Evaluation of the Aggregation Capability of the MPT Network Layer Multipath Communication Library and Multipath TCP

Ákos Kovács

Department of Telecommunications
Széchenyi István University
Egyetem tér 1, H-9026 Győr, Hungary
kovacs.akos@sze.hu

Abstract: Multipath communication techniques can bring in a new era for Cognitive Info-communication, due to ensuring resilient and high-speed data transfer. In this paper, we evaluated the MPT network layer multipath communication library, which creates an UDP tunnel, based on the GRE in UDP tunnel protocol. We compared the aggregation capability of MPT to that of MPTCP, which stands for Multi-Path TCP and based on TCP sub-flows to aggregate the transmission capacities of different physical interfaces and their potentially disjoint paths to ensure high network throughput. In this article, we used 100 Mbps and 1 Gbps speed channels to compare the aggregation capabilities of these two different multipath communication solutions. We used several scenarios for the evaluation. We tested both IPv4 and IPv6 both as underlying and as encapsulation protocols. We used several channels up to 12 to evaluate the aggregation capabilities with the industry standard iperf tool, even with different numbers of iperf threads. Meanwhile we measured the CPU usage of the two examined multipath technologies to get further insight into their operation. On the basis of our measurement results, we also set up a mathematical model of their channel aggregation capabilities.

Keywords: MPT; MPTCP; channel aggregation; multipath communication; performance analysis

1 Introduction

In everyday usage, most of the modern ICT devices have at least two or more communication interfaces like LTE modems, Wi-Fi or Ethernet cards, but we cannot utilize more than one of them for a single communication session due to technical limits: one particular TCP connection can be identified by four numbers: the source and destination IP addresses and the source and destination port numbers [1], [2]. To address this problem, the MPT network layer multipath communication library [3] was developed by the Faculty of Informatics, University of Debrecen, Debrecen, Hungary. Multipath communication like MPT was proposed as a possible new basis for future cognitive info-communication [4]. Cognitive info-communication is meant to draw up the possible communication between human and next-generation ICT systems [5]. It can be used to add more stability and available bandwidth to systems like Virtual Laboratories [6] and cloud applications [7]. Besides MPT, MPTCP (Multipath TCP) [8] is the other well-known multipath communication technology, which can be another good candidate for this purpose. Both technologies are able to utilize several Ethernet, WiFi or other types of channels to be used as a single communication channel. In this paper, we would like to measure how linearly the throughput increases after adding multiple NICs to the communication, we call this *aggregation*. In this paper, we compare the aggregation capability of these two solutions in various measurement scenarios.

The remainder of is paper is organized as follows: First, we give a short summary of the MPT related research results, and then introduce both MPT and MPTP. Second, we perform some experiments both at 100 Mbit/s and 1 Gbit/s meanwhile we measure the CPU utilization as well. After that, we present our model of the transmission rate. Next, we disclose our plans for future work, and conclude our paper.

2 Related Work

Several research papers were published on the analysis of MPT. FTP and stream transmission experiments were performed using Cisco routers and relatively low transmission rate links in [9]. MPT is also mentioned as an IPv4 or IPv6 integration technology for IoT (Internet of Things) [10]. MPT was successfully used as a solution for eliminating the effect of network breakdowns in case of HD video stream transmission [11]. The throughput aggregation capability of MTP was tested up to four paths in [12].

We have also evaluated the aggregation capabilities of MPT [13], but since then the developers made some important changes using GRE tunnelling protocol [14] in the communication architecture [15].

To evaluate the channel aggregation capability of the MPT communication library, we used MPTCP (Multipath TCP) [8] as a basis for comparison. MPTCP also uses multiple interfaces for communication, but it uses several TCP subflows to utilize the accessible Ethernet devices. The software library and documentation is available from [16].

3 Introduction to MPT and MPTCP

3.1 The MPT Communication Library

The MPT library enables multipath communication in network layer. The latest version of MPT is based on the IETF RFC 8086 ("GRE in UDP") [14] specification and the MPT communication library grants us to use multiple different paths [15]. The architecture of MPT is shown in Figure 1.

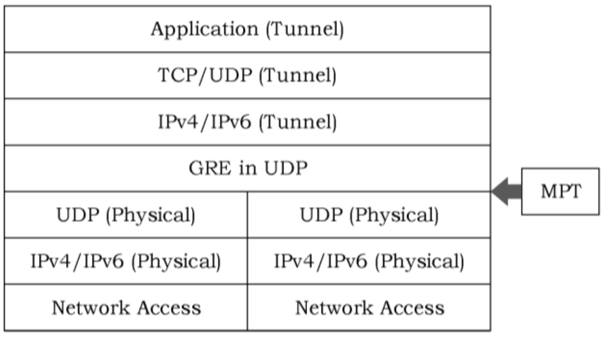

Figure 1
The architecture of MPT [15]

The IP packets are forwarded to the tunnel interface by the MPT communication library and encapsulated into a “GRE in UDP” segment, which will be transferred; see the frame structure in Figure 2. The applications and services can use IP without any modification because the MPT abstraction layer is invisible in the upper OSI layers like the application layer. The most important advantage of the MPT communication library is that it can use both TCP and UDP, contrasted to MPTCP, which is a TCP-only multipath communication solution.

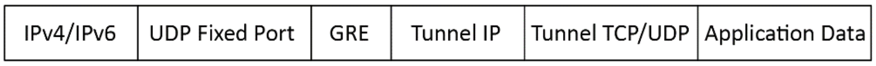

Figure 2
The PDU structure MPT "GRE in UDP" [3]

3.2 TheMultiPath TCP

The MPTCP Linux implementation is available from the early 2013 and it was developed at Université Catholique de Louvain, Belgium. Its main goal is to improve the more than 30 years old TCP protocol. It requires some modification in the kernel to utilize multiple available network interfaces for a single TCP communication session.

As an important difference from MPT, MPTCP does not use any tunnelling techniques. It uses all available network interfaces using special TCP sub-flows for a single MPTCP session. Besides, it requires some configuration [16].

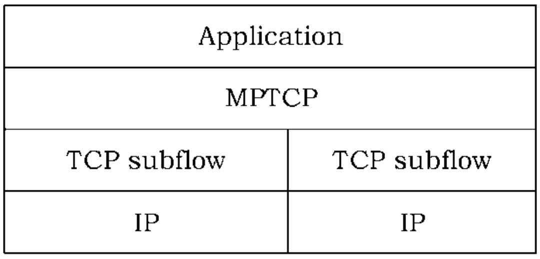

Figure 3
The architecture of MPTCP communication stack [15]

When using MPTCP, the protocol stack negotiates using the normal three-way-handshake if the server is capable of using multiple connections for increasing throughput. After that, additional TCP sub-flows can be used for data transfer. If multipath negotiation is succeeded, the MPTCP protocol stripes the data between the subflows. The MPTCP at the receiving side is capable of reconstructing the original order from the received packets. Each TCP sub-flow works as a normal TCP session with its respective congestion control and sequence numbering [18].

The main differences between these two technologies are that MPT must use the same number of physical interfaces both at the sending side and at the receiving side. MPT creates a GRE tunnel above them. Unlike MPT, the Multipath TCP uses several network interfaces on the client side each with its own default gateway settings. With this difference, MPTCP can use up to seven 100 Mbps links on the sending side and one 1 Gbit/s link on the server side. The only requirement is that both the server and the client must use MPTCP enabled kernel.

4 The Experiments

4.1 MPT Communications library Measurements at 100Mbit/s

In our measurements, we used the version GRE-2015-10-23-64bit and the topology of the test network was the same as we used in [13]. For the MPT communication library, we have to install the software both on the client and server sides. We have to delegate different network segment settings to all network interfaces. After that, these settings must be made in the MPT configuration files as well. We are able to use different IP protocol version for the tunnel interface at the same time, because we can add explicitly which version of IP protocol we want to use for our tests.

We used the industry standard iperf benchmark tool to evaluate the throughput of twelve 100 Mbps links. We wrote a script to add another available NIC to the communication after each successful measurement. We used both IP version as tunnel and as the underlying protocol as well. Because of the length of the script we do not include it here, but the key command for measuring with iperf was:

```
iperf -c 10.0.0.1 -t 100 -f M
```

This performed a 100 seconds long data transfer and the results were given in Mbytes/s. For a successful test, we have to use the "iperf –s" command at the server side to accept the iperf client connections. For reference, we used HTTP download test with the wget Linux tool.

```
wget -O /dev/null http://10.0.0.1/8GiB
```

With this command, we downloaded an 8 GB large file to the client, but to ensure that the disk writing speed does not limit our measurement results, we disposed it in /dev/null. On the server side we used RAMDISK to serve this download to avoid the disk reading speed limit.

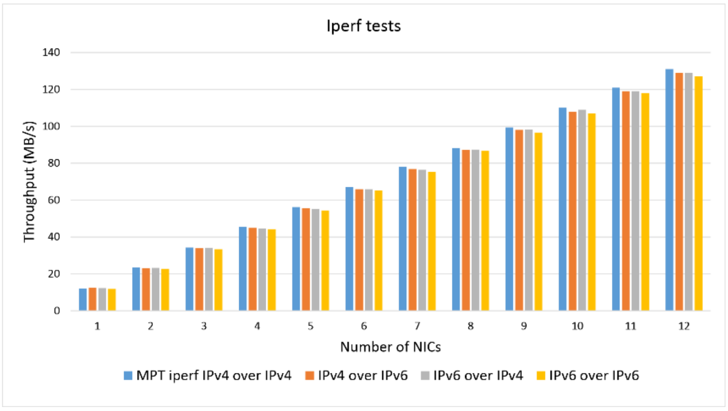

Figure 4
The MPT iperf tests

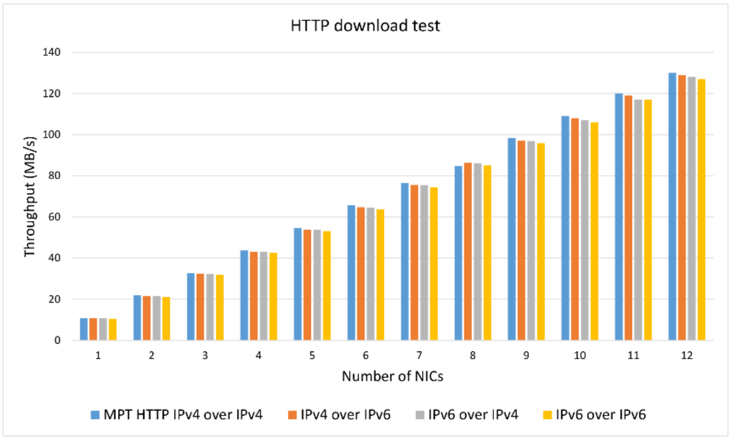

Figure 5
The MPT HTTP tests

The measurement results of the MPT communications library are shown in Figure 4. The throughput aggregation capability of the MPT library has been proven to be nearly ideal, the performance scaled up nearly linearly up to 12 NICs. We used different IP protocol versions for the tunnel or for the underlying interfaces, but we can see only a minor difference in the throughput. Compared to our measurements results before [13], this result shows a significant improvement of the MPT library.

The HTTP download tests using wget show very similar results in Figure 5.

4.2 MultiPath TCP Measurements at 100 Mbit/s

For the MPTCP test, we used the same infrastructure, but some configuration settings have been changed. First, we had to download and compile a different Linux kernel that supports Multipath TCP connection. After restarting the computers, the Multipath TCP enabled kernel automatically utilized the available network interfaces because of the new three-way-handshake algorithm. For that we can use a simple if-up.d script provided by the developers to avoid any handmade changes in network configuration.

This script generates different network configurations to all available network interfaces. It adds different gateway parameters to all network interfaces, where a TCP sub-flow can be made. If several interfaces can connect to an MPTCP enabled host it can be used for multipath communication. In our example we show the routing table of the eth1 network interface:

```
ip rule add from 10.1.1.2 table 1
ip route add 10.1.1.0/24 dev eth1 link table 1
ip route add default via 10.1.1.1 dev eth1 table 1
```

To ensure that our control interface (eth0 on both sides) is not involved in the communication we had to deny any communication expect SSH connection on each interface. This can be done simply with iptables rules.

The measurement script was almost the same expect we could not use the IP address added by the MPT logical interface, so we used one of the physical network interface parameters.

We can see from the results that after using 8 interfaces for communication, there was no further increase in throughput during the tests, see Figure 6 and 7. We contacted the developers to ensure no configuration mistake was made. They replied that the MTPCP (version 0.9) could utilize only up to 8 network interfaces for multipath communication [19].

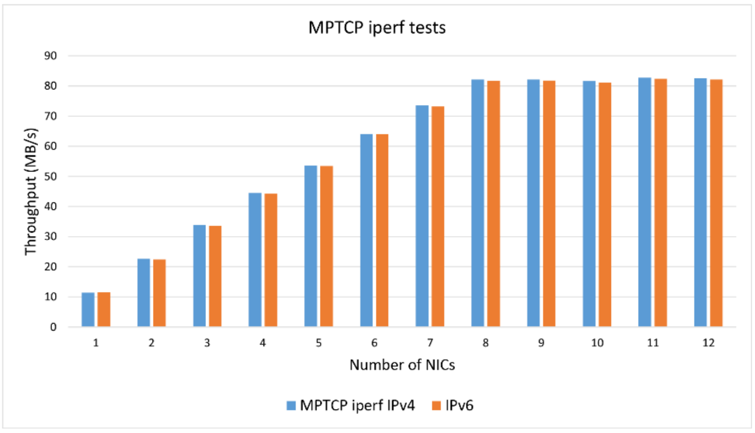

Figure 6
The MPTCP iperf tests

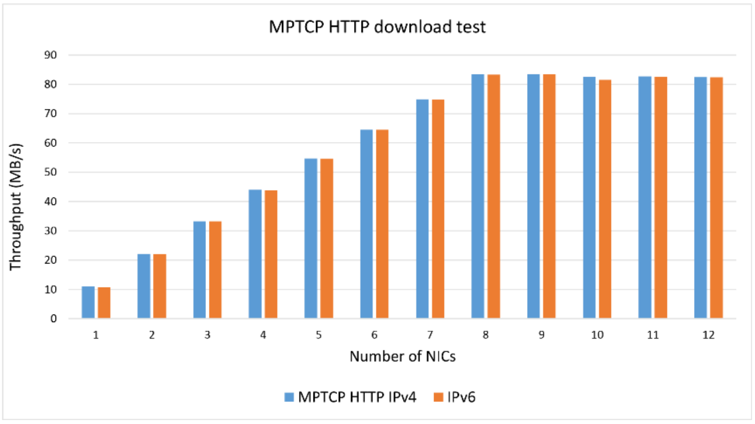

Figure 7
The MPTCP HTTP tests

We found a major difference between the aggregation capability of MPT and MPTCP: the Multipath TCP has a limitation to eight network interfaces. The aggregation capabilities of the MPT communication library and the Multipath TCP are compared in Figure 8.

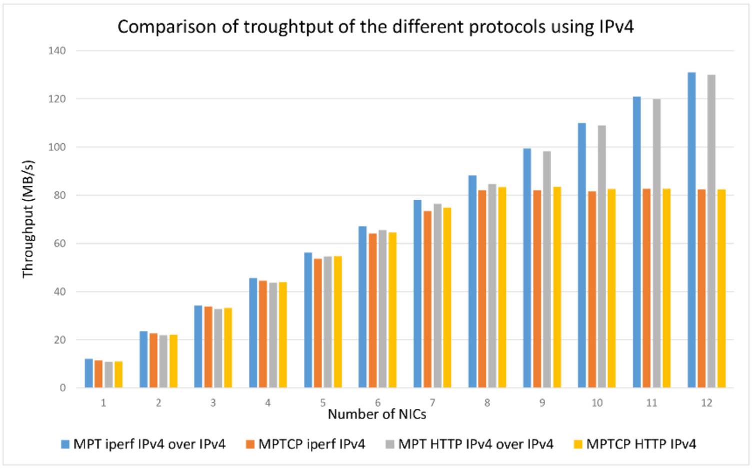

Figure 8

The MPTCP HTTP tests

To examine the CPU usage of both MPT and MPTCP, we performed further measurements. For that, we used a simple Bash shell script to measure the CPU usage on both sides (client and server). The results are shown in Figure 9 and 10.

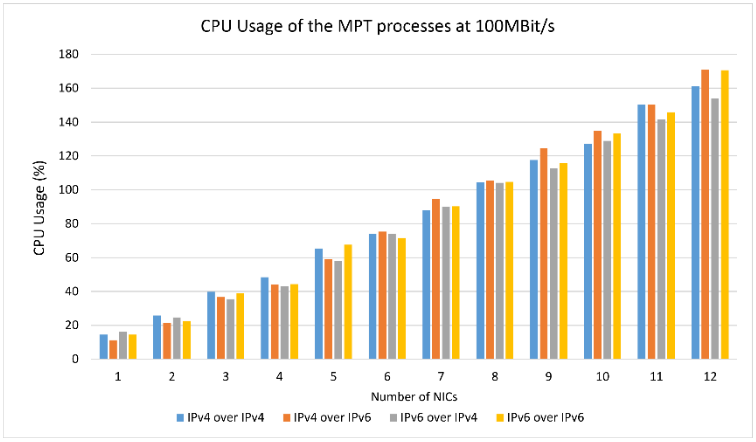

Figure 9

CPU Usage of MPT processes at 100 Mbit/s

As we can see the results, almost 2 cores of the available CPUs were used to ensure data transfer. The CPU usage of the MPT client and of the MPT server was almost the same. (The Linux system shows the CPU resource of a core as 100%. As our computers contain 4 cores, their maximum CPU capacity was shown as 400%). Although, the two computers we used were rather old, there were no problems using MPT at 100 Mbit/s speed.

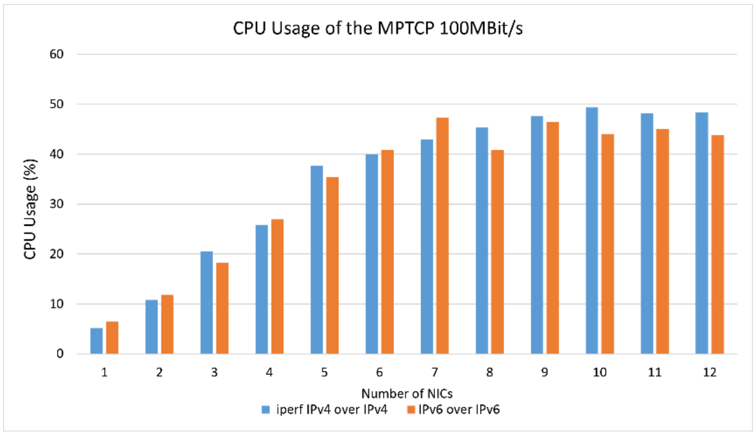

Figure 10
CPU Usage of MPTCP at 100 Mbit/s

As we can see MPTCP is much less CPU intensive as MPT. Because MPTCP is running in kernel space it consumes only the quarter as much as MPT. It changes the default Linux TCP stack so it does not have to create a GRE Tunnel for transferring data. However, it cannot utilize more than 8 NICs.

4.3 Experiments at 1000 Mbit/s

After successful measurements at 100 Mbit/s, we used the same configuration at 1000 Mbit/s. As the results of the iperf measurements and the results of the HTTP download tests were almost the same, we used only iperf measurements at 1000 Mbit/s. We only changed the topology of the test system by removing the Cisco switch from the original measurement setup, which was used to limit the communication speed to 100 Mbit/s, and connected the network interfaces of the two computers with patch cables directly. We tested each connection separately to ensure all the connection is at full speed. Then, we used the same scripts which we used at 100 Mbit/s. The results are shown in Figure 11 and 12.

As we can see in the results, MPT cannot fully utilize the capacity of the 1000 Mbit/s connections. Even if we use 2 NICs with gigabit speed the throughput is far from the 2 Gbit/s. It even does not reach the 1 Gbit/s. After we added the 9th NIC, MPT stopped transferring data. The iperf reported about 0.06 Mbit/s speed, which is negligible. There were also plenty of error messages indicating that many packets in the GRE tunnel were dropped because they did not arrive in the proper order or there were many duplicates. Sometimes one of the physical channels was also dropped for a second. Every time when that happened, MPT dropped different physical channels so we can rule out hardware failures or hardware incompatibilities.

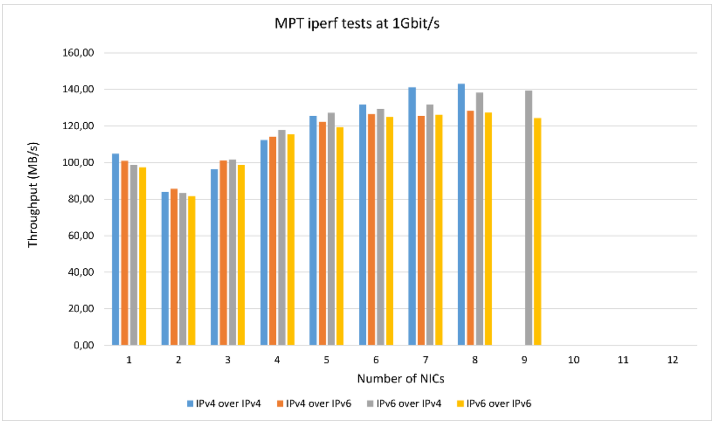

Figure 11
MPT iperf tests at 1 Gbit/s

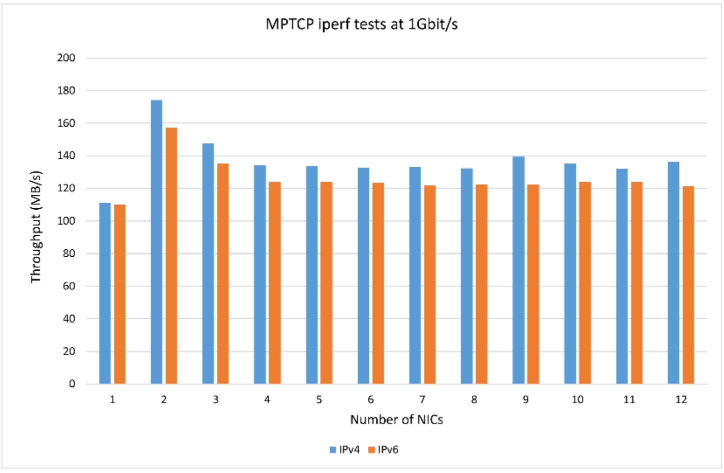

Figure 12
MPTCP iperf tests at 1 Gbit/s

If MPT cannot utilize the 9th NIC then, it's most significant advantage over MPTCP is lost. MPTCP also showed that there was some bottleneck in the system, so we measured it again with monitoring the CPU usage as well. The results are shown in Figure 13 and 14. We can see that MPT is utilizing almost 2 cores fully, but not using the remaining two cores. This performance requirement is almost the same as at 100 Mbits/s with 12 NICs. That implies some implementation issues. If MPT cannot utilize more than two CPU cores then it will be a major issue in high-speed systems. However, a question now arises: why could not MPT increase its CPU utilization above 180% - 200% while there was still free CPU capacity? The answer is that MPT was written as a serial program and because of that it is not able to fully utilize the available processing power of the multiple CPU cores.

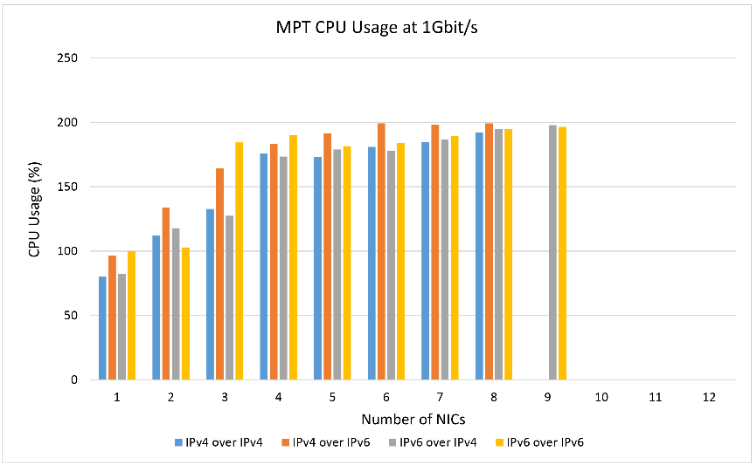

Figure 13
CPU Usage of MPT processes at 1 Gbit/s

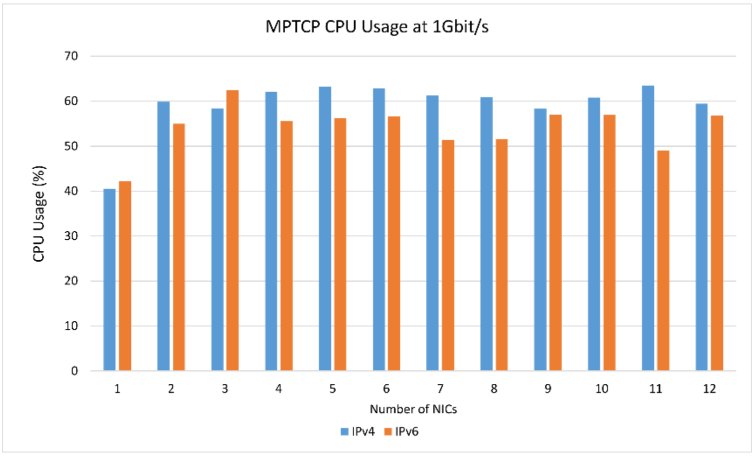

Figure 14
CPU Usage of MPTCP at 1 Gbit/s

We believe that MPT must be improved in this field; because the current trend of the CPU evolution is that the number of CPU cores is increasing rapidly instead of the clock speed [20]. MPTCP also showed some bottleneck at about 60 percent. Because MPTCP did not utilize even 1 core, we also measured the 1 Gbit/s tests with multi-thread iperf. First, we measured MPT again, but now, we used 4 threads for iperf server and client as well. We have done this with a simple modification in the command.

```
iperf -c 10.0.0.1 -t 100 -f M -P 4
```

The `-P 4` switch allows us to run iperf at 4 threads. When we used iperf with more than one thread it continuously exited after one measurement was done.

We had to write a simple script that monitors the iperf processes, and when it quits then restarts it with the proper parameters so that each measurement could be done automatically. The results are shown in the Figure 15 and 16.

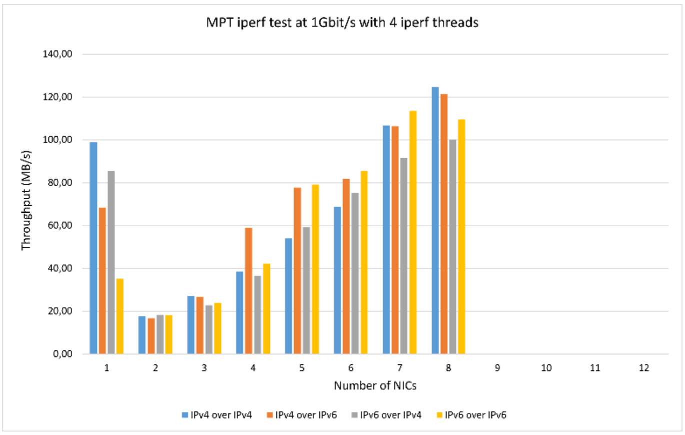

Figure 15
MPT iperf tests at 1 Gbit/s with 4 iperf threads

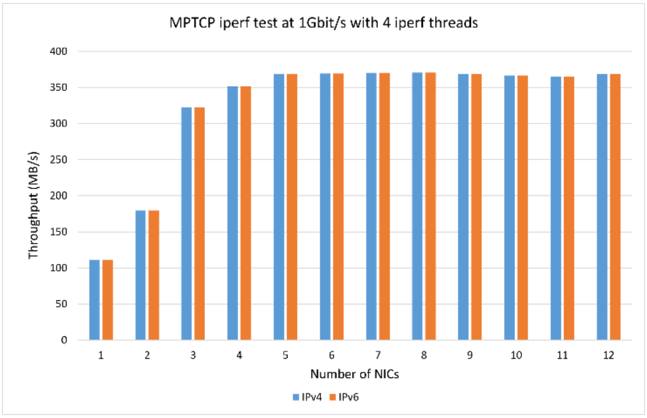

Figure 16
MPTCP iperf tests at 1 Gbit/s with 4 iperf threads

As we can see in Figure 15, MPT performance was even worse. If we use iperf with 4 threads MPT and iperf rival for the available computing power. It confirms that MPT has some issues when we use it in a relatively high-speed environment. On the other hand, MPTCP can exploit the 4 iperf threads. It utilizes all the available computing power and the data transfer rates were up to 370 MB/s, which is a very impressive outcome. Even when we use only 4 NICs of the twelve it can reach 350 MB/s, which is nearly linear speed boost. The CPU utilization in Fig. 18 shows us that the bottleneck was the CPU, MPTCP could utilize all the available core along with iperf so if we had been able to add more computing power, we could have reached even higher transfer rates.

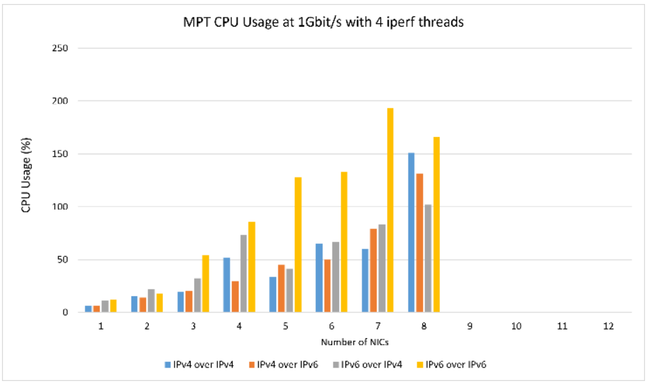

Figure 17
CPU Usage of MPT processes at 1 Gbit/s with 4 iperf threads

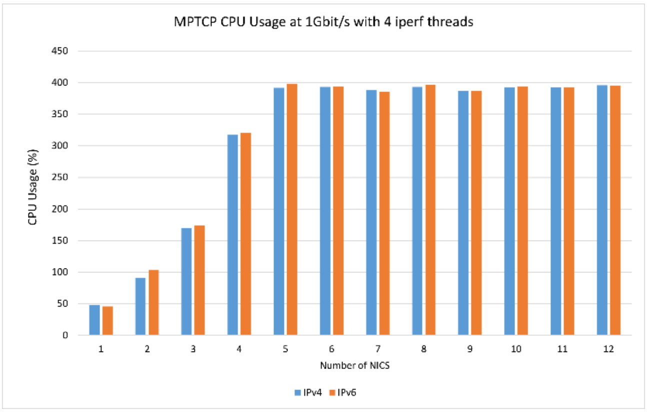

Figure 18
CPU Usage of MPTCP processes at 1 Gbit/s with 4 iperf threads

5 Modelling of the Throughput

5.1 Models for 100 Mbit/s Performance

The performance of MPT was linear in the whole range whereas the performance of MPTCP was linear only up to 8 NICs. The throughput of MPT as a function of the number of NICs and the throughput of MPTCP as a function of the number of the NICs can be simply modelled by equation (1) and equation (2), respectively.

$$T(\mathrm{n}) = \mathrm{n} \cdot T(1) \tag{1}$$

$$T(n) = \begin{cases} n \cdot T(1), if\ n \le 8 \\ 8 \cdot T(1), if\ n > 8 \end{cases} \quad (2)$$

In these cases the throughput was not limited by the performance of the CPU because the transmission speeds were low enough.

5.2 Models for 1000 Mbit/s Performance

The throughput of MPT was measured by using iperf with 1 or 4 threads and the results can be seen in Figure 12 and Figure 16. It can be seen in both figures that the performance drops using 2 NICs but then it grows nearly linearly in the function of the number of NICs until MPT collapses due to the lack of CPU performance at about 9 NICs. The MPT performance with 4 iperf threads is drastically decreased, but we can see some similarity to Figure 12. When we use 1 NIC, MPT can almost utilize that, but after we added the second NIC to the communication, there is a major relapse. After we add more NICs, the transfer rate starts to increase nearly linear. We can see from the Fig. 16 that the increasing is nearly linear but with bigger dispersion, which is caused by the 4 iperf threads. The MPT and the iperf processes rival for the available CPU resources. We used linear regression for model creation.

The throughput of MPT as a function of the number of NICs can be modelled by equation (3).

$$T(n) = \begin{cases} T(1), if\ N = 1 \\ \alpha \cdot \mathrm{T}(1) + n \cdot \beta \cdot \mathrm{T}(1), if\ 2 \le n < 8 \\ 0, if\ n \ge 9 \end{cases} \quad (3)$$

The throughput of MPTCP at 1000 Mbit/s was different. First, we can see a minor performance improvement because MPTCP can utilize more than one NICs at 1000 Mbit/s, but after that we can see an exponential decay to 4 NICs and after that it becomes constant.

When we use MPTCP with 4 iperf threads, we can achieve the best performance for now. If we add up to 4 NICs to the communication it is nearly linear, but after that it comes to saturation due to the exhaustion of all the available resources thus the transmission rate is about up to 400 MB/s with 4 or more NICs. We can see in Fig. 18 that all the CPU cores are running at maximum utilization. The throughput of MPTCP with 4 iperf threads as a function of the number of NICs can be approximated by equation (4).

$$T(n) = \begin{cases} n \cdot T(1), if\ n \le 4 \\ 4 \cdot T(1), if\ n > 4 \end{cases} \quad (4)$$

6 Plans for Future Research

As the most important advantage of MPT over MPTCP is that MPT uses UDP/IP and therefore, it is more suitable for use with real-time applications because of the elimination of TCP retransmissions, we also plan to test it with real-time applications.

Our plans for future research include the investigation how MPT can be used for the transmission of multicast traffic as well. The resilient nature of this multipath solution may efficiently complement the fault tolerance of the PIM-SM multicast routing protocol. We can rely on the methods and results of the following papers. A model for the fault tolerance of PIM-SM was published in [21]. The fault tolerance of PIM-SM was investigated under the XORP platform in [22]. An experimental analysis of the fault tolerance of PIM-SM under GNS3 was published in [23].

Another interesting area can be the testing of the aggregation of transmission capacities of the LAN, WiFi and LTE (as USB device) interfaces of different mobile devices or using them as alternative paths to provide resilient communication as studied and demonstrated in [24]. As our results show that MPT requires relatively high CPU capacity, the device selection for our future tests will benefit from the benchmarking methods for single board computers published in [25]. Also the results of the CPU performance measurements, where the computing performances of ten different single board computers were compared [26], will be of a great help for us.

As MPT can carry any version IP packets over any version IP networks, it can be used as a tunnelling solution to support IPv6 transition. Both IPv4 over IPv6 and IPv6 over IPv4 may be beneficial for the users depending on the given application scenario. When MPT is used as an IPv6 transition technology, its performance is to be measured according to RFC 8219 [27], which defines two types of setups for the benchmarking measurements of tunnels. The dual DUT setup measures the performance of an established tunnel and commercially available RFC 2544 [28] compliant testers can be used. The single DUT setup measures the individual performances of the endpoints, and it requires a special tester, which is not yet available. (We have found publications about RFC 8219 compliant testers for benchmarking DNS64 servers [29] or NAT64 gateways [30], but not for tunnel endpoints.) Thus, currently we can test the performance of MPT only according to the dual DUT setup. However, the application of standard RFC 2544 compliant testers will make the results of our planned benchmarking measurements comparable with the results of other RFC 2544 compliant benchmarking measurements.

Conclusion

We have evaluated the data transfer aggregation capability of the MPT network layer multipath library and of the MPTCP Linux implementation with several network interfaces up to twelve, both with 100 Mbps and 1 Gbps network speed as well as using any possible combinations of the different IP versions. The aggregation capability of the MPT communication library proved to be very good, when 100 Mit/s speed links were used. The throughput scaled up nearly linearly up to 12 network interfaces in all possible combinations of IPv4 and IPv6. It exceeded 120 Mbytes/s. The results of the HTTP tests were almost the same. So, we can say that MPT can utilize up to twelve interfaces at 100 Mbps on a slightly old computer. The results of the 1000 Mbps tests are different. As for implementation issues, MPT cannot utilize even 2 NICs at 1000 Mbps, and after we added more NICs to the communication, the results were even worse and above 8 NICs, MPT stopped working.

Multipath TCP has significant advantages, we do not have to use special software, and the configuration is made automatically. This can save time for network administrators. However, it can utilize only 8 network interfaces for multipath communication. We think it is not a significant disadvantage, because in an everyday use environment it is not common to use more than eight network interfaces for a single TCP session.

Because MPTCP is a kernel space implementation of the multipath communication it consumes less resources than MPT. Regarding this, it is able to utilize up to 4 NICs at 1000 Mbps by scaling up nearly linearly. We measured 350 MB/s transfer rate using 4 iperf threads and 4 NICs.

Because of the experienced high CPU consumption, these technologies are only recommended for those nodes that are responsible only for networking. We can use them as a high capacity gateway for cognitive applications.

Acknowledgement

This article is an extended version of paper [1].

References

[1] Á. Kovács, Comparing the aggregation capability of the MPT communications library and multipath TCP. 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom) 2016, Pages: 157-162, DOI: 10.1109/CogInfoCom.2016.7804542

[2] B. Almási A. Harman, An overview of the multipath communication technologies. In Proc. Advances in Wireless Sensor Networks 2013, Debrecen University Press, Debrecen, Hungary, Pages: 7-11

[3] B. Almási, MPT Multipath Communication Library. available: http://irh.inf.unideb.hu/user/szilagyi/mpt/

[4] B. Almási. Multipath communication: a new basis for the future Internet cognitive infocommunication. In Proc. 4th IEEE International Conference on Cognitive Infocommunications 2013, Budapest, Hungary, Pages: 201-204

[5] P. Baranyi, A. Csapo, G. Sallai Cognitive Infocommunications (CogInfoCom), Springer International, 2015

[6] T. Budai, M. Kuczmann Towards a modern, integrated virtual laboratory system, Acta Politechnica Hungarica. Vol. 15, No. 3, 2018, Pages: 191-204

[7] Sz. Szilágyi, I. Bordán, L. Harangi, B. Kiss. MPT-GRE A novel multipath communication technology for the cloud.In Proc. 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2018), August 22-24, 2018, Budapest, Hungary

[8] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, TCP extensions for multipath operation with multiple addresses, IETF RFC,6824, 2013 [Online] Available: https://tools.ietf.org/html/rfc6824

[9] B. Almási, Sz. Szilágyi, Multipath ftp and stream transmission analysis using the MPT software environment. International Journal of Advanced Research in Computer and Communication Engineering. Vol. 2, No. 11, 2013, Pages: 4267-4272

[10] Z. Gal, B. Almási, T. Daboczi, R. Vida, S. Oniga, S. Baran, and I. Farkas, Internet of things: application areas and research results of the FIRST project. Infocommunications Journal, Vol. 6, No. 3, 2014, Pages 37-44

[11] B. Almási, M. Kosa, F. Fejes, R. Katona, L. Pusok, MPT: A solution for eliminating the effect of network breakdowns in case of HD video stream transmission. In Proc. 6th IEEE International Conference on Cognitive Infocommunications. Győr, Hungary, 2015, Pages: 121-126

[12] B. Almási, Sz. Szilágyi, Investigating the throughput performance of the MPT multipath communication library in IPv4 and IPv6. International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems Vol. 5, No. 1, 2016, Pages: 53-60

[13] G. Lencse, A. Kovács, Advanced measurements of the aggregation capability of the MPT multipath communication library. International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems 2015, Vol. 4, No. 2, Pages: 41-48, DOI: 10.11601/ijates.v4i2.112

[14] E. Crabbe, L. Yong, X. Xu, T. Herbert. GRE-in-UDP encapsulation. IETF RFC 8086 [Online] Available: https://tools.ietf.org/html/rfc8086

[15] B. Almási, G. Lencse, Sz. Szilágyi, Investigating the multipath extension of the GRE in UDP technology. Computer Communications, Vol. 103, No. 1, 2017, Pages: 29-38, DOI: 10.1016/j.comcom.2017.02.002

[16] C. Paasch, S. Barre, et al., Multipath TCP in the Linux Kernel, available from http://www.multipath-tcp.org used version 0.91

[17] A. Ford, C. Raiciu, M. Handley, S. Barre, J. Iyengar, Architectural guidelines for multipath TCP development, IETF RFC 6182,2013 [Online] Available: https://tools.ietf.org/html/rfc6182

[18] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik and M. Handley, Improving datacenter performance and robustness with multipath TCP, 2011, Toronto, Canada

[19] MPTCP developers mailing list https://listes-2.sipr.ucl.ac.be/sympa/arc/mptcp-dev/2016-02/msg00018.html

[20] G. Lencse, I. Derka and L. Muka, Towards the efficient simulation of telecommunication systems in heterogeneous distributed execution environments.In Proc. 36th International Conference on Telecommunications and Signal Processing. Brno University of Technology, 2013, Pages: 314-310, DOI: 10.1109/TSP.2013.6613941

[21] G. Lencse and I. Derka, Towards the modelling of the fault tolerance mechanism of the PIM-SM multicast routing protocol in an IPTV environment. In Proc. European Simulation and Modelling Conference. Essen, Germany, 2012 Oct. 22-24, EUROSIS-ETI, Pages: 152-156

[22] G. Lencse and I. Derka, Investigation of the fault tolerance of the PIM-SM IP multicast routing protocol for IPTV purposes. Infocommunications Journal, Vol. 5, No. 1, 2013, Pages 21-28

[23] G. Lencse and I. Derka, Experimental analysis of the fault tolerance of the PIM-SM IP multicast routing protocol under GNS3. International Journal of Advanced Computer Science and Applications, Vol. 5, No. 5, 2014, Pages: 15-22, DOI: 10.14569/IJACSA.2014.050503

[24] F. Fejes, Multipath strategies and solutions in multihome mobile environments, In Proc. 7th IEEE Conf. on Cognitive Infocommunications. Wrocław, Poland, Oct. 16-18, Pages: 79-84, 2016, DOI: 10.1109/CogInfoCom.2016.7804529

[25] G. Lencse and S. Répás, Method for benchmarking single board computers for building a mini supercomputer for simulation of telecommunication systems. In Proc. 38th International Conference on Telecommunications and Signal Processing. Prague, Czech Republic, Brno University of Technology, 2015, Pages: 246-251, DOI: 10.1109/TSP.2015.7296261

[26] G. Lencse, S. Répás, Benchmarking further single board computers for building a mini supercomputer for simulation of telecommunication systems. International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, Vol. 5, No. 1, 2015, Pages: 29-36, DOI: 10.11601/ijates.v5i1.138

[27] M. Georgescu, L. Pislaru and G. Lencse, Benchmarking methodology for IPv6 transition technologies, IETF RFC 8219, Aug. 2017, DOI: 10.17487/RFC8219

[28] S. Bradner, J. McQuaid, Benchmarking methodology for network interconnect devices, IETF RFC 2544, Mar. 1999, DOI: 10.17487/RFC2544

[29] G. Lencse, D. Bakai, Design and implementation of a test program for benchmarking DNS64 servers, IEICE Transactions on Communications, Vol. E100-B, No. 6, Pages: 948-954, June 2017, DOI: 10.1587/transcom.2016EBN0007

[30] P. Bálint, Test software design and implementation for benchmarking of stateless IPv4/IPv6 translation implementations, in Proc. TSP 2017, Barcelona, Spain, July 5-7, 2017, Pages: 74-78