TÉCNICO LISBOA

Multipath TCP Protocols

Dinamene de Lima Correia Almeida Barreia

Thesis to obtain the Master of Science Degree in

Telecommunications and Informatics Engineering

Supervisor: Prof. Fernando Henrique Côrte-Real Mira da Silva

November 2014

Acknowledgments

I would like to thank my parents for all their support and encouragement throughout this journey, for always being there for me thereby allowing this project to be accomplished.

I would also like to acknowledge my dissertation supervisor Prof. Fernando Mira da Silva for its insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends that helped me grow as a person and were there for me during the good and bad times in my life. Thank you.

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.

Resumo

O paradigma das redes de comunicações encontra-se em constante mudança, tendo um número cada vez maior de dispositivos, móveis, necessitando de uma maior diversidade de acessos e recorrendo cada vez mais ao uso aplicações ubíquas. O TCP é o protocolo mais usado para transmitir e entregar informação de forma fiável através da Internet. No entanto, este protocolo recorre ao uso de uma única ligação, não aproveitando o multihoming e os diferentes caminhos que possam estar disponíveis entre os dispositivos, ou seja, dispositivos móveis e servidores com configurações de alta resiliência. O protocolo MPTCP está a ser desenvolvido para lidar com estas limitações do TCP. O protocolo MPTCP pretende fazer uso da diversidade de caminhos por forma a oferecer uma melhor conectividade da rede global, aumentando a capacidade de resistência a falhas, realizando o balanceamento de carga entre os caminhos disponíveis, quando existe mais do que um disponível, e para permitir o suporte de multihoming sem a necessidade de modificar os dispositivos que se encontram actualmente espalhados pela rede. O objetivo deste projeto é criar um plataforma de testes que possa avaliar os benefícios e limitações do protocolo MPTCP, nomeadamente em cenários de dispositivos móveis.

Palavras-chave: MultiPath TCP; Resiliência; Desempenho; Plataforma de Testes; Multihoming.

Contents

List of Figures

List of Tables

Listings

List of Acronyms

3G	third Generation
4G	fourth Generation
ACK	Acknowledgement
AP	Access Point
API	Application Programming Interface
bwm	bandwidth monitor
CFLB	Controllable per-Flow Load-Balancing
cmpTCP	Concurrent Multipath Transport Control Protocol
CMP-SCTP	Concurrent Multi-Path Stream Control Transmission Protocol
CMT	Concurrent Multipath Transfer
DMP-streaming	Dynamic MPath-Streaming
DNS	Domain Name System
DoS	Denial-of-Service
DSN	Data Sequence Number
DSS	Data Sequence Signal
DWC	Dynamic Window Coupling
ECMP	Equal-Cost Multi-Path
EWTCP	Equally-Weighted TCP
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISP	Internet Service Provider
LACP	Link Aggregation Control Protocol

LAN Local Area Network

LTE Long-Term Evolution

MIPv6 Mobility Support in IPv6

MIPv4 Mobility Support for IPv4

mTCP multiple paths TCP

MPTCP MultiPath TCP

MSS Maximum Segment Size

NAT Network Address Translation

OLIA Opportunistic Linked Increases Algorithm

P2P Peer-to-Peer

pTCP Parallel TCP

QoS Quality of Service

RLB Randomized Load Balancing

RTT Round-Trip Time

SBPP-SCTP Sender-Based Packet Pair SCTP

SCTP Stream Control Transmission Protocol

Shim6 Protocol Shim6

SPB Shortest Path Bridging

TCP Transmission Control Protocol

TMPP Transparent MPTCP Proxy

UDP User Datagram Protocol

VLAN Virtual LAN

WAN Wide Area Network

WLAN Wireless Local Area Network

WMN Wireless Mesh Networks

W-SCTP Westwood SCTP

Chapter 1

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 them difficult to deploy 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.

1.1 Motivation

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, offer larger path diversity and better fairness.

1.2 Objective

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.

In order to make this possible the project is divided in two phases. On the first phase we focused our attention on an academic analysis and research of the different approaches that have been proposed for multipath TCP protocols by the IETF. The second phase is where we implemented and tested the different scenarios and techniques for the MPTCP protocol, created a testbed and evaluated the possible application on Técnico Lisboa internal network.

This thesis describes the work carried out, combining both phases and proceeding to an objective analysis of the MPTCP protocol.

1.3 Document Structure

This document is structured in seven chapters, described as follows. Chapter 2 describes an overview description of the related work that has been developed on this topic, where we can namely identify the different IETF approaches for MPTCP. In Chapter 3 is detailed the structure of the current MPTCP implementation. Chapter 4 defines the project's architecture, an overview of the proposed solution for the implementation and a description of the key components. Chapter 5 describes the implementation process followed by its evaluation in Chapter 6. In Chapter 7 a summary and conclusions on the work developed can be found, taking into account the future of the project.

Chapter 2

State of the Art

This chapter reviews the concept and state of the art of MultiPath TCP (MPTCP), the different Internet Engineering Task Force (IETF) approaches to handle multipath and multihoming scenarios, and possible applications for this subject. A research on other related projects is also presented and discussed.

2.1 Multipath Approaches

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. Some support the notion that the best layer for its implementation should be the network layer to provide these benefits. Some even have implemented their solution in the application layer, resorting to specific applications that spread the chunks of files into different peers, only caring about increasing throughput. In local area networks, it is also possible to explore path diversity at the link layer.

The implementation of a multipath protocol at the transport layer. Benefits from the transparency to the network layer and provides a relative independence to the application layer, avoiding the requirement to change the well established link and network layers. On the other hand, since MPTCP is a TCP extension, conventional single path TCP applications can be used without any change at the application level.

In the following subsections, is analyzed in deeper detail some of these alternative approaches.

2.1.1 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.

2.1.2 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.

2.1.3 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, 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.

2.1.4 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.

2.2 Alternative IETF Multipath Approaches

Over the years, the IETF has created a work group that has been investigating the simultaneous use of multiple paths at the transport layer. MPTCP was not the first approach, but it is both the most recent and promising.

2.2.1 Multipath Solutions

One of the first efforts to implement a transport protocol that would allow multiple streams across different paths was Stream Control Transmission Protocol (SCTP) [2], which offers a reliable transfer of user messages between two SCTP endpoints. It provides during the startup association a list of multiple IP addresses in combination with an SCTP port so that each endpoint can form a compatible transport address. The motivation for this protocol was based on the limitations of the TCP for the actual applications in order to send data reliably on top of the User Datagram Protocol (UDP) [12]. Another motivation was to natively support multihoming at the transport layer. The major problem of this protocol comes from the fact that it is not easily deployed on the Internet, due to legacy devices that do not recognize the protocol and mark its traffic as malicious or unknown. It does not offer a strict ordering across the different streams and can not use aggregate bandwidth on multiple paths.

An interesting approach that was important for the evolution into this protocol was the *mTCP* [4]. This is an end-to-end protocol that can aggregate available bandwidth between parallel redundant paths, that are very common between a pair of hosts, achieving a better performance and more robustness to path failures. This protocol works by stripping a flow's packets across multiple paths. It can provide a higher end-to-end throughput and it also supports a shared congestion detection mechanism in order to avoid paths that have shared congestion.

Another approach is the protocol *pTCP*[5]. This protocol is a end-to-end transport layer approach for aggregation of bandwidth on multihoming agents (mobile hosts). This protocol can separate loss recovery from congestion control and also performs an intelligent stripping of the data across the available connections.

Other SCTP-based approaches, such as Sender-Based Packet Pair SCTP (SBPP-SCTP) and Westwood SCTP (W-SCTP), also have the ability to allow simultaneous use of multiple streams [13].

Another approach, the one that shows more resemblance to the protocol that is being studied, is the Concurrent Multi-Path Stream Control Transmission Protocol (CMP-SCTP) [3]. This protocol is an extension for the SCTP protocol that benefits from the multihoming feature in order to increase throughput between two multihomed endpoints. This extension adds fault tolerance by sending data concurrently on all paths, instead of using probes to test the available paths. It uses a single sequence space across an association of multiple paths, it assumes a mechanism of single congestion control and another of single retransmission on all the paths. The use of a single sequence space allows it to perform end-to-end load sharing with Concurrent Multipath Transfer (CMT) at the transport layer. The sender can receive acknowledgment information on either of the return paths, which achieves an effective load distribution at transport layer.

An interesting approach of an extension of SCTP with TCP New Reno [14] is Concurrent Multipath Transport Control Protocol (cmpTCP) [15]. This extension was based on a evolution of CMT protocol. To avoid congestion window management problems the authors developed a Markov model for its estimation, with the possibility of modification of each path independently. The authors have also introduced for each path a virtual retransmission queue and have implemented a different association, initiation and termination, in order to allow concurrency by all the paths.

In the last few years the IETF has published a series of documents that describe and discuss MPTCP. The guidelines used for this project are based on [16]. This RFC defined the structure for the MPTCP protocol. In [7] it can be found the different available extensions that are supported by MultiPath TCP. For this protocol to evolve, it was necessary a careful implementation with its interface application [17], in order to ensure backward compatibility to regular TCP.

2.2.2 Multihoming Solutions

An interesting aspect of MPTCP is that it can take advantage of using simultaneously several interfaces available in a device, allowing it to improve connectivity reliability in mobile equipments.

An approach for mobility is the Mobility Support in IPv6 (MIPv6) [18]. The MIPv6 is a protocol that allows nodes to be reachable while they're moving, as long as they're in a IPv6 Internet. This is possible because the mobile node has three IPv6 addresses, the first one being the node's home address, the second the node's link-local address and the third is the mobile node's care-of-address. These three addresses allow the home-agent and the foreign-agent to create a tunnel, while the node is away, which enables the home agent to know how to find the mobile node independently of its location. Before the MIPv6, there was also an implementation of mobility for IPv4, Mobility Support for IPv4 (MIPv4) [19]. This protocol was the base for the MIPv6, with the difference that it assumed that every node on the network had a unique address, which as become obsolete with the growth of the number of personal computers and the emergence of everyday objects connected to the Internet, such as mobile phones, tablets, watches, TVs, cars and so on. The major problem with these mobility solutions is that they do not provide seamless handover.

Another mobility approach that has multihoming capability is the Shim6 protocol [6]. This is a Layer 3 protocol, which means that this IPv6 solution allows the implementation of multipath at packet level instead of routing at connection level. The major drawback of this protocol is that the load balancing is carried out at the connection level, meaning that the throughput is highly affected by the slowest or most congested link.

Although the security part of the protocol is not the focus of this project, it is important to have in mind that there are some security threats for multihoming solutions. In [20] it is described the threats that exist for these type of solutions.

One of the most common threats on a topology like this one happens when the attacker is identified as a valid source/destination. Another threat can cause packets to be sent to an unknown locater, that no longer exists or is unreachable, making them to be dropped by the network. An attacker can perform redirection in order to create a third party Denial-of-Service (DoS). Finally it might accept packets from unknown or unverified sources, which can lead to packet injection on the receiver end.

Given that a multihoming solution is supposed to allow hosts to use multiple parallel connections, without impacting the protocols at the transport and application layers, this makes it more susceptible to these types of threats.

2.2.3 Alternative Approaches to Multipath Scenarios

Other projects are being developed, these projects are not yet a final version and have not been approved by the IETF, but they provide a different point of view of what's to come about the topic of this paper.

Beijnum has developed an application perspective of a multipath TCP protocol that is only implemented on the side of the sending host, without requiring any modifications to the receiver end [21].

This view allows the protocol to either use the available capacity on the different multiple paths, or dynamically find the best performing one, with the final objective to get a higher throughput.

In order to reduce the computational overhead for an MPTCP connection establishment and its TCP subflows, C. Paash and O. Bonaventure have been doing research available in [22]. For this study the authors assume that the MPTCP handshake proposed is executed in a controlled environment and that security attacks are not an issue.

A different point of view that is being developed, by Xue et al., is the Transparent MPTCP Proxy (TMPP) [23]. The idea behind their work is to allow MPTCP-unaware hosts to use multiple path support. It may be used in the access network or the operator's network. TMPP works like a gateway in a Network Address Translation (NAT) environment, by changing the received packets and retransmitting them again.

Since MPTCP is a recent subject, Bagnulo et al., is developing a study of the possible threats in this matter [24]. The authors have defined types of attackers and a combination of possible attacks in order to present different solutions to deal with the threats, always keeping in mind to *"make MPTCP no worse than the currently available single-path TCP"*.

The actual IETF version of MPTCP is described in deeper detail on Chapter 3.

Chapter 3

Multipath TCP

This chapter describes some of the most relevant properties and services of the current MPTCP draft, as described in RFC 6824 [7]. Also describes potential advanced applications for this protocol.

3.1 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 occurs when a session survives a brief connection break, by retaining the state of the end host before the 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 both IPv4 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 NATs. 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 the bandwidth fairly, 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.

In order to achieve the primary goals described before for MPTCP, the congestion control algorithm selected has to comply with the following goals:

- **Goal 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.*

- **Goal 2 : Do no harm** *A multipath flow should not take up any more capacity on any one of its paths than if it was a single path flow using only that route.*

- **Goal 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.*

These three goals are in accordance with the goals defined and have the capability to offer a better throughput solution for a MultiPath TCP protocol. The first goal assures that the multipath flow should never have a lower performance than the singular-path TCP flow on the fastest path available, being in accordance with the functional level goal defined for MultiPath TCP. The second goal is in accordance with the users compatibility point a view. The third and final goal is an addition to the MultiPath TCP goals to get the higher throughput available, by moving the traffic on to less congest links.

3.2 Transport Layer Structure

In this section 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 3.1. 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.

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.

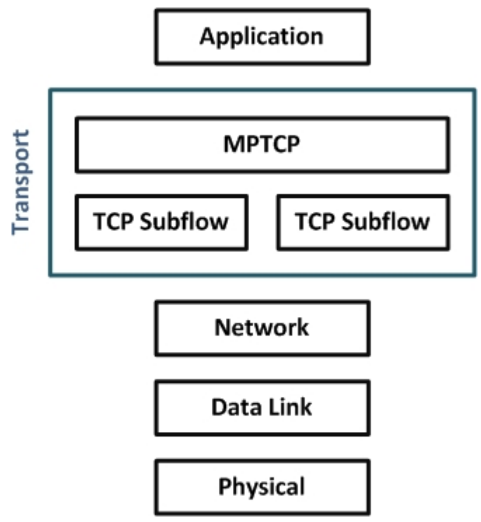


Figure 3.1: Transport Layer Structure

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, the connection-level re-ordering is performed whenever the packets are received 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 3.1 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 3.1: MPTCP Option Subtypes

3.3 Connection Handling

This section describes how MultiPath TCP handles connections, since the establishment to the conclusion of a connection.

3.3.1 Connection Establishment

At this point we describe how a new MPTCP connection starts and also how the establishment of new subflows occurs.

To establish a new connection the application will open a regular TCP socket, which will start the initial TCP subflow. As MPTCP has the need to be transparent to both network and application layer, until this point the connection is apparently the same as a regular TCP.

After the initial TCP connection, if both endpoints support MultiPath TCP capability, the MPTCP session can start on either host.

We will now explain a slightly simplified solution for a new MPTCP session and subflow establishment in order to describe the main idea, as shown in Figure 3.2.

To establish a new subflow from the Host A to Host B, the Host A receives from the DNS that it can reach Host B at the Address 1.

Now a three-way handshake is used similar to regular TCP, in order to continue transparent to the network, but at the options field it carries MPTCP specific options that it is only understood by the end-system stack.

The MP_CAPABLE option is inserted on the handshake to allow both hosts to inform each other that they have MPTCP capability. The first handshake in order to begin the MPTCP session from the Host A, starts by sending a SYN with a MP_CAPABLE option in order to notify the other host that it has MPTCP capability. The Host B responds with a SYN and ACK, with MP_CAPABLE option that announces token_B to the peer. The token is a local identification of the connection on a host. Now the Host A responds with a ACK, with MP_CAPABLE option announcing his token_A. At this step the MP_CAPABLE option is still present so the server can wait until the end of the handshake to create the state. A final ACK is sent to assure that the final MPTCP security data is received.

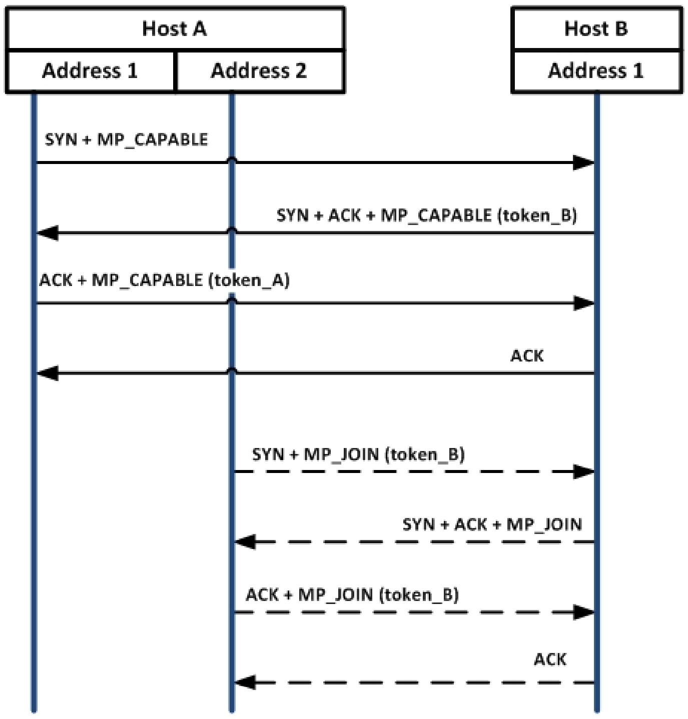


Figure 3.2: MPTCP Connection Initiation

The MPTCP session is now complete, from this point we explain how to add new subflows on a MPTCP session. At any point either hosts can try to establish new subflows.

Assume that Host A wants to create a subflow between its Address 2 and Host B Address 1. Again a new TCP three-way handshake starts, but the option in use is different, because now new subflows that will be created need to be attached to the existing MPTCP session. The option that will be used to create new subflows will be MP_JOIN. Host A can establish new subflow using the addresses <A.2,B.1>, Host A Address 2 with Host B Address 1, but it needs to be join to the correct context in Host B. To achieve this, Host A attaches token_B to the MP_JOIN option. Host B responds with SYN and ACK, with MP_JOIN option without a token, because Host A already has a state for that subflow. Then Host A sends ACK with the MP_JOIN, in the context of Host B. And the final ACK is sent and the subflow is ready to be used.

If a new address becomes available on one of the hosts, that new address needs to be announced to the other host before using it to connect a new subflow to the MPTCP session. First the host has to notify the other host with ADD_ADDRESS option of the MPTCP, containing the new address before starting a subflow handshake.

In the case where a subflow stops responding there is also a REMOVE_ADDRESS option to remove that address from the connection.

3.3.2 Exchanging Data

At this point it will be explained how MPTCP exchanges data across the multiple subflows.

To allow reliable and in-order delivery of the data across the subflows, MPTCP uses a 64-bit Data

Sequence Number (DSN) , which numbers all data sent over the MPTCP connection, and each subflow has its own 32-bit sequence number space as in regular TCP.

MPTCP specifies the mapping from subflow sequence space to data sequence space using a data sequence mapping, that reassembles the data stream and is sent over the Data Sequence Signal (DSS) option. The DSS contains the acknowledge and it is responsible for the information that allows the data sequence mapping. This MPTCP mapping can in the event of failure **retransmit** data on different subflows.

The data is scattered in segments over the available subflows and it needs to be **acknowledged** when received. To acknowledge the data MPTCP does a two layer acknowledgment. First the data is acknowledged on each subflow, as they behave like regular TCP flows. Then the data is acknowledged by the data sequence numbering by using a cumulative Data Acknowledgment, that is sent on one of the subflows.

3.3.3 Connection Release

At this point it will be described the closing of a MPTCP connection.

In a regular TCP connection the FIN is announced when an application calls close() on a socket, indicating that it has no more data to send. In MPTCP a FIN only affects the subflow on which is sent.

To actually close the connection in MPTCP there is an equivalent mechanism to the regular TCP FIN, that is referred to as the DATA_FIN. The DATA_FIN is acknowledged at the connection level with a DATA_ACK. The DATA_ACK message is only sent once all the data on the MPTCP connection has been successfully received.

When the first DATA_FIN is sent it triggers the return of both the DATA_ACK and DATA_FIN of the other host. The DATA_FIN only needs to be sent on one subflow and from the moment that it has been acknowledged, all remaining subflows should be closed with standard FIN exchanges. These FIN exchanges allows the middleboxes to clean up their state.

The MPTCP connection is considered closed once the DATA_FINs sent by both hosts have been acknowledged by DATA_ACKs.

The option MP_FASTCLOSE is used to abruptly close the whole connection, is only used by specific implementation.

3.4 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 [25] 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 [26], 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.

3.5 MPTCP Scenarios

Over the years there has been an interest in a protocol that not only offers seamless mobility, but that can also benefit from the use of all the available interfaces on a device. At this point it will be discussed different applications that have been envisaged for MPTCP implementation.

3.5.1 Congestion Control and MultiPath Routing Schemes

With the possibility of using multiple paths, concerns have arisen about congestion control of the data sent on those paths. A great deal of work was put into research on congestion control algorithms and path routing schemes that will optimally split the packets across the available paths, in order to achieve improved performance and be more resilient to problems on particular paths.

In [27] is shown that the implementation of some congestion control algorithms solutions for multipath TCP can be harmful, but is presented an algorithm that can improve throughput and fairness and be safely deployed on the Internet. Equally-Weighted TCP (EWTCP) [28] algorithm is a weighted version of multipath TCP and it can be fair to regular TCP traffic, but when used in multipath TCP is not very efficient, because it does not split traffic evenly across the available paths. In this paper is also analyzed the COUPLED congestion control algorithm, that makes better choices than the EWTCP, like falling back to regular TCP when there is only one path and solves some fairness problems. Nonetheless it has some issues, when the paths available have very different throughputs, which leads to unequal RTTs, it tends to typically send all its traffic on the less congested path. The solution presented is based on the SEMI-COUPLED congestion control algorithm that was modified in order to create an implementation for MPTCP, believing that it has a good selection of paths and a balanced congestion, also have designed it in order to ensure compatibility with existing TCP behavior.

Another effort on this subject is available in [29] and [30], where is presented a class of algorithms, that recurring to minimal congestion feedback, can optimally and stably split the traffic. Considering the availability of multiple paths, by overlaying the network with routers, the user can direct its flow by using the source routing capability available on those routers. It also evaluates the possibility of the use of multihoming capability, where the end hosts can separately address these paths. In this paper is showed that a decentralized control framework can in a natural way extend from a single path to a multipath case, and the total load through each link should not exceed its capacity, to maintain the stability of the system.

In [31] is studied the benefits of combining a multipath routing with a coordinated congestion control architecture. Performing load balancing at the lower level by using a Coordinated Multipath Congestion Controller that splits the session load across the lowest-cost available set of paths. At an higher level there is a periodical search for new paths using an algorithm that randomly selects them. This paper concludes that combining efficient routing with the use of a good resource pooling algorithm can bring robustness, resilience and performance improvement to the network, can also simplify the network dimensioning for providers.

In [32] can be found a modification to TCP, that provides a practical multipath congestion control algorithm, with the objective to be easily deployed and achieve TCP fairness, ensuring that MPTCP's design goals are met. It compares three congestion control algorithms : Fully Coupled, Linked Increases Algorithm and Uncoupled TCP.

The Fully Coupled algorithm considers the case where all the flows have similar RTTs, the major concern about this algorithm is that it causes what is referred to as *"flappiness"*. Flappiness is said to occur when the traffic of a multi-path connection tends to concentrate on one path and then another.

The Uncoupled TCP algorithm does not spread congestion, but it has difficulty to move traffic away from congested links. The Linked Increases Algorithm is proposed in order to reduce flappiness and combine resource pooling, it is a combination of the increase rule of the Fully Coupled and the decrease rule of the Uncoupled TCP. This algorithm can be too aggressive for one of its paths failing the second goal of MPTCP.

In order to compensate the different connections Round-Trip Time (RTT), it was studied a RTT Compensator Algorithm, but it sacrifices some congestion balancing for improved stability.

To implement this algorithm it is necessary to choose between undesirable flappiness and uncoupled subflows without network pooling resources, but this algorithm is still a variant of an improved and fair congestion control algorithm.

In [33] can be found a congestion control algorithm that is very similar to the one referred above, with the difference that this algorithm is more focused on the resource poolability when changes in the network occur. There is also a implemented Linux solution in order to test and research how reliable and deployable the algorithm can be, the algorithm showed that the resource pooling is an important step to provide an algorithm that makes good path decisions, by avoiding high-congested ones.

Improving reliability and performance on the Internet made it necessary to adjust the routing metrics to be more congestion sensitive, in [34] is studied how quickly routing can adapt, ensuring network stability. This method was based on a fluid-flow model in order to analyze the stability of an end-to-end algorithm, is shown that while the structural information may come from the network layer, it is more natural to perform load-balancing at the transport layer. The key constraint on each route is its round-trip time, for dynamic routing on their algorithm.

Based on Equal-Cost Multi-Path (ECMP) algorithm [35], it was proposed a method for next-hop selection based on an invertible function [36]. The paper provides an implementation and evaluation of the solution resorting to the Click modular router, where it analyzed the distribution of the packets. Finally this solution shows a small variation of the optimum case, the ability of control path does not have a significant impact on fairness on a local distribution.

3.5.2 Performance Evaluation of MultiPath Routing Schemes

In order to compare the different solutions formerly referred and their performance, in [37] it is introduced the Dynamic Window Coupling (DWC). DWC is an algorithm that tends to differentiate if the flows share or not a common bottleneck, and only couples MPTCP flows that share a common bottleneck. It is shown that with the possible scenarios of implementation, the design goals of the previous solutions can vary their performance from good to poor. The DWC can accurately detect a bottleneck and improve throughput. It also can solve Linked Increases Algorithm's problems, but at the expense of being less friendly to users with single TCP connections.

It was mentioned above the Linked Increases Algorithm for MPTCP, in sub section 3.5.1. This algorithm has its problems, its trade off between optimal resource pooling and responsiveness can reduce the MPTCP's throughput and be too aggressive to singular TCP users. So in [38] it was proposed the Opportunistic Linked Increases Algorithm (OLIA) in order to solve those problems and to respect MPTCP design goals. OLIA is also a window-based congestion control algorithm and compensates for different RTTs. It performs close to an optimal solution, solving Linked Increases Algorithm's problems, with a small probing cost, it satisfies the three design goals of MPTCP and is responsive and non-flappy.

One of MPTCP's benefits is how it can improve network performance. It has been made an effort to evaluate and research its results. In [39] is evaluated the MPTCP's performance from a point of view of throughput optimization and load sharing. This paper uses a testbed with three possible scenarios, which were, Ethernet-Ethernet, Ethernet-WiFi and 3G-WiFi. It shows that MPTCP has a better throughput in homogeneous paths than a singular TCP connection, it also performs an equitable load sharing on this type of paths. It has less throughput on paths with heterogeneous connections, but it offers improved resilience by automatically switching traffic upon failure.

3.5.3 Data Centers

With the possibility of MPTCP to improve data center's networks, there is some research on practical implementation in data centers and how it can improve them.

It was proposed a deterministic scheme that allows hosts to select the load-balanced path to use for a specific flow called Controllable per-Flow Load-Balancing (CFLB) [40], experimenting how it can improve a data center's performance and scalability by combining it with MPTCP. By testing the advantages of using MPTCP hosts together with CFLB-enabled network is shown that it can cover all paths with minimal cost. It also greatly improves the use of available diversity of paths offered by a data center network, between pairs of servers.

Data Center's have grown to an unprecedented size, due to the high quantity of applications created to be runned in distributed systems, stressing the network fabric within data centers. This originated a special interest in different data center topologies, in order to propose much denser interconnections, but they pose a major difficulty in routing. Currently it is implemented Randomized Load Balancing (RLB), which is not a favorable solution because it is probable that some links will never be selected while others will be overloaded. A solution for this problem is implementing MPTCP, which allows multiple connections between the servers with the automatically redirection of traffic according to a flows congestion. To see how feasible this solution can be, in [41] it is tested through simulation the effectiveness and robustness, of running MPTCP on data center's network. It was found some limitation on their practical implementation. Nonetheless, the results have been optimistic showing that MPTCP can significantly improve throughput and fairness with a simple solution, by uniting path selection with congestion control, such that traffic can move easily to paths that have spare available capacity.

It is impractical in a current data center to manually manage the tasks that need to be allocated, due to a large amount of servers and connections. With present solutions like in Amazon EC2 data center, that has a redundant structure, switches typically run a Equal-Cost Multi-Path (ECMP) routing variant. In order to create a number of flows significant to come close to generate traffic balancing, it needs a lot of TCP connections, this would not be scalable. Running a multi-stage topology with ECMP will very likely cause flows to collide on at least one link. An implementation in a large data center of MPTCP is evaluated in [42]. It proposes that exploring multiple paths with a congestion solution can be better for the data center, by offering a better throughput, due to the higher usage of the network and a fairer allocation of the capacity to flows. This work shows that the only place where MPTCP performed the same as single TCP was when the flows where limited at one of the hosts, either on the sender or the receiver. It also showed that MPTCP can actively relieve hot spots in the network, because it is very effective at finding unused capacity, and real performance benefits with a throughput up to three times higher.

3.5.4 Advanced Applications

The first implementation of a Linux kernel with MPTCP opened the doors to the research community to test different implementations and topologies to evaluate how good the protocol really is.

In [43], it was tested the performance, load-balancing, robustness and congestion on a topology that connects three servers with two switches, with multiple 1Gbps and 10Gbps to simulate LAN and WAN networks. One server is used to create only TCP traffic to generate congestion on a specific link. To simulate multiple connections, different VLAN's were created. With this experience the results were favorable by meeting the first and third goals of the MPTCP design, **improve throughput** and **balance congestion**, it also improved throughput and robustness, but the second goal, **do no harm**, was not verified as it was a bit unfair compared to single TCP. The tests showed that MPTCP works really well in a stable environment, same bandwidth and close RTTs, but behaving worse otherwise. In this article is also advised to be careful with the buffer sizes, because when the bandwidth increases the buffer and Maximum Segment Size (MSS) have to also increase, which leads to increased processing.

To make better usage of the MPTCP protocol, applications need to be aware of its implementation. The IETF is developing RFC 6897 [17] to take advantage of this solution. With applications aware of the available multipath implementation the services can be prioritized, creating a better Quality of Service (QoS) solution for its users. This type of solution is important to allow the applications to specify preferences and control the behavior of the application itself, when reacting to the type of service to be offered.

An interesting application of MPTCP is the a TCP-based multipath streaming scheme, Dynamic MPath-Streaming (DMP-streaming) [44]. This study of audio/video streaming has been motivated by the increasing availability of multiple paths between end hosts. Usually the retransmission mechanisms of the TCP protocol violates real-time requirement for multimedia streaming, due to packet loss and re-ordering at the receiving end. DMP-streaming dynamically distributes packets over multiple paths, it was created a analytical model to evaluate its performance on wired networks, it does not need probing the available network bandwidths and does not exploit error recovery nor rate adaptation. In this study is shown that a TCP-based multipath streaming scheme achieves satisfactory performance when the throughput has at least 1.6 times the video bitrate, with a few seconds of startup delay.

Chapter 4

TestBed

This chapter describes an overview of the different architecture setups for the testbed implementation. It also presents a description of the key components of this project.

This project aims to experiment with the use of different implementation and simulation technique models existent for MPTCP, to identify the limitations and potentials of the different approaches to this protocol. 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 3G4G connection through a network operator.

This testbed can be used to assert the viability of a possible application on Técnico Lisboa network.

4.1 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.

4.1.1 Multihoming Solutions

The first scenario consists on a multihoming solution implementation of MPTCP, using two network interfaces, WLAN and Ethernet, as show in Figure 4.1. 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.

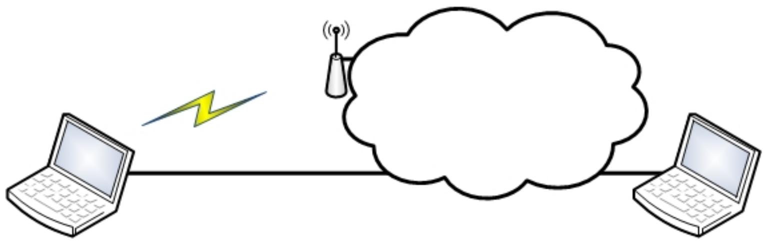


Figure 4.1: Multihoming MPTCP with WiFi and Ethernet Connection

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 4.2.

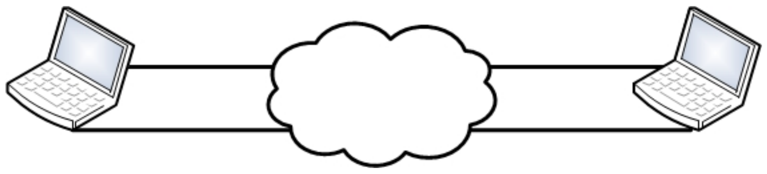


Figure 4.2: Multihoming MPTCP with two Ethernet Connections

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.

4.1.2 Subflow Analysis

The third scenario, depicted in Figure 4.3 and 4.4 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.

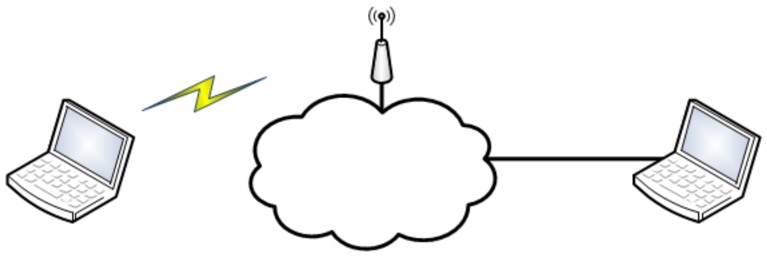


Figure 4.3: MPTCP Subflow Analysis WiFi

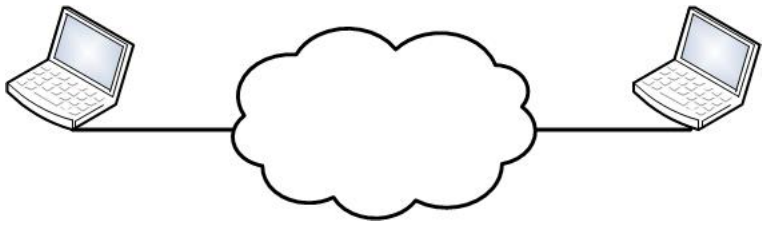


Figure 4.4: MPTCP Subflow Analysis Ethernet

We can capture the individual packets on each subflow in order to analyze the data transferred and draw conclusions about MPTCP functionality and efficiency. The capture of the packets will be performed at the endpoints resorting to the use of the Wireshark tool, with MPTCP support package installed.

This scenario also provides a direct comparison between MPTCP and TCP protocols.

4.1.3 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 4.5 and 4.6. This scenario consists on using two computers with MPTCP enabled implementation, computers connected to two network, and two others using regular TCP, computers connected to one network.

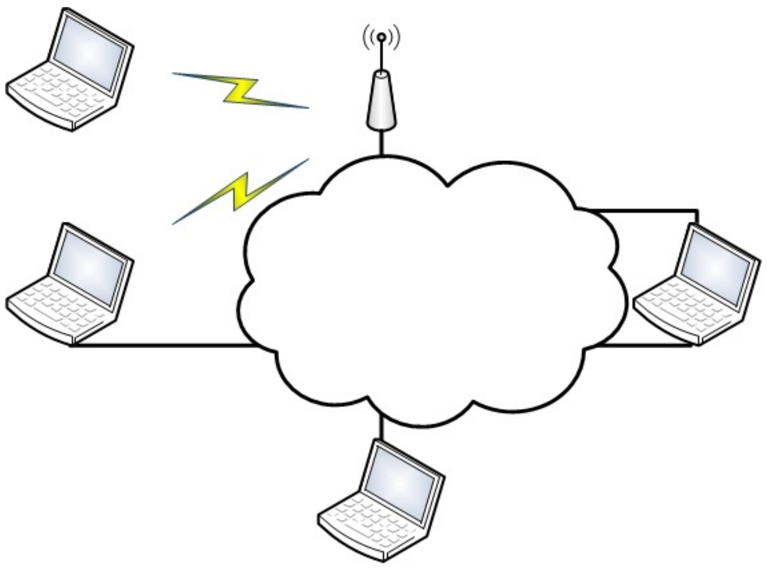


Figure 4.5: TCP and MPTCP Concurrent Scenarios with Ethernet and WiFi Connections

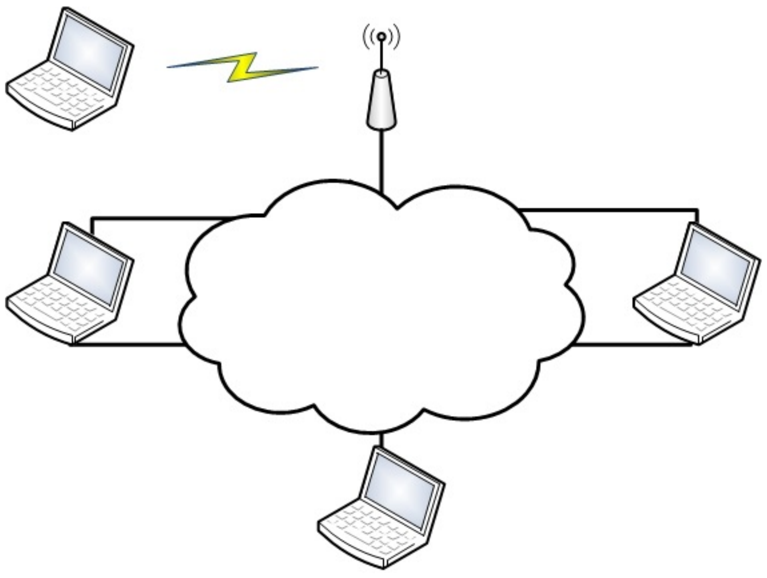


Figure 4.6: TCP and MPTCP Concurrent Scenarios with Ethernet Connections

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.

4.2 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 android operative system allows us to develop in a highly configurable and heavily supported platform.

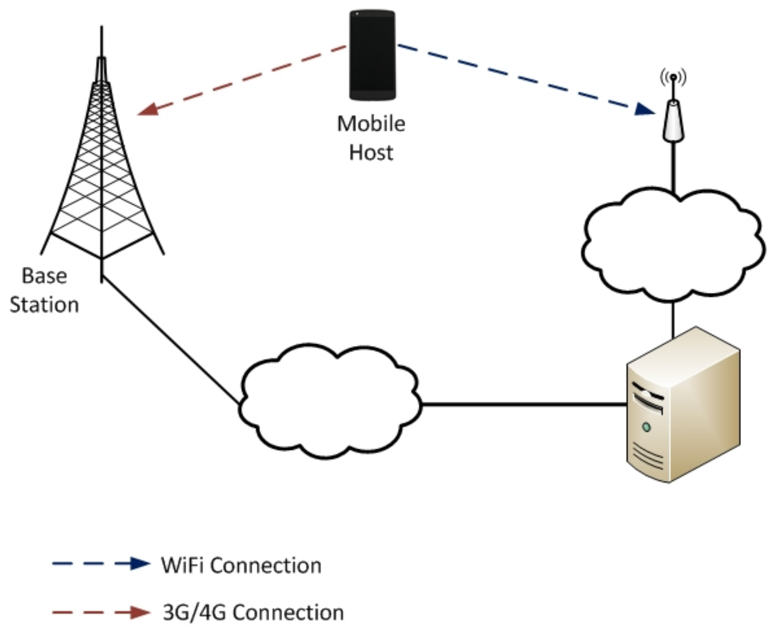


Figure 4.7: Mobile Device Scenario

To evaluate the throughput and handover we will use 3G/4G and WLAN interfaces, as shown in Figure 4.7. This scenario allows to evaluate the handover between networks and the resilience of the connection.

The developed testbed could also be implemented resorting to a virtual machine with multiple VLAN's. The VLAN's could be used to simulate the use of multiple network interfaces.

Chapter 5

Implementation

This chapter describes the different phases perform to achieve the MPTCP testbed implementation. The objective behind the testbed is to evaluate MPTCP performance in comparison to TCP in similar scenarios. 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. In order to modify the endpoints, there are different Linux Kernel implementations available with MPTCP capability. In this case we will choose the better solution in order to implement our testbed. This chapter 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 4. And finally it describes the tests used to evaluate those scenarios, knowing that the endpoints have the capability of supporting MPTCP connections.

5.1 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.

5.1.1 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 [45], on the laptops to use them as concurrent MPTCP clients. But the these devices do not support multihoming 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.

5.1.2 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 [45] 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^{TM} 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 4.1.3, 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.

5.1.3 Mobile Phone

The mobile client is a LG Nexus 5 with Android 4.4. We installed the MultiPath TCP - Linux Kernel Implementation [45] 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.

5.2 MPTCP Configuration

As stated before, desktops computers are using MPTCP enabled kernels. Now it will be explained how to configure the MPTCP options.

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

To modify the kernel parameters in order to configure the system variables at runtime, is used the ***sysctl*** interface. Below you can find the available options and their meaning.

net.mptcp.enabled

This option enables the use of MPTCP when it is set with the value 1. In order to disable just set the variable with the value 0.

net.mptcp.mptcp_syn_retries

This option configures the number of retransmitted SYN with the MP_CAPABLE option. After sending the number of defined retransmission the MP_CAPABLE option will not be sent on the SYN. The default value is 3. This option exists to handle middleboxes that drop SYNs with unknown TCP options.

net.mptcp.mptcp_checksum

This option enables the use of MPTCP checksum when set with the value 1. In order to disable just set the variable with the value 0.

NOTE Both sides (client and server) have to be disable not to use checksum.

net.mptcp.mptcp_ndiffports

This option configures the number of subflows desired to create across the same pair o IP addresses. The value can be set, if greater than 1.

net.mptcp.mptcp_path_manager

Path manager is a modular structure that allows to choose between compiled path-managers. This structure is needed to allow the creation of new subflows, also to advertise alternative IP addresses in the ADD_ADDR option. This option has three possible values:

default

If the path manager is set with this variable it will accept passive creation of new subflows. But the host will not create new subflows and will not announce the available IP addresses.

fullmesh

If the path manager is set with this variable it will create a full-mesh between the available subflows and all addresses.

ndiffports

If the path manager is set with this variable it will create the number of subflows defined in the **net.mptp.mptcp_ndiffports**.

The system can be configured to use multiple interfaces, multiple subflows per connection or just fallback to regular TCP. The options above are used by the implemented version, mentioned in section 5.1.2. During the development of this project there have been new version releases, since MPTCP still is an experimental project.

The Mobile solution is configured to support MPTCP environments, but at the time it does not offer configurable options, neither for the creation of different subflows nor multihoming support.

5.3 Network Routing

In conventional scenarios, 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.

The Listing 5.1 and 5.2 are an example of a generic routing policy for two Ethernet connections, on a host using MPTCP.

Listing 5.1: Example Ethernet Interface Configuration, eht0 and eth1

```
ifconfig eth0 192.168.1.2 netmask 255.255.255.0 gw 192.168.1.1 dev eth0
ifconfig eth1 192.168.2.2 netmask 255.255.255.0 gw 192.168.2.1 dev eth1
```

Assuming that we have two Ethernet connections, eth0 and eth1 with the configuration shown in Listing 5.1. We need to create a routing table for each interface. This is achieved by using ***ip-rule***, the routing policy database management of Linux.

Afterwards we need to configure the routing tables created, Listing 5.2. Finally, after configuring the tables we need to set the default route to use for the normal Internet traffic.

Listing 5.2: Routing Configuration Example two Ethernet Connections

```
# Creation of two different routing tables.
ip rule add from 192.168.1.2 table 1
ip rule add from 192.168.2.2 table 2

# Configure the routing tables.
ip route add 192.168.1.0/24 dev eth0 scope link table 1
ip route add default via 192.168.1.1 dev eth0 table 1

ip route add 192.168.2.0/24 dev eth1 scope link table 2
ip route add default via 192.168.2.1 dev eth1 table 2

# Default route for normal internet-traffic
ip route add default scope global nexthop via 192.168.1.1 dev eth0
```

First we configure all the traffic directed for the local network, in the example 192.168.1.0/24, not to use the gateway. Then we need to redirect the traffic destined for an outside the local network to be routed by a specific gateway, in the example 192.168.1.2.

After completing the routing policies configurations we will obtain a routing policy database, as the one shown in Listing 5.3.

Listing 5.3: Example the Result Routing Configuration, two Ethernet

```
0:      from all lookup local
32764:  from 192.168.2.2 lookup 2
32765:  from 192.168.1.2 lookup 1
32766:  from all lookup main
32767:  from all lookup default
```

The routing policy database is now complete with the three default routing tables used by the Linux kernel, which are the local, main and default tables. An the new tables for each interface in use.

The Listing 5.4 and 5.5 are an example of a generic routing configuration for an Ethernet with Wireless Local Area Network (WLAN) connections, on a host using MPTCP. This is a similar configuration as the one before, the only difference is that we configure an WLAN connection on the second interface.

Listing 5.4: Example Wireless Interface Configuration, wlan0

```
ifconfig wlan0 192.168.3.2 netmask 255.255.255.0 gw 192.168.3.1 dev wlan0
```

Listing 5.5: Routing Configuration Example Ethernet and WLAN Connections

```
# Creation of two different routing tables.
ip rule add from 192.168.1.2 table 1
ip rule add from 192.168.3.2 table 2

# Configure the routing tables.
ip route add 10.1.1.0/24 dev eth1 scope link table 1
ip route add default via 10.1.1.1 dev eth0 table 1

ip route add 192.168.3.0/24 dev wlan0 scope link table 2
ip route add default via 192.168.3.1 dev wlan0 table 2

# Default route for normal internet-traffic
ip route add default scope global nexthop via 192.168.1.1 dev eth0
```

Whenever an interface becomes available or changes configurations, the MultiPath TCP kernel requires these procedures to occur. 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. The script uses a static implementation using IPv4. 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 Chapter 4.

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

5.4 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 **tcpdump** tool manages to capture the packets sent over the available connections. Then, with **wireshark**, we proceeded to perform a offline analysis of the packets transfered. This combination of tools allows us to measure the performance of the protocols by evaluating the throughput of the data transfered, the load balancing between the flows, the handover between connections, the connection establishment time and the packets overhead.

With **iperf** we were able to obtain the network performance and evaluate the MPTCP protocol, how it reacts to connection interruptions in one of the paths and how it adapts to failures. Artificial failures were generated through physical intervention on one, or both, of the network interfaces. This tool allows us simulate other users in the same link, by creating a number of clients competing for the server's connection. We were also able to test the protocol reaction to links which have the same upstream and downstream bandwidth.

The **netstat** and **ping** are part of a collection of base networking utilities for Linux, net-tools, that have the capability of displaying all the available network connections, test the reachability of a host on a IP network, and has the ability to measure the RTT's between each packet delivered. These network utilities allows us to distinguish the MPTCP subflows.

The **ss** command utility gets information directly form the kernel, which offers a faster response than netstat. With this utility we can obtain each subflow statistics, because it offers a larger collection of information than the netstat.

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

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

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.

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

Chapter 6

Methodology and Evaluation

The main objective of this testbed is to compare the use of MPTCP and TCP protocols. The testbed was developed to evaluate the use of the MPTCP protocol on various type of network communications, in order to assess the viability of deploying the protocol on Técnico Lisboa internal network .

This chapter describes the methodology and evaluation of the developed testbed, to ensure the efficiency of this protocol.

In the Section 6.1 we begin by explaining the tests conducted and their scenarios, performed over the detailed architecture in Chapter 4. Section 6.2 describes the metrics used to produce an objective analysis of the MPTCP protocol implementation. The result of the tests carried out, using the tools described in Section 5.4, can be found on Section 6.3.

6.1 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 and the specifications about the hardware can be found on Section 5.1. As described in Section 5.2 we modify the system to support the desired scenario tests.

The Mobile Client needs to install the enabled MPTCP version of the Android 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, as stated before the conventional wireless and wired scenarios and mobile devices, described in Chapter 4.

The first context allows us to evaluate and 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.

6.1.1 Scenario A - Multihoming MPTCP with WiFi and Ethernet Connections

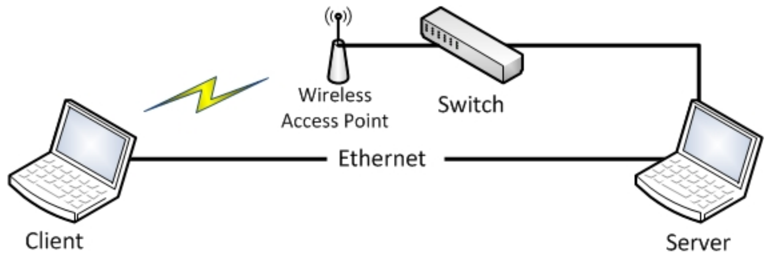


Figure 6.1: Test Scenario A - Multihomed with WiFi and Ethernet Connections

For this scenario we used two desktop computers, with MPTCP enabled distribution of Ubuntu Saucy. Using two connections, the first being a wireless connection using a Wireless USB Adapter connected to the a MikroTik Router. The second a RJ45 cable connecting the two computers directly. More specifications can be found on Section 5.1.

This first test scenario collects information of a MPTCP multihoming implementation using Ethernet and WLAN connections, as shown in Figure 6.1. The system should be configured to use multiple interfaces, as described in the Section 5.2. In this scenario the computers were configured to use two different network links, one for each network interface, similar to the routing examples described in Section 5.3. The Server is connected to the Switch AP in order to simulate a wireless access point connection in the network. The Client is connect to the Server via Ethernet, direct link, and via Wireless, through the Switch AP.

To obtain adequate data from the use of multihoming solution with MPTCP, this scenario is divided in eight sub scenarios.

Scenario A.1 This test allows us to gather information on the load balancing, performance and throughput when using a Ethernet and WiFi connections, when there are no other users on the network.

Scenario A.2 This scenario tests the protocol's reaction when the Ethernet link fails, providing information about the response time to recover and evaluate the resilience of the network.

Scenario A.3 This scenario is similar to the previous sub scenario, Scenario A.2, but in this case the Ethernet link is stable and the WiFi link fails.

Scenario A.4 This test scenario gathers information on the handover between different network connections. In the beginning of the test the Client is using a WiFi connection but then suddenly changes to Ethernet, closing the previous connection.

Scenario A.5 This test scenario is similar to the previous, Scenario A.4, but in this scenario the Client begins the connection using Ethernet and then changes to WiFi.

Scenario A.6 In this test the Client loses connection on both interfaces and recovers. This test allows us to collect information on protocol's reaction when it does not have an available interface to redirect the data and how it deals with loss of connection.

Scenario A.7 Tests the protocol’s reaction to links which offer the same upstream and downstream bandwidth, providing routing information and load balancing over the network links.

Scenario A.8 This test scenario simulates other users in the same network link, by creating five clients competing for the server’s connection. This allows us to observe the fairness of the protocol in a congested link.

6.1.2 Scenario B - Multihoming MPTCP with two Ethernet Connection

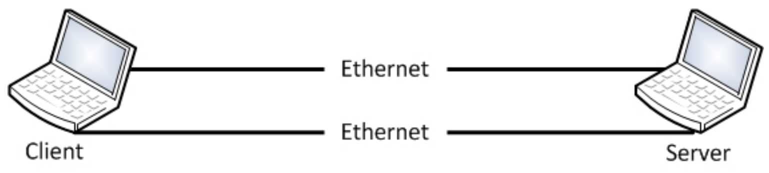


Figure 6.2: Test Scenario B - Multihomed with Two Ethernet Connections

For this scenario we used two desktop computers, with MPTCP enabled distribution of Ubuntu Saucy. Using two connections, both of them by using a RJ45 cable connecting the two computers directly on each interface. More specifications can be found on Section 5.1.

This scenario is similar to the previous scenario 6.1.1, being that it is also a MPTCP multihoming implementation, but in this case it uses two Ethernet connections, as shown in Figure 6.2. The system needs to be configured to use multiple interfaces, as described in the Section 5.2. In this scenario the computers should be configured to use two different network links, one for each network interface, similar to the routing examples on Section 5.3. The Server and Client are directly connected through Ethernet.

To obtain adequate data from the use of MPTCP multihoming solution, this scenario is divided in seven sub scenarios.

Scenario B.1 This test allows us to gather information on the load balancing, performance and throughput when using two Ethernet connections, when there are no other users on the network.

Scenario B.2 This scenario tests the protocol’s reaction when one of the Ethernet link fails, providing information about the response time to recover and evaluate the resilience of the network.

Scenario B.3 This test scenario gathers information on the handover between different network connections. In the beginning of the test the Client is using a network connection but then suddenly changes to another, closing the previous connection.

Scenario B.4 In this test the Client loses connection on both interfaces and recovers. This test allows us to collect information on protocol’s reaction when it does not have an available interface to redirect the data and how it deals with loss of connection.

Scenario B.5 In this test proceed to strangle one of the links, to 10 Mbps, to observe how the protocol sends the network traffic and how it deals with congested links.

Scenario B.6 Tests the protocol's reaction to links which offer the same upstream and downstream bandwidth, providing routing information and load balancing over the network links.

Scenario B.7 This test scenario simulates other users in the same network link, by creating a five clients competing for the server's connection. This allows us to observe the fairness of the protocol in a congested link.

6.1.3 Scenario C - Subflow Analysis

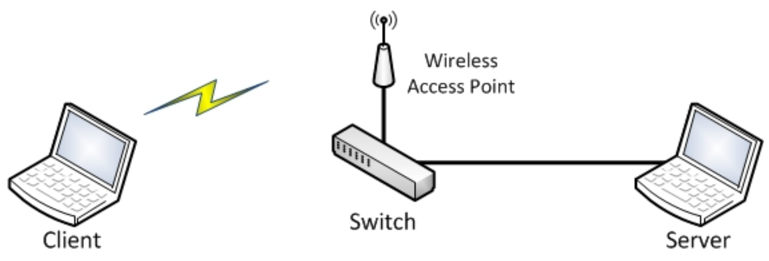


Figure 6.3: Test Scenario C - WiFi Connection

For this scenario we used two desktop computers, with MPTCP enabled distribution of Ubuntu Saucy. Using one Wireless USB Adapter connected to the a MikroTik Router. More specifications can be found on Section 5.1.

This scenario gathers information of the MPTCP subflows implementation, using Ethernet and WiFi connections separately, as shown in Figure 6.3 and Figure 6.4. The system should be configured to use three subflows, these configurations are set using the options described in Section 5.2. In order to obtain adequate data from the use of MPTCP subflows, this scenario is divided in eight sub scenarios.

The first four scenarios gather subflows information while using WLAN connection. The Client has a WLAN interface that connects to the Server through the Switch AP, Figure 6.3.

Scenario C.1 This test allows us to gather information on MPTCP subflows while using a WiFi connection, when there are no other users on the network.

Scenario C.2 In this test the Client loses connection on the network interface. This test allows us to collect information on protocol's reaction to loss of connection, recovery time and evaluate the resilience of the network.

Scenario C.3 Tests the protocol's reaction to links which offer the same upstream and downstream bandwidth, providing routing information and load balancing on the subflows.

Scenario C.4 This scenario simulates other users in the same network link, by creating a five clients competing for the server's connection. This allows us to observe the fairness between the subflows connections and evaluate the response to a congested link.

The last four scenarios gathers subflows information while using Ethernet connection. The Server and Client are directly connected through Ethernet, Figure6.4.

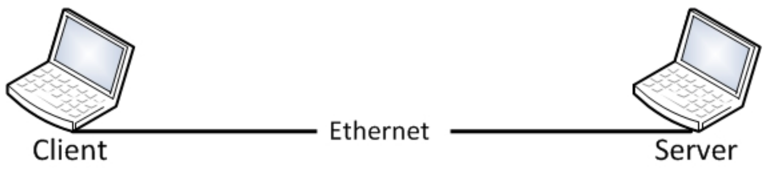


Figure 6.4: Test Scenario C - Ethernet Connection

For this scenario we used two desktop computers, with MPTCP enabled distribution of Ubuntu Saucy. Using one RJ45 cable to connected them directly. More specifications can be found on Section 5.1.

Scenario C.5 This test scenario is similar to the described Scenario C.1. But in this test scenario the Client is connected to the server using the Ethernet interface.

Scenario C.6 This test scenario is similar to the described Scenario C.2. But in this test scenario the Client is connected to the server using the Ethernet interface.

Scenario C.7 This test scenario is similar to the described Scenario C.3. But in this test scenario the Client is connected to the server using the Ethernet interface.

Scenario C.8 This test scenario is similar to the described Scenario C.4. But in this test scenario the Client is connected to the server using the Ethernet interface.

6.1.4 Scenario D - TCP Analysis

This scenario collects the TCP information using the same tests of the previous Scenario C, section 6.1.3. The system needs to be configured to have MPTCP disable, this configurations are set using the options described in Section 5.2. The data collected from using TCP, allows to make a direct comparison between the use of both protocols.

6.1.5 Scenario E - TCP and MPTCP Concurrent Scenarios

For this scenario we used two desktop and two laptop computers. The desktop computers, Client and Server, used the MPTCP enabled distribution of Ubuntu Saucy. The laptops, TCP Client1 and TCP Client2, used the Debian Wheezy distribution.

This scenario uses two different type of connections, WiFi and Ethernet. The desktop, Client, uses a Wireless USB Adapter connected to the server via a MikroTik Router, and a Ethernet connection through a Cisco-Linksys Router. The laptop, TCP Client1 is connected to the server via WiFi through the MikroTik Router. The laptop, TCP Client2 is connected to the server via Ethernet through the Cisco-Lynksys Router. More specifications can be found on Section 5.1.

This test scenario gathers information of a network with both MPTCP and TCP users, as shown in Figure 6.5 and Figure 6.6.

The Server is connected to the Switch AP to simulate a wireless access point in the network. The TCP Client1 connects to the Server via Wireless, through the Switch AP. While the TCP Client2 is

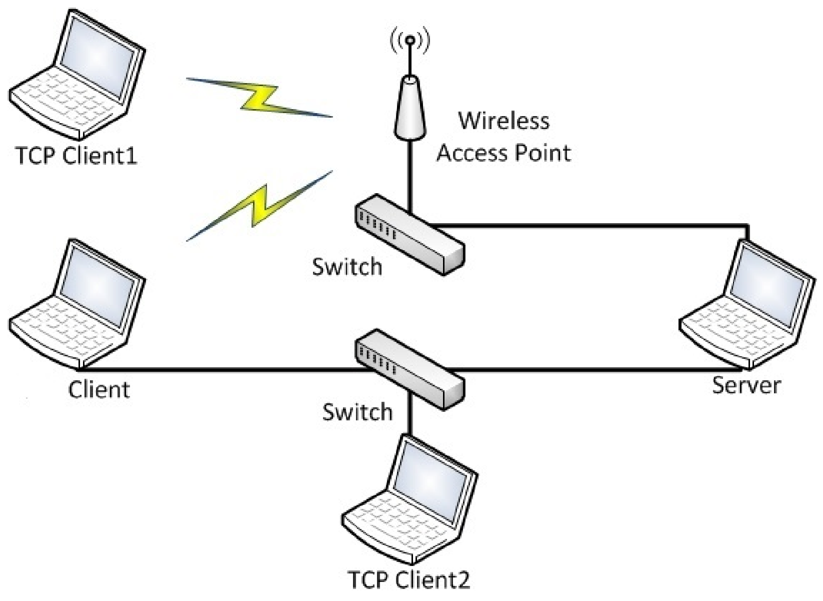


Figure 6.5: Scenario E - Ethernet and WiFi Connections

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.

This test scenario is divided in this scenario is divided in six sub scenarios.

Scenario E.1 In this test scenario the Server and Client use MPTCP enabled configured to use multiple interfaces, as described in the Section 5.2. The Client connects to the Server via Wireless, through the Switch AP. This scenario collects information to evaluate if the MPTCP multihoming is aggressive towards the TCP traffic over the network, Figure 6.5.

Scenario E.2 This scenario is similar to the Scenario E.1, but in the case, the Client is connected to the Server via Ethernet through the Switch, Figure 6.6.

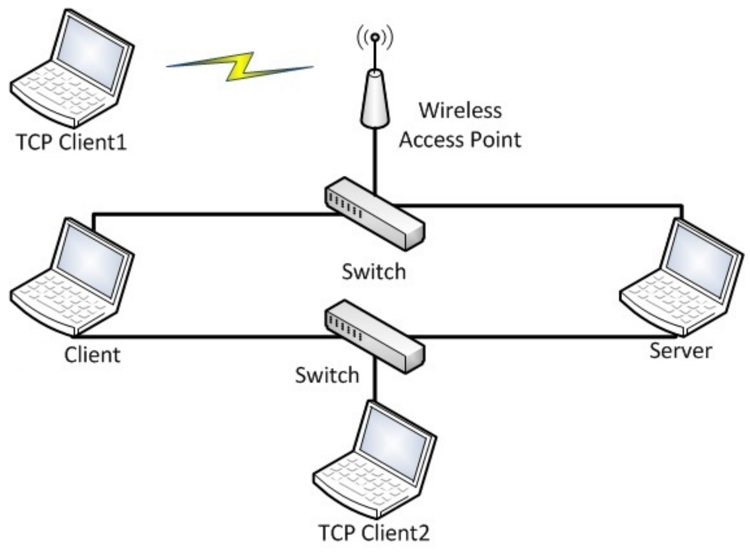


Figure 6.6: Scenario E - Ethernet Connections

This scenario uses two different type of connections, WiFi and Ethernet. The desktop, Client, uses two Ethernet connections to reach the server, the first through the MikroTik Router and the second through the Cisco-Linksys Router. The laptop, TCP Client1 is connected to the server via WiFi through the MikroTik Router. The laptop, TCP Client2 is connected to the server via Ethernet through the Cisco-Lynksys Router. More specifications can be found on Section 5.1.

Scenario E.3 In this test scenario the Server and Client have MPTCP enabled configuration to use three subflows, as described in the Section 5.2. The MPTCP Client connects to the Server via Wireless, through the Switch AP. This scenario collects information to observe how the MPTCP subflows react when there is TCP traffic over the network, Figure 6.5.

Scenario E.4 This scenario is similar to the Scenario E.3 but in this case, the Client is connected to the server using the Ethernet interface, through the Switch, Figure 6.6.

Scenario E.5 This test scenario collects the TCP information using the same test of the Scenario E.3. The Server and Client have the MPTCP configuration disabled, this configurations are set using the options described in Section 5.2. The Client connects to the Server via Wireless, through the Switch AP. This scenario allows us to evaluate regular TCP reaction to traffic sent over the network WLAN, Figure 6.5.

Scenario E.6
This test scenario collects the TCP information using the same test of the Scenario E.4. The Server and Client need to have MPTCP disable, this configurations are set using the options described in Section 5.2. The MPTCP Client connects to the Server via Ethernet, through the Switch. This scenario allows us to evaluate regular TCP reaction to traffic sent over the Ethernet link, Figure 6.6.

6.1.6 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 6.7, based on the architecture described on the Section 4.2. This scenario uses three types of connection, local WiFi connection, campus WiFi connection and 3G/4G connection provided by a commertial 3G/4G Internet Service Provider (ISP).

In 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.

In this scenario the Server and the Mobile User need to have MPTCP enabled solutions. The Server needs to have MPTCP enabled and to be configured to use multiple interfaces, as described in the Section 5.2. 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 5.1.3.

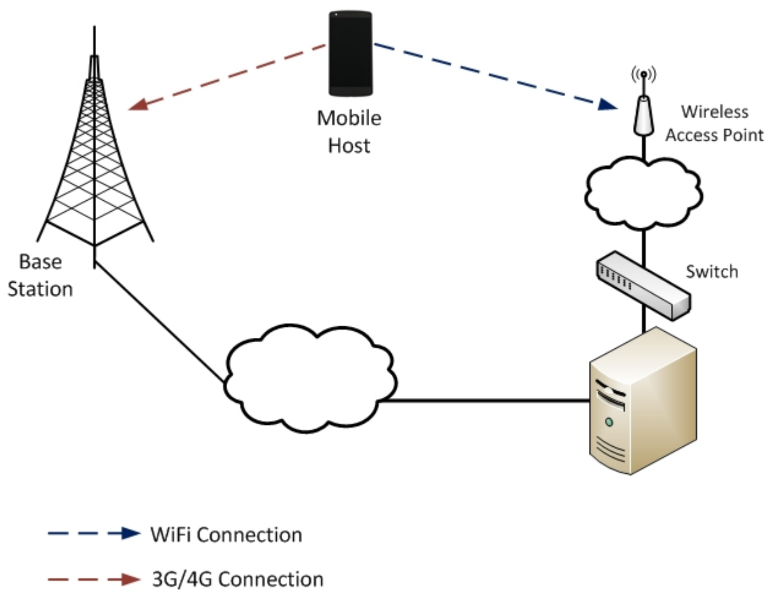


Figure 6.7: Scenario F - Mobile Connection

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.

This scenario is divided in nine sub scenarios, allowing us to gather information of the MPTCP implementation on a smart phone Android, such as performance, throughput and handover between networks.

Scenario F.1 In this scenario we collect information of using MPTCP protocol over a local WiFi connection. In order to do so, the Mobile User was connected to the Server via wireless, through the Switch AP.

The tests carried out in this scenario obtain information of the performance and throughput of the connection. The first test, tests the connection over the local WiFi when there are no other users on the network. The second test evaluates the protocol's reaction to loss of connection, recovery time and the resilience of the network, when the Mobile User loses the WiFi connection. This loss is accomplished through link failure simulation, on the link that connects the Switch AP to the Server. The third test allows us to evaluate the reaction of the protocol to links which offer the same upstream and downstream bandwidth. Lastly theres a test that simulates five clients competing for medium access to the server. Allowing us to observe the fairness between the load balancing carried out in a congested link.

Scenario F.2 In this scenario we collect information of using MPTCP protocol over the WiFi campus connection. In order to do that, the Mobile User was connected to the Server via Wireless, through one of the campus Access Point (AP). The objective of this scenario is to evaluate the protocol's reaction when using a wireless environment with bandwidth restrictions and load generated by several users.

This scenario assesses the same tests described on the Scenario F.1. The loss in this scenario is accomplished by link simulation failure, over the link that connects the Server to Técnico Lisboa internal network.

Scenario F.3 In this scenario we collect information of using MPTCP protocol in over a ISP network. In order to do that, the Mobile User needs to be connected to the Server, using the 3G/4G interface. Taking into account that the server has a Ethernet connection Técnico Lisboa internal network, which allows it to access the Internet. The objective of this scenario is to evaluate the protocol's reaction when used over public networks.

This scenario assesses the same tests described on the Scenario F.1. The loss in this scenario is accomplished by link simulation failure, on the link that connects the Server to Técnico Lisboa internal network.

Scenario F.4 This test scenario gathers information about the handover between mobile to the campus WiFi network. This test is interesting to evaluate the changes that occur when the Mobile User changes from Técnico Lisboa internal network, where there are bandwidth restrictions and load generated by several users, to a public network provided by the ISP.

Scenario F.5 This test scenario is the reverse of the previous test scenario, Scenario F.4. In this test we collect information about the handover between campus WiFi and to mobile network.

Scenario F.6 This test scenario gathers information about the handover between campus WiFi to the local WiFi network. This test is interesting to evaluate the changes that occur when the Mobile User changes from the campus WiFi network, where there bandwidth restrictions and load generated by several users, to the local WiFi network.

Scenario F.7 This test scenario is the reverse of the previous test scenario, Scenario F.6. In this test we collect information about the handover between the local WiFi to the campus WiFi network.

Scenario F.8 This test scenario gathers information about the handover between the local WiFi to the mobile network. This test is interesting to evaluate the changes that occur when the Mobile User changes from the local WiFi network, where there are no other users, to a public network provided by the ISP.

Scenario F.9 This test scenario is the reverse of the previous test scenario, Scenario F.8. In this test we collect information about the handover between the mobile to the local WiFi network.

6.1.7 Scenario G - Mobile Solution TCP

This scenario collects the TCP information using the first three test scenarios described previously on Scenario F. The Server was configured to have MPTCP disabled, this configurations are set using the options described in Section 5.2. The data collected from using TCP, allows to make a direct comparison between the use of both protocols.

6.2 MPTCP 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.

6.3 Results

This section presents the analysis results of the tests. The description of the test scenarios performed can be found in Section 6.1.

Several tests were performed for each type of test scenario described, specifically 10 tests. The test where performed in different times of day. All data was collected using the evaluation tools described on Section 5.4.

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.

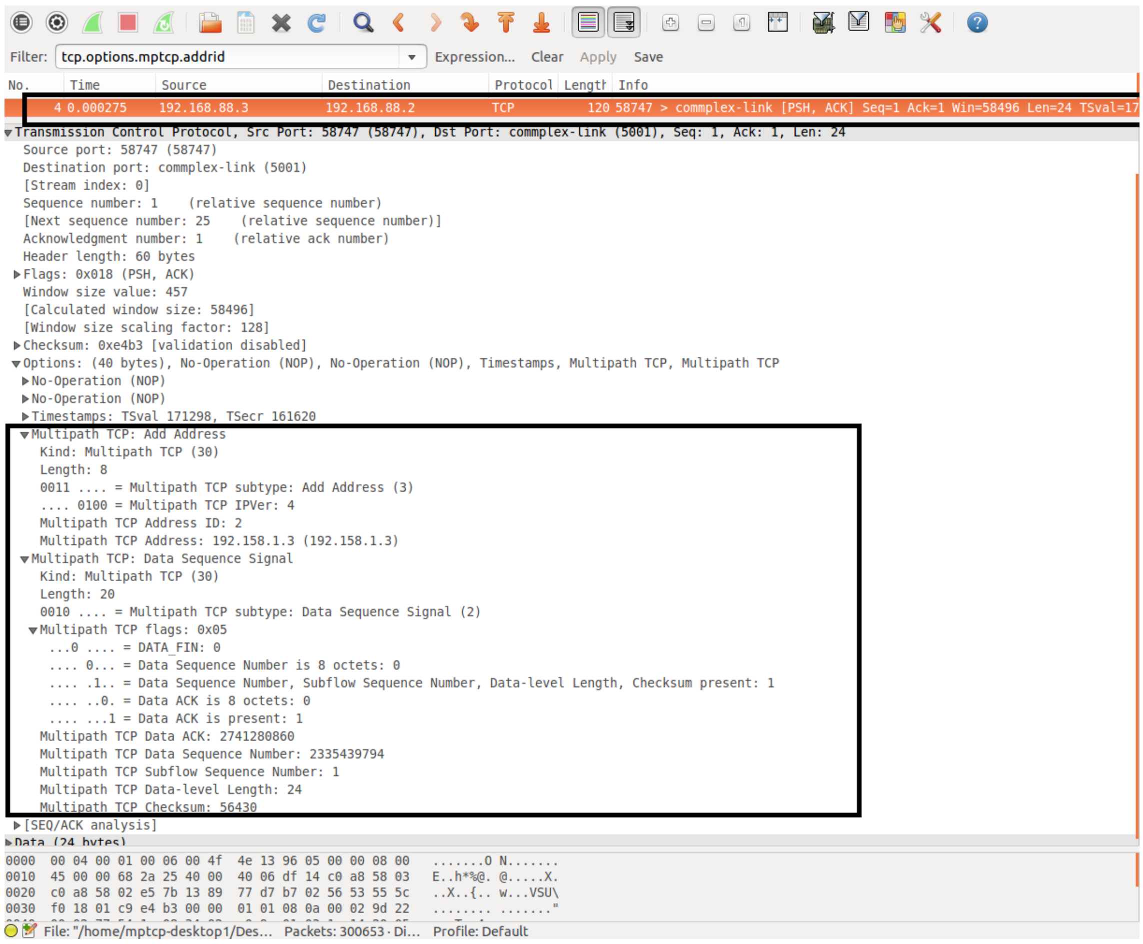
The tests performed in the context of mobile environment were measured for 60 seconds, because a longer connection would entail greater expense, due to the use of 4G networks. In this context the data was collected on Server side, due to limitations of the actual applications for Android with enabled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

The values that will be presented are an average of the several measurements obtained for each scenario.

6.3.1 Wireshark Capture of MPTCP Connection Handling

The Wireshark tool was used to examine the packets captured with the tcpdump tool. Following is presented the screenshots of the most relevant messages of the MPTCP protocol, that are sent to handle the connection, as described in Section 3.3. The MPTCP information is sent in the TCP Options field, as can be seen in Figures 6.8, 6.9 and 6.10.

These images where captured in the Scenario A.1. The Client has the addresses 192.168.88.3 and 192.158.1.3, while the Server has the 192.168.88.2 and 192.158.1.2.



```
Filter: tcp.options.mptcp.addrid          Expression...  Clear  Apply  Save

No.   Time       Source          Destination     Protocol Length Info
    4 0.000275   192.168.88.3    192.168.88.2    TCP         120 58747 > commplex-link [PSH, ACK] Seq=1 Ack=1 Win=58496 Len=24 TSval=17
▼Transmission Control Protocol, Src Port: 58747 (58747), Dst Port: commplex-link (5001), Seq: 1, Ack: 1, Len: 24
  Source port: 58747 (58747)
  Destination port: commplex-link (5001)
  [Stream index: 0]
  Sequence number: 1    (relative sequence number)
  [Next sequence number: 25    (relative sequence number)]
  Acknowledgment number: 1    (relative ack number)
  Header length: 60 bytes
 ▶Flags: 0x018 (PSH, ACK)
  Window size value: 457
  [Calculated window size: 58496]
  [Window size scaling factor: 128]
 ▶Checksum: 0xe4b3 [validation disabled]
 ▼Options: (40 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, Multipath TCP, Multipath TCP
  ▶No-Operation (NOP)
  ▶No-Operation (NOP)
  ▶Timestamps: TSval 171298, TSecr 161620
  ▼Multipath TCP: Add Address
     Kind: Multipath TCP (30)
     Length: 8
     0011 .... = Multipath TCP subtype: Add Address (3)
     .... 0100 = Multipath TCP IPVer: 4
     Multipath TCP Address ID: 2
     Multipath TCP Address: 192.158.1.3 (192.158.1.3)
  ▼Multipath TCP: Data Sequence Signal
     Kind: Multipath TCP (30)
     Length: 20
     0010 .... = Multipath TCP subtype: Data Sequence Signal (2)
   ▼Multipath TCP flags: 0x05
      ...0 .... = DATA_FIN: 0
      .... 0... = Data Sequence Number is 8 octets: 0
      .... .1.. = Data Sequence Number, Subflow Sequence Number, Data-level Length, Checksum present: 1
      .... ..0. = Data ACK is 8 octets: 0
      .... ...1 = Data ACK is present: 1
     Multipath TCP Data ACK: 2741280860
     Multipath TCP Data Sequence Number: 2335439794
     Multipath TCP Subflow Sequence Number: 1
     Multipath TCP Data-level Length: 24
     Multipath TCP Checksum: 56430
 ▶[SEQ/ACK analysis]
▶Data (24 bytes)
0000  00 04 00 01 00 06 00 4f  4e 13 96 05 00 00 08 00   .......O N.......
0010  45 00 00 68 2a 25 40 00  40 06 df 14 c0 a8 58 03   E..h*%@. @.....X.
0020  c0 a8 58 02 e5 7b 13 89  77 d7 b7 02 56 53 55 5c   ..X..{.. w...VSU\
0030  f0 18 01 c9 e4 b3 00 00  01 01 08 0a 00 02 9d 22   ........ ......."
File: "/home/mptcp-desktop1/Des...   Packets: 300653 · Di...   Profile: Default
```

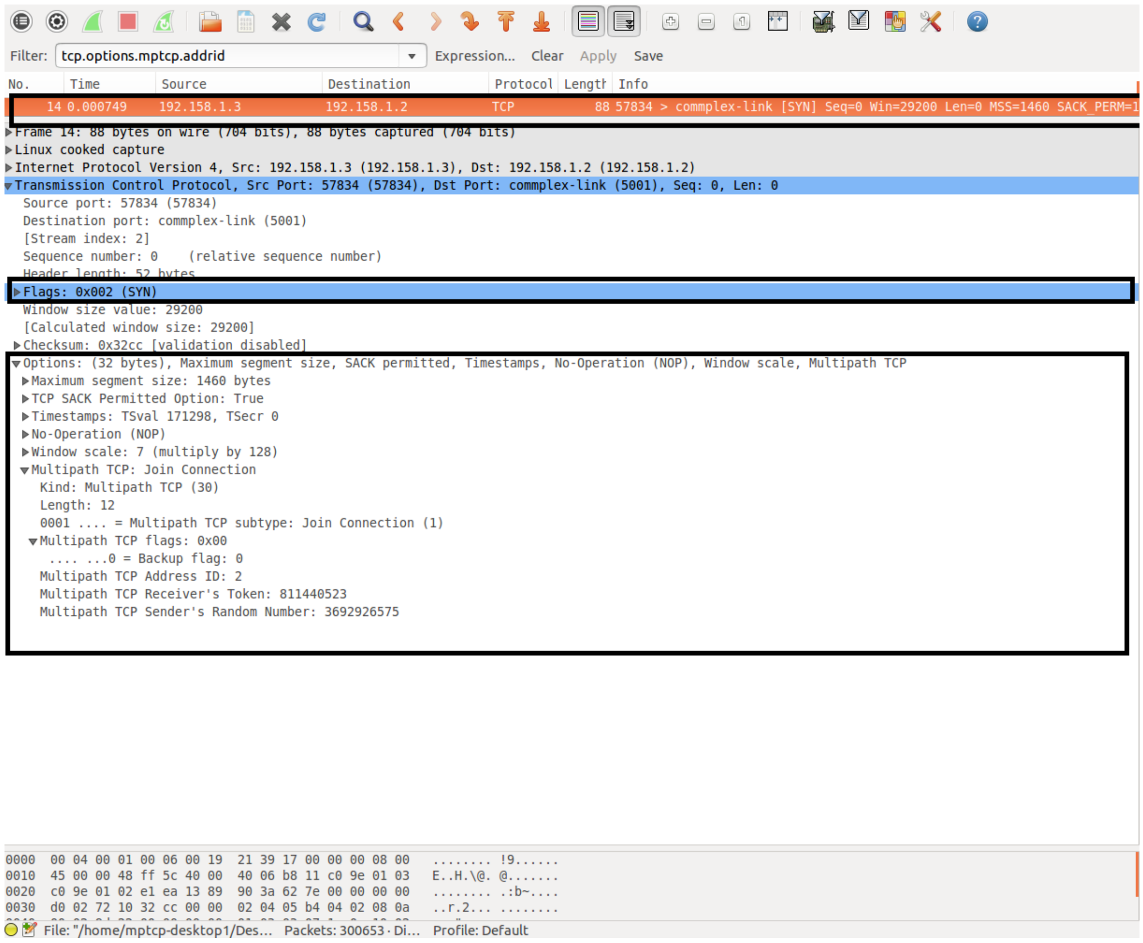
Figure 6.8: Wireshark screenshot showing MPTCP message for Beginning a connection

Figure 6.8 shows the Add Address MultiPath TCP option to establish a new connection. A host sends the Add Address, signal ADD_ADDRESS, to announce additional addresses in which it can be reached. In order to identify the new connection the protocol combines the new address with the address id. As it can be seen in the capture, the information exchange is carried out by the Server, informing the Client that it contains another additional address, which belongs to the other interface available, it also sends the protocol address id which is 2.

Figure 6.9 shows the Join Connection MPTCP option, to allow the host to create new subflows. A host sends Join Connection, flag MP_JOIN, to establish a new subflow with another host, after the add address and the protocol id already been exchanged. In the capture the Client exchanges the identification tokens and the address id in order to create a new subflow with the Server.

Figure 6.10 shows the MPTCP sending the DATA_FIN option selected to initiate the closing of the connection. The protocol also sets the TCP flag with FIN, to close the connection, in case the TCP options are dropped. In the capture Server is sending the DATA_FIN on one of the subflows being used, to close all the subflows.

With these three images we can see the MPTCP protocol is very similar to TCP protocol. And that



```
No.   Time        Source          Destination     Protocol Length Info
   14 0.000749    192.158.1.3     192.158.1.2     TCP          88 57834 > commplex-link [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
▶Frame 14: 88 bytes on wire (704 bits), 88 bytes captured (704 bits)
▶Linux cooked capture
▶Internet Protocol Version 4, Src: 192.158.1.3 (192.158.1.3), Dst: 192.158.1.2 (192.158.1.2)
▼Transmission Control Protocol, Src Port: 57834 (57834), Dst Port: commplex-link (5001), Seq: 0, Len: 0
   Source port: 57834 (57834)
   Destination port: commplex-link (5001)
   [Stream index: 2]
   Sequence number: 0    (relative sequence number)
   Header length: 52 bytes
  ▶Flags: 0x002 (SYN)
   Window size value: 29200
   [Calculated window size: 29200]
  ▶Checksum: 0x32cc [validation disabled]
  ▼Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, Multipath TCP
    ▶Maximum segment size: 1460 bytes
    ▶TCP SACK Permitted Option: True
    ▶Timestamps: TSval 171298, TSecr 0
    ▶No-Operation (NOP)
    ▶Window scale: 7 (multiply by 128)
    ▼Multipath TCP: Join Connection
       Kind: Multipath TCP (30)
       Length: 12
       0001 .... = Multipath TCP subtype: Join Connection (1)
      ▼Multipath TCP flags: 0x00
         .... ...0 = Backup flag: 0
       Multipath TCP Address ID: 2
       Multipath TCP Receiver's Token: 811440523
       Multipath TCP Sender's Random Number: 3692926575

0000  00 04 00 01 00 06 00 19  21 39 17 00 00 00 08 00   ........ !9......
0010  45 00 00 48 ff 5c 40 00  40 06 b8 11 c0 9e 01 03   E..H.\@. @.......
0020  c0 9e 01 02 e1 ea 13 89  90 3a 62 7e 00 00 00 00   ........ .:b~....
0030  d0 02 72 10 32 cc 00 00  02 04 05 b4 04 02 08 0a   ..r.2... ........

File: "/home/mptcp-desktop1/Des...  Packets: 300653 · Di...  Profile: Default
```

Figure 6.9: Wireshark screenshot showing MPTCP message for creating a new Subflow

it sends the information needed in the Option field of the TCP protocol. To ensure the routing and compatibility the MPTCP also sends the relevant information over the TCP fields, in case the middleboxes scattered over the network drop the TCP options field.

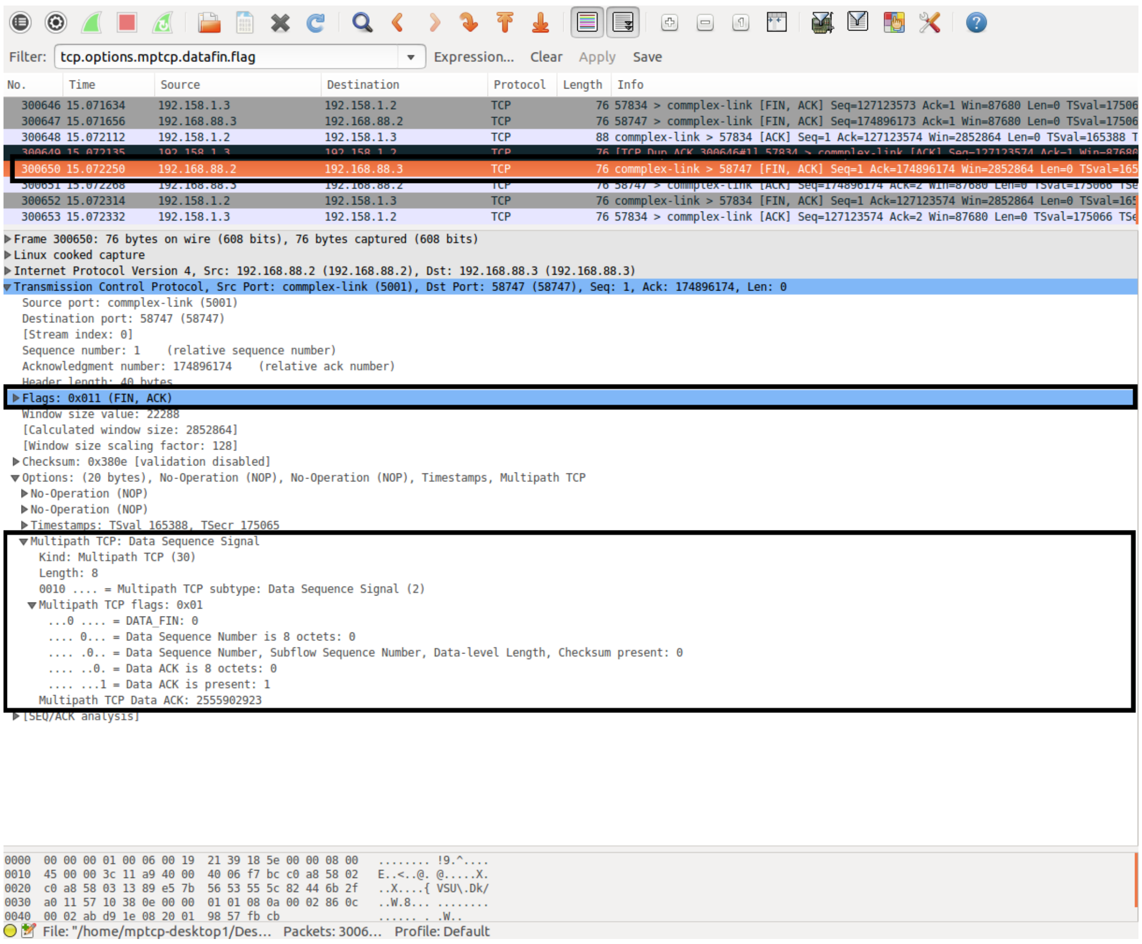
6.3.2 Throughput Measurements

In this subsection is described the performance results of both protocols in different test scenarios that were deployed, previously described in the Section 6.1. As stated before, the values presented are an average of several measurements obtained on each test scenario.

The Table 6.1 compares the performance of both protocols MPTCP and TCP, in the context of conventional wireless and wired connection scenarios, when there is no congestion on the transfer medium in use. As we will show below, 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 6.1, the connection of MPTCP Subflow has a lower throughput than the one obtain in TCP, scenarios C.1 and D.1 for WiFi, and scenarios C.5 and D.5 for Ethernet. This can be explained with the overhead created for each subflow. In this case the MPTCP connection was configured to use three subflows.

Comparing the MPTCP Scenario A.1 WiFi interface, it offers higher bandwidth than the similar one



Filter: tcp.options.mptcp.datafin.flag Expression... Clear Apply Save

```
No.    Time       Source          Destination     Protocol  Length  Info
300646 15.071634  192.158.1.3     192.158.1.2     TCP       76 57834 > commplex-link [FIN, ACK] Seq=127123573 Ack=1 Win=87680 Len=0 TSval=17506
300647 15.071656  192.168.88.3    192.168.88.2    TCP       76 58747 > commplex-link [FIN, ACK] Seq=174896173 Ack=1 Win=87680 Len=0 TSval=17506
300648 15.072112  192.158.1.2     192.158.1.3     TCP       88 commplex-link > 57834 [ACK] Seq=1 Ack=127123574 Win=2852864 Len=0 TSval=165388 T
300649 15.072135  192.158.1.3     192.158.1.2     TCP       76 [TCP Dup ACK 300646#1] 57834 > commplex-link [ACK] Seq=127123574 Ack=1 Win=87680
300650 15.072250  192.168.88.2    192.168.88.3    TCP       76 commplex-link > 58747 [FIN, ACK] Seq=1 Ack=174896174 Win=2852864 Len=0 TSval=165
300651 15.072268  192.168.88.3    192.168.88.2    TCP       76 58747 > commplex-link [ACK] Seq=174896174 Ack=2 Win=87680 Len=0 TSval=175066 TSe
300652 15.072314  192.158.1.2     192.158.1.3     TCP       76 commplex-link > 57834 [FIN, ACK] Seq=1 Ack=127123574 Win=2852864 Len=0 TSval=165
300653 15.072332  192.158.1.3     192.158.1.2     TCP       76 57834 > commplex-link [ACK] Seq=127123574 Ack=2 Win=87680 Len=0 TSval=175066 TSe
```

```
▶Frame 300650: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
▶Linux cooked capture
▶Internet Protocol Version 4, Src: 192.168.88.2 (192.168.88.2), Dst: 192.168.88.3 (192.168.88.3)
▼Transmission Control Protocol, Src Port: commplex-link (5001), Dst Port: 58747 (58747), Seq: 1, Ack: 174896174, Len: 0
   Source port: commplex-link (5001)
   Destination port: 58747 (58747)
   [Stream index: 0]
   Sequence number: 1    (relative sequence number)
   Acknowledgment number: 174896174    (relative ack number)
   Header length: 40 bytes
  ▶Flags: 0x011 (FIN, ACK)
   Window size value: 22288
   [Calculated window size: 2852864]
   [Window size scaling factor: 128]
  ▶Checksum: 0x380e [validation disabled]
  ▼Options: (20 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, Multipath TCP
    ▶No-Operation (NOP)
    ▶No-Operation (NOP)
    ▶Timestamps: TSval 165388, TSecr 175065
    ▼Multipath TCP: Data Sequence Signal
       Kind: Multipath TCP (30)
       Length: 8
       0010 .... = Multipath TCP subtype: Data Sequence Signal (2)
      ▼Multipath TCP flags: 0x01
         ...0 .... = DATA_FIN: 0
         .... 0... = Data Sequence Number is 8 octets: 0
         .... .0.. = Data Sequence Number, Subflow Sequence Number, Data-level Length, Checksum present: 0
         .... ..0. = Data ACK is 8 octets: 0
         .... ...1 = Data ACK is present: 1
       Multipath TCP Data ACK: 2555902923
  ▶[SEQ/ACK analysis]
```

```
0000  00 00 00 01 00 06 00 19  21 39 18 5e 00 00 08 00   ........ !9.^....
0010  45 00 00 3c 11 a9 40 00  40 06 f7 bc c0 a8 58 02   E..<..@. @.....X.
0020  c0 a8 58 03 13 89 e5 7b  56 53 55 5c 82 44 6b 2f   ..X....{ VSU\.Dk/
0030  a0 11 57 10 38 0e 00 00  01 01 08 0a 00 02 86 0c   ..W.8... ........
0040  00 02 ab d9 1e 08 20 01  98 57 fb cb               ...... . .W..
```

File: "/home/mptcp-desktop1/Des... Packets: 3006... Profile: Default

Figure 6.10: Wireshark screenshot showing MPTCP message for Closing the connection

using TCP, Scenario D.1. The same occurs when using the Ethernet interfaces, MPTCP Scenario A.1 Ethernet and TCP Scenario D.5. Through all obtained results, MPTCP showed a higher throughput, in multihoming scenarios with one subflow per network interface, than TCP for the same interface.

The Table 6.2shows the bandwidth capability of each network interface used in the context of mobile scenario. Table 6.2 shows that MPTCP protocol have better performance, Scenario F.2 and F.3, than the similar TCP ones, G.2 and G.3.

As it can be observed the MPTCP throughput in the Local WiFi network, Scenario F.1, is actually lower than in the TCP case, Scenario G.1. The lower throughput obtained in the Scenario F.1 can be considered a random event, since WiFi connections are subject to interference caused by networks concurring in the same channel. The IEEE 802.11 b/g WiFi singal occupies five channels in the 2.4GHz band frequency, this frequency only supports three channels without overlapping, to allocate the various WLAN networks. This can cause collisions in the used WLAN, thus lowering its throughput, due to other networks being used in the same channel. This interference can be seen on the Local WiFi Network link represented in Figures 6.19 and 6.17.

Table 6.3 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.

Wireless And Wired Connections			
Test Scenario	Protocol	Interface	Throughput (Mbps)
Scenario A.1	MPTCP	WiFi	47.8
		Ethernet	98.2
		Total	146
Scenario B.1	MPTCP	Ethernet	98.1
		Ethernet	98.1
		Total	196.2
Scenario C.1	MPTCP	WiFi	40.7
Scenario C.5	MPTCP	Ethernet	92.1
Scenario D.1	TCP	WiFi	43.8
Scenario D.5	TCP	Ethernet	94.1

Table 6.1: Throughput of Wireless And Wired Connections without Network Congestion

Mobile Environment			
Test Scenario	Network	Protocol	Throughput (Mbps)
Scenario F.1	Local WiFi	MPTCP	37.6
Scenario F.2	Campus WiFi	MPTCP	7.8
Scenario F.3	ISP Network	MPTCP	9.4
Scenario G.1	Local WiFi	TCP	43.5
Scenario G.2	Campus WiFi	TCP	2.5
Scenario G.3	ISP Network	TCP	8.6

Table 6.2: Throughput Measurements of Mobile Interfaces

When using Ethernet network links the MPTCP has shown to be fair to TCP users over the same network link, Table 6.3. 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 operative system implementation of the protocol.

Figure 6.11 shows the test Scenario B.5, to demonstrate how MPTCP acts when faced with a strangled link. As observed the throughput sent on the interface with a strangled connection the protocol tries to use all the bandwidth available in the link, thus increasing the overall throughput of the connection.

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. However, when using MPTCP with multiple subflows this can not be assured, due to the overhead that each subflow introduces in the connection. The higher the number of subflows, the greater the overhead introduced.

In congested environments, MPTCP is fair to TCP connections. However, this can not be assured when the congestion occurs on wireless communications.

Wireless And Wired Connections				
Test Scenario	User	Protocol	Interface	Throughput (Mbps)
Scenario E.1	Client	MPTCP	WiFi	33.2
			Ethernet	89.4
			Total	122.6
	TCP Client1	TCP	WiFi	2.1
	TCP Client2	TCP	Ethernet	94.1
Scenario E.2	Client	MPTCP	Ethernet	97.1
			Ethernet	77.1
			Total	171.2
	TCP Client1	TCP	WiFi	20.6
	TCP Client2	TCP	Ethernet	92.3
Scenario E.3	Client	MPTCP	WiFi	36.9
	TCP Client1	TCP	WiFi	0.2
Scenario E.4	Client	MPTCP	Ethernet	92.8
	TCP Client2	TCP	Ethernet	94.1
Scenario E.5	Client	TCP	WiFi	35.5
	TCP Clien1	TCP	WiFi	0.4
Scenario E.6	Client	TCP	Ethernet	94.1
	TCP Clien2	TCP	Ethernet	94.1

Table 6.3: Throughput of Wireless And Wired Connections in Congested Networks

6.3.3 Recovery Time

In this subsection is compared how both protocols react to network failure, as described on the scenarios in Section 6.1.The failures were cause through physical intervention. The values obtained present an average of the several measurements obtained on each test scenario.

Table 6.4 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 6.4 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.2, A.3, A.6, B.2 and B.3. In the case when MPTCP is using subflows it recovers much faster than MPTCP multihomed and TCP,

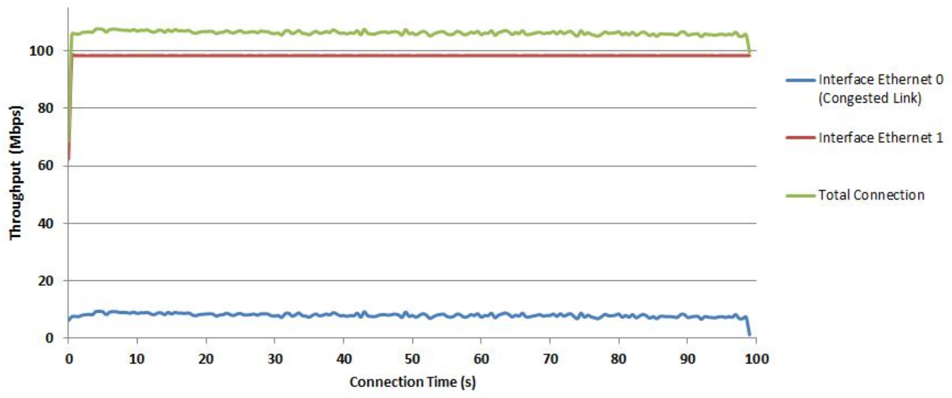


Figure 6.11: MPTCP with a Strangled Link Connection

Wireless And Wired Connections				
Test Scenario	Protocol	Interface	Link Failure	Recovery Time (s)
Scenario A.2	MPTCP	WiFi	Yes	15
		Ethernet	No	-
Scenario A.3	MPTCP	WiFi	No	-
		Ethernet	Yes	15
Scenario A.6	MPTCP	WiFi	Yes	15.5
		Ethernet	Yes	15
Scenario B.2	MPTCP	Ethernet	Yes	13
		Ethernet	No	-
Scenario B.4	MPTCP	Ethernet	Yes	12.5
		Ethernet	Yes	13
Scenario C.2	MPTCP	WiFi	Yes	6
Scenario C.6	MPTCP	Ethernet	Yes	4.5
Scenario D.2	TCP	WiFi	Yes	15.5
Scenario D.6	TCP	Ethernet	Yes	13

Table 6.4: Recovery Time of Wireless And Wired Connections

this occurs 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.1	Local WiFi	MPTCP	Yes	12.5
Scenario F.2	Campus WiFi	MPTCP	Yes	15.5
Scenario F.3	ISP Network	MPTCP	Yes	11
Scenario G.1	Local WiFi	TCP	Yes	13.5
Scenario G.2	Campus WiFi	TCP	Yes	32.5
Scenario G.3	ISP Network	TCP	Yes	29

Table 6.5: Recovery Time of Mobile Connections

Table 6.5 shows the time needed for the connection to recover from a failure, in the context of mobile scenarios. As shown in Table 6.5 the MPTCP is faster than TCP, to recover from failure.

The Scenario F.2 and F.3, take a less time to recover from failure, than Scenario G.2 and G.3. This happens becauseMPTCP keeps the link status for a longer time than TCP.

In conclusion MPTCP is more resilient to failure than TCP.

6.3.4 Handover

In this subsection we compare the different handover scenarios. 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.

The handover in the context of conventional wireless and wired scenarios, can be seen in the Figures 6.12, 6.13 and 6.14. While Figures 6.15 to 6.20 show the handover in the context of mobile environment

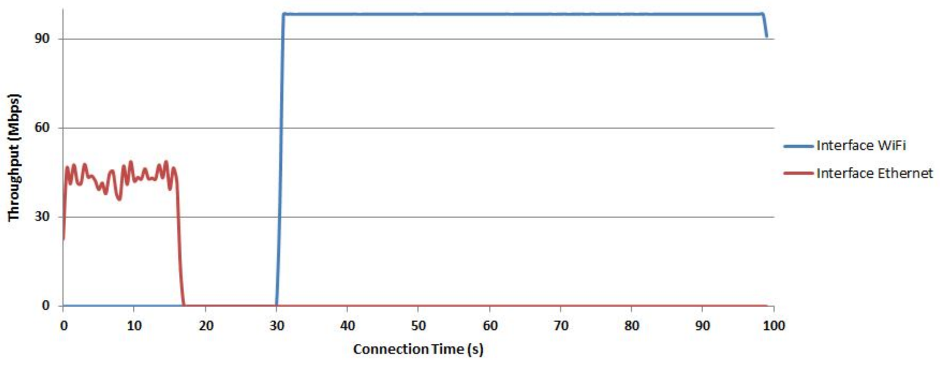


Figure 6.12: Scenario A.4 : MPTCP Handover between WiFi and Ethernet

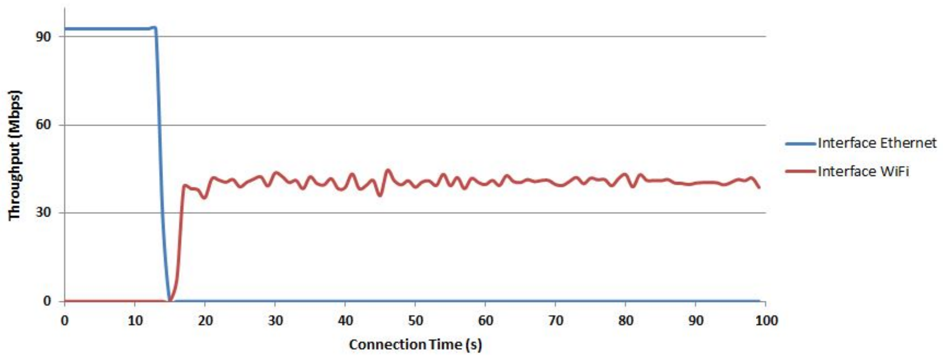


Figure 6.13: Scenario A.5 : MPTCP Handover between Ethernet and WiFi

scenarios.

Figure 6.12, shows the connection reaction when occurs an connection switches from using WiFi to the Ethernet interface. The connection takes 13 seconds to complete the handover.

Figure 6.13, shows the connection reaction when occurs an connection switches from using Ethernet to WiFi interface. The connection takes 0.5 second to complete the handover.

Figure 6.14, shows the connection reaction when occurs an connection switches from using one Ethernet to other Ethernet interface. The connection handover takes 4.5 seconds to complete.

The Figure 6.15 shows the handover between the 4G and Campus WiFi Network. The time to complete the handover is the time that the mobile needs to complete the authentication process to the Campus Network, which takes 3.5 seconds.

The Figure 6.16 shows the handover between the Campus WiFi and 4G Network. The time to complete the handover is less than 0.5 seconds.

The Figure 6.17 shows the handover between the Campus WiFi and Local WiFi Network. The time to complete the handover is less than 0.5 seconds.

The Figure 6.18 shows the handover between the Local and Campus WiFi Network. The time to complete the handover is the time that the mobile needs to complete the authentication process to the Campus Network, which takes 3.5 seconds.

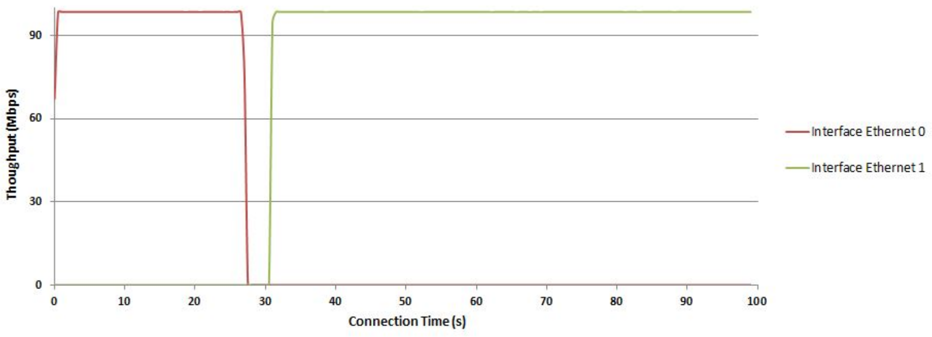


Figure 6.14: Scenario B.3 : MPTCP Handover between two Ethernet Connections

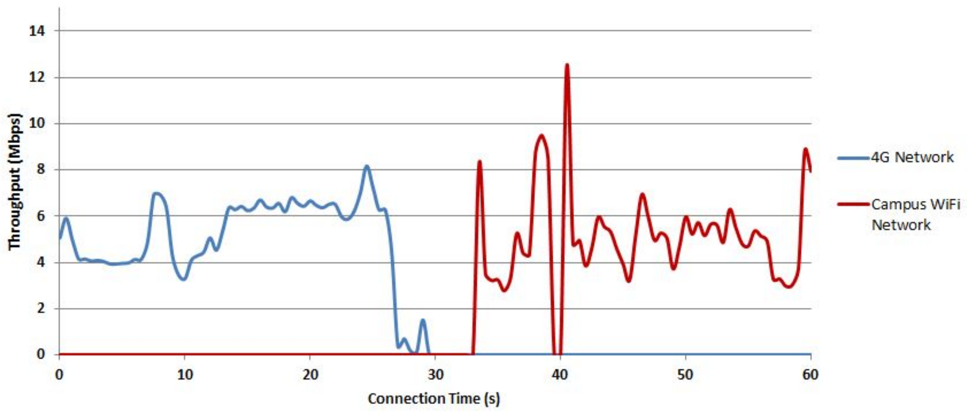


Figure 6.15: Scenario F.4 : MPTCP Handover between 4G and Campus WiFi Network

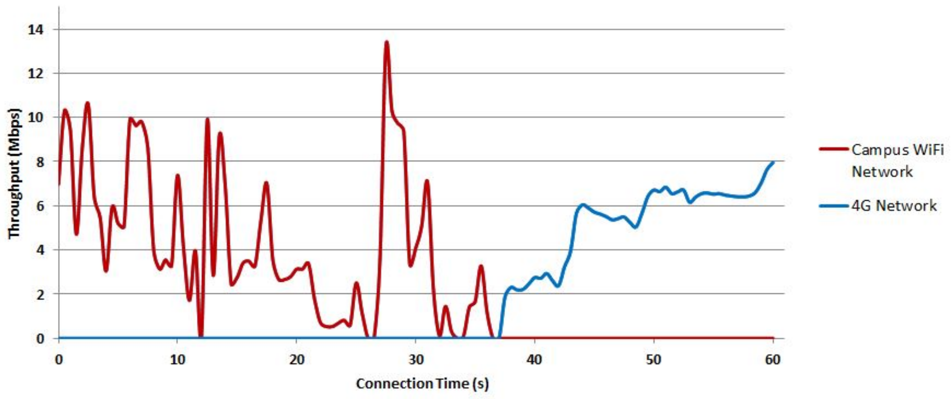


Figure 6.16: Scenario F.5 : MPTCP Handover between Campus WiFi and 4G Network

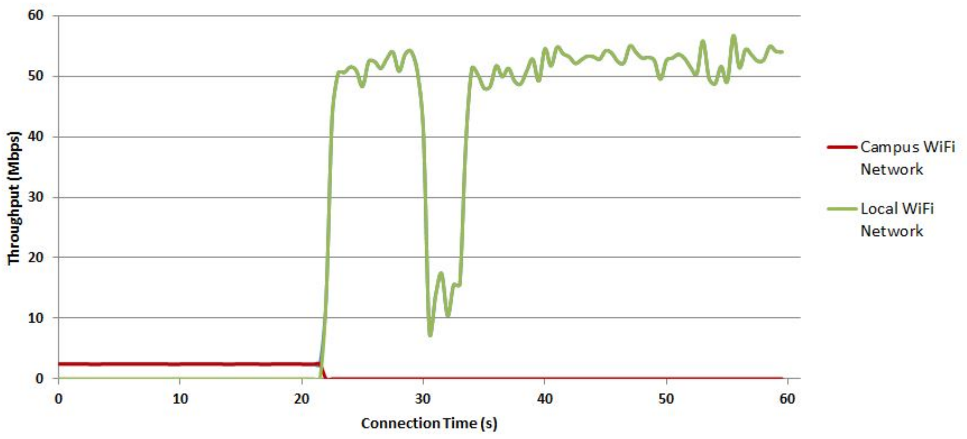


Figure 6.17: Scenario F.6 : MPTCP Handover between Campus WiFi and Local WiFi Network

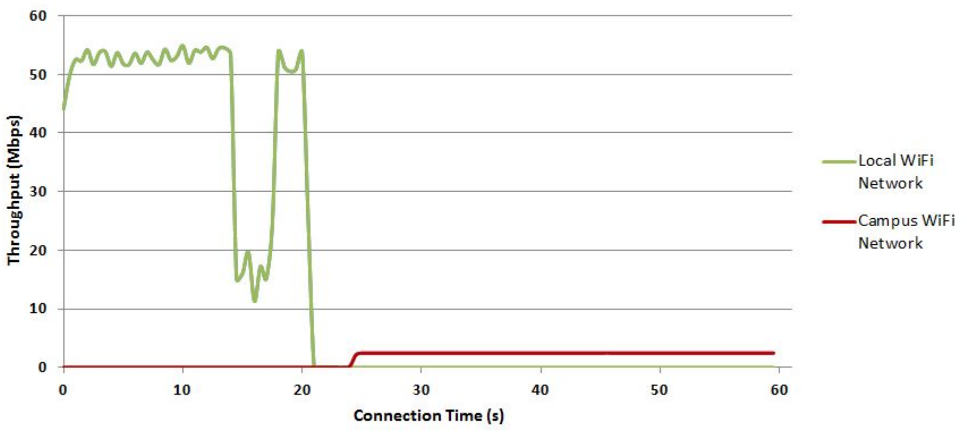


Figure 6.18: Scenario F.7 : MPTCP Handover between Local WiFi and Campus WiFi Network

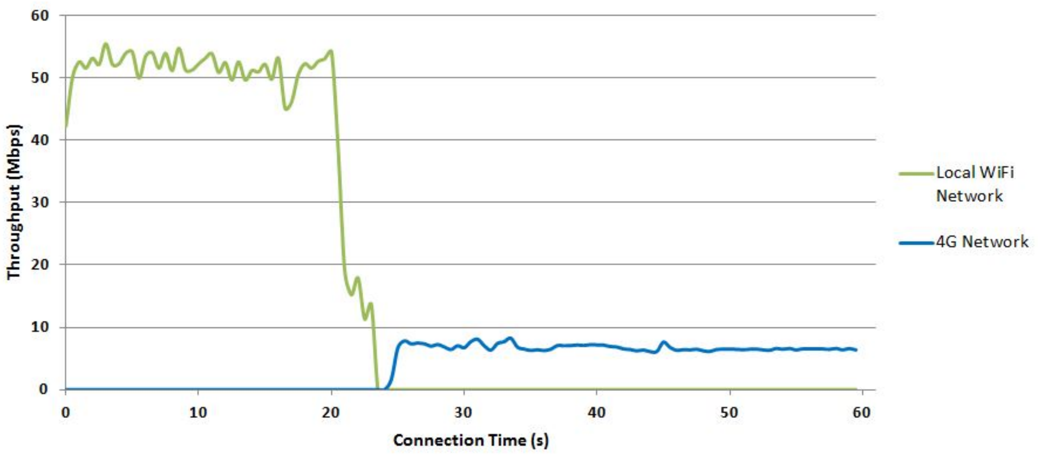


Figure 6.19: Scenario F.8 : MPTCP Handover between Local WiFi and 4G Network

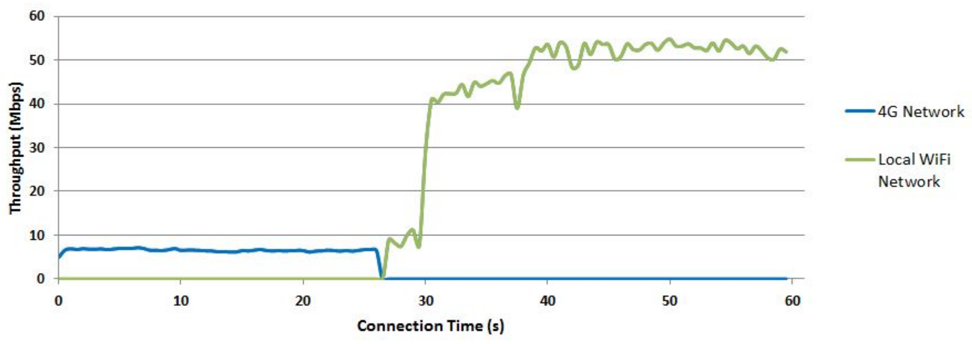


Figure 6.20: Scenario F.9 : MPTCP Handover between 4G and Local WiFi Network

The Figure 6.19 shows the handover between the Local WiFi and 4G Network. The time to complete the handover is less than 0.5 seconds.

Finally the Figure 6.20 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 conventional wireless and wired scenarios, the handover it is somewhat slow when connecting to a Ethernet interface, as it was expected with the network drivers in use. 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 avantage of keeping the same connection, even after the handover between different networks.

6.3.5 Upstream and Downstream Bandwidth Links

In this subsection is described the evaluation of fairness of both protocols in links that use similar upstream and downstream bandwidth.

Table 6.6 shows the percentage of bandwidth used for a upstream and downstream in both contexts. As shown in the Table 6.6, the MPTCP has a similar behavior to the TCP.

6.3.6 Concurrency on Network Links using Simulation

In order to test a more controlled environment to create concurrency over the network, it was simulated the use of five clients on each scenario on the Table 6.7.

As it can be seen in Table 6.7 the MPTCP only loses fairness in networks with constrained connection, Campus WiFi network. Otherwise the protocol reacts very similar as the TCP, in the same situations.

Links with similar Upstream and Downstream							
Test Scenario	Protocol	Context	Interfaces	Upstream		Downstream	
				(Mbps)	**(%)**	(Mbps)	**(%)**
Scenario A.7	MPTCP	Wired and Wireless	WiFi + Ethernet	63.91	**34**	124.74	**66**
Scenario B.8	MPTCP	Wired and Wireless	Ethernet + Ethernet	131.47	**49**	136.3	**51**
Scenario C.3	MPTCP	Wired and Wireless	WiFi	15.15	**36**	26.71	**64**
Scenario C.7	MPTCP	Wired and Wireless	Ethernet	27.64	**23**	91.44	**77**
Scenario D.3	TCP	Wired and Wireless	WiFi	14.1	**32**	29.7	**68**
Scenario D.7	TCP	Wired and Wireless	Ethernet	87.78	**50**	88.14	**50**
Scenario F.1	MPTCP	Mobile Environment	WiFi (Local)	9.13	**27**	24.8	**73**
Scenario F.2	MPTCP	Mobile Environment	WiFi (Campus)	1.39	**32**	2.97	**68**
Scenario F.3	MPTCP	Mobile Environment	4G	4.46	**27**	12.13	**73**
Scenario G.1	TCP	Mobile Environment	WiFi (Local)	15.12	**32**	32.19	**68**
Scenario G.2	TCP	Mobile Environment	WiFi (Campus)	0.56	**19**	2.46	**81**
Scenario G.3	TCP	Mobile Environment	4G	4.12	**27**	11.09	**73**

Table 6.6: Connection Links with Similar Upstream and Downstream

6.3.7 Subflows

In this subsection is shown subflow information obtained from using the netstat tool.

Listing 6.1: Subflow Information of Multihomed MPTCP

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Local Token Remote Token
tcp   0 0   0.0.0.0:5001      0.0.0.0:* LISTEN
tcp   0 0  localhost:ipp     0.0.0.0:* LISTEN
tcp   0 0  mptcpdesktop2-POWE:5001  mptcpdesktop1-POW:59824  ESTABLISHED
tcp   0 0  mptcpdesktop2-POWE:5001  mptcpdesktop1-POW:57846  ESTABLISHED
tcp6  0 0  [::]:12865        [::]:*          LISTEN
tcp6  0 0 ip6-localhost:ipp  [::]:*          LISTEN
mptcp 0 0 mptcpdesktop2-POWE:5001  mptcpdesktop1-POW:59824  ESTABLISHED
3435536843   3884930593
```

When using a default multihomeding implementation, the protocol creates one MPTCP connection and establishes the number of subflows needed, to have one for each available network interface. In the

Concurrency Between Simulated Clients														
Test Scenario	Protocol	Context	Interface	Throughput										
				Total	Client1		Client2		Client3		Client4		Client5	
				(Mbps)	(Mbps)	**(%)**	(Mbps)	**(%)**	(Mbps)	**(%)**	(Mbps)	**(%)**	(Mbps)	**(%)**
Scenario A.8	MPTCP	Wired and Wireless	WiFi + Ethernet	132.96	23.28	**18**	25.47	**19**	31.3	**23**	27.3	**21**	25.61	**19**
Scenario B.7	MPTCP	Wired and Wireless	Ethernet + Ethernet	185.69	51.91	**28**	38.62	**21**	34.81	**19**	33.37	**18**	26.98	**14**
Scenario C.4	MPTCP	Wired and Wireless	WiFi	41.05	8.03	**19**	8.12	**20**	8.07	**20**	8.5	**21**	8.33	**20**
Scenario C.8	MPTCP	Wired and Wireless	Ethernet	93.55	17.58	**19**	18.1	**19**	18.12	**19**	18.8	**20**	20.95	**23**
Scenario D.4	TCP	Wired and Wireless	WiFi	41.85	8.51	**20**	8.48	**20**	7.86	**20**	8.5	**20**	8.5	**20**
Scenario D.8	TCP	Wired and Wireless	Ethernet	94.27	18.86	**20**	18.84	**20**	18.86	**20**	18.85	**20**	18.86	**20**
Scenario F.1	MPTCP	Mobile Environment	WiFi (Local)	25.16	5.76	**23**	5.43	**22**	4.62	**18**	4.98	**20**	4.37	**17**
Scenario F.2	MPTCP	Mobile Environment	WiFi (Eduroam)	2.39	0.1	**4**	0.47	**20**	0.89	**37**	0.14	**6**	0.79	**33**
Scenario F.3	MPTCP	Mobile Environment	4G	12.22	1.88	**15**	2.91	**24**	1.3	**10**	3.25	**27**	2.88	**24**
Scenario G.1	TCP	Mobile Environment	WiFi (Local)	47.29	9.53	**20**	9.61	**20**	9.54	**20**	8.28	**18**	10.33	**22**
Scenario G.2	TCP	Mobile Environment	WiFi (Eduroam)	2.38	0.51	**21**	0.51	**21**	0.34	**14**	0.54	**23**	0.48	**21**
Scenario G.3	TCP	Mobile Environment	4G	11.92	2.22	**19**	4.57	**39**	1.94	**16**	1.6	**13**	1.59	**13**

Table 6.7: Load Balancing between Five Simulated Clients

listing 6.1 we can observe a normal MPTCP connection with two network interfaces.

In the listing 6.2 is shown that the protocol uses five MPTCP connections, one for each client. Knowing that a MPTCP connection supports three subflows, each of these subflows is created as a TCP connection.

6.3.8 Summary Analysis

The MPTCP protocol has been tested successfully using existing infrastructures and adapting only the endpoints.

After evaluating the test results, it can be concluded that MPTCP use multiple subflows in a single MPTCP session. Each subflow appears to the system as a TCP connection. In a multihoming MPTCP implementation it uses a subflow for each interface. When using a MPTCP subflow implementation it creates the number of desired subflows.

MPTCP has shown to be as fair as TCP, most of the time. Not only when in concurrency with TCP traffic over the network link, Table 6.3, but also concurrency between MPTCP simulated clients, as shown section 6.3.6. It has also offer the same fairness to links that use similar upstream and downstream bandwidth, as it was shown in section 6.3.5.

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 6.3.3 it was shown that it recovers from network failure more quickly than TCP and can perform handover of interfaces without losing the connection, it was shown section 6.3.4.

Listing 6.2: Subflow Information of Five Clients using MPTCP Subflow Implementation

```
Active Internet connections (servers and established)
Proto Recv−Q Send−Q Local Address          Foreign Address         State
Local Token Remote Token
tcp       0 0 localhost:ipp           0.0.0.0:*                  LISTEN
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:43650 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:41414 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:47972 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:51300 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:42783 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40473 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40477 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:56828 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:36989 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40476 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40475 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:33078 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:39708 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:41111 ESTABLISHED
tcp       0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40474 ESTABLISHED
tcp6     0 0 [::]:12865              [::]:*                     LISTEN
tcp6     0 0 ip6−localhost:ipp       [::]:*                     LISTEN
mptcp 0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40475 ESTABLISHED
2178822189   1509271798
mptcp 0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40476 ESTABLISHED
2150733962   3496238535
mptcp 0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40473 ESTABLISHED
843789494    3476187660
mptcp 0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40477 ESTABLISHED
2514456      3855578736
mptcp 0 0 mptcpdesktop2−POWE:5001 mptcpdesktop1−POW:40474 ESTABLISHED
3870348066   531314471
```

Chapter 7

Conclusions

7.1 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.

7.2 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.

Bibliography

[1] Postel, J.: Transmission control protocol. RFC 793, Internet Engineering Task Force (September 1981)

[2] Stewart, R.: Stream Control Transmission Protocol. RFC 4960 (Proposed Standard) (September 2007)

[3] Iyengar, J., Amer, P., Stewart, R.: Concurrent multipath transfer using sctp multihoming over independent end-to-end paths. Networking, IEEE/ACM Transactions on **14**(5) (2006) 951–964

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

[5] Hsieh, H.Y., Sivakumar, R.: A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. Wireless Networks **11**(1-2) (2005) 99–114

[6] Nordmark, E., Bagnulo, M.: Shim6: Level 3 multihoming shim protocol for ipv6. Technical report, RFC 5533, June (2009)

[7] Ford, A., Raiciu, C., Handley, M.: TCP extensions for multipath operation with multiple addresses. Internet-Draft draft-ietf-mptcp-multiaddressed-02.txt, IETF Secretariat, Fremont, CA, USA (July 2010)

[8] Wang, C., Sohraby, K., Li, B., Daneshmand, M., Hu, Y.: A survey of transport protocols for wireless sensor networks. Network, IEEE **20**(3) (2006) 34–40

[9] Allan, D., Ashwood-Smith, P., Bragg, N., Farkas, J., Fedyk, D., Ouellete, M., Seaman, M., Unbehagen, P.: Shortest path bridging: Efficient control of larger ethernet networks. Communications Magazine, IEEE **48**(10) (October 2010) 128–135

[10] Tarn, W.H., Tseng, Y.C.: Joint multi-channel link layer and multi-path routing design for wireless mesh networks. In: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, IEEE (2007) 2081–2089

[11] Cohen, B.: Incentives build robustness in bittorrent. In: Workshop on Economics of Peer-to-Peer systems. Volume 6. (2003) 68–72

[12] Postel, J.: User datagram protocol. RFC 768, Internet Engineering Task Force (August 1980)

[13] Perotto, F., Casetti, C., Galante, G.: Sctp-based transport protocols for concurrent multipath transfer. In: Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE. (2007) 2969–2974

[14] Floyd, S., Henderson, T., Gurtov, A.: The newreno modification to tcpś fast recovery algorithm. Technical report, RFC 2582, April (1999)

[15] Sarkar, D.: A concurrent multipath tcp and its markov model. In: Communications, 2006. ICC '06. IEEE International Conference on. Volume 2. (2006) 615–620

[16] Ford, A., Raiciu, C., Handley, M., Barre, S., Iyengar, J.: Architectural guidelines for multipath tcp development. RFC6182 (March 2011), www. ietf. ort/rfc/6182 (2011)

[17] Scharf, M., Ford, A.: Multipath tcp (mptcp) application interface considerations. Technical report, RFC 6897, March (2013)

[18] Johnson, D., Perkins, C., Arkko, J.: Mobility Support in IPv6. IETF. (June 2004) [Standards Track RFC 3775].

[19] Perkins, C.: IP Mobility Support for IPv4. IETF. (August 2002) [Standards Track RFC 3344].

[20] Nordmark, E., Li, T.: Threats Relating to IPv6 Multihoming Solutions. RFC 4218 (Informational) (October 2005)

[21] van Beijnum, I.: One-ended multipath TCP (May 2009)

[22] C. Paasch, E., Bonaventure, O.: MultiPath TCP Low Overhead (October 2012)

[23] K. Xue, J. Guo, P.H.L.Z.: TMPP for Both Two MPTCP-unaware Hosts (June 2012)

[24] Bagnulo, M., Bonaventure, O., Gont, F., Paasch, C., Raiciu, C.: Analysis of mptcp residual threats and possible fixes. Analysis (2013)

[25] Barré, S., Bonaventure, O., Raiciu, C., Handley, M.: Experimenting with multipath tcp. SIGCOMM Comput. Commun. Rev. **41**(4) (August 2010) –

[26] Barré, S., Paasch, C., Bonaventure, O.: Multipath tcp: From theory to practice. In Domingo-Pascual, J., Manzoni, P., Palazzo, S., Pont, A., Scoglio, C., eds.: NETWORKING 2011. Volume 6640 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 444–457

[27] Wischik, D., Raiciu, C., Greenhalgh, A., Handley, M.: Design, implementation and evaluation of congestion control for multipath tcp. In: Proceedings of the 8th USENIX conference on Networked systems design and implementation, USENIX Association (2011) 8–8

[28] Honda, M., Nishida, Y., Eggert, L., Sarolahti, P., Tokuda, H.: Multipath congestion control for shared bottleneck. In: Proc. PFLDNeT workshop. (2009)

[29] Han, H., Shakkottai, S., Hollot, C., Srikant, R., Towsley, D.: Overlay tcp for multi-path routing and congestion control. In: IMA Workshop on Measurements and Modeling of the Internet. (2004)

[30] Han, H., Shakkottai, S., Hollot, C.V., Srikant, R., Towsley, D.: Multi-path tcp: A joint congestion control and routing scheme to exploit path diversity in the internet. IEEE/ACM Trans. Netw. **14**(6) (December 2006) 1260–1271

[31] Key, P., Massoulie, L., Towsley, D.: Combining multipath routing and congestion control for robustness. In: Information Sciences and Systems, 2006 40th Annual Conference on. (2006) 345–350

[32] Raiciu, C., Wischik, D., Handley, M.: Practical congestion control for multipath transport protocols. University College of London Technical Report (2009)

[33] Wischik, D., Handley, M., Raiciu, C.: Control of multipath tcp and optimization of multipath routing in the internet. In: Network Control and Optimization. Springer (2009) 204–218

[34] Kelly, F., Voice, T.: Stability of end-to-end algorithms for joint routing and rate control. SIGCOMM Comput. Commun. Rev. **35**(2) (April 2005) 5–12

[35] Hopps, C.E.: Analysis of an equal-cost multi-path algorithm. (2000)

[36] van der Linden, S., Detal, G., Bonaventure, O.: Revisiting next-hop selection in multipath networks. In: Proceedings of the ACM SIGCOMM 2011 Conference. SIGCOMM '11, New York, NY, USA, ACM (2011) 420–421

[37] Singh, A., Xiang, M., Konsgen, A., Goerg, C.: Performance and fairness comparison of extensions to dynamic window coupling for multipath tcp. In: Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International. (2013) 947–952

[38] Khalili, R., Gast, N., Popovic, M., Upadhyay, U., Le Boudec, J.Y.: Mptcp is not pareto-optimal: Performance issues and a possible solution. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies. CoNEXT '12, New York, NY, USA, ACM (2012) 1–12

[39] Nguyen, S.C., Zhang, X., Nguyen, T.M.T., Pujolle, G.: Evaluation of throughput optimization and load sharing of multipath tcp in heterogeneous networks. In: Wireless and Optical Communications Networks (WOCN), 2011 Eighth International Conference on. (2011) 1–5

[40] Detal, G., Paasch, C., Linden, S.v.d., Mérindol, P., Avoine, G., Bonaventure, O.: Revisiting flow-based load balancing: Stateless path selection in data center networks. Computer Networks **57**(5) (2013) 1204 – 1216

[41] Raiciu, C., Pluntke, C., Barré, S., Greenhalgh, A., Wischik, D., Handley, M.: Data center networking with multipath tcp. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. Hotnets-IX, New York, NY, USA, ACM (2010) 10:1–10:6

[42] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., Handley, M.: Improving datacenter performance and robustness with multipath tcp. In: Proceedings of the ACM SIGCOMM 2011 Conference. SIGCOMM '11, New York, NY, USA, ACM (2011) 266–277

[43] Veerman, G., van der Pol, R.: Multipath tcp: Hands-on. Technical report (2012)

[44] Wang, B., Wei, W., Guo, Z., Towsley, D.: Multipath live streaming via tcp: Scheme, performance and benefits. ACM Trans. Multimedia Comput. Commun. Appl. **5**(3) (August 2009) 25:1–25:23

[45] C. Paasch, S.B.e.a.: Multipath tcp - linux kernel implementation (2014)