UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

MultiPath TCP Connections : Analysis and Models

Dissertation presented by
Rémi CHAUVENNE , Thibault LIBIOULLE

for obtaining the Master's degree in
Mathematical Engineering, Computer Science and Engineering

Supervisor(s)
Olivier BONAVENTURE

Reader(s)
Quentin DE CONINCK, Matthew PHILIPPE

Academic year 2016-2017

Acknowledgment

This Master's thesis would not have been as it is without the precious help of many persons. We would like to present our best thanks to those persons.

In particular, we would like to thank our supervisor, Pr. Olivier Bonaventure, and our assistant, Ir. Quentin De Coninck, for their availability, their patience and their continuous questionning which helped us to progress.

The MultiPath TCP team of the INL department for their kindness and the help they gave when we needed it.

All the researchers in the MultiPath TCP and TCP fields who work hard, day and night, to make TCP great again.

All the music artists who accompanied us during our days of work.

A special thank to the INGI sysadmin for the provision of a dedicated computer and its maintenance and to Malian De Ron for his useful layout template.

Finally, we would like to thank our friends, families and roommates who support us without any limit.

Abstract

MultiPath TCP is a recent TCP extension that allows the spreading of data on several paths. While the behavior of TCP mainly depends on its congestion control scheme, MultiPath TCP presents more subtle patterns. In order to better understand its behavior, this work first proposes broad measurements and analyzes of a large number of experiments. We compare the performance of MultiPath TCP with those of TCP with a new metric that we introduce and we observe how these performances evolve with different factors. We consider three kinds of connections : bulk-transfers, small files and lossy transfers. The second part of this work is a mathematical approach where a TCP throughput stochastic model is extended for MultiPath TCP. We propose a simple formula for the throughput and validate it through experiments.

Contents

1 Introduction

MultiPath TCP is a recent TCP extension that allows hosts to spread data over several paths. While the dynamic of TCP is mainly dependent on the congestion control scheme, MultiPath TCP exhibits more complex behaviors due to the mechanisms it implements. Especially, one can cite the scheduler that is responsible of the path selection and the congestion control schemes that are now coupled.

Previous works have looked at the behavior of such connections [49]. Some of them propose mathematical models [30, 41]. However, it is unclear if these models provide an easy and efficient way of predicting MultiPath TCP throughput.

In comparison, TCP has several well-known models for its behavior [32, 40]. One of them is based on stochastic processes and provides a simple way to compute the throughput knowing the loss rate and the round-trip time of the path [40]. We wish to achieve similar results for MultiPath TCP.

To fill this gap, this master's thesis first provides various measurements and analyzes. We use systematic approaches to test MultiPath TCP connections in large sets of experiments. The goal is to identify cases where it performs well or badly compared to TCP and the factors responsible of observed performances. Once the behavior has been explored, we provide a mathematical model of MultiPath TCP of one of the main congestion control scheme.

The core document is organized in five chapters that can be divided into three different fields. The first field presents various state of the art methods. The second regroups all our experiments and analyzes. The last field is about mathematical modeling.

More precisely, Chapter 2 introduces the necessary background for the remainder of the document. This includes details on the main components of MultiPath TCP, literature models for both TCP and MultiPath TCP and more general hints on how to experiment. This constitutes our state of the art.

Chapter 3 is dedicated to our first results and analyzes. We transfer large files during long-run connections with different applications. We introduce a new metric and our test setup. Chapter 4 focuses on the exchanges of small-files that might be more representative of actual Internet traffic [18]. We test the impact of the scheduler on such exchanges. Chapter 5 provides our last measurements. We evaluate the different congestion control schemes of the literature in lossy environments such as what might be experienced when using Wi-Fi. We also evaluate their fairness and their choice of efficient paths in more general congested

environments.

Chapter 6 brings the final touch. We extend the TCP stochastic model for MultiPath TCP. The model provides a simple formula for throughput that only depends on the loss rates and on the round-trip times of the paths connection. We validate our model on several tests.

Finally, Chapter 7 concludes this document. We summarize the main concepts of our work and we propose ideas for future work.

State of the Art 2

2.1 MultiPath TCP

In this section, we provide the necessary background about MultiPath TCP for this document. This includes both theoretical and practical information, with particular importance attached to observed behaviors. We assume basic understanding of TCP's principles. More details can be found in [22] for TCP and in [26, 35] for MultiPath TCP.

2.1.1 Overview

MultiPath TCP is an ongoing effort that aims at extending regular TCP to enable a transport connection to use multiple paths [26]. *Subflows*, which are basically flows of TCP segments, can be created for each path and thus a single *MultiPath TCP session* can allow multiple subflows to be used [14]. Implicitly, MultiPath TCP is supposed to improve performance, load-balancing and robustness of connections [14].

The design goals are [43] :

1. To use the network resources at least as well as TCP, without starving it ;
2. To be transparent for existing applications ;
3. To work in all scenarios where regular TCP works ;
4. To be implemented without using excessive memory or processing.

The first design goal ensures *fairness* to regular TCP users. To do so, MultiPath TCP uses *congestion control scheme* to limit its throughput in function of the network conditions [43]. We can thus expect the behavior of such connections to be dependent on the chosen scheme.

The Linux implementation architecture of MultiPath TCP is represented in Figure 2.1. One can observe that only the transport layer has been modified, such that applications can still use the standard Socket API. This fulfills the second design objective.

At the transport layer, an MultiPath TCP session starts with an *initial subflow*, then additional subflows can be added to the session along each path. All these subflows are equivalent to regular TCP connections, which allows the solution to be deployable on today's Internet and thus fulfills the third design goal [14, 35]. This also implies that each subflow has its own

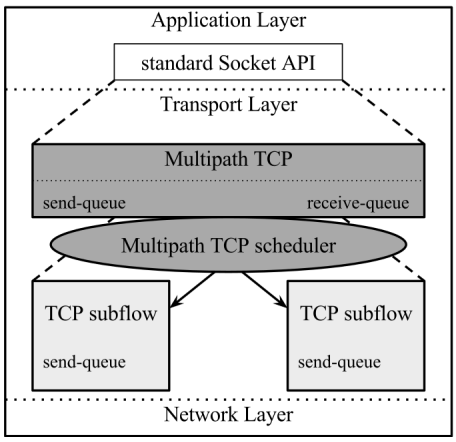


Figure 2.1: Architecture of MultiPath TCP Linux implementation (Source : [35])

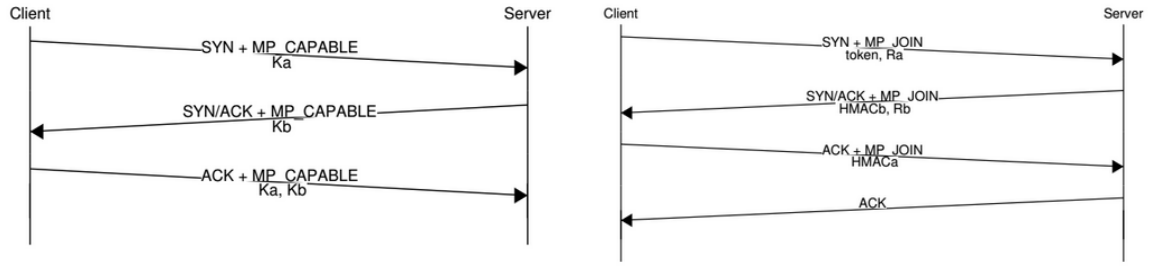
sequence number space to detect losses and to provide retransmissions mechanism. On top of that, MultiPath TCP owns a connection-level sequence numbers to allow reassembling the segments coming from different subflow with different network delays [26].

Once the connection has been established, data can be sent over any of the active subflows. The role of the *MultiPath TCP scheduler* is to choose on which one to send the data such that we pool resources in a efficient way while providing resistance to link failures. Bad scheduling decision can have negative effects on the connection performance, so one need to choose it carefully [37]. In the 0.91 version of the Linux implementation, one has the following choice [15] :

- *default* : It starts by sending data on the subflow with the lowest RTT until its congestion window is full. Then, it transmits on the second lowest RTT subflow, *et cetera* ;
- *roundrobin* : It sends a number of segments, which is configurable, on each subflow. The order on which subflows are chosen is constant and forms a circle. It has been shown to be particularly bad in terms of application delay [37] ;
- *redundant* : It duplicates the traffic on all the subflows. It is useful when one wants to achieve minimum latency and jitter.

Let us now take a look at the different MultiPath TCP operations. We consider an example where a client has two interfaces and a server that is single-homed. The establishment of a MultiPath TCP connection starts with a SYN segment sent by the client over one of its interfaces. This segment contains the MP_CAPABLE TCP option, which indicates that the client supports MultiPath TCP. It also contains a key chosen by the client. The server responds with a SYN+ACK segment containing the MP_CAPABLE option and a key chosen by

itself. The server concludes the establishment of the MultiPath TCP connection by sending back a ACK segment [14].



(a) Connection establishment (b) Add a new subflow

Figure 2.2: MultiPath TCP operations (Source : [20])

The client and the server can exchange data, but the client now wishes to use both of its available interfaces. So it restarts a connection establishment on its other interface, this time with a MP_JOIN option in the first SYN segment. In the same manner, the server replies with MP_JOIN option in a SYN+ACK segment. Finally, the client responds with a ACK segment and the new subflow is finally created. From a security point of view, the server needs to authenticate the client to be sure that the new subflow will be associated with the same client. At this end, HMAC can be computed using the keys that has been exchanged in the connection establishment and 32 random bits exchanged while adding the new subflow to the connection.

Subflows can also be removed using the REMOVE_ADDR option, where ADDR makes reference to the local IP address ADDRESS-ID associated with the subflow. The peer will be then informed that it should not send any data on this subflow anymore [36]. Finally, one can also backup a subflow that aims at avoiding it to send data, using the MP_JOIN or MP_JOIN option and a *backup bit* that indicates that the subflow should not be used anymore. It can be violated in the case where no more subflow is available [36]. The backup can be useful when considering monetary cost for example.

The *path manager* is the structure responsible for the subflows control, it decides when and how subflows are created. They are 3 main modes in the 0.91 version of the Linux implementation, which are [15] :

- *fullmesh* : It provides a full-mesh of subflows among the available addresses ;
- *ndiffports* : It creates X subflows along a same pair of IP-addresses ;
- *binder* : It uses the Loose Source Routing principle, i.e. it maps each subflow within

the community network to a unique gateway [13].

Finally, MultiPath TCP implements a new concept called *reinjection*. It allows data to be sent over more than one subflow, which can be useful when we loose a subflow and that we want to retransmit lost data over an other subflow.

2.1.2 Congestion Control Schemes

As introduced previously, the congestion control scheme limits dynamically the throughput of a subflow in case of congestion on it. The choice of the scheme will thus have an impact on the behavior of a MultiPath TCP connection and it is therefore relevant to give more details on state of the art methods.

A first naive solution for the congestion control on multiple subflows would be to use regular TCP congestion control independently on each subflow, this is called *uncoupled congestion control*.

However, this would cause a problem of *unfairness* with regular TCP [49]. Indeed, one can take the figure 2.3 as an example, where two clients share a single link. One of the client uses MultiPath TCP with two interfaces, while the other only uses TCP. Using an uncoupled congestion congestion control scheme, $\frac{2}{3}$ of the bandwidth would be used by the MultiPath TCP and $\frac{1}{3}$ by the TCP user, which is unfair. In this case, it is fair if both share $\frac{1}{2}$ of bandwidth.

Figure 2.3: Fairness of coupled congestion control scheme

We thus need *coupled congestion control*, i.e. the dynamic of congestion windows must be dependent of the congestion state of all other subflows. While designing such method, 3 objectives have to be considered [27]. First, it should improve the throughput, i.e. we should perform at least as well as regular TCP on the best available path. Secondly, a subflow should not take more capacity than a regular TCP connection on the same path. Finally, a MultiPath TCP connection should balance congestion by moving traffic away from congested paths.

Version 0.91 of the Linux implementation has four different schemes [15]. The default one is the *Linked Increase Algorithm* (LIA) [49]. Others are *Opportunistic Linked Increase Algorithm* (OLIA) [30], *Balanced Linked Adaptation Congestion Control Algorithm* (BaLIA) [41, 48] and

Delay-based Congestion Control for MultiPath TCP (wVegas) [16]. The three first are loss-based while the latter is delay-based. We now present them in more details.

Loss-based Congestion Control

Loss-based congestion control schemes consider that congestion occurs when they receive loss-indications. Table 2.1 provides a framework that compares the three different schemes which exist in the literature to our knowledge. The first row represents the amount by which the window size must be added when it receives an ACK. The second is the multiplicative decrease when it receives a loss-indication.

	LIA	OLIA	BaLIA
↗	$\min\left(\frac{\max_{i\in\mathcal{R}_u} w_i/\mathrm{rtt}_i^2}{\left(\sum_{i\in\mathcal{R}_u} w_i/\mathrm{rtt}_i\right)^2}, \frac{1}{w_r}\right)$	$\frac{w_r/\mathrm{rtt}_r^2}{\left(\sum_{i\in\mathcal{R}_u} w_i/\mathrm{rtt}_i\right)^2} + \frac{\alpha_r}{w_r}$	$\frac{w_r/\mathrm{rtt}_r^2}{\left(\sum_{i\in\mathcal{R}_u} w_i/\mathrm{rtt}_i\right)^2}\left(\frac{1+\beta_r}{2}\right)\left(\frac{4+\beta_r}{5}\right)$
↘	$w_r/2$	$w_r/2$	$w_r/2\cdot\min(\beta_r, 1.5)$

Table 2.1: Increase (first row) and decrease (second row) phases of loss-based congestion control schemes

LIA In comparison to TCP congestion control, LIA only modify the increase phase of the congestion avoidance mode, which specify how the congestion window must grow when receiving an ACK. For each ACK received on subflow r, we increase the congestion window w_r of subflow r by the amount precised in Table 2.1, where rtt_i is the round-trip time on path i and $\mathcal{R}_u$ is the set of all paths available. The maximum increase is thus $\frac{1}{w_r}$, to be at most as aggressive as TCP on any of the paths.

LIA achieves the first two objectives but not the last one, i.e. it does not achieve full resource pooling. To do so, it would require no traffic to be sent on paths with higher loss rates, which implies that we must couple both the increase and the decrease of congestion windows across subflows. But this would create new problem such as failure to detect free capacity when it becomes available and *flappiness*, i.e. the fact of using only one subflow when two subflows have similar levels of congestion. [49].

OLIA LIA has been shown to be not *Pareto-optimal*, in contrary of OLIA that is an alternative to LIA [30]. We now present in more details the mathematics behind it.

Let us define $l_r(t)$ such that

$$l_r(t) = \max\{l_{1r}(t), l_{2r}(t)\}$$

with $l_{1r}(t)$ the number of bits successfully transmitted over path between the last two losses seen on r and $l_{1r}(t)$ the number of bits successfully transmitted over path after the last loss. For each ACK on path r, OLIA increases w_r by the amount precised in Table 2.1, where α_r is calculated as follows :

$$\alpha_r = \begin{cases} \frac{1/|\mathcal{R}_u|}{|\mathcal{B}\backslash\mathcal{M}|} & \text{if } r \in \mathcal{B}\backslash\mathcal{M} \neq \emptyset \\ -\frac{1/|\mathcal{R}_u|}{|\mathcal{M}|} & \text{if } r \in \mathcal{M} \text{ and } \mathcal{B}\backslash\mathcal{M} \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

and

$$\mathcal{M}(t) = \{i(t) | i(t) = \arg\max_{p \in \mathcal{R}_u} w_p(t)\}$$

the set of the paths of u with the largest window sizes at time t,

$$\mathcal{B}(t) = \left\{ j(t) | j(t) = \arg\max_{p \in \mathcal{R}_u} \frac{l_p(t)}{\mathrm{rtt}_p(t)^2} \right\}$$

the set of the paths at time t that are presumably the best paths for u, as $1/l_r(t)$ is an estimate of packet loss probability on path r at time t.

OLIA has been shown to be as responsive and non-flappy as LIA and solves the problem of Pareto-Optimality. Moreover, it satisfies the three design goals of MultiPath TCP [30].

BaLIA OLIA can be unresponsive to network changes [48]. In fact, there is a trade-off between friendliness to single path TCP and responsiveness to the network changes. BaLIA has been designed in order to balance this trade-off.

Unlike LIA and OLIA, both increase and decrease phases are modified in comparison with TCP, as shown in Table 2.1. β_r is computed as follow :

$$\beta_r = \frac{\max_i(w_i/\mathrm{rtt}_i)}{w_r/\mathrm{rtt}_r}$$

Delay-based Congestion Control

Delay-based congestion control schemes consider that congestion occurs when the round-trip times grow. wVegas is the only scheme that is delay-based to our knowledge.

wVegas wVegas [16] uses the same decrease of the congestion avoidance phase, the fast retransmit and the fast recovery algorithm as TCP. It mainly differs from the precedent schemes in the way it enters in the congestion avoidance phase and the its ability to drain the link queues. First it adds a chance to enter it earlier which is a behavior inherited from TCP-Vegas. Secondly, the goal of wVegas, once it detects that the queueing delay is larger than some threshold, is to add a chance to obtain the more accurate propagation delay.

The queueing delay is calculated as $\text{queue\_delay}_r = \text{rtt}_r - \text{base\_rtt}_r$ and a the minimum value of this queueing delay is saved. Once a queue_delay_r is two time larger than the saved queueing delay, the congestion window is adapted as follows :

$$\text{cwnd}_r = 0.5 \cdot \text{cwnd}_r \cdot \frac{\text{base\_rtt}_r}{\text{rtt}_r}.$$

2.1.3 From Theory to Practice

As the theoretical aspects of MultiPath TCP have been described, one can turn it on practice and implement it on Linux Kernel. Such an implementation is described in [11] and is still subject to improvements, see [15].

The temptation is now high to evaluate and analyze the behaviors of MultiPath TCP on Linux Kernel. In the next subsections, we will discuss about how to realize experiments on MultiPath TCP, what we have to take into account and finally what kind of understanding it could provide.

The inherent objective of those experiments and evaluations is to *develop a basic understanding* : we try to obtain an overview of the system behaviors, to detect problems and to confirm expectations.

Design of the Experiments

In the scientific papers, there are lots of way to design experiments. Some are interested in a real-world experiment while the other could use a specific network designed only for a MultiPath TCP analysis purpose. Each experiment has particularities, some underlying goals and controlled or uncontrolled factors which influence it.

We will discuss here about 3 ways to experiment MultiPath TCP :

- *Testbeds* : Experiments are performed on a static network infrastructure and allows *replicable and rigorous* experiments. Using this testbeds, it is possible to vary different parameters :

  - The application which exchange data on the network (i.e. bulk transfer or rate-limited transfer);
  - The network parameters (such as the *Maximum Transmission Unit*);
  - The congestion control scheme, the path manager and the scheduler
  - The amount of data transmitted through a connection (by varying maximum file size of the file transfer application or by varying the maximum time for performance assessment) ;

  The testbeds tend sometimes to be really simple, connecting only one host to a server, but could be much more complicated and real-world oriented network, such as the NorNet Edge testbed [8]. These testbeds were used in some early publications [11], but also later [23, 37].

- *Real-World experiments* : As suggested in the name, it tends to simulate a user experiencing different use-cases on his device or to use real traffic. Those use-cases may be determined and fixed *a priori* — to compare results between two different MultiPath TCP-parameters tuning or two different network configurations (record-replay [21], bulk transfer [42]) — or the use-cases may be randomized, only directed by a human experience, such as in [19]. These kind of experiences do not always guarantee to be replicable but offers a good overview of what is experienced by an application user.

  As for the first type of experiences, we can vary some parameters such as the path-manager option, the scheduler, the congestion control, *etc.* But we can not easily modify network parameters.

- *Experimental design* : The *experimental design* is defined as the process of executing controlled experiments in order to collect information about a specific process or system [24]. A network protocol could be experimented in a system influenced by different controlled and uncontrolled factors and responds to it according to the laws of nature [38]. In computer science, this kind of statistical approach is not a main stream fashion, because of the predominance of determinism in the execution of an algorithm.

  As someone design such an experiment, he has to decide which factors he takes into account, whether it does or it does not influence the output of the experiment. In [38], the factors chosen are bandwidth limitations, propagation delay, maximum queuing delay, loss rate and congestion control. Different values of the factors are generated by a space filling approach and implemented here through the *WSP algorithm* described in section 2.3.

  Further than just *Developing a Basic Understanding*, the other objectives this method achieves are [31] :

– *"Finding Robust Decisions or Policies"* : If a specific output is desired, a goal of this method is to provide a correct configuration to produce it, even if uncontrollable factors influence the output ;

– *"Comparing Decisions or Policies"* : Given a specific set of factors, be able to estimate the behavior of the system.

Those approaches may be cross-checked and give different interpretations. Once an experiment outputs a result, we have to evaluate it and give meaning to the result.

Evaluations

Before launching and analyzing the experiments, the definition of the measures used to evaluate transmissions, and moreover MultiPath transmissions, must be given.

Common measures used to evaluate single-path connections are goodput or throughput. We may use those measures to evaluate MultiPath connections. With those measures, one can compare the throughput of a file download, using `curl`, or the throughput achieved in a given time, using `iperf`, achieved by either TCP or MultiPath TCP connections.

However, in real-world experiments, one could just want to know how the MutliPath TCP connection fulfills the available bandwidths. Such a measure can be performed and is called the *Aggregation Benefit* [29].

This measure takes value from -1 to 1 such as :

- $A_B = 1$: MultiPath TCP perfectly aggregates the capacity of all paths ;
- $A_B = 0$: MultiPath TCP performs as good as the path with the highest capacity
- $A_B = -1$: MultiPath TCP achieves a goodput of zero.

Mathematically, let S be a multipath aggregation scenario, with n paths. C_i is the capacity of the path i and C_{max} the highest capacity among all paths. For a goodput, g, on the MultiPath TCP connection, the aggregation benefit A_B is given by [29] :

$$A_B = \begin{cases} \frac{g - C_{max}}{(\sum_{i=1}^{n} C_i) - C_{max}} & \text{if } g \geq C_{max} \\ \\ \frac{g - C_{max}}{C_{max}} & \text{if } g < C_{max} \end{cases}$$

One can also evaluate the number of retransmissions and reinjections, the evolution of the congestion window, the distribution of the packets through different paths, etc, such as in

[19], [37] and [21]. The consistency of those measures lies in the interpretation and analysis these suggest.

We can now go to the most important point of the practical use of MultiPath TCP, the analysis of its behavior.

Analysis

Our analyzes must respect a scientific procedure and the study of the sensitivity, the averaging of multiple cases must be done to do not cope with inconsistent results and analyze the central tendency [38]. Thus a sensitivity analysis will be executed for each experiment.

The analysis of the results may lead to some validations of expectations, lead to the emphasis of problems or suggest an improvement. The experiments, their evaluation and their analysis are key roles in the understanding of the MultiPath TCP general behavior.

2.2 Mathematical Models of TCP and MultiPath TCP

We now present some state of the art mathematical models for both TCP and MultiPath TCP.

2.2.1 TCP

The behavior of regular TCP mainly depends on the congestion control scheme [33]. *Padhye et al.* introduced a stochastic model of the TCP-Reno protocol for bulk transfer where throughput at steady state is expressed as a function of loss probability and round trip time. It has been shown to catch several TCP behaviors and to provide good estimate [40].

Other models [32, 34] have been presented before but they do not perform as well as Padhye's one. Indeed, they do not catch the effect of timeout mechanism on the throughput, and therefore overestimate it [40].

These models are only available in the case of long transfer stream. In [17], they extend Padhye's model to catch start-up effects such as connection establishment and slow start. It might therefore be interesting for small connections.

As the model introduced by Padhye et al. is extended in Part 6 of this document for multipath flows, we present some details of it.

Padhye et al. Model

There are two variations of the model. The first only considers losses that are triple-duplicates indications and the second that also considers time-outs indications. We only present the first one. More details can be found in [40].

Let us first define a *TD period* to be a period between two losses that are triple-duplicates (TD) indications. We also define a *round* that represents the sending of a whole congestion window and the reception of the corresponding ACKs. For the i-th TD period, we note Y_i the number of packets sent in the period, A_i the duration of the period, and W_i the window size at the end of the period. Considering $\{W_i\}_i$ to be a regenerative process with rewards $\{Y_i\}_i$, one can prove that

$$B = \frac{\mathbb{E}[Y]}{\mathbb{E}[A]}$$

where B is the throughput of the TCP connection and $\mathbb{E}$ the expectation operator.

We thus need to find expression for Y_i and A_i. If we consider that all packets in the round are lost after a loss occurs, then we can show that :

$$Y_i = \alpha_i + W_i - 1$$

where α_i denotes the first packet that is lost in period i. The $W_i - 1$ comes from the fact that when a packet is lost, the sender has to wait for the triple-duplicates to arrive. During this time, it can still send a certain number of packets. It follows from the linearity of the $\mathbb{E}$ operator that :

$$\mathbb{E}[Y] = \mathbb{E}[\alpha] + \mathbb{E}[W] - 1$$

α follows a geometric distribution where the parameter of success is the loss rate p. For such distribution, one has :

$$\mathbb{E}[\alpha] = \frac{1}{p}$$

The evolution of the congestion for TCP NewReno between two periods is :

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b}$$

where X_i is the number of rounds in period i and b is related to the principle of delayed ACKs, e.g. if for two packets sent the receiver only sends one ACK then $b = 2$. It follows that :

$$\mathbb{E}[W] = \frac{2}{b}\mathbb{E}[X]$$

One can derive another expression for Y_i :

$$Y_i = \sum_{k=0}^{X_i/b-1} (\frac{W_{i-1}}{2} + k)b + \beta_i = \frac{X_i}{2}(\frac{W_{i-1}}{2} + W_i - 1) + \beta_i$$

where the last equality has been found using the expression of W_i. β_i is defined as the number of packets sent in the last round.

Making the approximation that $\{X_i\}$ and $\{W_i\}$ are are mutually independent sequences of i.i.d. random variables and that $\mathbb{E}[\beta] = \mathbb{E}[W]/2$, we find that :

$$\mathbb{E}[W] \approx \sqrt{\frac{8}{3(b \cdot p)}}$$

Finally we have to find the duration of a period i, i.e. A_i. The duration is directly proportional to the number of rounds :

$$\mathbb{E}[A] = (\mathbb{E}[X] + 1) * RTT$$

And so, the throughput is :

$$B \approx \frac{1}{RTT}\sqrt{\frac{3}{2(b \cdot p)}}$$

2.2.2 MultiPath TCP

The behavior of MultiPath TCP is expected to be more complex as the congestion control schemes are generally coupled through the several paths. Moreover, there is the scheduler on top. Few models seem to exist and it is unclear if they provide good estimate of actual MultiPath TCP throughput.

One can quote [41] that provides a fluid model that covers a broad class of MultiPath TCP algorithms. In [30], *Khalili et al.* gives a formula for the congestion windows at steady-state. We present the latter now as it provides similar result to the Part VI of this document.

Fixed-Point Analysis

A fixed-point analysis has been proposed in [30] about the congestion control scheme LIA and leads to a formula for the congestion window w_r of path r. It is based on the principle of fixed-point of [49], which states that at steady-state the rate of ACKs × average increase per ACK must equal rate of drops × average decrease per drop.

For similar RTTs, the increase phase of LIA is a/w_{tot} per received ACK, where a has been defined in Section 2.1.2. Then the fixed-point is :

$$\left(\frac{w_r}{RTT_r}(1 - p_r)\right)\frac{a}{w_{tot}} = \left(\frac{w_r}{RTT_r}p_r\right)\frac{w_r}{2}$$

where p_r is the loss rate of path r. Then one find :

$$w_r = \frac{1}{p_r} \cdot \frac{\max_p \sqrt{2/p_p}/rtt_p}{\sum_p 1/(rtt_p p_p)}$$

which means that LIA allocates a congestion window that is inversely proportional to the loss rate and such that the sum of throughputs is equal to the throughput a TCP flow would get with the same loss rate. Indeed, the throughput can be approximated by : w_r/rtt_r and the TCP throughput by $\sqrt{2/p_r}/rtt_r$ [30].

Generally speaking, there seems to be a need for modeling and evaluating MutliPath TCP behavior.

2.3 Space-Filling

When using experimental design, one wishes to define experiments such that we maximize the cover of the parameters space while maximizing the distance between 2 experiments. This is called *space-filling*. Well-know space-filling is *Latin Hypercube* but it suffers from the *curse of dimensionality*, i.e. it becomes harder to fill the space as we add dimensions, and it needs lots of computational power [47]. WSP algorithm also suffers from the curse of dimensionality, but a small extension, Adaptive WSP algorithm, solves this problem [12]. We now present the WSP algorithm.

2.3.1 WSP Algorithm

When the relation between inputs and outputs is not known, we want to design experiments such that points at which the response is observed are evenly distributed throughout the region. This is called a space-filling design.

One way to achieve this is to used the Wootton, Sergent, Phan-Tan-Luu's algorithm (WSP). It provides a high dimensional experimental design with a lower computations needed than other methods such as Latin Hypercube. The principle is to select points from a set of candidate points so as to be at a preset minimal distance (d_{min}) from every point in the defined multidimensional parameters space. The steps of the algorithm are

1. Generate a set of N candidate points
2. Compute the distances matrix D_{ij} of the N points, where i and j represent point i and point j respectively
3. Choose initial point O and a distance d_{min}

4. Eliminate the points I such as $D_{OI} < d_{min}$
5. Point O is replaced by the nearest point among the remaining points
6. Repeat step 4 and 5 until no more points are eliminated

The criteria used to estimate the quality of the distribution are the minimal distance between any two points and the cover measure. Higher minimal distance and lower value of cover measure implies a more regular distribution. If the size of the initial number of points is sufficient then it has been shown using the criteria that the type of the initial distribution has no importance.

2.4 Conclusion

So far, the reader is able to understand the basic principles of MultiPath TCP and how one may analyze its behaviors, through experiments and measurements. The reader has basic understanding of mathematical models developed for TCP connections and could be interested in how it could be extended to MultiPath TCP connections. Finally, the Space-filling technique was explained to give an idea of how the experimental design could be implemented.

Bulk-Transfers Analysis 3

When considering bulk-transfers of data, one can expect the subflows to be saturated. In the particular case of long run connections, the selection of subflow is controlled by the available space in the congestion window rather than the scheduler. This is called the *ack-clock* phenomena [37]. As a first approach and if we assume loss-free paths, these kind of connections can be useful to be analyzed as flows may reach an equilibrium. It should thus present less complex behaviors than for other kind of transfers.

In this chapter, we start by giving the design of our experiments. We briefly introduce the tools that we use. We also define the topology and the parameters of our network. Then we provide an analysis of the results of our measurements. We try to identify behaviors and cases in which MultiPath TCP connections perform better or not compared to simple TCP connections.

3.1 Design of the Experiment

One of our goals when designing an experiment is that we should collect both *soft* and *hard data*. Soft data are data acquired in a qualitative way. These are necessary to gain an intuitive understanding of MultiPath TCP behaviors in different situations. Hard data are data acquired in a quantitative way. These are necessary for mathematical analyzes and for reproducibility of our experiments [9].

As described in Section 2.1.3, there are different ways to experiment with MultiPath TCP. We decide to use an experimental design approach, i.e. the execution of controlled experiments to collect information about a system. In order to achieve controlled experiments, we wish to have a fixed topology and a way to set the parameters of our network. The *Mininet* [4] tool seems to respond to our needs as it proposes an emulation of network in which we can choose the topology and different parameters. The main benefits of such tool are that our results are reproducible and that we do not need a large number of physical machines for experimenting [39]. We also use the tool called *Minitopo* [5] that helps to the creation of the topology and brings useful configurations files for automated tests. The topology of the network we used for the experiments is shown in Figure 3.1. It is composed of a client that can establish two subflows with a server. This is similar to common scenario where a client has access to two networks, e.g. a Wi-Fi and a cellular network.

In the design of an experiment, one has to consider the different factors that will influence the

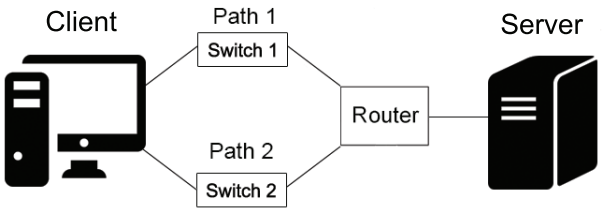


Figure 3.1: Topology of the network

performance of the transport protocol. Some are related to network (capacity of each link, delays, memory constraints, loss probability), other are related to the used protocols (the congestion controls scheme, the path-manager algorithm, the scheduler). We now present the considered factors.

3.1.1 Factors

The experimental design approach will define the range of the four factors that can be set for each path in the network simulated by *Minitopo* : the capacity, the delay, the maximum queueing delay and the loss probability. In order to generate the parameters, we used the *WSP algorithm* [12] that aims at filling the factors space with experiments to cover the widest range of possible scenarios. This allows us to have meaningful results as we test MultiPath TCP on a range that represents actual networks. Later, it will be possible to interpret the results of the experiments as real-world cases as the ranges have a real-world meaning. Here are in more details the different factors we consider and their proposed ranges as suggested in [39].

Bandwidth Capacity The bandwidth capacity is the maximum rate at which a user could send data on the path. As the user may use a large range of different networks (Wi-Fi, edge, 3G or LTE) the capacity of a path is set from 1 Mbps to 100 Mbps [39].

Propagation Delay The propagation delay is the latency of a path, i.e. the time needed by a packet to go from the sender to the receiver. As shown in [50], the propagation delay may vary from a negligible time in term of micro seconds to 200 ms, thus we set the range at 1 ms to 200 ms, as 0 ms cannot be set in Mininet.

Queuing Delay The size of the buffer at the bottleneck is related to the queueing delay. On the Internet nowadays, it often happens that buffers are over-sized causing a phenomena

called *bufferbloat* [25]. This causes an augmentation of the delay that can be huge. On the other hand, a perfect Active Queue Management would not add any delay. We thus consider a range of 0 to 2000 ms [39].

Loss Probability The probability for a packet to be dropped. It typically occurs when using Wi-Fi where packets are lost between the user and the access point. For those experiments, we decided to avoid this noisy behavior on the network and thus we set it to 0. However, losses can still occur due to congestion.

We define two types of networks based on their bandwidth-delay product (BDP) : low and high-BDP environments. Low-BDP environments correspond to relatively small values for the propagation and queueing delays while high-BDP can have very large values. Table 3.1 provides a summary of our domains of test, which are also based on [39].

	Low-BDP		High-BDP	
Factor	Min.	Max.	Min.	Max.
Capacity [Mbps]	1	100	1	100
Propagation Delay [ms]	1	25	1	200
Queuing Delay [ms]	0	100	0	2000
Loss Probability	0	0	0	0

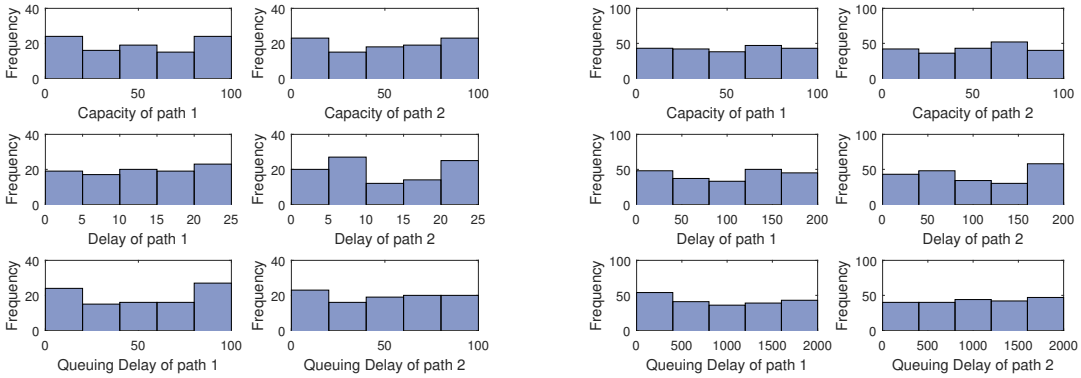
Table 3.1: Domains of the factors

There are thus 3 factors per path. Using the WSP algorithm on the 6-dimensional factors space, we obtain 98 experiments for the low-BDP environments and 213 for the high-BDP one. We justify the smaller number of experiments for the low-BDP environment by the fact that its factors space is smaller than the high-BDP one. In Figure 3.2, each factor is represented with a histogram. One can observe that their distribution are nearly uniform, which is a necessary criteria for space filling.

3.1.2 Setting the Queue Size

In Mininet, it is the maximum queue size that can be set rather than the maximum queueing delay. As we known that these factors are proportional, we have to find the relation between them. We first decided to use a formula that we found in Minitopo that sets the queue size proportionally to the bandwidth-delay product (BDP) :

$$queue\_Size = \frac{\text{BDP} \times queueing\_Delay}{1000 \times \text{MTU}} \tag{3.1}$$



(a) Low-BDP environments

(b) High-BDP environments

Figure 3.2: Domains of the factors

where BDP is expressed in bytes, $queueing_Delay$ in ms and $queue\_Size$ in packets. MTU is the *Maximum transmission unit* and is also expressed in bytes. We did ran the experiments but we observed that the queue size was a limiting factor in this case. We first identify an error of units within this formula as $queueing\_Delay$ has no unit. Furthermore, when designing such experiments, one would like to have $queue\_Size$ close to BDP as it is optimal. When analyzing the mean value for $queue\_Size$ using this formula, we observe very small value that are far from BDP in the low-BDP environment. The value obtained with equation 3.1 is, in fact, a multiplicative factor of the BDP smaller than 0.002, with our environments settings.

We thus decided to derive our own expression in the following manner. Suppose the client fulfills the queue in one round. In the best case, the queue could be emptied at a rate corresponding to the capacity of the link. And as we know the maximum queueing delay, we set a maximum queue size that can be emptied at this capacity rate during this maximum queueing delay :

$$queue\_Size = 10^3 \times queueing\_Delay \times \frac{capacity}{8 \times \text{MTU}} \quad [packets] \tag{3.2}$$

where $capacity$ is expressed in Mbps. This formula tends to give larger values than the optimal one, which could be more representative of actual networks [25]. Furthermore, it provides much better results in the low-BDP environments than formula 3.1. The final formula that we consider is the maximum between 10 packets and equation 3.2, such that we avoid queue sizes that are smaller than the initial send window.

We also considered to apply the space-filling on the queue size rather than queueing delay. However, it failed for two reasons. First, our range was way too small as we had chosen

a maximum size of 100 packets for both low and high-BDP environments. The queue size was the main limiting factor. Secondly, it should not be chosen randomly as in the case of space-filling but rather be somehow related to the path characteristics. The goal here is to have some coherence on the path characteristic so that we represent real-life networks.

Now that we have introduced the considered factors, we explain how we will analyze our results. We also provide a new metric to evaluate the connections.

3.2 Definition of a New Metric and Criteria

In the following sections, we will first analyze the aggregation benefit of our experiments. This measure has been defined in Section 2.1.3 and allows us to see how the multipath connection fills the available bandwidths. It could also be useful to compare these connections with TCP connections in the same case with a new metric. Also, we need to define precisely how we will analyze the results based on certain criteria.

Experimental Aggregation Benefit We decide to define the *experimental aggregation benefit* in order to measure how MultiPath TCP aggregates TCP flows running independently on the same paths :

$$E_{AB} = \begin{cases} \frac{B_p}{B_{tp_1}+B_{tp_2}} & \text{if } B_p \geq B_{tp_1} + B_{tp_2} \\ \frac{B_p - \max(B_{tp_1}, B_{tp_2})}{\min(B_{tp_1}, B_{tp_2})} & \text{if } B_{tp_1} + B_{tp_2} > B_p > \max(B_{tp_1}, B_{tp_2}) \\ \frac{B_p - \max(B_{tp_1}, B_{tp_2})}{\max(B_{tp_1}, B_{tp_2})} & \text{if } B_p < \max(B_{tp_1}, B_{tp_2}) \end{cases}$$

where B_p is the MultiPath TCP goodput and B_{tp_i} is the goodput of TCP on path i. We can interpret it as follows :

- $E_{AB} \approx 1$: MultiPath TCP performs as well as TCP on both paths. As it is experimental, it is possible for the aggregation to be a bit larger than 1 ;
- $E_{AB} = 0$: It performs as good as TCP on the path with the highest throughput;
- $E_{AB} = -1$: It achieves a goodput of zero.

In order to distinguish both metrics of aggregation, we will call the aggregation benefit: the theoretical aggregation benefit. We will always precise if we evaluate the *theoretical aggregation benefit*, A_B or the *experimental aggregation benefit*, E_{AB}.

Batch of Results When analyzing the results of experiments done with factors given by a space filling algorithm, one would like to sort the results on basis of a chosen criterion. In our graphs, we present the experiment results, r_{exp}, in n batch, B_i, of range $]x_i, x_{i+1}]$:

$$\text{B}_i = \left\{ r_{\text{exp}} | x_i < \text{criterion(exp)} \leq x_{i+1} \right\} \quad i \in 0, \ldots, n-1$$

For example, let the criterion be the delay of the first path in low-BDP environment and consider 5 batches. Then, as the maximum delay is 25 ms, the 5 ranges are :
$]0,5],]5,10],]10,15],]15,20],]20,25]$. All the experiments that are such that the first path has a delay lower than 5 ms will go in the first batch, then all the experiments where the delay of the first path is between 5 and 10 ms will go in the second batch, et cetera.

Factor Difference We decided to use the difference between the path characteristics as a criterion. We are interested in the factor difference, δ_{factor}, but also the factor absolute difference, $|\delta_{\text{factor}}|$:

$$\delta_{\text{factor}} = \text{factor}_{path_1} - \text{factor}_{path_2}$$

and $|\cdot|$ is the absolute value. The criterion δ_{factor} takes into account on which path the connection is started while the absolute value of it does not. We precise each time which criterion is used.

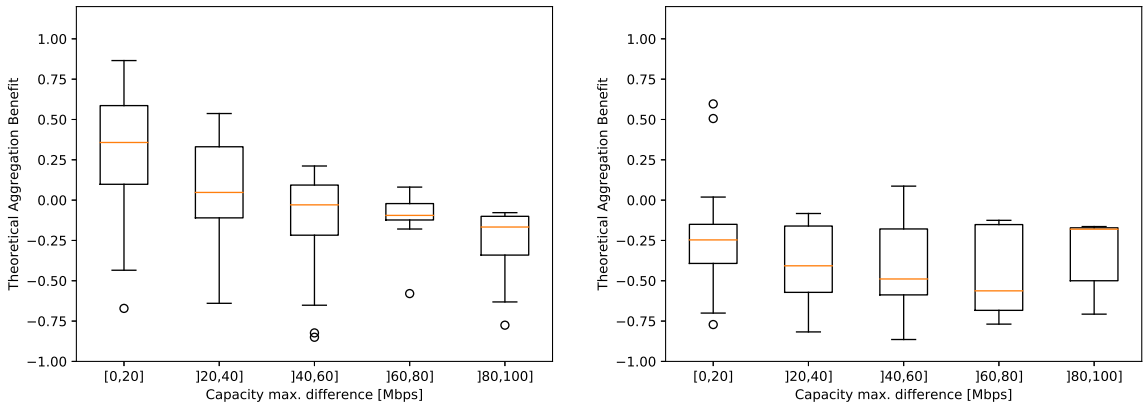
For example, let the criterion be the delay difference in low-BDP environment. One has : $\delta_{\text{delay}} = \text{delay}_{path_1} - \text{delay}_{path_2}$. Then we sort the experiments in the batches proposed in the previous example.

First Path Quality Another criterion we use is the first path quality : we define the best (worst) path as the path on which the single path TCP connection achieves its best (worst) throughput.

3.3 Measurements Results

In this section we present the main results about our analysis of generated traces. We try to identify cases where MultiPath TCP is performing well and badly using measures like the aggregation benefit, the number of reinjections and the number of retransmissions, and the traffic distribution.

In order to generate traffic, we use two different applications : `iPerf` and `cURL`. The version of the MultiPath TCP Linux Kernel implementation used for all the experiments is the re-



(a) Low-BDP environments

(b) High-BDP environments

Figure 3.3: Theoretical Aggregation Benefit as a function of the capacity difference

lease *v*0.91 on Ubuntu 14.04, Linux Kernel version 4.1.38. As we model it in Chapter 6, the congestion control scheme is LIA. The scheduler is the default one.

3.3.1 iPerf Analysis

`iPerf` [3] is a tool that measures the maximum achievable bandwidth on networks. In order to achieve this, it generates data to be transmitted such that subflows are saturated. Thus it can be categorized in the aggressive bulk-transfers, in which we are interested now. The `iPerf` sessions runs for 30 seconds to allow the subflows to reach an equilibrium. We use the version 3.1 of `iPerf`.

Theoretical and Experimental Aggregation Benefits

A first assumption that we would like to confirm is that both theoretical and experimental aggregation benefit should be smaller for connections where paths have higher differences in their factors. Indeed, the heterogeneity of the path characteristics may induce phenomena such as *Head-of-Line Blocking* that would reduce the overall performance of the MultiPath TCP connections [37]. We now analyze the results given in Figures 3.3 - 3.4. In Figure 3.3, we first show the theoretical aggregation benefit related to the capacity absolute difference for both low and high-BDP environments.

For the low-BDP environment shown in Figure 3.3a, one can observe that while the capacity

difference increases, MultiPath TCP clearly decreases its aggregation of the paths capacity and thus it leads to smaller goodput. For capacity differences larger than 80 Mbps, the aggregation becomes smaller than 0 for all the experiments. This can be explained by the fact that one of the two path has a really small capacity compared to the other. Thus this path rapidly undergoes congestion and the congestion control scheme choose not to use it. In a high-BDP environment, the theoretical aggregation benefit is significantly smaller than for low-BDP environment and the median of the aggregation benefits is below 0. Figure 3.3b shows it as a function of the capacity difference. As we explain in the traffic distribution analysis below, in high-BDP the best path tends to be the most used while the worst path is not used. This explains why the aggregation is below zero.

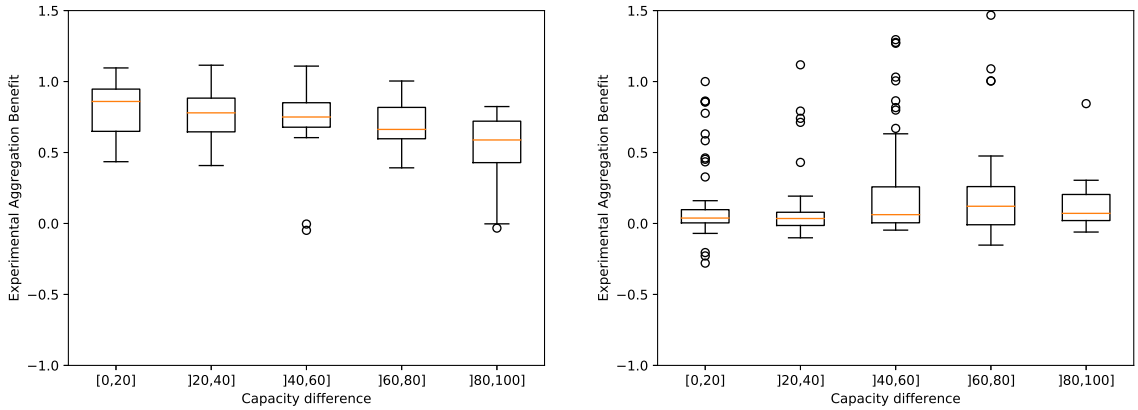
When we analyze these graphs, we have to be careful because the TCP connections might also be impacted by the path characteristics. We thus now present the results of the experimental aggregation benefit for both environment types.

In the figure 3.4a, we can observe that MultiPath TCP achieves its goal to aggregate two single-path connections goodput which use the *New Reno* congestion control scheme. As the theoretical aggregation benefit, the experimental aggregation decreases as the path capacity difference increases, i.e. MultiPath TCP tends to produce worse performance than two TCP connections when the difference increases. However, the impact is smaller than for theoretical aggregation benefit. We attribute this to the fact that TCP also suffers from congestion on small capacity paths.

Globally speaking, the capacity difference is thus an interesting factor and plays an important role in the MultiPath TCP performance, at least in low-BDP environments. We may notice that there are two outliers in the range $]40, 60]$. Those two outliers correspond to topologies where the queue size is close to 10 packets, which is too small and thus limits the performance.

For the high-BDP environments, we have observed that the aggregation of the bandwidths was below zero. As we study the experimental aggregation benefit shown in Figure 3.4b, MultiPath TCP has a slightly better median goodput than TCP on the best path. This confirms the fact that MultiPath TCP tends to only use the best path in high-BDP environments. Thus the factor differences are not relevant as only one path is used.

We now analyze the impact of the difference in the delays. Our assumption is that it should be an important criterion as we think that it is one of the main contributor to the fact that packets can arrive out-of-order. However, the difference of delays does not seem to impact the aggregation benefit as it is of the same magnitude while the delay difference increases, for both low-BDP and high-BDP environments. For the low-BDP experiments, the explanation can be that the delay range is narrow and the difference isn't big enough. For the high-BDP, we noticed earlier that it only uses one path during the bulk transfer, thus the performance



(a) Low-BDP environments

(b) High-BDP environments

Figure 3.4: Experimental Aggregation Benefit as a function of the capacity difference

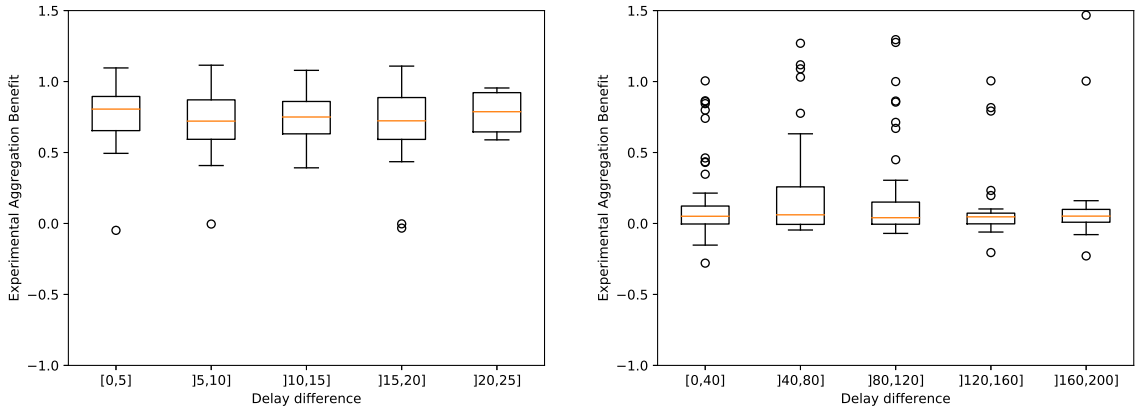
is no more related to the magnitude of the differences between path characteristics.

The difference of delays seems thus to have few impacts on the performance. We attribute this to the fact that the receive buffers are well-sized such that out-of-order packets are not discarded by the receiver.

Traffic Distribution

As we notice in the high-BDP environments that the traffic distribution over the path is unbalanced, we analyze more deeply the distribution of the traffic over each path. In the following figures, we only show the transmission ratio on first path due to the complementarity of the measure. There are two CDF curves per graph, the first one in red is the one when the factor difference is negative. The green dashed one represents the CDF of the ratio when the factor difference is negative. Thus, for the capacity factor, we expect that the path with the higher capacity will be more loaded, and for the delay factor, we expect that the path with the smaller delay will be more loaded.

In *low-BDP* environments, see Figure 3.6a, we can see an interesting behavior : less than 10% of the connections outputs more packets through the weaker path in terms of capacity. This confirms the fact that capacity difference is a determining factor on the behavior of the connection. In Figure 3.7a we are interested in the delay difference. We can see that the traffic distribution is not influenced by the this factor. We attribute this to the fact that the range of this factor is too narrow to affect MultiPath TCP performances.



(a) Low-BDP environments

(b) High-BDP environments

Figure 3.5: Experimental Aggregation Benefit as a function of the delay difference

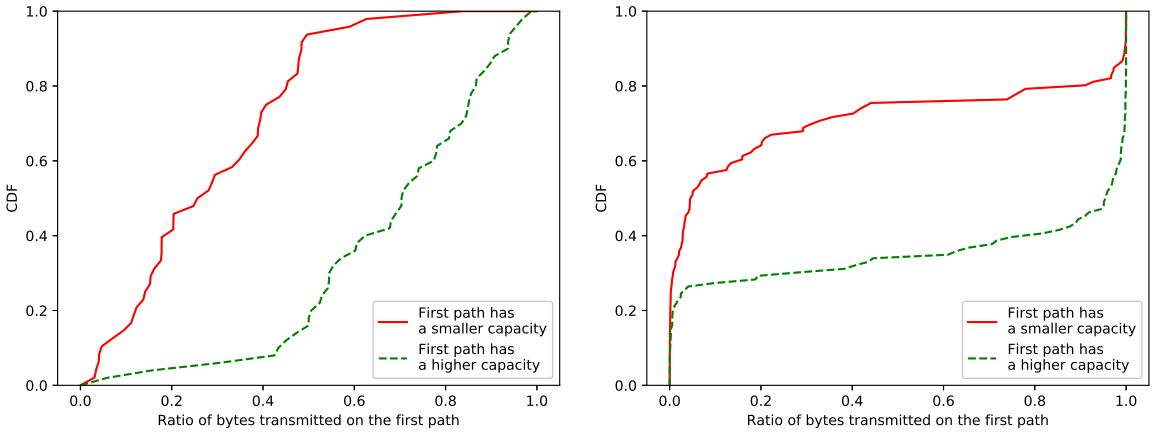
In *high-BDP*, see Figure 3.6a, for experiments where the capacity difference is positive, 30% of the connections sends packets only through the first path. In the same case, 35% of the experiments are such that the second path performs better than the first. In the cases where the delay difference is negative, in Figure 3.7b, 60% of the connections sends packets only through the first path. We believe high-BDP environments induce lots of phenomena such as timeouts due to large delays and queueing delays, and bufferbloat due to large queue sizes [25]. From these observations, we think that small difference in path quality is sufficient for the congestion control scheme to discriminate the worst path and only load the best one.

One last thing we notice is the central symmetry between the two curves of a graph. This symmetry could be explain by the fact that the traffic distribution is not influenced by the first path used to start the connection. We now analyze in more details the impact of the first path.

Impact of First Path

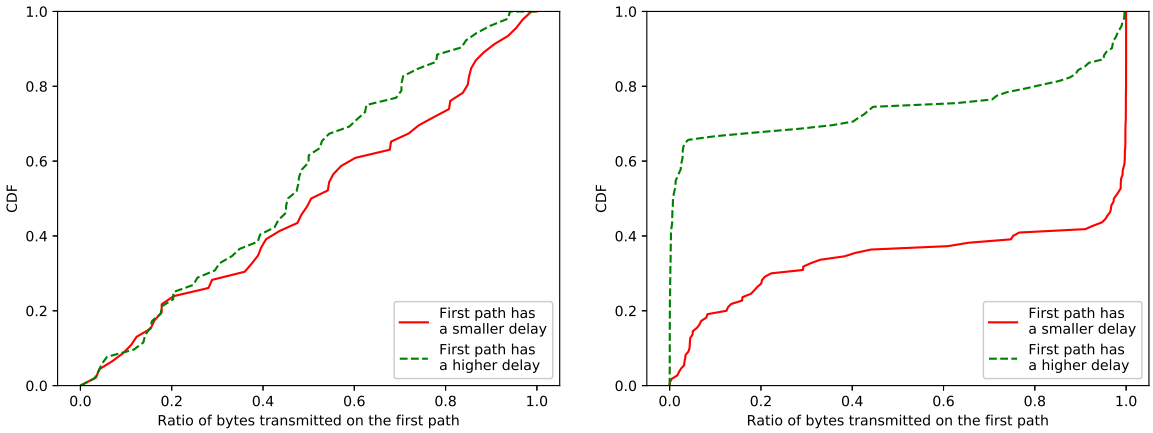
In [10], *Arzani et al.* analyze mathematically the impact of the choice of the first path. Their conclusion is that the MultiPath TCP goodput may be dependent on it due the coupled congestion control. As our previous results does not confirm this, we decide to analyze this impact in more details on our experiments. To plot the Figure 3.8, we use the First Path Quality criterion defined in Section 3.2 to divide the experiment in two batches.

We observe that the experimental aggregation benefit has the same magnitude when the first



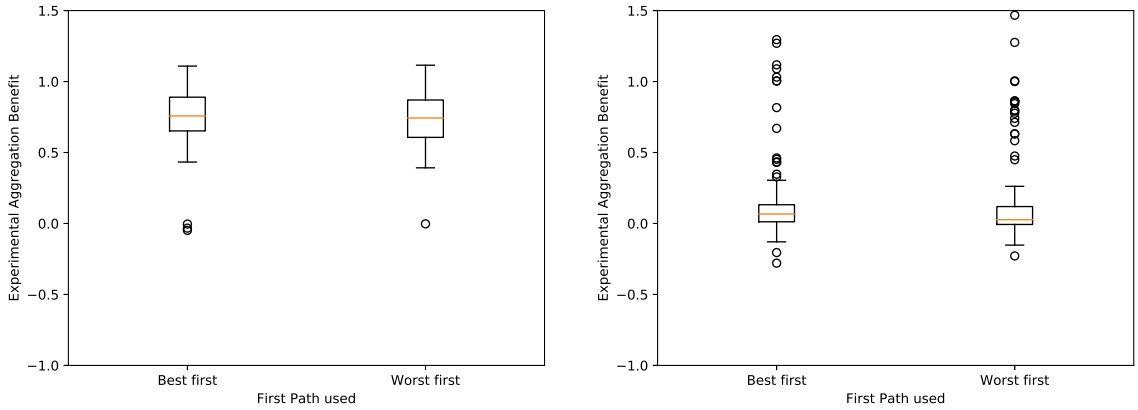
(a) Low-BDP environments (b) High-BDP environments

Figure 3.6: CDF of the traffic distribution on first path grouped by capacity difference



(a) Low-BDP environments (b) High-BDP environments

Figure 3.7: CDF of the traffic distribution on first path grouped by delay difference



(a) Low-BDP environments

(b) High-BDP environments

Figure 3.8: Experimental Aggregation Benefit with relation to the First Path Quality

path selected is the best or the worst one in low-BDP environments. The impact seems to be small for bulk-transfers and we expect it to be stronger for smaller exchanges. Furthermore, as it is due to the coupled congestion control, we expect it to be stronger in cases of losses or in cases of congestion. In the high-BDP environment, there is few impact on the aggregation benefit.

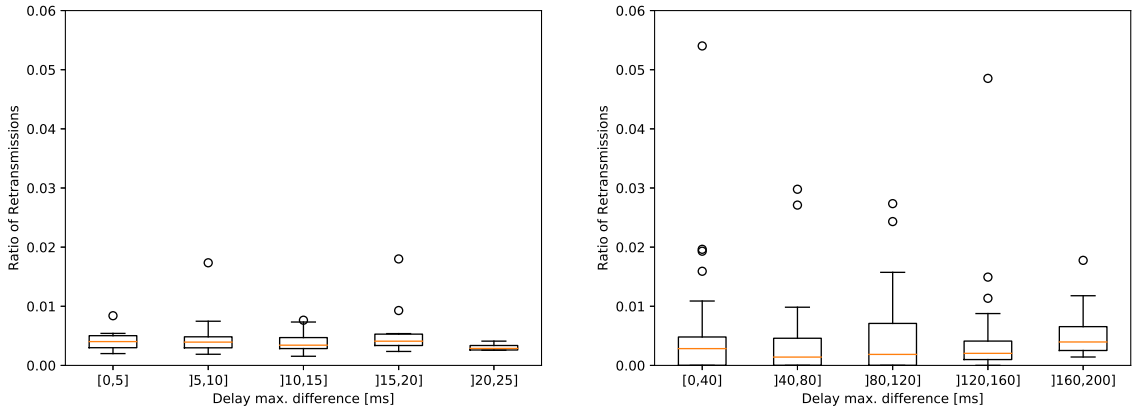
Retransmissions and Reinjections

In this section, we analyze the ratio between the number of packets retransmitted or reinjected and the number of packets successfully transmitted on TCP or MultiPath TCP connections.

Concerning the ratio of retransmissions, we would like to confirm our idea that it should be larger in high-BDP than in low-BDP environments. Figure 3.9 shows our results as a function of the delay difference.

The ratio of retransmissions is indeed slightly bigger for high-BDP environments. We attribute this to the large delays and queueing delays that might create timeouts, out-of-orders packets, head-of-line blocking and bufferbloat. These phenomena induce loss of packets and thus retransmissions. The delay difference has no impact on low-BDP environments. For high-BDP, the ratio tends to grow a bit with the difference. We also have analyzed this ratio with the capacity difference but there was no influence.

We now analyze the ratio of reinjected packets. Given the *Opportunistic Retransmission and Penalization* (ORP) proposed by Raiciu et al. in [43], we expect a higher reinjection rate as



(a) Low-BDP environments

(b) High-BDP environments

Figure 3.9: Ratio of retransmissions as a function of delay difference

the delay difference increases. Indeed, its principle is to reinject segments causing head-of-line blocking on a subflow that has space available in its congestion window. It also penalizes the subflow with the highest delay in order to reduce the effect of bufferbloat [37].

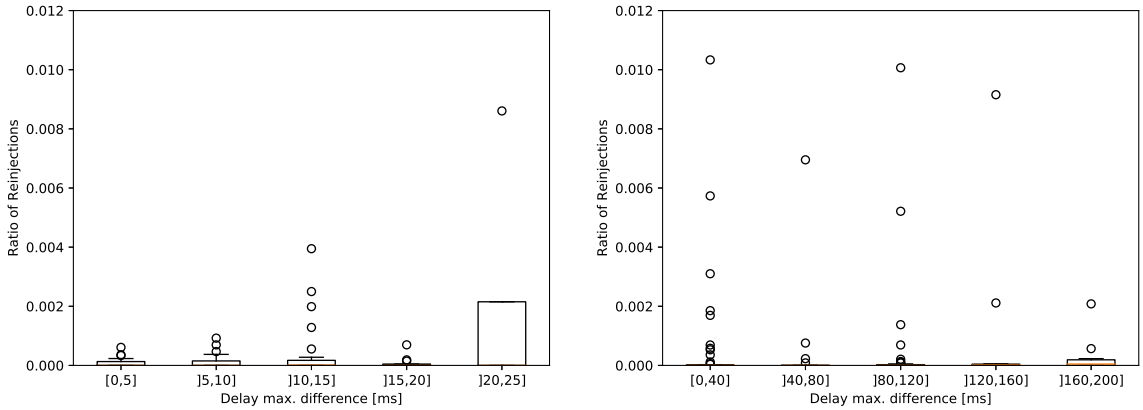
In low-BDP environments, we confirm the impact of the delay difference as shown in Figure 3.10a. We can see that for larger values of difference, we reinject more packets. This might explain why we did not observed any influence of the delay difference in low-BDP environments in the previous analyzes. Indeed, the reinjections allow to quickly overcome head-of-line blocking situations and to compensate the delay difference. This globally improves the goodput and reduces the effects of the delay difference.

In high-BDP environments, we observe no influence. As we have seen, MultiPath TCP tends to only use one path in these kind of environments. Thus there is less out-of-order packets and thus less head-of-line blocking, as packets only go through one path. So the ORP does not come into play and there is no effect when the delay difference increases.

3.3.2 cURL Analysis

As a second application, we choose the `cURL` [2] binary. This application allows to transfer data using, for example, the HTTP protocol over a MultiPath TCP connection. First, we decided to transfer files of larger size : 16 and 32 MB to keep analyzing the behavior of MultiPath TCP on long connections - with bulk transfers.

We have extended the `minitopo` tool to enable cURL experiments. To test this application,



(a) Low-BDP environments

(b) High-BDP environments

Figure 3.10: Ratio of reinjections as a function of delay difference

we use the same topology and network parameters as those we used for `iPerf`. The files transferred have sizes of 16 and 32 MB. As constant, we set a timeout of 30 seconds on the transfer and focus on the throughput rather than the time needed to transmit the full file.

As many results are similar to `iPerf` results, we only discuss the main differences between the two application and when there is a difference in the behaviour with the 16 or 32 MB files transmissions.

Experimental Aggregation Benefit

We show in Figure 3.11 the experimental aggregation benefit of our experiments with `cURL` on *low-BDP* environments, with the two different file sizes. One can observe that on both configurations, the aggregation is not as good as with `iPerf`, shown in Figure 3.3a. We also observe that the transfer of the largest file gives better results. We assign this to the effects of the slow-start. The start of the connection is violent and might create congestion. It takes time to be recovered and thus as the file size grows, this effect fades out. Furthermore, MultiPath TCP is more sensible than TCP as its growth of the congestion window is bounded by the growth of a TCP congestion window [49], i.e. it takes more time for MultiPath TCP to recover than TCP. This explains why the experimental aggregation is lower for smaller files.

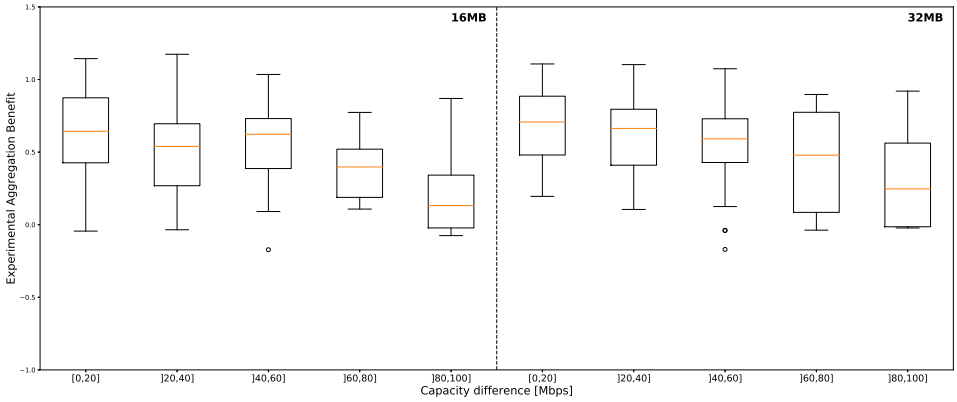


Figure 3.11: Experimental Aggregation Benefit as a function of the capacity difference

We now see the importance of studying smaller files. Chapter 4 is dedicated to this.

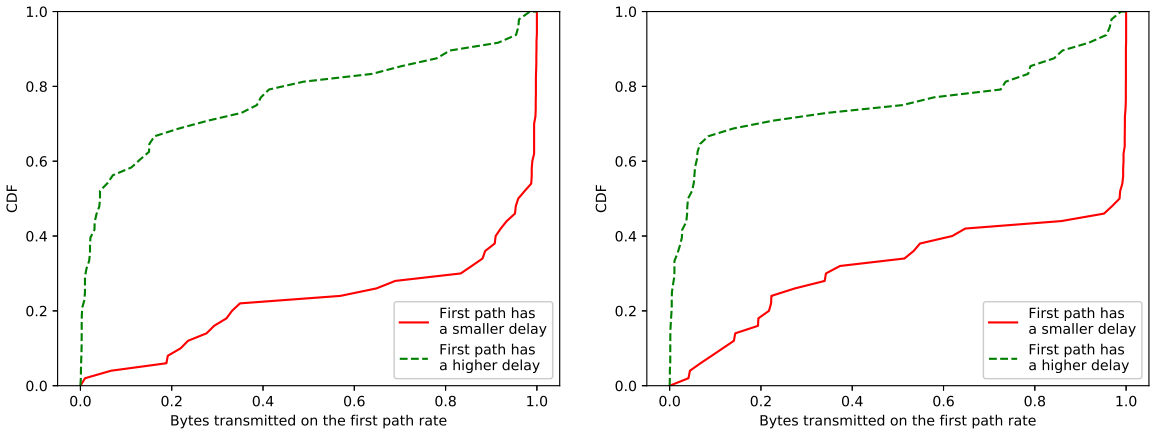
Traffic Distribution

Here, we compare the result obtained with `cURL`, in the high-BDP environments. We look at the CDF of the traffic distribution grouped by the capacity difference - red for the negative difference, dashed green for positive ones - in Figure 3.12.

We observe that even for 16 MB files, the distribution of the traffic tends to be discriminated in function of the delay. However this effect grows with the size of the exchanges file. The results are similar than the results for `iPerf`.

3.3.3 Sensitivity Analysis

When we consider sensitivity, we know that the higher is the number of experiments in our parameter space, the higher is the precision of our results. However due to large time constraints it is not possible to execute an overly large number of simulations. It is thus important to analyze the sensitivity of our results. Similar experiments in the same parameters space than our were conducted in [39]. To analyze the sensitivity, they generate 5 different space-filling and analyze the differences in 25th and 75th percentiles and median among each of these designs. Their conclusion was that 200 experiments were sufficient to have a good overview of the aggregation benefit of MultiPath TCP in the 6-dimensional parameters space.



(a) 16MB file

(b) 32MB file

Figure 3.12: CDF of the traffic distribution on first path grouped by delay difference in a High-BDP environment

Here, we generate 3 different space-filling sets of factor and we analyze the standard deviation of the median, the 25th and 75th percentiles of the theoretical aggregation benefit relative to its range, i.e. $[-1, 1]$. We launched the experiments with `iPerf` for 30s.

Environment	Median	25th percentile	75th percentile
Low-BDP	1.740%	0.194%	1.817%
High-BDP	1.297%	0.869%	0.114%

Table 3.2: Sensitivity analysis : Relative Standard deviation of the median, 25th and 75th percentiles

We can thus conclude that our choice of number of parameters set, i.e. 98 for low-BDP and 213 for high-BDP, is sufficient to have a good overview of the aggregation benefit of MultiPath TCP in our parameters space for bulk-transfers.

We assume this sensitivity analysis is also valid for the Experimental Aggregation Benefit as we also use the same MultiPath TCP throughput values and extend the theoretical definition. Nevertheless, to confirm this hypothesis, we should have experimented the sensitivity of the single path TCP connections throughput results.

3.4 Conclusion

In this chapter, we proposed a first approach to MultiPath TCP connections. We started by introducing the tools that we used in order to collect data. We presented the topology and the parameters of our network.

We entered into the practice phase. Various experiments on MultiPath TCP connections through two different applications have been executed. In order to analyze the obtained results, we defined a new metric which is based on the performance of single-path TCP connections : the *experimental aggregation benefit*. We also defined the criteria used to present our results, as the factor difference and the first path quality.

Some behaviors of MultiPath TCP have been observed. In our simulations for bulk-transfers in low-BDP environments, MultiPath TCP performs better when the different paths used are balanced, especially in term of capacities, which seems to be the main factor. The aggregation decreases once it becomes unbalanced.

In high-BDP environments, MultiPath TCP tends to only use the best available path. We attributed this to the condition of such type of networks : bufferbloat, large queueing delay, et cetera. We provided extensive analyzes on the traffic distribution.

We also tried to verify an assumption from *Arzani et al.* in [10], about the impact of the first path opening the MultiPath TCP connection. We did not remark any influence and we expect the results to be more convincing for smaller files. Experiments with `cURL` indicated that the size of the exchanged file has an impact on the aggregation. From these observations, we decide to analyze in more details the exchanges of small files in the next chapter.

Small Files Exchanges Analysis

4

In the previous chapter, we studied the behavior of MultiPath TCP connections in the case of bulk-transfers. While these kind of transfers are more representative of long-run connections, one should also be interested in smaller transfers that are more present in the current Internet [18].

Within this part, we extend our analysis to the exchange of small files. Similarly to Chapter 3, we start by giving the details of our experiment setup. We then analyze our measurements according to different factors. We study the impact of the size of the exchanged files and of the choice of the scheduler.

4.1 Design of the Experiment

This section is similar to Section 3.1 and thus contains fewer details. For more information, please refer to the latter.

We keep considering a client with two interfaces that exchanges data with a server. We use a fixed topology with one bottleneck for each client interface. These bottlenecks are characterized by three factors that are : the bandwidth capacity, the propagation delay and the maximum queueing delay. We still study two kind of environments depending on the bandwidth-delay product (BDP) : low-BDP that have small propagation and queueing delay and high-BDP that have greater values. The range for the factors are the same than the one given in Section 3.1.

In order to generate these factors, we use a space-filling approach with the WSP algorithm [12]. The algorithm provided 98 experiments for the low-BDP environment and 213 for the high-BDP one. We justify the smaller number of experiments generated in the low-BDP by the fact that the factor-space to fill is smaller than the high-BDP factors-space. As we analyze small file exchanges, one also has to define their size. We thus decide to choose the following set of sizes : 8 kB, 32 kB, 128 kB, 512 kB and 2 MB. Finally, we consider two different schedulers : the *default* or *Lowest-RTT-First* (Low-RTT) one (with the *Opportunistic Retransmission and Penalization* option enabled) which sends data on subflows with the lowest RTT until their congestion-window is full and the *Round-Robin* (RR) one. See Section 2.1.1 for more details about schedulers.

All these experiments were simulated in *Mininet* [4] using the version 0.91 of MultiPath

TCP on Ubuntu 14.04, Linux Kernel version 4.1.38. In order to measure the Experimental Aggregation Benefit, the files are exchanged using `cURL` [2] with a timeout of 30 seconds in case the transfer is not finished. The congestion control scheme used was `LIA`.

4.2 Measurement Results

In this section, we present our measurement results about small files exchanges using Multi-Path TCP. We mainly use the metric that has been defined in Section 3.2 : the *Experimental Aggregation Benefit*, which aims at comparing MultiPath TCP to TCP performances. We identify cases where MultiPath TCP performs well or badly.

4.2.1 Small Files Exchanges with `cURL`

We start by giving an overall analysis of our results, i.e. we analyze globally the experimental aggregation benefit for each file size and each scheduler. We are not interested in the theoretical aggregation benefit because the files are too small for the connection to fill the available bandwidth.

Figure 4.1 represents the experimental aggregation for low-BDP environments. The first thing that one can observe is that most experiments have negative values. We attribute this to the fact that the files are too small, such that MultiPath TCP does not have the time to benefit from the second subflow. As we compare it to the performance of TCP on the best path, we can thus say that the aggregation is close to 0 in the best case, i.e. when the first path is the best path. One also remarks that the variance diminishes with the file size and tends towards 0, until the file size becomes large enough for MultiPath TCP to aggregate the links performance.

We believe there is a trade-off between the file size and the start of the connection. The start of the connection imposes an exponential growth on the congestion windows. Such violent start might create congestion if the file size is enough for the transfer to fill the available bandwidth. For very small files, the slow-start does not imply congestion but only one path is used. Then for larger files, the slow-start might create congestion but we use the two paths. Furthermore, MultiPath TCP takes more time to recover from the slow-start effects than TCP because its growth in the congestion windows is bounded by the growth of TCP flow on the same path, in congestion avoidance mode. This explains why the experimental aggregation is still low for 2MB files. Then for even larger files, such as studied in Section 3.3.2, the slow-start effects tend to fade out and MultiPath performs nearly as good as TCP

on the two paths. We conclude that, for small files, MultiPath TCP performs better as the file size grows.

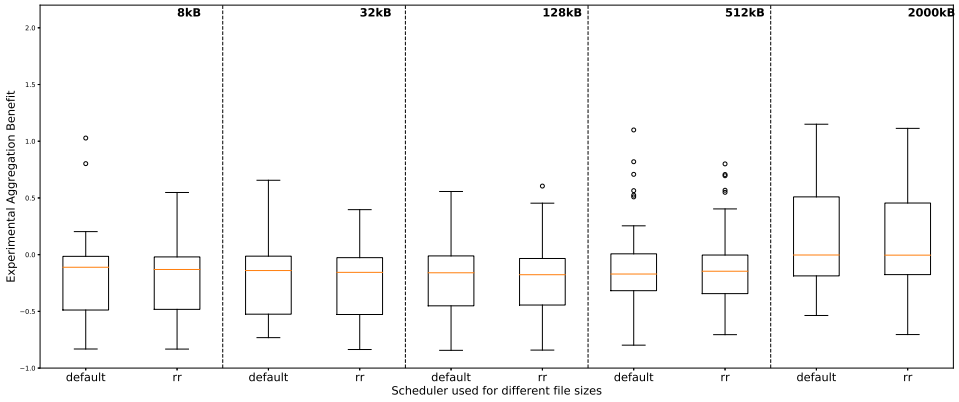


Figure 4.1: Experimental Aggregation Benefit of the considered sizes file in low-BDP environment

We now look at the impact of the scheduler. As explained in [37] by *Paasch et al.*, LowRTT and RR perform similarly in bulk-transfers due to the fact that TCP subflows are saturated and thus controlled by the *ack-clock*. On the other hand, the behavior of the schedulers has also been studied with application which limits the transmission rate and it has been showed that Round-Robin offers really bad performance for this case. Here, we look at the performance in terms of goodput. In Figure 4.1, we observe that Round-Robin performs similarly for very small files. Then for larger files, Low-RTT becomes a little bit better than Round-Robin. But globally the scheduler seems to have few impact on the performance of a MultiPath TCP connection. As previously said, the reason is the scheduler that becomes self-clocked. The scheduling of packets is managed by the available space in the congestion windows rather than the scheduler itself [37]. Globally speaking, the scheduler should rather be studied with rate limited application. This is left for future work.

We are now interested in high-BDP environments. As we have seen in Chapter 3 about bulk-transfers, MultiPath TCP performs less well in these kind of environments than in low-BDP environments as most of the experiments had Experimental Aggregation Benefit near 0. Let us first give an overall analysis that is summarized in Figure 4.2. The experiments for small files also offer bad performances as most of them have negative Experimental Aggregation Benefit. As in low-BDP environments, the variance decreases as the file size increases and really bad cases tend to occur less but cases where MultiPath TCP performs better than TCP on the best paths are rare even for larger files. We analyze further the experiments for which the experimental aggregation benefit takes value near -1.

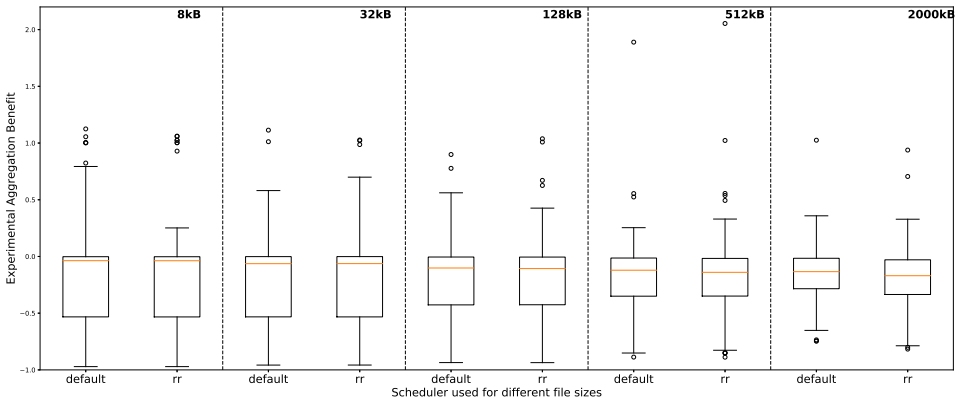


Figure 4.2: Experimental Aggregation Benefit of the considered sizes file in high-BDP environment

Influencing Factors

Now that we have a global overview of the behavior of MultiPath TCP when exchanging small files, one would like to enter into more details and find what are the main influencing factors. In order to achieve this, we continue to look at the Experimental Aggregation Benefit but we separate the results into different intervals that represent a certain difference in some factors between the two paths, e.g. the capacity or the delay difference. This allows us to see how the aggregation evolves as difference grows. The following figures are first divided into five parts, one for each size file. Then for each size file, there are three batches of experiments grouped using the factor difference criterion δ_{factor} introduced in Section 3.2. We use this criterion rather than the absolute factor difference, $|\delta_{factor}|$, to take in consideration the first path used. Indeed for small files, we expect the performance to be dependent of the first path used. Also, we decide to focus on the Low-RTT scheduler as the differences with the Round-Robin one are small.

Capacity Difference Let us first take a look at the capacity difference. We have seen in Chapter 3, that it was an important factor for bulk-transfers, at least in low-BDP environments. As the difference grew, the performances of MultiPath TCP were becoming worse. In Figure 4.3, we present our results for the exchange of small files in low-BDP environment. We first observe that the capacity difference has few impact for very small files in low-BDP environments. We attribute this to the fact that 8kB files are too small, such that the bandwidth capacities are not a limiting factor and thus do no have any effect on the transfers. For files between 32kB and 512kB, the choice of the first path takes an importance and the

E_{AB} median value increase with the capacity difference. We will discuss the effect of the first path choice later in this section. Finally, for the 2MB files, which is closer to the previously analyzed bulk-transfers, the absolute capacity difference takes more importance and has a stronger impact on the performance of a MultiPath TCP connection.

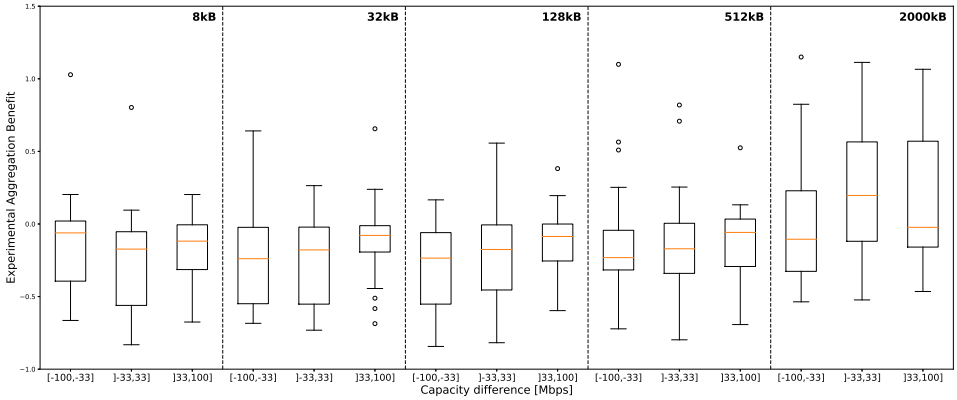


Figure 4.3: Experimental Aggregation Benefit for different intervals of capacity difference in low-BDP environment

We now analyze the impact of capacity difference in high-BDP environments. The results are shown in Figure 4.4. What we observe is that the capacity difference has few influence in the high-BDP environments. It could indicate that the bandwidth capacities are not reached by the connections in high-BDP environment, no matter the file size.

We can not observe a clear impact of the capacity on the MultiPath TCP performances for small files. We have to take a closer look to other factor to understand which one influence badly the performance.

Delay Difference We are now interested in the impact of the delay difference. One can see in Figures 4.5 and 4.6 that when the first path delay is better, i.e. we have a negative value for δ_{delay}, the Experimental Aggregation Benefit is way better for files from 8kB to 128kB for low-BDP environments and from 8kB to 512kB for high-BDP environments. However, the E_{AB} median value is not greater than 0 for those file size. Thus, MultiPath TCP only achieves to equalize the single path TCP, when it begins on the path with the smaller delay. As file size grows, this effect fades out and $|\delta_{factor}|$ becomes more relevant.

As a reminder, in bulk-transfers we observed the delay difference was not an influencing factor of the MultiPath TCP performance. Here the situation is different because the connection is

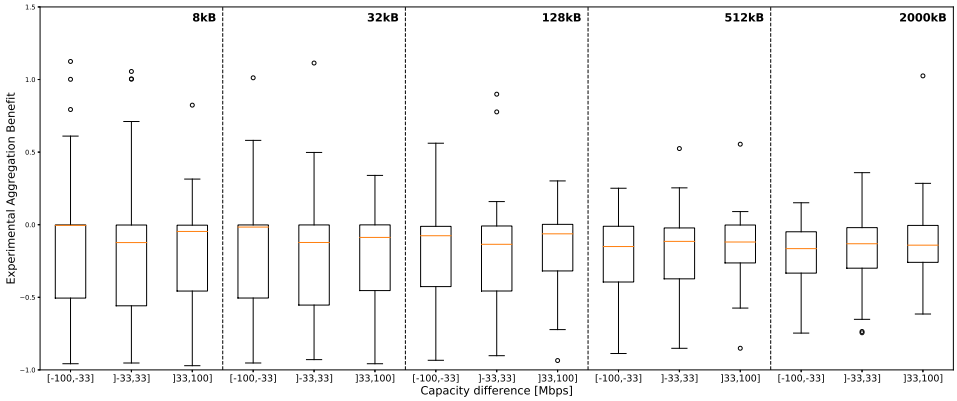


Figure 4.4: Experimental Aggregation Benefit for different intervals of capacity difference in high-BDP environment

not at steady-state. Paths with higher RTTs suffer more from the start of the connection. The higher RTT implies a larger time for the steady-state to be reached. As the file size grows, the steady-state might be reached and thus the effect of the difference in the delays fades out.

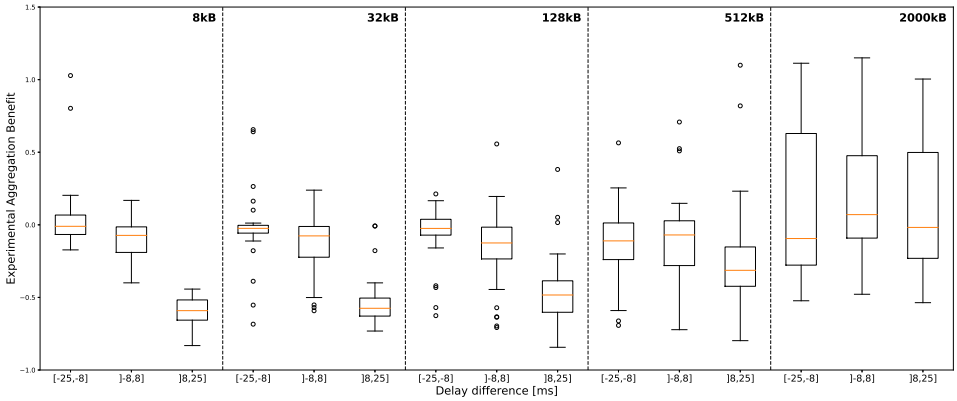


Figure 4.5: Experimental Aggregation Benefit for different intervals of delay difference in low-BDP environment

One may also notice that there are a lot of outliers for the Experimental Aggregation Benefit in the cases of 8kB, 32kB and 128kB files exchanges. This may be explained by the fact that those exchanges are more sensitive to environment noise than the bigger transmissions and

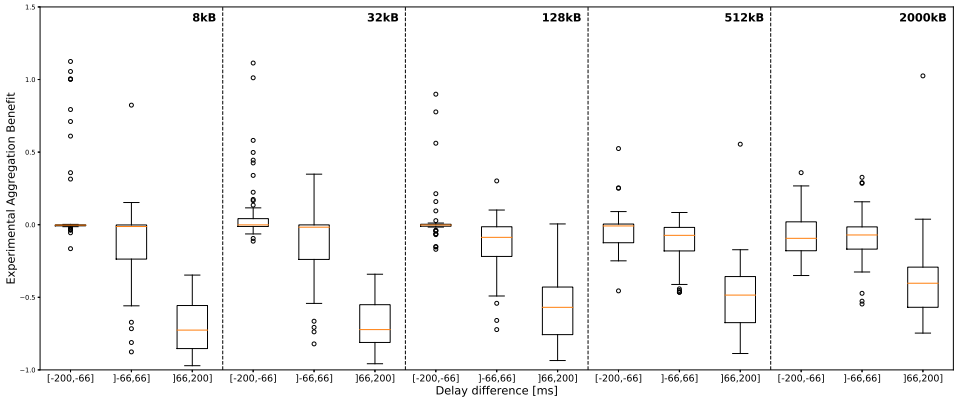


Figure 4.6: Experimental Aggregation Benefit for different intervals of delay difference in high-BDP environment

steady state connections analyzed earlier. To reduce this noisy phenomenon, we should repeat the experiments on each parameter set several times and use the median throughput of those experiments.

We noticed the importance of the first path chosen, which is even more apparent in high-BDP environments, as shown in Figure 4.6. We now analyze it in more details.

Impact of the First Path

Figures 4.7 and 4.8 give a better understanding of the previous analyzes. We define the best path as the path that has the highest goodput during the MultiPath TCP connection.

We divide our analysis by file size in specified environments :

- 8kB to 512kB in both environments and 2MB in high-BDP environments : The choice of the first path has an important impact on the performance of the connection. If we look at our previous results, in the two precedent sections about the capacity and the delay difference, we can see that this performance is worst when the delay difference grows and, in a minor magnitude, when the capacity difference becomes negative and decreases. Thus for those files, MultiPath TCP performs better if it chooses the path with the smaller delay and the bigger capacity to start the connection. As an important part of the files exchanged on the Internet through TCP and MultiPath TCP connections are files with this range of sizes [18], we get the importance of choosing the first path. By the way, the influence of the first path chosen decreases while the size of the file

transmitted increases.

- 2MB in low-BDP environments : The impact becomes less and less important and, as analyzed for bulk-transfers, the load balancing of the traffic on each path becomes better.

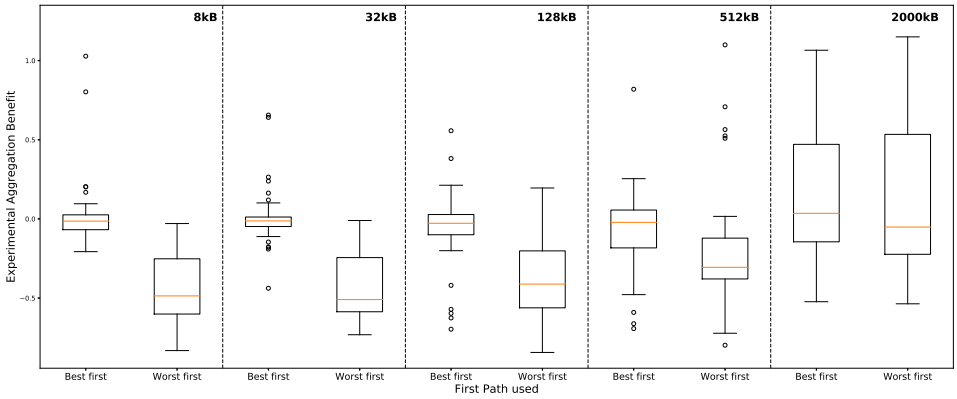


Figure 4.7: Impact of the choice of the first path in low-BDP environment

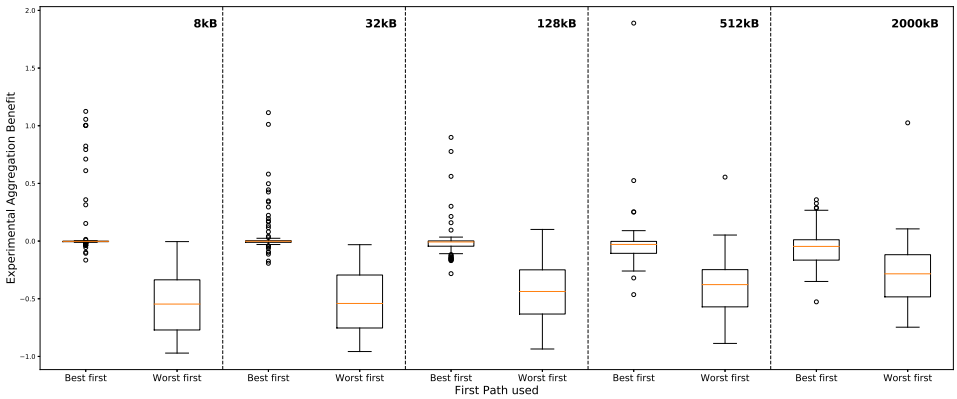


Figure 4.8: Impact of the choice of the first path in high-BDP environment

4.2.2 Request-response with `ApacheBench`

Beside the previous measurements, an interesting measure when exchanging small files is the average request-response time. This average request-response time may be computed using the `ApacheBench` [1] application which is a tool used to benchmark Apache servers. In this tests, we would like to compare the behaviors of the different schedulers with this type of application.

Our experiments were launched on the 98 low-BDP and 213 high-BDP environments, the goal was to benchmark the GET request-response time for files of size 8kB, 32kB, 128kB, 512kB and 2MB. The experiments last for `30s` each. As the results didn't help us to compare the different schedulers, we invite the reader to find the experiments results in the Appendix C.

4.2.3 Sensitivity Analysis

When designing experiments with a space-filling algorithm, one would be interested by the sensitivity of our results. In Section 3.3.3, we validated the non-sensitivity of the theoretical aggregation benefit for bulk-transfers. We present here the results for each file size in each environment type. The small values of standard deviation let us conclude, once again, that the number of experiments we launch for each environment is sufficient to have a good overview of the MultiPath TCP Aggregation Benefit. We thus repeated three times the experiments with `cURL`, default scheduler and three different set of factors generated by the WSP algorithm. Then we compare the median, 25th and 75th percentile of the theoretical aggregation benefit. Results are shown in Table 5.2. We conclude that our experiments results are not sensitive to the experimental design.

The sensitivity of the Experimental Aggregation Benefit is not explored here but we assume here that this measure is valid and not sensitive to the experimental design. We gave the explanation in the previous chapter, Section 3.3.3.

File size	Environment	Median	25th percentile	75th percentile
8kB	Low-BDP	0.037%	0.022%	0.117%
	High-BDP	0.011%	0.002%	0.032%
32kB	Low-BDP	0.092%	0.086%	0.446%
	High-BDP	0.027%	0.003%	0.095%
128kB	Low-BDP	0.214%	0.131%	1.579%
	High-BDP	0.058%	0.003%	0.236%
512kB	Low-BDP	0.490%	0.612%	1.389%
	High-BDP	0.061%	0.050%	0.631%
2000kB	Low-BDP	1.149%	0.740%	1.008%
	High-BDP	0.244%	0.178%	0.753%

Table 4.1: Sensitivity analysis : Relative Standard deviation of the median, 25th and 75th percentiles

4.3 Conclusion

In this chapter, we took a closer look at small files transfers and the effect of the scheduler on those connections.

We expected that the larger was the file transmitted, the better the MultiPath TCP performance would be. This is the case in our different experiments and we have shown that a major contributor was the choice of the first path in terms of delays.

On the other hand, the schedulers seem not to influence the performances, this result was obtained in [37] for bulk-transfers and we observed the same results for small-files transfers. Schedulers should in fact be studied with rate-limited applications.

The choice of the best path to start is thus a challenging and important problem to improve MultiPath TCP performances.

Congestion Control Schemes Evaluation

5

In the two previous chapters, we analyzed MultiPath TCP behaviors in different environments, with a fixed network topology and with different transfer sizes. On the other hand, we did not evaluate crucial phenomena that occur on everyday connections, which are the network congestion and the loss of packets.

As our first goal is to measure the different behaviors of MultiPath TCP, we have to analyze connections experiencing congestion in order to evaluate the different available congestion control schemes (CSS). In this chapter, we first compare the theoretical differences between the different schemes. Then we experiment connections on three different kinds of topology : a first one which is similar at a host point-of-view to previous topology used (see Section 3.1) but with random packet-losses, a second one to evaluate the fairness at a shared bottleneck and a last one to evaluate the ability of a scheme to choose the more efficient path.

5.1 Theoretical Differences between Congestion Control Schemes

The different congestion control schemes have been introduced in Section 2.1.2. In Table 5.1, we present a preliminary deduction about the behaviors of each scheme we consider in different kinds of topology. These assumptions are based on their respective research papers [16, 30, 41, 49] and on the analysis provided in [41].

CCS / NT	LIA	OLIA	BaLIA	wVegas
Lossy	✓Aggregation ✓Loss sensitive	✓Aggregation ✓Loss sensitive	✓Aggregation ✓Loss sensitive	✗Aggregation ✗Loss sensitive
Single shared bottleneck	✓Fairness	✓Fairness	✓Fairness	?Fairness
Multiple shared bottlenecks	✓Fairness ✗Responsiveness	✓Fairness ✗Responsiveness	✓Fairness ✓Responsiveness	?Fairness ?Responsiveness

Table 5.1: Assumption on Congestion Control Scheme (CCS) behaviors with relation to the Network Topology (NT)

As a reminder, LIA, OLIA and BaLIA are loss-based while wVegas is delay-based. Since the theoretical differences between LIA, OLIA and BaLIA are outside the scope of this work, we

rather describe the differences between their behaviors than really explain those differences in the following sections.

5.2 Adding Random Packet Loss

In this section, we study the effect of packet-loss on the performance of MultiPath TCP with the four different congestion control schemes. These packet-losses may typically occur in real-world cases when one uses a Wi-Fi connection for example.

5.2.1 Design of the Experiments

Similarly to Chapter 3, we consider a client that exchanges data with a server through two interfaces. In the previous chapters, we considered the loss probability to be equal to 0. Here we consider this probability as an experiment parameter and we define the domain of this parameter as : loss probability $\in \{0\%, 1\%, 2\%\}$[1]. This range of loss probability corresponds to the average loss on a Wi-Fi connection.

We continue to consider low and high-BDP environments as defined in Chapter 3. This gives us four parameters per path : the bandwidth capacity, the delay, the maximum queuing delay and the loss probability. Using the WSP algorithm on this 8-dimensional factors space, we obtained 91 different experiments for the low-BDP environments and 221 different experiments for the high-BDP environments.

To generate the traffic, we use `iPerf` [3] 3.1 with a transmission time of `30s` as the only non-default parameter. The version of the MultiPath TCP Linux Kernel implementation used for all the experiments is the release v0.91 on Ubuntu 14.04, Linux Kernel version 4.1.38, with default path manager and scheduler.

5.2.2 Measurement Results

We now analyze the different results obtained using the four congestion control schemes within the different environments. In the different figures, the values shown are either the Experimental Aggregation Benefit (see Section 3.2) or the Theoretical Aggregation Benefit (see Section 2.1.3). The Experimental Aggregation Benefit aims at comparing a multipath flow to a single-path flow on the best path, while the Theoretical Aggregation Benefit aims at

[1] We only use integer percentage as Mininet has this limitation.

evaluating how the multipath flow fills the available bandwidth. We precise each time which of the two metrics is used to enhance the difference between them for these experiments.

Our analysis is divided into two points. Firstly, we study MultiPath TCP general behaviors when packet losses occur. Then we compare in more details the congestion control schemes.

Global Analysis

We start by analyzing the MultiPath TCP behavior when losses occur no matter which congestion control schemes is being used.

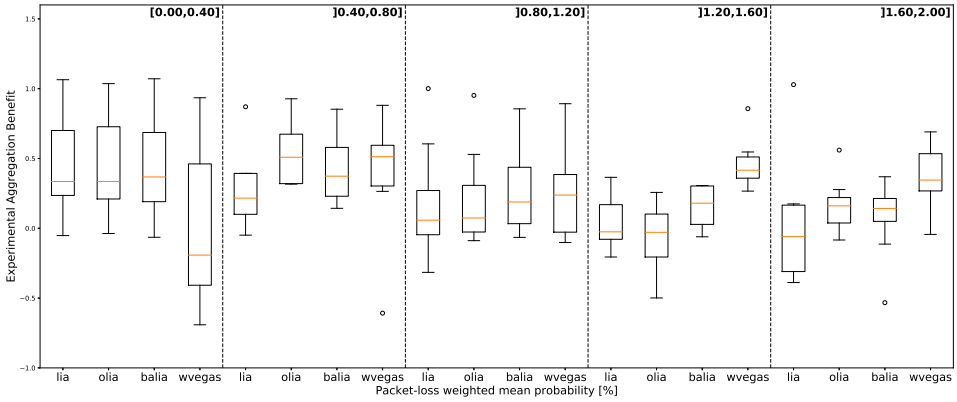
In the previous parts, we were studying how the aggregation was evolving when the difference between the characteristics of the two paths were growing. Now that we consider random losses, we changed the way to group the experiments. In fact, if we used the difference between the loss probability, one could consider two topology settings such that both have a loss probability difference of 0 between their paths. For the first setting, the two paths could have 2% of loss probability while for the second setting, grouped in the same batch, the paths could have a 0% loss probability. One should expect totally different behavior for both connections.

Thus instead of the loss probability difference, we decide to take the mean loss probability experimented by a MultiPath TCP connection. That is, if path 1 have 0%, path 2 1% of loss probability and if path 1 transmits 70% of the traffic then the mean loss probability for the whole connection will be $0 \times 0.7 + 1 \times 0.3 = 0.3\%$. When this mean grows, we expect the aggregation to be smaller at least for loss-based congestion control schemes. Figure 5.1 shows both theoretical and experimental aggregation benefit in low-BDP environment. In those figures, we can notice two distinct things, in one hand, MultiPath TCP with loss-based CCS suffers from random packet-loss in comparison with TCP, as shown in Figure 5.1a. This is caused by the CCS that moves the traffic away from the most congested path, and thus the experimental aggregation tends towards 0. In Figure 5.1b, the theoretical aggregation benefit is shown. As we expect, MultiPath TCP fulfills less the paths when the mean loss probability grows.

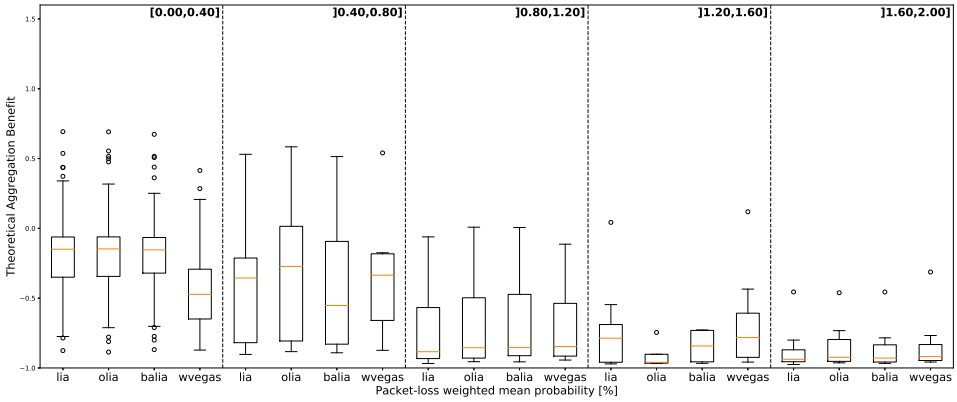
Congestion Control Schemes Evaluation

In Figures 5.1 and 5.2, we can see that the different congestion control schemes perform differently. It might be interesting to look in more details how each CCS behaves.

We first analyze the wVegas algorithm which is a delay-based congestion control scheme. The congestion window size adaptation is rather recomputed after each transmission round in



(a) Experimental AB

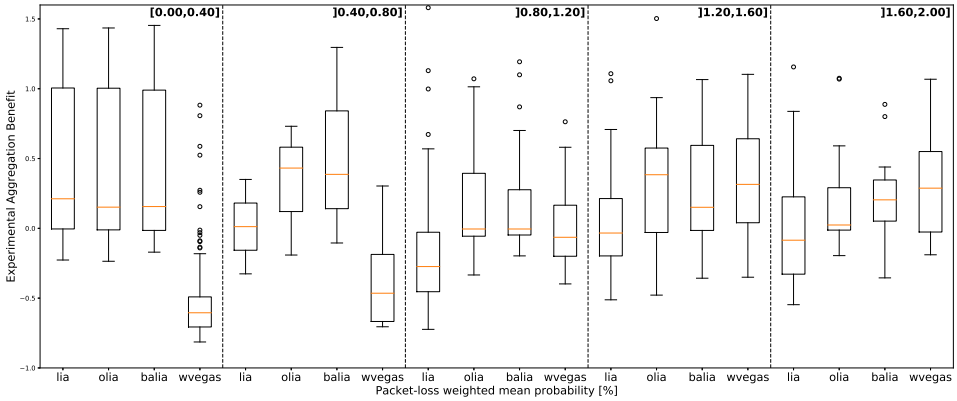


(b) Theoretical AB

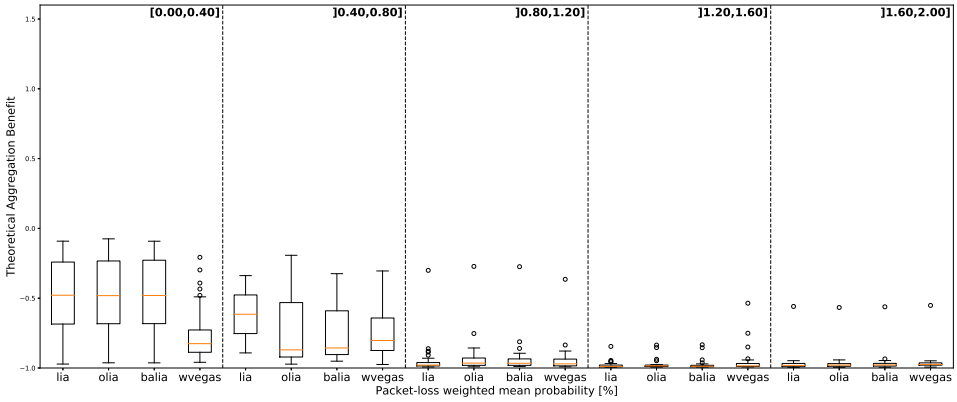
Figure 5.1: Aggregation Benefit wrt loss sum in low-BDP environment with packet-loss

function of the variation of the RTTs than after each loss. As we can see in Figures 5.1a and 5.2b, it performs badly when there are few losses, i.e. for $< 1\%$ of the packets, in comparison with other schemes. As we compared it with TCP NewReno (which is loss-based), it performs better when the mean loss rate grows. Indeed when a loss occurs, the congestion window size is not adapted. As the loss is assumed random, i.e. not caused by network congestion, the RTT before and after this loss can be assumed to be in the same magnitude. Thus the congestion window keeps the same size before and after a loss. This behavior might be interesting in this particular situation, e.g. when one uses Wi-Fi where we have random losses, because MultiPath TCP should not treat any type losses as indication of network congestion.

Secondly, for the three linked adaptation schemes, we can observe in those experiments results that for a small loss mean probability, their behaviors are similar. When the number of packet-losses increases, it appears that LIA achieves more rapidly the goal of load balancing as the median tends towards 0 more rapidly than the others.



(a) Experimental AB



(b) Theoretical AB

Figure 5.2: Aggregation Benefit wrt loss sum in high-BDP environment with packet-loss

Traffic distribution

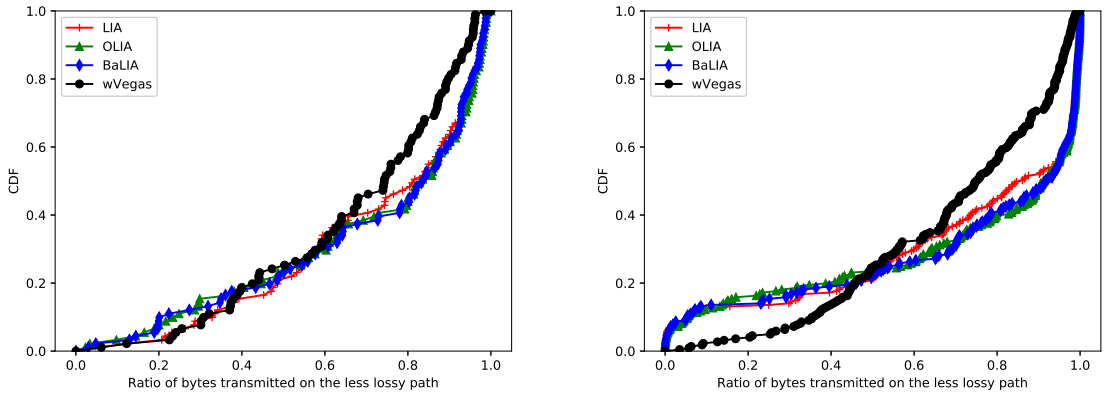
We are now interested by the load balancing between the two paths. To avoid loss events, one would try to under load the more lossy path and prefer to push more packets on the

less lossy path. We now wonder if the congestion control schemes for MultiPath TCP apply this intuitive principle and if they under use one of the two paths when there are too much packet-losses on this path.

When we analyzed the traffic distribution for bulk-transfers in Section 3.3.1 and 3.3.2, we grouped the experiments in two batches. For example, in one hand, the experiments with the smallest propagation delay on the first path and in the other hand the experiments with the smallest propagation delay on the second path. Here, we want an information on the lossy behavior of the paths and, particularly, we want to analyze the traffic distribution on the *less lossy* path.

We thus need a definition of the *less lossy* path : this could be for example the path with the lowest loss probability parameter. We found that this measure was not precise enough and did not take into account other TCP retransmissions that could occur in high-BDP environments for example. We defined the *less lossy* path as the path which has the lowest retransmission ratio in the single path TCP connections experiments which use the NewReno congestion control scheme.

We also decide not to consider whether the less lossy path is the first or the second path used to initiate the MultiPath connection. We show our results in the Figure 5.3.



(a) Low-BDP environments

(b) High-BDP environments

Figure 5.3: CDF of the Ratio transmitted on the *less lossy* subflow of the MultiPath TCP connection

In the low-BDP environments, Figure 5.3a, all of the four congestion control schemes behave similarly, even if we notice a small difference between the loss-based schemes and wVegas. The latter tends to always distribute the traffic on the more lossy path and doesn't push all the traffic on one unique path. In the high-BDP environments, Figure 5.3b, the behaviors are much more different between the two types of CCS. While wVegas tends to have the same

behavior as in low-BDP environments, the other CCS tend to distribute the major part of the traffic on one of the two paths. The loss-based CCS load severely the less-lossy path. On the graphs, one could observe that for 40% of our experiments, the ratio of bytes transmitted through the less lossy path is above 0.95, i.e. nearly all the traffic is pushed on one and only one path.

We see here that there are a difference between the loss-based and the delay-based CCS. While wVegas seems to have weaker performances for small values random losses, and the best relative performance when the loss probability grows. Here, we found a lot of similarities between the three other CCS in terms of performances and behaviors, however, we found that both BaLIA and OLIA are at least as good as LIA and often better than LIA.

5.2.3 Sensitivity Analysis

We present here the results of the sensitivity analysis as presented in Sections 3.3.3 and 4.2.3. We have redone two times the experiments in other space-filling factors. Once again the relative standard deviations are small for all of the three statistical values. The number of experiments we launched is thus sufficient to have a good overview on the behaviors of MultiPath TCP using the different CCS.

We don't analyze the sensitivity of the Experimental Aggregation Benefit and assume its validity for those experiments. The hypothesis we use is explained in the Section 3.3.3.

CCS	Environment	Median	25th percentile	75th percentile
LIA	Low-BDP	1.875%	0.488%	0.141%
	High-BDP	0.970%	0.815%	1.284%
OLIA	Low-BDP	1.159%	0.350%	0.176%
	High-BDP	0.869%	0.974%	1.221%
BaLIA	Low-BDP	1.385%	0.591%	0.177%
	High-BDP	1.216%	0.960%	1.496%
wVegas	Low-BDP	0.439%	0.363%	1.771%
	High-BDP	0.421%	0.668%	0.278%

Table 5.2: Sensitivity analysis : Relative Standard deviation of the median, 25th and 75th percentiles

5.3 Fairness at Shared Bottleneck

When one start studying MultiPath TCP connection and getting interested by the congestion control scheme, an obvious question that might be asked could be : "Why not just run regular TCP congestion control on each subflow ?" [49].

As explained in Section 2.1.2, ad absurdum, we can use the topology shown in Figure 5.4 with two clients, one with two interfaces linked to a server and the other with one interface linked to an other server. All the flows share a same bottleneck. If the MultiPath TCP connection uses regular TCP congestion control on each path then the three flows would get the same throughput and thus the total MultiPath flow would have twice as much throughput as the single path flow. This does obviously not respect the fairness principle. The connection reaches fairness when both connections share half the bandwidth.

Figure 5.4: A MultiPath TCP flow should take as much as a single TCP flow

We thus probe how the different congestion control schemes manage this fairness.

5.3.1 Design of the Experiments

We use the scenario given in Figure 5.4 where the bottleneck bandwidth capacity is fixed to 15 Mbps. We use `iPerf` application to generate a bulk-transfer. To measure the fairness, we launch two different scenarios of experiments :

1. A 5 minutes long experiment with both connections, the single-path and the multipath launched at the same time to have a steady state characterization.
2. A 2 minutes long experiment with three phases of 40 seconds : (I). The MultiPath TCP connection alone to reach a steady state, (II). Both connections to observe the congestion control responsiveness and (III) the MultiPath TCP connection alone to observe congestion control recovering.

We do not use a space-filling approach here because the goal is to analyze a particular situation and understand the behaviors of the different control scheme in a given situation. However,

as there is only one bottleneck, changing the parameters would not have an impact on the fairness.

5.3.2 Measurements Results

The fairness may be computed by different ways. Here, we will consider Jain's fairness index [28] and Min-Max fairness. The first one is computed as

$$\mathcal{J} = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \times \sum_{i=1}^{n} x_i^2},$$

with n equal to 2 which represents the two clients and x_i the throughput of client i. The allocation is said to be Jain fair when this index equals to 1. The second concept may be expressed as follows : an allocation is Max-Min fair if and only if the allocation is feasible and the only way to increase the allocation of a client would decrease the allocation of an other client. Thus here, we expect a Jain's fairness index of 1 and a maximal allocation of the bandwidth for both clients.

In the table below, we report Jain's indices, the allocation for each path and the percentage of the bottleneck capacity used by the connection. In order to cope with possible variations with the same environments settings, the experiments on each setting are repeated 5 times and we take the median value for the throughput[2] and the fairness index.

		LIA	OLIA	BaLIA	wVegas
MPTCP	Throughput(kb/s)	6922	7155	7608	7303
	Bandwidth used (%)	46.15	47.70	50.72	48.69
SP-TCP	Throughput(kb/s)	7909	7388,5	7162	7055
	Bandwidth used (%)	52.73	49.26	47.75	47.03
Both	Total throughput(kb/s)	14807	14736	14728	14350
	Total bandwidth used (%)	98.71	98,24	98.18	95.67
	Jain's Index	0.9923	0.9993	0.9974	0.9997

Table 5.3: Experiments I : Fairness in a steady state connection. The maximal bandwidth capacity is fixed to 15000kb/s

In the Table 5.3, we can see that all of the four algorithms have a Jain's fairness index higher than 0.99, as the range of this index is $[0.5, 1]$, those results may be considered as a good Jain fairness. Looking at the Min-Max fairness principle, the analysis is quite different. For wVegas, a better allocation exists without decreasing the total throughput as the bottleneck

[2]Pay attention to the fact that the median of sums is not equal to the sum of medians.

can be more loaded, it lacks at using all the available bandwidth and doesn't achieve a Max-Min fairness. However the wVegas has the better Jain's fairness index.

In the Table 5.4 we are studying our second type of scenarios where a TCP connection is launched after the MultiPath TCP connection. We first focus on the second phase, i.e. when both TCP and MultiPath TCP connections are open. We observe that wVegas is the only CCS which reach the same as the throughput obtained in the first experiment, see Table 5.3, within **40s** and has the best Jain's fairness. We explain this behavior by the fact that the queueing delay grows fast when the a new flow shares a bottleneck with a MultiPath TCP subflow and the congestion window is efficiently adapted. Concerning the loss-based schemes, one can observe that OLIA performs better than BaLIA and LIA.

			LIA	OLIA	BaLIA	wVegas
Phase I	MPTCP	Throughput (kb/s)	13438	13138	13697	13445
		Bandwidth used (%)	89.59	87.60	91.32	89.63
Phase II	MPTCP	Throughput (kb/s)	6719	7615	7055	6783
		Bandwidth used (%)	44.79	50.77	47.03	45.22
	SP-TCP	Throughput (kb/s)	5404	6612	5686	7579
		Bandwidth used (%)	36.03	44.08	37.91	50.53
	Both	Total Throughput (kb/s)	12584	14247	13072	14397
		Total bandwidth used (%)	83.90	94.98	87.14	95.98
		Jain's Index	0.9911	0.9949	0.9822	0.9976
Phase III	MPTCP	Throughput (kb/s)	10209	13899	14596	13667
		Bandwidth used (%)	68.06	92.67	97.31	91.12

Table 5.4: Experiments II : Fairness and Responsiveness, MultiPath connection last for **120s**, SP connection starts at time **40s** and end at time **80s**. The maximal bandwidth capacity is fixed to 15000kb/s.

Secondly, we can focus on the throughput in the phase III. In [41], *Peng et al.* showed that BaLIA was more responsive when the flow sharing the bottleneck stops. BaLIA rapidly recovers and fulfills the bandwidth while OLIA and LIA have a slower convergence. We confirm this analysis in our results where LIA has the worst throughput and BaLIA is, in fact, better than OLIA.

5.4 Efficient Path Choice

Finally, one could ask if a multipath connection will tend to use a path rather than another one to decrease the congestion on the network and choose the most efficient path. A network

often used to illustrate this question is the topology shown in Figure 5.5 based on [49], where there are 3 clients with 2 interfaces each, 3 bottlenecks and 3 servers. Each client has a flow which goes through one bottleneck and the other transits through the two other bottlenecks.

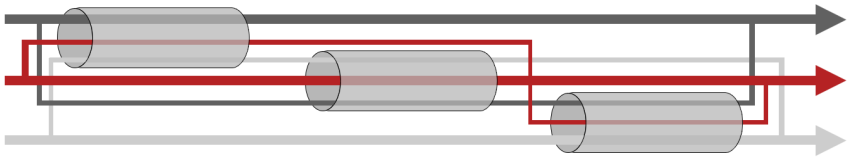


Figure 5.5: What path should each MultiPath TCP connections prefer?

A simple example is to set the same bandwidth capacity constraint to each bottleneck. Let us impose 10 Mbps. If we only think about fairness, each subflow should get a throughput of 3.33 Mbps so, each multipath flow would achieve a 6.66 Mbps throughput. But the optimal option is when each multipath flow only sends traffic through the path with only one bottleneck, e.g. the largest links in Figure 5.5. In this case, each multipath flow achieves 10 Mbps. Generally speaking, it appears that the multipath connection *should prefer the least congested path* [49].

5.4.1 Design of the Experiments

We use the case presented in Figure 5.5 as a base case for our study.

In our experiments, we set the bandwidth capacity at 10 Mbps for each bottleneck. The traffic is generated using `iPerf` application. The scenario is the following : we start three 5-minutes-long connections from the three clients to observe the steady state throughput and the mean allocation on each part. The connections are started one at a time, separated by a random number of seconds in the range [1,5].

5.4.2 Measurements Results

To compare the performances of the different congestion control, we compute two metrics : the average throughput and the mean allocation on each path for the four different algorithms.

We were not able to launch those experiments multiple times due to a non-deterministic bug, those results are obtained by launching the experiment only once per congestion control. As we don't average the results on multiple experiments, our unique result, shown in Table 5.5, may be noise sensitive.

Here we can note that the four control schemes use a big part of the most efficient path. In this case, all of the four CCS tend to fulfill the bandwidth of the first path used to initiate MultiPath TCP connection. We also observe that wVegas fills less the available bandwidth than the other. As we were not able to launch multiple experiments with this topology, we were not able to confirm those behaviors.

		LIA	OLIA	BaLIA	wVegas
Subflow I	Throughput (kb/s)	9424	9481	9697	8649
	Traffic distribution (%)	89.12	82.74	89.65	79.11
Subflow II	Throughput (kb/s)	1151	1978	1119	2280
	Traffic distribution (%)	10.88	17.26	10.35	20.80
Total	Throughput (kb/s)	10575	11459	10816	10929

Table 5.5: Efficient path choice : Statistics of the first client which opens a MultiPath TCP connection

5.5 Conclusion

In this chapter, we described the behavior of four different congestion control schemes and their impact on MultiPath TCP performances. We use a panel of scenarios to evaluate their strengths and weaknesses.

Through our experiments, we could analyze that the fundamental difference between the loss-based congestion control schemes and the delay-based one leads to fundamental difference in the behavior of MultiPath TCP. While it is a fair scheme, wVegas lacks in efficiently balancing the load on two subflows in environments where there is a low loss probability. However, this scheme might be useful in cases where the losses are not created by congestion but rather when they are random, e.g. when using Wi-Fi access point.

Since the base principle between the three other schemes is the same, they appear to have comparable behaviors. We also confirmed a result presented in [41] which shows the responsiveness of BaLIA in the case of a shared bottleneck.

This analyzes let us broaden our understanding of the MultiPath TCP congestion control scheme which is an important factor influencing the transmission performances in case of packet-losses or congested networks.

This concludes our analyzes chapters. We now enter into the modeling part.

A Model for MultiPath TCP Throughput and its Validation

6

Now that we have acquired extensive understanding about MultiPath TCP behavior, we wish to model it. In this part, we develop an analytic characterization of the steady state throughput as a function of loss rate and round-trip time of a bulk transfer using MultiPath TCP with two subflows. We test our model in various situations.

This part is thus organized in the following way. In Section 6.1 we furnish the details about the establishment of our model. Then in Section 6.2, we compare our predictions with a set of measurements of MultiPath TCP connections in lossy environments with other TCP flows.

6.1 Modeling MultiPath TCP Congestion Control

In this section, we provide an extension of the TCP throughput model given by *Padhye et al.* in [40]. This stochastic model has been presented in the State of the Art, Section 2.2. We start by giving an overview of the ideas that compose the model. We then introduce some hypotheses about the behavior of MultiPath TCP. Then the modeling of the different components is explained. We find some results previously presented in the literature. Finally, we bring a formula for MultiPath TCP throughput in a congested environment.

6.1.1 Overview

Let us define W_i the congestion window of path i, with $i = 1,2$, i.e. we only consider a connection with two paths. There are several ways of controlling the congestion window while using MultiPath TCP. We decide to model the linked increases algorithm that has been defined in [44] and that we now remind.

Algorithm Linked increases (LIA)

- increase W_i by $\min(a/W_{tot}, 1/W_i)$ per ack on path i ;
- decrease W_i by $W_i/2$ per loss event on path i.

where $a = \mathbb{E}[W_{tot}] \frac{\max_i \mathbb{E}[W_i]/RTT_i^2}{(\sum_i \mathbb{E}[W_i]/RTT_i)^2}$.

RTT_i is the round-trip time of path i and $\mathbb{E}$ the expectation operator. The min of the increase phase is necessary in order to ensure that the MultiPath TCP flow would take no more capacity on each path than a single-path TCP flow would [44].

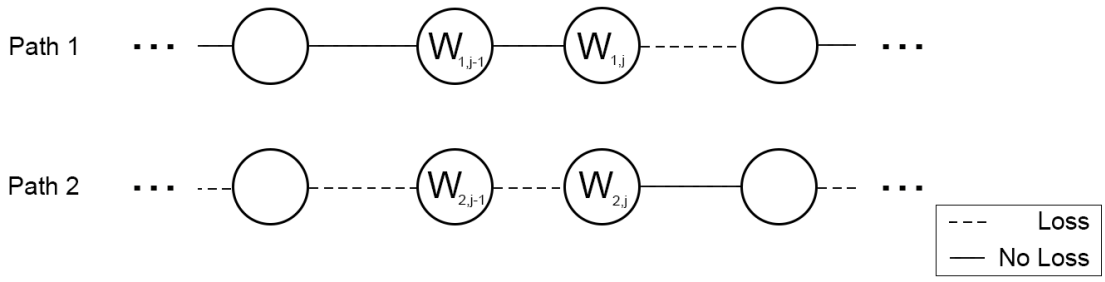


Figure 6.1: Representation of the Stochastic Processes
From the definition of the LI period, both processes evolve at same time

We define a Loss Indication period (LI period) to be a period between two loss indications, no matter on which path it happens. Loss indications can be of several types but we only consider triple-duplicates indications. For the j-th period, we define $P_{i,j}$ the number of packets sent on path i, and $W_{i,j}$ the window size of path i at the end of the period. The model also uses some constants that we now define. Each path i experiments a loss rate p_i. For a MultiPath TCP connection, one can compute a mean loss rate p. One may also be interested in the ratio r_i of loss occurring on path i. That is, when the connection is terminated we compute the number of losses that have occurred on path i divided by the total number of losses that have occurred. Finally, we define b, the number of packets acknowledged by a received ACK [40]. Table 6.1 and 6.2 give a summary of the variables and constants of our model, respectively. In the remainder of the text, each term is reintroduced when it is necessary.

Then we model the evolution of the congestion windows by two stochastic processes with states $\{W_{i,j}\}_j$, which can be represented as in Figure 6.1. From the definition of the period, the states of the processes evolve at the same times, i.e. when a loss occurs no matter on which path.

In our modeling, we assume that a loss in a period is independent of losses of any other periods. This assumption is based on [40]. Finally, this model is valid in a lossy environment or when the multipath flow is competing with other flows. When the multipath flow is experimenting no loss, one should expect the throughput to be the sum of single-path flows on the same paths [49].

In the remainder of this chapter, index i concerns the path considered while index j refers to the LI period. The $\sum$ operator concerns the two considered paths.

Variable	Definition
$W_{i,j}$	Congestion window of path i at the end of period j
$P_{i,j}$	Number of packets sent on path i during period j
P_j	Total number of packets sent during period j
B_i	Throughput of path i

Table 6.1: Variables of the model

Constant	Definition
p_i	Loss rate on path i
p	Mean loss rate
r_i	Ratio of loss occurring on path i
RTT_i	Round-trip time of path i
b	Number of packets acknowledged by a received ACK

Table 6.2: Constants of the model

6.1.2 Hypotheses

We now introduce three hypotheses that allow us to approximate a solution for MultiPath TCP throughput. We also present the implication of such hypotheses.

Hypothesis 1 Round-trip times are similar.

This hypothesis implies that the growth of the congestion windows, as described by the linked increases algorithm, is no more limited by the same growth a TCP flow would undergo. Indeed for equals RTTs, one can prove that [30] :

$$\min(a/W_{tot}, 1/W_i) = a/W_{tot}$$

This hypothesis might seem strong. However, we can still consider different RTTs in the rest of our equations. The goal of this hypothesis is in fact to remove the min of the equation that does not allow us to directly compute an expression of the throughput.

Hypothesis 2 The effects of the slow-start are negligible.

This implies that we neglect the start of the connection and that we only consider the steady-state of the transfer. One of the implication of such hypothesis is that the growth of the

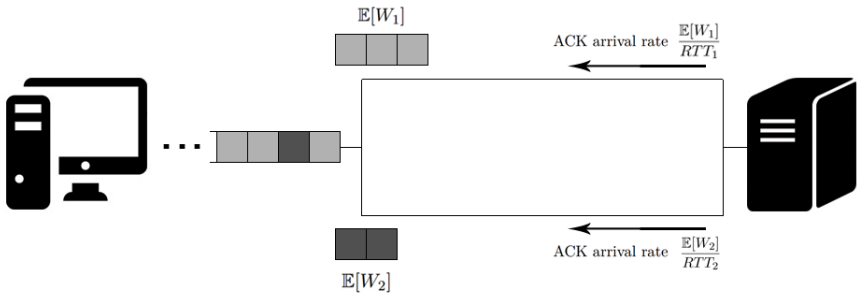


Figure 6.2: Example of scheduling of packets

congestion windows will be linear. Indeed, one has :

$$\frac{a}{\mathrm{W}_{tot,j}} \approx \frac{a}{\mathbb{E}[\mathrm{W}_{tot}]} = \frac{\max_i \mathbb{E}[\mathrm{W}_i]/\mathrm{RTT}_i^2}{\left(\sum_i \mathbb{E}[\mathrm{W}_i]/\mathrm{RTT}_i\right)^2} \qquad \forall j$$

The linearity in the growth of the congestion windows is consistent with the observations of the experiments of [44] about linked increases algorithm.

Hypothesis 3 The scheduling of the packets is self-clocked.

As it has been shown in [37] and in Chapter 4 of this document, the scheduler has few impacts on the throughput for bulk-transfers. The scheduling of packets is in fact self-organized by the *ack-clock phenomena*. This is due to the fact that the congestion windows are full and thus the scheduling of packets is organized by the arrival of ack which frees up space in the windows.

We thus decide to neglect the scheduler and to use the ack-clock phenomena to model the packets scheduling. We provide an example in Figure 6.2. One can first observe that the congestion window is bigger on path 1. We also suppose that the RTT is smaller on this path. Thus the first path is a strictly better path than the second one. The windows are full and are waiting for acks to free up space. The ack arrival rate is directly related to the throughput of the path that can be approximated by $\frac{\mathbb{E}[\mathrm{W}_i]}{\mathrm{RTT}_i}$ at the steady-state. As path 1 has a better throughput, more packets will be scheduled on it as shown in the list of packets waiting to be scheduled.

6.1.3 Modeling the Variables

In this section, we bring details about the modeling of each variables. We illustrate our equations with examples.

Mean number of packets sent during one LI period : $\mathbb{E}[P]$

We first wish to find the mean number of packets sent before a loss occurs no matter on which path it happens, i.e. during a LI period.

We thus need to compute the mean loss rate p that packets will undergo. From the ack-clock phenomena, this loss rate can be expressed as a weighted sum of the individual loss rate p_i where the weights are ratios of throughputs. From hypothesis 2 of steady-state, the throughput of path i can be approximated by : $\frac{\mathbb{E}[W_i]}{RTT_i}$. And thus one has :

$$p = \frac{p_1 \frac{\mathbb{E}[W_1]}{RTT_1} + p_2 \frac{\mathbb{E}[W_2]}{RTT_2}}{\frac{\mathbb{E}[W_1]}{RTT_1} + \frac{\mathbb{E}[W_2]}{RTT_2}}$$

From this mean loss rate, one can compute the number of packets sent during one LI period j, i.e. P_j. Indeed, this variable is the number of Bernoulli trials needed to get one success and thus follows the following distribution :

$$\text{Prob}(P_j = k) = (1-p)^{k-1} p$$

which is a geometric distribution. The mean number of packets sent during one LI period is thus given by :

$$\mathbb{E}[P] = \frac{1}{p}$$

which is the value we were looking for.

Example 1 Suppose $p_1 = 0.01$, $p_2 = 0.02$ and that the ratio of throughput is 0.7 for path 1 and 0.3 for path 2. As path 1 has a better throughput than path 2, we expect the mean loss rate to be closer to the loss rate of path 1. We have $p = 0.01 \times 0.7 + 0.02 \times 0.3 = 0.013$. Then the mean number of packets sent in one LI period is : $\mathbb{E}[P] = \frac{1}{p} = 76.92$ packets.

Mean number of packets sent on path i during one LI period : $\mathbb{E}[P_i]$

Now that we have the mean number of packets sent in one LI period, one can simply compute the mean number of packets sent on a certain path during the same period. Once again from hypotheses 2 and 3, it can be expressed as a ratio of throughputs at steady-state. Thus one has :

$$\mathbb{E}[P_i] = \frac{\frac{\mathbb{E}[W_i]}{RTT_i}}{\frac{\mathbb{E}[W_1]}{RTT_1} + \frac{\mathbb{E}[W_2]}{RTT_2}} \mathbb{E}[P] \tag{6.1}$$

Example 2 Taking the same values of constants than Example 1, one has $\mathbb{E}[P_1] = 0.7 \times 76.92 = 53.84$ and $\mathbb{E}[P_2] = 0.3 \times 76.92 = 23.06$. We easily verify that $\mathbb{E}[P_1] + \mathbb{E}[P_2] = \mathbb{E}[P]$.

This value is useful as the growth of the congestion window is directly dependent of the number of acks received on the path, and thus of the number of sent packets on the path.

Ratio of loss occurring on path i : r_i

In order to derive how the congestion windows evolve, we have to know the ratio of loss occurring on each path. We illustrate this concept by a first example.

Example 3 In Figure 6.1, fours states are represented. This implies that three losses occur between these four states from the definition of the LI period. One on path 1, two on path 2. Thus the ratios of loss are 1/3 and 2/3 for path 1 and 2 respectively.

We wish to find the ratio at steady-state. We take another example to derive an expression.

Example 4 Let us consider a whole bulk-transfers with same parameters than Examples 1 and 2. Then the ratio of loss of a path will be the quantity of losses that have occurred on this path divided by the total number of losses that have occurred. As we do not know those numbers, one may approximate them by using the steady-state ratio of throughput and the loss rate of each path. Thus : $r_1 = 0.01 \times 0.7/(0.01 \times 0.7 + 0.02 \times 0.3) = 0.54$ and in the same manner $r_2 = 0.02 \times 0.3/(0.01 \times 0.7 + 0.02 \times 0.3) = 0.46$. We easily verify that $r_1 + r_2 = 1$.

From Example 4 and Hypotheses 2 and 3, these ratios can thus be expressed in the following way :

$$r_i = \frac{p_i \frac{\mathbb{E}[W_i]}{RTT_i}}{p_1 \frac{\mathbb{E}[W_1]}{RTT_1} + p_2 \frac{\mathbb{E}[W_2]}{RTT_2}} \tag{6.2}$$

Dynamic of the Congestion Windows : $W_{i,j}$

We now have all the elements in order to derive an expression for the dynamic of the congestion windows.

Let us suppose arbitrarily that :

$$\frac{\mathbb{E}[W_1]}{RTT_1^2} \geq \frac{\mathbb{E}[W_2]}{RTT_2^2}$$

Then :

$$a = \mathbb{E}[W_{tot}] \frac{\mathbb{E}[W_1]/RTT_1^2}{\left(\sum_i \mathbb{E}[W_i]/RTT_i\right)^2}$$

From hypothesis 2, the dynamic of the congestion window for linked increase algorithm between two periods can be approximated by :

$$W_{i,j} = \begin{cases} \frac{W_{i,j-1}}{2} + \frac{P_{i,j}}{b} \times \frac{\mathbb{E}[W_1]/RTT_1^2}{\left(\sum_i \mathbb{E}[W_i]/RTT_i\right)^2} & \text{(if loss on i)} \\ W_{i,j-1} + \frac{P_{i,j}}{b} \times \frac{\mathbb{E}[W_1]/RTT_1^2}{\left(\sum_i \mathbb{E}[W_i]/RTT_i\right)^2} & \text{(if loss not on i)} \end{cases}$$

We know the ratio r_i of cases where the loss occurs on path i or not. Thus from the linearity of the expectation operator, one has :

$$\mathbb{E}[W_i] = r_i \left(\frac{\mathbb{E}[W_i]}{2} + \frac{\mathbb{E}[P_i]}{b} \frac{\mathbb{E}[W_1]/RTT_1^2}{\left(\sum_k \mathbb{E}[W_k]/RTT_k\right)^2} \right) + (1 - r_i) \left(\mathbb{E}[W_i] + \frac{\mathbb{E}[P_i]}{b} \frac{\mathbb{E}[W_1]/RTT_1^2}{\left(\sum_k \mathbb{E}[W_k]/RTT_k\right)^2} \right)$$

For $i = 1, 2$, this results in a system of two equations that can be solved using expressions 6.1 and 6.2. The resolution steps are omitted here but are presented in Appendices B. We only present the last step as it gives interesting results as shown in the following system :

$$\frac{\mathbb{E}[W_1]}{RTT_1} + \frac{\mathbb{E}[W_2]}{RTT_2} = \frac{\sqrt{2/(b \cdot p_1)}}{RTT_1} \tag{6.3}$$

$$p_2 \mathbb{E}[W_2] = p_1 \mathbb{E}[W_1] \tag{6.4}$$

Equation 6.3 is a direct consequence of the design of LIA. It can be read as follows : the sum of rates of the two paths is equal to the rate a TCP flow would get on the best path. Indeed, it can be shown that TCP allocates a window of $\sqrt{2/p_i}$ packets for path i. This equation has been used in order for MultiPath TCP to achieve fairness with TCP by computing the value of the aggressiveness factor a.

Equation 6.4 implies that if, for example, a path has a loss rate two times bigger than the other path, then MultiPath TCP will allocate it a congestion window that is two times smaller than the one of the other path.

Injecting equation 6.4 in 6.3, we finally find :

$$\mathbb{E}[W_i] = \frac{1}{p_i} \cdot \frac{\sqrt{2/(b \cdot p_1)}/RTT_1}{\sum_k 1/(p_k RTT_k)} \tag{6.5}$$

This result can be found by applying the fixed-point analysis of [49] and is also presented in a more general form in [30]. We can see that MultiPath TCP allocates a window size that is inversely proportional to loss rate of the path and such that the total rate is equal to the rate

a TCP flow would get on the best path. We do not bring the proof but one may suppose that this expression can be written in a more general way :

$$\mathbb{E}[W_i] = \frac{1}{p_i} \cdot \frac{\max_k \sqrt{2/(b \cdot p_k)}/RTT_k}{\sum_k 1/(p_k RTT_k)} \tag{6.6}$$

which is the same expression than given in [30].

6.1.4 A Formula for MultiPath TCP Throughput

Now that we have determined the congestion window at steady-state, we can furnish a simple approximation of the throughput. Let B_i the throughput of path i. Then we approximate this throughput by :

$$B_i = \frac{\mathbb{E}[W_i]}{RTT_i} = \frac{1}{p_i RTT_i} \cdot \frac{\sqrt{2/(b \cdot p_1)}/RTT_1}{\sum_k 1/(p_k RTT_k)}$$

which also supposes arbitrarily that $\frac{\mathbb{E}[W_1]}{RTT_1^2} \geq \frac{\mathbb{E}[W_2]}{RTT_2^2}$. Finally, one has :

$$B = B_1 + B_2 = B_1^{TCP}$$

which is the total throughput achieved by a MultiPath TCP connection in a lossy environment using LIA as congestion control scheme.

6.1.5 Discussion

One might be surprised by equation 6.3 that constrains MultiPath TCP throughput to the best TCP throughput. How could the throughput be improved in this case? This question has been studied in [49] and the answer lies in the fact that whether or not there are competing flows during the connection. When there is no competing traffic and if the links are underutilized, then there is no loss and MultiPath TCP throughput achieves the sum of TCP throughputs on the same paths. When there are competing flows, LIA determines the best path based on the current loss rate and tries to achieve a total throughput that is equal to the rate a TCP flow would get with this loss rate [49].

This model only considers two paths, one might ask if we could easily extend it to n paths. It is quite clear that the variables can be extend to n paths. The only problem would come from the system of equations that we would have to solve for the dynamic of the congestion windows. We do not know if such system can be solved. However we expect the solution to be the same as what we found for 2 paths, i.e. equation 6.6.

A last point that should be discussed is the validity of the model for a wide range of different RTTs. Indeed, hypothesis 1 suppose that the RTTs are similar in order to simplify the increase

phase of LIA. In this case, one has $a/\mathrm{W}_{tot} \geq 1/\mathrm{W}_i$ and thus the increase is only managed by a/W_{tot}. We would like to know in which cases $a/\mathrm{W}_{tot} \geq 1/\mathrm{W}_i$ occurs. However it seems very hard to predict and it seems that the only way to know would be to test the model on a very large range of domains. In the validation of Section 6.2.1, we test the model on a range $[10, 980]$ ms of difference. Model gives good results. However the other parameters are fixed. A better analysis should be more general and could be future work.

6.2 Model Validation

In this second section, we present the prediction of our model where a multipath flow is competing with other simple-path flows. We introduce a small tool based on Section 6.1 to easily compute the expected throughput.

In order to rapidly estimate the throughput, we have written a small Python script called `mptcp-calculator` [46]. We can use it in the following way :

```
$ python mptcp-calculator.py -p p1,p2 -r rtt1,rtt2 -b b -m MSS
```

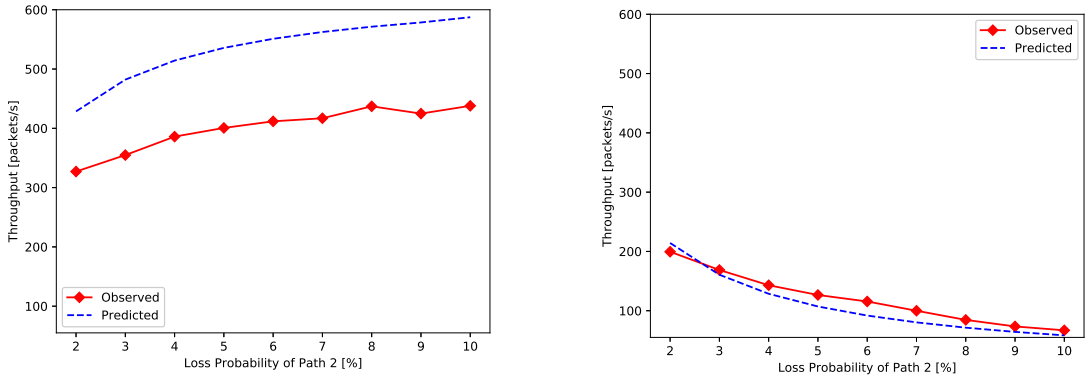
$p1$ and $p2$ are the loss rates, $rtt1$ and $rtt2$ are the round-trip times in milliseconds, b is the number of packets acked for one received ack and MMS is the maximum segment size. If option m is not specify, then the results will be expressed in packets per second. The s option allows to print more statistics.

6.2.1 Validation

We now propose a validation of our model in lossy environments. Each flow of the multipath connection goes through a bottleneck. We design two types of experiments. In the first one, we make the loss rate of the second path vary. In the second, we vary the round-trip time of the second path. The other parameters are fixed such that the first path is always the best one.

Variation of the Loss Rate

We fix both bottlenecks bandwidth capacities to 20 Mbps and round-trip times to 20 ms. The loss rate of first path is fixed to $0.01 = 1\%$. We make vary the second path loss probability from 2 to 10%. As *Mininet* only allows integer values for loss rate, this gives us 9 experiments. We run `iPerf` during 300 seconds in order to reach equilibrium and to reduce the effect of slow start. Results are shown in Figure 6.3.



(a) First path

(b) Second path

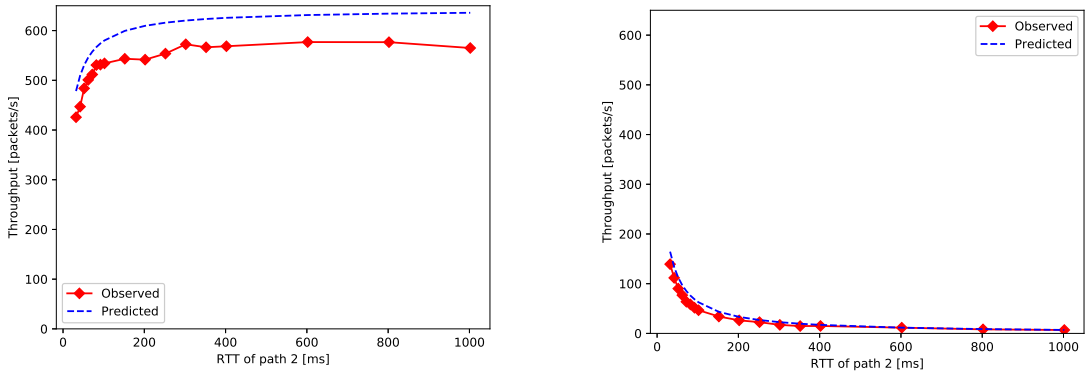
Figure 6.3: Validation on different loss rate

With such loss rate and RTT, the target throughput should be close to 700 packets/s. We start with the first path shown in Figure 6.3a. We observe that both curves grow in a similar way when the loss rate of the other path grows. The *root-mean-square error* (RMSE), which measures the differences between values predicted and observed, is 135.69 packets. In this case, the model always overestimate the empirical tests. We attribute this to the fact that the model does not take into account some of the limiting factor of MultiPath TCP such as head-of-line blocking.

Concerning the second path, the predictions provide satisfying results. The RMSE is 15.45 packets, which is quite smaller than for the first path. We also observe that both curves decrease in a same manner.

Variation of the Round-Trip Time

The bottlenecks bandwidth capacities are still fixed to 20 Mbps. The loss rates are $0.01 = 1\%$ for path 1 and $0.02 = 2\%$ for path 2. We set the RTT of path 1 to 20 ms and we vary the RTT of path 2 from 30 to 1000 ms in order to study the impact of the similar RTTs hypothesis. We have designed 15 experiments in this range of RTT.



(a) First path

(b) Second path

Figure 6.4: Validation on different round-trip times

The results are shown in Figure 6.4. For the first path, which is the best, the RMSE is 54.22 packets. We observe that both curves evolve in similar manner but that the model still overestimates the throughput. We attribute this to the fact that the model does not take into account phenomena such as head-of-line blocking which is typically observed when the RTTs are different.

Once again, the model for the worst path provides satisfying results. The RMSE is 14.40 packets which is quite low and the predicted curve is very close to the experimental observations.

The impact of the similar RTTs hypothesis seems to be limited in these cases.

6.3 Conclusion

We proposed a new model for MultiPath TCP throughput using linked increase algorithm. The model provides a formula for the congestion window that depends on the loss rates and round-trip times of each path. This formula has already been observed in the literature [30].

We tested our model on a small validation. The model tends to overestimate a bit the throughput of the best path while it gives really good results for the worst path. We attributed this to the fact that some phenomena that reduces MultiPath TCP throughput are not considered by the model.

We believe there is still work to do on this model, we now offer some lines of work. First, we

think that the model could be extend to several paths. However, the system of equations would be harder to solve. Then, the model could also consider other congestion control schemes. It seems possible at least for B*a*LIA. Finally, the model should be tested on a larger scale : real-world experiments, various topologies experiments, *et cetera*.

7 Conclusion

When we want to compare MultiPath TCP to TCP performances, a standard metric used is the Aggregation Benefit defined in [29]. However, this metric supposes that TCP fulfills the available bandwidth, which might not always be the case.

As a consequence, the first step of this master's thesis was to define a new metric such that we directly compare MultiPath TCP to TCP performances on the same paths. We called such metric the *Experimental Aggregation Benefit* as it is based on experimental rather than theoretical results for TCP connections. From its definition, we proposed that the theoretical aggregation benefit metric should be used for evaluating how MultiPath TCP fulfills the available bandwidths.

Once defined, we studied how these aggregations vary as function of paths heterogeneity through large number of experiments. These latter were generated with a space-filling algorithm in order to cover the most diverse possible cases. It includes two types of environments : low-BDP environments where delays are quite small and high-BDP environments which are likely to undergo bufferbloat due to large queueing delays.

The first kind of connections we analyzed was bulk-transfers. In low-BDP environments, the difference between the bandwidth capacity of the path is the main influencing factor. On the other hand, the delay difference has no impact for such connections. By studying the number of reinjections, we attributed this to the *Opportunistic Retransmission and Penalization* that reduces the harming effects of such difference. In high-BDP environments, we showed through a traffic distribution that only one path is used for most of experiments. The congestion control scheme tends to only use the best path in terms of throughput.

Then, we considered the exchanges of small files where we showed the importance of the start of the connection. We explained a trade-off between small files and the congestion created by larger files. For such experiments, the delay difference was the most impacting factor.

To conclude the first part of this work, we evaluated four different congestion control schemes. We described how they behave through different kind of experiments. BaLIA tends to be more responsive to network changes, while LIA struggles to achieve this goal. wVegas under-utilizes the available bandwidth but might be interesting in cases of random losses due to its delay-based principle. Various statistics were also given.

All these experiments were executed three times in order to study the sensitivity. We obtained robust results which implies that our analyzes can be generalized on a larger scale.

In the second part of the work, we did extend *Padhye's et al.* model for MultiPath TCP using LIA as congestion control scheme. We obtained a simple formula for throughput that depends on the loss rates and round-trip times of the paths. We have validated it through some experiments and the results were quite convincing.

7.1 Future Work

We now propose some lines of future work.

Firstly, we did not consider the impact of some factors on the connections. One could study how the aggregations vary with the maximum queueing delay or with the latency on the application for example.

Secondly, all experiments were simulated on a virtualization tool. It could be interesting to see if our results hold in real-life experiments.

Concerning the scheduler, it should be evaluated with rate limited application in order to avoid the self-clocking of the scheduling.

Finally, there is still work to do on the model. We believe it could be extended to other loss-based congestion control schemes, at least BaLIA. Also, the validation should be more general. For example, we could test it on real-life experiments. We could also simulate cases where MultiPath TCP is competing with other TCP flows and see how the model behaves.

Used Tools

A

In order to experiment MultiPath TCP connections and generate results, statistics and graphs, we used several tools we briefly present here.

A.1 Mininet

Mininet [4] is a useful tool to emulate a network topology, running on a single computer. With this tool, a user is able to add hosts, switches and links between them. Those links can be limited given a certain bandwidth, propagation delay, queueing delay or loss rate. This tool offers some limitations, among them, we can note that the loss rate, expressed as a percentage, may only have a integer value.

However, this tool offer a large python library to generate user's specific topology.

A.2 Minitopo

Minitopo [5] is a private tool developed by Benjamin Hesmans in the EPL-INGI-INL department. This python library is used to generate network topology, and to launch experiments on a Mininet environment. Also, it offers the possibility to easily set some useful system control, `sysctl`, values as the congestion control, the path manager or the scheduler used by MultiPath TCP. We used this tool to experiment MultiPath TCP with our experimental design and automate the execution of the experiment in our own developed tool [45].

In order to generate our own topologies and experiments, we extend this tool to be able to test `cURL` or `iPerf` on our topologies. Also, we adapted for example the HTTP server request handler to be able to exit the handler after a given timeout rather than serve for ever.

The only non-default `sysctl` values we modify with this tool are the congestion control scheme and the scheduler used by MultiPath TCP.

A.3 MPTCP Analysis Scripts

MPTCP Analysis Scripts [6] is a tool developed to easily analyze and output results from a MultiPath TCP packet capture. We used this tool to analyze each experiment we generated

with Minitopo and summarized different results, e.g. the different throughput and goodput, the number of MultiPath TCP reinjections and TCP retransmissions, to be able to plot our own graphs.

Graphs The graphs are generated using the pyplot [7] library.

Model : Solving the System

B

The system of equations that we have to solve is :

$$\mathbb{E}[W_i] = r_i \left(\frac{\mathbb{E}[W_i]}{2} + \frac{\mathbb{E}[P_i]}{b} \frac{\mathbb{E}[W_1]/RTT_1^2}{\left(\sum_k \mathbb{E}[W_k]/RTT_k\right)^2} \right) + (1 - r_i) \left(\mathbb{E}[W_i] + \frac{\mathbb{E}[P_i]}{b} \frac{\mathbb{E}[W_1]/RTT_1^2}{\left(\sum_k \mathbb{E}[W_k]/RTT_k\right)^2} \right)$$

for $i = 1, 2$.

Let us start with $i = 1$. Several simplifications yield to the following expression :

$$\frac{r_1}{2} = \frac{\mathbb{E}[P_1]}{b} \frac{1}{RTT_1^2 \left(\sum_k \mathbb{E}[W_k]/RTT_k\right)^2}$$

Using equations 6.1 and 6.2 for P_1 and r_1, one finds:

$$\frac{p_1 \mathbb{E}[W_1]}{2RTT_1} = \frac{\mathbb{E}[W_1]}{bRTT_1} \frac{1}{RTT_1^2 \left(\sum_k \mathbb{E}[W_k]/RTT_k\right)^2}$$

After simplifications, we finally find :

$$\left(\sum_k \mathbb{E}[W_k]/RTT_k \right) = \sqrt{\frac{2}{b \cdot p_1 RTT_1^2}} \tag{B.1}$$

We now consider $i = 2$. Same simplifications yield to :

$$\frac{r_2}{2} \mathbb{E}[W_2] = \frac{\mathbb{E}[P_2]}{b} \frac{\mathbb{E}[W_1]}{RTT_1^2 \left(\sum_k \mathbb{E}[W_k]/RTT_k\right)^2}$$

Using equations 6.1 and 6.2 for P_2 and r_2, one finds :

$$\frac{p_2}{2}\mathbb{E}[W_2]\left(\sum_k \mathbb{E}[W_k]/RTT_k\right)^2 = \frac{\mathbb{E}[W_1}{bRTT_1^2}$$

Using equation B.1, we find :

$$p_1\mathbb{E}[W_1] = p_2\mathbb{E}[W_2] \tag{B.2}$$

Injecting equation B.2 in annexes:eq, we finally have :

$$\mathbb{E}[W_i] = \frac{1}{p_i}\frac{\sqrt{2/(b\cdot p_1)/RTT_1}}{\sum_k 1/(p_k RTT_k)}$$

ApacheBench

C

Hereafter, we present an analysis of the difference between the schedulers while bench-marking request-response time of HTTP1.0 GET request on different file-sizes. This follows our Chapter 4. Since we didn't find clear differences, we present this analysis as an appendix.

The version of `ApacheBench` used in our experiments is the V2.3 included in Apache/2.4.7 for Ubuntu. The transfer protocol is the version 1.0 of HTTP which means that there is a new-one connection per request. As our goal is to benchmark sequential requests, we set the concurrency parameter to one. Also, we decide to limit the time of bench-marking to 30 seconds and we don't limit the maximum number of requests[1]. We request file of sizes from 8 kB to 2000 kB.

We analyze now the difference between the average request-response time using the two schedulers defined earlier. The measure used in the graphs is a ratio defined as :

$$R_N = \frac{N_{LowRTT} - N_{Round-Robin}}{N_{LowRTT} + N_{Round-Robin}} \frac{[\#pkts/s]}{[\#pkts/s]}$$

with N_X, the average number of request per time unit using the MultiPath TCP scheduler X.

R_N has no unit is bounded by -1 and 1 and means :

- $R_N > 0$: The performance of MultiPath TCP is higher with the LowRTT scheduler. In a same elapsed-time, MultiPath TCP with LowRTT is able to send and process more requests.
- $R_N = 0$: There are no difference in term of performance between the different schedulers.
- $R_N < 0$: The performance of MultiPath TCP is higher with the Round-Robin scheduler.

First, we analyze the results obtained with low-BDP environments, in the Figure C.1. Here, we divide our experiments by the absolute difference between the delays $|\delta_{delay}|$. We observe that the median values of the ratio are near from 0, thus the difference between the two schedulers, in term of performance, is small. However, has the file size increase, we can see the standard deviation increase for small delay differences and there are more outliers.

We analyzed the outliers, with $|R_N| > 0.1$, and we observe the different experiments that lead to those results :

[1] Despite the `ApacheBench` default limit of 50000 requests

		First Path			Second Path			
Nr.	File-size	Capacity [Mbps]	Delay [ms]	Queue-size [#pkts]	Capacity [Mbps]	Delay [ms]	Queue-size [#pkts]	R_N
1	128kB	2	1	10	30	6	35	-0.26
2	128kB	6	1	45	1	11	10	-0.30
3	2MB	96	18	136	98	14	65	-0.10

Table C.1: Ratio of the average number of request per time unit in low-BDP environment - Detailed topology of the outliers

For the two first outliers, the Round-Robin scheduler tends to have better results when the delay on the first path is so small that the LowRTT overload this path and under-load the second. For the third outlier, the two path have a large capacity and a small delay difference, the round-robin scheduler tends also to take advantage of this situation.

Strangely, we only observe those tendencies for one file size on each of those three environments. This suggest that there is an other factor which slightly influences the performance of the schedulers in low-BDP environments.

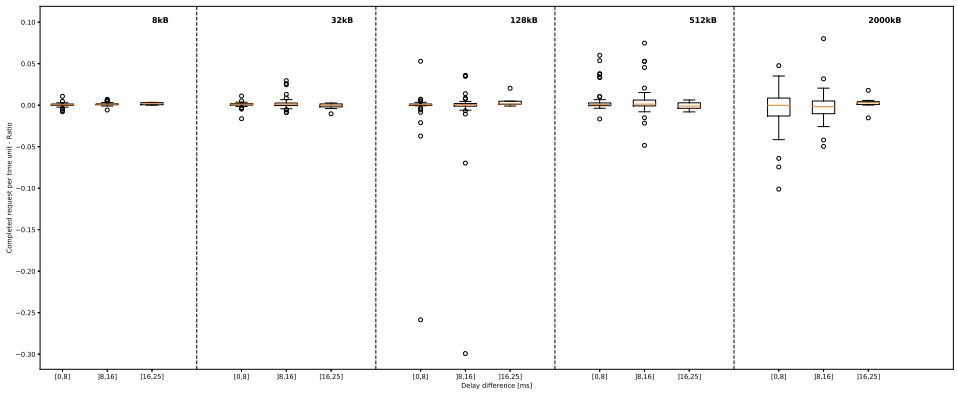


Figure C.1: Ratio of the average number of request per time unit in low-BDP environment

In high-BDP environments, we observe the median are, once again, near `0` and the standard deviation increase as the file size increase. Furthermore, the `LowRTT` scheduler tends to take advantage for some environments with large delay difference.

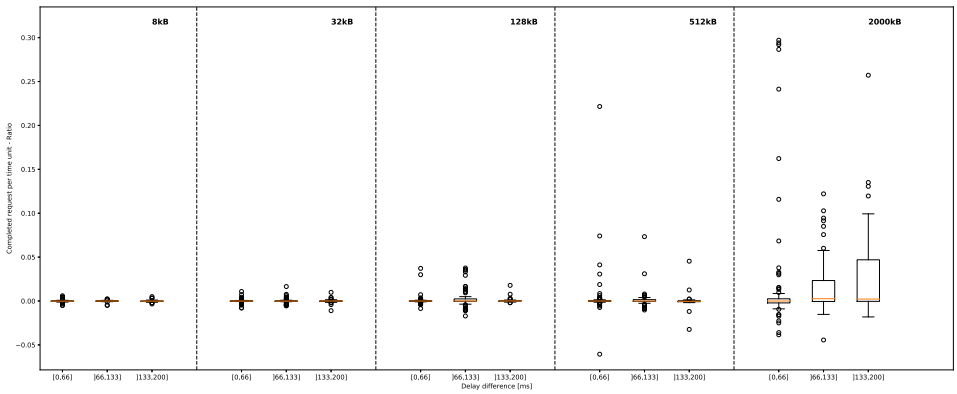


Figure C.2: Ratio of the average number of request per time unit in high-BDP environment

Conclusion

However, those results don't mark a clear difference between the two schedulers, this confirmed our previous results and led us to the conclusion the the schedulers do not have an impact in this kind of transfer.

List of Figures

List of Tables

Bibliography

[1] ab - apache http server benchmarking tool. https://httpd.apache.org/docs/2.4/programs/ab.html. Accessed: 2017-05-26. (Referenced in page 43.)

[2] curl, command line tool and library for transferring data with urls. https://curl.haxx.se/. Accessed: 2017-04-09. (Referenced in pages 29 and 36.)

[3] iperf, the ultimate speed test tool for tcp, udp and sctp. https://iperf.fr/. Accessed: 2017-04-09. (Referenced in pages 23 and 46.)

[4] Mininet, an instant virtual network on your laptop (or other pc). http://mininet.org/. Accessed: 2017-04-09. (Referenced in pages 17, 35 and 71.)

[5] Minitopo. https://bitbucket.org/bhesmans/minitopo. Accessed: 2016. (Referenced in pages 17 and 71.)

[6] Multipath tcp analysis scripts. https://github.com/multipath-tcp/mptcp-analysis-scripts. Accessed: 2016. (Referenced in page 71.)

[7] Python pyplot library. http://matplotlib.org/api/pyplot_api.html. Accessed: 2017-06-01. (Referenced in page 72.)

[8] Nornet edge, 2012. (Referenced in page 10.)

[9] Oivind Andersson. *Experiment!: planning, implementing and interpreting*. John Wiley & Sons, 2012. (Referenced in page 17.)

[10] Behnaz Arzani, Alexander Gurney, Sitian Cheng, Roch Guerin, and Boon Thau Loo. Deconstructing mptcp performance. In *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, pages 269–274. IEEE, 2014. (Referenced in pages 26 and 33.)

[11] Sébastien Barré, Christoph Paasch, and Olivier Bonaventure. Multipath tcp: from theory to practice. In *International Conference on Research in Networking*, pages 444–457. Springer, 2011. (Referenced in pages 9 and 10.)

[12] A Beal, M Claeys-Bruno, and M Sergent. Constructing space-filling designs using an adaptive wsp algorithm for spaces with constraints. *Chemometrics and Intelligent Laboratory Systems*, 133:84–91, 2014. (Referenced in pages 15, 18 and 35.)

[13] Luca Boccassi, Marwan M Fayed, and Mahesh K Marina. Binder: a system to aggregate multiple internet gateways in community networks. In *Proceedings of the 2013 ACM MobiCom workshop on Lowest cost denominator networking for universal access*, pages 3–8. ACM, 2013. (Referenced in page 6.)

[14] Olivier Bonaventure, Mark Handley, and Costin Raiciu. An overview of multipath tcp. *USENIX login;*, October 2012. (Referenced in pages 3 and 5.)

[15] S. Barre C. Paasch. Multipath tcp in the linux kernel, 2017. (Referenced in pages 4, 5, 6 and 9.)

[16] Yu Cao, Mingwei Xu, and Xiaoming Fu. Delay-based congestion control for multipath tcp. In *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, pages 1–10. IEEE, 2012. (Referenced in pages 7, 9 and 45.)

[17] Neal Cardwell, Stefan Savage, and Thomas Anderson. Modeling tcp latency. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1742–1751. IEEE, 2000. (Referenced in page 12.)

[18] Yung-Chih Chen, Yeon-sup Lim, Richard J Gibbens, Erich M Nahum, Ramin Khalili, and Don Towsley. A measurement-based study of multipath tcp performance over wireless networks. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 455–468. ACM, 2013. (Referenced in pages 1, 35 and 41.)

[19] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. A first analysis of multipath tcp on smartphones. In *17th International Passive and Active Measurements Conference*, volume 17. Springer, March-April 2016. (Referenced in pages 10 and 12.)

[20] Q. De Coninck and M. Baerts. Multipath tcp with real smartphone applications. 2015. (Referenced in pages 5 and 79.)

[21] Shuo Deng, Ravi Netravali, Anirudh Sivaraman, and Hari Balakrishnan. WiFi, LTE, or Both? Measuring Multi-Homed Wireless Internet Performance. In *Internet Measurement Conference 2014*, Vancouver, Canada, November 2014. (Referenced in pages 10 and 12.)

[22] K. R. Fall and W. R. Stevens. *TCP/IP Illustrated Volume 1: The Protocols*. 2011. (Referenced in page 3.)

[23] S. Ferlin, T. Dreibholz, and Ö. Alay. Multi-path transport over heterogeneous wireless networks: Does it really pay off? In *2014 IEEE Global Communications Conference*, pages 4807–4813, Dec 2014. (Referenced in page 10.)

[24] Ronald Aylmer Fisher et al. The design of experiments. *The design of experiments.*, (Ed. 5), 1949. (Referenced in page 10.)

[25] Jim Gettys and Kathleen Nichols. Bufferbloat: Dark buffers in the internet. *Queue*, 9(11):40, 2011. (Referenced in pages 19, 20 and 26.)

[26] Mark Handley, Olivier Bonaventure, Costin Raiciu, and Alan Ford. Tcp extensions for multipath operation with multiple addresses. 2013. (Referenced in pages 3 and 4.)

[27] Mark Handley, Costin Raiciu, and Damon Wischik. Coupled congestion control for multipath transport protocols. 2011. (Referenced in page 6.)

[28] Raj Jain, Dah-Ming Chiu, and William R Hawe. *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*, volume 38. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984. (Referenced in page 53.)

[29] Dominik Kaspar. Multipath aggregation of heterogeneous access networks. *SIGMultimedia Rec.*, 4(1):27–28, March 2012. (Referenced in pages 11 and 69.)

[30] Ramin Khalili, Nicolas Gast, Miroslav Popovic, and Jean-Yves Le Boudec. Mptcp is not pareto-optimal: performance issues and a possible solution. *IEEE/ACM Transactions on Networking*, 21(5):1651–1665, 2013. (Referenced in pages 1, 6, 7, 8, 14, 15, 45, 59, 63, 64 and 67.)

[31] Jack P. C. Kleijnen, Susan M. Sanchez, and Thomas W. Lucas. State-of-the-art review: A users guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17:263–289, 2005. (Referenced in page 10.)

[32] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, 1997. (Referenced in pages 1 and 12.)

[33] Jeonghoon Mo, Richard J La, Venkat Anantharam, and Jean Walrand. Analysis and comparison of tcp reno and vegas. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1556–1563. IEEE, 1999. (Referenced in page 12.)

[34] Teunis Ott, J Kemperman, and Matt Mathis. The stationary behavior of ideal tcp congestion avoidance. 1996. (Referenced in page 12.)

[35] Christoph Paasch. *Improving Multipath TCP*. PhD thesis, UCLouvain / ICTEAM / EPL, November 2014. (Referenced in pages 3, 4 and 79.)

[36] Christoph Paasch, Gregory Detal, Fabien Duchene, Costin Raiciu, and Olivier Bonaventure. Exploring mobile/wifi handover with multipath tcp. In *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, pages 31–36. ACM, 2012. (Referenced in page 5.)

[37] Christoph Paasch, Simone Ferlin, Ozgu Alay, and Olivier Bonaventure. Experimental evaluation of multipath tcp schedulers. In *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, pages 27–32. ACM, 2014. (Referenced in pages 4, 10, 12, 17, 23, 29, 37, 44 and 60.)

[38] Christoph Paasch, Ramin Khalili, and Olivier 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. (Referenced in pages 10 and 12.)

[39] Christoph Paasch, Ramin Khalili, and Olivier Bonaventure. On the benefits of applying experimental design to improve multipath tcp. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 393–398. ACM, 2013. (Referenced in pages 17, 18, 19 and 31.)

[40] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling tcp throughput: A simple model and its empirical validation. *ACM SIGCOMM Computer Communication Review*, 28(4):303–314, 1998. (Referenced in pages 1, 12, 13, 57 and 58.)

[41] Qiuyu Peng, Anwar Walid, Jaehyun Hwang, and Steven H Low. Multipath tcp: Analysis, design, and implementation. *IEEE/ACM Transactions on Networking*, 24(1):596–609, 2016. (Referenced in pages 1, 6, 14, 45, 54 and 56.)

[42] Costin Raiciu, Dragos Niculescu, Marcelo Bagnulo, and Mark James Handley. Opportunistic mobility with multipath tcp. In *Proceedings of the Sixth International Workshop on MobiArch*, MobiArch '11, pages 7–12, New York, NY, USA, 2011. ACM. (Referenced in page 10.)

[43] Costin Raiciu, Christoph Paasch, Sébastien Barré, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be? designing and implementing a deployable multipath tcp. In *USENIX Symposium of Networked Systems Design and Implementation (NSDI'12), San Jose (CA)*, 2012. (Referenced in pages 3 and 28.)

[44] Costin Raiciu, Damon Wischik, and Mark Handley. Practical congestion control for multipath transport protocols. 2009. (Referenced in pages 57 and 60.)

[45] Chauvenne Rémi and Thibault Libioulle. Multipath tcp measurements scripts. https://github.com/tlibioulle/mptcp-measurement-scripts. Accessed: 2017-06-09. (Referenced in page 71.)

[46] Chauvenne Rémi and Thibault Libioulle. Multipath tcp throughput calculator. https://github.com/remichauvenne/mptcp-calculator. Accessed: 2017-05-28. (Referenced in page 65.)

[47] Felipe A. C. Viana. Things you wanted to know about the latin hypercube design and were afraid to ask. 2013. (Referenced in page 15.)

[48] Anwar Walid, Qiuyu Peng, Jaehyun Hwang, and S Low. Balanced linked adaptation congestion control algorithm for mptcp. *IETF, Individual Submission, Internet Draft draft-walid-MPTCP-congestion-control-03*, 2015. (Referenced in pages 6 and 8.)

[49] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. 2011. (Referenced in pages 1, 6, 7, 14, 30, 45, 52, 55, 58, 63 and 64.)

[50] Bo Zhang, TS Ng, Animesh Nandi, Rudolf Riedi, Peter Druschel, and Guohui Wang. Measurement based analysis, modeling, and synthesis of the internet delay space. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 85–98. ACM, 2006. (Referenced in page 18.)

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve www.uclouvain.be/epl