Evaluation of MPTCP in Satellite Networks

Bachelor Thesis

Mattia Pedrazzi
"Bachelor of Science in Computer Science"

Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

11, February 2020

Prof. Dr. Torsten Braun
Communication and Distributed Systems Group

Institut für Informatik
University of Bern, Switzerland

Abstract

TCP/IP over satellite networks is an area of increasing interest as communication and aerospace companies such as Iridium, SpaceX and OneWeb are deploying and operating increasingly large constellations of communication satellites.

Satellite networks experience large round trip times and frequent signal interruptions. This can slow down or even prevent the opening of the TCP congestion window. A small TCP congestion window leads to a smaller overall throughput.

Multipath TCP (MPTCP) might be one solution to mitigate the effects of these problems by exploiting the path diversity offered by the network itself.

The goal of this thesis is to evaluate if the drawbacks of TCP connections over satellite networks can be mitigated using MPTCP. To achieve this, we simulate both TCP and MPTCP by using the NS3 network simulator in a GEO satellite network scenario.

First, we simulate MPTCP and TCP separately. We evaluate the single flow performance of MPTCP and the single connection performance of TCP as well as their performance with multiple MPTCP subflows and multiple TCP connections in relation to their congestion windows.

Then, we simulate the two protocols in a mixed scenario with both protocols coexisting in the same network to measure their throughput. We simulate MPTCP and TCP using two different data stream types and three different realistic error models.

Our results show that in general multiple separate TCP connections perform better than MPTCP in most circumstances, but when most paths are congested, MPTCP can achieve a significantly better throughput than TCP for a limited period of time.

Contents

1 Introduction

Satellite networks are subject to different conditions than land connections. Shadowing [1], atmospheric effects [2,3], the exposure to the space environment [4] and the large round trip times caused by the large distances [6, 7] degrade the quality of the communication. This causes packets to get dropped and therefore, lowering the achievable throughput. A solution to alleviate these problems, might be to use multiple available paths to send data to the same host [1]. In the context of TCP, a Multipath TCP version has been implemented, namely: MPTCP [8]. Multipath TCP (MPTCP) establishes multiple subflows to communicate between two hosts. MPTCP subflows behave similarly to standard TCP connections. These subflows can then, be mapped to different paths by using an adequate routing strategy. The data stream can subsequently be distributed by using a scheduler. The scheduler can choose to which subflow the data should be forwarded and therefore, also act as a load balancing mechanism to relieve the congested paths and send more data to the less congested paths. Even with a Multipath approach, the single Multipath TCP subflows are still subject to frequent interruptions in a satellite communication scenario, because the aforementioned signal degradation affects every satellite link separately. In satellite scenarios, the round trip time is often very high, e.g., about 600 seconds for geostationary satellites [6, 7]. In such scenarios, retransmissions cause the packet to be correctly received well over a second after its initial sending, reducing the throughput significantly. A proposed method to reduce retransmissions caused by packets being dropped is Network Coding [9]. Network Coding can be used to reduce the frequency of retransmissions when packets are not being received. Network Coding encodes the data stream by adding redundancy and decodes the packet stream at the receiving end by reconstructing the lost packets, thus, avoiding some of the needed retransmissions. There already are scientific works that evaluate the following concepts over satellite networks; different TCP versions [7], Network Coding [9] and MPTCP combined with Network Coding [1], but none that make use of end-to-end TCP and MPTCP connections over geostationary satellite networks.

In this work, we simulate Multipath TCP and standard TCP over satellite networks to compare their performance. We simulate a satellite network topology and we implement three realistic error models to simulate the conditions to which satellite links are subjected. The first error model we implement introduces a constant bit error ratio to the satellite links, the second error model produces ON and OFF error patterns to simulate the effects of shadowing [1] and the third error model is a combination of the former two error models. We simulate the two protocols by using two types of data streams, namely: a constant and continuous data stream and a data stream comprised of On/Off cycles to simulate traffic generated by video streaming [11]. At first, we simulate a single flow MPTCP connection and a single TCP connection to identify the different behaviors of the two protocols. If the two protocols behave in a different way with single flows or connections, they should also behave differently with multiple flows and multiple connections. The behavior of MPTCP with a single flow and of a single TCP connection should also help to explain the performance difference between MPTCP and TCP with a higher number of subflows and connections. Additionally, we illustrate the size of the congestion window of the simulated MPTCP and TCP connections as a function of the simulation time, since the performance of MPTCP and TCP at a certain moment in time is strictly dependent on the size of the congestion window at that moment. We obtain a second batch of results from simulations in which multiple MPTCP subflows and multiple TCP connections are simulated separately on the satellite network. We illustrate the network throughput and the congestion window of the aforementioned setups and analyze the relationship between the throughput and the congestion window. We then simulate mixed scenarios in which MPTCP and TCP connections coexist inside of the satellite network. We illustrate the throughput results and analyze them in relation to the congestion window.

The structure of this work is as follows: We describe the common problems that occur in satellite links that lead to signal degradation and in turn, worsen TCP operations in section 2. We illustrate some proposed solutions to solve the common problems in section 3. We describe how the MPTCP protocol is initialized and how it operates according to the IETF specifications [8] in section 4. We, then, illustrate the simulation scenario setup implemented in the NS3 simulator in section 5. The key components of the simulation setup are discussed in the following subsections: the network topology (subsection 5.1), the error models (subsection 5.2), which are based on the common problems that affect satellite links discussed in section 2, the data stream types (subsection 5.3), which are based on realistic applications, and the terminal protocol setups we use in our simulations in subsections 5.4, 5.5 and 5.6. The results of the NS3 simulations are then plotted and commented in section 6. In section 7 we conclude this work.

2 Problems in Satellite Networks

Satellite channels are subjected to substantially different conditions compared to cabled and terrestrial based broadband wireless channels. We have to consider multiple factors that can contribute to the degradation of the overall throughput. We list some of them in the following subsections.

2.1 Large Propagation Delay

Geostationary satellites are located at around 35,786 km above the earth's equator [10]. The direct consequence of the latter is a very high round trip time (RTT) of 540ms - 600ms [6, 7]. TCP opens its congestion window every time an acknowledgement packet is received. The time it takes for a packet to be acknowledged is roughly one RTT. Therefore, the larger the RTT, the longer it takes for packets to be acknowledged, thus, slowing the opening of the congestion window. The throughput of TCP is strictly dependent on the size of the congestion window, therefore a bigger RTT affects greatly the throughput of TCP. This affects both the Slow Start phase and the Congestion Avoidance phase [7].

2.2 High Bit Error Ratios

Atmospheric effects as well as weather conditions between the ground station and the satellite contribute to signal degradation causing jitter of the received signal power intensities, which result in bit errors [2, 3].
Another cause of bit errors can be found in the space environment itself since free particles can interfere with the on-board receiver of the satellite [4].
Bit errors cause packets to get dropped by the lower network layers and TCP interprets packet drops as a sign of congestion. This false assumption might lead to a premature Slow Start phase stop and to further congestion window reductions during the Congestion Avoidance phase.
The recovery from such an event is also not adequate. When TCP is subject to frequent packet losses it can either recover one packet per RTT or recover

multiple packets at a faster rate when a timeout occurs [5].

2.3 Shadowing

When working with mobile nodes that connect directly to the satellites, objects can get between the satellites and the dishes, obstructing the line-of-sight. This can happen in urban environments or anywhere where the line of sight can be obstructed, especially at high latitudes. This phenomenon is called Shadowing [1]. Shadowing produces longer signal drops, which severely impacts TCP performance [6]. Longer signal drops result in ON/OFF patterns that can be modeled into an error model [1].

2.4 Competition with Other Connections

When working with multiple independent TCP connections, the sessions with smaller RTTs and less losses will receive a bigger share of available resources, because those connections will open their windows faster and thus, compete with the slower ones [6].

2.5 Low On-Board Processing Power

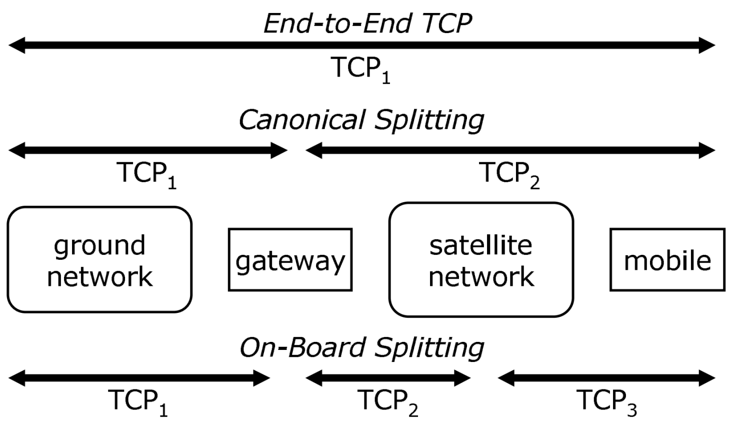
Intra-session network coding, discussed in section 3.2, or error correcting code in general are difficult to implement on a satellite because of processing power limitations of todays systems [9].

3 Related Works

3.1 Performance Enhancement Proxy

To mitigate the drawbacks caused by a large Round Trip Time of a satellite link, it is possible to split the TCP connection on board the satellite into two separate TCP connections [6].

Figure 1: On-Board split [6].



To achieve the split, a forwarding proxy agent is needed on the satellite, illustrated in Figure 1. The proxy agent is responsible for the data forwarding between the two TCP connections. One advantage is that each TCP connection is subject to half the Round Trip Time (RTT) compared to a non split connection that would extend from one ground station, over the satellite network, to another ground station. The smaller RTT reduces the speed of recovery in case of packet losses and TCP timeouts. An end-to-end TCP connection would require both the links not being shadowed (section 2.3) for a successful transmission, meanwhile with a split connection, one of the two earth-to-space connections can be under the effect of shadowing for a short period of time. The disadvantages are: additional on-board storage for the forwarding agent and the implementation of a forwarding agent. This approach offers a significant throughput increase, more than 200% for TCP Westwood and more than 100% increase for TCP New Reno [6].

3.2 Network Coding

Network Coding [9] enables the encoding of the packet stream by adding redundant packets to the stream, so that the stream can be decoded from the receiving host. Often a successful decoding and reconstruction of the packet stream does not require all encoded packets. This way it is possible to lose packets on the way but reconstruct the packet stream at the receiving end, thus improving throughput. Network Coding can be classified into two main categories: Intra-session coding [12] and Inter-session coding [12]. Intra-session coding is achieved by encoding the packets belonging to the same session, while Inter-session coding allows encoding among packets that belong to different sessions. Unfortunately, we are limited to intra-session network coding because satellite networks make use of a bent-pipe architecture [9] and the satellites have limited on-board processing power, as mentioned in section 2.5.
The encoding strategy is also to be considered. Random Linear Network Coding [9] using generations or a sliding window approach [9] has shown throughput improvement and shorter in-order delivery delays over the selective-repeat ARQ protocol [9]. The efficiency of such approach is strictly dependent on the amount of redundancy packets attached to each stream, because if too many packets are dropped, especially for smaller generations, the lost packets cannot be reconstructed later in the decoding phase. In fact, Network Coding cannot mitigate the effects of deep long fades. Therefore, in the case of satellite connections, Network Coding should be used to recover from isolated bit errors rather than the ON/OFF patterns described in section 5.2.2 [9].

3.3 General TCP Alternatives

Modifications of plain TCP might also represent a viable solution to improve throughput. This session discusses the TCP alternative implementations which are not specific to satellite connections.

3.3.1 TCP SACK

A big issue, working with plain TCP, are cumulative acknowledgements.
The cumulative acknowledgement strategy has the receiver waiting one RTT to learn about lost packets and the sender to retransmit correctly received packets when a timeout occurs [7].
Selective acknowledgements (SACK) [7] allow the receiver to inform the sender only about the correctly received packets, so that the sender can retransmit only the lost ones, thus, resulting in less ACK packets being sent. Additionally, the sender can recover multiple lost segments in a single round trip time.
TCP SACK can improve the throughput of TCP, when deployed over satellite networks. Nonetheless, at high packet error ratios, between 10^{-1} and 10^{-3}, TCP SACK performs worse than dedicated TCP satellite alternatives and TCP Westwood [7].

3.3.2 TCP Westwood

TCP Westwood [7] is a sender side modification of the congestion control algorithm. Westwood operates by using an additional variable, namely the Bandwidth Share Estimate (BSE), which estimates the total bandwidth used by the connection. The BSE is estimated by averaging the inter-arrival intervals of ACK packets. With the use of this estimate, Westwood reacts to triple duplicate ACKS and timeouts by dynamically adjusting the Slow Start threshold. TCP Westwood performs better than TCP SACK, but worse than the dedicated TCP alternatives for satellite networks [7](section 3.4). Moreover, TCP Westwood performs poorly when the packet loss rate is higher than a few percent [13].

3.4 Dedicated TCP Satellite Alternatives

Some dedicated TCP implementations for satellite connections have been tested and have shown advantages over plain TCP. This session discusses alternative TCP implementations specifically used to improve throughput in satellite connections.

3.4.1 Satellite Communication Transmission Protocol (SCPS-TP)

SCPS-TP [7] is an enhancement of plain TCP, which consists in implementation and specification changes to standard TCP to support satellite communications. Important to notice is that these enhancements affect the interoperability with standard TCP. An “SCPS-TP capable” option is added to the TCP header, and if the receiving host does not return this option, fallback to regular TCP is possible. SCPS-TP differs from the aforementioned TCP variables because it is able to distinguish the cause of a loss and recover properly without unnecessary congestion control feedbacks using the Selective Negative Acknowledgement option (SNACK) [7]. The SNACK option identifies and analyses the multiple holes inside of the receiving buffer and hastens recovery by enlarging the receiving window allowing the packets in flight to be received while learning about the lost packets. This mechanism allows for a continuous data transmission even if packets are continuously lost. This is paramount to make up for the false assumption plain TCP makes (section 2.2), because the cause of most packet losses is, in our case, bit errors rather than actual congestion [14].

3.4.2 Satellite Transport Protocol (STP)

STP [7] implements a system of POLL and STAT(us) notification packets [15]. The sender stores the sent packets for later potential retransmission, POLL packets are periodically sent from the sender to ask the receiver which packets have been received. The receiver answers with a STAT packet, so that the sender can explicitly be informed about the loss. The congestion of the backward channel depends only on the frequency of the POLL and STAT packets. In addition to this mechanism, Selective Negative Acknowledgements [7] can be used to inform the sender about a packet loss in half a RTT. The main disadvantage of STP is that the enhancement needed have to be implemented on both hosts involved in the communication for STP to operate correctly [15].

3.4.3 TCP Jersey

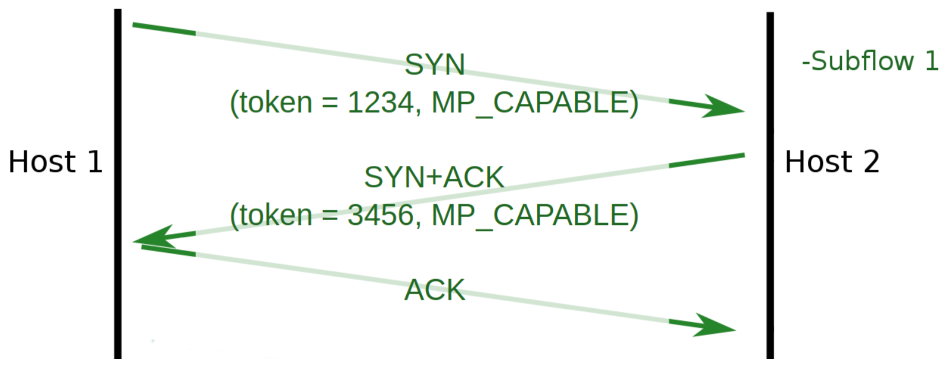
TCP Jersey [7, 16] discriminates the cause of a packet loss with help of two fundamental enhancements/implementations. Jersey uses an Available Bandwidth Estimation (ABE) [7] algorithm and an explicit congestion notification mechanism (Congestion Warnings) [7]. The bandwidth estimation algorithm monitors the ACK rate to compute an optimal congestion window and the receiver can be informed on how to regulate the data rate under congested conditions. The Congestion Warnings rely on ECN-like network router configurations that allow the routers to warn the sender about a congestion by marking the packets. The main disadvantage of TCP Jersey is that additional router configurations are needed for the protocol to function properly [7].

4 MPTCP Protocol Description

MPTCP [8] is a further enhancement of plain TCP. MPTCP distributes the data stream between multiple subflows harvesting the path diversity offered by the network [17].

4.1 Session Initiation

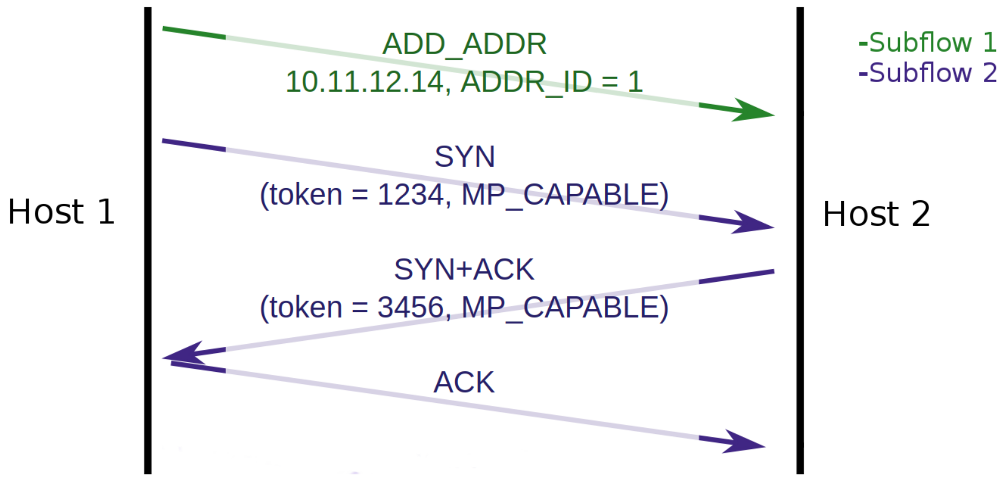
Figure 2: MPTCP Session Initiation [18].



As illustrated in Figure 2, an MPTCP connection [8] is initiated with a simple three-way SYN, SYN/ACK, ACK handshake. Each packet contains the MP_CAPABLE [8] TCP option, which itself contains a 64-bit key used to later authenticate the added subflows. The subflows are not directly identified by the use of the key, rather by using a 32-bit hash of the key. The key must be unique for any sender host at any point in time, thus, a mapping of the token to the key is required. It might happen that two different keys belonging to a single host hash the same token. Therefore, an MPTCP implementation should check for duplicates inside of the key-token mapping. Nonetheless, in the unlikely case of a hash collision, MPTCP should always be able to fallback to standard TCP operation. After the successful exchange of a MP_CAPABLE option the first subflow is established [8].

4.2 Subflow Initiation

Figure 3: MPTCP Subflow Initiation [18].



As shown in Figure 3, hosts learn about the existence of additional available addresses that can be used to establish new subflows, using the ADD_ADDR option [8]. Both hosts can initiate the establishment of a new subflow over an unused pair of addresses. New subflow initiations are possible only when a DSS option [8], discussed in section 4.3.1, is received and a MP_CAPABLE option has been exchanged successfully. As illustrated in Figure 3, a normal TCP SYN/ACK exchange is executed with a TCP MP_JOIN or MP_CAPABLE option on the packet, which have a different format for each exchange of three-way handshake. The 32-bit token, generated during the MP_CAPABLE exchange at the beginning of the session, is used to identify the correct MPTCP session [8].

4.3 Operation

4.3.1 Data Sequence Signal (DSS) Option

Figure 4: Data Sequence Signal (DSS) Option [8].

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------+----------------------+
|     Kind      |    Length     |Subtype| (reserved) |F|m|M|a|A|
+---------------+---------------+-------+----------------------+
|           Data ACK (4 or 8 octets, depending on flags)       |
+--------------------------------------------------------------+
|   Data sequence number (4 or 8 octets, depending on flags)   |
+--------------------------------------------------------------+
|              Subflow Sequence Number (4 octets)              |
+-------------------------------+------------------------------+
|  Data-Level Length (2 octets) |      Checksum (2 octets)     |
+-------------------------------+------------------------------+
```

The DSS option [8] is an MPTCP option which is added to the MPTCP packet. The DSS option carries the necessary signaling information which is used to reliably deliver and reconstruct the data at the receiving end. The structure of the DSS option is illustrated in Figure 4. The “Kind” field specifies which TCP option is being used. The “Subtype” field specifies the TCP option subtype. The MPTCP flags, in the top right corner of Figure 4, specify the following properties: “A” indicates that the Data ACK is present, “a” indicates that the Data ACK is 8 octets long, otherwise it is only 4 octets long, “M” indicates that the Data Sequence Number (DSN), Subflow Sequence Number (SSN), Data-Level Length and the Checksum are present, “m” indicates that the Data sequence number is 8 octets long, otherwise it is 4 octets long, and the “F” flag indicates that the “DATA_FIN” is present. The Data ACK is discussed is section 4.3.2. The Data sequence number and the Subflow sequence number are used in combination to reconstruct the data stream at the receiving host.

At a higher level of abstraction, the MPTCP data transfer operation consists of receiving a data stream from the application to, then, split it into the multiple established subflows. To achieve such operation, MPTCP needs to make use of two additional signaling variables: the Data Sequence Mapping [8] and the

Data ACK [8]. Both options can be singularly or simultaneously present inside the DSS option [8]. The Data Sequence Mapping (DSM) is used to reassemble the data stream coming from the subflows at the connection level. The DSM is used by the sender to ensure in-order delivery. It consists of a subflow sequence number and a byte length which validates the length of the DSM. A sender must not change the DSM after its declaration [8]. The Data Acknowledgement is discussed in section 4.3.2.

4.3.2 Data Acknowledgements

MPTCP has a cumulative connection-level acknowledgement, the "Data ACK" [8], which is a field of the DSS option. The behavior is similar to the cumulative ACK of standard TCP and it indicates how much data has been received. The Data ACK specifies the next Data Sequence Number (DSN) [8] that should be received. Like in standard TCP, the Data ACK is used to make sure that all the data and the necessary signaling have been correctly received. Since the receiving window is shared by all subflows that belong to the connection, the Data ACK points to the left edge of the advertised receive window to indicate how many packets have been correctly received. The subflows should independently acknowledge the data they have been assigned, like in a plain TCP connection.
Important to notice is that Selective Acknowledgements (SACK) [8] should preferably be used at the subflow level to improve performance.

4.3.3 Congestion Control

The behavior adopted by MPTCP subflows resembles TCP, because different subflows have different congestion windows. In spite of this, the subflow congestion windows have to be coupled in some way to ensure fairness at bottlenecks with single-path TCP and resource pooling [8, 19]. Moreover, MPTCP must not take more bandwidth on one path than a single-path TCP to be deployable in the current Internet [8]. For this reason, the proposed congestion control algorithms [19–21] address congestion control at the subflow

level and not at the connection level.
An MPTCP congestion control algorithm should achieve the following three goals [19]:

- A subflow should perform the same or better than a standard single flow TCP connection on the best available path.
- A subflow should not take away network resources from other subflows.
- If a subflow is congested, MPTCP should move as much traffic as possible off the congested path to the less congested paths.

There are different congestion control algorithm proposals for MPTCP subflows [19–21], but we are focusing on the Linked Increase algorithm because it is the one we have readily available in the NS3 MPTCP implementation we are using [22].

4.3.3.1 Linked Increases Algorithm In the Congestion Avoidance phase the congestion window increase after every subflow ACK is given, as per [19], by :

$$\Delta Cwnd = \frac{(\alpha \cdot A \cdot mss_i)}{totCwnd}$$

where A is the total amount of data correctly acked, mss_i is the Maximum Segment Size for the i subflow, $totCwnd$ is the sum of all the subflow congestion windows and α is given, as per [19], by:

$$\alpha = totCwnd \cdot \frac{max_i \frac{cwnd_i \cdot mss_i^2}{rtt_i^2}}{\left(\sum_j \frac{cwnd_j \cdot mss_j}{rtt_j}\right)^2}$$

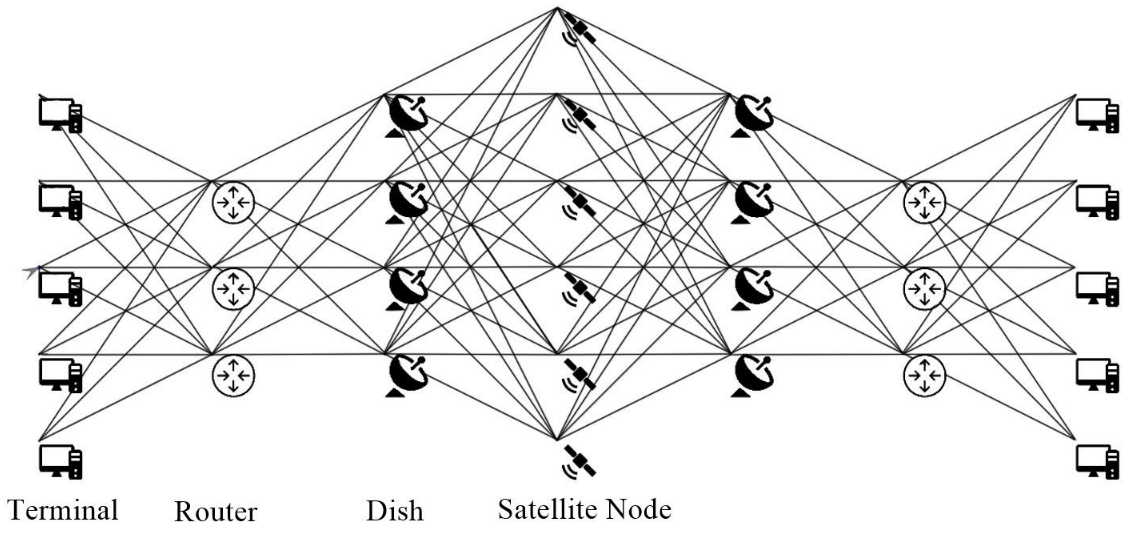
where rtt_i is the round trip time for the subflow i, the function $max_i\,(X_i)$ selects the biggest X_i among all the subflows and $cwnd_i$ is the current congestion window of the subflow i.

5 Evaluation of MPTCP in satellite scenarios

The goal of the simulations presented in this section is to evaluate if MPTCP improves the throughput compared to standard TCP in a satellite communication scenario. With the help of three different error models, discussed in section 5.2, first TCP and MPTCP are simulated separately and, then, in a mixed scenario as well. The throughput differences, the corresponding congestion windows and the main causes for the discrepancies are also discussed.

5.1 Simulation Scenario

Figure 5: Simulation Scenario.



The simulation is implemented in NS3 [23], which has some limitations that will be discussed later. Nonetheless the scenario tries to take into account most of the characteristics of a network topology which comprises a number of geostationary satellite nodes. Figure 5 presents the simulation topology we use. There are four types of nodes in our simulation topology, namely: the terminal, the router, the dish and the satellite nodes. The terminals on one end (Figure 5) try to establish a TCP or an MPTCP session, each with one of the other terminals on the other end of the network. In the case of MPTCP, an application is installed on every sending terminal and a packet sink is installed on the terminals on the receiving end. The application installation in the case of TCP is explained in section 5.4. There are five terminals on the left end (Figure 5) of the network, which, after the establishment of the connection, try to send data streams to the other five terminals on the receiving end of the network. The routers are responsible for the routing between the dishes and the terminals. The dishes connect the terrestrial nodes to the satellite nodes and they are the only nodes that connect directly to the satellites. Each node in the simulation has packet forwarding capabilities, an Internet stack and its own IP address.

A geostationary satellite is stationary relative to the earth's surface, hence we do not implement any mobility model for the satellite nodes. On the other hand the dish nodes may move in some use cases, so a mobility model should be implemented for them. But for our simulation we avoid using any mobility model and we implement an adequate error model.

What is paramount to consider are the characteristics of the satellite connections and of the terrestrial connections that lead up to the dishes. Every connection is implemented with an instance of the ns3::PointToPointNetDevice [24] for sake of simplicity and it eliminates the possibility to apply mobility to the nodes. The connections between the terminals and the routers and those between the routers and the dishes are simple land connections, i.e. with a 10ms delay and with a data rate of 500Mbps. On the other hand, the connections between the dishes and the satellites try to emulate the real conditions of connections with geostationary satellites, namely with a 275ms delay and a 20Mbps [1, 4] data rate for both the uplink and the downlink. Those parameters add up to a simulated round trip time of approximately 600ms at connection level, which is consistent with the literature [6].

The simulations are to be run with four different execution seeds and from a duration of 30 seconds, up to 480 seconds for each set of parameters to get a wider variety of results for the same network conditions. The results are then averaged and plotted.

5.2 Error Models

Error models are used to simulate packet and bit losses that might occur in the up- and downlink of a satellite connection. The physical reasons for losses over such connections are discussed in section 2. The error models are installed exclusively on the connections between dishes and satellites, using an instance of the ns3:ErrorModel [25] class.

5.2.1 Bit Error Ratio (BER) Error Model

Bit errors are common in satellite network simulations, as discussed in section 2.2. The Bit Error Ratio (BER) Error Model induces bit errors at a constant ratio over the course of an entire simulation. When this model is applied to a connection, the probability of a bit being corrupt is given by a discrete distribution with two possible values, namely: 1, meaning the bit is corrupt and 0 meaning the bit is not corrupt. The expected value of the distribution, which represent the frequency of bit errors, is then modulated from 10^{-3} to 10^{-9} over different simulations to emulate the possible BER ranges [2]. Therefore, it is expected that we obtain for both MPTCP and TCP the best throughput at $BER = 10^{-9}$ and lowest at $BER = 10^{-3}$.
The further models illustrated in this section are presented in the literature using Packet Error Ratio (PER) as a unit, but the BER Error Model is implemented using BER as a unit, thus it is paramount to know the relationship between BER and PER [26]:

$$PER = 1 - (1 - BER)^l$$

where l is the packet size in bits, in our case $l = 8 \cdot 10^3$ bits. We are using this relationship to plot the throughput data as a function of the PER in section 6.

5.2.2 ON/OFF Error Model

The ON/OFF Error Model emulates the losses caused by the signal degradation that occurs when the satellite approaches lower latitudes in the sky, as described in section 2.3. In this case an ON/OFF error pattern is generated. During the ON phase no packets are lost, meanwhile during the OFF phase packets are lost at a ratio p, similar to [1].

To obtain a different ON/OFF pattern at each execution (i.e. with each execution seed), the values of the ON phase and OFF phase duration are not constant over the whole simulation, rather extracted from a standard distribution. The implementation of such sequence is achieved using an instance of the ns3:NormalRandomVariable class [27], which allows us to compute random numbers from a standard distribution by specifying the variance and the expected mean value. The variance values are arbitrarily chosen, namely 0.5 seconds for the ON phase duration and 0.1 seconds for the OFF phase duration.

This error model takes into account two different scenarios:

In the first scenario we keep the mean ON phase duration at 6 seconds and the mean OFF phase duration at 1 seconds and we modulate the packet loss ratio p from $5 \cdot 10^{-1}$ to 10^{-6}.

In the second case we keep the mean OFF phase duration at 1 second and the packet loss ratio p at 10^{-2} and we modulate the mean ON phase duration from 4s to 8s, similar to [1].

5.2.3 BER Plus ON/OFF Error Model

The BER plus ON/OFF Error Model is a combination of both the BER Error Model (section 5.2.1) and the ON/OFF Error Model (section 5.2.2). This error model is therefore, also characterized by ON/OFF phases. During the ON phase, the BER Error Model is applied to the connection and during the OFF phase, packets get dropped at a ratio p, similar to the ON/OFF Error Model (section 5.2.2). The ON/OFF patterns are computed exactly in the same way as in the ON/OFF Error Model. The only effective difference between this error model and the ON/OFF Error Model is that, in the ON/OFF Error Model, there is no packet loss during the ON phase. On the other hand, in the BER plus ON/OFF Error Model, packets can also get dropped during the the ON phase.

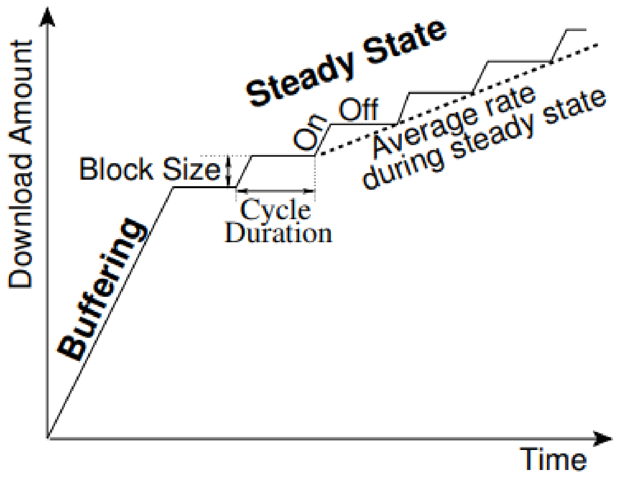
The parameters we use to execute the simulations that we use in this error model are the following: we keep p constant at a PER of 10^{-2}, we keep the mean ON phase duration and the mean OFF phase duration constant at 6 and 1 second respectively and we modulate the bit error ratio effective during the ON phase from 10^{-3} to 10^{-9}.

It is important to notice that NS3 allows only one error model instance per network device (connection) at a specific moment in time. This means that the ON/OFF Error Model cannot coexist on a connection with the BER Error Model, because that would require two instances of the ns3:ErrorModel class [25] to be present at the same time on the same network device. The solution is to shortly shut off the BER Error Model and turn on the OFF phase on the connection until the OFF phase time is over and then instantly re-replace the OFF phase with the BER Error Model. This renders the error model slightly less realistic because the BER and ON/OFF phase are caused by two different physical phenomena (section 2) that do not depend on each other. In a real world scenario these two phenomena can simultaneously be affecting a connection.

5.3 Data Stream / Applications

Multimedia represents a big part of today's Internet traffic. The two most common multimedia containers on the Internet are Adobe Flash and HTML5. Both use different streaming strategies. The relevance of Adobe Flash is decreasing since it will no longer supported after the year 2020, thus we focus on data streams generated by HTML5 videos. YouTube uses TCP to transfer video content [11]. The generated traffic is mainly characterized by two sending phases: a buffering phase followed by an On-Off phase.

Figure 6: Data Stream of HTML5 Videos [11].



5.3.1 Buffering Phase / Bulk Send Application

The buffering phase [11] is the first to be initialized and it consists of a constant stream of data, i.e. a data stream without holes, as shown in Figure 6. The data rate during the buffering phase is dictated by the available end-to-end bandwidth. The buffering phase ensures that the video player has enough stored data to compensate for the variation of bandwidth that might occur later in time during the video playback [11]. To simulate the buffering phase in NS3 we use a ns3::BulkSendApplication [28] because such an application is also implemented to adapt the send rate to the available bandwidth by continuously sending packets until the available TCP send window size reaches zero.

5.3.2 On-Off Phase / On/Off Application

The On-Off phase [11] or Steady State starts after the buffering phase is completed. The Steady State consists of periodic On and Off cycles. During the On period a block of data is sent at a rate which corresponds to the available bandwidth. During the Off period the TCP connection is idle [11]. This creates On and Off patterns, as shown in Figure 6.

There are two types of On-Off streaming strategies, namely: long On-Off cycles

and short On-Off cycles. Long On-Off cycles are characterized by blocks of data larger than 2.5MB and short On-Off cycles by blocks smaller than 2.5MB. YouTube makes use of short On-Off cycles when streaming both HTML5 and Adobe Flash videos using the Internet Explorer Browser [11], thus, we are focusing on short On-Off cycles for our simulations.
In NS3 we can emulate this On-Off behavior by using an ns3:OnOffApplication [29]. The ns3::OnOffApplication that we use for our simulations, has an average Off period of 2 seconds and an average On period of 0.5 seconds in which data is sent at a constant rate of 20Mbps. This is consistent with short On-Off cycles found in HTML5 YouTube videos.

5.4 TCP Simulation

In the first simulation phase, we evaluate TCP separately over the network displayed in Figure 5 to measure its throughput. In the first batch of simulations we install On/Off Application (section 5.3.2) instances on five terminals on the left end of the network (Figure 5). These applications then try to establish TCP sessions with the other five terminals on the right end of the network (Figure 5).
A single MPTCP connection usually makes use of more bandwidth than a single TCP connection, because it creates multiple subflows to communicate between two hosts. To have a fairer comparison between the two protocols, we make TCP use as many paths, to the same host, as MPTCP does. To achieve this we equally split the 20Mbps data stream into multiple separate plain TCP connections. This allows us to use more bandwidth than if we would use one single TCP connection, since we establish more paths to the same host. The network topology we use for our simulations (Figure 5) allows for 864 different paths with the same cost from one terminal on one end to any of the other five terminals on the other end. The routing strategy adopted to establish multiple connections over different paths is Equal-Cost Multipath (ECMP) routing [30]. This setup still has a disadvantage to MPTCP, in that it does not have the load balancing capability that MPTCP usually has [31]. In this setup, the

data stream is always equally distributed among all TCP connections.
In NS3 this can be simulated by creating a new separate application instance for each TCP connection, but in a real world scenario the data would come from a single application. Therefore, for a real world scenario, this assumes some sort of scheduler, which takes the data stream from one application, splits it and forwards it to multiple separate TCP connections. In addition to that, the overall order of delivery is not guaranteed when using multiple plain TCP connections, therefore, the packets that belong to the same stream have to be reordered before they are sent to the application layer. This reordering is not be taken into consideration. In contrast, the packets will be considered received as soon as they get acknowledged by the single TCP connection, no matter in which order.
For every error model (section 5.2) and for both data stream types (section 5.3) we simulate from one to fifteen separate TCP connections per terminal. Each terminal that coexists in the same simulation, has the same number of separate TCP connections and the same type of data stream.
The results of this setup with an On/Off application are illustrated in section 6.2.1 and with a Bulk Send application in section 6.3.1. The results are depicted in the following order: for each error model (section 5.2) we present four plots: the first plot represents the results we obtain with the five terminals (Figure 5) sending data with the use of one TCP connection each, the second plot with the use of five TCP connections each, the third plot with ten TCP connections each and the fourth with fifteen connections at each terminal. The total amount of TCP connections per simulation is therefore: 5, 25, 50 and 75.
The TCP implementation we are using is the standard TCP implementation available in NS3 using New Reno [32] as congestion control algorithm.
The TCP simulation scenarios implementation can be found in [33].

5.5 MPTCP Simulation

In the second simulation phase we simulate multipath TCP in the network shown in Figure 5. We install a single application on each of the five terminals on one end and they establish a connection and send data to the other five terminals on the other end of the network with the use of an MPTCP connection. The routing strategy we use to map multiple MPTCP subflows to different paths is Equal-Cost MultiPath (ECMP) [22], similar to our TCP simulations.

We separately simulate scenarios that make use of On/Off applications (section 5.3.2) and those that make use of Bulk Send applications (section 5.3.1). To achieve a similar setup as for TCP (section 5.4), we install one application per terminal which sends packets to one single MPTCP socket, so that each MPTCP simulation always has exactly five MPTCP sessions open at a time. Each open MPTCP connection, belonging to the same simulation, has the same number of subflows and the same data stream type, similar to the TCP simulation scenario (section 5.4). We simulate MPTCP using one to fifteen subflows per connection.

The results of the MPTCP simulations using the On/Off application instances are illustrated in section 6.2.2 and using the Bulk Send application in section 6.3.2. There are four plots for each error model in sections 6.2.2 and 6.3.2. The first plot is the throughput achieved with one subflow per MPTCP connection, the second with five subflows per MPTCP connection, the third with ten subflows per MPTCP connection and the fourth with fifteen subflows per MPTCP connection. Therefore, the total amount of subflows in the simulation scenario is 5 in the first case, 25 in the second case, 50 in the third case and 75 in the last case accordingly.

The MPTCP version we are using [34] makes use of Round Robin [35] as a scheduling algorithm to distribute the data between the different subflows.

The MPTCP simulation scenarios implementation can be found in [36].

5.6 Mixed Scenario

The mixed scenario is a mix between the TCP (section 5.4) and the MPTCP (section 5.5) simulation scenario. Two of the five hosts on one end of the network (Figure 5) make use of MPTCP to send data and two other nodes on the same end of the network make use of uncoupled TCP connections, so that both protocols should get a fair share of the bandwidth available, thus, leaving one terminal per end inactive.

We simulate On/Off applications (section 5.3.2) and Bulk Send applications (section 5.3.1) separately.

In each simulation, the number of uncoupled TCP connections per host is equal to the number of subflows per MPTCP connection. We simulate one TCP connection per "TCP host" simultaneously with one subflow per "MPTCP host", five TCP connections per "TCP host" simultaneously with five subflows per "MPTCP host", ten TCP connections per "TCP host" simultaneously with ten subflows per "MPTCP host" and fifteen TCP connections per "TCP host" simultaneously with fifteen subflows per "MPTCP host". Since we are using two MPTCP hosts and two TCP hosts, the total amount of TCP connections and MPTCP subflows per scenario is; two TCP connections plus two MPTCP subflows, ten TCP connections plus ten MPTCP subflows, twenty TCP connections plus twenty MPTCP subflows and thirty TCP connections plus thirty MPTCP subflows in the last scenario.

The results of this setup using the On/Off applications (section 5.3.2) are illustrated in section 6.2.4 and using the Bulk Send applications (section 5.3.1) in section 6.3.4. The Mixed Scenarios simulation implementation can be found in [36].

6 Results

In this section the simulation results are discussed. The horizontal axis of the Figures illustrated in this section represents the following parameters: the packet error ratio (PER) over the entire simulation for the BER Error Model (section 5.2.1), the OFF phase PER or the mean ON phase duration for the ON/OFF Error Model (section 5.2.2) and the ON phase PER for the ON/OFF plus BER Error Model (section 5.2.3). The vertical axis represents the throughput. To plot the data we obtain from our simulations, we average the throughput between the simulations that use the same error model (section 5.2), same protocol setup (sections 5.4, 5.5 and 5.6) and same PER or mean ON phase duration.

We obtain different results with the aforementioned configuration by modulating the simulation duration from 30 to 480 seconds and by using different execution seeds. These results are then averaged and used to compute the standard deviation, which in the following plots represent the confidence intervals.

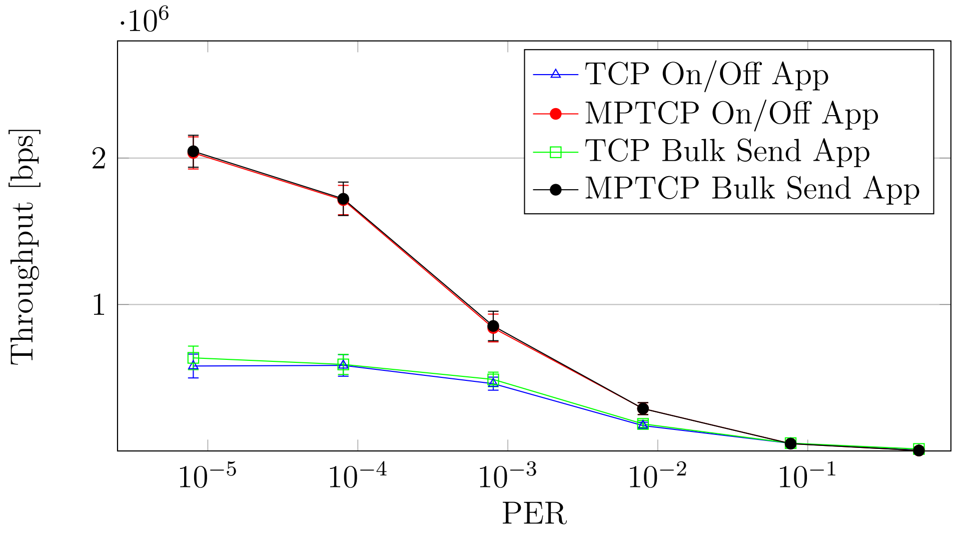
Our BER Error Model (section 5.2.1) and BER plus ON/OFF Error Model (section 5.2.3) implementations use the bit error ratio (BER) as a parameter, thus the relation discussed in section 5.2.1 is used to compute the equivalent PER for the following plots.

Important to notice is that, in this sections 6.2 and 6.3, the plotted throughput displays the throughput for the whole network and not for a single connection. This means the data points represent the mean single connection throughput multiplied by five in case of non mixed scenarios and multiplied by two for the mixed scenarios.

6.1 Single Flow/Connection

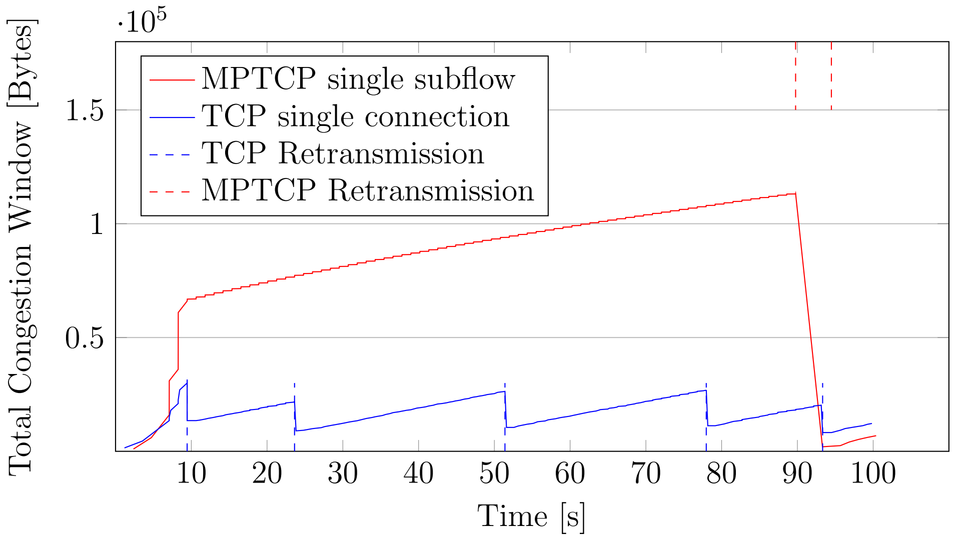
To understand the behavior of multiple TCP connections and of multiple MPTCP subflows in the simulation scenario discussed in section 5.1, it is useful to understand the behavior of one single flow or one connection.
In this section the throughput of one single MPTCP subflow and of one single TCP connection is plotted as a function of the packet error ratio (PER) in Figure 7. The congestion window of both protocols as a function of the simulation time is presented in Figure 8.

Figure 7: Throughput for different PERs by using the BER Error Model with one single TCP session and MPTCP with a single subflow.



The results we plot in Figure 7 are obtained with the following setup: one terminal on one end of the network (Figure 5) established a TCP or MPTCP connection respectively with one terminal on the other end of the network. We install an On/Off application (section 5.3.2) or a Bulk Send application (section 5.3.1) to generate the data stream and we apply the BER Error Model (section 5.2.1) to the dish-to-satellite connections. This configuration is guaranteed to reach a high level of congestion, since the dish to satellite connections can support a maximum data rate of 20 Mbps and the protocols establish a connection that uses only one path to connect to the receiving terminal. MPTCP outperforms TCP because the congestion window of MPTCP is, for most of the duration of the simulation, much higher than the congestion window of TCP, as illustrated in Figure 8.

Figure 8: Congestion Window as a function of the simulation time by using the BER Error Model at BER = 10^{-8}, a Bulk Send application, one TCP connection and one MPTCP subflow.



In Figure 8 we plot the congestion window as a function of the simulation time. The dashed lines on top of the graph represent the retransmissions executed by MPTCP and the ones on the bottom represent the retransmissions executed by TCP. This data comes from the same simulations described for Figure 7. The difference in the congestion window behavior accounts for the significant throughput difference of Figure 7. There are three main reasons for which the TCP congestion window cannot grow as fast as the MPTCP congestion window initially, namely:

1. The TCP and MPTCP scenarios are slightly different, the order in which the Bit Errors take place varies because the data comes from two different simulations instances, thus, resulting in slightly different simlations.

2. MPTCP does not treat ACKS duplicates with any MPTCP option unless it contains a DSS option and uses the ADD_ADDR option to separate different ACKS as per [8], meanwhile plain TCP NewReno does not make such distinction.

3. MPTCP uses Selective Acknowledgements (SACK) [8] on the subflow level, which leads to a lower ACK frequency, which in turn lower the likelihood of receiving three duplicate ACKs.

The result is a higher frequency of retransmission for TCP Sessions which closes the congestion window more frequently than for MPTCP subflows. This results in a smaller congestion window than MPTCP, and accounts for the smaller throughput of TCP.

Clearly visible in Figure 8 is the Slow Start Threshold of MPTCP, which is reached at around 10 seconds. Both protocols in every simulation have an initial Slow Start Threshold of 655 kB.

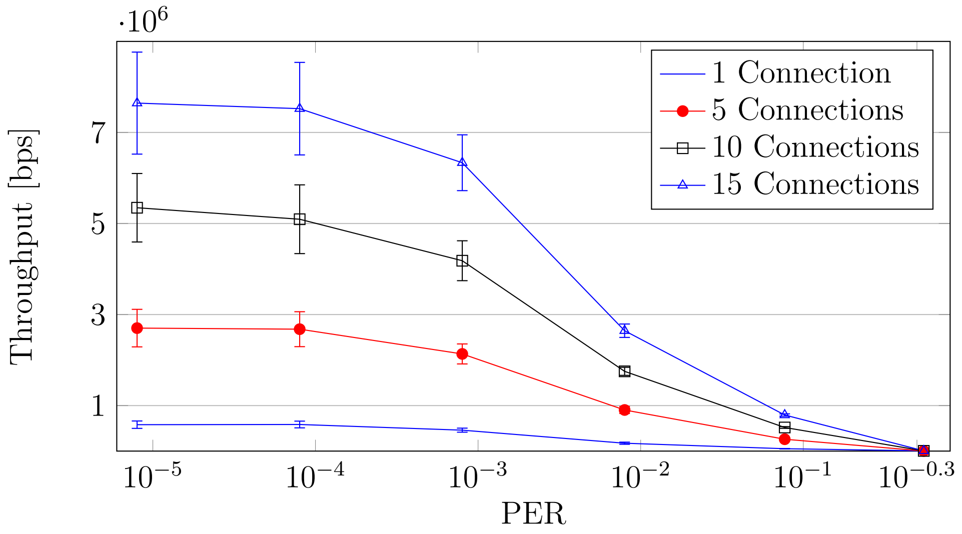
6.2 On/Off Application Results

This section displays the results obtained from the simulations using an On/Off data stream application on each terminal, which is described in section 5.3.2.

6.2.1 TCP

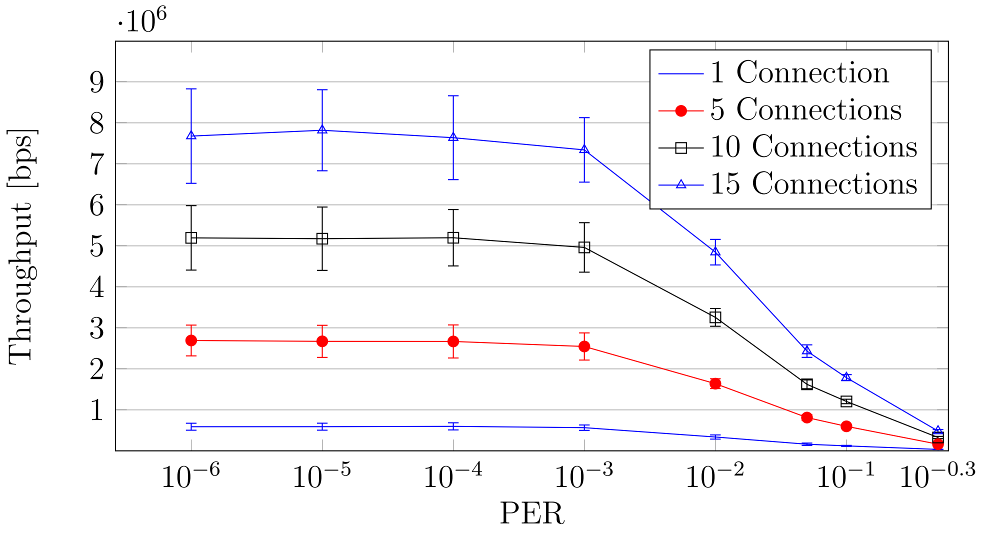
The setup that yields the following results is discussed in further details in section 5.4. Important to notice is that the following four graphs each display four plots, which represent the number of connections per terminal on the left end of the network (Figure 5). The legend of the following plots indicates the number of TCP connections per terminal. Since we use five terminals per end, 1 connection per terminal amounts to 5 total connections, 5 connections per terminal amount to 25 total connections, 10 connections per terminal amount to 50 total connections, and 15 connections per terminal amount to 75 total connections over the whole network. The vertical axis represents the overall network throughput and not the throughput per terminal.

Figure 9: Throughput for different PERs by using the BER Error Model with 1, 5, 10 and 15 different TCP connections per host.



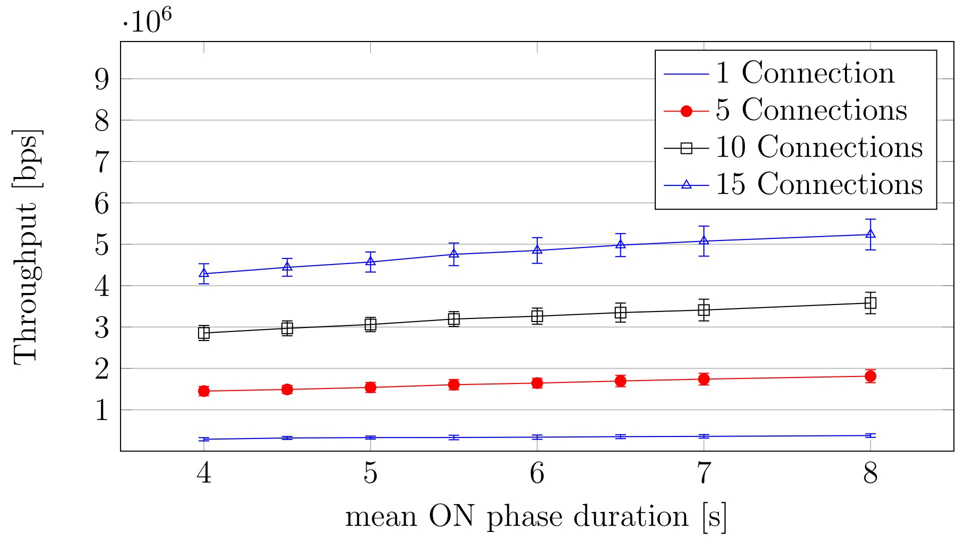
In Figure 9 we plot the throughput of TCP as a function of the packet error ratio. The error model used is the BER Error Model (section 5.2.1). The throughput is linearly proportional to the amount of TCP connections, e.g., at a packet error ratio of 10^{-5}, five connections per terminal yield five time the throughput of one connection per terminal, ten connections per terminal yield ten times the throughput that a single connection per terminal yields and fifteen connections per terminal yield almost fifteen times the throughput of a single connection per terminal. The throughput increases as the packet error rate decreases, because more packets can be received without retransmissions. Retransmissions delay the delivery time of the lost packets, thus, lowering the throughput.

Figure 10: Throughput for different PERs by using the ON/OFF Error Model with 1, 5, 10 and 15 different TCP connections per host.



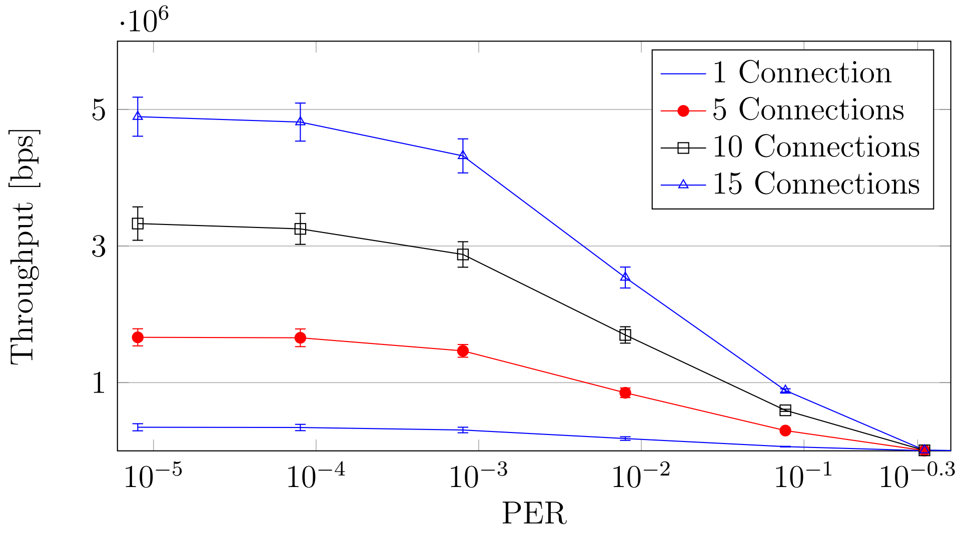
In Figure 10 we plot the throughput of TCP with the use of the ON/OFF Error Model (section 5.2.2). In contrast to Figure 9, the horizontal axis in Figure 10 does not represent the overall packet error ratio, rather the error ratio during the OFF period. The overall PER is, therefore, lower than the PER displayed on the horizontal axis, because the PER during the ON phase is zero. This is the reason we obtain higher throughputs than in Figure 9 for the same entries in the horizontal axis. The throughput increases almost linearly with the number of TCP connections per host, because the additional TCP connections are established on paths that are mostly not being used. At packet error ratios smaller than 10^{-4} the throughput stabilizes and even if the error ratios decreases, there is almost no throughput gain. This is due to the fact that the overall PER is very low, thus, the chance of packet a being lost is close to zero.

Figure 11: Throughput for different mean ON phase durations by using the ON/OFF Error Model with 1, 5, 10 and 15 different TCP connections per host.



In Figure 11 we plot the throughput of TCP with the use of the ON/OFF Error Model (section 5.2.2) as a function of the mean ON phase duration. This plot differs from that in Figure 10 in that the packet error ratio is kept constant at 10^{-2} and the mean ON phase duration is modulated from 4 to 8 seconds. The throughput grows linearly with the number of TCP connections. The On phase represents the period in time in which no packet error occurs. Therefore, if the mean ON phase duration increases and the packet error ratio during the Off phase stays constant, we expect a throughput increase, because the packet error ratio over the duration of the whole simulation decreases. In Figure 11, this expectation is met and these results are consistent with [1].

Figure 12: Throughput for different PERs by using the ON/OFF plus BER Error Model with 1, 5, 10 and 15 different TCP connections per host.

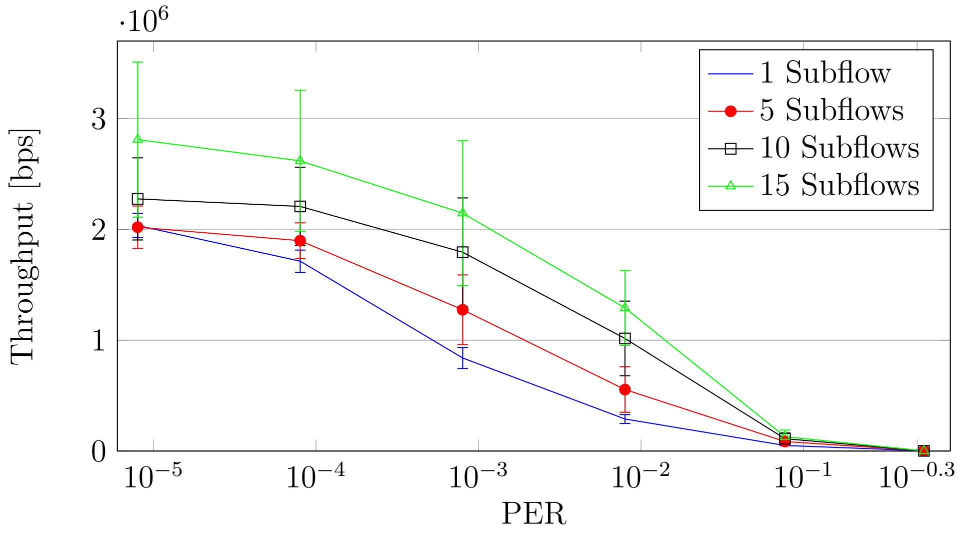


In Figure 12 we plot the throughput of TCP with the use of the BER plus ON/OFF Error Model, discussed in section 5.2.3, as a function of the packet error ratio during the On phase. The throughput is expected to be smaller than that of the ON/OFF Error Model (section 5.2.2) shown in Figure 10, because, unlike with the ON/OFF Error Model, bit errors can also occur during the ON phase and not only during the OFF phase. The throughput increases almost linearly with the number of connections, because the additional TCP connections can establish connections on paths that are mostly not used, similar to Figures 9, 10 and 11. The throughput increases as the packet error ratio decreases. This is due to the fact that, as less packets are dropped, less retransmissions are needed, and the overall in-order delivery time decreases. Thus, the throughput is increased.

6.2.2 MPTCP

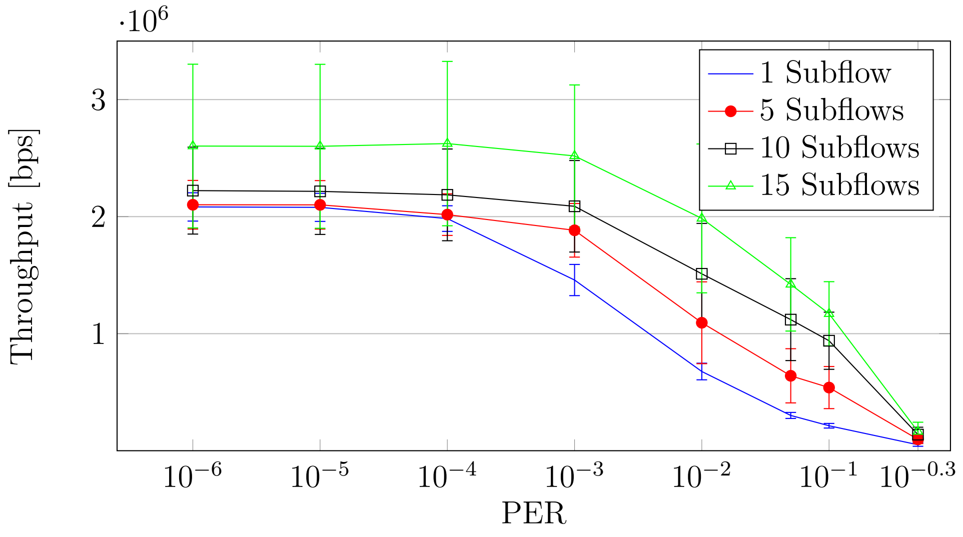
The setup that yields the following four plots is further discussed in section 5.5. Paramount for the understanding of the following plots is that each terminal on one end of the network makes use of one single MPTCP connection to send data to the other end of the network. Each graph illustrates four plots, the first plot represents the throughput obtained with each connection having one single subflow, the second plot with 5 subflows per MPTCP connection, the third with 10 subflows per MPTCP connection and the fourth with 15 subflows per MPTCP connection. The vertical axis represent the throughput for the whole network and not that for a single connection.

Figure 13: Throughput for different PERs by using the BER Error Model with 1, 5, 10 and 15 MPTCP subflows per host.



In Figure 13 we plot the throughput of MPTCP with the use of the BER error model (section 5.2.1) as a function of the packet error ratio, analogous to Figure 9. The throughput does not increases in a linear fashion with the number of subflows as in the previous TCP results in section 6.2.1. Nonetheless, the throughput difference between 1 and 15 subflows per connection is significant at packet error rates between 10^{-2} and 10^{-3}. In fact, in this case, the throughput is not 15 times higher as in the previous TCP results (section 6.2.1), but it is still more than twice as much as for a single subflow per connection. The lower throughput differential between the plots is caused by the congestion control algorithm coupling described in section 4.3.3. If one subflow is performing poorly because of the high round trip time and high packet error rate, the congestion window of all subflows gets substantially reduced as soon as they enter the Congestion Avoidance phase. The substantial reduction of the size of the congestion window, over the course of the simulation, is the cause of the large confidence intervals illustrated in Figure 13. The congestion window is further discussed in section 6.2.5.

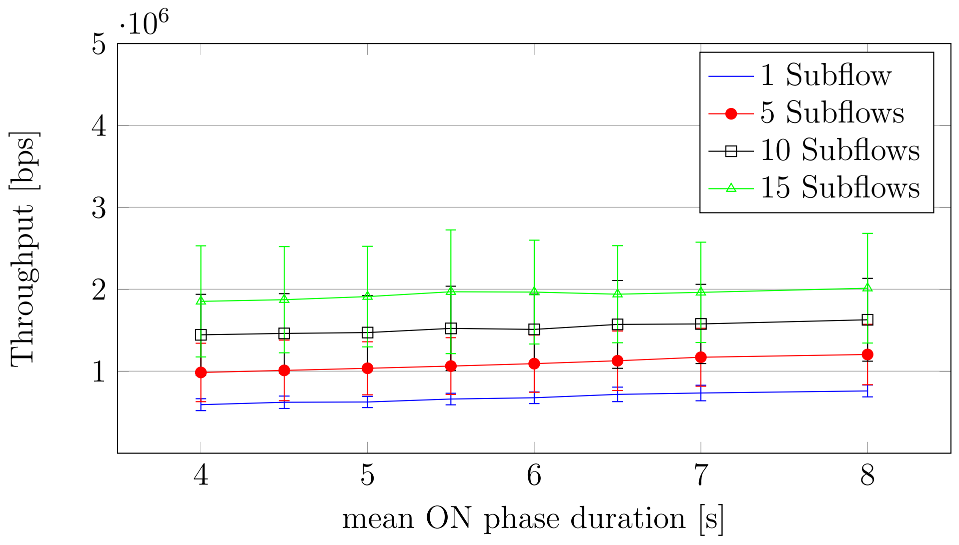
Figure 14: Throughput for different PERs by using the ON/OFF Error Model with 1, 5, 10 and 15 MPTCP subflows per host.



In Figure 14 we plot the throughput of MPTCP with the use of the ON/OFF Error Model (section 5.2.2) as a function of the packet error ratio during the OFF phase, analogous to Figure 10. At PERs between 10^{-1} and 10^{-3} the throughput difference between the throughput obtained with a different number of subflows per terminal is significant. In fact, at PERs between 10^{-1} and $5 * 10^{-1}$, 15 subflows per terminal can transfer six times the data that one single subflow per terminal can transfer. At PERs between $5 * 10^{-1}$ and 10^{-4}, 15 subflows can transfer about three times as much data as a single subflow can. At lower packet error ratios, between PERs of 10^{-4} and 10^{-6}, the throughput gain measured by adding multiple subflows to each

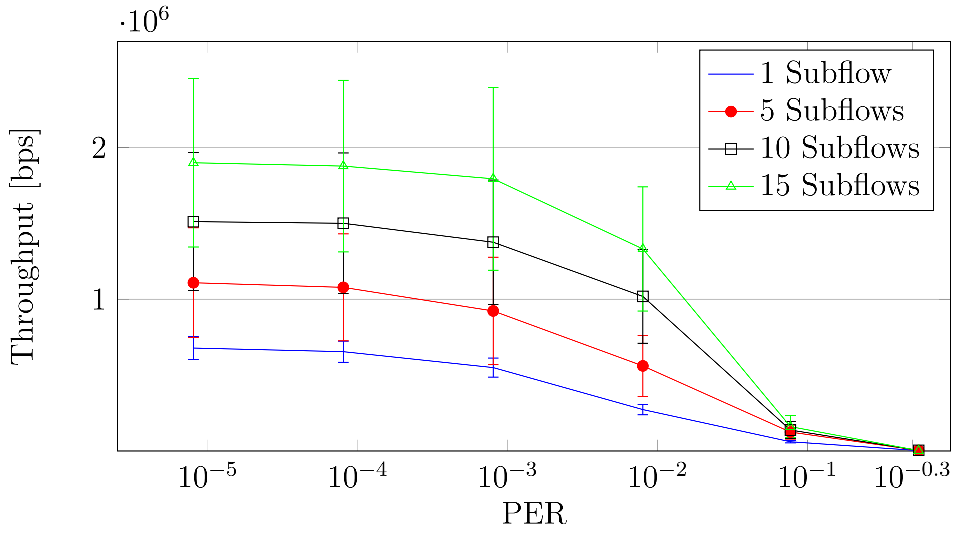
MPTCP connection is minimal up to 10 subflows. 15 subflows per terminal yield a more substantial throughput gain of about 500 kbps. From a packet error rate of 10^{-4} to 10^{-6} no significant improvement is measured as the PER is lowered. Nonetheless, the gain achieved by adding additional subflows per terminal is not as significant as for the TCP scenario illustrated in Figure 10, because the subflows are coupled by congestion control algorithm discussed in section 4.3.3.

Figure 15: Throughput for different mean ON phase durations by using the ON/OFF Error Model with 1, 5, 10 and 15 MPTCP subflows per host.



In Figure 15 we plot the throughput of MPTCP with the use of the ON/OFF Error Model (section 5.2.2) as a function of the mean ON phase duration, analogous to Figure 11. The mean OFF phase duration is kept constant at 1 second, the packet error ratio during the OFF phase is kept constant at 10^{-2} and the mean ON phase duration is modulated from 4 seconds to 8 seconds. Therefore the data points at 6 seconds correspond to the data points in Figure 14 at a PER of 10^{-2}. Theoretically, in terms of PER, the modulation of the mean ON phase duration from 4 to 8 seconds corresponds to a very small PER modulation around the PER $= 10^{-2}$ region in Figure 14. The modulation of the mean ON phase duration alters the overall packet error ratio, which in turn should alter the throughput. This throughput differential is measured, as illustrated in Figure 15, but it clearly does not stand out as much as in Figure 14, because we plot the PER logarithmically and the mean ON phase duration linearly.

Figure 16: Throughput for different PERs by using the ON/OFF plus BER Error Model with 1, 5, 10 and 15 MPTCP subflows per host.

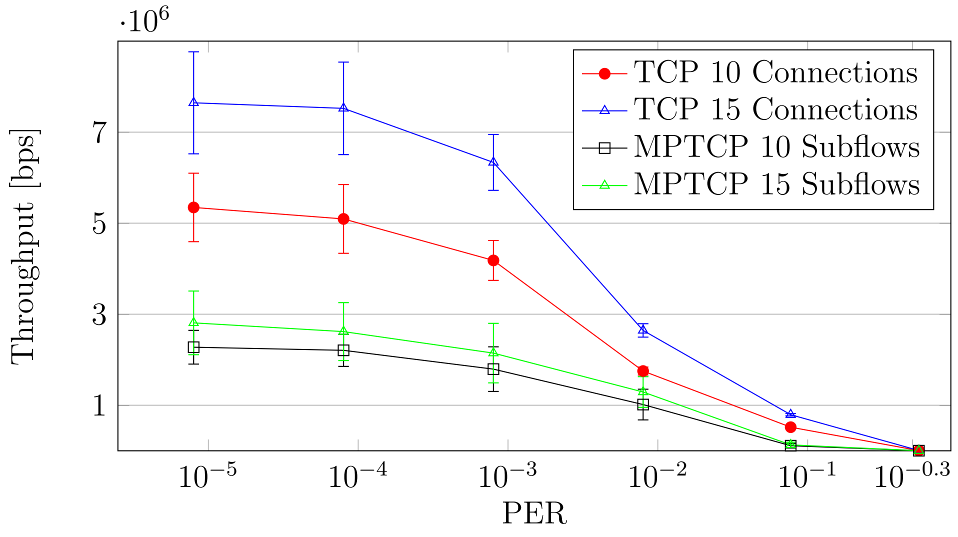


In Figure 16 we plot the throughput of MPTCP with the use of the BER plus ON/OFF Error Model (section 5.2.3) as a function of the ON phase packet error ratio, analogous to Figure 12. The throughput is lower than with the ON/OFF Error Model, illustrated in Figure 14, because in both the ON and the OFF phase the dish-to-satellite links are subject to packet errors. The throughput gain achieved by adding additional subflows to every MPTCP connection is more significant than the gain measured with the former error models in Figures 13, 14 and 15. At lower packet error rates, from 10^{-3} to 10^{-5}, the throughput gain by adding 4 subflows is about 400 kbps, by adding 9 subflows about 900 kbps and by adding 14 subflows about 1.2 Mbps. At higher packet error ratios, the gain tends to become smaller, nonetheless significant, when comparing a single subflow to 15 subflows. In fact, at a PER of 10^{-2}, 15 subflows can transfer six times as much data as one single subflow can transfer. As previously explained in section 5.2.3, the packet error rate during the OFF phase is kept constant at PER $= 10^{-2}$ and the PER during the OFF phase is modulated from $5 \cdot 10^{-1}$ to 10^{-5}. Therefore, since we plot the throughput as a function of the ON phase PER and not as a function of the overall PER, the entries of the horizontal axis in Figure 16 that are smaller than 10^{-2} represent a smaller overall PER. The entries bigger than 10^{-2} represent a bigger overall PER. The smaller overall PER is the reason for the bigger throughput gains achieved by adding additional subflows. In spite of this, the addition of more subflows is not as significant as adding multiple TCP connections, as shown in Figure 12.

6.2.3 TCP MPTCP Comparison

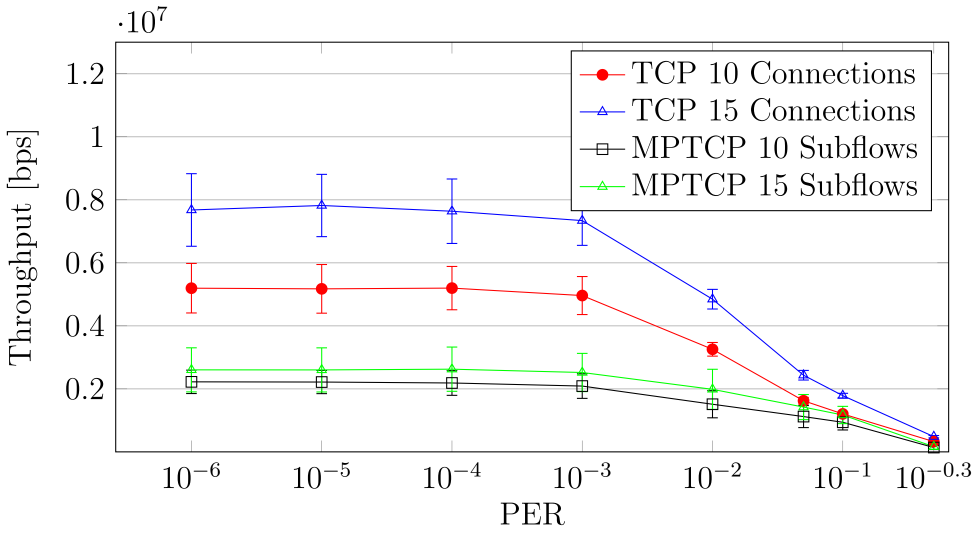
In this section we compare the data from section 6.2.1 to the data from section 6.2.2. No new simulation is executed, we take the data from the two previous sections and we compare it. We take the two setups with the most connections and subflows and we plot it in the same Figures, to compare how the best performing conditions for both MPTCP and TCP with an On/Off application (section 5.3.2) perform for each error model.

Figure 17: Throughput comparison for different PERs by using the BER Error Model.



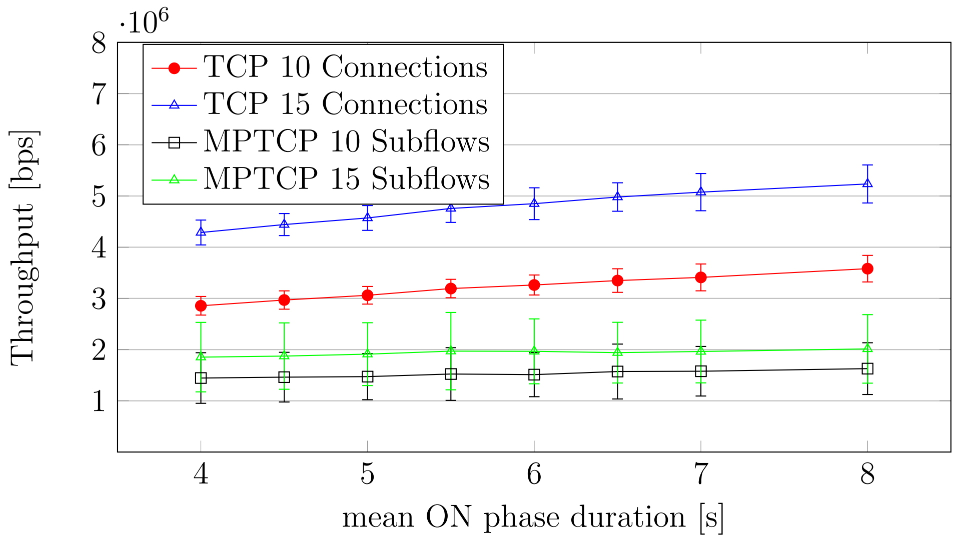
In Figure 17 we compare the data from Figures 9 and 13. At high packet error ratios, between the PERs of 10^{-1} to 10^{-2}, 15 separate TCP connections can transfer more than two times as much data as MPTCP with 10 and 15 subflows. At lower PERs, TCP can transfer significantly more data than MPTCP. 15 separate TCP connections can transfer almost three times as much data as MPTCP with 15 or 10 subflows at PERs smaller than 10^{-2}. This substantial throughput gain is due to the fact that different TCP connections are not coupled in any way, meanwhile MPTCP subflows are coupled by the congestion control algorithm (section 4.3.3), which enforces a congestion window, which is inversely proportional to the round trip time. In section 6.2.5, we further explain the behavior of the congestion window of TCP and MPTCP.

Figure 18: Throughput comparison for different PERs by using the ON/OFF Error Model.



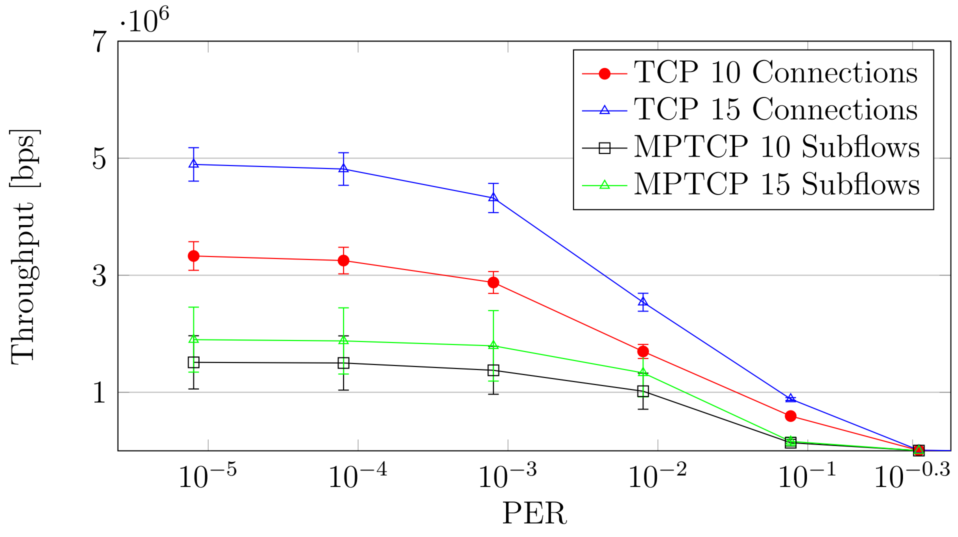
In Figure 18 we compare the data from Figures 10 and 14. 10 and 15 separate TCP connections can transfer more data than 10 and 15 MPTCP subflows at any simulated packet error ratio. At high packet error ratios, between $5 \cdot 10^{-1}$ and $5 \cdot 10^{-2}$, the throughput difference between TCP and MPTCP is not substantial. At lower packet error ratios, from 10^{-2} to 10^{-6}, the amount of data that TCP can transfer, is significantly higher. In fact, 10 separate TCP connections can transfer from 2 to 2.5 times more the data than 10 MPTCP subflows can transfer and 15 TCP connections can transfer from 2.5 to 4 times more the data than 15 MPTCP subflows can. The TCP throughput grows linearly with the number of TCP connection, while the MPTCP does not. The reason for this lyes on the congestion control algorithm discussed in section 4.3.3.

Figure 19: Throughput comparison for different mean ON phase durations by using the ON/OFF Error Model.



In Figure 19 we compare the data from Figures 11 and 15. At a mean ON phase duration of 6 seconds, the data points correspond to those in Figure 20 at PER $= 10^{-2}$. In terms of overall PER, the mean On phase modulation from 4 to 8 seconds corresponds to a small modulation around the region of PER $= 10^{-2}$ in Figure 18. The throughput change through the modulation of mean ON phase duration is, therefore, dependent on the slope at PER $= 10^{-2}$ in Figure 18. Fifteen separate TCP connections can transfer about 2.5 times more data than fifteen MPTCP subflows can and ten TCP connections can transfer about 2 times more data than ten MPTCP subflows can transfer. TCP performs better than MPTCP, because as packet losses occur, the congestion windows of the separate TCP connections do not affect each other when the congestion window is reduced. Nonetheless, as discussed in section 4.3.3, the congestion windows of a single MPTCP subflow are coupled as soon as they enter the Congestion Avoidance phase. The congestion window coupling, defined by the Linked Increases Algorithm (section 4.3.3.1), performs poorly when the round trip time is significantly high, therefore, significantly slowing the reopening of the congestion windows. The smaller total congestion window of MPTCP accounts for the smaller overall throughput of MPTCP. The congestion window is further discussed in section 6.2.5.

Figure 20: Throughput comparison for different PERs by using the ON/OFF plus BER Error Model.

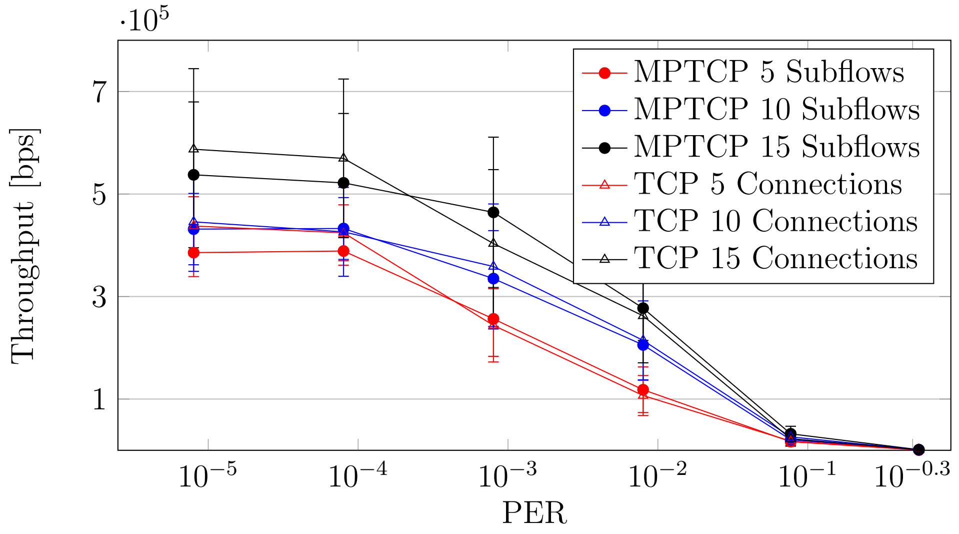


In Figure 20 we compare the data from Figures 12 and 16. Fifteen and ten separate TCP connections performs better than fifteen and ten MPTCP subflows at any of the simulated packet error ratios. At packet error ratios between 10^{-3} and 10^{-5} the difference between the TCP and the MPTCP throughput is significant. Ten TCP connections yield about 2 times more throughput than ten MPTCP subflows and fifteen separate TCP connections about 2.5 times more data than fifteen MPTCP subflows. TCP performs better than MPTCP, because the total congestion window of TCP is higher than the total congestion window of the MPTCP connections for most of the simulation time. As previously mentioned, this is due to the fact that the congestion window coupling of MPTCP impedes the congestion window to reopen after the subflows enter the Congestion Avoidance phase. The congestion window is further discussed in section 6.2.5.

6.2.4 Mixed Scenario

The mixed scenario is presented in further details in section 5.6. In this scenario we simulate MPTCP connections and TCP connections simultaneously. The setup that yields the following results is as follows: MPTCP with five subflows is simulated simultaneously in the same scenario with five separate TCP connections, ten MPTCP subflows per MPTCP host are simulated with ten TCP connections per TCP host simultaneously and fifteen MPTCP subflows per MPTCP host with fifteen TCP connections per TCP host. There are two TCP hosts and two MPTCP hosts simultaneously in every simulation. For better readability the plots that are extracted from the same simulation batch is colored the same. In the following four plots, the vertical axis represents the throughput achieved by the displayed protocol. Since there are only two hosts that make use of the same protocol, the plotted throughput is expected to be lower than for the simulations in sections 6.2.1, 6.2.2 and 6.2.3, which simulate five hosts using the same protocol.

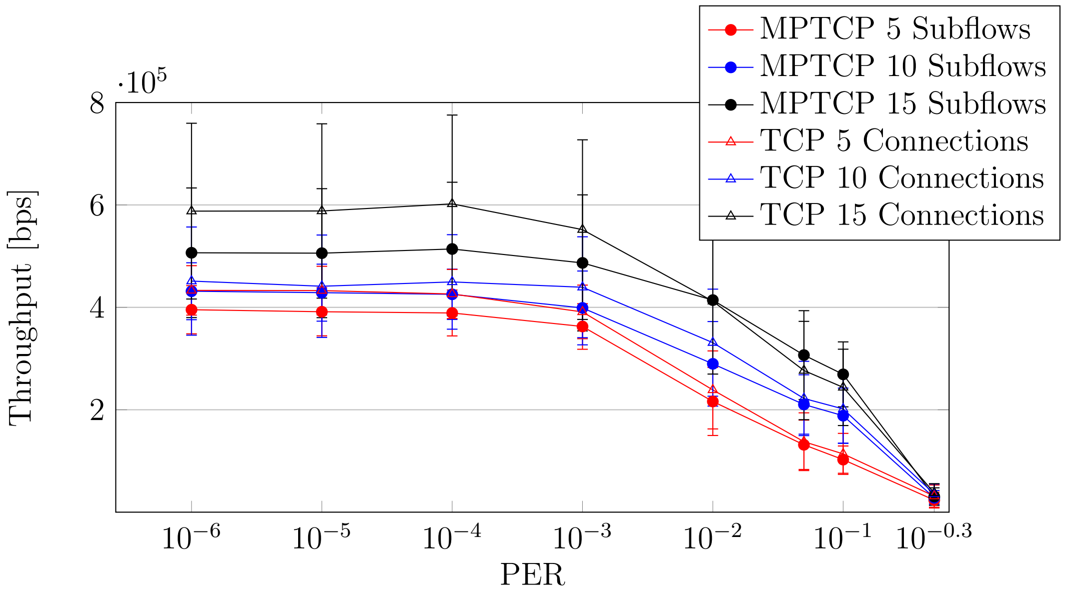
Figure 21: Throughput for different PERs by using the BER Error Model in mixed scenarios.



In Figure 21 we plot the throughput of the following three mixed scenarios: MPTCP with 15 subflows simulated simultaneously with 15 separate TCP sessions, MPTCP with 10 subflows simulated simultaneously with 10 separate

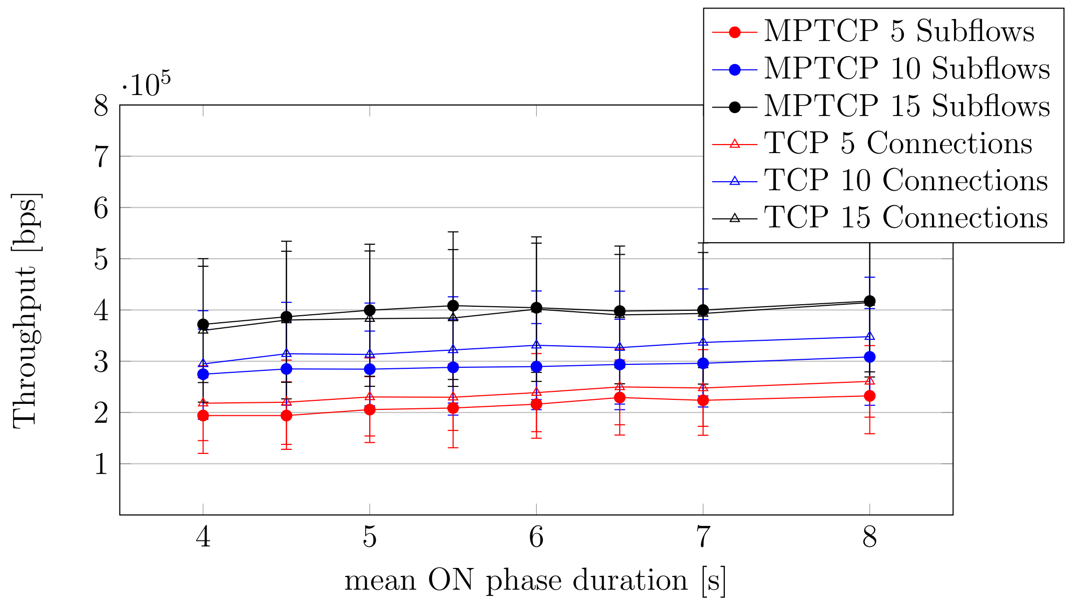
TCP sessions and MPTCP with 5 subflows simulated simultaneously with 5 separate TCP sessions using the BER Error Model (section 5.2.1). We plot the results as a function of the packet error ratio. Separate TCP connections do not gain performance linearly with the number of connections, unlike in the TCP-only scenarios (section 6.2.1). This is the result of the congestion window of MPTCP growing significantly faster during the Slow Start phase at the beginning of the simulation, as illustrated in Figure 25(section 6.2.5). This causes congestion on the shared paths, when multiple TCP connections and multiple MPTCP connections are simulated, thus, terminating the Slow Start phase of TCP. This results in a smaller TCP congestion window. The total congestion window of MPTCP reduces significantly in size over the course of the simulation, therefore, creating significant throughput fluctuations. This accounts for the big confidence intervals. TCP and MPTCP perform very similarly for the same amount of subflows as TCP connections. At high packet error ratios, between 10^{-2} and 10^{-3}, fifteen MPTCP subflows can transfer slightly more data than the coexisting fifteen TCP connections and from a PER of 10^{-4} to 10^{-6} fifteen TCP connections perform slightly better than fifteen MPTCP subflows. At low PERs, five TCP connections per terminal can transfer more data than when they are coexisting with five MPTCP subflows per connection. Nonetheless, these throughput differences are minimal, they amount to 50 kbps or less.

Figure 22: Throughput for different PERs by using the ON/OFF Error Model in mixed scenarios.



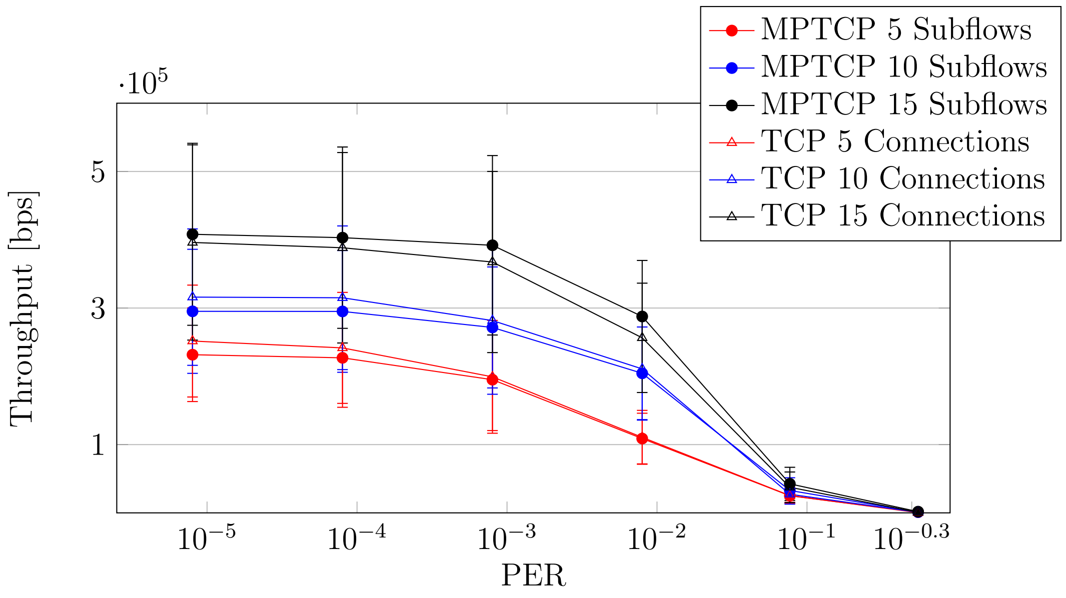
In Figure 22 we plot the throughput of the same mixed scenarios as in Figure 21 but with the use of the ON/OFF Error Model (section 5.2.2). We plot the results as a function of the packet error ratio during the OFF phase. At high packet error ratios, from PERs of $5 \cdot 10^{-1}$ to 10^{-2}, separate TCP connections can transfer as much data as the same number of MPTCP subflows. At lower PERs, this is no longer the case, ten and five TCP connections per TCP host and five MPTCP subflows per MPTCP host perform very similarly, but fifteen MPTCP subflows per MPTCP host can transfer about 100 kbps more than ten MPTCP subflows per MPTCP host. In addition, fifteen TCP connections per TCP host can transfer almost 100 kbps more than fifteen MPTCP subflows per MPTCP host. The aggressive growth of the MPTCP congestion window causes congestion on the shared paths when multiple subflows and TCP connections are simulated. This causes the Slow Start phase of TCP to be prematurely terminated, therefore, decreasing its throughput. Nonetheless, the congestion window of MPTCP decreases significantly over the course of the simulation, thus, allowing the congestion window of TCP to grow after the initial simulation phase. This degrades the overall throughput of TCP. Nonetheless, the congestion window of TCP is bigger than the congestion window of MPTCP over most of the course of the simulation. The congestion window is discussed in section 6.2.5 and plotted in Figure 25.

Figure 23: Throughput for different mean ON phase durations by using the ON/OFF Error Model in mixed scenarios.



In Figure 23 we plot the throughput of the same mixed scenarios as in Figures 21 and 22 with the use of the ON/OFF Error Model (section 5.2.2). We plot the throughput as a function of the mean ON phase duration, analogous to Figures 11,15 and 19. As the mean ON phase duration is increased we expect the throughput to increase, because the overall packet error ratio decreases, as previously mentioned for Figures 11 and 15. This throughput gain is measured, but its maximum is 500 kbps. MPTCP and the same amount of TCP connections as MPTCP subflows, perform very similarly throughout the mean ON phase duration modulation. As previously mentioned, in terms of overall PER, Figure 23 represents a small modulation around the PER $= 10^{-2}$ region in Figure 22. At such a high packet error ratio, the total TCP and MPTCP congestion windows stays small over the course of the simulation. Packet retransmissions occur at a similar frequency for both protocols, therefore keeping the total congestion windows of both protocols at almost the same size. The congestion window is further discussed in section 6.2.5.

Figure 24: Throughput for different PERs by using the ON/OFF plus BER Error Model in mixed scenarios.



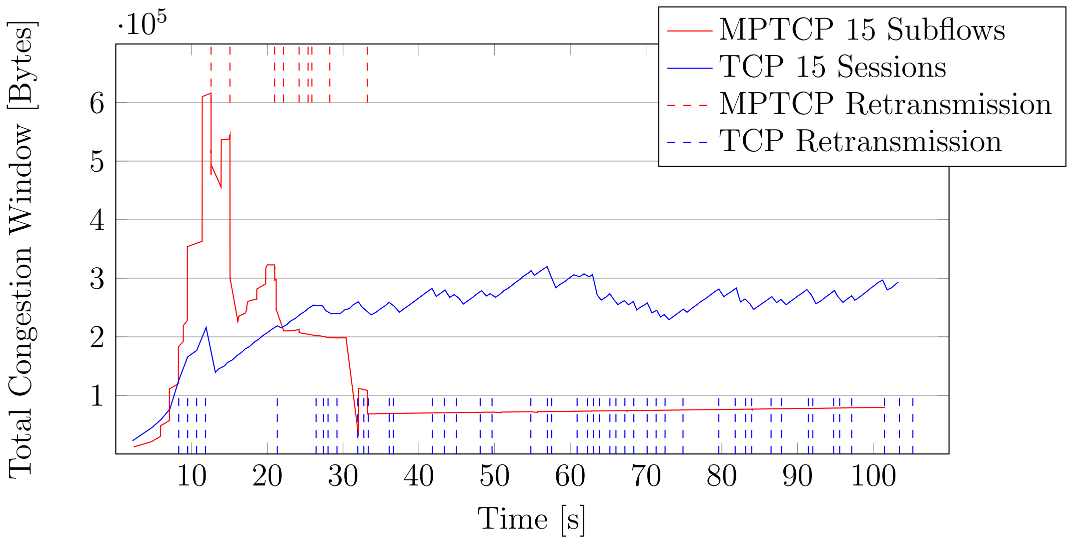
In Figure 24 we plot the throughput for the same three scenarios as in Figures 21, 22 and 23, but using the BER plus ON/OFF Error Model (section 5.2.3). We plot the results as a function of the packet error ratio during the ON phase. At each simulated PER, the coexisting MPTCP and TCP connections perform very similarly with this error model. The throughput of TCP does not improve linearly with the number of connections, rather it yields a similar throughput increase as the MPTCP-only simulations displayed in Figure 16. At high packet error ratios, around PER $= 10^{-2}$, ten MPTCP subflows and ten TCP connections can transfer two times as much data as five MPTCP subflows and five TCP connections, fifteen MPTCP subflows and fifteen TCP connections can transfer about three times as much data as five MPTCP subflows and five TCP connections. At lower packet error ratios, fifteen MPTCP subflows and fifteen TCP connections can transfer about two times the data five MPTCP subflows and five separate TCP connections can transfer. TCP performs worse than in the TCP-only scenarios, illustrated in section 12, because the aggressive MPTCP congestion window opening congests the shared paths. The Slow Start phase of TCP is interrupted because of the congestion and the opening of the congestion is significantly delayed until the MPTCP subflows enter the Congestion Avoidance phase and close their congestion window as a result. The performance of TCP is limited by the aggressive growth of MPTCP during the initial phase. MPTCP grows aggressively during the Slow Start phase. The faster growth of the MPTCP congestion window congests the paths that MPTCP and TCP share. This leads the TCP Slow Start phase to prematurely stop, thus, reducing the initial TCP congestion window. The congestion window is further discussed in section 6.2.5.

It is very important to notice that in Figures 21-24, MPTCP and TCP coexisting in the same network do not yield radically different throughputs, when they can take advantage of the same number of paths. This demonstrates that, when considering the whole duration of the simulation, MPTCP subflows and single-path TCP connections are fair to each other at the shared paths. Therefore, we can conclude that with the use of an On/Off application (section 5.3.2), TCP connections and MPTCP subflows take up an equal amount of bandwidth at the shared paths.

6.2.5 Congestion Window

The throughput of TCP and MPTCP connections is strictly dependent on the size of their congestions windows, therefore it is paramount to observe their size over the course of a simulation to understand all previous throughput plots.

Figure 25: Total congestion window size of MPTCP and TCP as a function of the simulation time by using the On/Off Application and the BER Error Model at BER=10^{-8}.



In Figure 25 we plot the total size of the congestion window at a single terminal, using the BER Error Model (section 5.2.1), as a function of the simulation time. The total congestion window of MPTCP is the sum of the congestion window of all subflows belonging to one single MPTCP connection, in this case of fifteen subflows, and the total congestion window of TCP is the sum of the congestion windows of each separate TCP connection at one terminal, in this case of fifteen TCP connections. The data comes directly from the simulation plotted in Figures 9 and 13 at a bit error ratio of 10^{-8}. The dashed lines on the bottom represent the TCP retransmissions and the ones on the top of the graph represent the MPTCP retransmissions. One dashed line may represent more than one retransmit, e.g., in Figure 25 MPTCP tries to retransmit a packet three times at around 12 seconds and four times at around 15 seconds, closing the congestion window significantly. As soon as the first retransmissions are executed, the subflows enter the Congestion Avoidance phase one by one, until around 32 seconds (figure 25), when all the subflows are in the Congestion Avoidance phase. The total MPTCP congestion window after 32 seconds (Figure 25) is dictated by the Linked Increases congestion control algorithm, discussed in section 4.3.3.1. For MPTCP the recovery is heavily impeded by the very small value of α (section 4.3.3.1) caused by the very high RTT value, while the independent NewReno TCP connection can recover at a faster rate, which is also slowed by the very high RTT, but faster than the Linked Increases congestion control algorithm. The pattern in Figure 25 is very similar for all scenarios using an On/Off application (section 5.3.2). In the case of mixed scenarios (section 6.2.4), Figure 25 does not apply completely, but Figure 42 (section 6.3.5) does. At a high MPTCP subflow number and separate TCP connection number, the likelihood of an MPTCP subflow sharing a chunk of their routing path with a TCP connection is high. MPTCP performs better initially during the Slow Start phase, as displayed in Figures 25 and 42. Therefore, the bandwidth available to TCP is smaller, thus, TCP performs worse initially than in the TCP-only simulations illustrated in section 6.2.1. Moreover, the frequency of ACKs sent by TCP is higher than that of MPTCP, because MPTCP uses Selective Acknowledgement packets. Therefore, since the backward channel, the channel the receiving host uses to send acknowledgment packets, is more congested, the inevitable congestion window decay, illustrated in Figures 25 and 42, that affects MPTCP is slightly delayed. This happens because it takes slightly more time for three duplicate ACKs to arrive at the sending terminal and induce a retransmission and a consequent congestion window reduction. This is why in some cases, as in Figures 21 and 24, MPTCP slightly outperforms TCP. Nonetheless this remains a temporary gain in throughput, because, as depicted in Figures 25 and 42, the closing of the congestion window is inevitable at longer durations. Moreover the large confidence intervals shown in Figures 21, 22, 23 and 24 are due exactly to this "closing" behavior, because the closing of the congestion window causes the measured throughput to be radically different between the simulations that last 30 seconds and the ones that last 480 seconds.

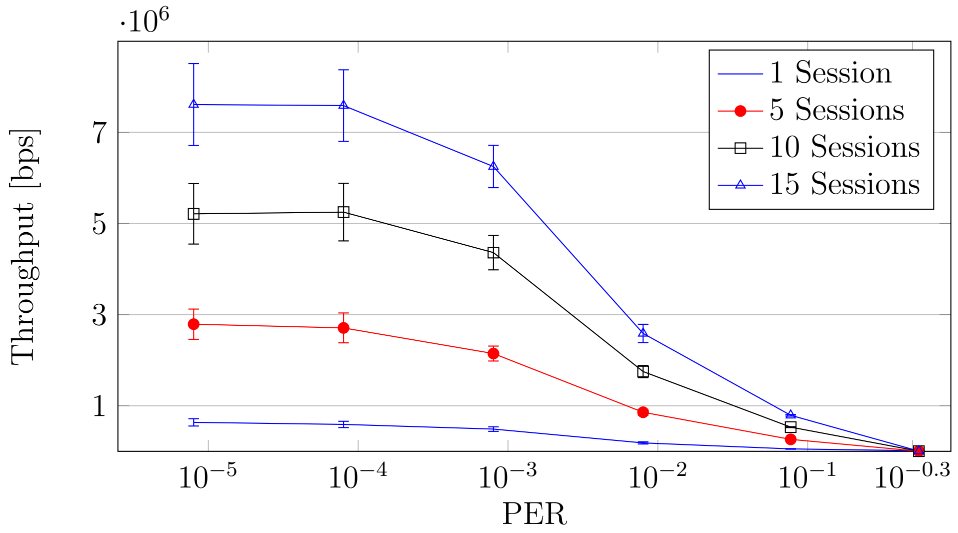
6.3 Bulk Send Application Results

In this section we plot the results of the simulations that make use of Bulk Send applications (section 5.3.1) to generate the data stream. The Bulk Send application is an application that continuously sends data trying to fill the available TCP send window. A further analysis, relating the congestion window to the throughput plotted in the following Figures, can be found in section 6.3.5.

6.3.1 TCP

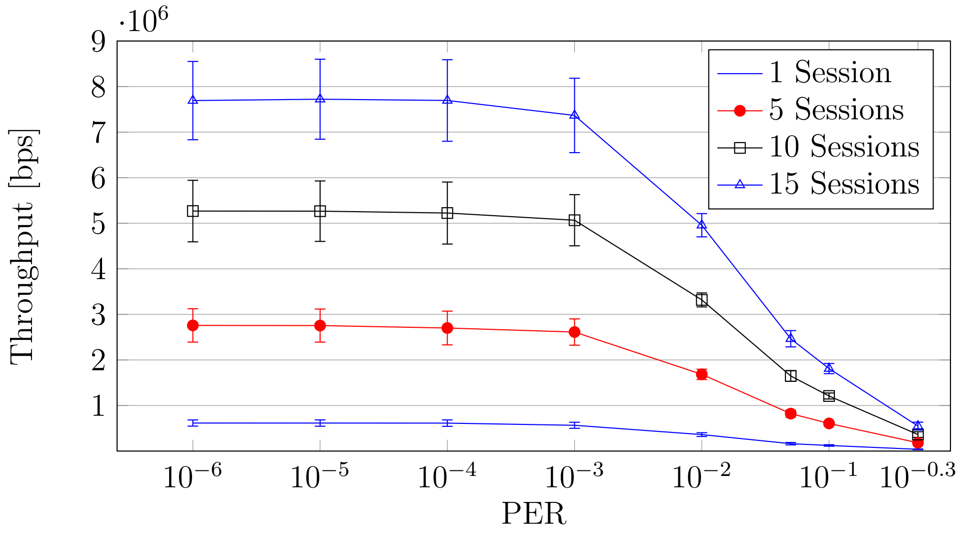
The setup used to obtain the following data is the same as that used in section 6.2.1, but instead of an On/Off application (section 5.3.2), we use a Bulk Send application (section 5.3.1).

Figure 26: Throughput for different PERs by using the BER Error Model with 1, 5, 10 and 15 TCP sessions per host.



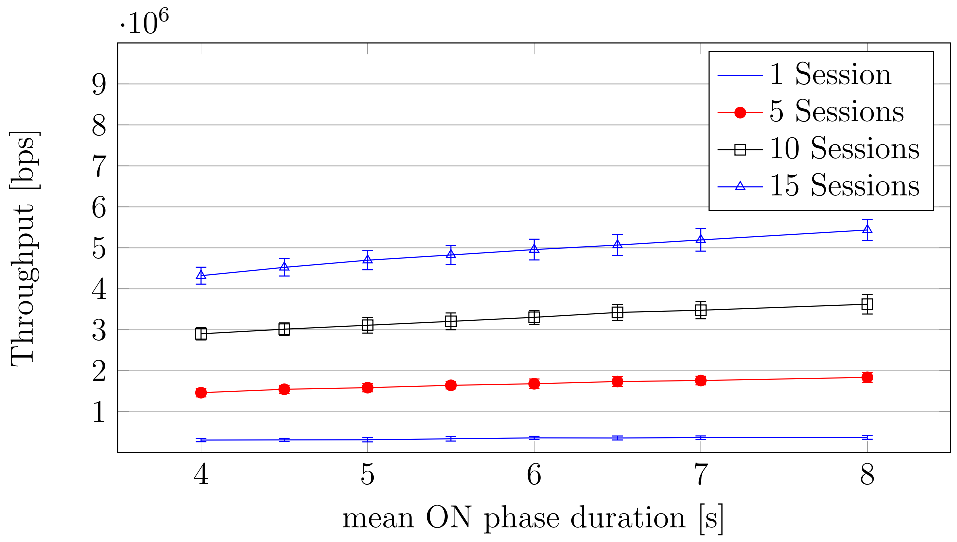
In Figure 26 we plot the throughput of TCP with the use of the BER Error Model (section 5.2.1) as a function of the packet error ratio, analogous to Figure 9. The throughput in this case is very similar to that in Figure 9. The throughput grows linearly with the amount of TCP connections per terminal. Five TCP connections can transfer five times more data than a single connection, ten TCP connections can transfer ten times more and fifteen TCP connections fifteen times more data than a single TCP connection per terminal.

Figure 27: Throughput for different PERs by using the ON/OFF Error Model with 1, 5, 10 and 15 TCP sessions per host.



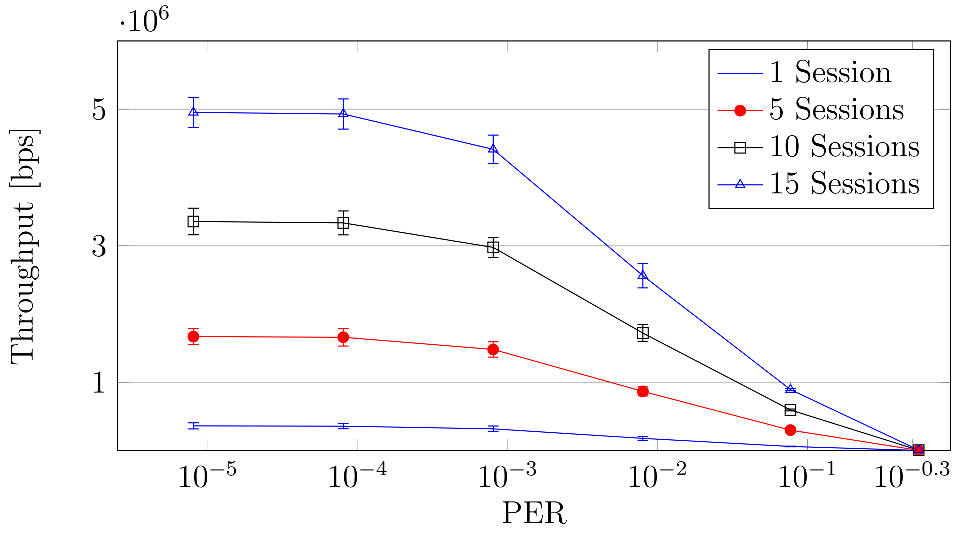
In Figure 27 we plot the throughput of TCP with the use of the ON/OFF Error Model (section 5.2.2) as a function of the packet error ratio during the OFF phase, analogous to Figure 10. The plotted data is still very similar to the data obtained with the On/Off application (section 5.3.2) illustrated in Figure 10. In fact, the data that TCP can transfer grows linearly with the amount of TCP connections per terminal for each of the simulated packet error ratios, similar to Figures 10 and 26.

Figure 28: Throughput for different mean ON phase durations by using the ON/OFF Error Model with 1, 5, 10 and 15 TCP sessions per host.



In Figure 28 we plot the throughput of TCP with the use of the ON/OFF Error Model (section 5.2.2) as a function of the mean ON duration, analogous to Figure 11. The throughput is very similar to Figure 11. The increase of the mean ON phase duration leads to a throughput increase, because the overall packet error ratio decreases, similar to Figure 11. The relationship between the amount of TCP connections per host and the overall throughput is linear, similar to Figures 26 and 27.

Figure 29: Throughput for different PERs by using the ON/OFF plus BER Error Model with 1, 5, 10 and 15 TCP sessions per host.

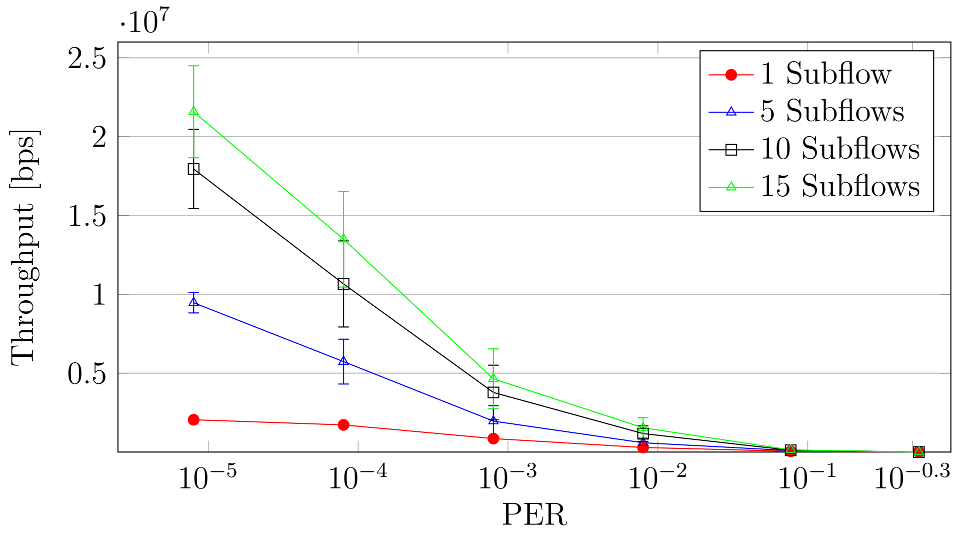


In Figure 29 we plot the throughput of TCP with the use of the ON/OFF plus BER Error Model (section 5.2.3) as a function of the ON phase packet error ratio, similar to Figure 12. The throughput is expected to be lower than with the use of the ON/OFF Error Model (section 5.2.2), because the overall packet error ratio is higher, since packet errors can occur in the ON phase as well. The plotted data is very similar to the data obtained with the use of the On/Off application illustrated in Figure 12. Nonetheless, with the same number of TCP connections and packet error ratio, the throughput is slightly better than with an On/Off application. This is simply due to fact that a Bulk Send application tries to push more data and more often through the TCP sockets than an On/Off application.

6.3.2 MPTCP

The setup used to obtain the four following Figures is the same as the setup used in section 6.2.2, but with the use of a Bulk Send application (section 5.3.1).

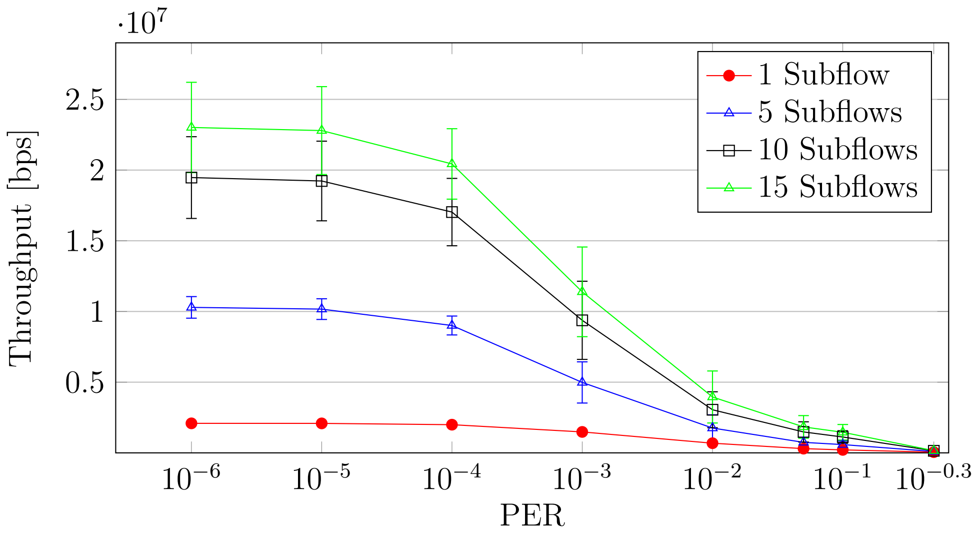
Figure 30: Throughput for different PERs by using the BER Error Model with 1, 5, 10 and 15 MPTCP subflows per host.



In Figure 30 we plot the throughput of MPTCP with the use of the BER Error Model (section 5.2.1) as a function of the packet error ratio, analogous to Figure 13. At PERs lower than 10^{-1}, adding subflows to the MPTCP connections increases the throughput significantly. At PER $= 10^{-5}$, MPTCP using five subflows transfers about three times the data that a single subflow can transfer, ten subflows transfer about nine times more data than a single subflow and fifteen subflows transfer eleven times more than a single subflow transfers. The throughput we achieve with a Bulk Send application is significantly higher than the results obtained with the use of a On/Off application illustrated in Figure 13. MPTCP with one subflow transfers approximately the same amount of data with both an On/Off application (section 5.3.2) and a Bulk Send application (section 5.3.1). Nonetheless, at PERs lower than 10^{-3}, in comparison with Figure 13, five subflows with a Bulk Send application transfer up to five times more data compared with an

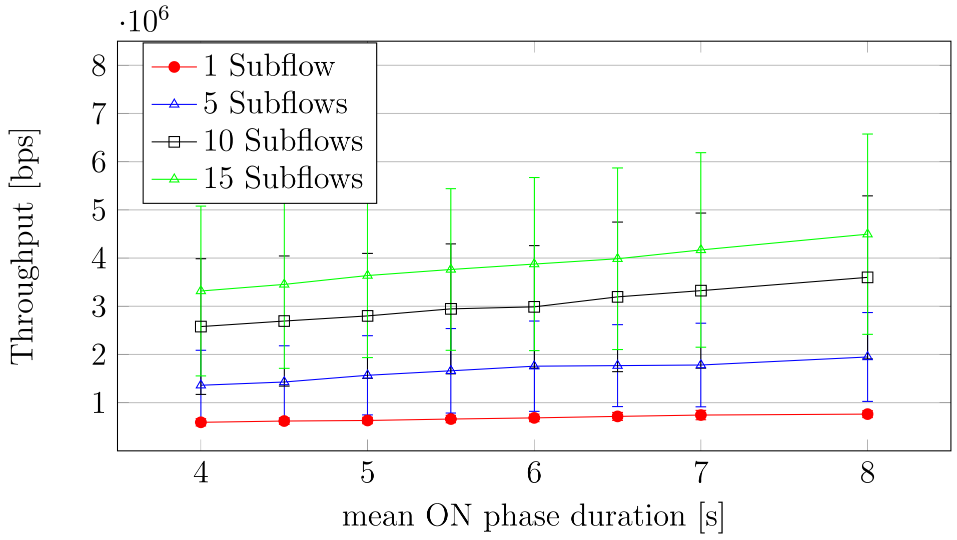
On/Off application, ten subflows up to eight times more and fifteen subflows up to eight times more data.

Figure 31: Throughput for different PERs by using the ON/OFF Error Model with 1, 5, 10 and 15 MPTCP subflows per host.



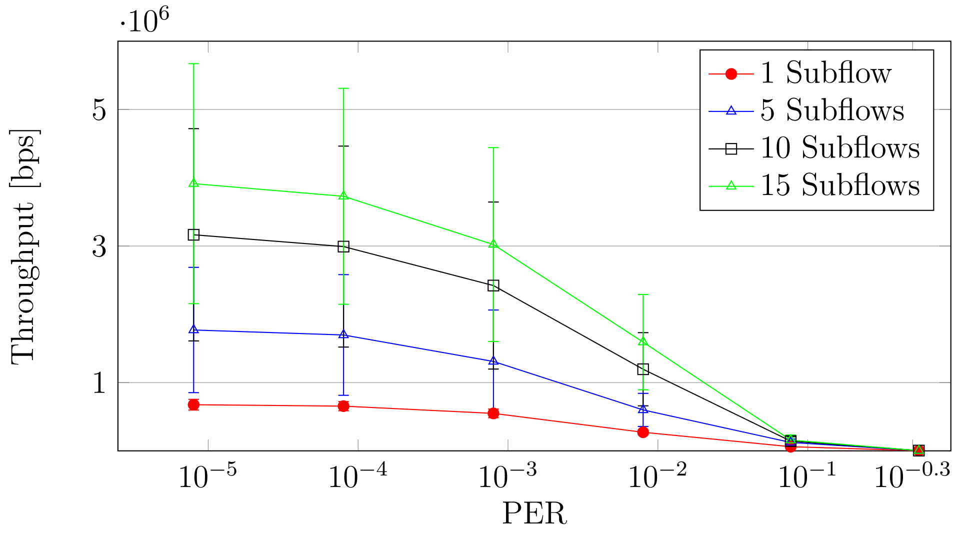
In Figure 31 we plot the throughput of MPTCP with the use of the ON/OFF Error Model (section 5.2.2) as a function of the OFF phase packet error ratio, analogous to Figure 14. The throughput we achieve with five subflows per MPTCP connections is roughly five times the throughput achieved with one single subflow, with ten subflows about eight times more and with fifteen subflows about eleven times more data can be transfered than with a single MPTCP subflow. The results obtained with a Bulk Send application (section 5.3.1) and multiple MPTCP subflows are significantly better than those obtained with an On/Off application (Figure 14). In fact, with fifteen subflows we achieve a throughput nine times higher with a Bulk Send application than with an On/Off application, with ten subflows the throughput is nine times higher and, with five subflows, is four times higher than the throughput obtained with an On/Off application. One single subflow does not show any difference between the two different applications. The reason of this throughput difference is due to the behavior of the congestion window, which is illustrated in Figure 42.

Figure 32: Throughput for different mean ON phase durations by using the ON/OFF Error Model with 1, 5, 10 and 15 MPTCP subflows per host.



In Figure 32 we plot the throughput of MPTCP with the use of the ON/OFF Error Model (section 5.2.2) as a function of the mean ON duration, analogous to Figure 15. As the mean ON phase duration increases, the throughput also increases for each MPTCP subflow setup. The throughput increases when multiple subflows are added to the MPTCP connections. Five subflows transfer approximately two times more data than a single subflow, ten subflows three times more data and fifteen subflows about six times more data than a single MPTCP subflow. The results achieved with a Bulk Send (section 5.3.1) application are, in this case, higher than those achieved with an On/Off application (section 5.3.2) illustrated in Figure 15.

Figure 33: Throughput for different PERs by using the ON/OFF plus BER Error Model with 1, 5, 10 and 15 MPTCP subflows per host.

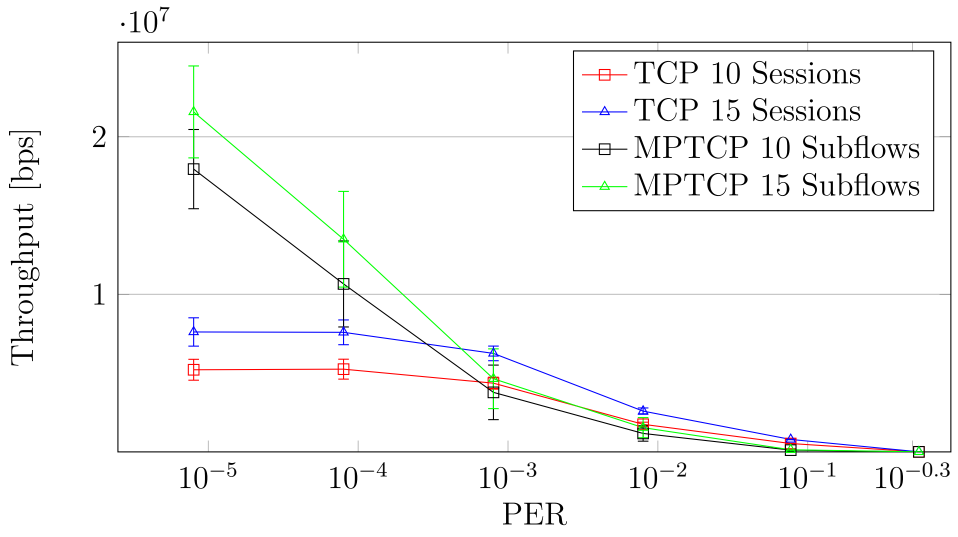


In Figure 33 we plot the throughput of MPTCP with the use of the BER plus ON/OFF Error Model (section 5.2.3) as a function of ON phase the packet error rate, analogous to Figure 16. The throughput increases with the number of subflows. Five subflows per connection yield about 2.5 times the throughput of one single subflow per connection, ten subflows about four times and fifteen subflows can transfer approximately five times the data that one single flow per MPTCP connection can transfer. Similar to the previous three Figures, the use of a Bulk Send application (section 5.3.1) with multiple MPTCP subflows yields a higher throughput than with an On/Off application (section 5.3.2). In fact, compared to Figure 16, using a Bulk Send application, five, ten and fifteen MPTCP subflows per connection can transfer about two times more data than with an On/Off application.

6.3.3 TCP MPTCP Comparison

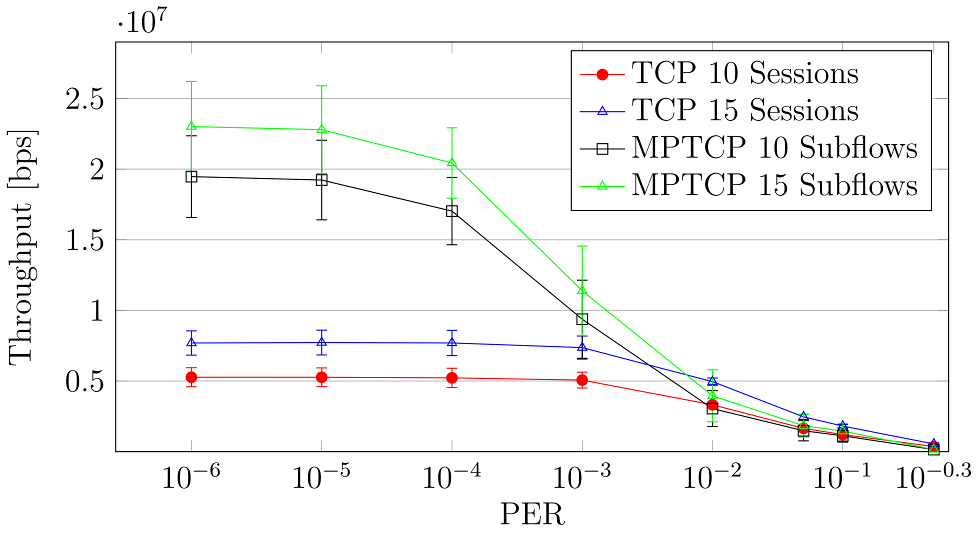
In this section we compare the results from sections 6.3.2 and 6.3.1 with the exact same method as in section 6.2.3.

Figure 34: Throughput comparison for different PERs by using the BER Error Model.



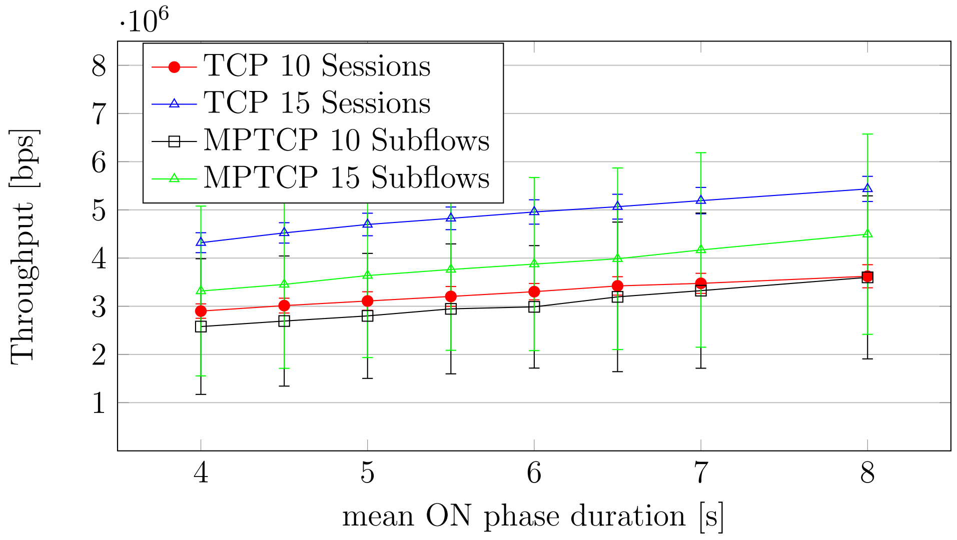
In Figure 34 we compare the results from Figures 26 and 30. At high packet error ratios, from $5 \cdot 10^{-1}$ to 10^{-3}, ten and fifteen TCP connections per terminal perform slightly better than ten and fifteen MPTCP subflows per terminal. At lower packet error ratios, MPTCP performs significantly better than TCP. At PER $= 10^{-4}$ the throughput gain by adopting an MPTCP connection with the same amount of subflows as TCP connections per host can yield to a throughput increase of about 500 kbps. The reason for this significant throughput difference is further explained in section 6.3.5.

Figure 35: Throughput comparison for different PERs by using the ON/OFF Error Model.



In Figure 35 we compare the data from Figures 27 and 31. Similar to the previous Figure, at high packet error ratios down to 10^{-2} TCP performs slightly better than MPTCP. At packet error ratios, lower than 10^{-2}, MPTCP performs much better even compared to a higher number of TCP connections than MPTCP subflows per host. In fact, at low PERs, MPTCP can transfer up to four times as much data as TCP with the same amount of TCP connections per host. Moreover, at PERs lower than 10^{-4}, even ten MPTCP subflows can transfer three times the data fifteen separate TCP connections can transfer. The reason for this throughput difference is explained in section 6.3.5.

Figure 36: Throughput comparison for different mean ON phase durations by using the ON/OFF Error Model.

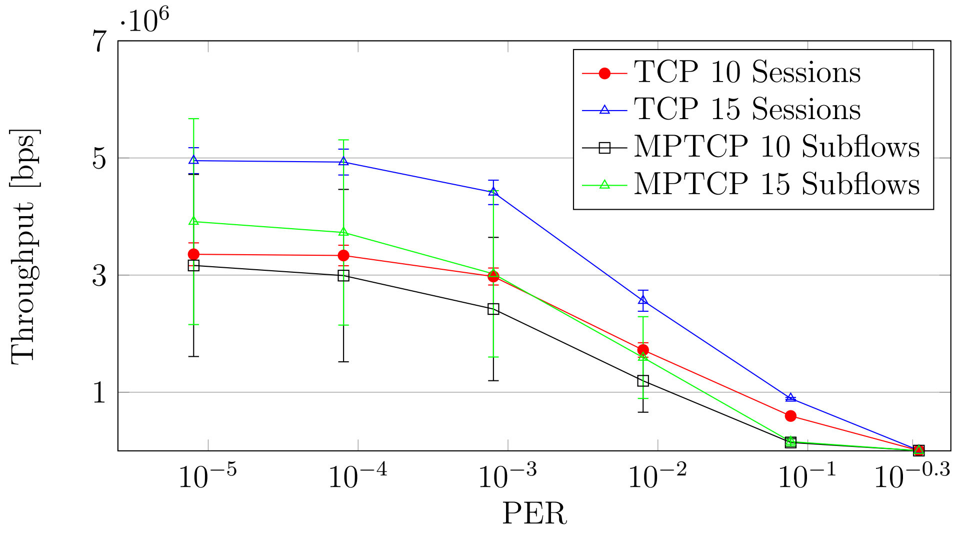


In Figure 36 we compare the results from Figures 28 and 32. The throughput increases as the mean ON phase duration increases. TCP performs better than MPTCP for the same amount of separate connections as MPTCP subflows. Ten TCP connections can transfer about 20 kbps more data than ten MPTCP subflows at a mean ON phase duration of 4 seconds. This difference gets smaller as the mean ON phase duration increases. Fifteen separate TCP connections can transfer about 1 Mbps more than fifteen MPTCP subflows throughout the whole mean ON phase duration modulation. This is because, at such an high PER, the MPTCP subflows enter the Congestion Avoidance phase earlier. The Linked Increases algorithm (section 4.3.3.1), which dictates the opening of the congestion window during the Congestion Avoidance phase, allows for a much more slow opening of the congestion window than the NewReno Congestion Control algorithm used by TCP. Therefore, this allows TCP to send more data than MPTCP.

In Figure 36, the mean OFF phase duration is kept constant at 1 second and the OFF phase packet error ratio is kept constant at 10^{-2}. Therefore, the data points at a mean ON phase duration of 6 seconds correspond to the data points in Figure 35 at a packet error ratio of 10^{-2}. In terms of PER, the mean

ON phase duration modulation from 4 to 8 seconds represents a small PER modulation around the region of PER = 10^{-1} in Figure 35.

Figure 37: Throughput comparison for different PERs by using the ON/OFF plus BER Error Model.

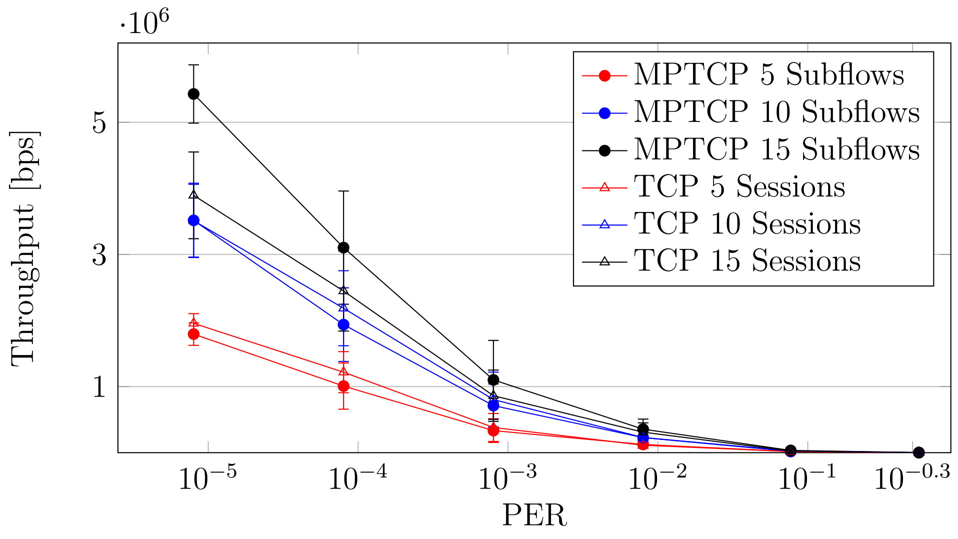


In Figure 37 we compare the results from Figures 29 and 33. Unlike in Figures 17 and 18, MPTCP subflows can transfer less data than the same number of TCP connections as MPTCP subflows per terminal with this error model. At high packet error ratios, even ten TCP connections per terminal transfer more data than fifteen MPTCP subflows per terminal. At PER = 10^{-2}, the throughput difference between ten TCP connections and ten MPTCP subflows amounts to approximately 50 kbps and as the packet error ratio decreases, the throughput difference also decreases until it is almost zero at PER = 10^{-5}. Moreover, the throughput yielded by fifteen TCP connections is about 60 kbps bigger than the throughput yielded by fifteen MPTCP subflows. In Figure 37 the horizontal axis represents the ON phase packet error ratio. The OFF phase packet error ratio is constant at 10^{-2} and so are the ON phase duration and the OFF phase duration at 6 and 1 second, respectively. Therefore, the entries on the horizontal axis that are lower than 10^{-2} represent a higher overall packet error ratio. Because of this, the second half of Figure 34, starting from a PER of 10^{-3} to $5 \cdot 10^{-1}$, corresponds to the horizontal axis of Figure 37 in terms of overall PER. Therefore, the reason TCP performs better than MPTCP, is because we are simulating the higher end of packet error ratios and it is, thus, consistent with Figures 34 and 35.

6.3.4 Mixed Scenario

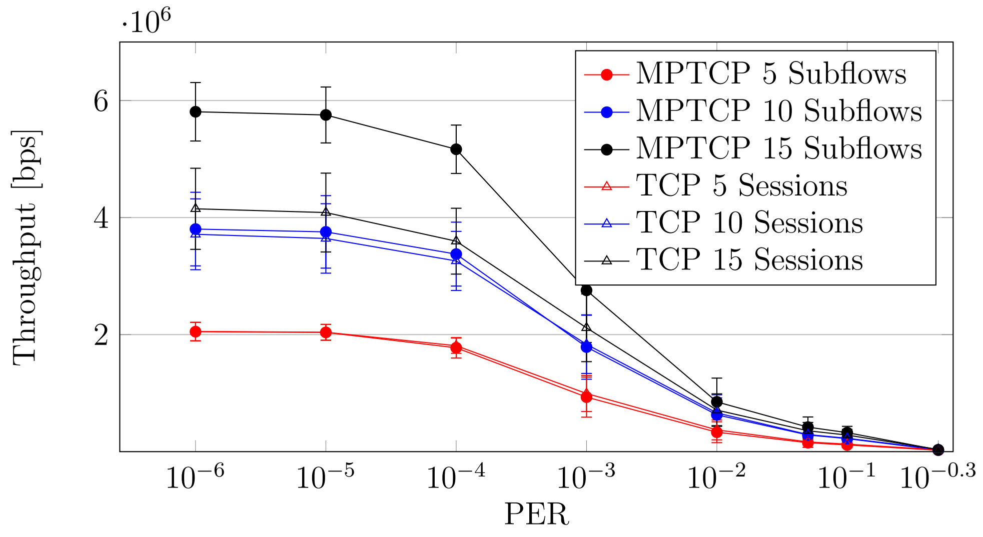
The mixed scenario is presented in section 5.6, the data is plotted with the same method as in section 6.2.4.

Figure 38: Throughput for different PERs by using the BER Error Model in mixed scenarios.



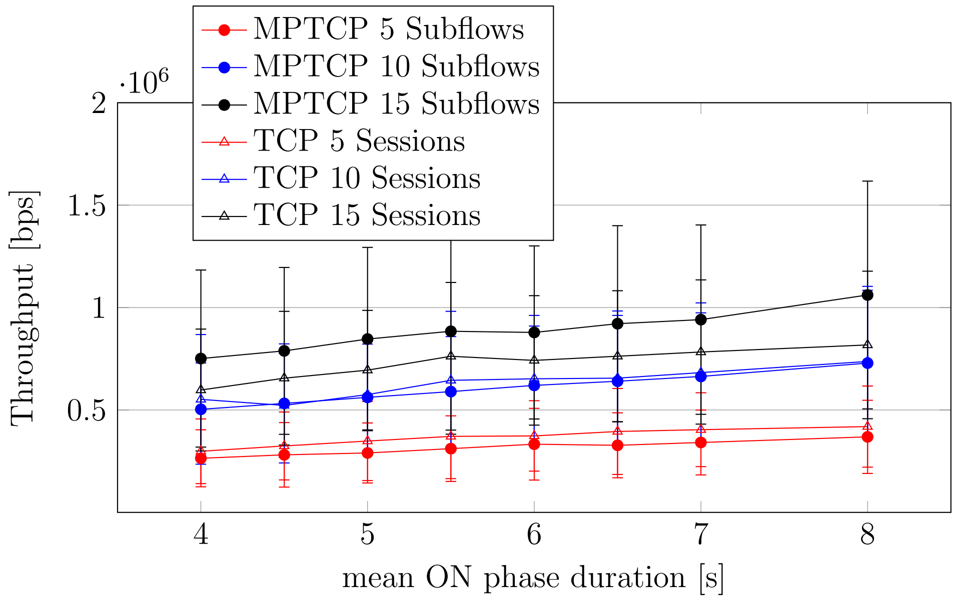
In Figure 38 we plot the same three scenarios as in Figure 21 as a function of the packet error ratio, but by using a Bulk Send application (section 5.3.1). MPTCP performs very similarly to TCP. Five MPTCP subflows can transfer almost as much data as five TCP connections and ten MPTCP subflows can transfer as much data as ten TCP connections. Nonetheless, fifteen MPTCP connections transfer more data than fifteen TCP connections. At a packet error ratio of 10^{-5}, fifteen MPTCP subflows can transfer about 1 Mbps more than fifteen separate TCP connections.

Figure 39: Throughput for different PERs by using the ON/OFF Error Model in mixed scenarios.



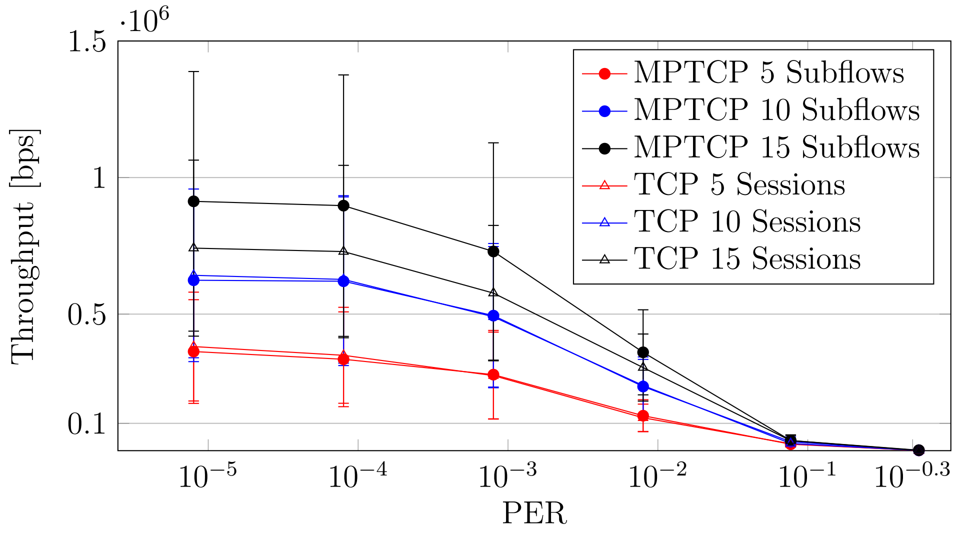
In Figure 39 we plot the results of the same three scenarios as in Figure 22 as a function of the OFF phase packet error ratio, but with the use of a Bulk Send application (section 5.3.1). Five MPTCP subflows can transfer as much data as five TCP connections and ten subflows can transfer as much data as ten TCP connections. Fifteen MPTCP subflows can transfer approximately 1.5 times more data than fifteen TCP connections can transfer. Moreover, MPTCP can transfer 1.3 times more data than it can transfer in the MPTCP-only scenario illustrated in Figure 31 than if it coexist with two TCP connections (Figure 39). This is because the total MPTCP congestion window grows aggressively during the Slow Start phase. Because of this, when multiple subflows are used, MPTCP takes advantage of more bandwidth than TCP. This creates congestion in the shared paths, therefore, not allowing TCP to further open its congestion window.

Figure 40: Throughput for different mean ON phase durations by using the ON/OFF Error Model in mixed scenarios.



In Figure 40 we plot the same three scenarios as in Figure 24, but with the use of a Bulk Send application (section 5.3.1). As the mean ON phase duration increases, the throughput also increases, similar to Figure 23. TCP and MPTCP perform very similarly. Five MPTCP subflows yield a throughput almost as high as five separate TCP connections and ten MPTCP subflows transfer as much data as ten separate TCP connections. Nonetheless, fifteen MPTCP subflows can transfer up to 20 kbps more than fifteen TCP connections.

Figure 41: Throughput for different PERs by using the ON/OFF plus BER Error Model in mixed scenarios.

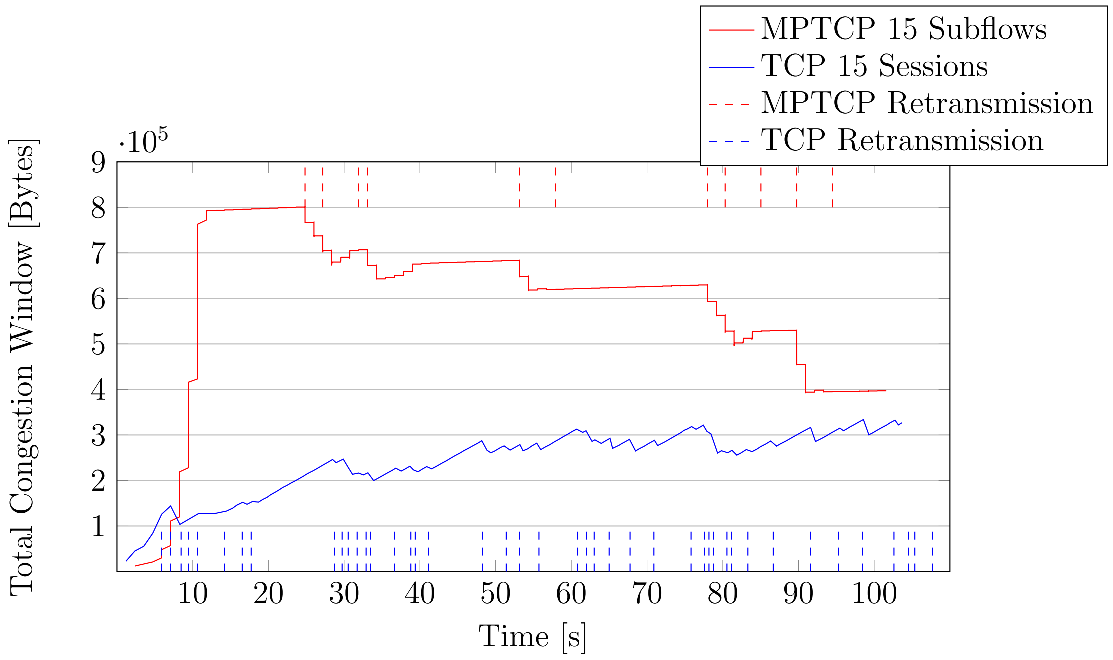


In Figure 41 we plot the same three scenarios as in Figure 24 as a function of the ON phase packet error ratio, but with the use of a Bulk Send application (section 5.3.1). As in the previous three Figures, five MPTCP subflows can transfer as much data as five separate TCP connections and ten MPTCP subflows can transfer as much data as ten separate TCP connections can transfer. Fifteen MPTCP subflows transfer approximately 20 kbps more than fifteen separate TCP connections at low packet error ratios. From a PER of 10^3 to 10^{-1} this throughput difference shrinks from 20 kbps at 10^{-3} to almost zero at PER = 10^{-1}.

It is important to notice that TCP and MPTCP share the bandwidth fairly for scenarios with ten MPTCP subflows and ten TCP connections per terminal at the same time, and for scenarios with five MPTCP subflows and five separate TCP connections. Nonetheless, fifteen MPTCP subflows and fifteen TCP connections are not as fair to each other, if they are sharing chunks of their paths. In fact, in this case, MPTCP takes advantage of more bandwidth than TCP with each simulated error model, as illustrated in the four previous Figures.

6.3.5 Congestion Window

Figure 42: Total congestion window size of MPTCP and TCP as a function of the simulation time by using the Bulk Send Application and the BER Error Model at BER=10^{-8}.



In Figure 42 we plot the total congestion window of both TCP and MPTCP, with the use of the BER Error Model (section 5.2.1) at a PER of 10^{-5}, as a function of the simulation time. We use the same method as in Figure 25, but the simulations in this case use a Bulk Send application (section 5.3.1). By using a Bulk Send application, the whole network is much more congested than by using an On/Off application (section 5.3.2). The MPTCP subflows congestion windows grow at a rapid rate and even reach the Slow Start threshold. Because of the congested paths, it takes more time to receive three duplicate ACKs. This results in a bigger window initially, but for longer durations the closing of the window is imminent, since as soon as all subflows are in Congestion Avoidance phase the window opening is dictated by the Linked Congestion Control (section 4.3.3.1) algorithm, which only allows for a small opening at each ACK, as previously discussed in section 6.2.5. The initially bigger congestion window of MPTCP is the reason of the bigger

MPTCP throughput displayed in section 6.3.3. The aggressive congestion window opening resulting in a gradually lower congestion window is the reason of large confidence intervals. The simulations are executed with a duration that ranges from 30 to 480 seconds and in scenarios, where the total congestion window can aggressively open, to then taper down significantly, the total throughput measured at 30s is usually much higher than the one recorded at 480s.

In the case of mixed scenarios (section 6.3.4), more paths reach congestion, because the Bulk Send application sends data more frequently than the On/Off application. The aggressive growth of the congestion window, illustrated in Figure 42, congests the paths where TCP and MPTCP coexist, thus, interrupting the Slow Start phase of TCP. Using a Bulk Send application, the slightly better performance of multiple MPTCP subflows is caused by the fact that the total MPTCP congestion window keeps its size, after the Slow Start phase, longer than by using an On/Off application. In this case, more packets are present inside the network (more bytes in flight) and more paths are congested, thus, the three duplicate ACKs are often dropped during transmission and take more time to arrive at the host. Moreover, TCP sends more packets in the backwards channel, further slowing the arrival of the three duplicate MPTCP ACKs. This allows the MPTCP congestion window to stay open longer than by using an On/Off application. This improves the throughput of MPTCP.

7 Conclusion

We identified the applications of satellite connections and the problems that deteriorate their performance. We identified the possible alternatives to mitigate most common problems that TCP encounters in satellite networks. We built a simulation scenario making use of a plausible satellite network topology. We implemented a custom error model to take into account the possible signal degradation typical to satellite connections. We identified the two types of data streams to simulate a real world application and we simulated MPTCP and TCP connections separately and in a mixed scenario.
With the use an On/Off application, MPTCP-only scenarios yield a much lower throughput than multiple TCP connections in a TCP-only scenario. In fact, in those cases, if comparing the throughput yielded by the same number of TCP connections as MPTCP subflows, TCP can often transfer more than twice as much data as MPTCP can. In mixed scenarios, in which TCP and MPTCP connections are present at the same time, TCP and MPTCP can transfer the same amount of data over the satellite network, confirming that fairness between the two protocols is guaranteed when using On/Off applications.
With the use of a Bulk Send application in MPTCP-only scenarios, at low packet error ratios, MPTCP can transfer up to three times more data than TCP. Nonetheless, in the mixed scenarios, MPTCP with five subflows can transfer as much data as five separate TCP connections and ten MPTCP subflows can transfer as much data as ten separate TCP connections. Thus, with these setups, fairness between MPTCP and TCP is guaranteed with the use of a Bulk Send application. Always in the mixed scenarios, fifteen MPTCP subflows perform better than fifteen separate TCP connections. In fact, MPTCP can transfer up to almost 2Mbps more than TCP. With a Bulk Send application, the lower MPTCP throughput is caused by the congestion control algorithm. The congestion control algorithm allows for a minimal congestion window opening, because the opening is inversely proportional to the round trip time, which in the case of satellite networks, is very high at about 600

seconds. Therefore, we recognize that dedicated alternative congestion control algorithms should be implemented to improve the MPTCP throughput, when the connection is subjected to high round trip times.

References

[1] Giovanni Giambene, Doanh Kim Luong, Van Anh Le and Muhammad Muhammad. *Network coding and MPTCP in satellite networks.* 2016 8th Advanced Satellite Multimedia Systems Conference and the 14th Signal Processing for Space Communications Workshop ASMS/SPSC, IEEE, 2016.

[2] Tatsuyuki Hanada, Kiyotaka Fujisaki, Mitsuo Tateiba. *Average Bit Error Rate for Satellite Downlink Communications in Ka-band under Atmospheric Turbulence Given by Gaussian Model*, 2009 Asia Pacific Microwave Conference, IEEE, pp. 1092-1095, 2009.

[3] Donekeo Lakanchanh, Pongputhai Udomareyasap, Nipha Leelaruji, Narong Hemmakorn. *Propagation Effect by Rain in Ku and Ka band Satellite Communication System*, 2006 International Symposium on Communications and Information Technologies, IEEE, pp. 970-973, 2006.

[4] P.W. Marshall, C.J. Dale, M.A. Carts, K.A. LaBel. *Particle-induced bit errors in high performance fiber optic data links for satellite data management*, IEEE Transactions on Nuclear Science, IEEE, vol. 41, pp. 1958-1965, 1994.

[5] Lang Tanja, Floreani Daniel, Dadej Arek. *TCP Throughput over Links with High Bit Error Rate*, Converged Networking. INTERWORKING 2002. IFIP — The International Federation for Information Processing, Springer, vol. 119, pp. 223-234, 2002.

[6] M. Luglio, M. Y. Sanadidi, M. Gerla, J. Stepanek. *On-Board Satellite "Split TCP" Proxy*, IEEE Journal on Selected Areas in Communications, IEEE, vol. 22(2), pp. 362-370, 2004.

[7] M. Tropea, P. Fazio. *Evaluation of TCP versions over GEO satellite links*, 2013 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), IEEE, pp. 86-90, 2013.

[8] A. Ford, C. Raiciu, M. J. Handley, and O. Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses*, RFC 6824, 2013.

[9] J. Cloud, M. Medard. *Network Coding over SATCOM: Lessons Learned*, Springer International Publishing, pp. 272-285, 2015.

[10] Elert, Glenn *Orbital Mechanics I*, The Physics Hypertextbook, 2019.

[11] Ashwin Rao, Arnaud Legout, Yeon-sup Lim, Don Towsley, Chadi Barakat, Walid Dabbous. *Network Characteristics of Video Streaming Traffic*, CoNEXT, 2011.

[12] Atilla Eryilmaz, Desmond S. Lun. *Control for Inter-session Network Coding*, LIDS PUBLICATION, No. 2722, 2006.

[13] Maulin Patel, Nisarg Tanna, Pratik Patel, Raja Banerjee. *TCP over Wireless Networks: Issues, Challenges and Survey of Solutions*, The University of Texas at Dallas, 2001.

[14] Robert C. Durst, Gregory J. Miller, Eric J. Travis. *TCP extensions for space communications* , Wireless Networks (1997) 3, Springer, pp. 389-403, 1997.

[15] T. R. Henderson, Randy H. Katz. *Satellite Transport Protocol (STP): An SSCOP-based Transport Protocol for Datagram Satellite Networks*, University of California at Berkeley, 1997.

[16] Kai Xu, Ye Tian, Nirwan Ansari. *TCP-Jersey for Wireless IP Communications*, IEEE Journal on Selected Areas in Communications, IEEE, Vol. 22, pp. 747-756, 2004.

[17] Braun, Torsten, Du Pengyuan, Gerla Mario. *Integration of Multipath TCP and Network Coding for Satellite Networks*, 2017.

[18] `https://en.wikipedia.org/wiki/Multipath_TCP`. Accessed: 2020-2-8

[19] C. Raiciu, M. Handly, D. Wischik. *Coupled Congestion Control for Multipath Transport Protocols*, RFC 6356, 2011.

[20] Ramin Khalili, Nicolas Gast, Miroslav Popovic, Jean-Yves Le Boudec. *Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP*, IETF, Internet-Draft, 2014.

[21] A. Walid, Q. Peng, J. Hwang, S. Low. *Balanced Linked Adaptation Congestion Control Algorithm for MPTCP*, IETF, Internet-Draft, 2016.

[22] Kheirkhah, Morteza and Wakeman, Ian and Parisis, George. *Multipath-TCP in ns-3*, CoRR, arXiv, `http://arxiv.org/abs/1510.07721`, 2015.

[23] George F., RileyThomas R., Henderson. *The ns-3 Network Simulator*, Modeling and Tools for Network Simulation, Springer Berlin Heidelberg, pp. 15-34, 2010.

[24] *ns3::PointToPointNetDevice Class Reference*, `https://www.nsnam.org/doxygen/classns3_1_1_point_to_point_net_device.html`

[25] *ns3::ErrorModel Class Reference* `https://www.nsnam.org/doxygen/error-model_8cc_source.html`. Accessed: 2019-11-26

[26] Martin Jacobsson, Christian Rohner. *Estimating Packet Delivery Ratio for Arbitrary Packet Sizes Over Wireless Links*, IEEE Communications Letters, IEEE, Vol. 19, No. 4, 2015.

[27] *ns3::RandomVariableStream Class Reference*, `https://www.nsnam.org/doxygen/classns3_1_1_normal_random_variable.html`

[28] *ns3::BulkdSendApplication Class Reference* `https://www.nsnam.org/doxygen/bulk-send-application_8h_source.html`. Accessed: 2019-11-27

[29] *ns3::OnOffApplication Class Reference* `https://www.nsnam.org/doxygen/onoff-application_8h_source.html`. Accessed: 2019-11-27

[30] M. Chiesa, G. Kindler, M. Schapira, *Traffic Engineering With Equal-Cost-MultiPath: An Algorithmic Perspective*, IEEE/ACM Transactions on Networking, Vol. 25, no. 2, pp. 779-792, 2017.

[31] O. Bonaventure, C. Paasch, G. Detal. *Use Cases and Operational Experience with Multipath TCP*, RFC 8041, 2017.

[32] T. Henderson, S. Floyd, A. Gurtov, Y. Nishida. *The NewReno Modification to TCP's Fast Recovery Algorithm*, RFC 6582, 2012.

[33] Mattia Pedrazzi. *TCP Simulations*, `https://github.com/UsernameN0tAvailable/tcp_simulations`

[34] Kheirkhah Morteza, Wakeman Ian, Parisis George. *Multipath-TCP in ns-3*, `http://arxiv.org/abs/1510.07721`, CoRR, 2015.

[35] Arpaci-Dusseau, Remzi H., Arpaci-Dusseau, Andrea C. *Operating Systems: Three Easy Pieces*, Arpaci-Dusseau Books, Chapter: Scheduling Introduction, 2014.

[36] Mattia Pedrazzi *MPTCP Simulations* `https://github.com/UsernameN0tAvailable/mptcp_simulations`

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe o des Gesetzes vom 5. September 1996 über die Universität zum Entzug des aufgrund dieser Arbeit verliehenen Titels berechtigt ist.

Bern, DD.MM.YYYY

(Unterschrift)