THINK INDIA JOURNAL

ISSN:0971-1260

Vol-22-Issue-40-December-2019

Conclusive Analysis and Research on MPTCP Environment

Abhishek Upadhyay[1] and Vidya Kubde[2]

[1]Department of Information Technology, Datta Meghe College of Engineering, Airoli, Maharashtra

Abstract

Internet is an amazing network, but with the presence of so many devices and various other issues not everyone is able to experience a good internet connec- tion. MPTCP that is multipath transmission control is an ongoing effort in combining multiple network interfaces/ip-addresses. This is achieved by mod- ifying the currently existing regular TCP/IP suite. The modification spreads the data across various sub flows. One of the most frequent problems that researchers face in MPTCP is setting up of a decent environment for experi- menting and testing. This paper is meant to throw a light on currently existing methods of Installing/enabling MPTCP and setting up Network simulator for the same.

Keywords: MPTCP, Ubuntu, Server, Client, TCP, Installation, NS2, Deployment of MPTCP, BWM, Netdata.

Introduction

MPTCP is a work in progress, under the brilliant efforts of IETF an experimental version for Linux kernel have been released. MPTCP stands for Multipath transmission control prol and in its most basic form, is simply a method to build a connection between two hosts instead of two interfaces (unlike TCP). In standard TCP, the connection is established between two ip addresses and each of these connections are identified by a four-tuple <src ip-addr,src port,dest ip-addr,dest port>. Henceforth, in TCP/IP one can only create only one connection using a single link. MPTCP on the other hand employs a new unique concept of sub flow system which is used to gather standard TCP. Identification of sub flows occurs on the basis of usual three-way handshake, after which the application can add or remove some sub flows.

MPTCP also possess a DSS option which contains a sequence number as such and hence- forth allow receiving the data from multiple sub flows in their original order without undergoing any kind of corruptions. It uses a modified re transmission protocol to handle congestion control and reliability.

MPTCP in its own way opens up a huge possibility of so many new things and so many new implementations, it not only has a huge potential as a security alternative but also is capable on improving the quality of life. It not only has huge potentials in server, datacentres but also as a fantastic resource for a client.

This paper is intended for the users who are interested in MPTCP and also the researchers who want to start out with mptcp research and create decent solutions employing MPTCP to its full potential. The entire paper is divided in three sections, namely:

1. Introduction to MPTCP
2. Relevant Literature Survey
3. Necessary Requirements
4. Procedure for setting up the installation in various environments
5. Various tools to enable the users to utilize and test the MPTCP

Certainly after discussing the installation methodologies we shall discuss about the various congestion control methods in MPTCP and about scheduling.

ISSN:0971-1260

Vol-22-Issue-40-December-2019

Literature Survey

MPTCP design is born of sheer motivation from the crucial need of application congeniability and network compatibility. Application congeniability implies that applications that today run over TCP should work without any change over Multipath TCP. Next, Multipath TCP must op- erate over any Internet path where TCP operates. Almost every path on today's Internet include some if not all, middle boxes, such as Network Address Translators, various kinds of transparent proxies , and firewalls. Unlike IP routers, all these devices are fairly aware about the TCP connections they forward and affect them in special ways. Coming up with a TCP extensions that can traverse through all these middle boxes without undergoing severe alterations have proven to be very difficult and challenging. Multipath transmission control protocol is very much alike mptcp albeit not same. Henceforth it shall be beneficial to recap the basic operations of normal TCP. A normal tcp connection comprises of three major stages.

- connection establishment
- data transfer
- connection release

A TCP connection is initiated in a three-way handshake fashion. To start a TCP connec- tion,the client sends a SYN (for synchronize) packet to the port on which the server is listening, this involves using an ephemeral port on the client device thereby can be summarized as an attempt to deliberately establish a one to one port connection to Server. The SYN packet con- tains the source port and initial sequence number chosen by the initiator (the client), and it may contain TCP options that are used to negotiate the use of TCP extensions. The server replies with a SYN+ACK (which unremarkably stands for "synchronization acknowledged") packet, acknowledging the SYN and providing the servers initial sequence number and the options that it supports, Upon not receiving a SYN+ACK the client simply retries or quits on its attempt to make connection.

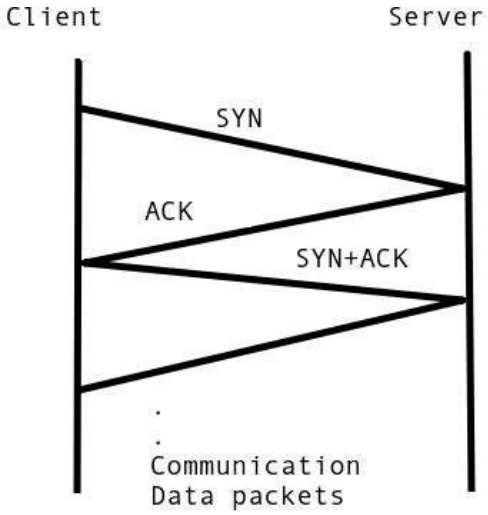


Figure 1: TCP in Action

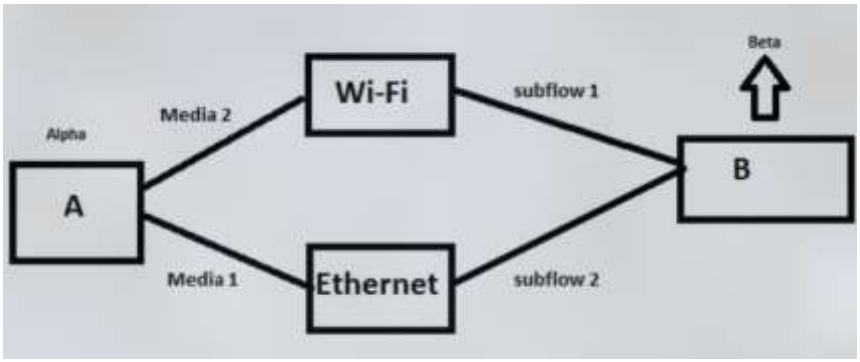
The client acknowledges the SYN+ACK (which can be called or written as SYN+ACK+ACK, "Synchronization Acknowledgement Acknowledged") 1, and the connection is now fully estab- lished. All subsequent packets in the connection use the IP addresses and ports used for the initial handshake, technically so is achieved by composing the tuple that distinctively identifies the connection.

After the handshake is established, the client and the server can send segments (Data pack- ets). The sequence number associated with the segments is used to identify the data in the different segments, reorder them, and detect losses if any. The TCP header, contains a cumu- lative acknowledgment, essentially a

ISSN:0971-1260

Vol-22-Issue-40-December-2019

number that acknowledges received data by telling the sender which is the next byte expected by the receiver. Various techniques are used by TCP to retransmit the lost segments, the loss of segments can happen because of many reasons like congestion control or link failures to name a few. After the data transfer is over, the TCP con- nection must be closed. A TCP connection can be closed abruptly if one of the hosts sends a Reset (RST) packet, but the usual way for termination of a connection is by using FIN packets. These FIN packets mention the sequence number of the last byte sent and then The connection is terminated after the FIN segments have been acknowledged by both client and server.



Multipath Transmission control protocol, MPTCP, allows multiple ”subflows” to be set up for a single MPTCP session. An MPTCP session starts with an initial subflow, which is simi- lar to a regular TCP connection as described above.Once MPTCP subflow is set up, additional subflows can be established. Each additional subflow also looks similar to a regular TCP con- nection 2, complete with SYN handshake and FIN tear-down, but rather than being a separate connection, the subflow is bound into an existing MPTCP session. Data for the connection can then be sent over any of the active subflows that has the capacity to take it.

Currently, MPTCP has been implemented in Linux kernel as such. It is available for Linux kernel, FreeBSD, F5 Network, citrix Netscaler, Apple ios7 Apple MacOS X 10.10, proxy ver- sion 0.9 by Alcatel-lucent and a lot many systems. Individually, one can possibly compile the mptcp for one’s system. One of the major problems that researchers face upon trying to work on MPTCP is the setting up of a decent environment. No doubt that there are enough resources on the internet which can help one out in setting up of an environment, but they all lack one thing, that is the error resolving techniques. Some of the posts are not even up-to-date to the latest details and updates. The purpose of this research paper is simply to set up a basic environment for testing and experimenting with MPTCP. It can also be used by novice users to use MPTCP in their day to day life, though it is not recommended. It must be noted that MPTCP is still in its early experimental phase. Fortunately the people are already working on it and making it better. Also, we will see how to set up the simulation environment of mptcp in Ubuntu ma- chines. The choice of Ubuntu machine is mostly because of its easier usage and easier setup. For other types of available machines, the process almost remains the same (if its available in the software repository, or else one will have to compile from the source).

Requirements

1. Ubuntu Machines, should work with version 12.04+, but the newer it is the better it will be. (Minimum one, if only client is meant to be setup and existing MPTCP systems are to be used as servers, otherwise two) Particularly MPTCP has been implemented for various systems ranging from Ubuntu, Debian based machines to Raspberry pi and using vagrant on hosts without Linux kernel support.

But personally the procedure for setting it up on Ubuntu machines is much easier than any other device or systems. Ofcourse, this is a subjective viewpoint but it must be noted that for raspberry pi system or the

ISSN:0971-1260

Vol-22-Issue-40-December-2019

system for which mptcp doesnt exists in standard repository, the procedure becomes hectic and involves compilation.

2. Two or more internet mediums [if not going for simulations]. Most of the computers already come equipped with an Ethernet port which can be directly employed. The prob- lem occurs with obtaining the second internet media. The two existing options for this purpose are:

a). Wi-Fi adapter: An USB Wi-Fi adapter can be employed and setup to achieve the purpose of alternate internet media.

b). Bluetooth/USB tethering using android phone, this is a limited option as it does not allow modification of gateways and ip tables easily, but are still usable.

3. Stable internet connections on available mediums. Using a laptop with inbuilt Wi-Fi adapter and Ethernet port prove best for the same.

Procedure to install MPTCP on a machine

1. The gpg apt-key must be added for securely accessing and downloading the MPTCP enabled package.

sudo apt-key adv –keyserver hkp://keys.gnupg.net –recv-keys 379CE192D401AB61

2. Now we need to add the repository to apt sources file, one can use any existing editor to add the url deb ”https://dl.bintray.com/cpaasch/deb stretch main” to /etc/apt/sources.list.d/MPTCP.list or just use the following command to directly append the repository to apt source file.

sudo sh -c ”echo ’deb https://dl.bintray.com/cpaasch/deb stretch main’
> /etc/apt/sources.list.d/MPTCP.list”

Now the MPTCP can be installed by running

sudo apt-get update && apt-get install linux-mptcp

```
root@akuma:~# sudo apt-key adv --keyserver hkp://keys.gnupg.net --recv-keys 379CE192D401AB61
Executing: /tmp/apt-key-gpghome.sDU12WzTVR/gpg.1.sh --keyserver hkp://keys.gnupg.net --recv-keys 379CE192D401AB61
gpg: key 379CE192D401AB61: "Bintray (by JFrog) <bintray@bintray.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
root@akuma:~#  sudo sh -c "echo 'deb https://dl.bintray.com/cpaasch/deb stretch main' > /etc/apt/sources.list.d/mptcp.list"
root@akuma:~#  sudo apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Ign:5 https://dl.bintray.com/cpaasch/deb stretch InRelease
Get:6 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [557 kB]
Get:7 https://dl.bintray.com/cpaasch/deb stretch Release [1,842 B]
Get:8 https://dl.bintray.com/cpaasch/deb stretch Release.gpg [821 B]
Get:9 https://dl.bintray.com/cpaasch/deb stretch/main amd64 Packages [4,653 B]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [207 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [6,996 B]
Get:12 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [744 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/universe Translation-en [194 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [6,384 B]
Get:15 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [3,468 B]
Get:16 http://archive.ubuntu.com/ubuntu bionic-security/main amd64 Packages [282 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-security/main Translation-en [103 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [4,296 B]
Get:19 http://archive.ubuntu.com/ubuntu bionic-security/restricted Translation-en [2,192 B]
Get:20 http://archive.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [127 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic-security/universe Translation-en [71.8 kB]
Get:22 http://archive.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [3,748 B]
Get:23 http://archive.ubuntu.com/ubuntu bionic-security/multiverse Translation-en [1,952 B]
Fetched 2,574 kB in 35s (73.8 kB/s)
Reading package lists... Done
root@akuma:~#   sudo apt-get install linux-mptcp
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  linux-headers-4.14.105.mptcp linux-image-4.14.105.mptcp
The following NEW packages will be installed:
  linux-headers-4.14.105.mptcp linux-image-4.14.105.mptcp linux-mptcp
0 upgraded, 3 newly installed, 0 to remove and 237 not upgraded.
Need to get 52.3 MB of archives.
After this operation, 311 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://dl.bintray.com/cpaasch/deb stretch/main amd64 linux-headers-4.14.105.mptcp amd64 20190318151730 [11.2 MB]
```

Figure 3: MPTCP installation

ISSN:0971-1260

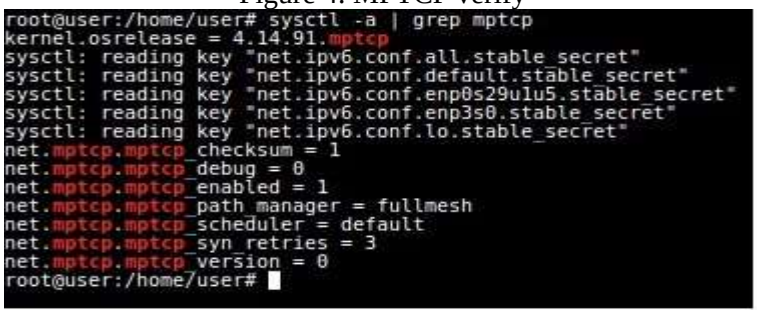
Vol-22-Issue-40-December-2019

Once installed (check figure 3), the machine must be rebooted, during reboot press shift before the bootlogo appears and you should be dropped in the kernel selection menu. Select the MPTCP kernel and then it's time to configure the routing.

Also, installation can be verified by executing(check figure 4):

dmesg — grep MPTCP

Figure 4: MPTCP verify



after that reboot the machine.

Configuring routing in MPTCP enabled machine. Since there are more than one network interfaces, its crucial to define the network gateways corresponding to correct network interface, so can be achieved by configuring the routing table. Let's assume that one must have two internet mediums: eth0, eth1

eth0: IP - 192.168.1.100

Subnet - 255.255.255.0

Gateway - 192.168.1.1

eth1: IP - 192.168.2.102

Subnet - 255.255.255.0

Gateway - 192.168.2.1

then configurations can be performed by

Create two different routing table corresponding to each interface

ip rule add from 192.168.1.100 table 1

ip rule add from 192.168.2.102 table 2

configure the two different tables

ip route add 192.168.1.0/24 dev eth1 scope link table 1

ip route add default via 192.168.1.100 dev eth1 table 1

ip route add 192.168.2.0/24 dev eth0 scope link table 2

ip route add default via 192.168.2.102 dev eth0 table 2

default route for selection of normal internet traffic

ip route add default scope global nexthop via 192.168.1.100 dev eth1

In case if this method doesn't work well or throws up incomprehensible errors, one can directly use the pre-built scripts.

curl "https://raw.githubusercontent.com/multipath-tcp/MPTCP-scripts/master/scripts/rt table/

ISSN:0971-1260

Vol-22-Issue-40-December-2019

MPTCP up" > /etc/network/if-up.d/MPTCP up && chmod 777 /etc/network/if- up.d/MPTCP up && curl "https://raw.githubusercontent.com/multipath-tcp/MPTCP- scripts/master/scripts/rt table/

MPTCP down" > /etc/network/if-post-down.d/MPTCP down && chmod 777 /etc/network/if- post-down.d/

MPTCP down

Now so as to test whether one is using mptcp or not one can use following set of commands to check it out.

curl http://www.multipath-tcp.org

Yay, you are MPTCP-capable! You can now rest in peace.

if mptcp is not used it prints "Nay, Nay, Nay, your have an old computer that does not speak MPTCP. Shame on you!"

MPTCP and Schedulers

Each and every one of wireless communication connections, such as balloon-based aerial wire- less networks,personal area networks and hotspots,HSPA+ access in high speed trains and satel-

lite based Internet connections have not only slow speed but also very significant rate of loss. MPTCP stands upto the test even in such harsh environments. However, MPTCP doesnt achieve the so results on its own. It has its own unique feature of a scheduler. Imagine being in a scenario where in the task can be handled by many interfaces, in such a case a program or a piece of code that enables the selection of a better interface and allows enhanced MPTCP performance and prevents TCP fall back is called as scheduler. These schedulers are available in large numbers in various formats, these basically are the scheduling algorithms (such as FCFS,RRT) which have been adapted for the noble purpose of MPTCP. A common man doesnt knows which algorithm must be used in what scenario, so our future research basically is to find the optimal algorithms for various Scenarios and situation.

We have also began experimenting with various schedulers to find that one is better suited than other in a scenario. For example, during a simple file transfer from MPTCP enabled server the snapshots were downloaded. This was done in order to verify (as well as justify) the following things:

1. Impact of FCFS scheduling
2. Reliability of MPTCP
3. Impact of RR Scheduling.

The default scheduler is basically designed with the purpose of fallback, henceforth in most of the cases the default falls back to TCP protocol stack right away in any minor inconvenience. Please note that download starts at 12:12

Roundrobin fashion of scheduling is one of the most common ways, it selects one subflow after the other in round-robin fashion. Such an approach albeit guarantees that the capacity of each path is fully utilized as the distribution across all subflows is equal but in extreme cases

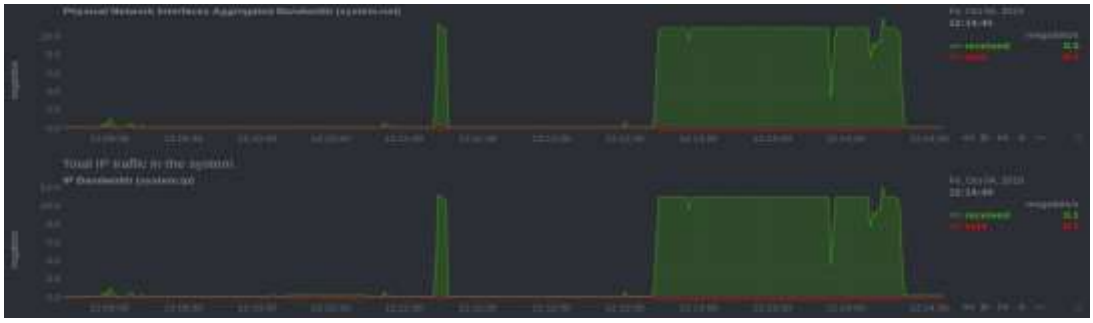


Figure 5: Downloading via default scheduler

ISSN:0971-1260

Vol-22-Issue-40-December-2019

Figure 6: Downloading in Identical scenario using RR

where the bulk data needs to be transmitted, the scheduling does not really happen in round- obin manner. Henceforth it can be postulated that round robin is fairly good for short burst data transmission.

Conclusion

MPTCP is a revolutionary technique to enhance our current internet experience. The speed difference which was obtained even on very weak network connections was great. There are various scheduler algorithms which can be modified and used as well. There is still a lot that is needed to be done when it comes down to MPTCP

For example, we need better handover algorithms, better scheduling algorithms more devel- opers and programmers to use the concepts and also to enable MPTCP in servers for extended performance.

As a last bit of experimentation, following diagram includes a download graph. In the screenshot we basically have two interfaces,

1. Ethernet
2. Mobile tethering (usb cable)

During the download, the download was many times interrupted by plugging and unplug- ging ethernet cable, stopping the tethering. Thanks to MPTCP the download was completed within a good amount of time and without errors refer to: Figure 7.

References and Notes

1. Multipath TCP - C. Paasch and O. Bonaventure Communications of the ACM, vol. 4, no. 51-57, 4 2014.

ISSN:0971-1260

Vol-22-Issue-40-December-2019

2. C. Paasch and S. Barre, Multipath TCP in the Linux Kernel, available from. [Online]. Available: http://www.multipath-tcp.org
3. M. H. O. Bonaventure and C. R. U. login, An overview of Multipath TCP, 2012.
4. Multipath-tcp patch residence. [Online]. Available: https://code.google. com/archive/p/multipath-tcp/downloads
5. X. Chen, R. Jin, K. Suh, B. Wang, and W. Wei. Network Performance of Smart Mobile Handhelds in a University Campus WiFi Network. In ACM IMC, 2012.
6. G. and, patch for installing NS2. [Online]. Available: https://github.com/ wangzhizhou/Network-Simulator-Installation-Guideline-For-Ubuntu Online . Available: tcp.org/pmwiki.php/Main/HomePage
7. H. H. C. I. Be, Designing and Implementing a Deployable Multipath TCP. San: Jose, 2012.
8. FreeBSD Project. tcp internet transmission control protocol. FreeBSD Kernel Interfaces Manual.
9. M. Handley. Why the internet only just works. BT Technology Journal, 24:119129, 2006.
10. P. Key, L. Massoulie, and D. Towsley. Path selection and multipath congestion control. In Proc. IEEE Infocom, May 2007.
11. J. R. Iyengar, P. D. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. IEEE/ACM Trans. Netw., 14(5):951964, 2006.
12. A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP extensions for multipath operation with multiple addresses, Jan 2012. IETF draft (work in progress).