Advanced Measurements of the Aggregation Capability of the MPT Network Layer Multipath Communication Library

Gábor Lencse, Ákos Kovács

***Abstract*—The MPT network layer multipath communication library is a novel solution for several problems including IPv6 transition, reliable data transmission using TCP, real-time transmission using UDP and also wireless network layer routing problems. MPT can provide an IPv4 or an IPv6 tunnel over one or more IPv4 or IPv6 communication channels. MPT can also aggregate the capacity of multiple physical channels. In this paper, the channel aggregation capability of the MPT library is measured up to twelve 100Mbps speed channels. Different scenarios are used: both IPv4 and IPv6 are used as the underlying and also as the encapsulated protocols and also both UDP and TCP are used as transport protocols. In addition, measurements are taken with both 32-bit and 64-bit version of the MPT library. In all cases, the number of the physical channels is increased from 1 to 12 and the aggregated throughput is measured.**

***Keywords*—channel capacity aggregation, network layer multipath communication, performance analysis, TCP/IP protocol stack, tunneling**

I. Introduction

Multipath communication is a hot research topic today. There were different solutions invented: the multipath technology can be used in different layers (link layer, network layer, transport layer) see our little survey in the next section. Now, we focus on the MPT network layer multipath communication library [1], which one was developed at the *Faculty of Informatics*, *University of Debrecen*, Debrecen, Hungary. It can be freely downloaded for 32-bit and 64-bit Linux operating systems as well for Raspberry Pi from [2]. It makes possible to aggregate the transmission capacity of multiple interfaces of a device. Its performance, especially its channel aggregation capability for two channels was analyzed in [3] and for four channels in [4] using serial links with the speed of a few megabits per second.

We measured the channel aggregation capability of the MPT network layer multipath communication library using significantly increased number of physical channels and transmission speed compared to the earlier test of other researchers [3] and [4]. Our preliminary results concerning the 32-bit version of the MPT library measured by the industrial standard `iperf` tool using the UDP transport layer protocol were published in our conference paper [5], which one is now extended with the HTTP measurements (using TCP) and with the testing of the 64-bit version of the MPT library.

The remainder of this paper is organized as follows. First, the different multipath solutions are surveyed in a nutshell. Second, a brief introduction is given to the MPT network layer multipath communication library. Third, our test environment is described. Fourth, our experiments are described, the results of our high number of measurements are presented and discussed. Fifth, the directions of our future research are outlined. Finally, our conclusion is given.

Manuscript received February 26, 2015, revised May 9, 2015.

G. Lencse is with the Department Telecommuications, Széchenyi István University, Győr, Hungary (phone: +36-96-613-665, fax: +36-96-613-646, e-mail: lencse@sze.hu)

Á. Kovács is with the Department Telecommuications, Széchenyi István University, Győr, Hungary (e-mail: kovacs.akos@sze.hu)

II. A Short Survey of Multipath Solutions

A. Multipath TCP – a Transmission Layer Solution

Multipath TCP [6] is probably the most well-known multipath solution. MPTCP uses multiple TCP sub-flows on the top of potentially disjoint paths, see Fig. 1. Therefore it can be used for the aggregation of the transmission capacity of the underlying paths. Its channel aggregation can be very efficient: a single data-stream was transmitted at the rate of 50Gbps over six 10Gbps Ethernet Links using MPTCP [7]. MTPCP is actively researched and analyzed from different viewpoints see e.g. [8] and its references or count the Google Scholar hits for "Multipath TCP".

However, multipath TCP has its limitations and drawbacks, too. TCP provides a reliable byte stream transmission, which one is appropriate for several applications such as web browsing, sending or downloading e-mails, etc. However, its retransmission mechanism is undesirable for other applications such as IP telephony, video conference or other real-time communications where some packet loss (with low ratio) can be better tolerated than high delays caused by TCP retransmissions. Consequently, multipath TCP is not suitable for these types of applications.

B. MPT Library – the Only Network Layer Solution

The MPT network layer multipath communication library [1] uses UDP/IP protocols on the top of each link layer connection and creates an IP tunnel over them. Thus both TCP and UDP can be used over the IP tunnel, see Fig. 2. Therefore retransmissions can be omitted if they are not required. This design makes MPT more general than MPTCP thus permitting MPT more areas of applications.

The MPT library may be used for many different purposes including file and stream transmission [4], cognitive infocommunication [9], wireless network layer roaming problems

[10] and changing the communication interfaces (using different transmission technologies) without packet loss [11] (it is also called vertical handover between 3G and WiFi). For further publications about MPT, see [12] and [13].

As far as we know, MPT is the only network layer multipath communication solution.

C. OLiMPS – a Link Layer Solution

The Openflow Link-Layer Multipath Swithcing [14] is a novel solution, which uses the logic of the link-layer, that is, it calculates routes as if the nodes were connected with LANs, however, it can also operate over WANs [15].

D. Other Similar Solutions

There are some other solutions, which deal with multiple interfaces, however they are not always real multipath solutions.

The *Multiple Interfaces* Working Group of IETF has already produced many useful documents [16]. They focus on the problem that a host has multiple interfaces which are connected to different provisioning domains [17] and the interfaces can be simultaneously used for communication. It is not necessarily a multipath solution: for example, one application may use the first interface, and another one may use the second one.

Proxy Mobile IPv6 [18] allows a mobile node to connect to the same PMIPv6 domain through different interfaces. The *NETEXT* Working Group of IETF proposed a draft RFC [19] which specifies protocol extensions to PMIPv6 to distribute specific traffic flows on different physical interfaces.

III. MPT in a Nutshell

A. The Architecture of MPT

Fig. 2 shows the layered architecture of the MPT network layer multipath communication library. The most important difference from MPTCP is that MPT creates a new logical interface on the endpoint host, through which the applications can communicate, therefore the applications can use any transport layer protocol: either TCP or UDP, whichever is appropriate for them. The MPT software processes the packets from the tunnel interface. MPT makes a packet-by-packet decision about which path to choose and then encapsulates the packet into a new UDP/IP packet and finally sends it out through the appropriate link-layer interface [1].

B. The Configuration and Usage of the MPT Library

The MPT library distribution contains an easy to follow user guide [20]. To be able to use MPT between two computers, the software must be installed on both of them. One of them should be configured as *server* and the other one as *client*, but the applications see it completely symmetrical. The MPT library has simple and straight forward configuration files where the different parameters (e.g. the number of physical connections, the Linux network interface names and IP addresses for each channel, the name of the tunnel interface, etc.) can be set. When both sides are configured and the MPT

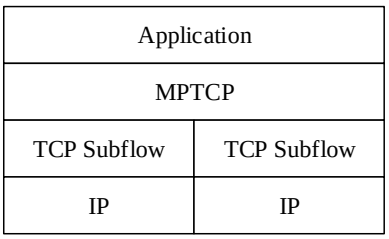

Fig. 1. The architecture of the MPTCP protocol stack [6]

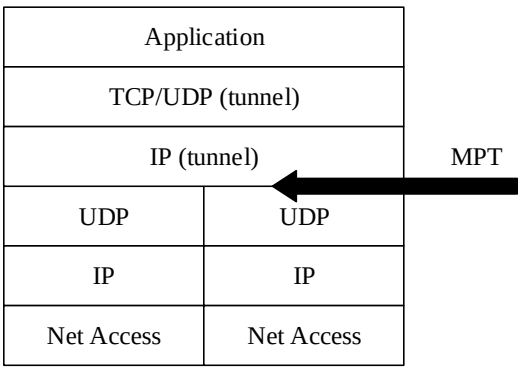

Fig. 2. The layered architecture of the MPT software [3]

software is started on both computers, the applications can use the tunnel interfaces for communication in the usual way. The MPT library distributes the user's traffic for all the configured physical channels thus the user can take the advantage of the multiple network interfaces.

IV. Test Environment

A. Hardware and Basic Configuration

Two DELL Precision Workstation 490 computers were used for our tests. Their basic configuration was:

- DELL 0GU083 motherboard with Intel 5000X chipset
- Two Intel Xeon 5140 2.33GHz dual core processors
- 8x2GB 533MHz DDR2 SDRAM (accessed quad channel)
- Broadcom NetXtreme BCM5752 Gigabit Ethernet controller (PCI Express, integrated)

Three Intel PT Quad 1000 type four port Gigabit Ethernet controllers were added to each computers. The 3x4=12 Gigabit Ethernet ports were used for the measurements and the integrated one was used for control purposes. The computers were interconnected by a Cisco Catalyst 2960 switch limiting the transmission speed to 100Mbps and separating the 12 physical connections by VLANs.

In our experiments, both IPv4 and IPv6 was used as the underlying and as the tunnel IP version (it means 2x2 series of experiments). Fig. 3 shows the topology and the IP address configuration of the test network used in the IPv4 tunnel over

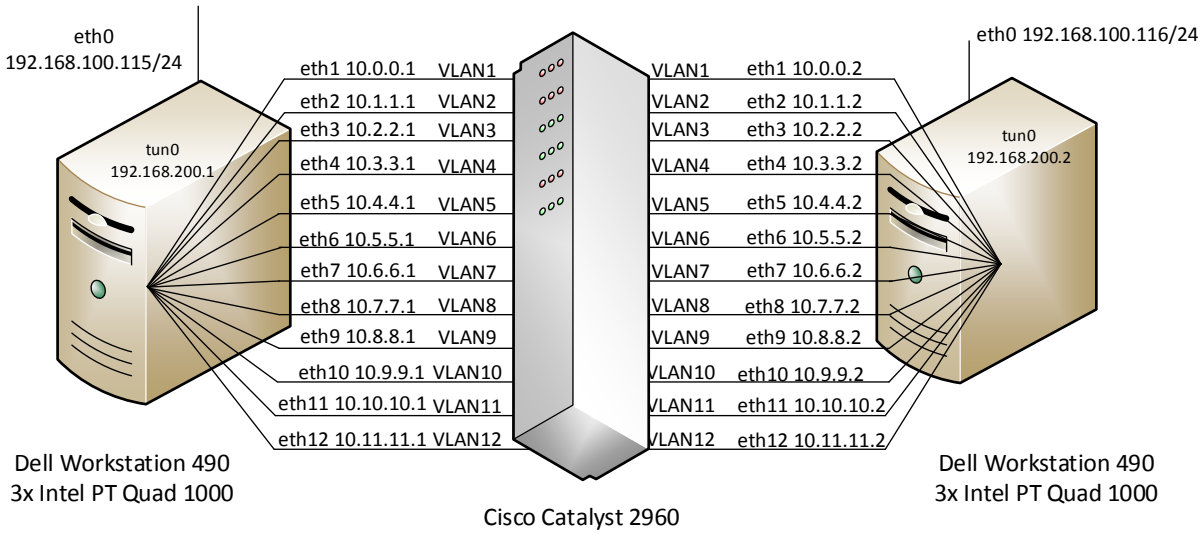

Fig. 3. The topology of the test network (IPv4 tunnel over IPv4 connections)

IPv4 connections tests. The same topology was used for the other three experiments, too. Debian wheezy 7.4 GNU/Linux operating system was installed on both computers.

B. Configuration of the MPT Software

The version of the MPT library can be identified by the name of the file which contains the date in the YYYY-MM-DD format: mpt-lib-2014-03-25.tar.gz was used first. This version of the MPT library contained precompiled 32-bit executables with statically linked libraries thus we did not need to compile it. The contents of the following two configuration files were set as follows. (Their path is relative to the installation directory of MPT.) The beginning of the conf/interface.conf file was:

```
########## Interface Information: ############
12      # The number of the interfaces
65020   # The local cmd port number
1       # Accept remote new connection request
########## Tunnel interface #################
tun0    # INT. NAME, must be the tunnel interface
192.168.200.1/24   #IPv4 address and prefix length
fd00:de:200::1/64  #IPv6 address and prefix length
############## ETH1 interface ################
eth1
10.0.0.1/24
fd00:de:201::1/64
############## ETH2 interface ################
eth2
10.1.1.1/24
fd00:de:202::1/64
```

And it was similar for all the other interfaces, which we do not list to save space. The different types of tunnels were specified in separate connection files. The IPv4 tunnel over IPv4 paths was defined in the conf/connections/IPv4overIPv4.conf file:

```
#### Multipath Connection Information: ####
1       # The number of the connections
########## New Connection ################
TILB    # CONNECTION NAME
3       # SEND(1)/RECEIVE(2) CONNECTION UPDATE
4       # IP VERSION
192.168.200.1   # LOCAL IP
65022           # LOCAL DATA PORT
192.168.200.2   # REMOTE IP
65022           # REMOTE DATA PORT
65020           # REMOTE CMD PORT
12              # NUMBER OF PATHS
0               # NUMBER OF NETWORKS
2               # KEEPALIVE TIME (sec)
5               # DEAD TIMER (sec)
0               # CONNECTION STATUS
0               # AUTH. TYPE
0               # AUTH. KEY
######### Path 0 information: ################
eth1                    # INT. NAME
4                       # IP VERSION
00:15:17:54:d7:30       # LOCAL MAC ADDR
10.0.0.1                # LOCAL IP
00:00:00:00:00:00       # GW MAC ADDR
0.0.0.0                 # GW IP
10.0.0.2                # REMOTE IP
100                     # WEIGHT IN
100                     # WEIGHT OUT
1                       # PATH WINDOW SIZE
0                       # PATH STATUS
######## Path 1 information: ################
eth2                    # INT. NAME
4                       # IP VERSION
00:15:17:54:d7:31       # LOCAL MAC ADDR
10.1.1.1                # LOCAL IP
00:00:00:00:00:00       # GW MAC ADDR
0.0.0.0                 # GW IP
10.1.1.2                # REMOTE IP
100                     # WEIGHT IN
100                     # WEIGHT OUT
1                       # PATH WINDOW SIZE
0                       # PATH STATUS
```

It was also set in the same manner for all the other paths of this connection and for the other connections as well. Note that the configuration files followed strict format, even the comment only lines had to be present. We recommended this to be changed for the commonly used free style configuration files with keyword parsing in [5]. The authors of MPT responded quickly and keyword parsing is provided in the most current version of MPT [2].

V. Experiments and Results

The channel aggregation capability of the MPT library was measured with two different methods: using the industrial de facto standard `iperf`, and file transfer by the `wget` Linux program over the HTTP[1] protocol. These two methods were selected because `iperf` uses UDP and `wget` uses TCP as transport layer protocols. As it was mentioned before, both IPv4 and IPv6 were used as the IP protocol for the tunnel and also as the IP protocol for the underlying channels. In addition to that, both 32-bit and 64-bit versions of the MPT library were tested. It means altogether 2x2x2x2=16 series of measurements, were the number of physical channels were increased from 1 to 12. Thus we performed 16x12=192 different tests. The tests were automated by scripts. Due to space limitations, we cannot include the complete measurement scripts, but the key commands only. The ones below belong to the IPv4 tunnel over IPv4 measurements. The `iperf` command was:

```
iperf -c 192.168.200.1 -t 100 -f M
```

This command performed a 100 seconds long test and printed the throughput in MB/s units. This is called the client side in `iperf` terminology. On the other side, the server was started with the following command line:

```
iperf -s
```

A file of 1GiB size was downloaded using HTTP with the following command line:

```
wget -O /dev/null http://192.168.200.1/1GB
```

This command downloaded the file but did not write it on the hard disk rather disposed it in `/dev/null` so that the disk writing speed would not influence our measurement results. And also the file named 1GB was put on RAM drive at the server computer to eliminate the reading from the hard disk.

The results of our measurements using the 32-bit MPT library are discussed first in details and the 64-bit results are presented later. And within the 32-bit results, we begin with the results of the `iperf` measurements; now they are presented and then discussed.

A. Results of the Iperf Measurements

The results of the `iperf` test are shown in Fig. 4. Whereas two of them (IPv4 over IPv4 and IPv6 over IPv4) are nearly linear in the whole range, the two other ones (IPv4 over IPv6 and IPv6 over IPv6) are nearly linear until 7 NICs and then they show saturation or even a small degradation until the end of the range. Our results suggest that only the version of the underlying IP protocol makes a significant difference in the channel capacity aggregation performance of the MPT library and the version of the encapsulated IP has only a minor influence on it.

When the underlying protocol was IPv4, the throughput was linear up to 12 NICs, which means that the throughput aggregation capability of the MPT library proved to be very good, and we could not reach the limits of MPT library. (These

[1]File transfer by FTP was also tested, but its performance results were so close to that of HTTP that they were finally omitted.

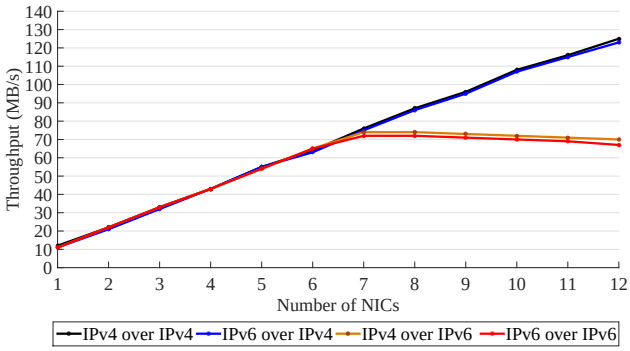

Fig. 4. Throughput results of `iperf` tests

results are very important, because MPT has been tested up to 4 physical channels having only a few Mbps speed before our experiments.)

When the underlying protocol was IPv6, the performance limit of the system was reached at 7 NICs. The maximum values were 74MB/s and 72MB/s in the case of the IPv4 over IPv6 and IPv6 over IPv6 tests, respectively. (The further increase of the number of NICs resulted in some degradation of the throughput, their respective values were 70MB/s and 67MB/s at 12 NICs.) Note that this is the performance of our system composed of the above described hardware and software. We asked ourselves whether it was a built-in limit of the MPT library or it was the performance limit of the hardware that we used for testing?

B. Investigation of the Reason of the IPv6 Performance Limit

1) Checking the CPU utilization: We measured the CPU utilization of the MPT software during the experiments on both the client and on the server during all the 4 series of experiments thus we got 2x4=8 graphs. The CPU usage of the MPT client and of the MPT server was practically the same. The version of the upper IP protocol made no significant difference, therefore we include only two significantly different ones of them. The CPU utilization of the MPT client during the IPv4 over IPv4 measurements is shown in Fig. 5. Even though the time scale is not presented (because no timestamps were logged with the CPU utilization values), the 12 measurements can be easily identified: they are separated by gaps with 0% CPU usage between them. The CPU utilization shows some fluctuations, but its near linear growth can be well observed. It reached the 160-180% interval at 12 NICs. It was checked that the CPU utilization of the `iperf` program was always under 50% thus there was free CPU capacity available from the 400% of the four CPU cores. The CPU utilization of the MPT client during the IPv6 over IPv6 measurements is shown in Fig. 6. It reached 160% at 7 NICs and it fluctuated around 160% for higher number of NICs. There is a visible correspondence between the CPU utilization and the throughput, see Fig. 4.

2) Measurements with faster CPUs: The Intel Xeon 5140 2.33GHz dual core processors of the test computers were replaced by Intel Xeon 5160 3GHz dual core processors. The

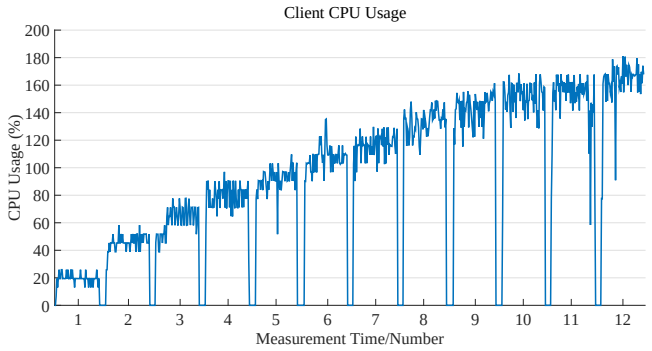

Fig. 5. MPT CPU utilization, IPv4 tunnel over IPv4

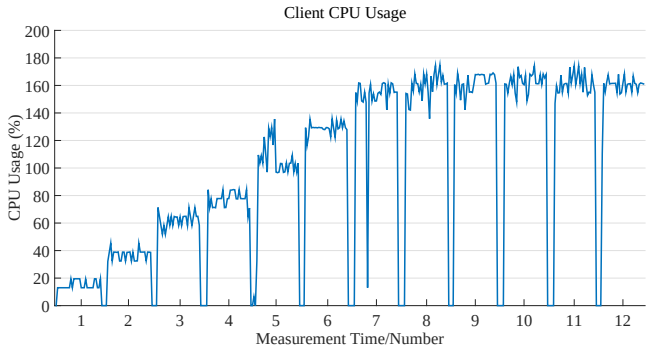

Fig. 6. MPT CPU utilization, IPv6 tunnel over IPv6

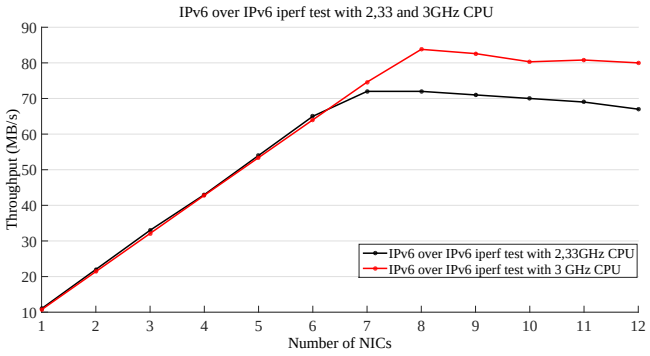

Fig. 7. The throughput results of the `iperf` test of an IPv6 tunnel over IPv6 using 3GHz CPUs

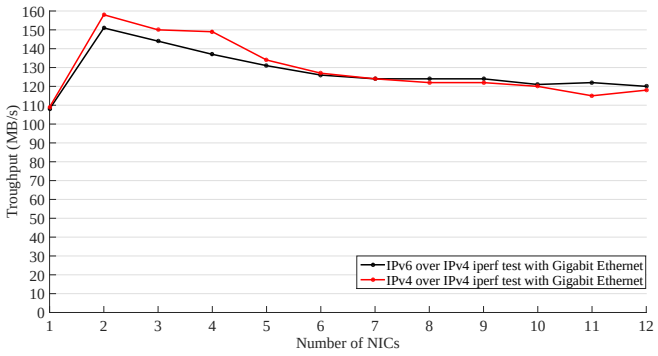

Fig. 8. The throughput results of the `iperf` test of an IPv6 or IPv4 tunnel over IPv4 using Gigabit Ethernet

IPv6 tunnel over IPv6 paths experiments were repeated with the faster CPUSs. Fig. 7 shows the throughput results. It can be observed that the faster CPUs made it possible to fully utilize the capacity of 8 NICs and the degradation started from 9 NICs. This result convinced us that the aggregation capability of MPT does not have a built-in limit, rather it depends on the performance of the CPUs.

However, a question now arises: why could not MPT increase its CPU utilization above 180% while there was still free CPU capacity? The answer is that MPT was written as a serial program and thus it is not able to fully utilize the available processing power of the multiple CPU cores. (The higher than 100% utilization is probably achieved by the overlapping of sending and receiving packets.) We believe that it would be worth improving MPT in this field, because the current trend of the evolution of the CPUs is that the number of cores is increased instead of the clock speed.

After the completion of these measurements, the original Intel Xeon 5140 2.33GHz dual core processors were put back into the test computers and they were used in all the following experiments.

C. *Investigation of the IPv4 Performance Limit*

As it can be seen in Fig. 4, the throughput scaled up nearly linearly up to 12 NICs when the underlying protocol was IPv4. We were interested in the performance limit of the system, but we could not insert more NICs into our Dell computers as they had only 3 PCI Express slots. Therefore, we increased the speed of the NICs to 1Gbps by removing the Cisco switch and interconnecting the two times 12 Ethernet ports of the two computers directly.

The results are shown in Fig. 8. In both tests, the throughput reached its maximum value (of 158MB/s and 151MB/s when the tunnel protocol was IPv4 and IPv6, respectively) at 2 NICs and it degraded for higher number of NICs (down to 118MB/s and 120MB/s at 8 NICs), but it remained still higher than the throughput of a single NIC. This is in correspondence with the values of the CPU utilization in Fig. 9. (The graph actually shows the CPU utilization of the IPv4 over IPv4 case, but the CPU utilization of the IPv6 over IPv4 case looked the same, thus we did not included it.)

D. *Results of the Wget Measurements*

The results are shown in Fig. 10. Unlike with the `iperf`, performance limits can be observed in each graph, and there are also differences between the first two graphs. The HTTP performance of the IPv4 tunnel over IPv4 shows somewhat saturation at 11 and 12 NICs, but the performance is still growing. The HTTP performance of the IPv6 tunnel over IPv4 shows not only saturation but even it definitely degrades at the end of the graph (from 100MB/s at 10 NICs to 90 MB/s at 12 NICs). The HTTP throughput of the IPv4 tunnel over IPv6 reaches its maximum value of 70MB/s at 7 NICs, and it degrades for higher number of NICs (its value is 60MB/s

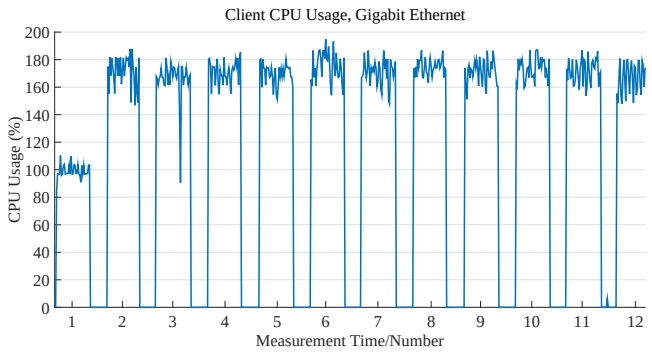

Fig. 9. MPT CPU utilization (from the total of 400%), IPv4 tunnel over IPv4, Gigabit Ethernet

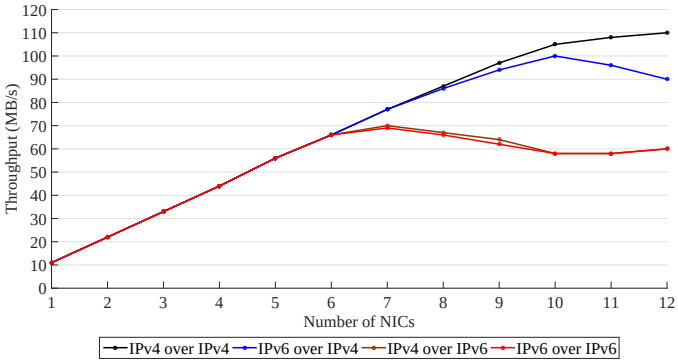

Fig. 10. Throughput results of `wget` tests

at 12 NICs). The HTTP performance of the IPv6 tunnel over IPv6 is nearly exactly the same.

Our HTTP throughput results confirm that the version of the underlying IP protocol makes the major difference in the channel capacity aggregation performance of the MPT library, but they indicate that the version of the encapsulated IP may also have a minor influence on it. However, the results of the `wget` measurements differ from the results of the `iperf` measurements because now we could reach the performance limits of our test system even when the underlying protocol was IPv4. Very likely it is caused by the higher CPU usage of the TCP protocol stack than that of the much simpler UDP. When the underlying protocol was IPv6, we reached the HTTP performance limit of the system at 7 NICs. The further increase of the number of NICs resulted in some degradation of the throughput.

E. Results with the 64-bit MPT Library

The authors of MPT library published the precompiled 64-bit version after the completion of our measurements for [5]. There we mentioned our intention of testing the 64-bit version to see if there is a difference in the performance of the 32-bit and the 64-bit version of the MPT library. We expected that the 64-bit version may more effectively handle the 128 bits long IPv6 addresses. The 64-bit results are presented in the same order as the 32-bit ones: first the `iperf` results and then the `wget` results.

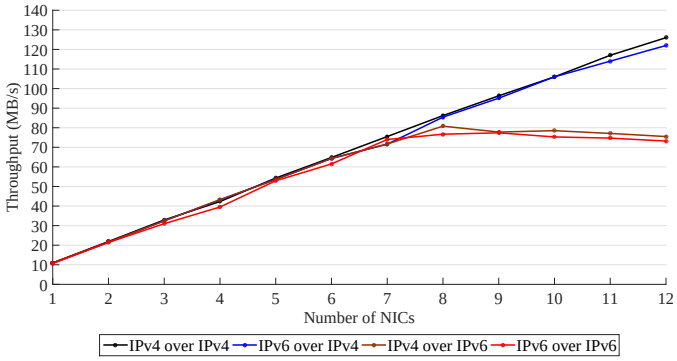

Fig. 11. Throughput results of `iperf` tests (64-bit)

1) Results of the iperf measurements: The results of the 64-bit `iperf` test are shown in Fig. 11. When IPv4 was used as the underlying protocol, the throughput scaled up nearly linearly up to 12 NICs, as we expected. When IPv6 was used as the underlying protocol, the throughput reached its maximum value of at 8 NICs. In the IPv4 over IPv6 case, the maximum value of the throughput was 81MB/s at 8 NICs, which is only by 7MB/s higher than that for the 32-bit case, where maximum value of 74MB/s (see Fig. 4) in throughput has been reached already at 7NICs.

The 64-bit library did not result in the convincing performance improvement that we expected before.

2) Results of the wget measurements: The results of the 64-bit `wget` test are shown in Fig. 12. The graphs are rather similar to graphs of the 32-bit case (see Fig. 10), though the throughput results are somewhat better here. The HTTP perfomance of the IPv4 over IPv4 is linear up to 11 NICs (instead of 10). The HTTP performance of the IPv6 tunnel over IPv4 shows no performance degradation for 11 and 12 NICs, what is an advantage of the 64-bit version over the 32-bit version of the MPT library. The HTTP performance of the IPv4 tunnel over IPv6 reaches its maximum value at 7 NICs. The maximum place of the throughput result curve of the 32-bit test is the same (Fig. 10), but here the maximum value is a little bit higher: 74.4MB/s instead of 70MB/s. And the linear degradation here is bit better than the degradation was in the 32-bit case. The HTTP performance of the IPv6 tunnel over IPv6 is also somewhat better, but rather similar to that of the 32-bit case.

Though the 64-bit version of the MPT library did not fulfill our performance expectations, but the 64-bit results are definitely never worse than those of the 32-bit version, and in many cases the 64-bit version brings some slight performance increase.

VI. Directions of our Future Research

So far, we have tested the performance and throughput aggregation capability of the MPT library in itself. We also plan to compare them with that of the standard MPTCP.

As the most important advantage of MPT over MPTCP is that MPT uses UDP/IP and therefore it is much suitable for use with real-time applications because of the elimination of

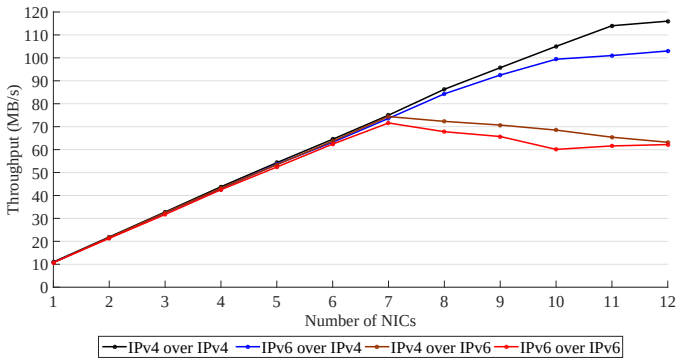

Fig. 12. Throughput results of `wget tests` (64-bit)

TCP retransmissions, we also plan to test it with real-time applications.

We also intend to test MPT as a tunneling tool. MPT seems to be a universal tunnel software in the context of IPv6 transition since it can be used as either of an IPv4 or an IPv6 tunnel over either of IPv4 or IPv6 connections.

VII. Conclusion

The throughput aggregation performance of the MPT network layer multipath communication library was examined up to twelve 100Mbps link layer connections. Measurements were taken with both `iperf` (over UDP) and `wget` (over TCP) using both 32-bit and 64-bit MPT libraries.

As for the 32-bit MPT library and `iperf` measurements, when the underlying protocol was IPv4, the throughput scaled up linearly up to 12 NICs (exceeding 120MB/s) regardless of the version of the encapsulated IP (IPv4 or IPv6). When the underlying protocol was IPv6, the throughput scaled up linearly up to 7 NICs (exceeding 70MB/s) regardless of the version of the encapsulated IP, but it could not increase more for higher number of NICs rather it showed a small degradation.

It was proved that the above performance limit depends on the computing power of the CPUs and it is not a fixed built in feature of the MPT library.

MPT was also tested with 12 Gigabit Ethernet connections to find the performance limit of our system when the underlying protocol was IPv4. It was reached at two NICs having the values of 158MB/s and 151MB/s when the tunnel protocol was IPv4 and IPv6, respectively.

As for the 32-bit MPT library and `wget` measurements, the results were similar to those of the `iperf` measurements with the exception, that we could reach the performance limit of the system even when the underlying protocol was IPv4 due to the higher CPU usage of the TCP protocol stack than that of the much simpler UDP.

As for the measurements with the 64-bit MPT library (using both `iperf` and `wget`), the results were close to the results of the measurements with the 32-bit MPT library, producing only usually a little performance benefit depending on the given test but the 64-bit results were never worse than the 32-bit ones.

We conclude the MPT network layer multipath communication library proved to be a good tool for the aggregation of the capacity of several high speed channels.

References

[1] B. Almási, A. Harman, "An overview of the multipath communication technologies", In *Proc. Conf. on Advances in Wireless Sensor Networks 2013 (AWSN 2013)*, Debrecen University Press, Debrecen, Hungary, ISBN: 978-963-318-356-4, pp. 7–11, 2013.

[2] B. Almási, "MPT library", precompiled version can be downloaded from: `http://irh.inf.unideb.hu/user/almasi/mpt/`

[3] B. Almási, Sz. Szilágyi, "Throughput performance analysis of the multipath communication library MPT", In *Proc. 36th Int. Conf. on Telecommunications and Signal Processing (TSP 2013)*, Rome, Italy, Jul. 2-4, 2013, pp. 86–90. DOI: 10.1109/TSP.2013.6613897

[4] B. Almási, Sz. Szilágyi, "Multipath ftp and stream transmission analysis using the MPT software environment", *Int. J. of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 11, pp. 4267–4272. Nov. 2013.

[5] G. Lencse, Á. Kovács, "Testing the channel aggregation capability of the MPT multipath communication library", In *Proc. World Symposium on Computer Networks and Information Security 2014 (WSCNIS 2014)*, Hammamet, Tunisia, Jun. 13-15, 2014, ISBN: 978-9938-9511-9-6, Paper ID: 1569946547.

[6] A. Ford, C. Raiciu, M. Handley and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses", IETF, Jan. 2013, RFC 6824.

[7] C. Paasch, G. Detal, S. Barré, F. Duchêne, O. Bonaventure, "The fastest TCP connection with multipath TCP", ICTEAM, UCLouvain, Louvain-la-Neuve, Belgium, available: `http://multipath-tcp.org/pmwiki.php?n=Main.50Gbps`

[8] R. Khalili, N. Gast, M. Popovic, J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: performance issues and a possible solution", *IEEE/ACM Trans. Networking*, vol. 21, no. 5, pp. 1651–1665, Oct. 2013, DOI: 10.1109/TNET.2013.2274462

[9] B. Almási, "Multipath communication – a new basis for the future Internet cognitive infocommunication", In *Proc. CogInfoCom 2013 Conf.*, Budapest, Hungary, Dec. 2-5, 2013, pp. 201–204, DOI: 10.1109/CogInfoCom.2013.6719241

[10] B. Almási, "A simple solution for wireless network layer roaming problems", *Carpathian Journal of Electronic and Computer Engineering*, vol. 5, no. 1, pp. 5–8, 2012.

[11] B. Almási, "A solution for changing the communication interfaces between WiFi and 3G without packet loss", In *Proc. 37th Int. Conf. on Telecommunications and Signal Processing (TSP 2014)*, Berlin, Germany, Jul. 1-3, 2014, pp. 73–77.

[12] Z. Gál, B. Almási, T. Dabóczi, 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, pp. 37–44, Sep. 2014.

[13] B. Almási, Sz. Szilágyi, "Investigating the throughput performance of the MPT multipath communication library in IPv4 and IPv6", *Journal of Applied Research and Technology*, to be published

[14] H. Newman, A. Barczyk, M. Bredel, "OLiMPS: openflow link-layer multipath switching", *DOE ASCR NGN PIs Meeting*, Rockville, Sept. 16-17, 2014, slides available: `http://www.orau.gov/ngnspi2014/presentations/newman_h.pdf`

[15] A. Barczyk, M, Bredel, H. Newman, R. van der Pol, "Openflow-based multipath networks in the WAN", In *Proc. TERENA Networking Conference (TNC 2013)*, Maastricht, Netherlands, Jun. 3-6, 2013, available: `https://tnc2013.terena.org/getfile/192`

[16] Multiple Interfaces Working Group, "Mif status pages", available: `https://tools.ietf.org/wg/mif/`

[17] M. Blanchet, P. Seite, "Multiple interfaces and provisioning domains problem statement", IETF, Nov. 2011, RFC 6418.

[18] S. Gundavelli (Ed.), K. Leung, V. Devarapalli, K. Chowdhury, B. Patil, "Proxy mobile IPv6", IETF, Aug. 2008, RFC 5213.

[19] CJ. Bernardos (Ed.), "Proxy mobile IPv6 extensions to support flow mobility", IETF, Draft, available: `https://tools.ietf.org/html/draft-ietf-netext-pmipv6-flowmob-12`

[20] B. Almási, "MPT library user guide", can be downloaded from: `http://irh.inf.unideb.hu/user/almasi/mpt/`

Gábor Lencse received his MSc in electrical engineering and computer systems at the Technical University of Budapest in 1994, and his PhD in 2001.

He has been working for the Department of Telecommunications, Széchenyi István University in Győr since 1997. He teaches Computer networks, Computer architectures, IP-based telecommunication systems and the Linux operating system. Now, he is an Associate Professor. He is responsible for the specialization of the information and communication technology of the BSc level electrical engineering education. He is a founding member of the Multidisciplinary Doctoral School of Engineering Sciences, Széchenyi István University. The area of his research includes discrete-event simulation methodology, performance analysis of computer networks and IPv6 transition technologies. Dr. Lencse has been working part time for the Department of Networked Systems and Services, Budapest University of Technology and Economics (the former Technical University of Budapest) since 2005. There he teaches Computer architectures and Computer networks.

Ákos Kovács received MSc in electrical engineering with specialization in infocommunication systems and services at the Széchenyi István University in 2013.

He started working as laboratory engineer at the Department of Telecommunications in 2008. During this time he got familiar with high-end computer systems, virtualization and cloud computing. He also has high skills in the field of computer networks, and networking security. He teaches computer networks and virtualization technology in BSc and holds practical lessons in MSc in the field of IP-based telecommunication.