Multipath TCP Protocols

Dinamene Barreira, Fernando Mira da Silva
Técnico Lisboa
Av. Prof. Doutor Aníbal Cavaco Silva, Taguspark, Oeiras, Portugal
Email: {dinamene.barreira, fernando.silva}@tecnico.ulisboa.pt

***Abstract*—The paradigm of data networks is ever changing, with the increasing number of devices, mobility, access diversity and ubiquitous applications. Nowadays, Transmission Control Protocol (TCP) is the most popular protocol to transmit and deliver information reliably over the Internet. However, conventional TCP makes use of a single path connection, not taking advantage of multihoming and multiple paths that are increasingly available to end point devices, namely mobile devices and servers in high resilient configurations. MultiPath TCP (MPTCP) has been developed to address these TCP limitations. The MPTCP protocol aims to make use of path diversity, in order to offer a better overall network connectivity, increasing resilience to failures, performing load balance between available paths, when more than one is available, and to allow multihoming support without the need to modify the already existing devices currently scattered over the network. The objective of this project is to create a testbed that can assess the benefits and limitations of MPTCP protocol, namely in mobile application scenarios.**

Keywords*—*MultiPath TCP, Resilience, Performance, Testbed, Throughput, Multihoming

I. Introduction

The evolution of portable devices, such as mobile phones, tablets and laptops, made it important to always be reachable and have a high throughput connection, since most of the critical applications that run on these devices spread their computation across cloud systems, resorting to data centers spread around the world.

At the same time, many devices developed the capability of connecting to the Internet with at least two different interfaces in each type of device, such as WiFi and 3G, or Ethernet and WiFi, in order to optimize the available communication infrastructures. On the other hand, the data centers spread across the world usually support multihoming, being connected to two or more networks in order to improve resilience for the services provided.

For many years the Transmission Control Protocol (TCP) [1] has been a fundamental component of the Internet protocol stack and the most reliable communication protocol for data transmission, but it only allows a single path between a source and a destination. Although the basic model of TCP understands the essential mechanisms required to control flow and congestion, by itself it does not assure real-time delivery in cases of critical congestion connections or breaks on a support link. With the increasing mobility of devices, with ubiquitous and critical applications, it has become very important to have reliable connections.

The Internet is coming to a point where the high increase on the number of users, providers and services are beginning to stress its scalability, so it is important to adapt the existing protocols to make them able to explore the benefits that may arise from existing multipath connectivity.

With the objective to work around the limitations of conventional TCP protocol and increase the reliability of connections, several authors have proposed different approaches, like Stream Control Transmission Protocol (SCTP) [2], Concurrent Multi-Path Stream Control Transmission Protocol (CMP-SCTP) [3], multiple paths TCP (mTCP) [4], Parallel TCP (pTCP) [5], Protocol Shim6 (Shim6) [6] and others. All these protocols have limitations that make difficult to deploy them on the Internet, as it will be discussed later on.

Recently the Internet Engineering Task Force (IETF) has created a work group to develop a standard of a multipath protocol at the transport layer that can be easily deployable. Having this in consideration an extension for the TCP protocol has been proposed, the MultiPath TCP (MPTCP) [7], where each connection between two points can actually use multiple parallel routes, using congestion detection techniques to determine the actual path to be followed.

As we will see, MPTCP [8] has the potential to increase throughput, reliability and flexibility in connections. The fact that it is an extension of TCP and has backward compatibility makes it easier to be deployed on the Internet than other previous proposals.

The main goal of this project is to study and evaluate the new emerging MPTCP protocol, with the objective of creating a functional MPTCP testbed.

The motivation for implementing a multipath TCP protocol is to improve robustness and performance of end-to-end connections. This solution allows the use of multiple paths by the same TCP connection to maximize resource usage, increase redundancy and resilience.

This protocol offers multihoming capability, resorting to the use of different available network interfaces on current devices, providing a better throughput. For mobile devices this protocol can allow smooth connection handover, without loosing application connectivity. The use of multipath connections also offers benefits to data center operations, since it can contribute to improved throughput, larger path diversity and better fairness.

II. State of the Art

Over time there has been some debate about what approach should be taken in order to provide a protocol that can benefit from the simultaneous use of several paths so as to exploit the available resources in the network, also being interesting if it could be made possible the efficient use of multiple interfaces to ensure constant connectivity on mobile devices.

Researchers argue different implementation solutions. In the following subsections, we analyze the different layers in which an implementation could take place.

A. *Link Layer*

A solution to provide multipath at link layer is to use Link Aggregation Control Protocol (LACP). This protocol uses a link layer bundling approach. Link bundling occurs by aggregation of switch ports in order to use multiple network connections in parallel to increase throughput and create redundancy in case of link failure.

A solution that takes advantage of a multipath scheme at the link layer is the Shortest Path Bridging (SPB) [9]. The IEEE 802.1aq based on IEEE 802.1Q Ethernet Bridging, allows the use of multiple equal cost paths in mesh Ethernet network environments. This solution supports a much larger solution of layer two topologies which can be used at the data center nodes, but it can not make use of multiple interfaces available.

Another multipath approach implementation for the link layer has been suggested to achieve higher throughput at Wireless Mesh Networks (WMN) [10]. This solution needs to implement a multi-channel link layer in combination with multi-path routing in order to efficiently and intelligently route the traffic to achieve a better throughput in these networks.

The link layer is responsible for the channel and packet scheduling, the first is used to control which channel the information will be received and the second when to send the packets. The multipath-routing scheme is responsible for selecting the best two paths to the gateway.

A solution at this level has great potential to achieve a good performance and higher end-to-end throughput for this type of networks, offering these benefits by resorting to decomposing the traffic across different channels, scheduling different times and finally using different routes.

Layer 2 multipath solutions are mostly restricted at the local area networks, and do not cope to possible multipath diversity that is often available at the network layer.

B. *Network Layer*

Implementing a multipath TCP protocol at the network layer seems natural. In this case it would have a single connection at the transport layer and the packets would be scattered across different flows. The load balancing is performed at the connection level and not at packet level, in conjunction with the fact that this solution only offers a single connection at the transport layer the throughput will be dictated by the most congested link or the slowest, because the congestion control of the different paths are aggregated by the transport protocol.

This solution, however, may mitigate unnecessary retransmissions at the transport layer due to packet re-ordering, causing a significant reduction of the connection's throughput. One of the causes of this unnecessary retransmission, occurs because most of the network devices scattered across the Internet do not support and do not recognized the traffic generated. In order to implement a multipath solution at the network layer it is necessary to resort to an upgrade of the network devices currently in use, scattered across the Internet.

C. *Transport Layer*

The implementation of a multipath protocol at the transport layer has the possibility of gathering information like capacity, latency and congestion state at each path used. With this information it is possible to react to congestion in the network and move the traffic to avoid the congested paths.

A multipath implementation at this layer allows it to be transparent for both upper and lower layers, which means that it will use multiple flows that look just like regular TCP connections and the traffic moves without been retained at middleboxes that exist on the network.

The implementation of multipath protocol at the transport layer, it can offer functions of path management, packet scheduling, congestion control and even subflow interface without needing to modify the upper and lower layers. In this sense, in order to support a multipath link, it is only required that the endpoints support the protocol and it is not necessary to update any existing router or layer three component between the endpoints.

D. *Application Layer*

An example of application layer solutions are Peer-to-Peer (P2P) protocols, as BitTorrent[11], the multipath approaches that works at chunk granularity and has the objective of increasing throughput. They achieve their objective by downloading the different chunks of a file through different peers, choosing to download from the fastest servers available.

BitTorrent achieves job-level resource pooling in many-to-one transfers. It behaves like uncoupled multipath congestion control, by running independent congestion control on each path, with paths having different end-points.

It is possible to develop a multipath application that can provide multihoming for the available devices and servers using P2P protocols. The problem with this type of solution is that it can bring limitations to other users of the network, being so that an implementation at this level will offer an unfair competition for the resources available.

In fact it makes use of path diversity given that there are multiple servers available, but does not cope with end to end multipath.

III. MULTIPATH TCP

This chapter describes some of the most relevant properties and services of the current MPTCP draft, as decribed in RFC 6824 [7].

A. *Goals*

In this section is described the primary goals defined for MPTCP and the rules that it must abide to.

The goals that MultiPath TCP aims to meet at a **functional level** is to improve throughput and resilience. In order to improve throughput a MultiPath TCP connection over multiple paths should not achieve a throughput worse than a single TCP connection over the best path in that group of paths. To improve resilience MultiPath TCP paths must be interchangeable and must never be less resilient than a regular single-path TCP.

The goals that MultiPath TCP aims to meet at **application compatibility** is to follow the same service model as TCP, offer backward compatibility and support some kind of TCP's session continuity. In order for MultiPath TCP to follow TCP's service model it needs to ensure that the delivery is in-oder, reliable and byte-oriented. However MultiPath TCP might not be able to provide the same level of consistency of a single TCP connection throughput and latency. MultiPath TCP has to provide backward compatibility to existing TCP API's in order to allow its use by existing applications, only needing to be upgraded at the end host's operating systems.

MultiPath TCP should support some kind of TCP's session continuity. However the circumstances may be different. In regular TCP session continuity is when a session can survive a brief connectivity break by retaining the state at the end host before a timeout occurs, the Internet Protocol (IP) addresses will remain constant during that time. However in MPTCP a different interface might appear. So it is desirable to support a kind of *break-before-make* session continuity. A *break-before-make* session is when it interrupts the failed connection before establishing a new one.

The goals that MultiPath TCP aims to meet at **network compatibility** is to be compatible with the Internet as it exists today, retain the ability to fall back to regular TCP and should have the ability to work with bothIPv4 and IPv6 interchangeably. In terms of compatibility with the Internet as it is today, MultiPath TCP is constrained to appear as TCP does to be able to traverse predominant middleboxes, such as firewalls and Network Address Translation (NAT)s. It may be needed to MultiPath TCP fall back to regular TCP when there are a group of incompatibilities too great to overcome for the multipath extension on a path. The need for MultiPath TCP to have the ability of interchangeably work with both IPv4 and IPv6 happens when a connection operates over both IPv4 and IPv6 networks.

At the **users compatibility** point of view the goals are to enable new MultiPath TCP flows to coexist with existing single-path TCP flows, without being too aggressive towards them. In a shared bottleneck MultiPath TCP flows may not purposely harm users using single-path TCP flows, beyond the impact that would occur from another single-path TCP flow. Also on a shared bottleneck multipath flows must share with fairness the bandwidth, as it would occur at a shared bottleneck with single-path TCP.

From a **security** point of view, MultiPath TCP has to provide a service no less secure than regular, single-path TCP.

B. Transport Layer Structure

In this topic we describe the structure that MPTCP uses at the transport layer.

The MPTCP splits the transport layer into two sublayers as it can be seen in Figure 1. The upper sublayer is responsible for gathering the information necessary to manage the connection and operates end-to-end. The lower sublayer is responsible for the subflows, in order to make them be seen as a single TCP flow and allows the TCP component to operate segment-by-segment. This structure was designed in order to be transparent to both the higher and the lower layers.

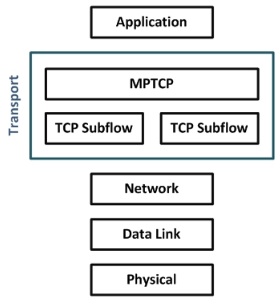

Fig. 1: Transport Layer Structure

In order to manage the multiple TCP subflows below the MPTCP extension has to implement path management, packet scheduling, subflow interface and congestion control functions.

The **path management** is responsible for detecting and using the available paths between two hosts. This function is also responsible for the mechanism of signaling alternative addresses to hosts and to set up new subflows joined to an existing MultiPath TCP connection.

At the **packet scheduling** is where the byte stream received from the application is broken into segments in order to transmit them on one of the subflows available. Also at the packet scheduling is where the connection-level re-ordering happens when it receives the packets from the TCP subflows.

To allow the correct ordering of the segments sent on the different subflows, the MPTCP design uses a data sequence mapping, by associating the segments to a connection-level sequence numbering. In order to have the correct information of the subflows available the packet scheduler depends upon the information acquired from the path management component.

The **congestion control** function is responsible for coordinating the congestion control across the subflows. This coordination is responsible for scheduling which segments should be sent on which subflow and at what rate, is also a part of packet scheduling.

The **subflow interface** is responsible to transmit on the specified path, the segments received from the packet scheduling component. Upon receiving a segment the subflow passes the data to the packet scheduling for connection-level reassembly.

Since MPTCP underneath uses TCP for network compatibility, it ensures in-order, reliable delivery. To detect and retransmit lost packets at the subflow layer TCP adds to the segments its own sequence numbers.

Internet Assigned Numbers Authority (IANA) has created a sub-registry to be used by MPTCP in the TCP Options field, as defined in RFC 6824 [7]. The TCP Option reserved for MPTCP is the Kind 30. It has a variable length and a 4-bit subtype field entitled "MPTCP Option Subtypes". The subtypes are listed in Table I and will be briefly described through the rest of this document.

Value	Symbol	Designation
0x0	MP_CAPABLE	Multipath Capable
0x1	MP_JOIN	Join Connection
0x2	DSS	Data Sequence Signal
0x3	ADD_ADDR	Add Address
0x4	REMOVE_ADDR	Remove Address
0x5	MP_PRIO	Change Subflow Priority
0x6	MP_FAIL	Fallback
0x7	MP_FASTCLOSE	Fast Close
0x8-0xe	Unassigned	
0xf	Reserved for Private Use	

TABLE I: MPTCP Option Subtypes

C. MPTCP Implementations

To test the feasibility of the MPTCP protocol, several implementations were developed and tested, some of which will be discussed and had a big influence in the direction taken by this project. With the evolution of different implementations of MPTCP it became interesting to implement in other scenarios, such as the use of smartphones with multiple network interfaces, data centers, mobile communications and multihomed networks. In [12] can be found a group of other implementations that would be interesting to evaluate with real network data. MPTCP is largely scalable and benefits from the fact that does not require changes at the application layer

The success of MPTCP will mainly depend on how it can easily be implemented in the real world. That is the objective of [13], which was the first Linux kernel implementation of MPTCP, enabling the evaluation of different implementation scenarios in order to show how the results can be affected by those choices. In order to provide a nearly perfect implementation the solution still needs to be improved.This solution also needs to implement multipath aware retransmission mechanisms, because this solution still implements TCP retransmission mechanism on each subflow. It was used a testbeb to analyze the performance, the throughput, the delay on the receiver buffer and showed that the coupled congestion control of MPTCP is fairer than the single TCP congestion control scheme.

IV. Testbed

In order to do an experimental study of the implementation proposals, we perform experimental tests and benchmarking by developing a testbed.

This testbed takes into account two very distinct contexts. The first has the objective to evaluate the implementation of MPTCP on LAN environments, where we proceed to assess the conventional wired and wireless scenarios. The second context has the objective to evaluate MPTCP on a mobility environment, using not only WLAN but also 3G/4G connection through a network operator.

A. Conventional Wireless and Wired Scenarios

The testbed, in this wireless and wired context, consists on the use of four static environment scenarios. In order to achieve the results desired for the different scenarios, the computers used need to be running a Linux Kernel with MPTCP enabled implementation.

1) Multihoming Solutions: The first scenario consists on a multihoming solution implementation of MPTCP, using two network interfaces, WLAN and Ethernet, as show in Figure 2. The computers in use have two network interfaces and are configured to use two different network links. This scenario allows us to obtain information of load balancing used by the MPTCP protocol when encountered with different network interface connections. It also enables us to evaluate the handover between the different networks and the connection recovery in case of failure in the network.

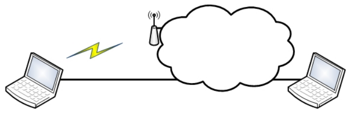

Fig. 2: Multihoming MPTCP with WiFi and Ethernet Connection

The second scenario is similar to the first, seen as it is a multihoming solution implementation of MPTCP, being so that in this scenario we use two network interfaces, both Ethernet, as show in Figure 3.

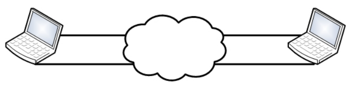

Fig. 3: Multihoming MPTCP with two Ethernet Connections

In this scenario we can evaluate how the load sharing is accomplished when used in homogeneous networks. We eventually will proceed to strangle one of the connections. This scenario will also allow us to evaluate the resilience of the network.

2) Subflow Analysis: The third scenario, depicted in Figure 4 and 5 consist on moving the traffic using only one of the network links, in order to evaluate MPTCP subflows and throughput. We test this scenario with a wired and wireless connection, in order to evaluate the difference between the use of each network interface.

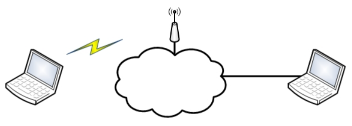

Fig. 4: MPTCP Subflow Analysis WiFi

We can capture the individual packets on each subflow in order to analyze the data transferred and draw conclusions about MPTCP functionality and efficiency.

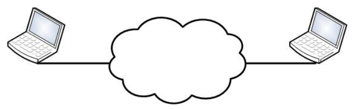

Fig. 5: MPTCP Subflow Analysis Ethernet

3) TCP and MPTCP Concurrent Scenarios: Last but not least the four scenario is an implementation of a simulated network, as it is shown in Figure 6 and 7. This scenario consists on using two computers with MPTCP enabled implementation, computers connected to two networks, and two others using regular TCP, computers connected to one network.

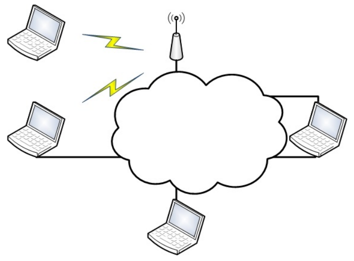

Fig. 6: TCP and MPTCP Concurrent Scenarios with Ethernet and WiFi Connections

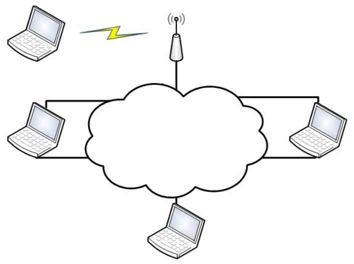

Fig. 7: TCP and MPTCP Concurrent Scenarios with Ethernet Connections

This scenario provides an implementation that shows how the MPTCP protocol acts when there are regular TCP flows on the same network, in order to evaluate its fairness and how it reacts to network bottlenecks.

B. Mobile Solution

In our solution we test a mobile implementation for MPTCP, in order to evaluate the capability of mobility offered by the protocol and its handover between different network protocols. This is an interesting approach that we implemented in an android smart phone to take advantage of its 3G/4G and WLAN capability, in order to evaluate how the protocol offers a better performance and reliability of the connection while moving.

The use of a mobile phone with the android operating system allows us to develop in a highly configurable and heavily supported platform.

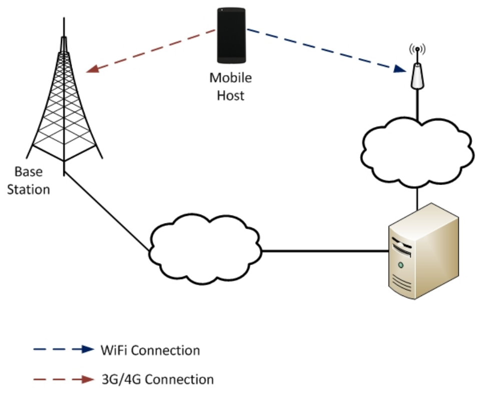

Fig. 8: Mobile Scenario

To evaluate the throughput and handover we will use 3G/4G and WLAN interfaces, as shown in Figure 8. This scenario allows to evaluate the handover between networks and the resilience of the connection.

V. Implementation

This section describes the different phases perform to achieve the MPTCP testbed implementation. As it has been referred before, to implement a multipath solution we only need to modify the endpoints of the network, end users or servers. This section also describes the main components required by the testbed, not only software but also hardware. After choosing the right solution, we proceed to explain the necessary components for the network solution to create the different scenarios referred to in Chapter IV. And finally it describes the tests used to evaluate those scenarios, knowing that the endpoints have the capability of supporting MPTCP connections.

A. Requirements

To implement the testbed we needed to find an operating system that would be easily configurable and with full MPTCP support. Since Microsoft Windows and Macintosh Operating System (Mac OS) are not an open source solution, we choose to use the Linux Operating System, both for the endpoint devices and the server. Linux is one of the most common open source operating system. This testbed was implemented in three different types of devices, such as laptops, desktop computers and a mobile phone.

The mobile solution uses the most reliable and configurable environment, the Android mobile operating system.

1) Laptops: The laptops used were ASUS Eee PC 900A, with the Debian GNU/Linux Whezzy release. The Whezzy release is ideal to install in this laptop because is not heavy for the device. The laptops have a built-in 802.11 b/g WLAN, and a Fast Ethernet interface.

We have installed a enabled MultiPath TCP - Linux Kernel Implementation [14], on the laptops to use them as concurrent MPTCP clients. But the these devices do not support multi-homing between the two network interfaces, WLAN and fast

Ethernet. The device only allows one interface to be used at a time.

Taking this into account we decided to use the laptops as TCP clients over the simulated network scenario.

2) Desktop Computers: The desktop computers were initially chosen to use a Debian Whezzy distribution, but we ended up using Ubuntu Saucy distribution. The Ubuntu Saucy is a Debian-based Linux operating system that offers a wider range of drivers support.

We installed a enabled MultiPath TCP - Linux Kernel Implementation [14] v0-88[1], on the desktop computers. In order to evaluate the use of MPTCP we needed to install networking tools that were MPTCP aware. We also have modified an application that could measure the network used on each interface in real time.

The desktop computers are equipped with two network interfaces, a gigabit Ethernet and a fast Ethernet. In order to create the wireless connection on the client side we used EZ Connect N 150Mbps Wireless USB2.0 Adapter (SMCWUSBS-N3), that supports IEEE 802.11 b/g/n connection. From the server side we used the router MikroTik RouterBOARD 2011UAS-2HnD, which also supports IEEE 802.11 b/g/n connection. The capabilities of the router used was the switch with and wireless access point. At the scenario with multiple users, section IV-A3, it was used the Cisco-Linksys WRT160N Wireless-N Broadband Router as a switch to connect the different Ethernet users to the server.

We tried to evaluate a scenario that used more than two network interfaces, but the system at the time it did not offer support for a implementation like this.

3) Mobile Phone: The mobile client is a LG Nexus 5 with Android 4.4. We installed the MultiPath TCP - Linux Kernel Implementation [14] v0-86 available for the device. This device is equipped with 3G, 4G/LTE and WLAN interface, that supports IEEE a/b/g/n/ac connection. The external network interface used in this project depends on the network coverage of the service provider.

This distribution still does not allow multihoming, of 4G with WLAN simultaneously. Nonetheless, we used it to evaluate the reaction to network failure, handover between networks and comparison of using MPTCP and TCP connections.

B. Network Routing

In conventional scenarios, in order to configure the network routing we only need to be concerned with the outgoing interface and the host destination. This occurs because the Linux kernel assumes that the host only uses a default gateway and interface. Since MPTCP allows the use of multiple addresses on various interfaces, by giving a different source or destination address to each subflow, it is not enough to use a default configuration of the network routing.

Linux routing policies have the capability to allow the kernel to redirect the traffic to use a specific routing table, according to the source address. In order to identify the available paths per interface, we need to configure a routing table for each interface.

[1]http://www.multipath-tcp.org

Whenever an interface becomes available or changes configurations, the MultiPath TCP kernel requires to be reconfigured. It also requires to manually remove the previous configurations, by deleting the policy rule and flush the routing table associated. Since this configurations are inefficient and time consuming, it is more effective to have a script with the necessary configurations ready to deploy. The NetworkManager is responsible for managing every interface on the Linux kernel. This solution could be extended to support IPv6, but for testing purposes we have chosen to use only IPv4. Whenever the NetworkManager detects a new interface, or loses a previous one, it automatically configures the interface accordingly.

The routers used in the implementation of the testbed do not use any specific configuration, beyond the required for being in the same network as described in the scenarios on Section IV.

The device used in the mobile solution is automatically configured by the network manager implemented by the Andoid MPTCP enabled solution.

C. Evaluation Tools

The experimental data was obtained with the use of well known tools, such as **tcpdump**, **iperf**, **ping**, **netstat** and **wireshark**. These tools were previously modified in order to support the MultiPath TCP protocol. We also used the iproute2, a collection of user space utilities, more specifically the **ss** command utility, to obtain network statistics. Also developed an application that allows us to listen to the data sent on each interface.

The developed application is similar to the bwm tool, which is a bandwidth monitoring tool for Linux environments, but instead of just showing the information on the console, the application stores the data collected on each interface on a text file every 1/2 second.

On the mobile device we needed to install two applications for network evaluation. The first being an **iperf** application called iPerf for Android [2], to allows us to connect with the server. The second one was necessary to test the connection itself, to assure that the mobile phone was automatically configure to the right environment. In order to do that , we used the application Network Tools [3]. The experimental data needed to evaluate the mobile scenario was collect by the server side.

The MPTCP protocol is still at an early stage of development, therefore it is difficult to find networking tools with MPTCP support, specially for the mobile environment.

With the use of the performance and management tools described above, we can obtain all the experimental data needed to carried out the a valid tested solution. We will then compare the different test results obtained to the regular TCP solution, in order to prove that MultiPath TCP performs as good or better in this circumstances.

[2]https://play.google.com/store/apps/details?id=com.magicandroidapps.iperf

[3]https://play.google.com/store/apps/details?id=su.opctxo.android.networktools

D. Tests

The tests were performed over a real environment scenario, in order to distinguish from other solutions who resort to use of simulated environments. The equipments used to implement the testbed are described in Section V-A.

The Server and Client have a MPTCP enabled version of the OS.

The Mobile Client needs to install the enabled MPTCP version of the Andoid 4.4 in order to use the protocol. When the tests require to use TCP, in order obtain data to compare to the same type of connections, the smart phone needed to be setup with the original Android 4.4 operative system.

In order to get better results to compare the use of both protocols the tests were performed on two different contexts, described in Section IV. The first context allows us to evaluate the test the use of MPTCP over conventional wired and wireless networks. While the second is more focused on mobile environments.

To implement the testbed we relied on several test scenarios described as follows.

1) ***Scenario A*** *- Multihoming MPTCP with WiFi and Ethernet Connections:* This test scenario collects information of a multihoming implementation of MPTCP using Ethernet and Wireless Local Area Network (WLAN) connections, as shown in Figure 2. The system was configured to use multiple interfaces.In this scenario the computers were configured to use two different network links, one for each network interface.The Server is connected to the Switch AP to simulate a wireless access point in the network. The Client is connect to the Server via Ethernet, direct link, and via Wireless, through the Switch AP.

In this scenario we gathered information on load balancing, performance and throughput. We have also tested the MPTCP protocol reaction when the client loses connectivity on each interface or both, to evaluate the network resilience.

2) ***Scenario B*** *- Multihoming MPTCP with two Ethernet Connection:* This scenario is similar to the previous scenario V-D1, being that it is also a multihoming implementation of MPTCP, but now it uses two Ethernet connections, as shown in Figure 2. The system needs to be configured to use multiple interfaces.In this scenario the computers should be configured to use two different network links, one for each network interface.The Server and Client are directly connected through Ethernet network links.

In this scenario we gathered information on load balancing, performance and throughput. We have also tested the MPTCP protocol reaction when the client loses connectivity on each interface or both, to evaluate the network resilience.

3) ***Scenario C*** *- Subflow Analysis:* This scenario is used to collect information of the MPTCP subflows implementation, using WiFi and Ethernet connections separately, as shown in Figure 4 and Figure 5. The system was configured to use three subflows, for each MPTCP connection.In order to obtain adequate data from the use of MPTCP subflows, this scenario is divided in eight sub scenarios. Initially the Client has a WLAN interface that connects to the Server through a Switch AP.Than we test when the Client has a Ethernet interface that connects to the Server

In this scenario we gathered information on load balancing, performance and throughput, separately on both connections. We have also tested the subflows of the MPTCP protocol reaction when the client loses connectivity, to evaluate the network resilience.

4) ***Scenario D*** *- TCP Analysis:* This scenario collects the TCP information using the same tests of the Scenario V-D3. The system was configured to have MPTCP disable.The data collected from using TCP, allows to make a direct comparison between the use of both protocols.

5) ***Scenario E*** *- TCP and MPTCP Concurrent Scenarios:* This test scenario is used to collect information of a network with both MPTCP and TCP users.

This scenario has two network configurations. The first network configuration uses the scenario shown in Figure 6. The MPTCP Client is connected with Ethernet to the Server, through the Switch, and WiFi through the Switch AP. The TCP Client1 connects to the Server via Wireless, through the Switch AP. While the TCP Client2 is connected to the Server through the Switch. While the second configuration, uses the scenario shown in Figure 7 The MPTCP Client is connected with two Ethernet to the Server, one through the Switch the other through the Switch AP. The TCP Client1 connects to the Server via Wireless, through the Switch AP. While the TCP Client2 is connected to the Server through the Switch.

The TCP Client1 and TCP Client2, do not have MPTCP support. They sent data to the server using TCP protocol.

In this test we gathered information in order to evaluate if the connections using the multihoming MPTCP are fair to the TCP connections on the same links. We also proceed to evaluate the fairness between MPTCP subflows and TCP connections, when in competing bandwidth. And collected information using only TCP connections to provide a comparative analysis.

6) ***Scenario F*** *- Mobile Solution MPTCP:* This scenario is set in the context of mobile environments. In order to assess the MPTCP implementation on this type of environments we used the architectural scenario described in Figure 8, based on the architecture described on the Section IV-B. This scenario uses three types of connection, local WiFi connection, campus WiFi connection and 3G/4G connection provided by a commercial 3G/4G Internet Service Provider (ISP).

In this scenario the Server and the Mobile User must have MPTCP enabled solutions. The Server needs to have MPTCP enabled and to use multiple interfaces. The Mobile User needs to support MPTCP connections. This was accomplished by installing the Android 4.4 - MultiPath TCP - Linux Kernel Implementation, as noted in Section V-A3.

The Server in this scenario uses two connections. The first is a connection to the Switch AP to simulate a wireless access point in the network. The second is a link connection to Técnico Lisboa internal network.

The local WiFi connection allows us to gather information of the use of the protocol on a wireless environment without any restrictions. While the campus WiFi connection, Eduroam, allows us to evaluate the protocol's reaction when using a wireless environment with bandwidth restrictions and load generated by several users. The 3G/4G connection allows to perceive the use of the protocol over public networks.

This scenario gathers information of the MPTCP implementation on a smart phone Android, such as performance, resilience to failure and handover between networks.

7) ***Scenario G*** *- Mobile Solution TCP:* This scenario allows us to gather TCP information using the same architectural scenarios described on the previous test scenario V-D6. The Server was configured to have MPTCP disable.The data collected from using TCP, allows to make a direct comparison between the use of both protocols.
Several tests were performed for each type of test scenario described. All data was collected using the evaluation tools described on Section V-C.

The tests performed in the context of conventional wired and wireless networks had a duration of 100 seconds each, since no significant changes were found in tests with a longer time.

The tests performed in the context of mobile environment were measured for 60 seconds, because a longer connection would entail greater expense, due to the use of 4G networks. In this context the data was collected on Server side, due to limitations of the actual applications for Android with enabled MPTCP implementation.

E. Work Evaluation

We will use the following metrics described in order to evaluate the proposed solution.

Through **subflow analysis** we can verify that the subflows available on the network really belong to a single MultiPath TCP session. We need to also be able to verify that the different connections are being used to efficiently transfer the data.

The **network implementation** allows us to evaluate if the protocol can be applied to the existent infrastructures without the need for any adaptation, except at the endpoints. It allows us to evaluate if the protocol is fair to other connections available in the network, by analyzing the concurrence between the MPTCP subflows and the existing traffic.

With the **experimental data** we can evaluate the performance of the protocol. Evaluating if the load balancing is actually being effective on the network. Comparing if the cumulative throughput in order to effectively prove that it is equal or greater than a single TCP connection, with the same characteristics.

Additionally we can verify that the connection stays active after a network physical failure, to test the protocol's resilience to failure.

VI. Results

This section presents the analysis results of the tests. The description of the test scenarios performed can be found in Section V-D.

1) Throughput Measurements: In this subsection is described the performance results of both protocols in different test scenarios. The values presented are an average of several measurements, obtained on each scenario.

Mobile Environment			
Test Scenario	Network	Protocol	Throughput (Mbps)
Scenario F	Local WiFi	MPTCP	37.6
	Campus WiFi	MPTCP	7.8
	ISP Network	MPTCP	9.4
Scenario G	Local WiFi	TCP	43.5
	Campus WiFi	TCP	2.5
	ISP Network	TCP	8.6

TABLE II: Throughput Measurements of Mobile Interfaces

The Table II shows the bandwidth capability of each network interface used in the context of mobile scenario, Scenario F. Overall MPTCP protocol have better performance, on the Campus WiFi and the 4G network, than the similar TCP ones, in Scenario G.

As it can be observed the MPTCP throughput in the Local WiFi network, Scenario F, is actually lower than in the TCP case, Scenario G. The lower throughput obtained in the Scenario F can be considered a random event. This can be caused by collisions in the WLAN, thus lowering its throughput, due to other networks being used in the same channel.

Wireless And Wired Connections			
User	Protocol	Interface	Throughput (Mbps)
Client	MPTCP	WiFi	33.2
		Ethernet	89.4
		Total	122.6
TCP Client1	TCP	WiFi	2.1
TCP Client2	TCP	Ethernet	94.1
Client	MPTCP	Ethernet	97.1
		Ethernet	77.1
		Total	171.2
TCP Client1	TCP	WiFi	20.6
TCP Client2	TCP	Ethernet	92.3
Client	MPTCP	WiFi	36.9
TCP Client1	TCP	WiFi	0.2
Client	MPTCP	Ethernet	92.8
TCP Client2	TCP	Ethernet	94.1
Client	TCP	WiFi	35.5
TCP Clien1	TCP	WiFi	0.4
Client	TCP	Ethernet	94.1
TCP Clien2	TCP	Ethernet	94.1

TABLE III: Scenario E - Throughput of Wireless And Wired Connections in Congested Networks

Table III compares the performance of both protocols, MPTCP and TCP, when they are competing with each other for bandwidth access, in the context of conventional wireless and wired connection scenarios.

As we shown in the table III, the aggregation of multiple paths by MPTCP provides a total bandwidth that is close to the sum of individual TCP flows. As shown in the Table **??**, the connection of MPTCP Subflow has a lower throughput than the one obtain in TCP, scenarios C and D, for Ethernet and WiFi. This can be explained with the overhead created for each subflow.

When using Ethernet network links the MPTCP has shown to be fair to TCP users over the same network link. However, when MPTCP competes in WLAN networks, it has shown to really affect the connection of TCP users. This could be a problem on the operating system implementation of the protocol.

In conclusion, MPTCP protocol improves the connections throughput per interface, in comparison to TCP. Which means, that it also improves the throughput of the overall connection.

In congested environments, MPTCP is fair to TCP connections.

2) Recovery Time: In this subsection is compared how both protocols react to network failure, as described in Section V-D. The failures were cause through physical intervention.

Wireless And Wired Connections				
Test Scenario	Protocol	Interface	Link Failure	Recovery Time (s)
Scenario A	MPTCP	WiFi	Yes	15
		Ethernet	No	-
	MPTCP	WiFi	No	-
		Ethernet	Yes	15
	MPTCP	WiFi	Yes	15.5
		Ethernet	Yes	15
Scenario B	MPTCP	Ethernet	Yes	13
		Ethernet	No	-
	MPTCP	Ethernet	Yes	12.5
		Ethernet	Yes	13
Scenario C	MPTCP	WiFi	Yes	6
		Ethernet	Yes	4.5
Scenario D	TCP	WiFi	Yes	15.5
		Ethernet	Yes	13

TABLE IV: Recovery Time of Wireless And Wired Connections

Table IV shows the time needed for the connection to recover from a failure, in the context of wireless and wired connections. As it can be seen MPTCP recovers from failure in the same time, or less, than a TCP connection.

As shown in Table IV when using MPTCP with multihoming, the time to recover from failure is influenced by the interface that takes the longer time to recover, scenarios A and B. In the case when MPTCP is using subflows it recovers much faster than MPTCP multihomed and TCP, this is due to the fact that the connection remains active for a longer period of time, depending on the number of subflows being used.

Mobile Environment				
Test Scenario	Network	Protocol	Link Failure	Recovery Time (s)
Scenario F	Local WiFi	MPTCP	Yes	12.5
	Campus WiFi	MPTCP	Yes	15.5
	ISP Network	MPTCP	Yes	11
Scenario G	Local WiFi	TCP	Yes	13.5
	Campus WiFi	TCP	Yes	32.5
	ISP Network	TCP	Yes	29

TABLE V: Recovery Time of Mobile Connections

Table V shows the time needed for the connection to recover from a failure, in the context of mobile scenarios.

As shown in Table V the MPTCP is faster than TCP, to recover from failure. The Scenario F, take a less time to recover

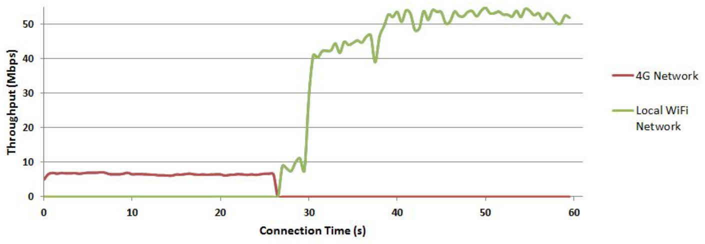

Fig. 9: Scenario F : MPTCP Handover between 4G and Local WiFi Network

from failure, than Scenario G, on each network. This happens becauseMPTCP keeps the link status for a longer time than TCP.

In conclusion MPTCP is more resilient to failure than TCP.

3) Handover: In this subsection we show how the MPTCP connection reacts to handover, while maintaining the same connection to the Server. There is no TCP handover, because when a interface loses connection and another connects, the TCP creates a new connection to the server, closing the previous one.

Figure 9 shows the handover between the 4G and Local WiFi Network. The time to complete the handover is less than 0.5 seconds.

In the context of mobile environment scenarios the handover is almost seamless, except when the mobile user needs to authenticate.

This shows that MPTCP has the advantage of keeping the same connection, even after the handover between different networks.

4) Summary Analysis: The MPTCP protocol has been tested successfully using existing infrastructures and adapting only the endpoints.

MPTCP has shown to be as fair as TCP, most of the time.

The MPTCP has shown to improve connection performance, by improving throughput of the overall connection.

The MPTCP also showed to be more resilient than TCP. In section VI-2 it was shown that it recovers from network failure more quickly than TCP.

Finally it offers handover between different networks without losing the connection, it was shown section VI-3.

VII. Conclusions

A. Syntheses

In order to create a MultiPath TCP (MPTCP) testbed to identify the protocol potential and limitations, it was necessary to study the evolution of the different approaches taken to produce a multipath protocol.

In the first part of this thesis we addressed mutipath implementation at different network layer. From this analysis we justified why a multipath solution should be tested at the transport layer, since it brings the possibility of being transparent to both the application and the network layers.

The work developped in this thesis confirms that the evolution of the multipath TCP protocol shows several challenges in order to be easily deployable without the need of other changes at the network and application level. The IETF MPTCP working group efforts have also been analyzed and taken into account for the creation of the testbed.

In this work we have also described the recent structure of the implementation for MultiPath TCP, by naming the goals defined for this protocol, explaining its structure and how it works, since the start until the closing of a session.

A wide range of test scenarios were developed in this testbed in order to evaluate the use of MPTCP protocol on real network environments. It was shown that MPTCP has the potential to improve connectivity resilience or bandwidth in datacenters and mobile scenarios, as well as in other cases where multihoming and multi path connections are available. Moreover, we made an extensive evaluation of the MPTCP protocol, and we verified that it can in fact balance network congestion, offering higher throughput and being more resilient to failures over the network. In several scenarios, MPTCP showed overall performance better than conventional TCP.

Summarily we believe that this testbed shows that an implementation of this protocol over Técnico Lisboa network will increase its overall performance. In summary, we we believe that this testbed shows that an implementation of this protocol on some Técnico Lisboa services where multipath connections are available may increase their overall resilience and performance.

B. Discussion and Future Work

The work developed so far, for a multipath protocol at the transport layer, with a Linux kernel implementation with MPTCP, allowed the research community to investigate different topologies and implementations.

Future work could focus on developing an implementation that would allow the use of a greater number of interfaces, with several types of connection. Another point of focus, could be directed to the development of a congestion control algorithm that would allow the protocol to compete more fairly in wireless communication networks.

Another approach could be to develop MPTCP aware applications. Applications that would allow to prioritize the type of traffic being used, on each interface. Other applications could be developed to allow the configuration of the enabled MPTCP kernel in smartphones, in order to further test different approaches of the protocol for this type of devices.

References

[1] J. Postel, "Transmission control protocol," Internet Engineering Task Force, RFC 793, September 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[2] R. Stewart, "Stream Control Transmission Protocol," RFC 4960 (Proposed Standard), Internet Engineering Task Force, September 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4960.txt

[3] J. Iyengar, P. Amer, and R. Stewart, "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 5, pp. 951–964, 2006.

[4] M. Zhang, J. Lai, A. Krishnamurthy, L. L. Peterson, and R. Y. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths." in *USENIX Annual Technical Conference, General Track*, 2004, pp. 99–112.

[5] H.-Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts," *Wireless Networks*, vol. 11, no. 1-2, pp. 99–114, 2005.

[6] E. Nordmark and M. Bagnulo, "Shim6: Level 3 multihoming shim protocol for ipv6," RFC 5533, June, Tech. Rep., 2009.

[7] A. Ford, C. Raiciu, and M. Handley, "TCP extensions for multipath operation with multiple addresses," Working Draft, IETF Secretariat, Fremont, CA, USA, Internet-Draft draft-ietf-mptcp-multiaddressed-02.txt, Jul. 2010. [Online]. Available: http://tools.ietf.org/id/draft-ietf-mptcp-multiaddressed-02.txt

[8] C. Wang, K. Sohraby, B. Li, M. Daneshmand, and Y. Hu, "A survey of transport protocols for wireless sensor networks," *Network, IEEE*, vol. 20, no. 3, pp. 34–40, 2006.

[9] D. Allan, P. Ashwood-Smith, N. Bragg, J. Farkas, D. Fedyk, M. Ouellete, M. Seaman, and P. Unbehagen, "Shortest path bridging: Efficient control of larger ethernet networks," *Communications Magazine, IEEE*, vol. 48, no. 10, pp. 128–135, October 2010.

[10] W.-H. Tarn and Y.-C. Tseng, "Joint multi-channel link layer and multipath routing design for wireless mesh networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE.* IEEE, 2007, pp. 2081–2089.

[11] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.

[12] S. Barré, O. Bonaventure, C. Raiciu, and M. Handley, "Experimenting with multipath tcp," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. –, Aug. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=2043164.1851254

[13] S. Barré, C. Paasch, and O. Bonaventure, "Multipath tcp: From theory to practice," in *NETWORKING 2011*, ser. Lecture Notes in Computer Science, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, Eds. Springer Berlin Heidelberg, 2011, vol. 6640, pp. 444–457.

[14] S. B. e. a. C. Paasch, "Multipath tcp - linux kernel implementation," 2014. [Online]. Available: http://www.multipath-tcp.org