Congestion Control Algorithm for MPTCP." Working Draft, IETF Secretariat, Internet-Draft draft-walid-mptcpcongestion-control-03, July (2015).

[9] Cao, Yu, Mingwei Xu, and Xiaoming Fu. "Delay-based congestion control for multipath TCP." In Network Protocols (ICNP), 2012 20th IEEE International Conference on, pp. 1-10. IEEE, 2012.

[10] "MPTCP Linux Kernel Implementation." MultiPath TCP. Accessed March 01, 2016. http://mptcp.info.ucl.ac.be/.

[11] Wischik, Damon, Costin Raiciu, Adam Greenhalgh, and Mark Handley. "Design, Implementation and Evaluation of Congestion Control for Multipath TCP." In NSDI, vol. 11, pp. 8-8. 2011.

[12] Honda, Michio, Yoshifumi Nishida, Lars Eggert, Pasi Sarolahti, and Hideyuki Tokuda. "Multipath congestion control for shared bottleneck." In Proc. PFLDNeT workshop, pp. 19-24. 2009.

[13] Raiciu, Costin, Damon Wischik, and Mark Handley. "Practical congestion control for multipath transport protocols." University College London, London/United Kingdom, Tech. Rep (2009).

[14] "Ifstat(1) - Linux Man Page." Ifstat(1): Report InterFace STATistics. Accessed March 01, 2016. http://linux.die.net/man/1/ifstat.

AUTHORS

Hamzah M A Hijawi received his BS in computer system engineering from Birzeit University, Ramallah, Palestine, in 2011. He is currently working with Exalt technologies as a software engineer since 2010 for Cisco Systems, Inc. as a software engineer for deep packet inspection project NBAR, and currently pursuing his Master of Computer Science from Arab American University, Jenin, Palestine. His researches interest include computer networking, information security, wireless sensor network and data Mining.

Mohammad M N Hamarsheh received his BS in electrical and computer engineering from An-najah National University, Nablus, Palestine, in 1999, and his MS and PhD degrees in computer and communication engineering and communications and network engineering, in 2002 and 2006, respectively from University Putra Malaysia, Serdang, Malaysia. He worked as the head of the FBG fabrication unit with Photronix, Cyberjaya, Malaysia. He was involved in optical CDMA and fiber Bragg grating research and development. He worked as a lecturer with faculty of information science and technology, Multimedia University, Malaysia, from 2005 to 2008. He is currently an assistant professor at Arab American University, Jenin, Palestine. His research interests include computer networking and security, optical code division multiple access systems, fiber Bragg gratings, dense wavelength division multiplexing, and polarization effect on optical fiber communication.