The ISC Int'l Journal of Information Security

August 2019, Volume 11, Number 3 (pp. 113–122)

http://www.isecure-journal.org

Selected Paper at the ICCMIT'19 in Vienna, Austria

Evaluating Multipath TCP Resilience against Link Failures☆

Mohammed J.F. Alenazi [1,*]

[1] *College of Computer and Information Sciences, Department of Computer Engineering, King Saud University. Riyadh, Saudi Arabia*

ARTICLE INFO.

Keywords:
Path diversity, Resilient networks, Multi-homing, Multi-path TCP, Fault tolerance, Reliable data transfer, Handover, Performance evaluation.

Abstract

Standard TCP is the de facto reliable transfer protocol for the Internet. It is designed to establish a reliable connection using only a single network interface. However, standard TCP with single interfacing performs poorly due to intermittent node connectivity. This requires the re-establishment of connections as the IP addresses change. Multi-path TCP (MPTCP) has emerged to utilize multiple network interfaces in order to deliver higher throughput. Resilience to link failures can be better supported in MPTCP as the segments' communication are maintained via alternative interfaces. In this paper, the resilience of MPTCP to link failures against several challenges is evaluated. Several link failure scenarios are applied to examine all aspects of MPTCP including congestion algorithms, path management, and subflow scheduling. In each scenario, the behavior of MPTCP is studied by observing and analyzing the throughput and delay. The evaluation of the results indicates MPTCP resilience to a low number of failed links. However, as the number of failed links increases, MPTCP can only recover full throughput if the link failure occurs on the server side. In addition, in the presence of link failures, the lowestRTT MPTCP scheduler yields the shortest delivery time while providing the minimum application jitter.

© 2019 ISC. All rights reserved.

1 Introduction

The Internet has grown into a popular and critical infrastructure where millions of users access it for services including: e-commerce, education, entertainment, healthcare, and government services, etc. The convenience of mobility offered by wireless networks makes user access much more popular compared to wired access. A disruption on the availability of services due to network link failures can have significant adverse impact on the user's experience. The lack of access to critical online services can cause substantial financial loss or threaten user health and safety.

Thus, it is crucial to design and build resilient networks with the ability to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation [1]. One effective network faliure recovery strategy is to use path diversification between communicating nodes. If the primary network path fails, disjoint paths are used

☆ The ICCMIT'19 program committee effort is highly acknowledged for reviewing this paper.

* Corresponding author.

Email address: mjalenazi@ksu.edu.sa

ISSN: 2008-2045 © 2019 ISC. All rights reserved.

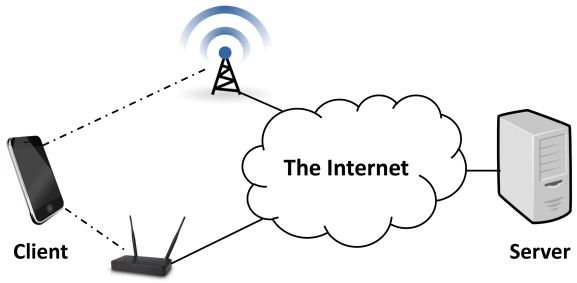


Figure 1. MPTCP Use Case.

to restore communication. Path diversification is an effective solution providing multiple alternative paths between disconnected pairs to gain high communication reliability [2]. The problem of finding several shortest paths between a given source and destination is known as the k shortest path problem (KSP), where k is the number of required paths. KSP has been widely used in many applications, including car navigation systems, robot motion planning, wireless sensor networks and video streaming [3–5]. Several algorithms have been introduced to provide path selections based on full-disjoint and partial-disjoint requirements, while maximizing diversity and minimizing incurred cost [6–9].

TCP uses a single path between communicating parties. In the event of path failure, the TCP connection times out and closes, requiring the setup of a new connection to resume service provision. Motivated to address this issue, the Internet Engineering Task Force (IETF) approved an Internet draft for MPTCP in 2011 to support multipath connectivity [10, 11]. MPTCP aims to improve connection resilience by switching traffic to alternative interfaces in the presence of path failure [11, 12]. In addition, MPTCP improves throughput by bandwidth aggregation over multiple interfaces. Several operating systems, including Linux and FreeBSD have available implementations for MPTCP [13]. Moreover, hosting companies and large IT vendors (e.g. Apple) are already providing support for MPTCP [14, 15].

Most smart phones are equipped with two interfaces that can access the Internet, namely, 802.11 and Long-Term Evolution (LTE). A common use-case for MPTCP that attracted Apple and other mobile-device vendors is to utilize such two interfaces to improve throughput as shown in Figure 1. Furthermore, MPTCP can offer performance improvements to the seamless smart phones handover among different access points/base stations. A previous study has shown that the MobileIP takes approximately ten seconds to handover from one node to another [16]. MPTCP shows great improvement for hand over delays of around three seconds [17].

In this paper, an extensive evaluation of MPTCP resilience to link failure is presented. The study is based on the Linux implementation of MPTCP which is composed of several configurable components affecting its performance. This includes congestion algorithms, number of interfaces, path management, and subflow scheduling. Several scenarios are considered in order to examine the effect of each component on the network resilience in the presence of link and interface failures. For each scenario, a number of performance metrics are observed and analyzed including throughput, total, delivery delay, and application jitter. Experimental results for each scenario are analyzed and guidelines for optimal MPTCP configuration to achieve a high resilience against links failures are presented.

The paper is organized as follows: Related work is reviewed in Section 2 followed Section 3 by a high-level description of the MPTCP protocol. In Section 4, discussion of the investigated MPTCP performance in response to link failures is provided. This includes providing details on the experimental tools, setup, network topology, and evaluation scenarios. Discussion of the results obtained is also presented. Finally, conclusions and outline of future work are presented in Section 5.

2 Related Works

In this section, we present the utliztion of k-shortest paths in the feild of communication networks. Then, we present several studies that utlize MPTCP to deliver packets via muliple interfaces.

2.1 K Shortest Paths

Traditional algorithms to find the shortest path or k-shortest paths including the Ford algorithm [18] and the Dijkstra algorithm [19], along with several improvements to tackle the issue of negative cycles. Several algorithms have been devised to determine the k shortest paths between a pair of nodes, based on selection objectives [20]. Among these algorithms, the most popular k-shortest path algorithm is proposed by Yen [21], which deals with non-negative link cost.

In the context of computer networks, path diversification has been studied and utilized to solve several communication problems. Algorithms proposed in previous research vary on their objective in determining diverse shortest paths. Some algorithms aim to determine topologically diverse paths while others specify geographically diverse paths or disjoint paths [22, 23]. The concept of diverse paths has been investigated to find diverse paths, k-diverse paths, and k-shortest diverse paths. The existing literature covers techniques based on shortest path algorithm with the incremen-

tal removal of used edges from graph transformations [24, 25]. Bhadari presents efficient algorithms to compute edge-disjoint and vertex-disjoint paths [26]. However, these algorithms are based on finding completely diverse paths. Bhandari also discusses an algorithm that finds the maximally diverse paths between a pair of nodes using a modified Dijkstra's algorithm.

Sitanayah et al. presented two k-shortest path algorithms for topology planning of wireless sensor network. The first algorithm, counting-paths, counts the number of disjoint paths from each sensor node to the sinks and finds the k disjoint paths. The second algorithm, named GRASP-ARP, is a local search algorithm to deploy a minimum number of additional relays at the possible candidate locations. Simulation results showed that their approaches requires fewer relay nodes for larger problems than other baseline k-shortest path algorithms. Moreover, their algorithms showed significant improvement on computation time [27].

Cheng et al. [23] presented a protocol stack, ResTP–GeoDivRP, which utilizes geographical multipaths to cope with node and links failures. This stack combines the transport and network layers to provide reliable services to the application layer. It detects challenges and bypass the challenged regions to improve throughput. Their approach was evaluated in backbone networks with several area-based challenges. The results showed that the proposed stack achieved higher throughput and robustness than MPTCP in the face of geographically-correlated challenges.

2.2 Mutipath Transport

Several studies have considered the performance and resilience of MPTCP. Nguyen et al. [12] presented an evaluation of MPTCP examining its bandwidth utilization, end-to-end delay, and packets reordering. Their testbeds supported three two-path scenarios: Ethernet-only, Ethernet and 802.11, 802.11 and 3G. Evaluation of the results indicated that MPTCP performance is highly influenced by interface selection. An evaluation of MPTCP's congestion algorithms (OLIA, BALIA, wVegas, and LIA) is presented by Nguyen et al. [28]. The study results indicated that wVegas throughput outperformed the other congestion algorithms in a multipath environment. However, wVegas did not yield the best throughput in a single path setting. In addition, the network selection for the first subflow initialization has a significant impact on the achieved throughput. The throughput with 802.11 outperformed LTE, with an improvement of approximately 200%.

Scheduler performance evaluation has also been considered in the literature. Paasch et al. [29] introduced a framework to experimentally evaluate schedulers in a wide variety of environments in both emulated and real-world experiments. Their evaluation results showed that there are two attributes associated with bad schedulers. Firstly, scheduler interface selection with high-RTT can increase head-of-line blocking. Another attribute of a bad scheduler is the limited receive-window, which may not fully accommodate all received subflows. Moreover the authors introduce a scheduler that favors subflows with the lowestRTT to reduce delay-jitter compared to a simple round-robin (RR) scheduler.

Alheid et al. [30] investigated the behavior of MPTCP and its performance in networks with high out-of-order packets. They considered the performance of MPTCP with four TCP packet reordering mechanisms: D-SACK, Eifel, TCP-DOOR, and F-RTO. Their results showed that MPTCP throughput can be improved using the DSACK approach. On the other hand, F-RTO provides better throughput with an uncoupled controller and has lower memory requirements compared to the other mechanisms.

Previous work has also proposed algorithms and techniques to improve the performance of MPTCP. Lee et al. [31] proposed the Receive Window Adaptive Multipath TCP (RWA-MPTCP) algorithm to provide a better throughput performance in mobile networks. Their results showed that using RWA-MPTCP increases throughput while decreasing RTT variation. Sandri et al. [32] introduced the Multiflow approach that uses MPTCP and Open Flow. Their approach aims to select subflows with disjoint paths, i.e. no shared links. For evaluation, they deployed Multiflow in a testbed where shared bottlenecks occur at the endpoints of the link. They demonstrated that Multiflow improved end-to-end throughput and link-failure resilience [33]. Demir et al. [34] proposed MPTCP-H, an extension to MPTCP, to improve server resilience against Denial-of-Service (DoS) and Distributed DoS (DDoS) attacks. They compared MPTCP-H to UDP and standard TCP. The results showed that MPTCP-H outperformed both UDP and TCP in terms of latency. Their results also indicate that MPTCP-H, with its lightweight mechanism, can mitigate the attacks originating from inside the substation network. Dong et al. [35] introduced a design and implementation for LAMPS, an MPTCP scheduler. LAMPS selects subflows based on loss and delay in order to maintain steady performance at various traffic patterns. Their experimental evaluation showed that LAMPS minimizes latency while using a low amount of memory. Moreover, the scheduler handles the high packet loss rate with optimal bandwidth usage.

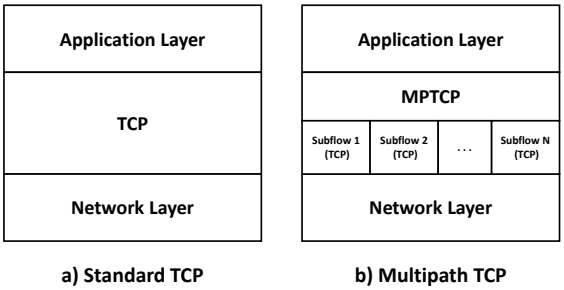


Figure 2. MPTCP architecture comparison with standard TCP.

3 MPTCP Overview

Standard TCP communication uses a single pair of IP addresses to deliver in-order packets between two communicating parties. Thus, there is a limitation in using one interface even if multiple interfaces are available at each end of the TCP connection. MPTCP utilizes multiple interfaces to establish simultaneous sub-connection paths between peers. By exploiting all available interfaces at both ends of the communication, MPTCP aims to improve network resilience to interface failure (unavailability) and enhance throughput [11]. The MPTCP path is defined by two endpoints, the client IP and server IP addresses. The subflow in MPTCP is a single-path standard TCP, in which the component takes segments from the packet scheduler transmitted to the receiver. MPTCP is implemented at the transport layer and supported by the same socket used in standard TCP for application layer transparency. Standard TCP headers are used for each subflow to be transparent to the network layer as shown in Figure 2. The main implementation components of MPTCP are described in the following sections.

3.1 Path Manager

The path manager function is responsible for detecting and using multiple IP-pairs between the two communicating hosts. It can utilize multiple IP addresses at one or both hosts based on the selected path manager. For example, if the path manager is set to *fullmesh*, there will be a subflow between each two pairs. *Ndiffports*, an alternative path manager, specifies the number of subflows allowed to be used across the same pair of IP addresses. The *binder* path manager is used when utilizing multiple geographically distributed gateways over the Internet [36]. In this paper, the path manger is set to full-mesh since it creates a subflow between each pair of IP-addresses, which ensures that all interfaces are utilized.

3.2 Subflow Scheduler

The packet scheduler breaks application messages into byte stream segments to be transmitted via available subflows at the sender side. On reception, MPTCP manages the receiver byte stream from subflows to allow segments to be correctly re-ordered. The packet scheduler acquires information on available subflows from the path manager. Queued segments are then sent into the next selected subflow. Various types of schedulers have been used for subflow selection. This includes lowestRTT, round robin, and redundant schedulers. The lowestRTT, the default scheduler for MPTCP, sends segments on subflows with the lowest RTT until their congestion window reaches its threshold. Next, the segments are sent over the next lower RTT subflows. With the redundant scheduler, segments are duplicated and sent over all available subflows with the objective to achieve lowest latency by sacrificing link bandwidth [11].

3.3 Congestion Algorithms

MPTCP uses congestion control across subflows to ensure the fair distribution of bandwidth across multiple connections in a shared bottleneck link [10]. Although, standard TCP congestion algorithms can be used to handle each subflow in MPTCP, an unfair share occurs in multiple path flows in a common bottleneck. Thus, several MPTCP congestion algorithms have been implemented to ensure high throughput, fairness, and friendliness. In the current paper, various Linux implemented congestion control algorithms are used, including: the Linked Increased Algorithm (LIA) [37, 38], the Opportunistic Linked Increases Algorithm (OLIA) [39], the Balanced Linked Adaptation Algorithm (balia) [40], and the weighted Vegas algorithm (wVegas) [41].

3.4 Interface Failure Handling

In addition to utilizing multiple interfaces, MPTCP is designed to provide seamless transition from one interface to another in case of interface removal. When full-mesh is used, a subflow is created between each interface, for example, given two hosts with three interfaces installed in each, as shown in Figure 3. The full-mesh path manager allocates nine subflows. MPTCP selects some or all of these subflows for transmitting segments based on the selected subflow scheduler. In case of a link failure, the operating system notifies MPTCP via `mptcp_v4_rem_raddress` [11],

4 Results and Discussions

In this section, the evaluation work carried out to investigate MPTCP performance in response to link

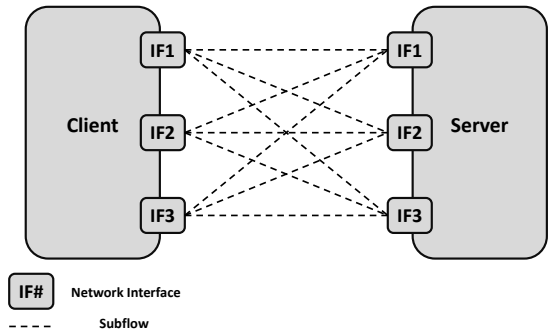


Figure 3. MPTCP example of three interfaces and full-mesh path manger.

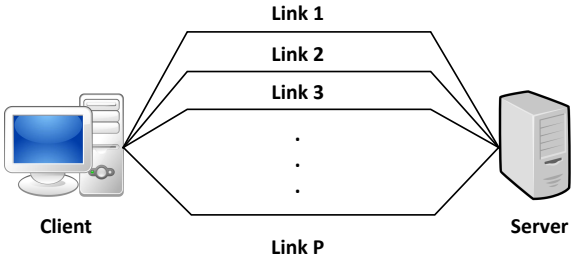


Figure 4. Evaluation Topology in Mininet Emulator.

Table 1. Default Emulation Parameters.

Parameter	Values
Emulator	Mininet 2.2.2
Operating System	Ubuntu 17.10
Transport Protocol	MPTCP 0.94
CPU	eight-core 3.5 GHz
Memory	16GB
Link Bandwidth	1 Mbps
Link Delay	10 ms
MPTCP Congestion	MPTCP-lia
MPTCP Scheduler	LowestRTT
MPTCP Path Manager	Full-mesh

failures is provided. This includes details on the experimental tools, setup, network topology, and evaluation scenarios. Discussions of the obtained results is also provided.

4.1 Experimental Environment

In all experiments performed, a stable MPTCP 0.94 implementation is used on Ubuntu 17.10. The experimental host has an eight-core 3.5 GHz processor with 16GB of RAM. The data traffic is generated using iperf. Networks are emulated using Mininet 2.2.2, which delivers network-connected virtual Linux hosts with a given topology. The emulation parameters and testing network topology used in all experiments are shown in Table 1 and Figure 4, respectively. lowestRTT (MPTCP-defualt), is used as the scheduler scheme when evaluating congestion algorithms. MPTCP-lia is used as a congestion algorithm when evaluating various scheduler schemes.

The testing topology includes two hosts, denoted as client and server, connected by eight links. Each link has a bandwidth of 8Mbps with a 10ms delay. The maximum number of supported interfaces in the current implementation of MPTCP is eight. Comparing standard TCP and MPTCP to send data generated via iperf, it is observed that MPTCP uses all eight links to transfer data reliably between the client and server while TCP only uses one interface. Clearly, it can be seen that MPTCP has almost 700% throughput improvement over standard MPTCP in the same network as shown in Figure 5. This shows how MPTCP utilizes available interfaces to significantly improve throughput.

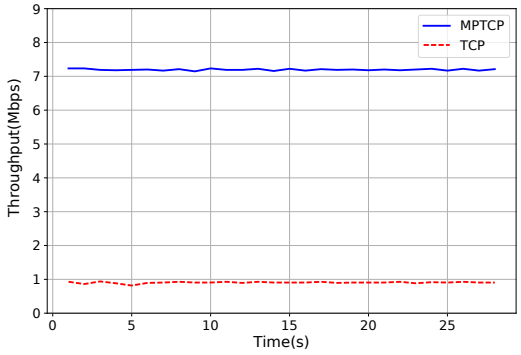


Figure 5. MPTCP and TCP Comparison.

4.2 Congestion Algorithms

Experiments are performed to study the effect of MPTCP congestion algorithms on link failure resilience. Four MPTCP congestion algorithms are evaluated: MPTCP-lia, MPTCP-olia, MPTCP-balia, and MPTCP-wVegas (disscussed in Section 3.3). For each congestion algorithm, iperf is used to send data from the server to the client while MPTCP is enabled. During the data transfer, a link failure is applied which removes 50% of the available links between the 20th and 40th seconds of the MPTCP connection. The throughput generated when applying the four congestion algorithms is depicted in Figure 6. MPTCP-lia and MPTCP-balia utilize all interfaces before they fail with a steady throughput of 7 Mbps. MPTCP-olia and MPTCP-wVegas have a throughput fluctuation between 6 Mbps and 7 Mbps up to

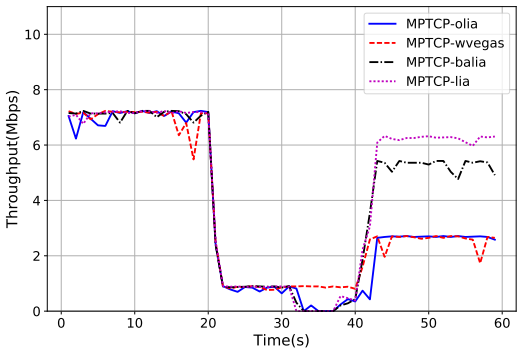


Figure 6. MPTCP Congestion Control Comparison.

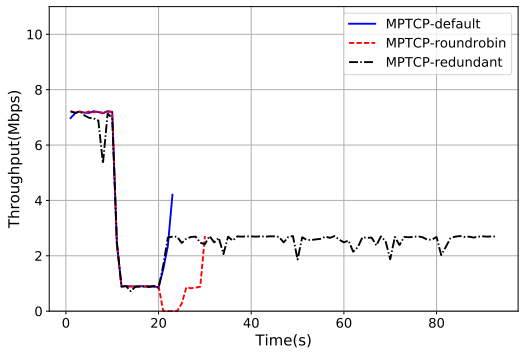


Figure 7. Scheduler Algorithms Throughput Comparison.

the 20th second. Once link failure is applied, it can be noted that although failure is imposed on only 50% of the available links, the throughput drops to about 1 Mbps. This represents 85% of the original throughput. It is expected that full throughput recovery occurs once the linked failures are again operational. However, none of the congestion algorithms recover full throughput after the links are up and running again. It is observed that MPTCP-lia yields the best recovery from link failures. On the other hand, MPTCP-olia and MPTCP-wVegas provide the worst link failure resilience.

4.3 Scheduler Schemes

Three MPTCP scheduling schemes are considered: lowestRTT (MPTCP-default), round-robin, and redundant (discussed in Section 3.2). The performance measures used to evaluate each scheduler scheme are throughput, total delivery time of a data unit and application-jitter (variance of data units arrival time). For each scheduler, 15MB of data is sent over the 8 interfaces from the client to the server. While sending the data, link failure is applied between the 10th and 20th seconds of the MPTCP connection.

4.3.1 Throughput and Total Delivery Time

The throughput results of each scheduler against link failure are shown in Figure 7. It is observed that all the scheduling methods eventually deliver the 15MB data, with a variation on the total delivery time. Undoubtedly, the lowestRTT scheduler yields the shortest delivery time of approximately 23 seconds. This is due to a correlation between link failure and RTT values. Once the interfaces fail, the RTT values start to get larger and become less favored for selection by the scheduler. Thus, the lowestRTT scheme has the best network resilience among the three studied scheduling schemes. The round-robin method delivers the entire data within 30 seconds. It does not react

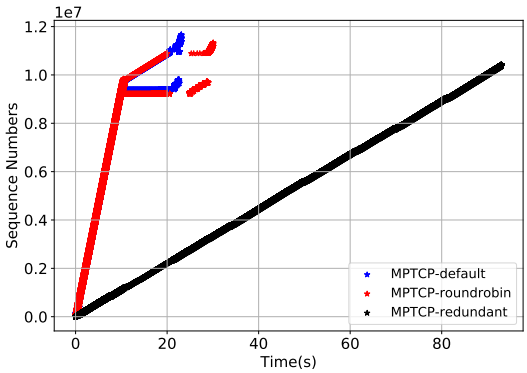


Figure 8. Scheduler Algorithms Sequence Numbers Comparison.

quickly to the failure and recovery of interfaces, as it keeps sending data to failed interfaces even if they incur high RTT values. The redundant method yields the worst total delivery time of approximately 92 seconds. This is because it wastes the available capacity with redundant data.

4.3.2 In-Order Delivery

The arrival time of segment sequence numbers during link failure when the three schedulers are applied is shown in Figure 8. The in-order delivery of segments affecting application layer jitter can be inferred by sequence numbers. All three schedulers have smooth in-order delivery with no significant application jitter before link failure. However, after the links start to fail at the 10th second, the segments begins to arrive in out-of-order fashion for the lowestRTT and round robin schedulers. The redundant scheduler sequence numbers do not fluctuate during the link failures, indicating that segments are delivered in-order even when the network experiences failed links. For applications that do not require a fixed bandwidth, such as VoIP and live streaming, the redundant scheduler provides the lowest application jitter, leading to the

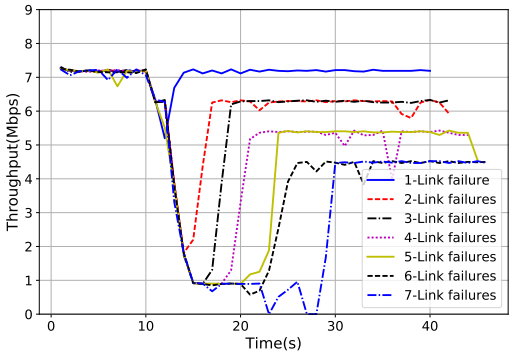


Figure 9. Client-only port failure.

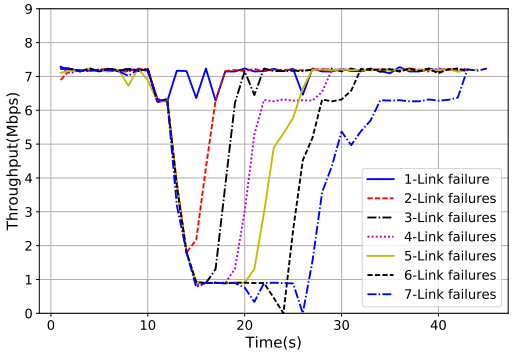


Figure 10. Server-only port failure.

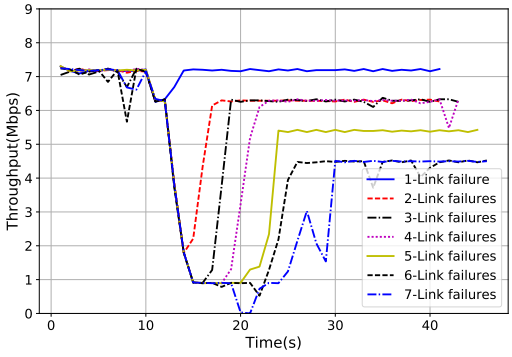


Figure 11. Client and server port failure.

best user-experience.

4.4 Multiple Link Failures and Recovery

The experimental setup is designed to fail a specified number of interfaces, denoted as n, failing one interface every two seconds. The failed interfaces are selected deterministically based on their order. For example, if $n = 2$, then interfaces 1 and 2 are removed. Once all n interfaces are down, one interface is brought back every two seconds until all the n previously failed interfaces are fully operational. The interval between each link break and restoration is set to two seconds in order to provide enough time to MPTCP to be notified by the operating system about the interface removal. Multiple experimental runs are performed with $n = [1, 2, 3, 4, 5, 6, 7]$ based on three arrangements: client-only port failure, server-only port failure, and client-and-server port failures. As each host has eight interfaces in the evaluation topology used, it is worth noting that they are never all put into fail state together to avoid closing the MPTCP connection due to timeout.

4.4.1 Client-only Port Failure

In this setup, the interfaces fail at the client end while the server interfaces are operational. Thus, only the MPTCP client notices the port removals while the server keeps advertising its disconnected IP addresses. The first interface failure begins at the 10th second of the experiment. The result of applying the seven failure scenarios, 1-Link to 7-Link failures, is shown in Figure 9. Before applying link failure, until the 10th second of the experiment, the throughput is approximately 7.2 Mbps. For the 1-Link failure scenario, it can be seen that the throughput decreases to almost 5 Mbps as the link fails. When the link is recovered after two seconds, the throughput increases back to the original value. This shows that with a failure duration of two seconds, the MPTCP does not time out and it utilizes the recovered link with full advantage. However, with the 2-Link and 3-Link failure scenarios, it is observed that the throughput does not fully recover staying bellow 6.5 Mbps after the links are back to operational status. Analyzing the other link failure scenarios, it can be clearly seen that as the number of failing links increases, MPTCP poorly utilizes the recovered links as depicted in Figure 9. By observing the pcap traces of these experiments, it was found that sub-flows associated with the removed and recovered interfaces are reused for the 1-link failure experiment. However, not all subflows are reused in a higher number of links failures. This indicates that the operating system correctly notifies MPTCP of interface removal. Thus, MPTCP does not utilize the recovered interfaces correctly with more than two interfaces.

4.4.2 Server-only Port Failure

An alternative arrangement is studied with the interface failure occurring only at the server end while the client interfaces are operational. Hence, only the server MPTCP notices the removal of interfaces while the client continues to advertise its disconnected IP addresses. The same seven failure scenarios are ap-

plied, 1-Link to 7-Link failures, beginning at the 10th second of the experiment. The throughput was observed to be approximately 7 Mbps before the link failures. At the 10th second, the 1-Link failure throughput drops to 6 Mbps, since only one 1 Mbps link fails, and once the link is restored the throughput returns to the original rate. For the worst scenario, where seven links fail, the throughput is almost 7 Mbps before the link failures. At the 10th second, the throughput drops to approximately 1 Mbps since seven links are removed. After the links are restored, the throughput is also restored to the original rate. Unlike the client-only port failures, it is observed as shown in Figure 10, that the throughput is fully recovered for all the link failures. Evidently, this behavior indicates MPTCP implementation issues, as no difference is expected in MPTCP performance in response to client and server link failures.

4.4.3 Client and Server Port Failures

Failure is applied to both the client and server, with both sides stopping to advertise their disconnected IP addresses. The result of applying the seven link removal scenarios is shown in Figure 11. For 1-Link failure, the throughput drops as the interface is removed and is fully recovered when the link is back to operational status. However, as the number of failed links increases, the observed MPTCP recovery is worse off. In fact, only in the 1-Link port failure scenario a full throughput recovery is achieved. In none of the other scenarios a full recovery occurs once links are back to operational mode.

The results indicate that MPTCP is resilient to link-failure with a low number of failed links. Evidently, in the three arrangements discussed above, it can be seen that MPTCP fully recovers from link failures with only one interface. However, as the number of failed links increases, a performance variation is noticed based on which side of the communication the failure occurs. MPTCP can only recover full throughput if the link failures occur only at the server side. Throughput is not recovered when the failed interfaces are client-based. Given the fact that MPTCP can fully recover against 7-link failures at the server side, and fail to fully recover if the failures occur at the client side, it can clearly be seen that there are issues with the implementations. It is believe that this asymmetric MPTCP behavior and performance at both communication ends might be due to an issue with the MPTCP Linux implementation. It is recommended to resolve such issues in later versions of MPTCP.

5 Conclusions and Future Work

Traditional Internet is heavily based on Standard TCP which is used by almost all online applications as a reliable transfer protocol. A design limitation in the standard TCP is the inability to support more than one connection, even if multiple communication interfaces are available. This makes TCP-supported communication and applications relying on it more prone to failure, undermining the user experience. By utilizing multiple interfaces, MPTCP has emerged to provide seamless handover during link failures and to support higher throughput.

In this paper, a comprehensive evaluation of Linux MPTCP implementation resilience against link failures is provided. Various evaluation scenarios are considered in order to examine MPTCP's congestion algorithms, path management, subflow scheduling against link failures. Measurements of MPTCP throughput and end-to-end delays are collected and analyzed during link failures and after their recovery. The results indicate that MPTCP is resilient to link-failurs with a low number of failed links. However, as the number of failed links increases, MPTCP can only recover full throughput if the link failure occurs at the server side. The analysis of the results indicates that the lowestRTT scheduler yields the best network resilience against network failures. In addition, the redundant scheduler provides in-order segments during the link failures making it the optimal choice for real-time and VoIP applications.

For future work, evaluation on real-world testbeds such as PlanetLab are planed [42]. The behavior of MPTCP in a geographically distributed environment and high-speed networks will be evaluated.

5.1 Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through the Research Project No. R5-16-03-03.

References

[1] James P. G. Sterbenz, David Hutchison, Egemen K. Çetinkaya, Abdul Jabbar, Justin P. Rohrer, Marcus Schöller, and Paul Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*, 54(8):1245–1265, 2010.

[2] Justin P. Rohrer, Abdul Jabbar, and James P.G. Sterbenz. Path Diversification for Future Internet End-to-End Resilience and Survivability. *Springer Telecommunication Systems*, 56(1):49–

67, May 2014.

[3] Yao Wang, Shivendra Panwar, Shunan Lin, and Shiwen Mao. Wireless video transport using path diversity: Multiple description vs layered coding. In *Proceedings of the International Conference on Image Processing*, volume 1, pages I–21–I–24, 2002.

[4] Wilton Henao-Mazo and Angel Bravo-Santos. Finding diverse shortest paths for the routing task in wireless sensor networks. *Proc. ICSNC*, pages 53–58, 2012.

[5] A.C. Begen, Y. Altunbasak, and O. Ergun. Multi-path selection for multiple description encoded video streaming. In *Proceedings of the IEEE International Conference on Communications (ICC '03)*, volume 3, pages 1583–1589, May 2003.

[6] Eiji Oki, Nobuaki Matsuura, Kohei Shiomoto, and Naoaki Yamanaka. A disjoint path selection scheme with shared risk link groups in GMPLS networks. *IEEE Communications Letters*, 6(9):406–408, September 2002.

[7] Yufei Cheng, M. Todd Gardner, Junyan Li, Rebecca May, Deep Medhi, and James P.G. Sterbenz. Optimised Heuristics for a Geodiverse Routing Protocol. In *Proceedings of the IEEE 10th International Workshop on the Design of Reliable Communication Networks (DRCN)*, pages 1–9, Ghent, Belgium, April 2014.

[8] Shweta Jain and Samir R Das. Exploiting path diversity in the link layer in wireless ad hoc networks. *Ad Hoc Networks*, 6(5):805–825, 2008.

[9] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. In search of path diversity in ISP networks. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC '03)*, pages 313–318, New York, NY, USA, 2003. ACM.

[10] Alan Ford, Costin Raiciu, Mark Handley, Sebastien Barre, and Janardhan Iyengar. Architectural guidelines for multipath tcp development. RFC 6824, 2011.

[11] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP extensions for multipath operation with multiple addresses. RFC 6824 (Experimental), January 2013.

[12] Sinh Chung Nguyen, Xiaofei Zhang, Thi Mai Trang Nguyen, and Guy Pujolle. Evaluation of throughput optimization and load sharing of multipath tcp in heterogeneous networks. In *Wireless and optical communications networks (WOCN), 2011 eighth international conference on*, pages 1–5. IEEE, 2011.

[13] Nigel Williams. Implementing a multipath transmission control protocol (mptcp) stack for freebsd with pluggable congestion and scheduling control. 2016.

[14] Olivier Bonaventure and S Seo. Multipath tcp deployments. *IETF Journal*, 12(2):24–27, 2016.

[15] Olivier Mehani, Ralph Holz, Simone Ferlin, and Roksana Boreli. An early look at multipath tcp deployment in the wild. In *Proceedings of the 6th International Workshop on Hot Topics in Planet-Scale Measurement*, HotPlanet '15, pages 7–12, New York, NY, USA, 2015. ACM.

[16] S. K. Sivagurunathan, J. Jones, M. Atiquzzaman, Shaojian Fu, and Yong-Jin Lee. Experimental comparison of handoff performance of sigma and mobile ip. In *HPSR. 2005 Workshop on High Performance Switching and Routing, 2005.*, pages 366–370, May 2005.

[17] D. Nunes, D. Raposo, D. Silva, P. Carmona, and J. S. Silva. Achieving human-aware seamless handoff. In *2015 International Conference on Distributed Computing in Sensor Systems*, pages 254–259, June 2015.

[18] L. Ford. Network flow theory. 1956.

[19] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[20] MH MacGregor and WD Grover. Optimized k-shortest-paths algorithm for facility restoration. *Software: Practice and Experience*, 24(9), 1994.

[21] Jin Y Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716, 1971.

[22] Wayne D. Grover. *Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.

[23] Y. Cheng, T. A. N. Nguyen, M. M. Rahman, S. Gangadhar, and J. P. G. Sterbenz. Geodiverse routing protocol with multipath forwarding compared to mptcp. In *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 142–149, Sept 2016.

[24] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4(2), 1974.

[25] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2), 1984.

[26] Ramesh Bhandari. *Survivable Networks: Algorithms for Diverse Routing*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[27] Lanny Sitanayah, Kenneth N Brown, and Cormac J Sreenan. A fault-tolerant relay placement algorithm for ensuring k vertex-disjoint shortest paths in wireless sensor networks. *Ad Hoc Networks*, 23:145–162, 2014.

[28] Kien Nguyen, Mirza Golam Kibria, Kentaro Ishizu, and Fumihide Kojima. A study on performance evaluation of multipath tcp implementa-

tions. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, SoICT 2017, pages 242–248, New York, NY, USA, 2017. ACM.

[29] Christoph Paasch, Simone Ferlin, Ozgu Alay, and Olivier Bonaventure. Experimental evaluation of multipath tcp schedulers. In *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, CSWS '14, pages 27–32, New York, NY, USA, 2014. ACM.

[30] Amani Alheid, Dritan Kaleshi, and Angela Doufexi. Performance evaluation of mptcp in indoor heterogeneous networks. In *Proceedings of the 2014 First International Conference on Systems Informatics, Modelling and Simulation*, SIMS '14, pages 213–218, Washington, DC, USA, 2014. IEEE Computer Society.

[31] Jin Seong Lee and Jaiyong Lee. Multipath tcp performance improvement in mobile network. In *Ubiquitous and Future Networks (ICUFN), 2015 Seventh International Conference on*, pages 710–714. IEEE, 2015.

[32] M. Sandri, A. Silva, L. A. Rocha, and F. L. Verdi. On the benefits of using multipath tcp and openflow in shared bottlenecks. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 9–16, March 2015.

[33] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[34] Kubilay Demir and Neeraj Suri. Towards ddos attack resilient wide area monitoring systems. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ARES '17, pages 99:1–99:7, New York, NY, USA, 2017. ACM.

[35] E. Dong, M. Xu, X. Fu, and Y. Cao. Lamps: A loss aware scheduler for multipath tcp over highly lossy networks. In *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pages 1–9, Oct 2017.

[36] Luca Boccassi, Marwan M. Fayed, and Mahesh K. Marina. Binder: A system to aggregate multiple internet gateways in community networks. In *Proceedings of the 2013 ACM MobiCom Workshop on Lowest Cost Denominator Networking for Universal Access*, LCDNet '13, pages 3–8, New York, NY, USA, 2013. ACM.

[37] Costin Raiciu, Mark Handley, and Damon Wischik. Coupled congestion control for multipath transport protocols. Technical report, 2011.

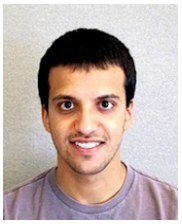
[38] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, volume 11, pages 8–8, 2011.

[39] Ramin Khalili, Nicolas Gast, Miroslav Popovic, et al. Opportunistic linked-increases congestion control algorithm for mptcp. 2013.

[40] Q. Peng, A. Walid, J. Hwang, and S. H. Low. Multipath tcp: Analysis, design, and implementation. *IEEE/ACM Transactions on Networking*, 24(1):596–609, Feb 2016.

[41] Yu Cao, Mingwei Xu, and Xiaoming Fu. Delay-based congestion control for multipath tcp. In *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, pages 1–10. IEEE, 2012.

[42] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.



Mohammed J.F. Alenazi is an Assistant Professor of Computer Engineering at King Saud University. He received his Ph.D. in Computer Science from the University of Kansas in 2015. He received his B.S. and M.S. degrees in Computer Engineering from the University of Kansas in 2010 and 2012 respectively. His research interests are in resilient networks, software defined networks, Internet of Things, multipath transport protocols, and wireless sensor networks. He is a member of the IEEE, Communications Society and member of ACM SIGCOMM.