Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2014, Vol.6, No.3

TCP Performance Enhancement over Wireless Mesh Networks by means of the Combination of Multi-RAT Devices and the MPTCP Protocol

[1]David Gómez, [1]Pablo Garrido, [2]Carlos Rabadán, [1]Ramón Agüero and [1]Luis Muñoz

Communications Engineering Department, University of Cantabria

Avda. de Los Castros S/N, 39005, Santander (Spain)

E-mail: [1]{dgomez,pgarrido,ramon,luis}@tlmat.unican.es,

[2]{carlos.rabadan}@alumnos.unican.es

Received: March 28, 2014 Accepted: June 15, 2014 Published: August 14, 2014

DOI: 10.5296/npa.v6i3.5387 URL: http://dx.doi.org/10.5296/npa.v6i3.5387

Abstract

The last trends at communications realms, in particular, wireless technologies, where it is more and more usual that devices carry more than one interface (i.e. multi-RAT, Radio Access Technology), to get access to the Internet, question the classic single-path paradigm, imposed by the mainstream transport protocol, TCP. In this work we assess the behavior of Multipath TCP (MPTCP), which allows the transparent breakdown of a single TCP session into multiple simultaneous subflows. This straightforward feature might lead to remarkable performance enhancements, yielding as well a stronger resilience against failures within any of the routes. Moreover, we evaluate three different routing algorithms (*link*, *node* and *zone disjoint*) that aim to discover the optimal route configuration of disjoint paths over a Wireless Mesh Network (WMN), exploiting the possibilities arisen by this brand new protocol. We use the obtained results to evaluate, by means of simulation, the behavior of the MPTCP protocol, showing that the aggregated performance is significatively higher than that of achieved by the traditional single-path and single-flow TCP.

Keywords: MPTCP, Multi-homed devices, Congestion control algorithms, Multipath routing algorithms, Wireless Mesh Networks

1. Introduction

Since its inception, the Internet have seen how one of its original protocols, TCP (and its successive modifications), has deservedly become the most widespread transport solution over a number of alternatives (i.e. UDP, Stream Control Transmission Protocol - SCTP, Data Congestion Control Protocol - DCCP, etc.). However, when this protocol was originally proposed, the predominant technology was single-interface wired devices (and networks). Over time, this trend has gradually evolved and nowadays the communication environment is rather different. For instance, most of the latest servers and data centers are becoming multi-homed and they support redundant topologies that allow reaching them over different paths simultaneously.

Besides, one of the key aspects of this "revolution" is the huge growth of wireless devices, whose impact has flipped most of the last mile network archetypes. This sort of equipment carries multiple interfaces belonging to different Radio Access Technologies (RATs)[1]; this might lead to a situation (in the short-term) in which all of them cooperate. For instance, it might become possible to combine the traffic between IEEE 802.11 wireless and Ethernet connections, obtaining a higher aggregated throughput.

Some of these devices will be able to get interconnected amongst themselves, leading to the so-called Wireless Mesh Networks (WMNs). In this sort of topologies, it will be (most of the times) necessary using several hops to reach the destination, by means of intermediate intermediate relay nodes. In order to establish one (or more) paths, routing algorithms shall provide the set of appropriate paths to communicate any pair of nodes nodes within the network (*unicast* transmissions). On the other hand, there are two main mechanisms and protocols using such algorithms, *reactive* (or "on-demand") and *proactive*. Those which belong to the first group only exchange discovery or maintenance messages when needed, whilst the second group periodically updates the routing tables, thus causing a higher overhead.

The research community has made a great effort to look for an appropriate way to support reliable connections by harnessing this new potential. One of the most important outcomes of these studies is the MultiPath TCP (MPTCP) protocol, which can be seen as a natural evolution of TCP, allowing the *effective and simultaneous* use of a numbers of paths (subflows) to handle a unique transport connection. Its control mechanisms distribute the available load between the active subpaths, which are tightly linked to the entity's active IP address. By means of this modification, an application could benefit from a higher throughput and resilience against failures.

Traditionally, most of the multipath analysis were bound to wired topologies (where graph theory has a more straightforward applicability and the links between nodes are completely controllable); however, it is more and more usual finding works that deal with the analysis of the behavior of such schemes over wireless networks (i.e. [1–3]), which have been proven to severely jeopardize the performance of connection-oriented (mainly, TCP) protocols, since a poor quality

[1]Furthermore, a big percentage of them (i.e. laptops, portable all-in-one desktops, etc.) also includes a wired connection, typically Ethernet.

link introduces a substantial number of packet losses.

The content of this work is split into two clearly differentiated parts: on the first hand, we have ported a fully-fledged MPTCP implementation, based on the work initially carried out by Chihani et al. [4], over the `ns-3` platform [5]. By means of a thorough simulation campaign, we have assessed the performance of some of the algorithms to handle the congestion of each subflow; besides, we have compared its behavior to the one achieved by TCP over a simple "diamond" topology, leading to significant throughput improvements. On the second hand, we pose the necessity of relying on multipath routing mechanisms so that an arbitrary source node might reach a potential destination over a multi-hop random topology through multiple simultaneous flows. Namely, we evaluate the behavior of three different routing algorithms, as previously listed and compared in [6]: namely, Link Disjoint (LD), Node Disjoint (ND) and Zone Disjoint (ZD), in order to find the optimal set of disjoint paths over a WMN; afterwards, using the results of the first phase, we assess, based on an extensive simulation campaign over the `ns-3` simulator, the MPTCP performance over this type of topologies, showing the enhancement compared to a traditional single-path TCP scheme.

In order to address the aforementioned objectives, this paper is structured as follows: Section 2 summarizes the main contributions on MPTCP and the main multipath routing solutions, highlighting those works addressing the interaction with wireless mesh topologies. In Section 3 we outline the main characteristics of the MPTCP framework we have used to improve the performance of legacy TCP. Section 4 introduces the three routing algorithms that will be exploited for the MPTCP characterization After that, Section 5 presents the simulation procedure that we have followed and illustrates its most relevant results. Finally, Section 6 concludes the paper, posing a number of challenges to be addressed in the future.

2. Related work

The growing interest on the multi-homing capabilities brought about by the technology during the latest years (e.g. multi-homed servers accessed through different Internet Service Providers (ISPs) connections, data-centers implement redundant routes that are triggered in case of failures, portable devices carrying interfaces belonging to different wireless technologies, etc.), has motivated a new operation scheme in which the traffic could be sliced and simultaneously transported through potentially disjoint paths. To provide an answer to these challenges, Internet Engineering Task Force (IETF) created a working group (namely, the *Multipath TCP Working Group*), which focuses on the implementation of a solution to multiplex the traffic of a single TCP session between multiple routes. The cornerstone of the protocol can be found in [7], which, in addition, relies on a number of extensions (i.e. [8, 9]) that define the basis of the proposed framework, as well as a new scheme to manage the congestion of the different subflows, respectively. Besides, the increasing popularity of wireless technologies has led to the creation of another extension [10], which will propose a set of modifications in order to adapt the operation of this protocol over wireless channels.

MPTCP is not the unique solution that aimed at breaking down the barriers imposed by TCP, where the connections were constrained to a single flow. On an earlier stage, IETF, through the SIGTRAN (SIGnalling TRANsport) working group, laid the foundations of multipath communication by defining the SCTP [11]. Both of them share many similarities, like the possibility to use different IP addresses at the terminal devices, allowing multipath traffic. On the other hand, the differences between them are rather remarkable: whilst the MPTCP's main objective focuses on the improvement of TCP performance (throughput and resilience) by means of the simultaneous transmission over different subpaths, Stream Control Transmission Protocol (SCTP) addresses the redundancy and mobility for multi-homed nodes, without parallelizing the traffic over more than one path at the same time.

We can find in the literature a number of works that, like ours, characterize the behavior of MPTCP over wireless channels. Based on a Linux Kernel public implementation (available in [12]), Lim and Valdez [13] show the actual improvement of MPTCP transmission compared to the legacy TCP performance over a combination of Ethernet, IEEE 802.11 and 3G links. Besides, they also assess the capacity of holding up a transport session during a vertical handover. Whilst their measurement campaign is based on an emulated channel with pre-configured packet data rates (0% in Ethernet, 2% in 3G and 3% in IEEE 802.11), the authors compare their results to those obtained via real experimentation [14] and conclude that, although the quality at the links was quite good during the whole simulation campaign, we can significatively improve the overall performance. On the other hand, Nguyen et al. [15] use a different implementation to assess the throughput and load distribution between two subflows over scenarios which combine Ethernet, IEEE 802.11 and 3G technologies. Nonetheless, in this case the results are not so encouraging, since they show that MPTCP performance over different technologies is lower than the one shown by TCP (using the best path). The authors attribute this to the negative impact of the poor reordering scheme when applied to links characterized by rather different parameters; however, Nguyen et al. do not provide details about the techniques used during their simulations (`ns-2`). In addition, it is worth highlighting more recent contributions, such as the one carried out by Chen et al. in [2], where the authors assess "in the wild" the performance of different WiFi and cellular combinations (LTE and 3G) upon the download of files with different sizes. Their results show that MPTCP normally improves the performance achieved by TCP; however, they also assert that file sizes have a key impact upon MPTCP behavior. Finally, they state that the choice of a congestion control algorithm would not have a special relevance on small file transfer, yielding alike results; for large downloads, the performance obtained by legacy solutions does not behave as well as newer alternatives, such as *Opportunistic Linked Increases Algorithm (OLIA)* [16]. On the other hand, Paasch et al. [3] also exploit the Linux Kernel framework of MPTCP to highlight several limitations of the implementation, obtained under a (as they refer to) "Experimental Design". According to their results, these shortcomings are typically linked to the congestion control algorithms and the scheduler which is in charge of deciding how to multiplex the packets among the different subflows.

It is also worth highlighting the work carried out by Chihani et al. [4], which is, to our best knowledge, the first MPTCP's `ns-3` implementation. They were based on the recommendations

proposed by the IETF's working group and reproduced the operation of the protocol within the simulator framework. The authors compare the performance of two different packet reordering algorithms using an FTP transmission between two nodes connected through two parallel point-to-point wired links. They also analyze the behavior of the two corresponding congestion windows.

In addition, in this work we exploit different routing algorithms to be used by multi-path strategies, assessing their potential benefits over WMNs. The use of multi-path communications might lead to a greater performance; moreover, they will bring about a more resilient connections, dynamically adjusting the load over the various paths, according to the particular network conditions.

As mentioned earlier, the classification of routing protocols for wireless multi-hop networks embraces two main groups. Both of them are based on mechanisms to discover and maintain routes over multi-hop networks. The *proactive* protocols (represented by *Optimized Link State Routing - OLSR* [17]), update the routing information by periodically flooding the network with topological information, thus introducing a remarkable overhead. On the other hand, the *reactive* or *on-demand* protocols (for instance, *Ad hoc On-demand Distance Vector routing - AODV* [18]) reduce, as much as possible, the exchange of control messages, triggering them only if needed.

The first generation of routing protocols was thought to operate with *single-path* strategies, where in case of a route breakdown, the source node would need to start a new discovery process, in order to find alternative paths to reach the destination. The increasing interest on multi-path solutions during the last years opens new possibilities, and the new protocols are required. Most of the existing solutions are modifications of single-path protocols and can be classified according to how they select the alternative paths to the shortest one: *LD*, which only excludes the links of the previously calculated routes (e.g. *Ad hoc On-demand Multipath Distance Vector - AODVM*), *ND*, which does not allow any intermediate node to be active in two different routes (e.g. *Geographic Multipath routing Protocol - GMP*) and, finally, *ZD*, which inhibits the redundant participation of both the previously used nodes as well as their corresponding neighbors (e.g. *Zone Disjoint Multipath extension of the Dynamic Source Routing - ZD-MPDSR*).

Some of the most relevant works within this research line were carried out by Meghanathan [6, 19], who, by means of graph theory, realizes a complete performance analysis of the *link*, *node* and *zone disjoint* algorithms over WMNs, where he thoroughly studies, through different simulation campaigns, different performance metrics that characterize the behavior of different routing schemes (e.g. number of routes found, average number of hops, average time between single/multipath route discoveries, etc.) Moreover, Waharte et al. [20] carry out another analysis that focused on LD and ND, paying special attention to the potential interferences between the different subflows (since they share the same channel), and estimate the resulting throughput as a function of the nodes' coverage area and their position within the scenario. Unlike Meghanathan's contribution, which only addresses the fundamental analysis of the routing algorithms, Waharte et al. apply end-user traffic (over UDP) to compare the performance of the different routing multipath solutions to that shown by a single-path scheme.

After finding the set of disjoint paths between a source node and a destination, we need to develop a solution to split a single connection into multiple subflows. Some proposals, based

on the modifications of the legacy TCP operation have been already made, such as *mTCP* [21], *Reliable Multiplexing Transport Protocol (R-MTP)* [1] or *parallel TCP (pTCP)* [22], among others. The relevance of this type of communications is supported by the presence of standardization bodies, such as IETF, in the development of new protocols and techniques. In this sense, there are two working groups exclusively devoted to the design and implementation of the most relevant multi-path solutions: SCTP [11] and MPTCP [7]. The former one uses multiple routes in order to provide some redundancy against failures, or to ease the mobility between different networks without breaking a session (at the transport level), but it does not support (yet) the simultaneous transmission over different paths; on the other hand, MPTCP focuses on the improvement of the TCP performance by multiplexing traffic load over different resources.

3. The MPTCP protocol

MPTCP is originally defined as a modified version of the mainstream transport level protocol, TCP. It supports a number of extensions that allow a terminal to split a single session into multiples subflows across the network to simultaneously transport the information. The protocol will be in charge of distributing the load between the different "sub-connections", flowing over potentially disjoint paths, even belonging to different technologies. The protocol has been first tailored to work with end-to-end multipath strategies, where one (or more) of the terminal devices must have more than one valid IP address. Its basic principle is rather simple: if a terminal owns multiple points of connection (interfaces), this can be exploited, simultaneously dividing the traffic between different subconnections. Thanks to these multipath strategies, the overall performance is improved, as well as the robustness of the communication.

One of the main features of MPTCP is its backwards compatibility to its base protocol, TCP (or any of its variants), to facilitate potential migration strategies. In order to ease this migration from legacy protocols, ensuring its compatibility with TCP, RFC 6824 [7] establishes that any MPTCP implementation must be able to support any non-MPTCP-aware application; in such cases, the services will not be able to differentiate between MPTCP and TCP transport level connections.In this sense, for any non-MPTCP-aware application, an MPTCP session will be alike any legacy TCP connection.

The first aspect to be highlighted can be seen as that MPTCP is a TCP [23] modified version, able to split a transport connection between different paths (or subflows) simultaneously. Another element worth mentioning is that it should be transparent to the upper layers, i.e. an application would not distinguish between MPTCP and the legacy TCP.

The next three goals summarize MPTCP's main objectives that have to be fulfilled in order to stand as an appropriate alternative to the legacy TCP:

1. **Improve throughput**: The performance achieved by a multipath flow should be, at least, as high as the legacy single-flowed TCP's (over the best available path).

2. **Do not harm**: An MPTCP subflow should not take more resources than the ones a single TCP flow would need using only one of the paths.

3. **Balance congestion**: Upon a congestion situation, MPTCP shall move as much traffic as possible off the most congested paths (while keeping the first two goals).

As shown in Fig. 1, the MPTCP layer operates at the transport layer, covering all the functionalities supported by TCP. Its operation has been divided into two parts:

1. The higher sublayer is in charge of the application-oriented tasks, such as the session initialization/finalization, subflow establishment, detection and use of multiple paths, etc.

2. The lower one, focused on networking aspects, individually manages each of the subflows created during the connection initialization phase. For that purpose, there will exist as many entities as the number of the existing subflows in the connection

Below these sub-entities, a different IP instance will be associated to each of the subflows (it is worth highlighting that, at least one of the terminal nodes, has to include more than one IP interface). As mentioned before, this architecture fulfills a transparent operation, towards the upper layers as well as the lower ones.

We outline below the main functionalities addressed by the upper sub-level:

Connection establishment. In order to respect a backwards compatibility, an MPTCP session will be initiated with the traditional TCP's *three-way-handshaking* exchange. In a second stage, provided that it is possible using more than one path, different TCP sessions will be established, and the combinations among the IP addresses may lead to additional subflows.

Sequence numbering. MPTCP will preserve the TCP's sequencing scheme, but this will be specific on each subflow. Furthermore, to allow the correct application data reordering at the receiver node, an additional sequence numbering space is required to assemble the pieces of information coming from the different paths, that allows to recompose the pieces of information which come from the different paths. Therefore, it establishes (both at transmission and reception) a mapping between the upper sub-level and the subflows sequence number, which will be carried as an optional field within the MPTCP header, the Data Sequence Number (DSN).

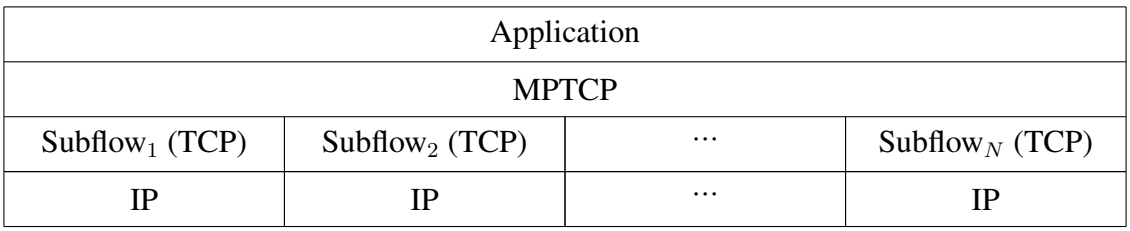


Figure 1: MPTCP protocol architecture

Packet reordering. The fact that the connection is split into several simultaneous subflows over different paths, increases the possibility of receiving out-of-order segments. For this purpose, it becomes essential to define schemes that allow recovering the information in an efficient way, so as to minimize the number of spurious retransmissions. Although there are other proposals, in this work we rely on the *DSACK* [24] mechanism, an extension of the well-known *Selective Acknowledgement (SACK)* algorithm, that allows to acknowledge and to selectively retransmit those information chunks which are considered essential, thus avoiding the unnecessary retransmission of whole data blocks.

Packet distribution. Another mechanism (at the transmitter side) between the different subflows is also needed. For the sake of simplicity, the solution in the current protocol implementation is based on a traditional *Round Robin (RR)*, which fairly distributes the data segments between the available subpaths. The upper sub-level will alternatively send down a segment to each of the subflows, repeating the process while the corresponding transmission buffer has information waiting to be sent.

Besides all these functionalities, it is worth mentioning that MPTCP keeps some of the most widespread TCP mechanisms, such as *Slow Start*, *Fast Retransmit* and *Fast Recovery*.

In order to appropriately distribute the traffic between the various subflows, an interface between the two MPTCP sub-levels manages the traversal information between them. For that, it must employ congestion control mechanisms to fulfill the three aforementioned main objectives. These algorithms must bring about an efficient *resource pooling* (in [25], this term is coined as "making a collection of resources behave like a single pooled one").

In the scope of this work an independent congestion window is assigned to each of the subflows belonging to an MPTCP session. In order to ensure a certain *resource pooling* [25]), it becomes necessary that the evolution of the congestion windows is not completely isolated from the others, but they should be coupled.

The four different algorithms which can be used to manage congestion supported by the MPTCP framework are provided below. We use the notation w_i to refer the congestion window size of the *i-th* subflow, while w represents the overall window size: $w = \sum_i w_i$, where $i = 1, \cdots, N$, being N the total number of active subflows.

1. **Uncoupled subflows**: This is a naive solution, in which each of the congestion windows (w_i) behaves completely isolated from the others, like two independent TCP connections. Equation (1) reflects the window behavior according to a successful/failed segment tranmission.

$$w_i = \begin{cases} w_i + \frac{1}{w_i}, & \text{if ACK received on path i} \\ \frac{w_i}{2}, & \text{if loss event on path i} \end{cases} \quad (1)$$

2. **Fully coupled**: In this case, the evolution of the congestion windows is tightly linked, as shown in (2), where, unlike the previous expression, every window growth rate is the same

(i.e. the sum of all the congestion windows instead of its individual size); when one of the subflows experiments a higher packet error rate, MPTCP, assuming a congestion situation, will allocate more traffic onto the less saturated path (**Goal 3**). A problem may arise if the load of a particular subflow suffers a burst of consecutive packet drops, which might lead to a minimum window size ($w_i \approx 0$) that will be maintained during a long time. To return to a stable situation, the other subflow must suffer a high number of losses until both congestion windows are balanced again ($w_i \approx w_j$). This effect is referred as "flappiness" in [26].

$$w_i = \begin{cases} w_i + \frac{1}{w}, & \text{if ACK received on path i} \\ max(w_i - \frac{1}{2}, 1), & \text{if loss event on path i} \end{cases} \tag{2}$$

3. **Linked increases**: In order to overcome the effect observed in the previous case, Raiciu et al. [14] propose a modification in the *Additive Increase* expression, which is able to mitigate the "flappiness", with the cost of a worse resource pooling, thus avoiding an unfair load distribution between the subflows. The new parameter, α, establishes the congestion window growth factor, as shown in (3).

$$w_i = \begin{cases} w_i + \frac{\alpha}{w_i}, & \text{if ACK received on path i} \\ \frac{w_i}{2}, & \text{if loss event on path i} \end{cases} \tag{3}$$

4. **RTT compensator**: This latter algorithm is actually a modification of the previous one, to be applied when the paths show rather different Round Trip Time (RTT) values (i.e. different physical technologies, appearance of a bottleneck over a particular path, higher traffic load, etc.). Like the previous mechanism, the congestion window growth factor is controlled by the α parameter, gathering its *Additive Increase Multiplicative Decrease (AIMD)* response in (4).

$$w_i = \begin{cases} w_i + \min(\frac{\alpha}{w}, \frac{1}{w_i}), & \text{if ACK received on path i} \\ \frac{w_i}{2}, & \text{if loss event on path i} \end{cases} \tag{4}$$

As was seen in the last two algorithms, the congestion window growth is modified by means of the α parameter, known as *aggressiveness factor*, defined in (5):

$$\alpha = w \cdot \frac{max(\frac{w_i}{RTT_i^2})}{(\sum_i \frac{w_i}{RTT_i})^2} \tag{5}$$

The throughput of a subflow depends on a number of metrics (all of them assigned by α): the packet loss rate, the maximum segment size and the RTT.

However, it is worth highlighting that the performance supported by MPTCP is far from being optimal. For instance, Khalili et al. stated in [16] two main problems: (1) The update of regular

TCP users to MPTCP might reduce the throughput of other users without any benefit to these ones. (2) MPTCP users could be excessively aggressive towards TCP users. Namely, they attributed them to the *Linked Increases* algorithm, specifically due to the excessive amount of load offered over congested paths. As a solution, they proposed *OLIA*, an evolution of the aforementioned *Linked Increases* which aim at alleviating these harmful effects.

4. Multipath routing algorithms

After describing the operation of the MPTCP protocol, where we have checked that we can find real solutions to support a reliable transport session over several paths simultaneously, it is deemed necessary to make use of different and novel routing protocols, whose main goal consists in finding an optimal set of disjoint paths to simultaneously carry the traffic load using multiple subflows, over a WMN scenario[2]. In order to describe the operation of each of the algorithms (*link*, *node* and *zone disjoint*), we will employ a traditional graph theory notation, as shown below. it is worth highlighting that we have based this assessment from the work presented by Meghanathan in [6], sharing his algorithm description and notation.

Let $G(V, E)$ be the graph representing the scenario over which we want to get the set of paths (using the LD, ND or *ZD* algorithms)[3] between the source and the destination nodes (s and d, respectively). The set V represents the group of vertices (nodes) deployed within the scenario, and E (edges) is the set of existing links. We will establish a link between two nodes if the distance between them is shorter than the corresponding range of transmission. In this work we will use homogeneous nodes, and all of them will share the same coverage.

The first step to get the set of paths is the same for the three algorithms: the Dijkstra's algorithm is used to find the shortest path between s and d. If there is, at least, one route in G, it is stored in the corresponding set (P_L, P_N o P_Z, for the LD, ND or ZD algorithms, respectively). After that, the graph ($G \rightarrow G'$) is updated with the constraints imposed by each of the algorithms. Below we show the procedure followed by each of the solutions:

- **Link Disjoint (LD)**. We will remove from G all the links that were found with the Dijkstra's algorithm, thus building a new graph $G'(V, E^L)$. The procedure is repeated as many times as there is a route $s - d$ (Dijkstra's algorithm is used again), incorporating the resulting path to P_L. When the algorithm is finished, P_L contains the set of link disjoint paths of the original graph G.
- **Node disjoint (ND)**. In this case, the graph is modified by deleting the nodes belonging to the previously selected path. Therefore, after each iteration a new route is added to P_N and

[2]Classical routing strategies do not provide such operation, limiting to a single route between the source and the destination.

[3]In this work, since the nodes do not move, the subjacent topology will stay static during the simulation time; therefore, we only need to calculate the routes once.

a modified graph $G'(V^N, E^N)$ is built. The procedure is executed as long as s can discover a route to d.

- **Zone disjoint (ZD)**. This is the most restrictive algorithm, since it severely limits the graph between successive iterations, deleting the nodes belonging to the previous route, as well as their neighbors; as a result, the original graph G is modified to $G'(V^Z, E^Z)$. When it becomes impossible finding new routes, the algorithm returns the set of routes P_Z.

In this work the routing tasks have been performed on an external framework, outside the MPTCP implementation, using a proprietary tool developed in C++, which generates a random scenario to establish the graph $G(V, E)$. Afterwards, the route selection procedure was performed by means of a single process.

Since the main objective of this work is to analyze the performance of MPTCP, which simultaneously delivers the information over multiple (disjoint) paths, we will only consider as valid those sets (P_L, P_N or P_Z) with more than one path between s and d.

5. Simulation and results

In this section we address the description of the simulation procedure followed to compare the performance achieved by a multi-homed MPTCP configuration over wireless mesh networks to that shown by a legacy single-flowed TCP transmission over a simple wireless scenario. In addition, we assess the performance of the three previously introduced multipath routing algorithms, which will be further used to define the input (static) routing tables of the nodes over randomly deployed topologies.

5.1. MPTCP protocol characterization

In order to evaluate the performance of MPTCP, we have carried out a thorough simulation campaign over the `ns-3` network simulator, using for that purpose a code adaptation taken from the original work carried out by Chihani et al. [4] (namely, our main contribution was to make an $1:1$ port from an old `ns-3.6` version to a newer one, `ns-3.13`). Through a simple "diamond" topology, as shown in Fig. 2, we have configured the testbed as follows:

- The source node S_1 sends a $20MB$ file to the destination, D_1. Since D_1 is not within the coverage area of S_1, some relaying nodes (R_1, R_2) take care of forwarding the packets towards D_1.
- As defined in [7], the endpoints (at least one of them) must be multi-homed to ensure the correct operation of MPTCP, since the draft establishes the presence of multiple IP addresses as a necessary condition to set the different paths. In this particular experiment, every node

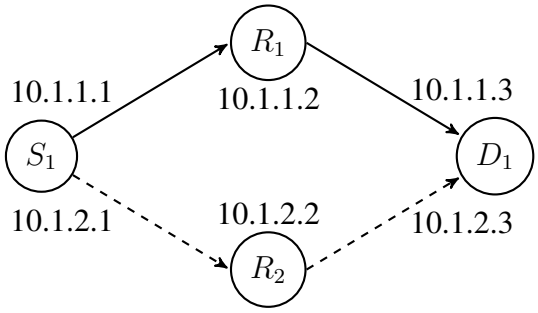


Figure 2: Canonical topology to showcase the benefits of MPTCP

incorporates two IEEE 802.11 interfaces, each of them associated to a different IPv4 subnetwork.

- We aim at establishing the highest performance of the different transport protocol configurations. We mimic a saturated scenario, where the source node always has a segment waiting to be transmitted (as it is the case of the elastic traffic and TCP). We will compare the behavior of the four different congestion control algorithms introduced in Section 3.
- The interfaces use the IEEE 802.11b specification, and the number of transmissions attempts is fixed at four.
- In order to emulate a hostile environment (closer to a more realistic scenario), each link will have a different Frame Error Rate (FER) value, so as to study the behavior of the transport layer implementations under non-ideal conditions.

The main goal of this first stage is to assert the enhancement brought about by the implementation of an MPTCP scheme that exploits the potential of the multi-homed devices over heterogeneous scenarios; therefore, we will compare the performance achieved with the three different configurations, depicted below.

1. The first one corresponds to a legacy TCP transmission (in particular, *New Reno*), which will use just a single flow. By default (it is worth mentioning here that we have used static routing tables), the path followed by the TCP data segments is $S_1 \rightarrow R_1 \rightarrow D_1$.
2. The second case uses MPTCP at the transport layer, thus creating up to two different subflows ($S_1 \rightarrow R_1 \rightarrow D_1$ and $S_1 \dashrightarrow R_2 \dashrightarrow D_1$), both of them configured over a single wireless interface. Hence, the communication will share the same channel and, due to the broadcast nature of the wireless medium, the transmissions will be prone to interfere the other, leading to more contention and increasing the collision probability, thus jeopardizing the overall system performance.
3. Finally, starting on the basis of the previous configuration, we will emulate the possibility of having independent interfaces working on two orthogonal channels. This change will avoid the contention between the stations of different paths, leading to a performance increase.

5.1.1 MPTCP's congestion control mechanisms over a lossless channel

The first performance results to be discussed were obtained over an ideal scenario, where packet losses due to the hostile conditions of wireless channel propagation could be considered neglected; this way, a lost frame could only be caused by a collision[4]. Fig. 3 shows the measured throughput of an arbitrary run for the most interesting configurations: first, in Fig. 3a we can observe the behavior of a legacy TCP transmission, which reaches a stable throughput of ~2.5 Mbps with a single flow. This value is of relevance, since one of the MPTCP's design goals (namely, **Goal 2**) establishes that the bandwidth taken by an MPTCP subflow should not be greater than the one achieved by the legacy TCP. Fig. 3b represents the outcome of an MPTCP single-interface transmission, showing the throughput of each of the subflows as well as their sum and the overall performance average value. We can observe that, although the subflow distributions are alike (in terms of throughput), their sum of them does not surpass the one exhibited by TCP, due to the interaction between the flows contenting for the same channel (i.e. collisions, MAC-related idle times, etc.), which clearly jeopardizes the overall performance.

On the other hand, Fig. 3c illustrates the behavior of the *Uncoupled* congestion control algorithm (the results obtained with *Linked Increases* and *RTT compensator* were rather similar), where, after an initial transitory effect, the throughput of the two subflows reaches a stable maximum value as well as a fair pooling, leading to an overall performance remarkably higher than the one seen for the TCP case (the gain is around 48%). It is worth highlighting that none of the subflows had a higher throughput than the one which would have been achieved by a raw TCP connection, as mentioned above. Finally, Fig. 3d reflects the impact of the so-called "flappiness" effect, exhibited by the *Fully Coupled* algorithm. Since TCP (or MPTCP) is not able to differentiate the actual nature of a segment loss (whether it was due to a collision or the congestion), every packet loss will be considered as a congestion indicator; in this sense, and following MPTCP's **Goal 3**, if one of the subflows suffered from more packet losses than the other one, MPTCP would send more traffic through the "better" path, thus reducing the sending data rate (congestion window decrease) over the "worst" subflow. Besides this unbalanced resource pooling, it is worth mentioning that the overall performance is damaged, since the aggregated throughput severely decreases, compared to an algorithm which avoids the flappiness problem ($\sim 47\%$ lower).

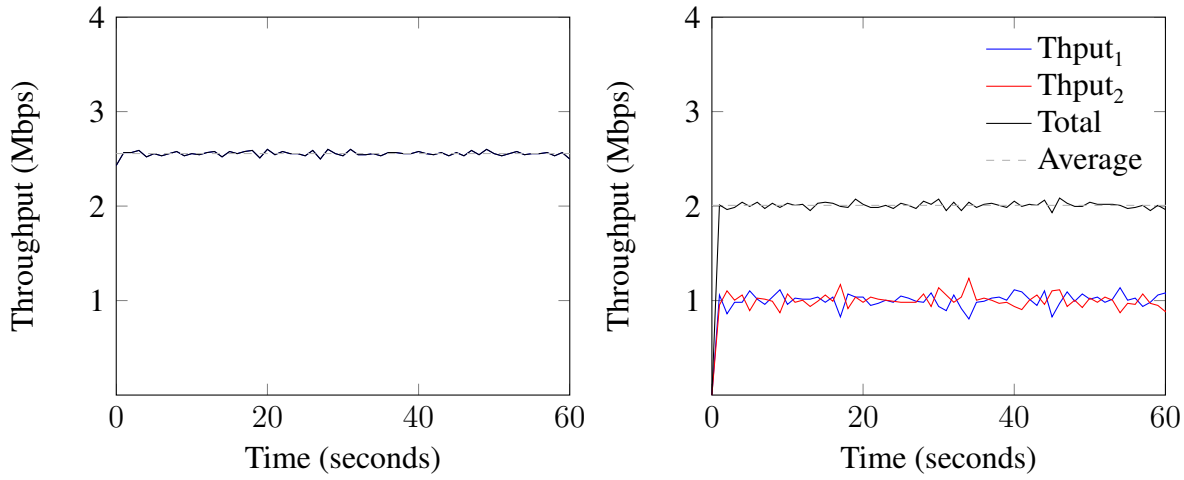
5.1.2 Performance response over packet erasure channels

In a second setup, the links between some nodes are prone to drop packets, in particular the directed links $S_1 \rightarrow R_1$ and $S_1 \rightarrow R_2$ suffered from these losses[5]. Fig. 4 illustrates the throughput evolution[6] as the channel quality decreases. The worst performance corresponds to a single-channel wireless MPTCP transmission (over the same interface), because of the following reasons:

[4]The scenario and the corresponding thresholds were configured so as to avoid the *hidden terminal* problem.

[5]We assume that the probability of losing a backwards TCP acknowledgement is zero.

[6]We have plotted the average throughput as well as the 95% confidence interval of 50 runs for each experiment, showing an almost negligible variance.



(a) Standalone TCP

(b) MPTCP (1 interface)

(c) Uncoupled (2 interfaces)

(d) Fully coupled (2 interfaces)

Figure 3: MPTCP's congestion control algorithm over an ideal wireless channel

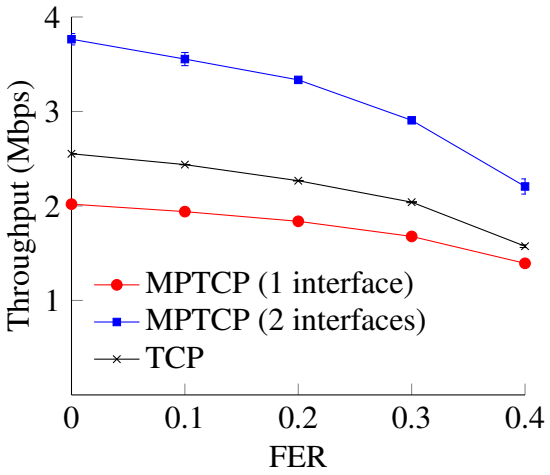


Figure 4: Throughput obtained over lossy links

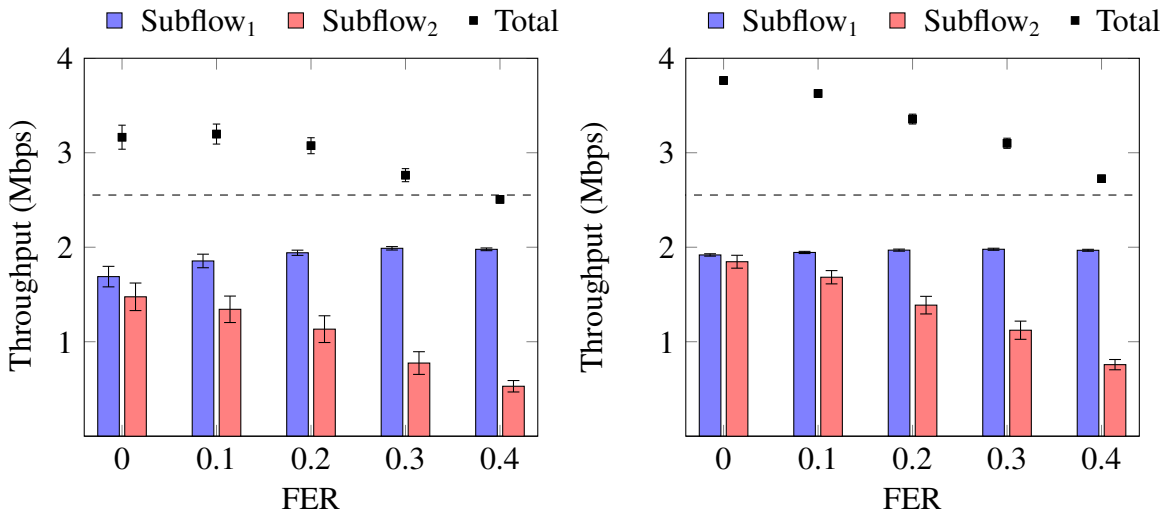
first, the MPTCP congestion control algorithms limit the sending rate in order not to surpass the legacy TCP bandwidth (**Goal 2**); on the other hand, since all nodes share the same channel, every transmission must contend for the channel, thus leading to high idle times.

Finally, the results yield that the use of MPTCP over a multi-channel scenario significatively improves TCP performance ($\sim 48\%$) when the FER is null, and it also outperforms the other alternatives for the rest of configurations.

5.1.3 Load balancing performance test

At last, to properly conclude the assessment over this simple topology, we focus on the analysis of the congestion control algorithms, and their capability to balance the load between the different subflows under asymmetric channel conditions. We therefore assess how the different congestion control algorithms are able to compensate the traffic from a channel characterized by a low quality (**Goal 3**). Fig. 5 illustrates both the average throughput (for each subflow) and the 95% confidence interval for the 50 independents runs executed for every configuration. Besides, we plot the aggregated throughput (black-squared point) and its corresponding confidence interval. Last, the dashed horizontal line represents the average throughput achieved by TCP over a lossless scenario.

Finally, we have slightly modified the FER configuration, so as to mimic an "asymmetric" behavior: whilst the upper link ($S_1 \rightarrow R_1$) remains lossless, the lower one ($S_1 \rightarrow R_2$) is configured so to drop packets according to the corresponding FER value. Through this simple modification, we wanted to assess the feasibility of the MPTCP's **Goal 3**, fairly distributing the congestion, offloading traffic from the most congested paths (recall that TCP is not able to differentiate from congestion or propagation packet losses). Fig. 5a shows the results using a *Fully Coupled* algorithm, where we can appreciate a higher variability and a lower aggregated throughput than with



(a) Fully coupled (asymmetric)

(b) RTT compensator (asymmetric)

Figure 5: Subflow bandwidth sharing under asymmetric channel conditions

the *RTT compensator* (Fig. 5b), especially when the FER values are low (i.e. $FER \leq 0.1$). This difference is due to the *flappiness* effect of the former case, which under-utilizes the medium, thus yielding a lower aggregated bandwidth. On the other hand, we can appreciate, in both mechanisms, that the error rate goes up, the first path ($S_1 \rightarrow R_1 \rightarrow D_1$) gradually increases its load (bounded by the legacy TCP ideal performance), showing that a percentage of the traffic (in principle belonging to the second path) is successfully delivered through the first one, thus partially mitigating the performance loss.

5.2. Performance of multipath routing algorithms and MPTCP over random topologies

In this second stage, and as a previous step to carrying out the performance analysis of the MPTCP protocol, we used a proprietary software to analyze the operation of different multipath routing approaches presented in Section 4. In particular, the tool takes the following operation: (1) deploy the nodes within the scenario, (2) execute the three multipath routing algorithms and (3) generate the output files that will be afterwards used on `ns-3` to perform the corresponding simulation campaign. We have established a set of aspects to be considered:

- The nodes will be deployed within a 100×100 meters squared area.
- Initially, disconnected graphs are discarded; i.e. only scenarios in which there is, at least, one path between any pair of nodes.
- In this work we do not consider node mobility, so nodes stay static during the simulation time.

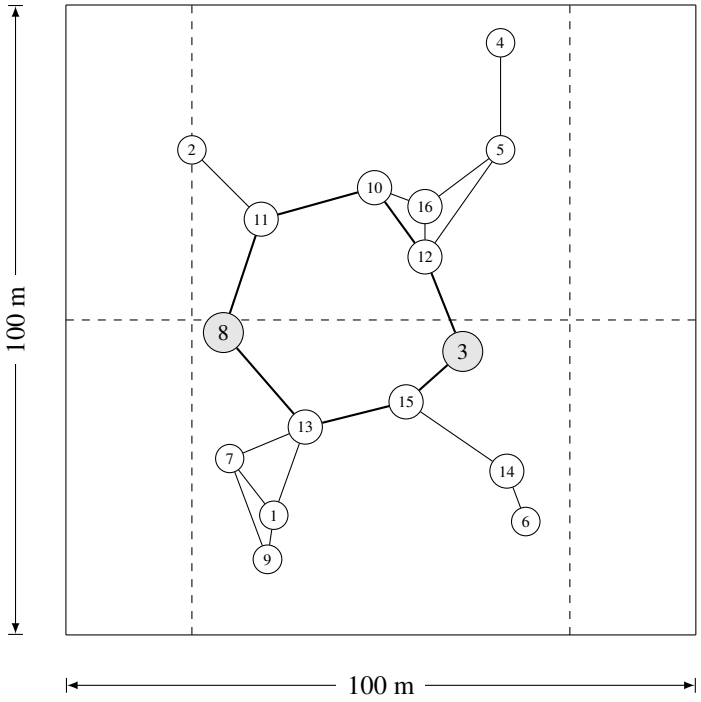


Figure 6: Illustrative topology (16 nodes)

- The coverage area of the nodes (disk radius model) is 20 meters.
- The source-destination nodes are selected so as to ensure same consistency to the multipath routes; by taking two points $(20, 50)$ and $(80, 50)$ as references (as shown in Fig. 6); we select the source as the closest one to the first point and the receiver the closest to the latter reference point.

As an illustrative example, Fig. 6 shows a random deployment of 16 nodes. In this particular topology we see that node 8 will take the transmitter role and node 3 will be the receiver. With regard to the selected routes, the three algorithms will provide the same result: the shortest path is $8 \rightarrow 13 \rightarrow 15 \rightarrow 3$, while the second option would be, for the three algorithms, $8 \rightarrow 11 \rightarrow 10 \rightarrow 12 \rightarrow 3$.

5.2.1 Multipath algorithms characterization over random topologies

Fig. 7 shows the percentage of multipath "feasible" topologies (those which there were two or more disjoint paths divided by the total number of runs)[7] as a function of the number of nodes.

[7]The experiment consisted in 1000 independent runs.

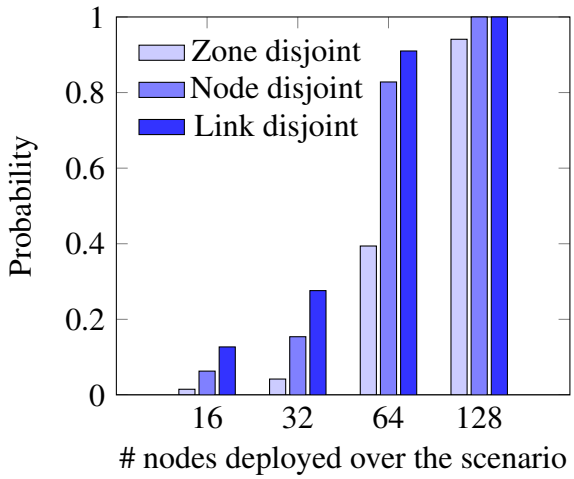


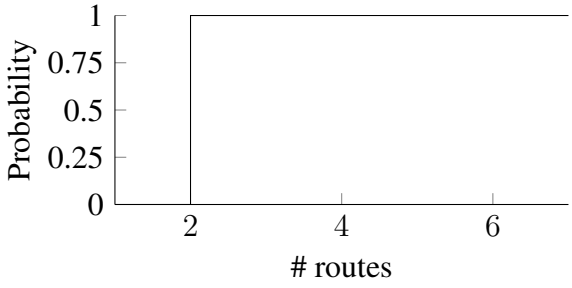
Figure 7: Probability of finding a multipath strategy using the different routing algorithms

We can appreciate that *LD* always exhibits the best behavior, closely followed by *ND*; on the other hand, *ZD* appears as the most restrictive alternative.

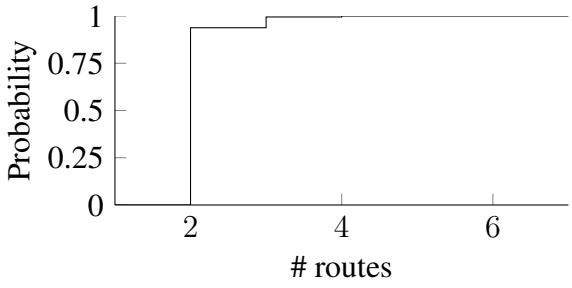
After the first comparison, a new constraint was added. Only those topologies with, at least, two different routes (for the three routing algorithms) were considered. We used 32 nodes (all of them fulfilling the previous constraints) and generated 1000 scenarios. It is worth mentioning that, to get such a high number of deployments, many other scenarios were discarded, since, as shown in Fig. 7, only 4.2% of the 32-node scenarios were multipath for the ZD algorithm (i.e. ZD found two or more paths between the two edge nodes).

First, Fig. 8 shows the *cdf* of the total number of routes found by each of the studied schemes. As could have been expected, *LD* is the algorithm which provides the higher number of alternative paths, since it is the scheme which makes fewer changes to the graph between successive iterations. *ND* appears the intermediate solution, showing a non-negligible probability to discover three disjoint paths. On the other hand, the strong constraints imposed by *ZD* avoids finding more than two simultaneous routes.

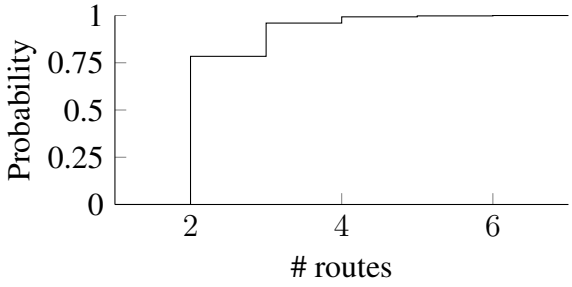
Another insightful metric is the *cdf* of the number of hops of the two preferred routes, shown in Fig. 9. As we can infer from the discussion given in Section 4, the shortest path (1st iteration) is the same for all the schemes, since all of them use the Dijkstra's algorithm to find it. However, the second alternate route length shows the same behavior as the previous statistic: *LD* finds, in the second iteration, the shortest path to reach the destination; *ND* appears again as the solution with the second shortest route, being *ZD* the scheme providing the longest paths. This is of outer relevance, since the number of hops will have a remarkable influence on the aggregated performance, since the greater the length of these paths the lower the throughput of the corresponding subflow.



(a) Zone disjoint



(b) Node disjoint



(c) Link disjoint

Figure 8: *cdf* of the number of found routes for the different algorithms

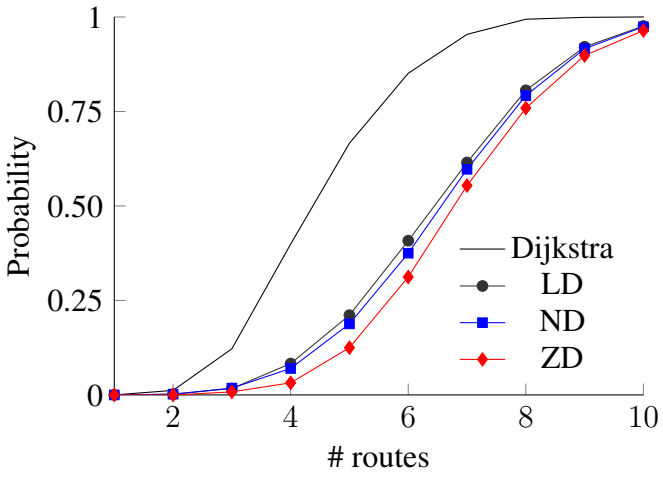


Figure 9: Number of hops *cdf* for the two preferred routes

5.2.2 MPTCP protocol behavior over packet erasure channels

Next to the analysis of the three different multipath routing algorithms over random deployments, we will now describe the simulation campaign, carried out again over `ns-3`. We use the outcome of the previous analysis, and nodes are connected by means of IEEE 802.11 links. It is worth recalling that we have ensured that all the simulated scenarios have, at least, two disjoint routes for each of the algorithms (we discarded these topologies not fulfilling this requirement). The input of this stage is the output of the previous one, in particular the following pieces of information: (1) the location of the nodes, and (2) the routes returned by the *LD*, *ND* and *ZD* algorithms (P_L, P_N and P_Z, respectively).

As well as in the previous experiment, we perform the simulation testbed by means of three different transport level solutions: single-path TCP, single interface MPTCP and multi-interface MPTCP.

Besides, it is worth mentioning some additional aspects about the simulation setup:

1. The main scenario setup keeps the same parameters as the previous campaign: one single 20 MB transmission, saturation conditions, IEEE 802.11b links between the nodes, four transmissions attempts per packet (IEEE 802.11 retransmission scheme), etc.

2. Since an external process is used to obtain the routes with the three algorithms, the routing scheme is based on static routes.

3. There is a single cause of packet losses: the collisions between simultaneous node transmissions. We will consider ideal channels, where the frame losses rate due to the wireless propagation effect is null.

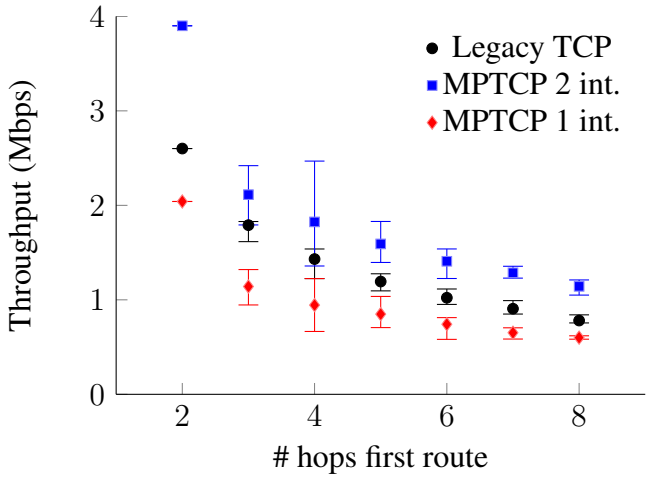


Figure 10: Average throughput as a function of the number of hops of the first route

Fig. 10 shows the overall performance for the three schemes[8]. It represents the average, maximum and minimum values of throughput as a function of the number of hops used by the shortest path. We can clearly appreciate the improvement brought about by using the two channel scenario, achieving a higher aggregated throughput (e.g. 50% for the 2-hop scenario), compared to the traditional single-path TCP. However, we can find few cases with a lower performance, corresponding to these situations in which the second path needs many more hops than the first one. On the other hand, we can appreciate the limitation shown by the multipath strategy over a single channel WMN, since the contention caused by the high number of nodes contending for the channel leads to long idle times and a high probability of collision. The consequence is that the overall performance is quite lower than the one observed by the legacy TCP.

6. Conclusions and future work

The latest changes and evolutions regarding networking trends (i.e. multi-RAT mobile devices, multi-homed servers, data centers with redundant paths, etc.) have induced the research community to pose the following question: *Is the Internet becoming multipath?*

In order to face this new paradigm, several solutions have sprung to deal with the inherent challenges of carrying out simultaneous transmissions over different paths, possibly showing rather different physical attributes (i.e. packet error probabilities, transmission bit rates, etc.). MPTCP outstands as one of the most relevant solutions at the transport level. It can provide a secure and reliable connection using multiple subflows.

[8]We only report the results achieved with the routes provided by the LD algorithm, which correspond to a better performance.

In this work we have ported the MPTCP framework (initially developed by Chihani et al. [4]) to a newer version of the `ns-3` (namely, its 13th iteration). It implements a fully-fledged transport layer, as specified in its architectural guidelines [7], completely transparent to the upper layers and primary focused on the fulfillment of the following three main goals: *improve throughput*, *do not harm* and *balance congestion*.

It is worth highlighting that the vast majority of the existing works focus on the performance assessment of these techniques over wired scenarios. Opposed to this, we have studied the impact of wireless communications over MPTCP, since it is well known the remarkable degradation suffered by TCP when operated over error-prone wireless links.

Over a two-stage simulation campaign, we have firstly conducted a thorough assessment of the behavior of MPTCP a simple "diamond" topology. First, we have compared the four different congestion control algorithms by which might be used: *Uncoupled*, *Linked increases*, *RTT compensator* and *Fully Coupled*. Under ideal conditions, the first three schemes behave quite alike, fairly dividing the bandwidth between the two subflows. On the other hand, the latter one suffers from the "flappiness" effect, and most of the resources are alternatively taken by one of the subflows, reducing the aggregated throughput. Additionally, we introduced packet losses within the links, comparing the performance achieved by MPTCP over two different configurations to the traditional single-path TCP. We have observed that MPTCP outperformed TCP up to ~48% over ideal and non-interfering channels. Finally, we have assessed the MPTCP capability to balance the subflow load upon a negative situation. To this end, we have modified the error pattern so that just one of the paths was prone to loose packets. As a result, we verified the capability of different MPTCP configurations to offload as much traffic as possible towards the less damaged subflows, minimizing the performance degradation.

In a second stage, we have presented three different algorithms (LD, ND and ZD) which were used to obtain the best set of disjoint paths over generic WMN topologies. We have focused on the use of multipath strategies over WMNs. We have compared their performance, in terms of feasibility (probability that there are two or more paths in a scenario), number of discovered paths and route length required to reach the destination node in such paths. According to the achieved results, the *ZD* algorithm seems too restrictive for the search of multiple disjoint paths. Afterwards, using the outcomes of this first stage (the node deployment and the different routes between the source and the destination nodes), we compared the performance offered by the *MPTCP* protocol to the one exhibited by the legacy TCP, by means of another `ns-3` simulation, leading to improvements of about $\sim 50\%$ in some of the cases.

After this work, there are still a number of challenges and open issues that shall be addressed in the future, as the ones which are highlighted below.

- Assess the impact of realistic wireless channel models over the MPTCP performance, since the ones supported "by default" by the simulator are not able to appropriately reflect the memory of a real wireless link. Several works have demonstrated that this sort of channels severely impacts the performance of connection-oriented protocols [27, 28].

- Analyze different routing schemes, exploiting the presence of multi-channel devices by means of appropriate graph-theory models, as the one proposed by Yang *et al.* in [29].
- Finally, it would be interesting to add mobility to the nodes, so as to study other additional statistics, such as path stability and duration.

Last, but not least, it is worth highlighting that all our work follows the Generic Public License (GPL) requirements and the source code and the corresponding documentation is fully available in [30].

Acknowledgements

The authors would like to express their gratitude to the Spanish government for its funding in the project "Connectivity as a Service: Access for the Internet of the Future", COSAIF (TEC2012-38574-C02-01).

References

[1] L. Magalhaes and R. H. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Network Protocols, 2001. Ninth International Conference on*, Nov. 2001, pp. 165–171. [Online]. Available: http://dx.doi.org/10.1109/icnp.2001.992896

[2] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A Measurement-based Study of MultiPath TCP Performance over Wireless Networks," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 455–468. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504751

[3] C. Paasch, R. Khalili, and O. Bonaventure, "On the Benefits of Applying Experimental Design to Improve Multipath TCP," in *Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, December 2013, available from http://dl.acm.org/authorize?6960036.

[4] B. Chihani and D. Collange, "A multipath TCP model for ns-3 simulator," *CoRR*, vol. abs/1112.1932, 2011.

[5] "The ns-3 network simulator," http://www.nsnam.org/.

[6] N. Meghanathan, "Performance Comparison of Link, Node and Zone Disjoint Multi-path Routing Strategies and Minimum Hop Single Path Routing for Mobile Ad Hoc Networks," *CoRR*, vol. abs/1011.5021, 2010. [Online]. Available: http://dx.doi.org/10.5121/ijwmn.2010.2402

[7] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, " TCP Extensions for Multipath Operation with Multiple Addresses," *RFC*, no. 6824, January 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6824.txt

[8] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural Guidelines for Multipath TCP Development," RFC 6182 (Informational), Internet Engineering Task Force, Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6182.txt

[9] C. Raiciu and M. H. D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols," *RFC*, no. 6356, January 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6356.txt

[10] G. Hampel and T. Klein, " Enhancements to Improve the Applicability of Multipath TCP to Wireless Access Networks," individual, IETF Internet Draft – work in progress 00, June 2011. [Online]. Available: http://tools.ietf.org/html/draft-hampel-mptcp-applicability-wireless-networks-00

[11] R. Stewart, "Stream Control Transmission Protocol," RFC 4960 (Proposed Standard), Internet Engineering Task Force, Sep. 2007, updated by RFC 6096. [Online]. Available: http://www.ietf.org/rfc/rfc4960.txt

[12] "MPTCP - Linux Kernel implementation." [Online]. Available: http://mptcp.info.ucl.ac.be/

[13] M. Lim and J. Valdez, "MPTCP Wireless performance," http://reproducingnetworkresearch.wordpress.com/2012/06/06/mptcp-wireless-performance-draft/.

[14] C. Raiciu, D. Wischik, and M. Handley, "Practical congestion control for multipath transport protocols," *UCL Technical Report*, no. 6824, January 2009.

[15] S. C. Nguyen and T. M. T. Nguyen, "Evaluation of multipath TCP load sharing with coupled congestion control option in heterogeneous networks," in *Global Information Infrastructure Symposium (GIIS), 2011*, 2011, pp. 1–5. [Online]. Available: http://dx.doi.org/10.1109/giis.2011.6026698

[16] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "Mptcp is not pareto-optimal: Performance issues and a possible solution," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 5, pp. 1651–1665, Oct 2013. [Online]. Available: http://dx.doi.org/10.1109/tnet.2013.2274462

[17] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626 (Experimental), Internet Engineering Task Force, Oct. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt

[18] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), Internet Engineering Task Force, Jul. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[19] N. Meghanathan, "Stability and Hop Count of Node-Disjoint and Link-Disjoint Multi-path Routes in Ad Hoc Networks," in *Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on*, Oct 2007, pp. 42–42. [Online]. Available: http://dx.doi.org/10.1109/wimob.2007.4390836

[20] S. Waharte and R. Boutaba, "Totally disjoint multipath routing in multihop wireless networks," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 12, 2006, pp. 5576–5581. [Online]. Available: http://dx.doi.org/10.1109/icc.2006.255550

[21] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ser. ATEC '04. Berkeley, CA, USA: USENIX Association, 2004, pp. 8–8.

[22] H.-Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts," in *Proceedings of the 8th annual international conference on Mobile computing and networking*, ser. MobiCom '02. New York, NY, USA: ACM, 2002, pp. 83–94. [Online]. Available: http://dx.doi.org/10.1145/570645.570656

[23] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Internet Engineering Task Force, Sep. 1981, updated by RFCs 1122, 3168, 6093. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[24] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP," RFC 2883 (Proposed Standard), Internet Engineering Task Force, Jul. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2883.txt

[25] D. Wischik, M. Handley, and M. B. Braun, "The resource pooling principle," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 5, pp. 47–52, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1145/1452335.1452342

[26] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath TCP," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 29–29. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228338

[27] R. Agüero, M. García-Arranz, and L. Muñoz, "Accurate simulation of 802.11 indoor links: a bursty channel model based on real measurements," *EURASIP J. Wirel. Commun. Netw.*, vol. 2010, pp. 16:1–16:12, April 2010. [Online]. Available: http://dx.doi.org/10.1155/2010/380410

[28] M. Zorzi, A. Chockalingam, and R. Rao, "Throughput analysis of TCP on channels with memory," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 7, pp. 1289–1300, 2000. [Online]. Available: http://dx.doi.org/10.1109/49.857929

[29] Y. Yang, J. Wang, and R. H. Kravets, "Interference-aware load balancing for multihop wireless networks," no. UIUCDCS-R-2005-2526, 2005. [Online]. Available: http://www.ideals.uiuc.edu/handle/2142/10974

[30] "Source code and documentation of the MPTCP implementation (ns-3.13)," https://github.com/dgomezunican/multipath-ns3.13.

Copyright Disclaimer