Transport layer multipath transmission:

* Study MPTCP
* Set up an emulation network and use MPTCP protocol to transmit a file from source to destination via different paths.
* Observe and test several features of MPTCP such as load balancing,.
* Pros and cons of MPTCP
* Study MPTCP
* **Introduction** (can also reference pages 4-25 from :<http://multipath-tcp.org/data/MultipathTCP-netsys.pdf> -> see youtube video for explanation of slides)

As bandwidth demand increases, the traffic throughput of the worldwide network each month also rises. Organizations are increasing their fiber infrastructure dramatically. Additionally, the wireless world is expanding and increasing its infrastructure. To cope with all those bandwidth demands and increasing infrastructure the Internet is in need of new future technologies. MultiPath TCP [(MPTCP)](#bookmark115) is one promising technology, which aims to enable devices to use multiple network paths at the same time. However, this is not always possible because not all end-nodes support multiple network interfaces. That said, many environments will usually support multiple paths in the case of failover or for robustness. For instance, a datacenter will often utilize the multiple paths for load balancing; however, load balancing different flows does not guarantee the full utilization of the maximum bandwidth possible. MPTCP however should be able to utilize all possible route and increase the maximum bandwidth throughput and robustness.

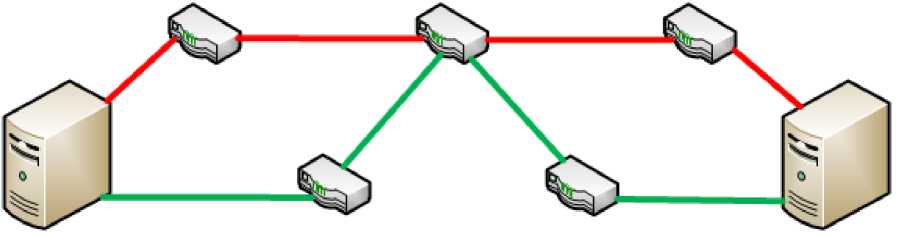


Figure 1: Topology where Multipath TCP is being used

One other possible application of the MPTCP lies within the mobile world. Mobile devices now a day like smart phone, tablet, and so on typically have multiple interfaces for Wifi and 2G, 3G, 4G-LTE and so on. For TCP only one of those connection is possible to be used. With MPTCP, it is possible to make use of both the Wifi and cellcular data network at the same time, making it easier to switch between the two network reducing the delay, interruption when switching network.

Another promising example is the mobile world. Phones hove multiple interfaces for example 3G and Wifi. With regular [TCP](#bookmark121) it is only possible to use one of those for a connection. The concept of multi-homing in combination with [MPTCP](#bookmark115) makes it possible to use both the 3G and Wifi link at the same time. This has of course its pros and cons but at least [MPTCP](#bookmark115) can make this possible. One of the pros is for example the easy migration of your connection from 3G to Wifi, your phone can use the same connection and does not have to establish a new one. In figure [2](#bookmark8) is shown how this looks like from a mobile phone point of view.

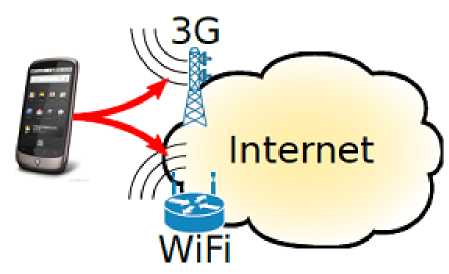
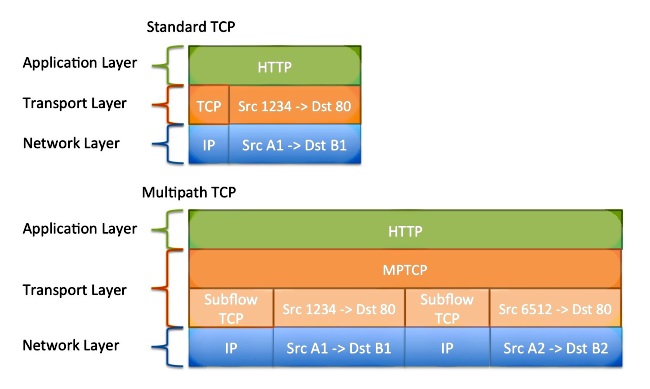


Figure 2: Mobile phone with Multi-Homing [[5]](#bookmark144)

* **Definition of mptcp & goal**

MultiPath TCP (MPTCP) is an effort towards enabling the simultaneous use of several IP-addresses/interfaces by a modification of TCP that presents a regular TCP interface to applications, while in fact spreading data across several subflows.

MPTCP is a new protocol being developed by Internet Engineering Task Force's (IETF) aims at allowing the use of multiple IP-addresses/interfaces at the same time. By a modification of TCP, MPTCP will present a normal TCP connection the application, but in fact the data will be spread across several subflows.



<https://www.cisco.com/c/en/us/support/docs/ip/transmission-control-protocol-tcp/1>16519-technote-mptcp-00.html

* + 1. How does MPTCP work?

MPTCP builds on the TCP protocol, thereby extending the capabilities of the underlying transport layer. The MPTCP functionality option can be disabled or enabled; to be enabled, the client and server have the option enabled. No additional Application Programming Interfaces (API) need to be installed to use MTCP since it only uses options already present in TCP. The regular TCP socket API will be extended and [MPTCP](#bookmark115) aware applications can make use of those extensions. In addition to easy installation, another advantage of MTCP is that middle-boxes, firewalls won’t intercept it, as it is treated as TCP.

In the scenario that the server lacks MPTCP support, the protocol will utilize the default regular TCP. In the case where there are multiple connections available, MPTCP will create TCP subflows; each subflow will be treated as a connection from one interface to another. MPTCP, by default, will attempt to create subflows between all the interfaces that can be detected to create a full-mesh. For example, if one device has two IPs on the same interface, the device will attempt to connect to both of them and each will be treated as a subflow. Figure 4 provides a high level overview that illustrates how MPTCP functions.

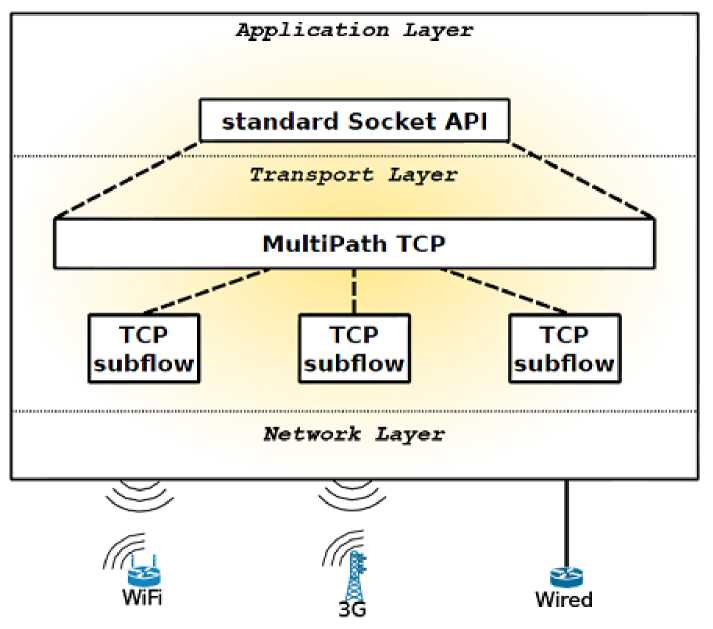


Figure 4: Architectural diagram of MPTCP [[5]](#bookmark144)

* + 1. MPTCP design Goals

During the development process, some important protocol goals were established. We will talk about the three most important goals in detail; they are taken from RFC 6356:

1. Improve Throughput: A multipath flow should perform at least as well as a single path flow would on the best of the paths available to it.
2. Do no harm: multipath flow should not take up more capacity from any of the resources shared by its different paths than if it were a single flow using only one of these paths. This guarantees it will not unduly harm other flows.
3. Balance congestion: A multipath flow should move as much traffic as possible off its most congested paths, subject to meeting the first two goals.

The first rule ensures that the best performance is always achieved or at least equal to the best performance TCP would get on the fastest route. The second rule handles the congestion and fairness part. If one has a TCP and a MPTCP connection over the same link they would share the bandwidth equally. The last one states that the less congested link should always be chosen.

* + 1. MPTCP Handshake

In this section, we will describe how the MPTCP handshake works. At first the server A sends a SYN packet to server b, which is similar to a normal TCP connection. But this packet also contains something different from the standard TCP packet, a MP\_CAPABLE option in the TCP option field. After that the server B responds with a SYN/ACK packet also including this MP\_CAPABLE option if it supports MPTCP. Both packet contains necessary information for MPTCP connection like authentication keys and additional information. Lastly, server A send a response with a ACK to server B which includes both keys from the previous two packets. Then the first connection (subflow) is set up and similar to TCP is a three-way handshake.

This section will cover how the [MPTCP](#bookmark115) handshake works and looks like, the handshake is shown in figure [5](#bookmark35) and is written in a Internet draft [[13]](#bookmark152). First ‘Server A’ sends a SYN packet to ‘Server B’, which is the same as a general [TCP](#bookmark121) connection would do. This packet however also contains a MP\_CAPABLE option in the [TCP](#bookmark121) option field. Than ‘Server B’ responds with a SYN/ACK packet also with this MP\_CAPABLE option (only when it can use [MPTCP)](#bookmark115). Both packets also contain authentication keys and additional information needed by [MPTCP](#bookmark115). At last ‘Server A’ responds with a ACK to ‘Server B’ which also contains both keys from the previous packets. Now the first connection (subflow) is setup and the same as [TCP](#bookmark121) is a three way handshake.

Afterward the initial setup is done and the protocol tries to create extra subflows. Since both servers have the same implementation, they can both create additional subflows, although it is recommended that the senders is the only one doing this. A subflow creation is also a three-way handshake (SYN, SYN/ACK, ACK) but now an additional MP\_JOIN option is also included. Server A also sends with the initial SYN a server B’s token, which is created with the keys exchanged earlier. This is used for the identification and authentication of the first subflow connection. A nonce is sent to prevent replay attacks. An address ID is also sent to specify the address and differentiate the subflow from the others. Because of this address ID, the connection is not dependent on the source address.

Now the initial setup is done and the protocol tries to create additional subflows. Since the im­plementation is in both servers the same, they can both create additional subflows, however it is recommended the senders only do this. The creation of a subflow is also a SYN, SYN/ACK and ACK handshake but now an additional MP\_JOIN option is added. ‘Server A’ also sends with the first SYN ‘Server B’s token’, which is generated with the keys exchanged earlier. This is used to identify and authenticate the first subflow connection. Also a nonce is sent to prevent replay attacks. The address ID sent is to specify the address and make the subflow unique from the others. Because of this address ID the connection is independent from the source address. The flags can be used for more advanced options (for example use it only as back-up link).

As the subflow MP\_JOIN SYN is verified by server B it sends a response with an SYN/ACK. This ACK includes a message authentication code (MAC), a random nonce and server B’s ID address. The MAC functionality is for the authentication of a subflow. Then server A replies with a ACK packet and the MP\_JOIN option also with the additional data as with server B. After that server B verify this packet with another ACK, because server A has to make sure that the server B received the packet. SHA-1 is being used for all authentication technique.

When the subflow MP\_JOIN SYN is verified by ‘Server B’ it responds with an SYN/ACK. This ACK contains a Message Authentication Code (MAC), a random nonce and the address ID for ‘Server B’. The MAC is used for the authentication of a subflow. After this ‘Server A’ responds with a ACK packet and the MP\_JOIN option also with the additional data as with ‘Server B’. After this ‘Server B’ needs to verify this packet with another ACK, since ‘Server A’ has to be sure ‘Server B’ received the packet. For all the authentication technique’s SHA-1 is being used.

* + 1. MPTCP Path Management

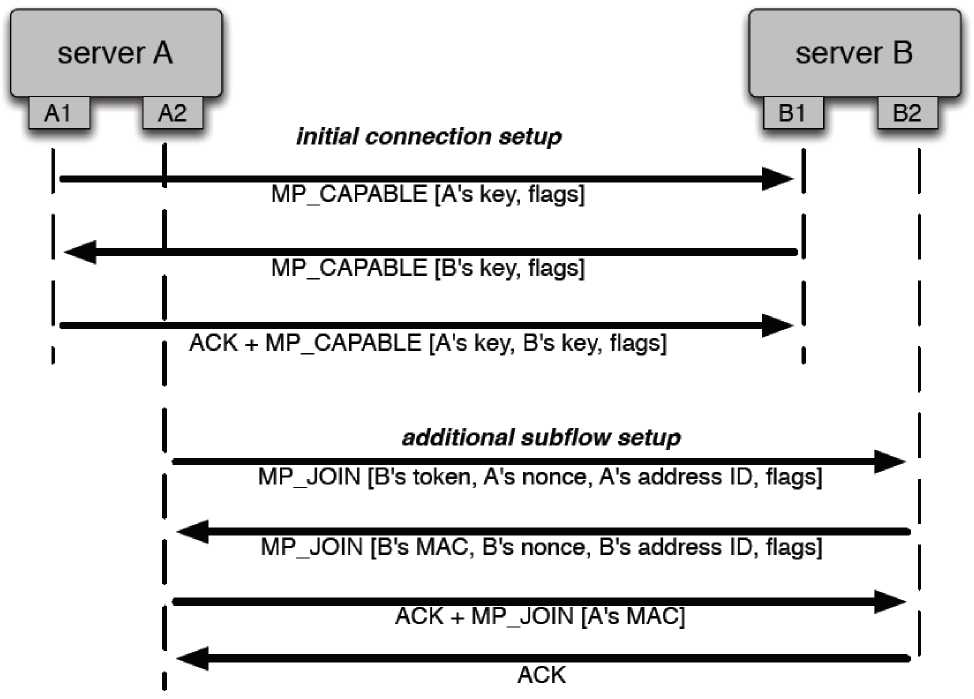


Figure 5: [MPTCP](#bookmark115) handshake [[14]](#bookmark153)

After the handshake is performed, the initial and first subflow is created. Then MPTCP must ensure that both hosts have exchanged all the IP addresses they currently possess. The sending host uses this information and creates a full-mesh from all the possible combinations. Connections that are not responsive will be dropped when the timer times out. For interfaces with both a IPv4 and IPv6 address, MPTCP will create two separate subflows, one for each address.

[MPTCP](#bookmark115) exchanges IP address with packets who have the subtype add-address. After this a join subtype is send to create the actual new subflow with those new known addresses. Another nice feature [MPTCP](#bookmark115) has is that it is capable of removing subflows with the subtype remove-address. When a host discovers one of its links is broken it can very fast send over a working link that that particular link does not work anymore. This way the other end does not have to wait for time-outs on the broken link and can just drop that subflow. Most of the subtype options are shown in section

1. in detail.
2. MPTCP Congestion Algorithm

The congestion algorithm is a necessary component of all reliable transport protocols. As a result of the presence of multiple links, the existing algorithm used for TCP needs to be extensively modified to deal with congestion control over multiple links as seen in Figure 6.

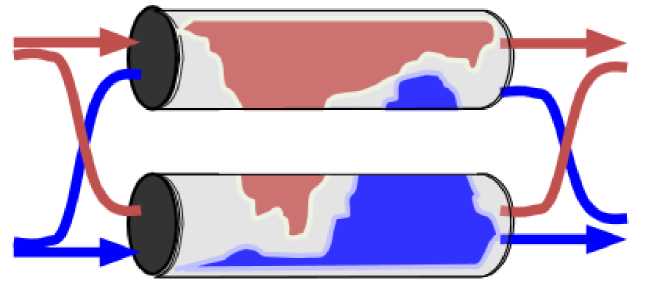


Figure 6: How [MPTCP](#bookmark115) should handle congestion [[10]](#bookmark149)

The congestion algorithm for [MPTCP](#bookmark115) needs to satisfy the MPTCP design goals mentioned earlier. Chapter 3 of the RFC 6356 document describes the details of the algorithm and how it meets all the design goals. Each subflow is a window; the congestion mechanism dictates that the congestion window be increased with each ACK (1/cwnd), but decreased by a half when a ACK is dropped. Compared to the existing congestion control mechanism, the only thing that is changed is the window increase rule, as illustrated in equation 1.

The increase rule depends on alpha; the window increase amount is proportional to alpha, more specifically the sum of the window sizes of all paths. This means that subflows of a larger window size will experience a greater increase, resulting in less congestion. The sum of all window parts must satisfy the second MPTCP design goal.

alpha 1

cwndi = cwndi *+* min *,*

cwndtot cwndi

C:\Users\Hieu\Desktop\media\image9.pngIn equation 2, alpha is the increase parameter. The first term takes into account the congestion window size and the round trip time on all different paths; it compares the results with the normal TCP, thereby satisfying MPTCP design goal one. The second part directs traffic from a congested link to an uncongested one, thereby satisfying MPTCP design goal three. if it is getting better or worse than normal [TCP](#bookmark121), which meets goal one. The second part of alpha moves traffic from a congested link to a uncongested link and meets goal three.

*alpha*

*cwndtot*

*/*cwndi\*mss*2 \*

maxi ( RTT2 -)

(i

cwndj\*mssj 2

RTTi

Equation 3 is the decrease formula for MPTCP; it is very similar to TCP’s formula but it is used for a subflow instead.

C:\Users\Hieu\Desktop\media\image10.pngcwndi

2

The algorithm ensures that MPTCP chooses the best available path. Figure 7 illustrates the choosing of the shortest and least congested paths. Poor path choices like taking longer routes or using congested links will result in inefficiency. By picking the most optimal paths, MPTCP will achieve the best throughput.

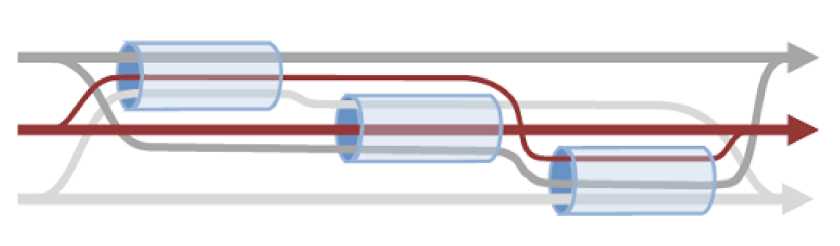


Figure 7: [MPTCP](#bookmark115) path decision taking [[11]](#bookmark150)

1. MPTCP and buffer sizes

MPTCP’s buffer size differs slightly from TCP’s buffer size. Equation 4 shows how TCP’s buffer size is calculated; it depends on the maximum round trip time and maximum the link can carry in bits.

BuffersizeTCP = RTTmax \* LinkMaxblts *(4)*

In contrast, MPTCP’s buffer size equation is slightly different. Instead of specifying just the maximum one link can carry, we must specify the total maximum bits that the combination of all links can carry. It is noted in the RFC 6356 document that : "If we want to allow all paths to keep sending while any path is fast retransmitting, the buffer must be doubled". Equation 5 specifies how MPTCP’s buffer size is calculated. It should be noted that MPTCP requires significantly more buffer space than TCP. For example if the round trip time is 36ms and there are two 1Gb/S links, applying formula 5 would give: 0.036 \* 2000000000 \* 2 = 1440000006ft = 18MB.

BuffersizeMPTCP = RTTmax \* AllLinksMaxhlts \* *2 (5)*

* Pros and cons of MPTCP

http://www.enterprisenetworkingplanet.com/netsp/the-dangers-and-promise-of-multipath-tcp.html