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

Many of our ICT devices (e.g. smart phones, tablets, notebooks) have multiple communication interfaces (e.g. Ethernet, WiFi, HSDPA/LTE) but we can use only one of them at a time due to technical reasons: the endpoint of a TCP/IP communication is identified by an IP address plus a port number and the IP addresses are always bound to the network interfaces [1]. The MPT network layer multipath communication library [2] was developed at the Faculty of Informatics, University of Debrecen, Debrecen, Hungary. It makes possible to aggregate the transmission capacity of the 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. The MPT library may be useful for many different purposes including stream transmission [4], cognitive info-communication [5], wireless network layer roaming problems [6] and changing the communication interfaces [7] (it is also called vertical handover between 3G and WiFi). For further publications about MPT, see [3] and [9].

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 measured by the industrial standard iperf tool using the UDP transport layer protocol were published in a conference paper [10], 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, a brief introduction is given to the MPT network layer multipath communication library. Second, our test environment is described. Third, our experiments are described, the results of our high number of measurements are presented and discussed. Fourth, the directions of our future research are outlined and a recommendation is given for the further development of the MPT library. Finally, our conclusions are given.

II. MPT IN A NUTSHELL

A. The Architecture of MPT

The innovation of the MPT network layer multipath communication library can be highlighted by a comparison with the much more well-known Multipath TCP [11]. MPTCP uses multiple TCP sub-flows on the top of potentially disjoint paths, see Fig. 1. This is a good solution for the aggregation of the transmission capacity of the underlying paths. The reliable byte stream transmission offered by TCP is a proper solution for a class of applications such as web browsing, sending or downloading e-mails, etc. However, it is undesirable for another class of 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. The MPT network layer multipath communication library uses UDP/IP protocols on the top of each link layer connection and creates an IP tunnel on 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.

B. The Configuration and Usage of the MPT Library

MPT is network layer multipath communication library developed under Linux and can be downloaded from [10]. The distribution contains an easy to follow user manual. When setting up MPT, the software must be present at both endpoints. One of them should be configured as server and the other one as client, but the applications see it completely
The architecture of the MPTCP protocol stack

<table>
<thead>
<tr>
<th>Application</th>
<th>MPTCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Subflow</td>
<td>TCP Subflow</td>
</tr>
<tr>
<td>IP</td>
<td>IP</td>
</tr>
</tbody>
</table>

Fig. 1. The architecture of the MPTCP protocol stack [11]

The layered architecture of the MPT software

<table>
<thead>
<tr>
<th>Application</th>
<th>TCP/UDP (tunnel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP (tunnel)</td>
<td>UDP</td>
</tr>
<tr>
<td>IP</td>
<td>IP</td>
</tr>
<tr>
<td>Net Access</td>
<td>Net Access</td>
</tr>
</tbody>
</table>

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

symmetrical. It has simple and straightforward configuration files where the details must be given (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.), see more details later on. When both sides are configured and the MPT software is started on both computers, the applications can use the tunnel interfaces for communication in the usual way. It is the task of the MPT library to distribute the user’s traffic for all the physical channels to be able to take the advantage of the multiple network interfaces.

### III. 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, thus they both had 3x4+1=13 Gigabit Ethernet ports, from which the integrated one was used for control purposes and the other ones were used for the measurements. The computers were interconnected by a Cisco Catalyst 2960 switch limiting the transmission speed to 100Mbps and separating the 12 physical connections by VLANs.

Different versions of IP (v4 or v6) were used for our experiments. Fig. 3 shows the network that was used in the IPv4 tunnel over IPv4 connections tests. Debian wheezy 7.4 GNU/Linux operating system was installed on both computers. For the first series of experiments, the network interfaces of the computers were configured as shown in Fig. 3.

#### 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. We set it up following the instructions of the user manual [2]. It was a simple and straightforward task. 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:

```bash
# Interface Information: #
# The number of the interfaces 65020 # The local cmd port number
1 # Accept remote new connection request
# Tunnel interface #
# IPv4 address and prefix length 192.168.200.1/24 # 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 due to space limitations. Whereas the IP settings of the interfaces could be described in a common file for IPv4 and IPv6, the different types of tunnels are to be given in separate connection files. The IPv4 tunnel over IPv4 paths was defined in the conf/connections/IPv4overIPv4.conf file:

```bash
# 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 # NUMBER OF PATHS
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
00:15:17:54:d7:30 # 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
```

And it was similar for all the other interfaces, which we do not list due to space limitations. Whereas the IP settings of the interfaces could be described in a common file for IPv4 and IPv6, the different types of tunnels are to be given in separate connection files. The IPv4 tunnel over IPv4 paths was defined in the conf/connections/IPv4overIPv4.conf file:
responded quickly and keyword parsing is provided in the most files with keyword parsing in [10]. The authors of MPT to be changed for the commonly used free style configuration comment only lines had to be present. We recommended this that the configuration files followed strict format, even the 

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Eth0</td>
<td>0.192.168.100.115/24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth1</td>
<td>10.0.0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth2</td>
<td>10.1.1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth3</td>
<td>10.2.2.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth4</td>
<td>10.3.3.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth5</td>
<td>10.4.4.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth6</td>
<td>10.5.5.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth7</td>
<td>10.6.6.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth8</td>
<td>10.7.7.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth9</td>
<td>10.8.8.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth10</td>
<td>10.9.9.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth11</td>
<td>10.10.10.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eth12</td>
<td>10.11.11.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tun0</td>
<td>192.168.200.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 [10]. The authors of MPT responded quickly and keyword parsing is provided in the most current version of MPT [12].

### IV. 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** protocol. These two methods were selected because **iperf** uses **UDP** and **wget** uses **TCP** as transport layer protocols. In addition to that, 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 MBytes/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. Following the order of [10], (which contained only **iperf** measurements using the 32-bit MPT library) 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 the MPT library. (These 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, we reached the performance limit of the system at 7 NICs. The further increase of the number of NICs could not result in the increase of the throughput, rather some degradation of the throughput can be observed. At this point, we may only state that this is the performance of our system composed of the above described hardware and software. But we are interested in the limits of the MPT library and not that of the hardware used for testing.

### B. Investigation of the Reason of the IPv6 Performance Limit

1) Checking the CPU utilization: We have checked and logged the CPU utilization of the MPT software during the measurements. We did so on both the client and on the server during all the 4 series of measurements 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 typical ones of them. Fig. 5 shows the CPU utilization of the MPT client during the IPv4 over IPv4 measurements. The gaps with 0% CPU usage can be easily identified. Even though CPU utilization shows some fluctuations, its near linear growth can be observed. It reached 160-180% at 12 NICs. Note 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. Fig. 6 shows the CPU utilization of the MPT client computer during the IPv6 over IPv6 measurements. The CPU utilization 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 throughput of the IPv6 tunnel over IPv6 paths scenario was measured and Fig. 7 shows the results. The faster CPUs made it possible to fully utilize the capacity of 8 NICs and the degradation started from 9 NICs. This is an important result because it convinced us that the aggregation capability of MPT does not have a built in limit, rather it depends on the performance of the CPUs. It is another issue 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. As for the current trend of the evolution of the CPUs,
it would be desirable to improve MPT in this field.

C. Investigation of the IPv4 Performance Limit

We could not reach the throughput capacity limit of the system in the two tests where the underlying protocol was IPv4. As our Dell computers had only 3 PCI Express slots, we could not insert more NICs. Therefore we removed the Cisco switch and interconnected the 12 Ethernet ports of the two computers directly, thus they were enabled to operate in gigabit mode. (The original 2.33GHz CPUs were kept.) The results of the IPv4 over IPv4 tests are shown in Fig. 8. The throughput reached 160MB/s at two NICs and it degraded for higher number of 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 results of the IPv6 over IPv4 tests are shown in Fig. 10. The throughput reached its maximum value at two NICs and it degraded for higher number of NICs, but it remained still higher than the throughput of a single NIC. The CPU utilization graph is not included because it is undistinguishable from the one shown in Fig. 9.

D. Results of the Wget Measurements

The results of the wget tests are shown in Fig. 11. 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 (about 10%) degradation of the throughput can be observed at the end of the graph. The HTTP throughput of the IPv6 tunnel over IPv6 reaches its maximum value at 7 NICs, and it degrades from 70MB/s to 60MB/s for higher number of 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 the version of the encapsulated IP may also have a minor influence on it. The wget measurements show the difference from the iperf measurements that 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 [10]. 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.

1) Results of the iperf measurements: The results of the 64-bit iperf test are shown in Fig. 12. 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 at 8 NICs. It is somewhat higher than the 7 NICs limit of the 32-bit case (see Fig. 4), but it is not a very significant difference. 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. 13. The graphs are rather similar to graphs of the 32-bit case (see Fig. 11), though the throughput results are somewhat better here. The HTTP performance 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. 11), 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.

V. DIRECTIONS OF OUR FUTURE RESEARCH AND A RECOMMENDATION FOR THE FURTHER DEVELOPMENT OF THE MPT LIBRARY

We plan to compare the performance and throughput aggregation capability of the MPT library with that of the standard MPTCP.

As unlike MPTCP, MPT uses UDP, therefore it is also worth testing MPT with real time applications.

We also plan to test MPT as a tunneling tool. MPT seems to be promising one 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.

As for the further development of MPT, we have a recommendation. Enabling MPT to fully utilize the computing power of multiple CPU cores would improve its overall performance when using it for the aggregation of several high speed channels in multi-core environments.
VI. CONCLUSIONS

We have tested the throughput aggregation capability of the MPT multipath communication library up to twelve 100Mbps link layer network connections with all the possible combinations of IPv4 and IPv6 as the underlying or the top protocols. 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 regardless of the version of the encapsulated IP (IPv4 or IPv6). It exceeded 120MB/s for 12 NICs. When the underlying protocol was IPv6, the throughput scaled up linearly up to 7 NICs regardless of the version of the encapsulated IP, but there the throughput reached its performance plateau (with a value higher than 70MB/s) and it showed somewhat degradation for higher number of NICs.

We have shown that the above performance limit depends on the computing power of the CPU and it is not a fixed built in feature of the MPT library.

With the help of 12 Gigabit Ethernet connections, we have also shown that the behavior of the system is similar also in the case when IPv4 is applied as the underlying protocol: the system reached its performance plateau at two NICs (its value was about 160MB/s) and then the throughput showed somewhat degradation for higher number of NICs.

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 and sometimes practically no performance benefit depending on the given test.

We conclude that the MPT multipath communication library is a good tool for the aggregation of the capacity of several channels.

We have given the directions of our future research and also a recommendation for the further development of the MPT library.

REFERENCES


