Proceedings of the 11th International Conference on Applied Informatics
Eger, Hungary, January 29–31, 2020, published at http://ceur-ws.org

The Effects of Different Congestion Control Algorithms over Multipath Fast Ethernet IPv4/IPv6 Environments

Szabolcs Szilágyi, Imre Bordán

Faculty of Informatics, University of Debrecen, Hungary
szilagyi.szabolcs@inf.unideb.hu
bordanimre@gmail.com

Abstract

The TCP has been in use since the seventies and has later become the predominant protocol of the internet for reliable data transfer. Numerous TCP versions has seen the light of day (e.g. TCP Cubic, Highspeed, Illinois, Reno, Scalable, Vegas, Veno, etc.), which in effect differ from each other in the algorithms used for detecting network congestion.

On the other hand, the development of multipath communication technologies is one today's relevant research fields. What better proof of this, than that of the MPTCP (Multipath TCP) being integrated into multiple operating systems shortly after its standardization.

The MPTCP proves to be very effective for multipath TCP-based data transfer; however, its main drawback is the lack of support for multipath communication over UDP, which can be important when forwarding multimedia traffic. The MPT-GRE software developed at the Faculty of Informatics, University of Debrecen, supports operation over both transfer protocols.

In this paper, we examine the effects of different TCP congestion control mechanisms on the MPTCP and the MPT-GRE multipath communication technologies.

Keywords: congestion control, multipath communication, MPTCP, MPT-GRE, transport protocols.

1. Introduction

In 1974, the TCP was defined in RFC 675 under the Transmission Control Program name. Later, the Transmission Control Program was split into two modular

architectures, namely the Transmission Control Protocol (TCP), and the Internet Protocol (IP), which are the two corner stones of what we call internet today.

The TCP is a connection-oriented, reliable, ordered end-to-end protocol that operates in the fourth (transfer) layer of the ISO OSI reference model, and in the third layer of the quad-layer TCP/IP protocol stack. It supports the connection and communication of processes running on different endpoints with reliable transfer of dataflows. The processes are able to connect to each other using so-called sockets that are identified by an IP-address and port number pair for each endpoint. Thus, a communication session can be defined with the help of a socket-pair.

In the beginning, the TCP only provided reliable, ordered dataflow transfer without any included congestion control mechanisms. Later on, with the internet gaining in popularity, problems started to arise due to the ever more frequently occurring network congestions. Network congestion is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle. Typical effects include queueing delay, packet loss or the blocking of new connections. A consequence of congestion is that an incremental increase in offered load leads either only to a small increase or even a decrease in network throughput [1]. Network protocols that use aggressive retransmissions to compensate for packet loss due to congestion can increase congestion, even after the initial load has been reduced to a level that would not normally have induced network congestion. Such networks exhibit two stable states under the same level of load. The stable state with low throughput is known as congestive collapse. Congestion control modulates traffic entry into a telecommunications network in order to avoid congestive collapse resulting from oversubscription. This is typically accomplished by reducing the rate of packets, by preventing senders from overwhelming the network [2].

In the early eighties, it was on Jakobson's suggestion (see [3]) that a rudimentary congestion control mechanism was implemented on a system level into the TCP. This version was named TCP Tahoe. As the internet continued to gain ground and different communication technologies and trends started to emerge, the initial TCP version proved to be less effective. E.g. current day fast, and high delay networks have different system requirements, than for example wireless data transfer. Numerous TCP versions have seen the light of day since the eighties, which effectively proves that no TCP version was yet conceived that can be applied in every type of network environment while satisfying all the requirements. Our experiments in multipath environments were limited to the following versions: TCP Vegas, TCP Reno, Scalable TCP (SCTCP), TCP Veno, HighSpeed TCP (HSTCP), TCP-Illinois, TCP Cubic. More detailed information about the operating principles of the TCP versions can be found in the following papers: [4, 5, 6, 7, 8, 9, 10, 11, 12].

The use of multipath communication technologies can greatly improve the effectiveness of infocommunications systems, primarily with regards to maximum data transfer rate, redundancy, and roaming capabilities. Several multipath solutions exist; however, probably the most prominent one is the Multipath TCP (MPTCP – [13]), which has lately been integrated into their operating systems by Apple and

Cisco as well. The MPTCP is practically the multipath extension of the traditional TCP, building its operating principle on the use of TCP-subflows. Besides its numerous advantages, the MPTCP has a main drawback as well, namely that it supports communication over TCP only, not allowing the use of the UDP. Multimedia applications, however, typically use the UDP. The development of the multipath communication technology MPT-GRE (see [14]) building on a completely new architecture set off to eliminate this problem. Practically speaking, a new tunnel layer and interface was introduced. The applications operate conventionally above this layer, handing their transferrable data over to the logical interface of the tunnel. The MPT-GRE software is then responsible for mapping the data traffic to the physical interfaces on both the sending and receiving end. The MPT-GRE is virtually the multipath extension of the GRE standard, which supports communication over both the TCP and the UDP, and both IPv4 and IPv6. The performance analysis of these multipath systems is discussed in the following papers: [15, 16, 17].

In this paper, we examine the effect of different congestion control algorithms on the MPTCP and the MPT-GRE solutions with regards to file-transfer performance and CPU utilization, laid out in the following structure: in the second chapter we introduce the measurement environment, then go on to present our measurement results in chapter three. The final chapter contains our conclusions together with the identified development possibilities and suggestions.

2. The measurement environment

Our measurements were carried out in a quad-path Fast Ethernet environment provided to us by the Faculty of Informatics, University of Debrecen. As shown on Figure 1, the environment with independent paths was set up between the network interfaces of two server computers. As by default the network interfaces supported 1 Gbps speeds, we limited their throughput to 100 Mbps on all four interface-pairs. The specifications of the machines were as follows:

- Gigabyte Z77-D3H motherboard with Intel Z77 chipset
- 8 threads (4 physical cores) on an Intel Core i7-3770K 3.50GHz processor
- 4x4GB 1600MHz DDR3 SDRAM
- Intel PT Quad 1000 Gigabit Ethernet server interface
- Ubuntu 16.04 LTS (Xenial Xerus) 64-bit operating system with 4.4.0-62-generic Linux kernel module

We used CAT6 STP cables to establish the paths. The motherboards' integrated network interface cards were reserved for remote access and they were disabled during the measurements, removing any impact they could have had on the measurement process.

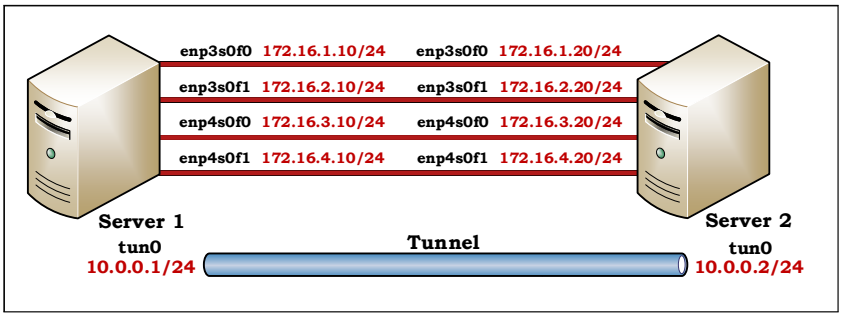

Figure 1: Quad-path Fast Ethernet measurement environment

2.1. Using different versions of TCP

Like it shows in the system specifications introduced above, Linux Ubuntu operating system was installed on both servers. The modern Linux environments use the TCP Cubic version by default. The currently enabled congestion control algorithm can be displayed using this command:

```
sysctl net.ipv4.tcp_congestion_control
```

The list of TCP congestion control algorithms currently supported by the operating system can be queried via the following command:

```
sysctl net.ipv4.tcp_available_congestion_control
```

A specific congestion control algorithm (e.g. TCP Vegas) can be selected as shown here:

```
sudo sysctl net.ipv4.tcp_congestion_control=vegas
```

2.2. Preparation of MPT-GRE and MPTCP based measurements

Afterwards, the MPT-GRE and the MPTCP multipath environments were installed and configured per the instructions available on their respective developer websites. The MPT-based and MPTCP-based measurements were carried out separately, as the latter being a kernel-level implementation required loading a dedicated kernel module on system boot. We performed the following three measurement types in both cases: *iperf3* throughput measurement of memory-to-memory data transfer, FTP-based file transfer throughput measured using *wget*, and CPU utilization readings while using different TCP versions. The measurements were automated using Python scripts that are publicly available[1]. Each measurement was repeated ten

[1]Python scripts – `https://irh.myqnapcloud.com:/share.cgi?ssid=0wbCCTP`

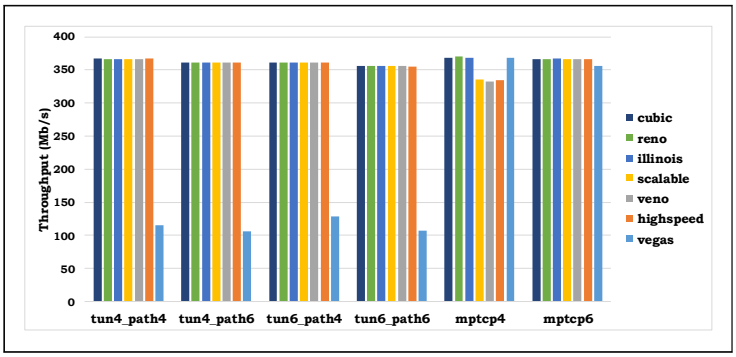

Figure 2: Testing the *ipferf3* and FTP throughput performance of the MPT-GRE and MPTCP in quad-path IPv4/IPv6 Fast Ethernet environments

times. The results showed a minimal, one percent deviation in each case. We carried out the measurements in both IPv4 and IPv6 environments, similar to earlier experiments (see e.g. [15, 16, 17]).

3. Measurement results

We began with the *iperf3*-based scenarios. The computer seen on the left side of Figure 1 was used as the *iperf3* server, while the computer on the right was used to run the Python scripts for the measurements. As the next figure shows (Figure 2), both the MPT-GRE and the MPTCP were effective in aggregating the capacity of the four independent paths. With the exception of TCP Vegas, each case resulted in a throughput measurement of about 360 Mbps. The MPT-GRE surprisingly under-performed when used in conjunction with TCP Vegas: the throughput fluctuated around 100 Mbps. On the other hand, the MPTCP proved effective when paired with either of the seven TCP versions, both in IPv4 and IPv6 quad-path environments.

The FTP-based performance analysis brought nearly identical results to those of the *iperf3* measurements (see Figure 2). In this scenario, a 1 GB file was downloaded into the memory of the client computer using the *wget* command. The TCP Vegas environment proved to be the most detrimental to the MPT-GRE results in these runs as well. The MPTCP also performed well. Interestingly, its operation over IPv6 seemed slightly more effective and stable, than over IPv4. It managed to achieve throughputs over 360 Mbps with each of TCP versions.

The following two figures (Figure 3 and Figure 4) illustrate the results of the most effective and the least effective FTP-based measurements.

The most effective path aggregation and thus the highest throughput could be observed when utilizing the TCP Cubic algorithm. In this case, compared to the single-interface baseline measurement, the download time dropped to a quarter

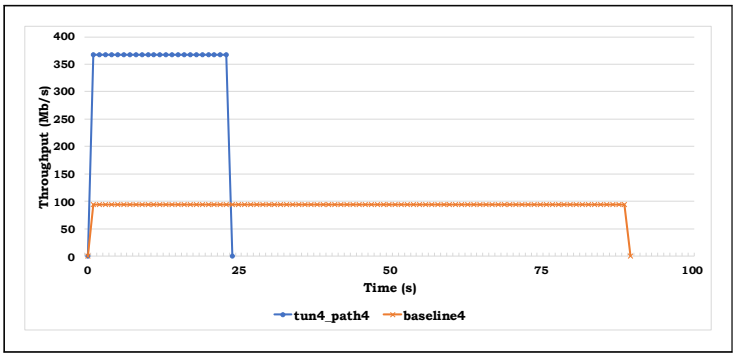

Figure 3: MPT-GRE IPv4-IPv4 *wget* throughput performance using 4 interfaces and the Cubic TCP congestion control algorithm

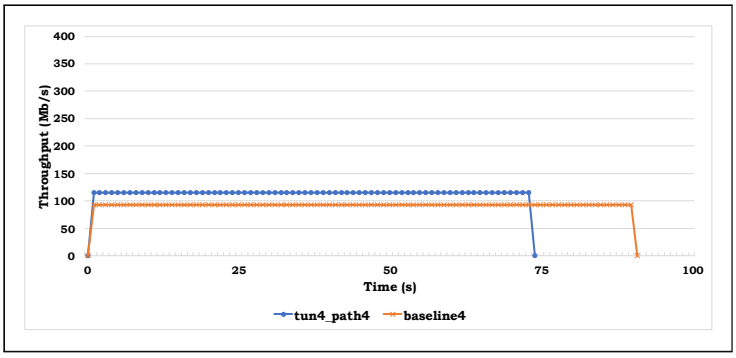

Figure 4: MPT-GRE IPv4-IPv4 *wget* throughput performance using 4 interfaces and the Vegas TCP congestion control algorithm

(3.1), while the throughput quadrupled (3.2). The worst results were acquired using the TCP Vegas algorithm, where the MPT-GRE performed only about ten percent above its single-path baseline. This meant its file download time was almost identical to the one measured in the single-path environment.

$$t(n) = \frac{t(0)}{n}[s], \text{ where } n = 4, \tag{3.1}$$

$$T(n) = n \cdot T(1)[Mb/s], \text{ where } n = 4. \tag{3.2}$$

The next two figures (Figure 5 and Figure 6) also show two measurement extremes, this time with regards to CPU utilization.

The same trend can be observed as before: the TCP Cubic algorithm proved to be the most suitable for both multipath environments. We can see the almost linear increase of CPU utilization with the number of interfaces and paths used. The MPTCP came out on top in this round as well since it exhibited lower resource

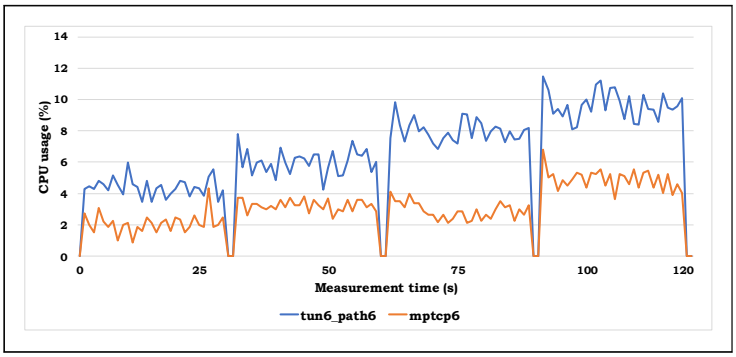

Figure 5: MPT-MPTCP CPU utilization comparison in a quad-path IPv6 Fast Ethernet multipath network environment in case of TCP Cubic and *iperf3*

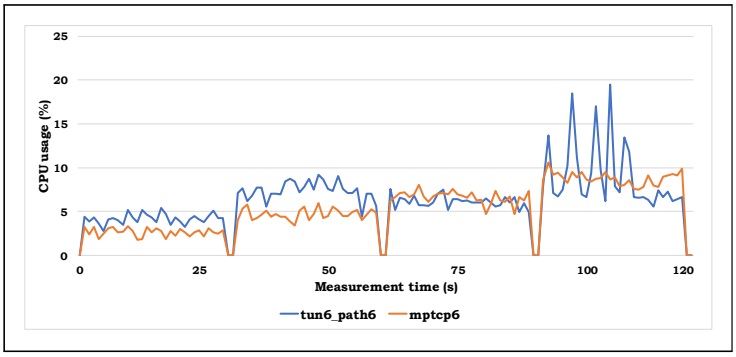

Figure 6: MPT-MPTCP CPU utilization comparison in a quad-path IPv6 Fast Ethernet multipath network environment in case of TCP Vegas and *iperf3*

usage compared to that of the MPT-GRE software. The TCP Vegas algorithm, this time, interestingly, closed the gap on the resource utilization of the two multipath technologies. However, quite a significant oscillation can be observed with regards to the MPT-GRE values in the quad-path IPv6 environment.

The last figure (see Figure 7) shows a summary of the CPU utilization of the MPT-GRE and the MPTCP multipath configurations when running *iperf3* in quad-path Fast Ethernet IPv6 environments. Looking at the averaged percentage values, we can see again that the resource demand of the MPTCP is somewhat lower. In our view, this can be attributed to the fact that the MPTCP is a kernel-level multipath solution, while the MPT-GRE is a user-space implementation. Despite this, both solutions can be considered effective.

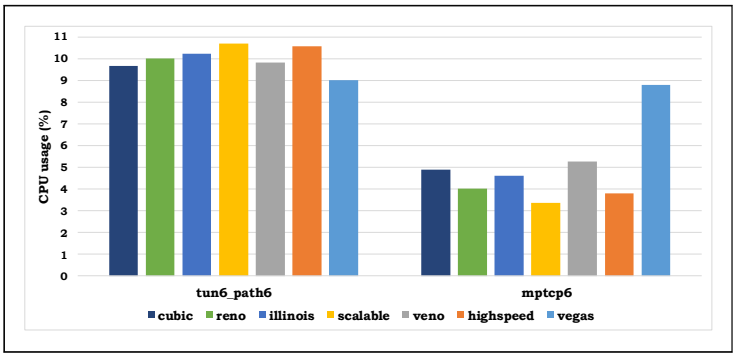

Figure 7: MPT-MPTCP CPU utilization comparison in a quad-path IPv6 Fast Ethernet multipath network environment in case of TCP Cubic, Reno, Illinois, Scalable, Veno, HighSpeed and Vegas using *iperf3*

4. Conclusions

In this paper we examined the effects of seven different TCP congestion control algorithms (TCP Cubic, TCP Reno, TCP-Illinois, Scalable TCP, TCP Veno, High-Speed TCP and TCP Vegas) on the MPT-GRE and the MPTCP multipath solutions in quad-path IPv4/IPv6 Fast Ethernet environments. We can conclude that the lowest system performance was achieved when utilizing the TCP Vegas congestion control algorithm, while the most optimal back-end configuration for the multipath solutions proved to be the TCP Cubic algorithm, which at the time of publication is the default congestion control algorithm used in both the latest Linux and Windows operating systems[2]. Our further objectives include extending our measurements to multipath Gigabit Ethernet IPv4/IPv6 environments as well as to include more recently released TCP congestion control algorithms (e.g. the Google-developed TCP BBR[3] from 2016, or the Elastic-TCP proposed in 2019 [18]).

Acknowledgements. This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

References

[1] Al-Bahadili, H., Simulation in computer network design and modeling: Use and analysis, *Hershey, PA: IGI Global* (2012), 282.

[2] `https://en.wikipedia.org/wiki/CUBIC_TCP`
[3] `https://github.com/google/bbr`

[2] Akiene, P. T., Kabari L. G. Simulation of an Optimized Data Packet Transmission in a Congested Network, *Network and Complex Systems*, Vol. 5 (2015), 29–37.

[3] Jacobson, V., Congestion Avoidance and Control, *Proceedings of the 1988 ACM SIGCOMM Symposium* (1988), 314–329.

[4] Móczár, Z., Molnár, S., Towards the Transport Protocols of Future Internet, *Infocommunications Journal*, Vol. 6 (2014), 3–9.

[5] Yang, P., et. al., TCP Congestion Avoidance Algorithm Identification, *IEEE/ACM Transactions on Networking*, Vol. 22 (2014), 1311–1324.

[6] Pande, A. P., Devane, S. R.., Study and Analysis of Different TCP Variants, *IEEE: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (2018), 1–8.

[7] Lin, J., et. al., Extensive evaluation on the performance and behaviour of TCP congestion control protocols under varied network scenarios, *Elsevier: Computer Networks*, Vol. 163 (2019), 1–21.

[8] Mateo, P. J., et. al., Analysis of TCP Performance in 5G mm-Wave Mobile Networks, *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)* (2019), 1–7.

[9] Mishra, A., et. al., The Great Internet TCP Congestion Control Census, *Proceedings of the ACM on Measurement and Analysis of Computing Systems* (2019), 1–24.

[10] Park, J., Kim, H., Choi, J., Improving TCP Performance in Vehicle-To-Grid (V2G) Communication, *Electronics*, Vol. 8 (2019), 1206–1223.

[11] Turkovic, B., Kuipers, F. A., Uhlig, S., Interactions between Congestion Control Algorithms, *IEEE: 2019 Network Traffic Measurement and Analysis Conference (TMA)* (2019), 161–168.

[12] Polese, M., et. al., A Survey on Recent Advances in Transport Layer Protocols, *IEEE Communications Surveys and Tutorials*, Vol. 21 (2019), 3584–3608.

[13] `http://multipath-tcp.org/`, *The MPTCP Project official website* (2020).

[14] https://irh.inf.unideb.hu/~szilagyi/index.php/en/mpt/, *The MPT-GRE Project official website* (2020).

[15] Almási, B., Szilágyi, S., Throughput Performance Analysis of the Multipath Communication Library MPT, *TSP 2013 – The 36th International Conference on Telecommunications and Signal Processing*, (2013), 86–90.

[16] Almási, B., Lencse, G., Szilágyi, S., Investigating the Multipath Extension of the GRE in UDP Technology, *Computer Communications*, Vol. 103 (2017), 29–38.

[17] Szilágyi, S., Bordán, I., Harangi, L., Kiss, B., Throughput Performance Comparison of MPT-GRE and MPTCP in the Gigabit Ethernet IPv4/IPv6 Environment, *Journal of Electrical and Electronics Engineering*, Vol. 12 (2019), 57–60.

[18] Alrshah, M., A., Al-Maqri, M., A., Othman, M., Elastic-TCP: Flexible Congestion Control Algorithm to Adapt for High-BDP Networks, *IEEE Systems Journal*, Vol. 13 (2019), 1336–1346.