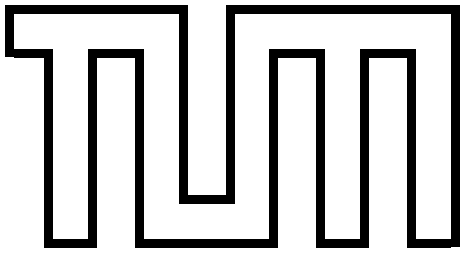


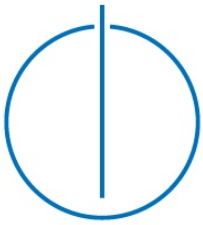
Technische Universität München

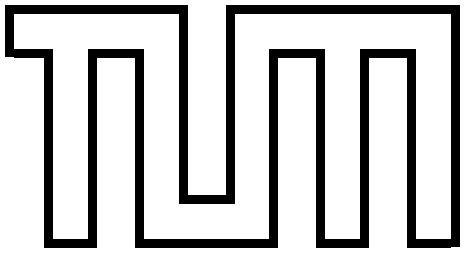
Fakultät für Informatik

Master's Thesis in Informatik

Feasibility Analysis of Multipath TCP for the Connectivity of Remote Piloting Operations of Drones and Flying Taxis

Kaushik Chavali





Technische Universität München

Fakultät für Informatik

Master's Thesis in Informatik

Feasibility Analysis of Multipath TCP for the Connectivity of Remote Piloting Operations of Drones and Flying Taxis

Machbarkeitsstudie zur Nutzung von Multipath TCP für die Fernsteuerung von Drohnen und Flugtaxis

Author: Kaushik Chavali

Supervisor: Prof. Dr.-Ing. Jörg Ott

Advisor: Aygün Baltaci, Mike Kosek

Submission: 15.03.2022

I assure the single handed composition of this master's thesis only supported by declared resources.

München, 14.03.2022

(Kaushik Chavali)

Acknowledgements

I thank my mentors, Aygün Baltaci and Mike Kosek, for their continuous support and guidance. They made extra efforts to provide me with all the resources at my disposal to realize the work. I appreciate the fruitful discussions we had throughout the Thesis. I thank Hendrik Cech for bringing me to speed with his previous experimentations. I am grateful to Dr. Nitinder Mohan, who shepherded the research.

I thank the constructive conversation with Dr. Brenton Walker critical to the realization of the testbed. I credit Simon Zelenski, who executed configuration changes on the emulation node as per the need.

I thank the Chair of Connected Mobility for providing the resources to conduct critical research. Finally, I thank my family and friends for their continued support.

I acknowledge Dr. Wöllik Helmut, Uran Christoph, Horvath Kurt, Egger Valentin from Carinthia University of Applied Sciences, Austria, to provide the Starlink measurement data from their experiments and collaboration.

Abstract

Next-generation cellular networks (4G/5G) and the emergence of low Earth orbit (LEO) constellations support applications with stringent Quality of Service (QoS) requirements such as remote piloting operations of aerial vehicles. However, aerial vehicles operate in harsh environments where wireless networks can be unreliable. We employ multipath connectivity to ensure link redundancy and improved reliability and use Multipath TCP (MPTCP) as a transport. Implemented as a TCP Extension to prevent protocol ossification, MPTCP is a recently standardized transport protocol that simultaneously utilizes multiple paths. MPTCP's performance largely depends on the selected schedulers and congestion controls

We propose a state-of-the-art multipath emulation testbed employing the MoonGen Long-Term Evolution (LTE) emulation script and the OpenSAND satellite communication (Satcom) emulator, which provides a host of configuration options. We model and simultaneously emulate cellular LTE and LEO Satcom links. We study the protocol performance using a holistic approach and evaluate various multipath scheduler and congestion control combinations. The preliminary results show MPTCP as a suitable protocol for such operations with a specific class of schedulers and congestion controls under emulated conditions.

Contents

Chapter 1

Introduction

Unmanned Aircraft Systems (UAS) or drones are a class of aircraft that does not require a pilot on-board to operate, and it elicits the need for remote piloting operations of such vehicles. In such a case, a pilot controls the aircraft from a farther ground station via robust communication channels, also known as a Command and Control (C2) link between the ground station and the aerial vehicle. Although there is a push for the fully autonomous operation of UAS, it poses several technical and regulatory challenges.

Drones are compact, lightweight flying objects realized through the advancement and interaction of various state-of-the-art technologies developed through the 21st century. It includes, for example, a Lithium-ion battery that is a lightweight energy storage system and enables the drone to achieve extended flight times.

There are also advances in the communication technologies used for UAS. Traditionally, drones have radios which operate in the licensed radio spectrum and offer radio line-of-sight communication. Commercial off-the-shelf drones communicate over the unlicensed spectrum and require a visual line-of-sight from the vehicle to the operator.

There is an increasing shift in the use of UAS for commercial purposes, for example, capture aerial imagery, surveillance, agriculture, search and rescue, etc. An emerging class of UAS is flying taxis meant to transport passengers swiftly and efficiently through congested cities. It requires beyond the visual line-of-sight (BVLoS) piloting by a remote operator.

The advancement in mobile telecommunication technologies with the deployment of LTE networks satisfies the QoS requirements for BVLoS operations which were not possible with older Third Generation (3G) cellular networks owing to high latencies and low bandwidth.

Another supporting communication technology for BVLoS operations is the Satcom network. Traditionally, the satellite constellations used for telecommunications are in the geostationary orbit (GEO) or medium Earth orbit (MEO). The reason being a few satellites can serve a large area on Earth. It comes with a disadvantage that since GEO and MEO are farther from the surface of the Earth, the signal encounters massive propagation delays resulting in latencies not suitable for real-time applications such as BVLoS operations of a UAS.

Many companies are deploying monumental satellite constellations in LEO with the recent advances in launch vehicles designs that are reusable and reduce satellite launch costs. LEO satellites are located very close to the Earth's surface compared to GEO and MEO, and hence they incur significantly lower signal propagation delays. It opens up the use of Satcom networks for more demanding applications with strict QoS requirements such as live streaming, online gaming, and in our case, the remote piloting of UAS. And enable broadband access to masses competing with terrestrial networks in terms of offered link quality and bandwidth.

International Civil Aviation Organization (ICAO) is an agency under the United Nations and deals with all aspects of civil aviation. The circular [Inte 12] identifies technical and operational issues on the integration of UAS into the non-segregated civil airspace. Its goal is to provide a regulatory framework through Standards and Recommended Practices (SARPs) and Procedures for Air Navigation Services (PANS) to ensure safe operations of UAS alongside conventional aircraft.

The focus of International Civil Aviation Organization (ICAO) from a regulatory standpoint is on suggesting high-level performance metrics, for example, specifying the minimum performance requirements for the communication links.

ICAO defines remotely-piloted aircraft (RPA) as an aircraft piloted by a licensed remote pilot situated at a ground station external to the aircraft. The main task of the remote pilot is to monitor the aircraft, communicate with the Air Traffic Controller (ATC), and be responsible for the safe operation of the aircraft throughout the flight.

The remotely-piloted aircraft system (RPAS) is composed of the RPA, associated remote pilot station(s), the C2 links, and any system-level component required for flight operations.

The circular also discusses the possibility of handovers of remote piloting responsibilities during the flight. For example, the aircraft operations shifted from one pilot to another or from one ground station to another. Considerations to automatically transfer the C2 link between the remote pilot stations are required. The safety and security of the

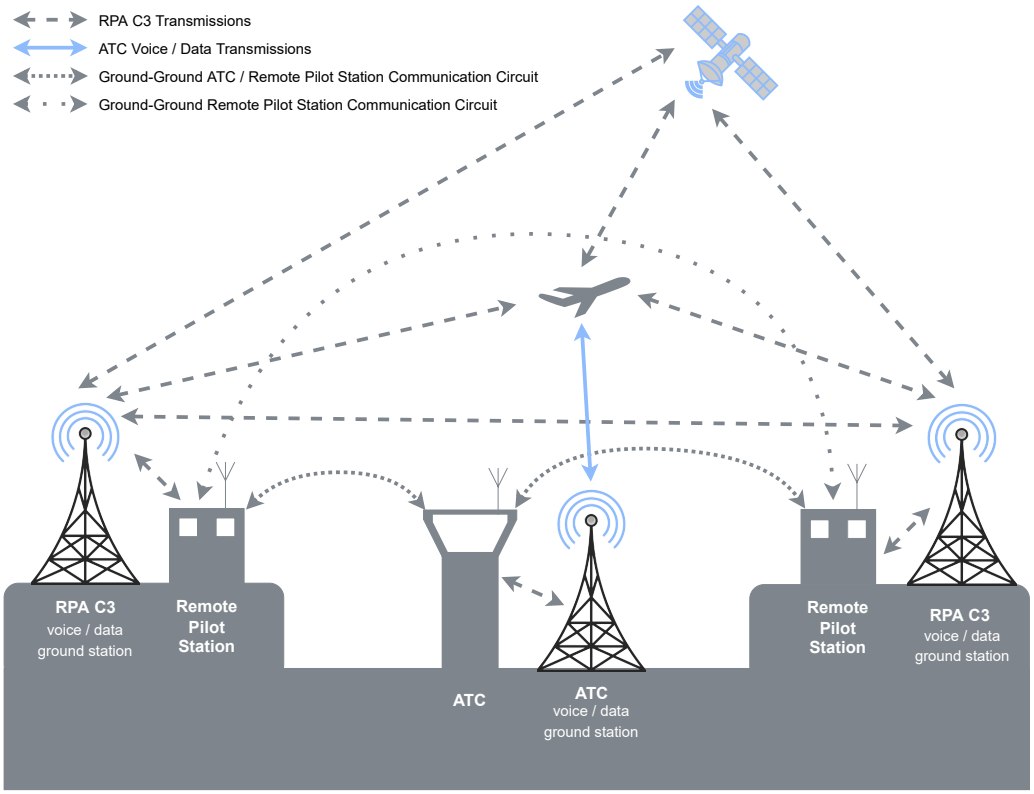


Figure 1.1: Communication Links involved in a RPAS. (Source: [Inte 12])

communication link from attacks due to malicious actors is a concern.

Long-haul operations require a change in the remote pilot station and the C2 link during the flight. As the aircraft moves away from the primary pilot station, there is an increase in C2 and communication performance issues. This effect will be especially prominent in dense urban environments with congested links compared with a flight path through oceanic and remote environments. In that case, there is a need to hand over the C2 link from a farther ground station to a nearby ground station.

The C2 data links need to be certified in addition to flight equipment so that they meet the required performance thresholds to ensure the safe operation of UAS. Loss of the C2 link needs consideration, and autonomous flight control systems must be available to ensure continuity of operations until the data link is re-established.

Redundancy in an aircraft system is of utmost importance to ensure safe operations in case of failures. The circular proposes the need to achieve a similar level of redundancy for an RPAS that includes the RPA, the remote pilot station, and the C2 data links. Establishing multiple data links for the C2 will be one such solution to ensure redundancy.

In addition to the C2 link, an additional communication channel connects the ATC and the remote pilot. It is known as the Control, Command, and Communication (C3) link, as shown in **Figure 1.1**. The C3 link must satisfy high QoS requirements, i.e., it must have high reliability, continuity, and integrity. The circular proposes two methods to establish such a link. First, a traditional air-to-ground approach involves a direct link from the ATC to the RPA, and the RPA relays the communication to the remote pilot station. A second solution is establishing ground-to-ground communication between the ATC and the ground station. It might require setting up new ground infrastructure and protocols.

The thesis investigates the C2 link between the RPA and the ground station. First, we introduce related work where the application QoS bounds are defined to ensure stable and reliable remote piloting operations in Chapter 2. Chapter 3 gives a brief overview of the protocol to support the application. Chapter 4 proposes a state-of-the-art multipath emulation testbed to emulate the C2 links that utilize the cellular LTE and the LEO Satcom infrastructure. Chapter 5 conducts a feasibility study to employ MPTCP as the transport to meet the defined QoS limits. We analyze the transport layer performance with a range of MPTCP schedulers and congestion control (CC) algorithms. Chapter 6 concludes the work and provides future directions to the research.

Chapter 2

Related Work

2.1 Application-level QoS requirements

The Master's Thesis [Cech 21] investigates the QoS metrics for remote piloting of aerial vehicles using cellular networks. The text models the downlink traffic using a constant 5 Mbps data stream, taking into account future upsurge in Control and Non-payload Communication (CNPC) traffic with more demanding use-cases like flying taxis. And to benchmark the transport under a higher load than what is required to meet the QoS requirements.

The uplink video and telemetry traffic are modeled as a 20 Mbps variable bitrate (VBR) traffic considering video with medium-high quality settings. Specifically, a 4K video with 60 frames per second (FPS) and a frame size of 3840×2160 employing the popular H.264 video compression standard.

The latency threshold for the remote piloting application is a point of contention. But [Balt 21] reveal that the 250 milliseconds (ms) can be considered an upper bound for uplink and downlink. The authors collected data on UAV communication from real-world measurements using commercial off-the-shelf drones. Further, 3GPP specifies a reliability threshold concerning Packet Error Rates (PER) as 10^{-3} for the bulk of UAS operation, while a PER of 10^{-4} for telemetry traffic that stems from UAS during take-off and landing. The Radio Technical Commission for Aeronautics (RTCA) specifies $> 99.976\%$ communication availability and $> 99.9\%$ communication continuity for CNPC traffic carried on C2 links for BVLoS operations, as mentioned by the authors.

2.2 Mobility and mutipath protocols

[Paas 12] analyzes MPTCP's performance and energy consumption when performing mobile wireless handovers on a heterogeneous LTE and WiFi networks. They introduce various MPTCP-operational modes designed keeping handovers in mind. The **Full-MPTCP Mode** behaves as a regular MPTCP connection with a full mesh of TCP subflows to provide throughput aggregation. The **Backup Mode** assigns priorities to subflows to prefer one subflow based on user requirements. The **Single-Path Mode** behaves like Stream Control Transmission Protocol (SCTP). They conclude that the **Full-MPTCP Mode** offers a smooth handover and good performance for real-time applications.

[Wang 19] perform active and passive measurement studies of the TCP performance over LTE under mobility on high-speed rails. They measure performance metrics like goodput, latency, loss rate and investigate LTE handovers. They employ TCP Cubic and TCP BBR as the congestion controls. BBR offers up to 36.5 percent more goodput when compared to measurements with Cubic. The measured performance depends on network coverage and signal strength. They highlight the need to develop dedicated protocol mechanisms for mobility scenarios. They recommend employing heterogeneous cellular and satellite links under mobility to improve efficiency and robustness.

[Yap 12] aims to use multiple wireless radios on a smartphone to achieve resource pooling. They refactor Android's networking stack so that the application host can use underlying TCP connections over multiple radio interfaces via a gateway acting as a load-balancer for seamless connectivity, faster connections, and lower cost. The redundancy is achieved at the network layer employing Open vSwitch. They discuss addressing ambiguity, connection discovery, middlebox interference, and heterogeneity in the network paths.

[Balt 14] conduct a longitudinal study to measure the reliability of mobile broadband networks, especially the UMTS network. They find cellular mobile broadband unreliable in terms of availability with a loss of connection on a stationary mobile node ranging at least 10 minutes per day for more than 20 percent of connections. However, they show the benefit of aggregating multiple mobile networks operators to improve robustness and achieve five-nines of connection availability.

[Saxe 20] proposes a UDP-based application-level multipath solution employing torrents and network coding (NC) designed to work with heterogeneous satellite and terrestrial networks for the use-case of UAV. NC removes the need to develop an efficient scheduler since a linear combination of packets is sent over the available paths. NC eliminates the need for Forward Error Correction (FEC) schemes, reduces the cost of retransmissions and

acknowledgments, and offers throughput gains. In addition, it provides network security since the packets are coded. The use of torrents removes the need for rate adaptation algorithms. They claim the scheme offers up to 70 percent goodput gains compared to MPTCP. However, additional parameters are not analyzed. However, wireless network coding implementations are not optimized for energy efficiency, and computation is required to code and decode packets [Katt 08].

2.3 Emulation Testbeds

NorNet is a large-scale Internet testbed for multi-home systems and applications [Drei 13]. [Drei 15] adds support for MPTCP to evaluate multipath protocol mechanisms in wired networks. [Kval 14] extends the infrastructure to work with mobile broadband networks.

[Neme 13] proposes an experimental multipath testbed on the large-scale PlanetLab network and provides tools to test various multipath TCP mechanisms like congestion controls and bottleneck fairness. The author also points out that installing, configuring, and integrating MPTCP software components consumes a significant amount of time for researchers. They allow the user to configure the network topology using configuration files, creating an OpenFlow-based overlay network that tunnels traffic according to the configured topology.

However, we do not find heterogeneous link emulation employing the LTE and the SATCOM links in the literature.

Chapter 3

Background

3.1 Multipath TCP (MPTCP)

3.1.1 Introduction

TCP is the most widely used Transport on the Internet. However, it is unable use multiple paths between the two endpoints if they exist. It results in a missed opportunity to fully utilize the end-to-end network capacities and provide reliability to the connection. Stream Control Transmission Protocol (SCTP) standardized by the Internet Engineering Task Force (IETF) in RFC 4960 [Stew 07] offers multihoming support. But it suffers from protocol ossification and an inability to simultaneously send segments through multiple paths.

Multipath TCP (MPTCP) v0 standardized by IETF in RFC 6824 [Ford 13] and obsoleted by MPTCP v1 in RFC 8684 [Ford 20] is the most recent effort to provide multihoming support to the connections at the transport layer. The protocol implemented as a TCP Extension builds upon an already existing TCP protocol avoiding the ossification problem. MPTCP offers multihoming support and the ability to simultaneously send segments on multiple subflows. It allows better network resource usage and provides resiliency against network failure. MPTCP furnishes the same type of service as TCP, i.e., connection-oriented service with in-sequence delivery of byte streams. MPTCP offers flow control and congestion control to not overwhelm the receiver and the network by a fast sender.

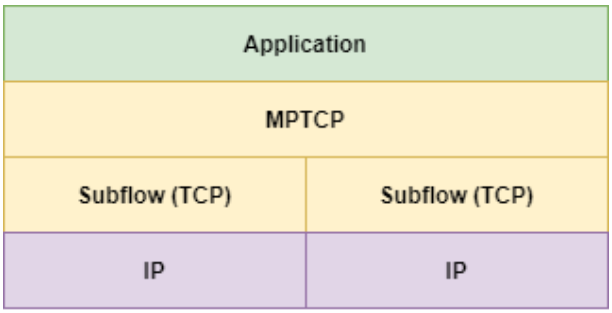


Figure 3.1: MPTCP protocol stack with the application byte-stream segmented and scheduled over the underlying TCP subflows.

3.1.2 MPTCP Architecture

Figure 3.1 shows the MPTCP protocol stack. MPTCP operates at the transport layer and is a collection of additional features on top of standard TCP. A path is a sequence of links between the sender and the receiver identified by the 4-tuple, i.e., the source address and port and the destination address and port. A subflow is a flow of segments over a discrete path and is a component of an MPTCP connection. MPTCP connection is a set of one or more TCP subflows over which the application can communicate between end hosts. A host identifies a multipath connection with a locally unique identifier known as a token or connection identifier (Connection ID) [Ford 20].

3.1.3 Functional decomposition of MPTCP

The MPTCP protocol can be functionally decomposed into four main components which are listed below [Ford 11].

- **Path Management.** The path management component of MPTCP detects and utilizes multiple paths between two hosts. The presence of multiple IP addresses on one or both hosts triggers the mechanisms to establish new subflows to an existing MPTCP connection.
- **Packer Scheduling.** The MPTCP scheduling component uses the services of the path management component to detect the presence of multiple subflows. It fragments the byte stream received from the application into segments transmitted via the underlying subflows. The MPTCP design employs data sequence mapping that associates segments on the subflows to a connection-level sequence space. It enables the correct reordering of segments at the hosts arriving over different subflows.

- **Subflow Interfaces.** The subflow interface takes the segments from the scheduling component and transmits them over the specified single-path TCP interface, ensuring reliable, in-order delivery. The subflow employs its own sequence space to detect and retransmit lost packets on the subflow level. On reception, the subflow passes the reassembled data to the packet scheduling component for connection-level reassembly using the data sequence mapping.
- **Congestion Control.** The function coordinates the congestion control across the subflows, ensuring that an MPTCP connection is not unfair to competing single-path TCP flows on the bottleneck link. It is a part of the packet scheduling component and determines which segments should be sent, at what rate on which subflow.

3.1.4 Operation Overview

RFC 8684 gives an operational overview of the MPTCP protocol [Ford 20]. The protocol operation is transparent to the application. However, MPTCP acts as a regular TCP flow at the network layer with segments carrying the MPTCP option. The MPTCP operational modes use TCP options to create, remove, and utilize the subflows to send data. MPTCP falls back to regular TCP if the host is not MPTCP capable.

MP_CAPABLE

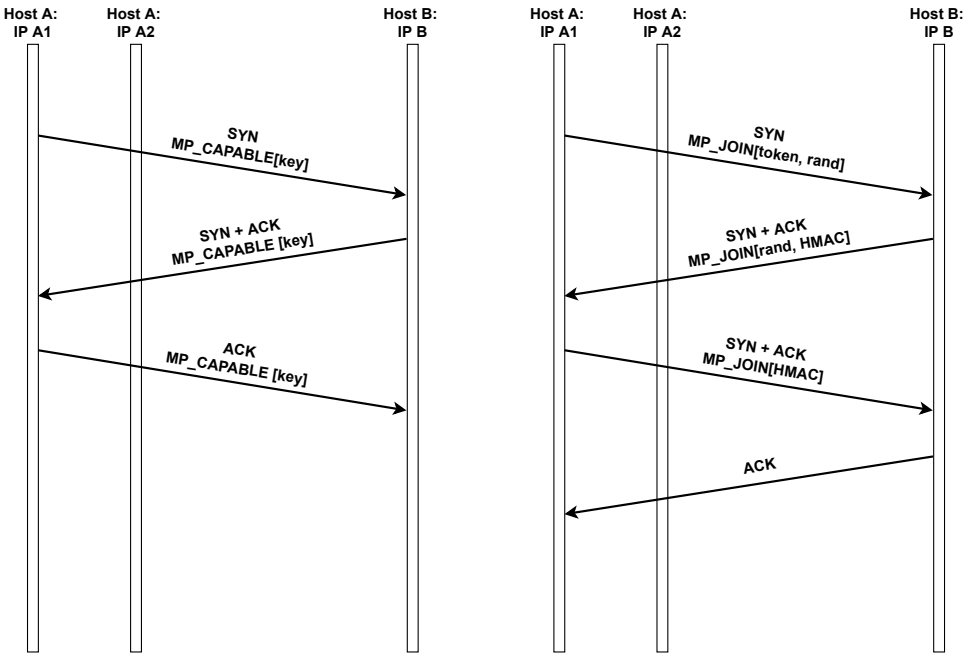
Initiating an MPTCP connection is similar to regular TCP with SYN, SYN/ACK, and the initial ACK with the additional *MP_CAPABLE* option. The option verifies if the remote host is multipath capable. If that is the case, it exchanges keys to authenticate the establishment of additional subflows, as shown in textbfFigure 3.2.

MP_JOIN

MPTCP_JOIN option is used to associate a new subflow with an existing MPTCP connection. Associating a new subflow is similar to regular TCP with SYN, SYN/ACK, and ACK segments with the *MP_JOIN* option and the shared keys from the initial connection, as shown in **Figure 3.2**.

ADD_ADDR

MPTCP supports the addition and removal of addresses on a host both implicitly and explicitly. The *ADD_ADDR* option explicitly signals the availability of additional IP addresses associated with the multihomed client to the remote host. In which case, the remote host establishes the connection with the *MP_JOIN* option.



(a) MPTCP Initial Handshake with $MP\_CAPABLE$ option.

(b) Associate a new subflow with an existing MPTCP connection with the MP_JOIN option.

Figure 3.2: MPTCP Connection Setup with a TCP three-way handshake employing TCP Options and keying material.

3.1.5 MPTCP Schedulers

MPTCP Scheduler is a component that schedules segments of an MPTCP connection across the underlying TCP subflows. There are two primary ways to trigger the MPTCP scheduler. First, the scheduler runs as soon as it receives data from the application buffer. The scheduler pushes data into the underlying sub-flows. The second is a pull-based approach in which the scheduling takes place at transmission time when an acknowledgment (ACK) frees up space in the subflow's congestion window (CWND).

There are three primary decisions that the scheduler has to take during scheduling segments. Decide which subflows to send data on concerning their available CWNDs. Which subflow to choose if there are multiple subflows available. And finally, how much data to send on the selected subflow.

[Paas 14] defines two main constraints an MPTCP scheduler needs to consider. These constraints affect the performance metrics like the goodput and application delay, notably in heterogeneous networks with varying link characteristics such as link delay. **Figure 3.3** illustrates the problem of MPTCP HoL Blocking and Receive Buffer Blocking.

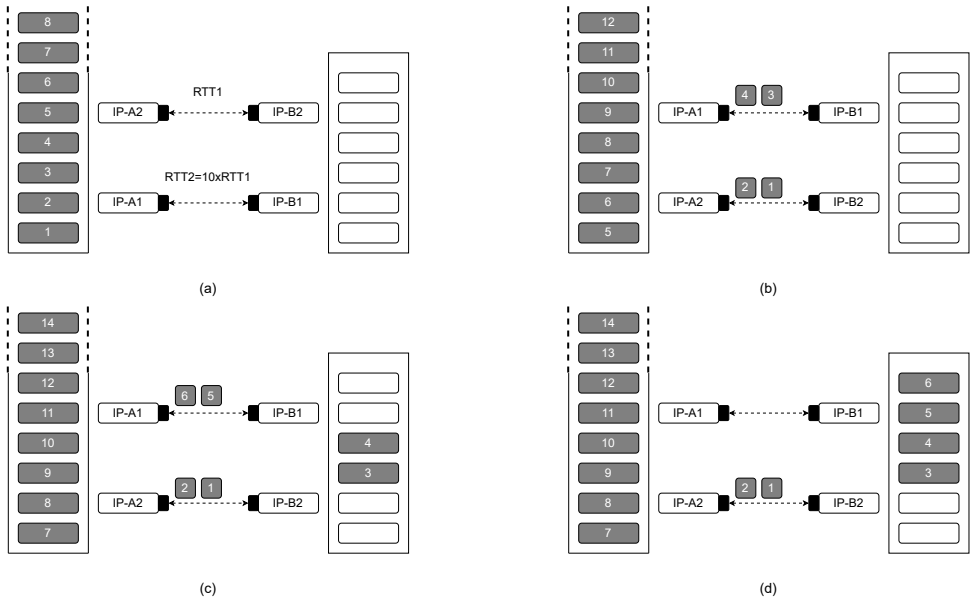


Figure 3.3: Exemplar of the Receiver Buffer and HoL Blocking on paths with heterogeneous RTTs. (a) Initial path latencies. (b) Segements transmitted. (c) HoL Blocking (d) Reciever Buffer Blocking [Hurt 19]

- **Head-of-line Blocking**

  Consider a multipath connection over heterogeneous links such as the LTE and the SATCOM links investigated in the Thesis. In general, the segments that traverse the SATCOM subflow experience higher RTTs than the LTE subflow. It causes packets scheduled over the lower-RTT subflow to arrive faster at the receiver than those scheduled over the higher-RTT subflow. MPTCP, like TCP, guarantees the in-order delivery of segments to the application. The differential arrival of segments builds up the out-of-order queue at the receiver, causing burstiness and delaying data delivery to the application and known as head-of-line (HoL) blocking [Scha 06]. HoL blocking can cause degradation in the application QoE, especially for video streaming.

- **Receive-window limitation**

  MPTCP maintains a receive window shared between the subflows in addition to subflow-level out-of-order (OFO) queues [Ford 20]. The receive-window limitation problem stems from the insufficient memory allocation for the OFO data received in case of either packet loss or in-network packet reordering, which is common in heterogeneous networks with delay differences across subflows. The configured receive buffer size is integral for enabling increased goodput. The recommendation for the receive buffer size is the total subflow bandwidth multiplied by the maximum RTT of any path multiplied by two to account for retransmissions [Barr 11].

Lowest-RTT Scheduler

The Lowest-RTT First (LRF) scheduler is the default MPTCP scheduler. It schedules segments onto the subflow with the lowest-SRTT estimation until it has filled its CWND. The data is then sent on the subflow with the next highest-SRTT estimation. As soon as the CWNDs of the subflows fill up, the received ACKs dictate the scheduling decisions of the subsequent segments, and the scheduler is said to have become ACK-clocked. Additional extensions to the LRF scheduler deal with the problem of HoL blocking and receive-window limitation, which is listed below.

- **Retransmission and Penalization (RP)**

  [Raic 12] proposes Opportunistic Retransmission to re-inject segments causing HoL blocking on the subflow, which has space in its CWND. It allows the receiver to recover from HoL blocking swiftly. Additionally, the CWND of the high RTT subflow is penalized by reducing its send rate and avoiding the effect of bufferbloat on the subflow. This scheme targets a reduction in latency, jitter, and buffer occupancy, in addition to a goodput increase. It takes a reactive approach to packet scheduling.

- **Bufferbloat Mitigation (BM)**

  Bufferbloat Mitigation scheme targets the cause of bufferbloat due to large buffers on routers and switches by monitoring the drift between the minimum SRTT and the current SRTT estimation and dynamically limiting the amount of data sent to each subflow if it passes a certain threshold. In doing so, it takes a proactive approach to packet scheduling.

Round-robin Scheduler

The round-robin (RR) scheduler schedules segments in a round-robin fashion, thus ensuring equal link utilization on both suflows. However, the scheduler suffers from ACK-clocking for bulk data transfers as described in the LRF scheduler preventing it from scheduling in a true RR fashion. The scheduler is not designed to be used in the real world but rather for academic and testing scenarios.

Redundant Scheduler

The MPTCP Redundant scheduler proposed by [From 16] schedules packets redundantly over all the available subflows reducing latency and jitter, especially in links with a high packet loss and retransmissions. It uses extra bandwidth to reduce latency.

BLEST Scheduler

Algorithm 1 BLEST (Source: [Hurt 19])

Require: $srtt_{\mathrm{F}} < srtt_{\mathrm{S}}$

if $can\_send(F)$ **then**
    $selected\_subflow = F$
else if $can\_send(S)$ **then**
    $rtts = srtt_S / srtt_F$
    $X = MSS_F \times (CWND_F + (rtts - 1)/2) \times rtts$
    if $X \times \lambda \leq MPTCP_{SW} - MSS_S \times (inflights_S + 1)$ **then**
        $selected\_subflow = S$
    end if
end if

[Ferl 16] proposes send-window Blocking Estimation (BLEST) MPTCP scheduler to minimize the HoL-blocking in heterogeneous networks. It employs a proactive approach to packet scheduling by estimating if scheduling a segment over a specific subflow will cause HoL-blocking at the receiver. It then decides whether to send segments over the slower subflow or skip and wait for a faster subflow to be available to avoid the risk of send-window blocking, as shown in **Algorithm 1**. The estimation is derived from the flow's CWND and RTT.

3.1.6 Congestion Control

The MPTCP Congestion Control (CC) algorithms control the transmission rates of the underlying TCP subflows in response to varying link conditions and congestion. MPTCP classifies the CC algorithms into coupled and uncoupled CC described below. [Raic 11] defines the three primary goals a multipath CC algorithm should satisfy.

G1 Improve throughput. A multipath flow should perform at least as well as a single path flow would on the best of the paths available to it.

G2 Do no harm. A multipath flow should not take up more capacity from any of the resources shared by its different paths than a single flow using only one of these paths. This guarantees it will not unduly harm other flows.

G3 Balance congestion. A multipath flow should have as much traffic as possible off its most congested paths, subject to meeting the first two goals.

The first two goals ensure bottleneck fairness. The last goal provides the resource pooling concept [Wisc 08]; if each multipath flow sends more data through its least congested path, the traffic in the network will move away from the congested areas. It improves robustness

and overall throughput, among other things. The way to achieve resource pooling is to effectively "couple" the congestion control loops for different subflows.

Uncoupled Congestion Control

Uncoupled CC strategies treat the underlying TCP subflows independently with their instance of a CC running. [Raic 11] identifies a series of issues that plague the single path algorithms in a multipath context. The primary one deals with fairness in the context of uncoupled CC algorithms; the multipath flow gets more than its fair share bandwidth in the bottleneck. The second issue deals with the concept of resource pooling. In the presence of multiple available paths, the CC algorithm should behave like one shared link with a bigger capacity by transferring more traffic using the least of the congested paths, thereby increasing the overall network efficiency and its robustness to failures. The uncoupled CC strategy fails to satisfy Goals 1 and 2 described above.

For our evaluations of uncoupled CC protocols, we employ the two loss-based CC algorithms, TCP NewReno [Gurt 12] and TCP Cubic [Rhee 18], which take packet loss as a congestion signal and run independently on each available path.

Coupled Congestion Control

The coupled CC algorithms link the increase function of the CC running on underlying suflows. By doing so, the CC can dynamically adapt the overall aggressiveness of the multipath flow and ensure bottleneck fairness and resource pooling. It aims to satisfy all the goals mentioned above. The congestion control primarily executes in the congestion avoidance phase. The slow start, fast retransmit, and fast recovery algorithms work the same as in standard TCP.

The coupled CC must achieve fairness to single path flows at the bottleneck, responsiveness to path changes, offer goodput aggregation, and resource pooling.

The Linked Increased Algorithm (LIA) [Raic 11], Opportunistic Linked-Increases Algorithm (OLIA) [Khal 13], Balanced Linked Adaptation Algorithm (BALIA) [Wali 16], and the weighted Vegas (wVegas) [Cao 12] the four primary CC algorithms are available in the MPTCP Linux kernel implementation. LIA, OLIA, and BALIA are loss-based CC algorithms, while wVegas is a delay-based CC algorithm that takes packet delay as a congestion signal.

Linked Increased Algorithm (LIA)

The Linked Increased Algorithm (LIA) [Raic 11] couples the additive increase function of the subflows and uses unmodified TCP behavior in case of a drop. It relies on traditional

TCP mechanisms to detect drops and retransmit data. The controller tries to answer how much bandwidth a multipath user should use in total, even with a shared bottleneck. It aims to set the multipath flow's aggregate bandwidth the same as a regular TCP flow on the best available path to the multipath flow. The multipath flow estimates loss rates and round-trip times (RTTs) and computes the target rate to estimate a regular TCP flow bandwidth. Then it adjusts the overall aggressiveness parameter alpha to achieve the desired rate.

The increase formula in **Table 3.1** takes the minimum between the computed increase for the multipath subflow and the increase TCP would get in the same scenario. The aggressiveness factor α is such that the multipath throughput is equal to the throughput TCP flow would be on the best path.

Opportunistic Linked Increases Algorithm (OLIA)

OLIA addresses the suboptimality of MPTCP with LIA. To provide responsiveness to network changes, LIA departs from optimal load balancing. [Kell 05] provide theoretical results and state that load balancing in multipath routing is optimal in static networks with symmetrical RTTs. However, in a dynamic multipath setting, the routing is unresponsive, i.e., fails to detect free capacity and flappy, i.e., when multiple good paths are available, it randomly flips traffic between them.

OLIA is an algorithm inspired by the utility maximization framework and simultaneously offers responsiveness and congestion balancing. The algorithm defines the CWND increase as a function of the quality of paths. The first term in the increase function provides optimal congestion balancing, and the second term offers responsiveness to react to changes in the current windows. Additionally, the authors claim OLIA outperforms LIA in all the studied scenarios [Khal 13].

Balanced Linked Adaptation (BALIA)

BALIA explicitly balances the tradeoff between friendliness and responsiveness. LIA can be unfriendly to SPTCP and OLIA, unresponsive to network changes. BALIA allows for CWND oscillations up to an ideal level to provide a good balance between the two. On a single path, BALIA reduces to TCP NewReno.

Weighted Vegas (wVegas)

wVegas estimates the link queuing delay to realize path congestion and then proactively adapt the CWND [Kimu 18].

A summary of the coupled CC algorithms is shown in **Table 3.1**.

Table 3.1: A summary of MPTCP coupled CC algorithms [Kimu 18, Wali 16, Khal 13, Raic 11, Cao 12]

	Alpha Parameter	**Congestion Window Increase**	**Congestion Window Decrease**
LIA:	$\alpha = w_{\text{total}} \dfrac{\max\left(w_r/\tau_r^2\right)}{\left[\sum_{k\in\mathcal{R}}\left(w_k/\tau_k\right)\right]^2}$	For each ACK received on subflow r, $w_r = w_r + \min\left(\dfrac{\alpha}{w_{\text{total}}}, \dfrac{1}{w_r}\right)$	For each packet loss on subflow r, $w_r = w_r - \dfrac{w_r}{2}$, as in standard TCP
OLIA:	$\alpha_r = \begin{cases} \frac{1/\lvert\mathcal{P}\rvert}{\lvert\mathcal{C}\rvert}, & \text{if } r \in \mathcal{C}, \\ \frac{1/\lvert\mathcal{P}\rvert}{\lvert\mathcal{W}\rvert}, & \text{if } r \in \mathcal{W}, \lvert\mathcal{C}\rvert > 0, \\ 0, & \text{otherwise} \end{cases}$	For each ACK received on subflow r, $w_r = w_r + \left(\dfrac{w_r/\tau_r^2}{\left[\sum_{k\in\mathcal{R}}\left(w_k/\tau_k\right)\right]^2} + \dfrac{\alpha_r}{w_r}\right)$	For each packet loss on subflow r, $w_r = w_r - \dfrac{w_r}{2}$, as in standard TCP
BALIA:	$\alpha_r = \dfrac{\max\{x\}}{x_r}$, where $x_r = \dfrac{w_r}{\tau_r}$, and $x = \left[\sum_{k\in\mathcal{P}}\left(w_k/\tau_k\right)\right]^2$	For each ACK received on subflow r, $w_r = w_r + \left[\dfrac{x_r}{\tau_r\ x_k}\ \left(\dfrac{1+\alpha_r}{2}\right)\ \left(\dfrac{4+\alpha_r}{5}\right)\right]$	For each packet loss on subflow r, $w_r = w_r - \left[\dfrac{w_r}{2}\ \min\left(\alpha_r, \dfrac{3}{2}\right)\right]$
wVegas:	**if** $(\delta_r > \alpha_r)$, **then** $\alpha_r = \omega_r\ \alpha_{\text{total}}$, where $\delta_r = \left(\dfrac{w_r}{\hat{\tau}_r} - \dfrac{w_r}{\bar{\tau}_r}\right)\ \hat{\tau}_r$, $\omega_r = \dfrac{x_r}{\sum_{k\in\mathcal{R}} x_k}$, and $x_i = \dfrac{w_i}{\bar{\tau}_i}$	For each transmission round on subflow r, $w_r = \begin{cases} w_r - 1, & \text{if } \delta_r > \alpha_r, \\ w_r + 1, & \text{if } \delta_r < \alpha_r \end{cases}$ **if** $(q_r > \bar{\tau}_r - \hat{\tau}_r)$, **then** $q_r = \bar{\tau}_r - \hat{\tau}_r$ **if** $(\bar{\tau}_r - \hat{\tau}_r \geq 2q_r)$, **then** $w_r = w_r\ \dfrac{\hat{\tau}_r}{2\bar{\tau}_r}$	For each packet loss on subflow r, $w_r = w_r - \dfrac{w_r}{2}$, as in standard TCP

3.1.7 Application Considerations

RFC 6897 summarizes the performance impact of MPTCP on the application, discusses compatibility issues of MPTCP in combination with non-MPTCP-aware applications, and finally describes a basic API that is a simple extension of TCP's interface for MPTCP-aware applications [Scha 13].

The Thesis focuses on the performance implications of MPTCP on the application compared to SPTCP in terms of the achieved throughput, delay, and resilience. MPTCP aims to improve the performance of the transport via the concept of resource pooling. Additionally, it should provide a connection that performs at least as well as the one using SPTCP.

Throughput

The draft specifies three considerations regarding the achieved application-level throughput or goodput. MPTCP provides, in most cases, aims to provide bandwidth aggregation utilizing the underlying subflows. The flexibility to add and remove subflows as path availability changes create variation in connection bandwidth. Additionally, the MPTCP signaling adds a small overhead to the connection. Hence the use of MPTCP instead of SPTCP results in a smaller goodput.

Latency

MPTCP trades in latency for throughput and resilience. Heterogenous subflows add to the application's perceived jitter, depending on the configured schedulers and congestion controls. The draft recommends running MPTCP in a high-reliability rather than a high-throughput mode for latency-sensitive applications by setting the secondary flow as a backup flow. It also highlights that if one of the subflows fails, the increased retransmissions inside MPTCP can affect the application's perceived latency which is still better than the connection terminated in the case of SPTCP.

Resilience

MPTCP provides resilience in link failures by utilizing the secondary subflows and retransmitting lost packets. It also supports both break-before-make and make-before-break handovers between subflows and survives unavailability or change in IP addresses.

Chapter 4 proposes a state-of-the-art heterogeneous multipath emulation testbed employing the MPTCP protocol.

Chapter 4

State-of-the-art Multipath Emulation Testbed

This chapter presents a novel multipath emulation testbed capable of concurrently emulating cellular LTE, and Satcom links. It provides a powerful tool to run replicable experiments and evaluate transport and application layer protocols with a host of configuration options. We achieve this by employing the MoonGen LTE emulation script [Stra 20, Cech 21] and the OpenSAND Satcom emulator [Brun 19, Beil 21, Pola 22] as previous works. To realize this, we use a similar framework that runs the Satcom emulation and augment it to run with the LTE emulation as described later. In addition to this, we abstract both the emulators and provide a single set of configuration options to run end-to-end single-path and multipath measurements.

We implement additional features to both the emulators, like an accurate LTE Handover Model tailored for our use-case. Ability to add variable delays when emulating the Satcom link derived from running packet-level network simulations on a LEO simulator is also provided.

In the Thesis, the particular focus is on emulating the LEO Satcom link and the characteristics of LTE network under mobility in the air. However, the testbed provides a high amount of flexibility to configure a multitude of permutations and combinations in the single-path and multipath context. We measure MPTCP's performance as a transport with different multipath schedulers and congestion controls.

4.1 Introduction

A network experiment technique is a methodology by which one accesses the characteristics of a network system, application, or protocol to gauge its performance, conformance to design, or product specification. Network simulation, network emulation, and real-world testing are the most common network experiment techniques [Beur 13].

Network emulation is a network experiment technique that employs an experimental setup containing physical network components, either hardware or software, and components reproduced virtually through computer models. In comparison, the experimental design of the network simulation is based entirely on computer models. And real-world testing is based solely on physical network systems, applications, and protocols.

As opposed to simulation, network emulation brings realism to network experiments. It employs protocols running on real network devices while inheriting the reproducibility and control from simulation. Reproducibility and control over the experiments are hard to achieve in real-world testing as the network devices are under varying load conditions and environments. In that sense, network emulation bridges the gap between simulation and real-world testing.

[Beur 13] defines the criteria for evaluating network experiment techniques. Network emulation has moderate experimental costs, the ability to execute experiments in real-time, reasonable control over experimental conditions, moderate reliability of results, and ease of use when benchmarked against simulation and real-world testing.

To this end, we employ network emulation to emulate both the links required to evaluate the MPTCP protocol for the defined use case of remote piloting aerial vehicles. The following sections introduce the MoonGen LTE emulator and the OpenSAND Satcom emulator to emulate the cellular and the LEO Satcom links.

4.2 LTE emulation

We employ the LTE emulator proposed by [Stra 20] built on the Intel Data Plane Development Kit (DPDK) [Linu 21] and MoonGen [Emme 15] packet processing tools to achieve realistic LTE link emulation. The authors employ the tools mentioned above to support the emulation of LTE, which is a complex and dynamic stateful system. Unlike traditional network emulators based on event-driven systems, the DPDK framework offers a polled mode operation to the emulator. It ensures the emulator has better control over the packet timings.

The LTE emulator supports emulation of LTE-specific network properties such as heterogeneous uplink and downlink rates, packet losses, concealed loss recovery at the link layer utilizing the Hybrid Automatic Repeat Request (HARQ), and power-saving features such as the Discontinuous Reception (DRX).

Additionally, we contribute to the LTE emulation script by modeling the handover parameters from the real-world cellular measurements conducted from an aerial vehicle.

The LTE emulation script and the configured parameters are modeled after the results of a measurement study conducted by [Beck 14] that examines the application-level performance of LTE on a commercial network. We employ the same parameters to configure our LTE link unless mentioned otherwise.

DPDK is a set of libraries and drivers for fast packet processing that supports many processor architectures. It replaces the traditional Network Interface Card (NIC) driver with a Poll-Mode Driver (PMD), allowing fast and synchronous packet processing in userspace.

MoonGen adds an abstraction to the DPDK framework and enables programming the packet generation and processing logic with the help of user-controlled Lua scripts. MoonGen's novelty lies in the fact that it can precisely control the inter-packet timings by using dummy frames. The use of dummy frames prevents packet bursts when packets queue at the operating system and hardware buffers. This novel mechanism is called CRC-based Rate Control.

4.2.1 Architecture

Figure 4.1 shows the architecture of the MoonGen LTE emulation script. The script consists of four processing threads, a pair of forward and receive threads in each direction, to emulate a full-duplex link. A packet-size ring or a latency queue data structure facilitates link emulation.

The receive thread pulls frames out of the DPDK RX descriptor ring, timestamps them, and queues them into the latency queue data structure. The bulk of the emulation takes place at the forward thread, which pulls out the frames from the latency queue, checks the frame's arrival timestamp, and emulates them according to the state of the LTE system at that instant. If it is time to forward the frame, the forward thread schedules it onto the DPDK TX descriptor ring for transmission. The forward thread processes only one frame at a given time like an actual LTE medium access control (MAC) sublayer by employing the first-in, first-out (FIFO) queuing discipline. It ensures that the transport-layer loss

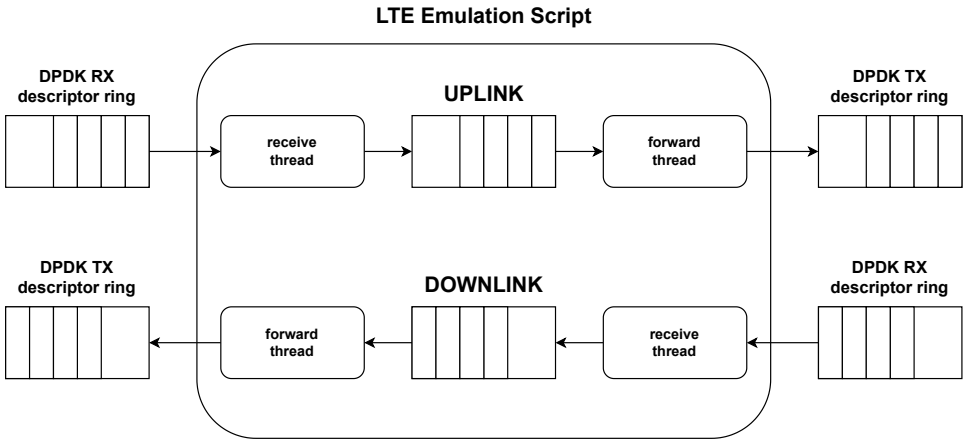


Figure 4.1: MoonGen LTE Emulatoion Script [Stra 20]

detection is not triggered. The latency queue is part of the modeled LTE system and hence sized accordingly. Thus, if it fills up, frames are dropped.

4.2.2 Modeled Parameters

Bandwidth

The authors emulate the heterogeneous uplink and downlink bandwidth with the help of MoonGen's CRC-based rate control. The uplink and downlink bandwidth are configured to 40 Mbps and 38 Mbps, respectively, following the measurements from [Beck 14].

Latency

The emulated baseline latencies are 30 ms and 10 ms in the uplink and downlink direction, respectively. In an LTE network, the UE needs to schedule frames for transmission to the eNB. On the other hand, in the downlink direction, the eNB can forward frames to the UE as soon as they arrive. Hence the heterogeneous nature of latencies in the uplink and the downlink directions.

Packet Losses and HARQ

There are two types of losses emulated by the LTE emulation script. The first is the concealed losses at the link layer, and the second is the packet losses occurring at the LTE backhaul network. The LTE HARQ mechanism corrects link-layer losses and hides them from upper network layers. This type of loss is transparent to the transport, which only detects a slight increase in packet delays but no loss. The losses in the LTE backhaul network occur due to software errors or buffer overflows at routing hops along the communication path. This type of loss can influence the CC at the transport.

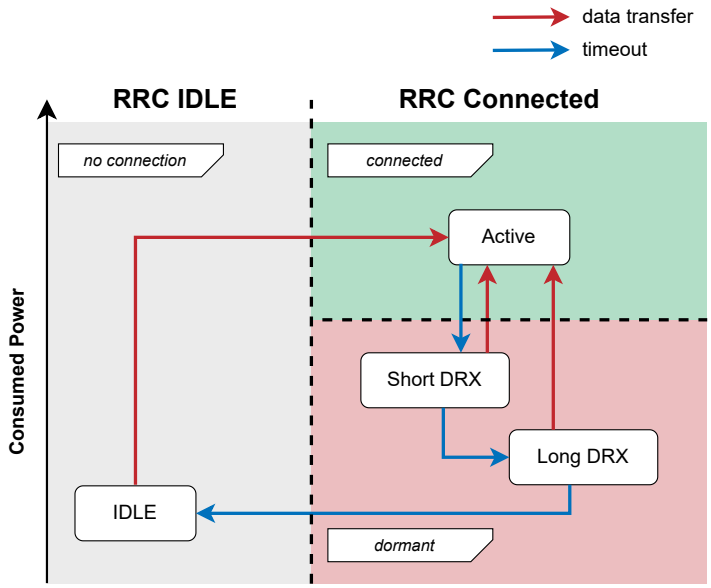


Figure 4.2: LTE Discontinuous Reception (DRX) states. [Sook 19]

The authors configure the packet losses with a rate of 0.001 and concealed losses at the rate of 0.005. However, we estimate the packet losses from the aerial LTE measurements with a rate of 0.0006 and use this value.

Queue Depth

The queue depth of the packet-sized queue is 350 packets and 1000 packets in the uplink and the downlink direction, respectively. The authors of [Stra 20] justify that the configured values replicate the link behavior more accurately than those estimated in [Beck 14].

DRX states

The LTE DRX is part of the LTE power-saving feature that adaptively powers down the radio on the UE during idle periods to reduce energy consumption. **Figure 4.2** shows the state transition diagram of the DRX states. The Radio Resource Controller (RRC) is responsible for enforcing the DRX states.

A UE connected to the eNB can be idle or connected. The UE does not transmit any frames in an idle state and powers down its radio. It occasionally turns on its radio to check for any transmission from the eNB every cycle period. The device transitions to the connected state if frames are scheduled to be received from the eNB. The authors emulate the latency from link establishment to be 70 ms. The link bandwidth is initially set to 5 Mbps and then linearly scaled to the emulated bandwidth over 500 ms.

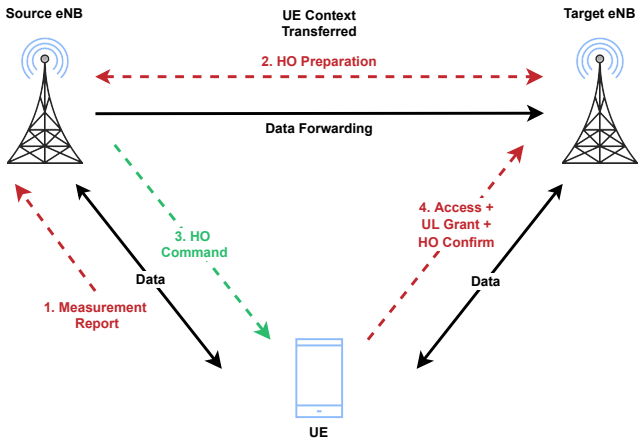


Figure 4.3: LTE Handover Mechanism. [Wang 19]

In the RRC connected or the Continuous Reception (CR) state, the radio can actively send data in the uplink and downlink without waiting for an 'on' cycle.

After an inactivity period (T_{IN}) ranging from 0.32-2.56 seconds, the RRC state transitions to Short DRX. In the short DRX state, the UE occasionally monitors the channel with periodic 'on' intervals of TSC. The authors set the T_{IN} value to 200 ms and the T_{SC} value to 6 ms for the Short DRX state.

Inactivity in the Short DRX state after a specific duration triggers a transition to the Long DRX state. In the Long DRX state, the radio switches on less frequently. The inactivity timer (T_{IN}) from short DRX to Long DRX state is 2298 ms, and the cycle period T_{LC} is set to 12 ms.

After a few Long DRX cycles with no transmission, the RRC switches the device state to RRC_IDLE. The inactivity timer is 7848 ms for this transition. In the RRC_IDLE state, the idle cycle period (T_{IC}) is 50 ms.

LTE Handovers

LTE Handover is a mechanism that is triggered when a UE moves from a cell managed by one eNB to another in the face of diminishing signal strength, as shown in **Figure 4.3**. It can also occur due to network management decisions. LTE Handovers follow the break-before-make paradigm, i.e., the existing connection breaks before a new connection initiates.

[Cech 21] implements the handover mechanism in the MoonGen LTE emulator script. Two significant parameters are modeled: the Handover Interruption Time (HIT) duration and the Handover Frequency. The author achieves this by scheduling a Handover with

a frequency sampled from a normal distribution. During a Handover Event, the forward thread of the LTE emulator stops processing any frames and restarts packet processing after the completion of the HIT duration. As a result, the frames queue in the packet-sized queue, and if it fills up, frames are discarded. Handovers induce increased packet latencies.

4.2.3 Modeling Handover Interruption Time (HIT)

The HIT value is taken from [Han 15], where the authors model the HIT as a function of the number of people/cell/minute using statistical data obtained from real-world cellular measurements. The measurements reveal the HIT value on average ranges from 17.1-19.34 ms with a standard deviation (SD) of 0.516-1.202 ms. A normal distribution can model the observed data with a low variance around the mean.

Additionally, [Cech 21] conducts measurements to gauge the handover behavior of a cellular network from an aerial vehicle. The captured data has a higher variance than observed in [Han 15], with a mean duration of 20.01 ms and SD of 195.13 ms. The handover duration is primarily clustered around 1.9 ms, 3.8 ms, and 200 ms.

[Cech 21] argues that the "black-box" characteristics of the LTE infrastructure and the handover management prevented them from drawing any definitive conclusions from the captured data and suggests a multimodal distribution or a mixture model is required to fit the data.

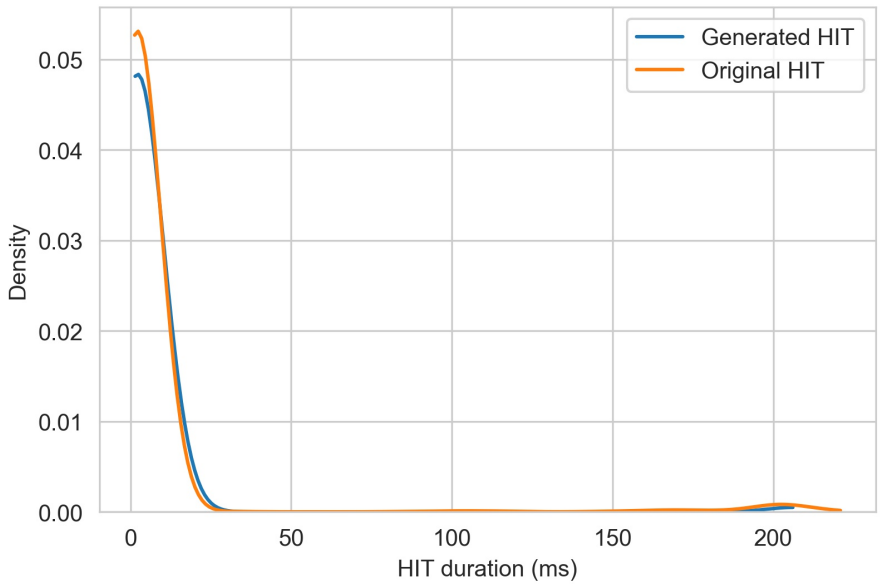


Figure 4.4: Sampling a HIT duration from the modeled Gaussian Mixture Model.

Table 4.1: Gaussian Mixture Model Parameters.

Gaussian [k]	**Mean** [μ_k]	**SD** [σ_k]	**Weights** [p_k]
1	1.9681	0.2484	0.8383
2	3.9050	0.9414	0.1214
3	9.3879	5.6246	0.0095
4	203.1832	21.2318	0.0174

We revisit the handover data collected by aerial LTE measurements and model the Handover Interruption Time (HIT) as a Gaussian Mixture Model (GMM), which learns its parameters using the Estimation Maximisation (EM) algorithm. The EM algorithm is a form of unsupervised learning. We show that the GMM fits the real-world HIT dataset, and we proceed to implement the model in the MoonGen LTE emulation script.

A multimodal dataset has several different modes or regions of high probability mass with areas of smaller probability mass in between. The HIT duration has multiple modes. Hence it can be represented as a mixture of several components, where each element has a simple parametric form such as a Gaussian [Gros 22]. Chapter 7.3 provides a mathematical background for the computation.

The PDF is a convex combination, or weighted average, of the PDFs of the component distribution. **Table 4.1** represents the parameters of the GMM with a cluster size of four. Equation 7.5 and 7.6 describes the PDF of HIT values. **Figure 4.4** shows the HIT duration from the original dataset and the generated values from the GMM with E-M. The generated values closely fit the initial distribution [1].

4.2.4 Emulation Setup

[Stra 20] employed the Emulab testbed with three nodes, i.e., the client, the server, and the MoonGen emulation node, to conduct experiments with no traffic shaping between nodes. They also employed a Data Acquisition and Generation (DAG) card to make accurate and precise packet timings measurements and remove any clock synchronization issues on the distinct physical nodes. They tap the experimental interface on the client and the server and record traffic in both directions. All nodes part have an 8-core CPU running at 2.10GHz, 16GB of RAM, and four Intel I350 Gigabit Ethernet interfaces on the experimental network. They employ one NIC on the client and the server and two NICs on the MoonGen emulation node. Additionally, the setup uses 1GB hugepages (default 2MB) which is a requirement for DPDK for faster and contiguous memory accesses.

[1] We ignore outliers with HIT over 250 ms when designing the model.

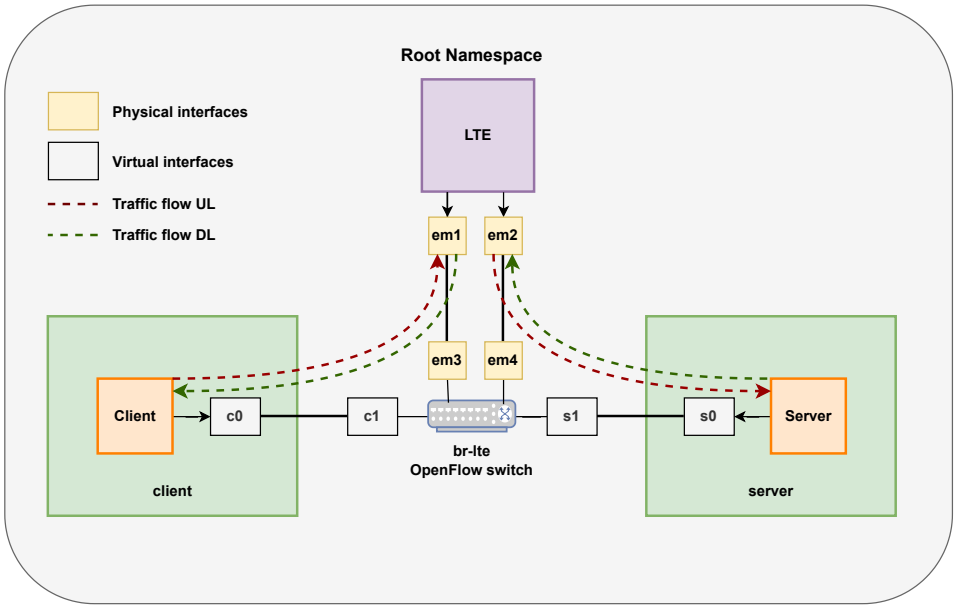


Figure 4.5: MoonGen LTE Emulator Setup on a single physical machine employing four physical NICs and virtual networking components.

Our configuration goal had to consider the possibility of integrating the OpenSAND testbed used for Satcom emulation with the LTE emulator, key to Multipath transport measurements. And in an ideal case, the joint LTE-Satcom testbed should run on the same physical machine to enable seamless orchestration of the measurement infrastructure.

To that end, we employ a setup consisting of network namespaces and Open vSwitch with OpenFlow rules configured to run all three experimental nodes, i.e., the client, the server, and the MoonGen emulation node on the same physical machine.

Figure 4.5 shows the setup of the MoonGen LTE emulation script. The physical machine has four 6-core Xeon CPUs running at 3.4GHz, 128 GB of RAM, a pair of Intel I350 1G, and Intel 82599 10G NICs. The client (UE) and the server (eNB) run on their respective namespaces, and the MoonGen emulation script runs on the root namespace. The client and the server use the virtual Ethernet (VETH) pairs to bridge their respective network namespaces to the root.

The Intel 82599 10G NIC ports, **em1, and em2**, are bound to the Data Plane Development Kit (DPDK) Poll Mode Driver (PMD) for emulation and process packets in the userspace. The Intel I350 1G NIC ports, **em3 and em4**, use the standard kernel drivers. The ports **em1 and em2** connect to ports **em3 and em4** via a direct physical connection over an Ethernet crossover. It allows the DPDK applications on userspace to communicate with the Linux kernel stack. The Open vSwitch taps the kernel interfaces and

Table 4.2: OpenFlow Rules configured on the Open vSwitch.

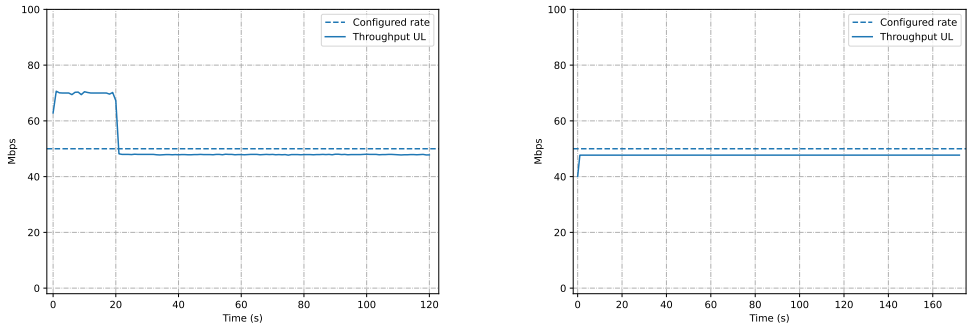
No.	Rule
1	ovs-ofctl add-flow br-lte in_port=c1,dl_type=0x0800,actions=output:em3
2	ovs-ofctl add-flow br-lte in_port=s1,dl_type=0x0800,actions=output:em4
3	ovs-ofctl add-flow br-lte in_port=em3,dl_type=0x0800,actions=output:c1
4	ovs-ofctl add-flow br-lte in_port=em4,dl_type=0x0800,actions=output:s1
5	ovs-ofctl add-flow br-lte in_port=c1,dl_type=0x0806,actions=output:s1
6	ovs-ofctl add-flow br-lte in_port=s1,dl_type=0x0806,actions=output:c1

enables virtual switching between the namespaces and the root employing the configured OpenFlow rules shown in **Table 4.2**. For example, the traffic flow from the client to the server consists of a route through the LTE emulator, which emulates the LTE link. A dashed line shows the path for uplink (UE-eNB) and downlink (eNB-UE) traffic in the **Figure 4.5**.

4.2.5 Validation

We extensively benchmarked the latency and bandwidth emulation features of the LTE emulation script before integrating it with the joint testbed. A previous setup [Cech 21] on a virtual machine (VM) experienced problems with MoonGen's rate-limiting features. It includes a warm-up phase in which the emulator's rate-limiting kicks off with a delay directly proportional to the emulated rate. The LTE emulation script can emulate bandwidth heterogeneously in each direction. The author also observes a counter interaction between the uplink and downlink streams in the case of duplex tests.

After consultations with the author of the original MoonGen LTE emulation script, we could source the problem and attribute it to the way MoonGen's CRC-based Rate Control



(a) Setup on VM with a warm-up phase **(b)** Bare-metal setup with no warm-up phase

Figure 4.6: Throughput benchmarks from uni-directional transfers.

works. To precisely control the inter-packet timings on hardware and operating system, MoonGen employs dummy frames with a bad CRC. It prepends dummy frames with bad CRC onto good frames, preventing frame buffering and packet bursts. An upstream switch or the receiving NIC then discards the dummy frames. The NIC continuously transmits frames at full capacity and prevents the card from buffering and bursting when good frames dispatch at less than line rate [Stra 20].

The setup on the VM employed a restructured MoonGen LTE emulation script to run on a single NIC at the emulation node. It used supplementary OS threads to do packet filtering and traffic shaping. But as it turns out, the setup is incompatible with the way MoonGen does rate-limiting and hence the observed artifacts and counter interactions with duplex traffic.

Figure 4.6 shows the lack of the unwarranted warm-up phase on the bare-metal setup. The emulator instantaneously respects the emulated rate. Here the LTE target rate is set to 50 Mbps. The iPerf3 sender generates UDP traffic at 20 Mbps more than the configured limit.

The tools used to conduct latency and throughput benchmarks include ICMP ping and iPerf3. The latency emulation accuracy of the LTE emulator is tested by successively sending 300 ICMP ping packets with an interval of 0.01, taking the average of the reported round-trip times (RTTs). The emulated latencies are between 10 and 60 ms with a step size of 5 ms. A low bandwidth stream is generated in the background using iPerf3 to warm up the LTE DRX states before latency measurements. **Figure 4.7** plots the result of the measurements. The difference between the measured and the configured RTT is 0.146 ms on average with a SD of 0.00454 ms. [2]

The following measurements evaluate the traffic shaping ability of the MoonGen LTE emulation script. Here Uplink (UL) refers to data transfer from the client (UE) to the server (eNB) and Downlink (DL) from the server (eNB) to the client (UE). By default, the emulator includes the IP and the Ethernet overhead in its throughput regulation [Stra 20].

The performance of the LTE emulator is evaluated with duplex traffic. The DL is configured to rate-limit traffic to 40 Mbps while the UL targets a range from 10 to 100 Mbps for UDP and 30 to 120 Mbps for TCP iPerf3 tests. **Figure 4.8(a)** shows the results for tests employing UDP segments. An iPerf3 sender generates traffic at a rate that is 20

[2]The results employ the 10G and 1G NIC in each direction on the emulation node (em1 and em2) for MoonGen. Later, we restructured the testbed to use homogenous 10G NICs in both directions for DPDK and observed similar results.

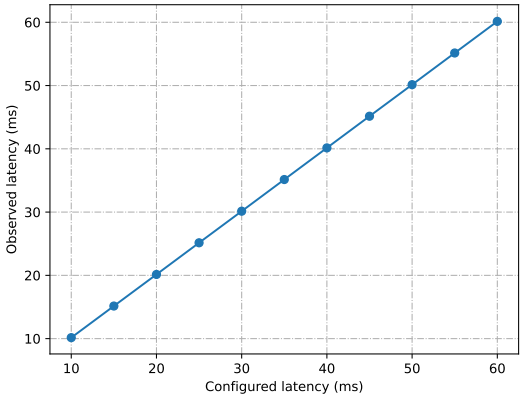
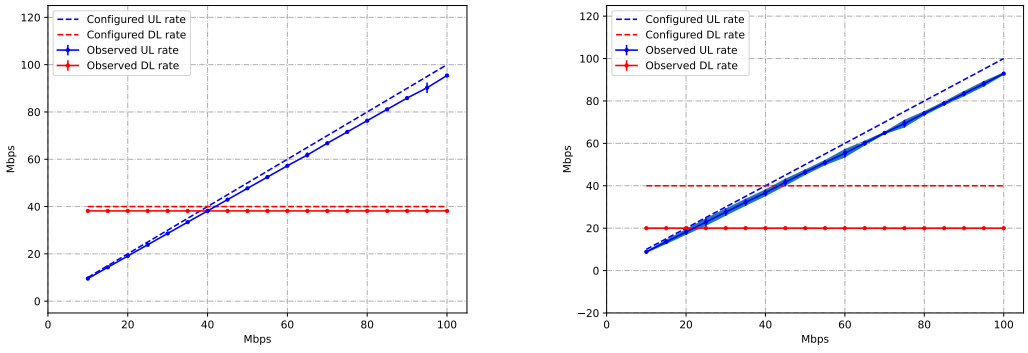


Figure 4.7: Latency benchmarks show the difference between the measured and the configured RTTs.

Mbps more than the configured rate. Both UL and DL traffic are shaped correctly.

For connection-oriented protocols like TCP with data flows in one direction and acknowledgment (ACK) traffic in the reverse direction, there should be enough space for both data and ACKs in the emulated links, especially when testing with duplex traffic. Since the emulator does not differentiate between data and ACK segments, the TCP feedback mechanism is delayed if it drops ACKs when rate-limiting. **Figure 4.8(b)** shows the result of tests employing TCP segments. The iPerf3 sender generates traffic at 20 Mbps less than the configured rate in the DL, and hence the emulator's rate-limiting is not triggered. Meanwhile, in the UL direction, the iPerf send rate is 20 Mbps more than the configured rate on the emulator. Hence the rate-limiting is only triggered in the UL direction. The plot shows that the emulator can keep up with the send rate in the DL, with a slight variance in the UL when the send rate exceeds the configured rate.



(a) Throughput benchmark with UDP duplex traffic

(b) Throughput benchmark with TCP duplex traffic

Figure 4.8: Throughput benchmarks for LTE emulation script running on the bare-metal hardware.

4.3 Satcom emulation

We employ the OpenSAND Satcom emulator framework to emulate the satellite link. The framework is a derivative of work by [Brun 19], [Beil 21], and [Pola 22].

[Beil 21] develops an automated measurement testbed for Satcom networks to measure the performance of transport protocols such as TCP and QUIC over long delay networks. It builds up the work of [Brun 19] and employs the OpenSAND Satcom emulation tool. OpenSAND emulates an end-to-end satellite communication (Satcom) system, especially the Digital Video Broadcasting - Return Channel via Satellite (DVB-RCS2), employing multiple user terminals and gateways and a satellite emulation node [Thal 22]. The testbed can emulate multiple satellite orbits and measures the goodput, congestion window (CWND) evolution, connection establishment times (CON-ESTs), and times to first byte (TTFBs) of the protocols mentioned above. It employs the use of Performance-enhancing proxies (PEPs) with the help of the PEPsal tool [Cain 06] for TCP and qperf tool for QUIC.

[Pola 22] extends the Satcom testbed to enable application layer (HTTP/3) measurements. The ability to emulate packet losses in the satellite backhaul network, and ground delays using netem instances, exposes OpenSAND's variable delay emulation via an API.

4.3.1 Setup

Figure 4.9 shows the environment created to emulate the Satcom using network namespaces. A Linux network namespace is a logical copy of the network stack with its routes, network devices, and firewall rules [Eric 22]. Virtual Ethernet (veth) pairs bridge the network namespaces.

The measurement tools (ICMP ping, iPerf3, and qperf) run at the end hosts, i.e., osnd-cl (client) and the osnd-server (server). The osnd-st (terminal), osnd-sat (satellite), osnd-gw (gateway) are the OpenSAND emulation entities that emulate the Satcom network. The OpenSAND entities are designed to run on separate physical machines and connect to the same emulation network for internal communication. The osnd-emu (emulation) namespace enables this. The measurement traffic does not pass this network—the data relating to the emulation flow through the tap interfaces on the satellite terminal and the gateway connected to the proxies. This interconnect is via a Linux bridge and the configured IP routes.

The PEPs run at the osnd-stp (pep) and osnd-gwp (pep) namespaces. If the PEP is

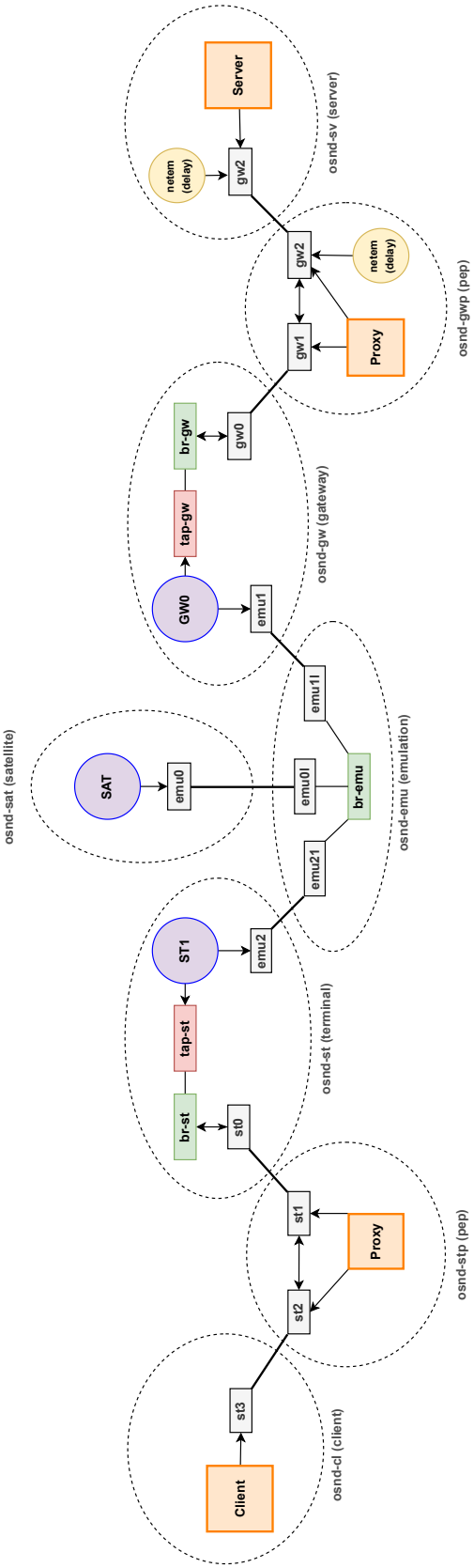


Figure 4.9: OpenSAND Satcom emulation platform configured using network namespaces.

disabled, they act as gateways and transparently forward packets without splitting the connection.

4.3.2 Parameters

The OpenSAND Satcom testbed provides various configuration options to gauge the performance of transport and application protocols under varying link conditions and setups. The link configured via command-line parameters consists of three types of parameters in each scenario. Environmental parameters control the conditions of the experiment, such as the signal attenuation and loss, modulation and coding (MODCOD), and satellite orbit (LEO, MEO, GEO).

Transport parameters deal with the configuration of the transport protocols, such as the congestion control (CC) algorithms and buffer sizes.

Measurement parameters control what is measured. The parameters consist of the transport layer protocols tested, the number of execution runs, and whether to enable priming the Satcom link with a ping before the measurements start.

4.3.3 Emulation Script

An emulation script controls the OpenSAND Satcom emulation. It helps set up and tear down the measurement infrastructure for each measurement run. Each scenario can consist of multiple measurement runs, and the setup phase configures measurement parameters. Once the environment is up, the measurements can start. After which, the script tears up the environment. It ensures that a new set of experiments is not affected by the remnants of the previous runs, such as partially filled buffers.

An accompanying post-processing script written in Python batch processes the results and generates plots. A detailed explanation of the emulation script and the measurement parameters and values follows in Chapter 5.

Next, we propose multipath link emulation framework employing the heterogenous Satcom and LTE links. We augment the OpenSAND Satcom testbed to include the MoonGen LTE emulation script and provide an abstraction to execute multipath transport measurements on the joint testbed.

4.4 Heterogenous Link Emulation

4.4.1 Testbed Setup

We employ a containerized environment creating ten Linux network namespaces for the joint emulator, each with its function, enabling us to run the entire testbed on a single physical machine. **Figure 4.10** gives a architectural overview of the testbed.

4.4.2 Emulation Environment

The emulation environment consists of namespaces that behave as end-hosts, proxy nodes, emulation nodes, and delay nodes.

End-hosts

The end-hosts exist at either end and host the measurement tools. The client **(osnd-moon-cl)** is multi-homed with access to both the LEO and the Satcom networks. We configure appropriate routes such that the client can exploit both the interfaces and send traffic simultaneously with MPTCP protocol. However, the server **(osnd-moon-sv)** is not multi-homed. Instead, an additional gateway **(osnd-moon-sv)** connects the LTE and the Satcom links. It is a design choice as the MPTCP assumes that one or both hosts are multi-homed and multi-addressed [Ford 20]. It allows us to emulate delays between the gateway **(osnd-moon-svgw)** and the server **(osnd-moon-sv)** using netem instances. The testbed configures a MPTCP scheduler and CC algorithm at the end-host (sender).

Proxy nodes

Performance Enhancing Proxies (PEP) can transparently run on the **(osnd-stp)** and the **(osnd-gwp nodes)** of the Satcom link. The nodes act as a gateway without a PEP using routes configured with the iproute2 utility.

Currently, PEPsal is integrated as a TCP-based performance enhancing proxy (PEP), used for both single-path and multipath measurements [Cain 06]. Additionally, the testbed supports qperf, a QUIC based load generation tool in a proxy mode on a single path.

Delay nodes

In addition to the link delays emulated individually by both the emulators, we add ground delay emulation at both the client and the server-side. The use case of remote piloting aerial vehicles aims to model additional delays experienced by the user equipment due to varying altitudes on LTE. Similarly, it represents delays experienced between the server

gateway where the LTE and Satcom networks interconnect and the ground stations located on the global Internet.

The OpenSAND framework sets ground delays at the **osnd-gwp** and the **osnd-svgw** namespaces for each flow direction. The testbed additionally provides an ability to configure ground delay at the **osnd-moon-cl, osnd-moon-clgw** for the LTE and **osnd-moon-cl, osnd-stp** for the Satcom on the client-side. And at **osnd-moon-svgw, osnd-moon-sv** on the server-side.

Emulation nodes

The OpenSAND emulation entities run on the **osnd-st, osnd-gw, osnd-sat, and osnd-emu** namespaces. The LTE emulation runs on the root. The emulation nodes are responsible for shaping the traffic flowing between the end-hosts according to the configured emulation parameters.

4.4.3 Emulation Script

As shown in **Figure 4.10**, the multipath emulation script bases its architecture on the existing OpenSAND emulation framework. On top of that, we compose scripts to make the MoonGen LTE emulation work with the containerized environment utilizing Linux network namespaces. Afterward, we integrate and abstract the heterogeneous emulation frameworks to provide a unified testbed to run end-to-end network measurements.

The Bash scripts are modular to enable fast prototyping and provide flexibility to the testbed. It allows for the separation of concerns as each script set is responsible for a different function such as setup, teardown, and measure.

Setup phase

Three sets of setup scripts exist to build the Satcom, the LTE, and the integrated environment. We first independently set up the OpenSAND Satcom and the MoonGen LTE environment. Next, we reconfigure the end-hosts to adapt them for multipath operation. According to the employed routing strategy, it includes inserting appropriate network routes on end-hosts. The routing strategy includes single-path routing via the LTE or Satcom interface and multipath routing via both interfaces. In the case of multipath operation, the script configures the MPTCP path manager, congestion control, and scheduler.

Additionally, ground delay values are emulated on the delay nodes if specified in the configuration. Finally, the script boots the OpenSAND Satcom and MoonGen LTE emulators with emulator-specific configuration parameters.

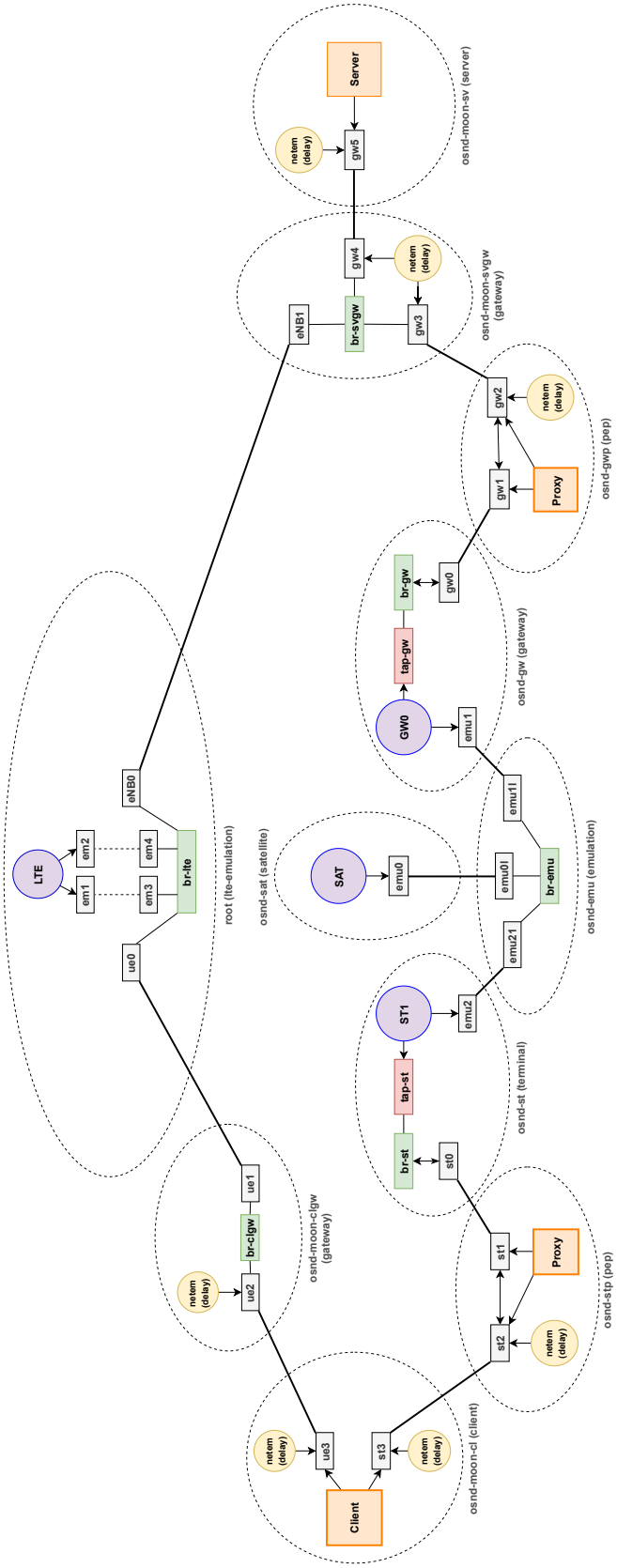


Figure 4.10: The end-hosts exist at either end where the client is multi-homed with access to the LTE network (top) and the Satcom network (bottom). The server is not multi-homed, and instead, an additional gateway connects the LTE and the Satcom links to the server. The link emulation runs parallelly on the emulation nodes (center). The ground delays are configured at both the server-side and the client-side gateways.

Measurement phase

After the testbed setup, the **execute-measurements.sh** script starts automated network measurements. It reads the configuration files and executes latency, goodput, timing, and QoE measurements as per the scenario. Multipath measurements capture packet traces on both the client interfaces and the server interface using libpcap, which the evaluation scripts analyze.

Teardown phase

Once the measurements finish, the teardown script gracefully tears down and cleans up the emulation environment. It ensures that any residual configuration on the previous runs does not affect future runs. All the network namespaces, including the virtual networking components such as the Linux bridges, virtual Ethernet pairs, OpenFlow switches, are removed.

4.4.4 Abstraction Layer

```
1    -N 1 -O LEO -r LTE -c cccc
2    -N 5 -O LEO -E 10,10 -r SAT -c rrrr
3    -N 10 -O LEO -E 10,10 -L 0.001 -r MP -S default -c lia -H -W -Y
```

Listing 4.1: Sample scenario configuration on the LTE-Satcom emulator.

We provide abstraction via configuration files and command-line parameters, customizing each measurement scenario. The LTE link parameters are relatively stable, exposing them via configuration files, and the command-line options configure a host of protocol and environment-specific variables. **Listing 4.1** shows a sample scenario configuration.

Environment parameters

The environment parameters define the environment of the emulation, specifically related to the Satcom link. The signal attenuation values emulate the varying link conditions, and however, we set it to zero to mimic clear-sky conditions. The orbit specifies the satellite orbit, either LEO, MEO, or GEO.

Transport parameters

The transport parameters are categorized into QUIC and (MP)TCP-specific. We only modify the MPTCP-specific parameters like the employed scheduler, congestion control, and the path manager for our measurements. The QUIC-specific parameters work on the single-path and do not interfere with the MPTCP-parameters.

Measurement parameters

The measurement parameters are independent of each scenario. They are used to configure delay on the OpenSAND emulator, capture logs, disable specific goodput or timing-related measurements, the seconds to prime the OpenSAND emulator, the number of runs for each scenario, and the ground delays.

It also includes the parameter to select the routing strategy, which enables single-path measurement over the Satcom or LTE links and multipath runs over both the links.

General parameters

The general parameters enable reading the scenario configuration file, printing the help message, script version, adding tags to the output folder, and showing system statistics.

4.4.5 Use cases

Single-path use case

The emulator provides a facility to carry out measurements on a single path. Single path measurements help gauge the performance of multipath protocols in terms of parameters such as the achieved bandwidth aggregation under the same emulated conditions.

Measurements on the single path can use either the LTE or the Satcom link. The emulator supports measuring transport layer Quality of Service (QoS) parameters with TCP and QUIC. Additionally, it can measure application layer Quality of experience (QoE) parameters utilizing HTTP/1.1 and HTTP/3 over QUIC.

Multipath use case

In the multipath context, the iperf application utilizes both the underlying heterogeneous LTE and the Satcom paths transparently with MPTCP. The metrics such as goodput, path latencies, packet losses, and retransmissions are measured.

4.4.6 Measurement and Analysis Tools

We employ the iPerf3 tool for goodput measurements. We provide an option to configure the generated application bandwidth differentially in the uplink and downlink directions according to the defined QoS requirements.

However, iPerf3 results only show the protocol performance at the MPTCP connection level. Thus, we integrate additional tools to dive into the MPTCP subflow-level performance metrics and analyze individual TCP flows.

We use tcpdump and libpcap[1] to capture packet traces on all end-host network interfaces and save them for further processing. It allows us to estimate subflow level link utilization, the per-packet RTTs, packet losses, retransmissions, etc.

The evaluation script is modified to work with the additional multipath testbed parameters and process the output of the traces captured during the emulation. We employ the Captcp[4] tool, a TCP analyzer for PCAP files, to parse subflow level goodput metrics. The tshark[5] tool extracts RTTs, packet losses, retransmissions, out-of-order packets, and duplicate acknowledgments.

The evaluation scripts process the emulation results and store them into a pandas[2] data frame arranged according to the configured scenarios. The py-gnuplot[3] library generates the plots as a time series. Additionally, we create empirical Cumulative Distribution Function (ECDF) graphs to compare multiple MPTCP schedulers, and congestion controls permutations.

4.4.7 LEO Variable Delay Simulation

We interface an existing LEO simulation framework to emulate variable packet delays on the OpenSAND Satcom emulator.

HYPATIA is a packet-level simulation framework employing the ns-3 discrete event simulator [nsna 22]. It aims to model existing and upcoming LEO constellations' dynamic but predictable network behavior. It studies the latency, link utilization fluctuations over time, and implications of these variations for congestion control and routing [Kass 20].

It enables modeling arbitrary scenarios with a choice of LEO satellite constellations, ground station (GS) location, routing, congestion control, and queuing behavior. The simulation generates RTT traces for individual flows in the simulation. We implement a utility that takes the raw traces and processes the RTT values to be compatible with OpenSAND emulation. Additionally, we can record packet data and store it in a PCAP file at the simulation's sender, receiver gateways, and satellite nodes.

We make a few abstractions to realize this. Hypatia generates round-trip time values during a simulation, while OpenSAND accepts one-way delay (OWD) values set on the ST and the GW. For simplification, we assume that the delays are symmetric.

[1]tcpdump and libpcap - https://www.tcpdump.org/
[4]captcp - http://research.protocollabs.com/captcp/
[5]tshark - https://www.wireshark.org/docs/man-pages/tshark.html
[2]pandas - https://pandas.pydata.org/
[3]py-gnuplot - https://pypi.org/project/PyGnuplot/

The ns-3 simulator generates timing and RTT values with nanosecond precision. However, OpenSAND accepts timing in seconds and packet delays in milliseconds. Due to this limitation, we average the delay values over one-second intervals.

Chapter 5 discusses the results obtained from the single path and Multipath TCP measurements over the LTE-Satcom emulator.

Chapter 5

Results

We employ the LTE-Satcom joint testbed to measure the performance of the MPTCP transport and study its feasibility to support remote piloting of aerial vehicles. We take a holistic approach and study all possible permutations of the selected congestion control and scheduling algorithms to evaluate their feasibility toward meeting the QoS requirements of the remote piloting scenario. Therefore, we configure the link parameters following real-work measurements to represent the scenario in an emulator environment correctly.

5.1 Testbed Configuration

We repeat all Transmission Control Protocol (TCP) and MPTCP measurements for **ten runs** of **30 seconds**. The results represent all phases of TCP congestion control, including slow-start, congestion avoidance, and congestion detection. The MPTCP **v0.95** and the LTE-Satcom emulation script version **2.2.4** is used for the measurements. The kernel **defaults** for TCP are retained.

We emulate the Satcom network's LEO link. Additionally, OpenSAND is primed for 5 seconds with Internet Control Message Protocol (ICMP) pings before starting the measurement. Priming mitigates an artifact of the OpenSAND emulator, which causes increased initial round-trip time (RTT) and connection establishment times on a freshly created environment [Beil 21]. The LTE Handover emulation is not enabled to keep the analysis straightforward.

5.1.1 Link Configuration

The section lists the parameters configured for the emulation on the heterogeneous LTE and Satcom links, e.g., bandwidth, baseline latencies, packet losses, and ground delays. Here, uplink (UL) refers to the link between the client to the server. And downlink (DL) refers to the connection between the server to the client. **Table 5.1** lists a summary of the chosen values.

Bandwidth

- **LTE.** We emulate rates of **38 Mbps** and **40 Mbps** in the DL and UL directions, respectively. The chosen values are a good representation of achievable LTE link capacity per user based on previous measurements [Stra 20].
- **Satcom.** The link configured with a Modulation and Coding (MODCOD) scheme achieves a link capacity of roughly **19 Mbps** in the UL and DL [Beil 21]. The forward band with a total bandwidth of 50 megahertz (MHz) on the signal carrier is the OpenSAND default. The return band splits into two carrier signals. The premium carrier is removed due to a lack of concurrent or prioritized traffic in the emulation, and it results in a single carrier with a bandwidth of 19.98 MHz [Beil 21]. The MODCOD schemes employed and available link capacities are Satcom operator-specific and proprietary. Without reliable knowledge, we choose to keep the defaults.

Baseline Latencies

- **LTE.** The LTE link emulates heterogeneous baseline latencies of **10 ms** and **30 ms** in the DL and UL, respectively. The values modeled after real-world measurements are a good approximation of latencies experienced by frames on the LTE network [Stra 20].
- **Satcom.** We emulate symmetric latencies modeled after an LEO Satcom link. We performed minutely traceroute measurements for 48 hours using the Starlink LEO

Table 5.1: A summary of configured values for the emulation.

Parameter	LTE	Satcom
Bandwidth	38 Mbps (DL), 40 Mbps (UL)	∼19 Mbps (DL, UL)
Baseline latencies	10 ms (DL), 30 ms (UL)	9 ms (DL, UL)
Ground delay	0 ms	10 ms (DL, UL)
Packet Loss	0.00006	0.00166
Concealed Loss	0.005	-

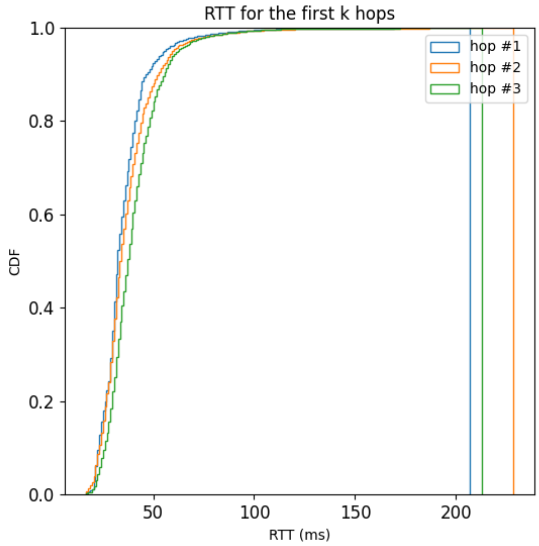


Figure 5.1: LEO Latencies on the Starlink network

network, where the Client Dish was located in Garching, and the Server in an AWS data center in Frankfurt. We estimate the first-hop mean RTT of **36 ms**, as shown in **Figure 5.1**. We configure symmetric delays of **9 ms** on the OpenSAND gateway (GW) and satellite terminal (ST) in each direction.

Ground Delays

- **LTE.** We do not configure ground delays on the LTE system. We argue that the LTE emulation script is modeled after application layer measurements [Beck 14] and takes into account the ground delays in the LTE backhaul network.
- **Satcom.** We estimate ground delays of **10 ms** in each direction from the second and third hop latencies during the Starlink measurements, as shown in **Figure 5.1**.

Packet Losses

- **LTE.** We set the concealed loss value to **0.005** [Stra 20] and estimate packet losses occurring at the backhaul from traces collected on the LTE network with an aerial vehicle [Cech 21]. We set packet loss to a value of **0.00006**.
- **Satcom.** The Satcom emulator currently does not emulate link-layer losses since we lack the prospect to measure them due to the lack of probes at either the ST or the GW on the shared medium between the LEO satellite and the dish. Hence we estimate transport layer losses from the end-to-end measurements carried out with the Starlink system and set it to a value of **0.00166**. We performed an iperf

measurement for 900 seconds (s) using the Starlink LEO network, where the Client Dish was located in Garching, and the Server in an Amazon Web Services (AWS) data center in Frankfurt.

5.1.2 MPTCP Configuration

The Linux MPTCP kernel implementation supports a modular structure for MPTCP path-manager, scheduler, and CC algorithms. The analysis consists of scenarios that are permutations of the CC algorithms and the schedulers configured at the sender. We believe this aids us in selecting not only the best permutation to achieve our goal but also in comparing and finding out the trade-off between different available combinations.

Path Manager

We configure the MPTCP path manager at runtime with the sysctl command, `net.mptcp.mptcp_path_manager`. The **full-mesh** path manager used for the measurements creates a full-mesh of subflows among all available subflows. We only configure a single subflow for each pair of IP addresses, although it supports multiple subflows [UCLo 22].

Congestion Control

We employ the CC algorithms introduced in Chapter 3. **TCP NewReno and TCP Cubic** for uncoupled and **LIA, OLIA, BALIA, wVegas** for coupled CC. The choice of the algorithms stems from their default implementation in the Linux MPTCP kernel.

Scheduler

The measurements employ the default MPTCP **Lowest-RTT first (LRF), round-robin, redundant, and BLEST** schedulers introduced in Chapter 3.

5.1.3 Traffic Generation

We take a first step towards analyzing MPTCP to support remote piloting operations. We program a single unidirectional TCP flow to generate a constant bit rate (CBR) traffic using the iPerf3 tool at a rate of **20 Mbps** in the DL direction from the server to the client, as discussed in Section 2.1. It aims to model the bulk data transfer like a live video streaming application between the aerial vehicle and the ground station. A unidirectional stream simplifies the analysis of the protocol's behavior. Without an additional flow in the UL, in which case, TCP Acknowledgments (ACKs) run in the DL and interact with the flow. The duplex analysis serves as a future research direction.

5.2 Baseline Performance

We execute a set of measurements with no emulated losses on both the LTE and the Satcom links. We observe that MPTCP congestion controls are sensitive to packet losses and influence the MPTCP scheduler's scheduling decisions. Hence we monitor and verify the link emulation and MPTCP configurations under lossless conditions without triggering the CC's packet loss behaviors.

Single-path TCP performance

First, we look at the single-path TCP (SPTCP) measurements over the LTE and the Satcom links. **Figure 5.2** shows the goodput graphs for SPTCP with TCP NewReno and TCP Cubic CC algorithms. We select uncoupled congestion controls for single-path analysis because MPTCP's loss-based coupled congestion controls fall back to TCP NewReno behavior with a single available path. Additionally, the testbed only supports NewReno and Cubic for SPTCP measurements.

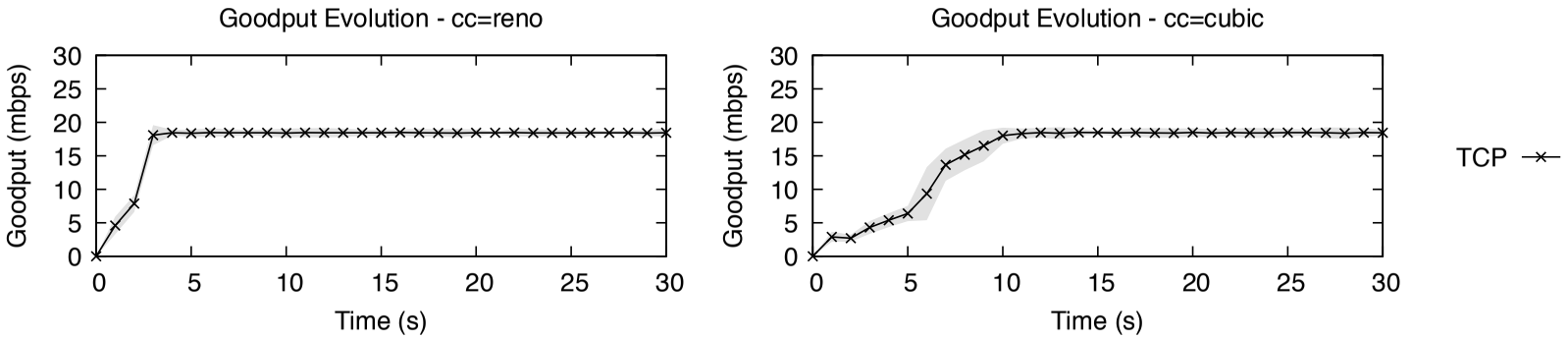
The ramp-up phase over the Satcom link is more prolonged when compared to the LTE link, and the reason for this is that the LEO link has a higher delay than the LTE link. But once TCP reaches its steady state, both links can keep up with the send rate.

Similarly, both congestion controls perform identically in the LTE link during the TCP ramp-up phase. Still, in the Satcom link, Cubic takes close to 10 seconds longer than NewReno, which takes 5 s to achieve the steady-state. It is due to the Hybrid slow start mechanism that TCP Cubic implements for high-bandwidth, and long-distance networks [Floy 04].

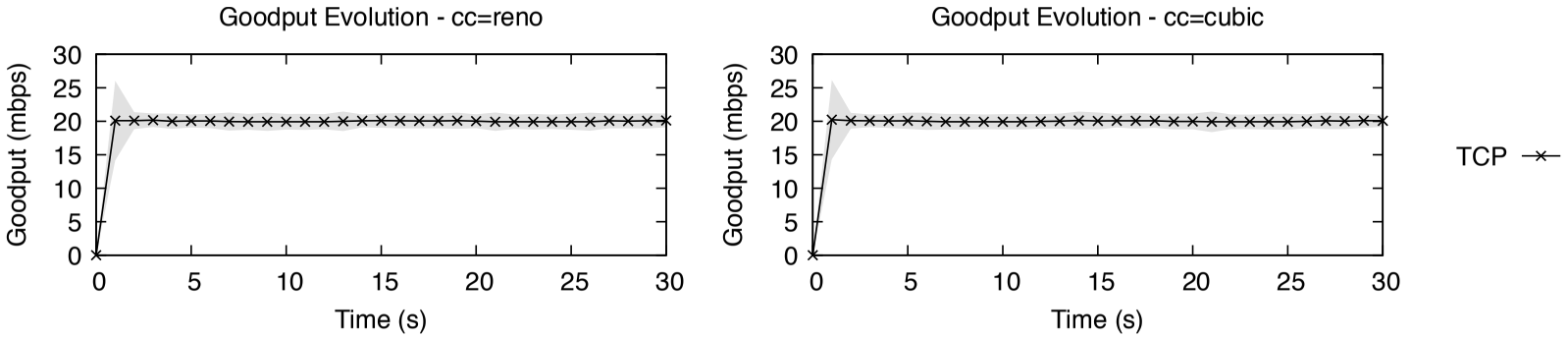
Multipath TCP performance

Additionally, we perform baseline measurements with MPTCP with no emulated losses. **Figures 5.3 and 5.4** show the performance of various MPTCP schedulers with the default LIA congestion control. All the investigated schedulers can keep up with the application sending rate, but they utilize underlying TCP subflows differently.

The lowest-RTT first (LRF) scheduler exclusively schedules segments onto the LTE subflow and avoids Satcom subflow. With enough capacity on the LTE link and no packet losses, it alone can keep up the application sending rate and achieve the desired goodput. Like the LRF scheduler, the BLEST scheduler employs the lowest RTT LTE subflow. As the name suggests, the round-robin scheduler schedules segments equally over the subflows in a cyclic fashion and uses them throughout the measurements. The flows converge in the plot at around 5 seconds.



(a) SPTCP goodput on the Satcom link with TCP NewReno and Cubic CC without emulated losses.



(b) SPTCP goodput on the LTE link with TCP NewReno and Cubic CC without emulated losses.

Figure 5.2: The baseline SPTCP goodput measurements without emulated losses on the LTE and Satcom links employing TCP NewReno and Cubic CCs.

The redundant scheduler duplicates segments and transmits them across subflows. It is expected behavior for redundant schedulers as they tend to transfer in total twice as much application data when compared with traditional TCP [From 16].

The redundancy affects the LTE subflow at the configured iPerf3 sending rate of 20 Mbps; the redundant scheduler initially transmits with packet duplication at 40 Mbps, close to the link capacity of the emulated LTE link leading to congestion on the link. The Linux implementation of a redundant scheduler offers two recovery modes if a redundant packet gets dropped.

The conservative mode retransmits the packet on the same subflow, leading to consistent TCP behavior on the link. The aggressive mode reuses the same TCP sequence number to transmit new data with a different global sequence number. We observe a conservative behavior being used in the face of congestion and retransmissions, as seen in the plot showing significant packet losses on the LTE link wasting bandwidth on both links without latency improvements. The aggregate link utilization is close to 50 Mbps as the flow progresses.

Figures 5.5 and 5.6 show the performance profiles of different MPTCP congestion control algorithms with the default LRF scheduler. All loss-based congestion control algorithms perform similarly in terms of goodput. In this scenario, LIA, OLIA, and BALIA default to TCP NewReno behavior for the subflow on the best path, i.e., the LTE.

Meanwhile, wVegas, a delay-based congestion control algorithm, utilizes both sub-flows throughout the measurement duration and sends segments on both subflows for its smoothed round trip time estimation (SRTT) and window adaptation scheme.

The last row in **Figures 5.5 and 5.6** show the measured per-packet RTT on both sub-flows plotted as an average over multiple runs. wVegas reduces per-packet RTT on the best path by a factor of close to 10 ms when compared with the rest. Also, since it employs the Satcom subflow, a steady RTT estimation is seen over the entire run without voids.

The section validates the optimal operation of the emulation testbed with the configured TCP and MPTCP options. Next, we analyze protocol performance on lossy links modeled after real-world measurements and study how the MPTCP congestion control algorithms and schedulers handle the connection to see if they meet the defined QoS parameters.

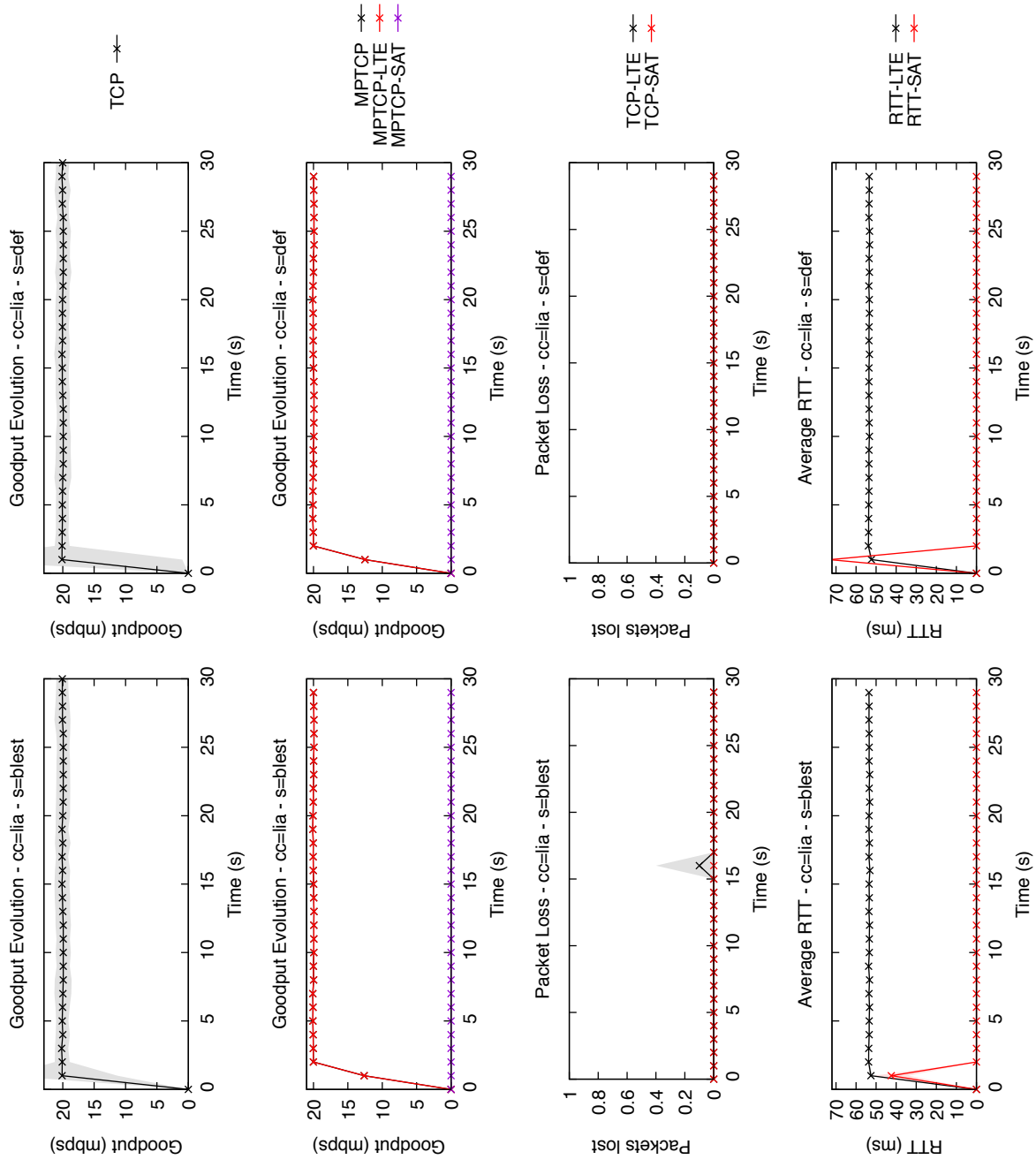


Figure 5.3: MPTCP baseline performance of the LIA CC with the BLEST and default LRF schedulers (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

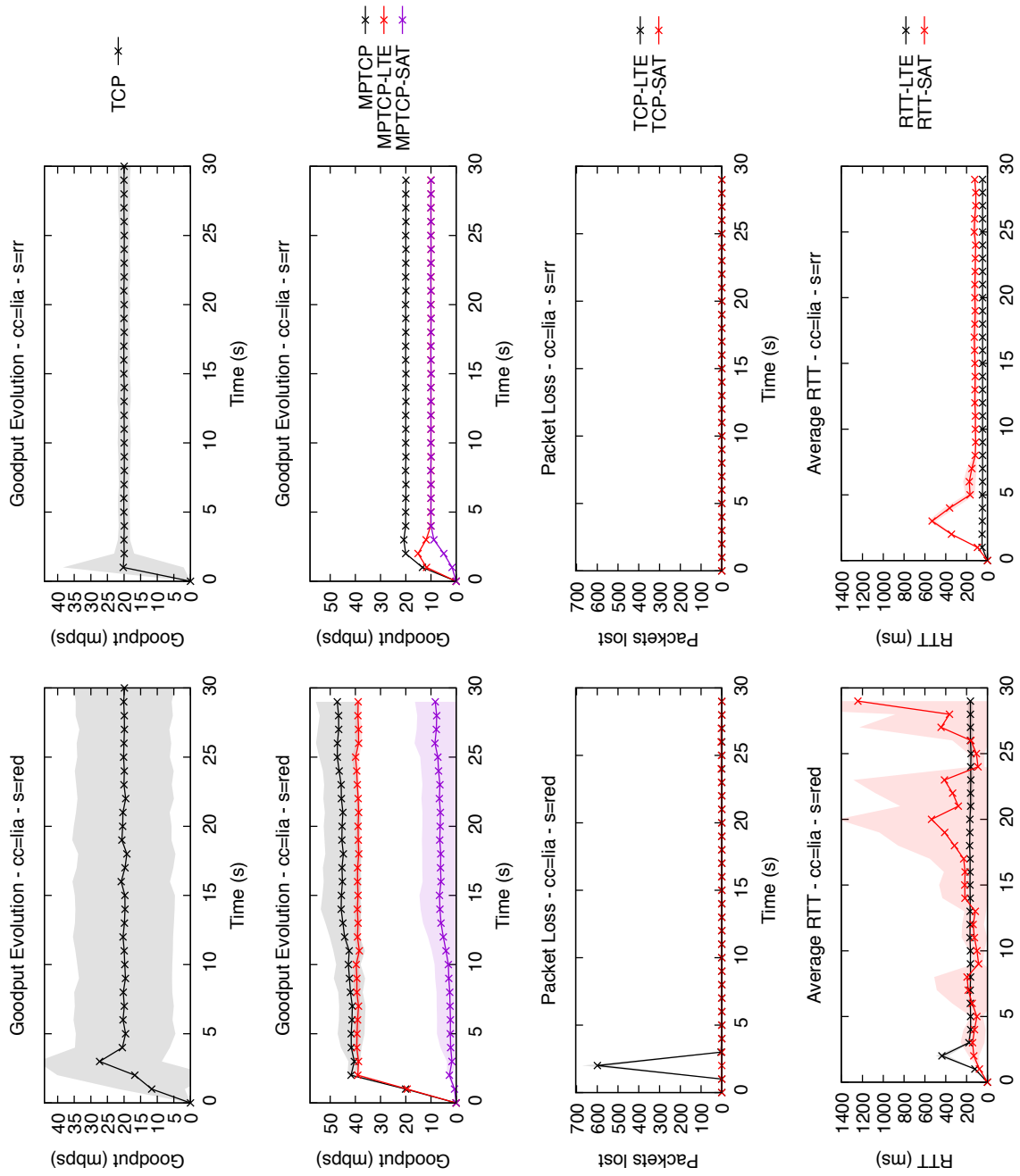


Figure 5.4: MPTCP baseline performance of the LIA CC with the Redundant and RR schedulers (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

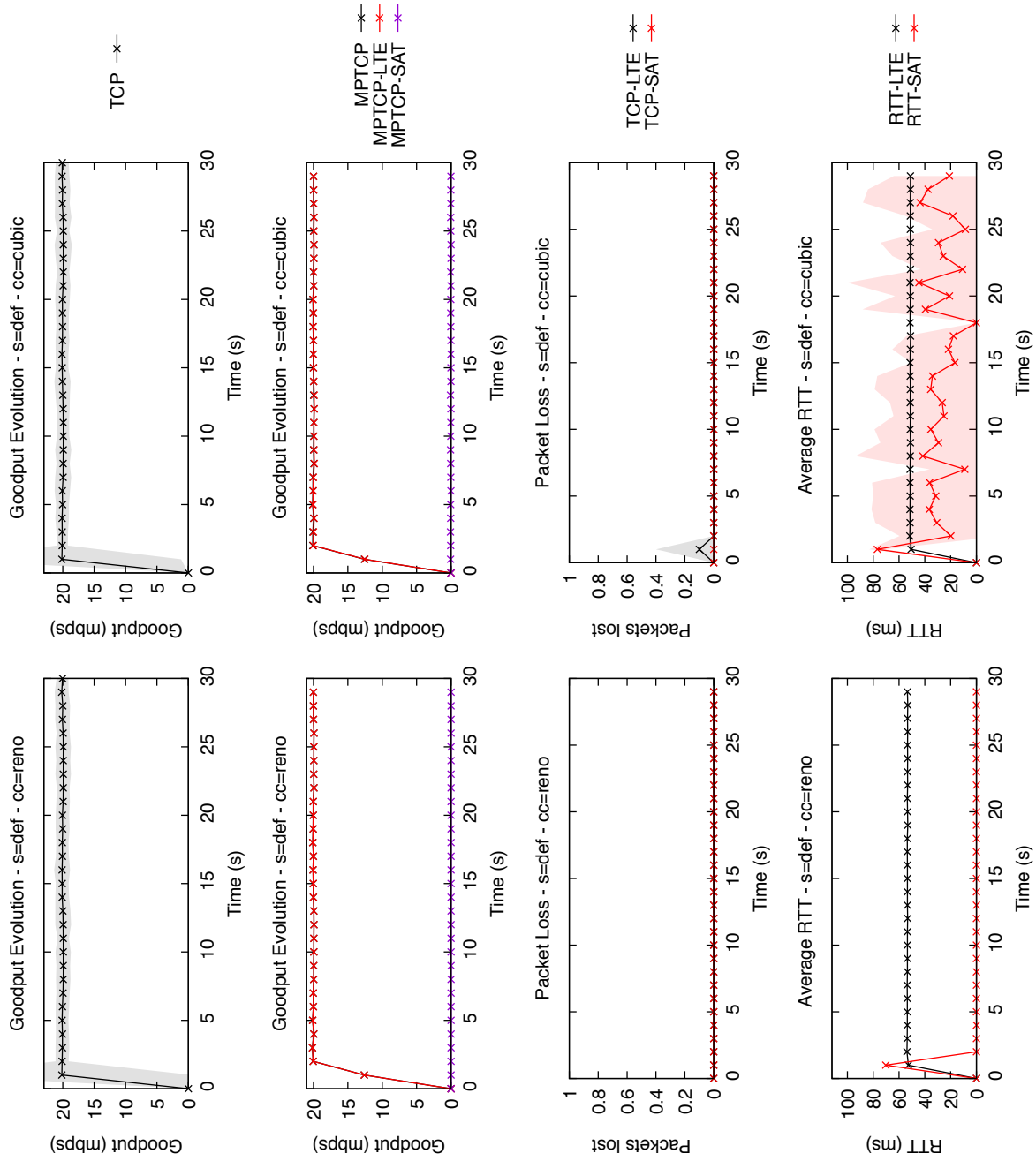


Figure 5.5: MPTCP baseline performance of the default LRF scheduler with the NewReno and Cubic CCs (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

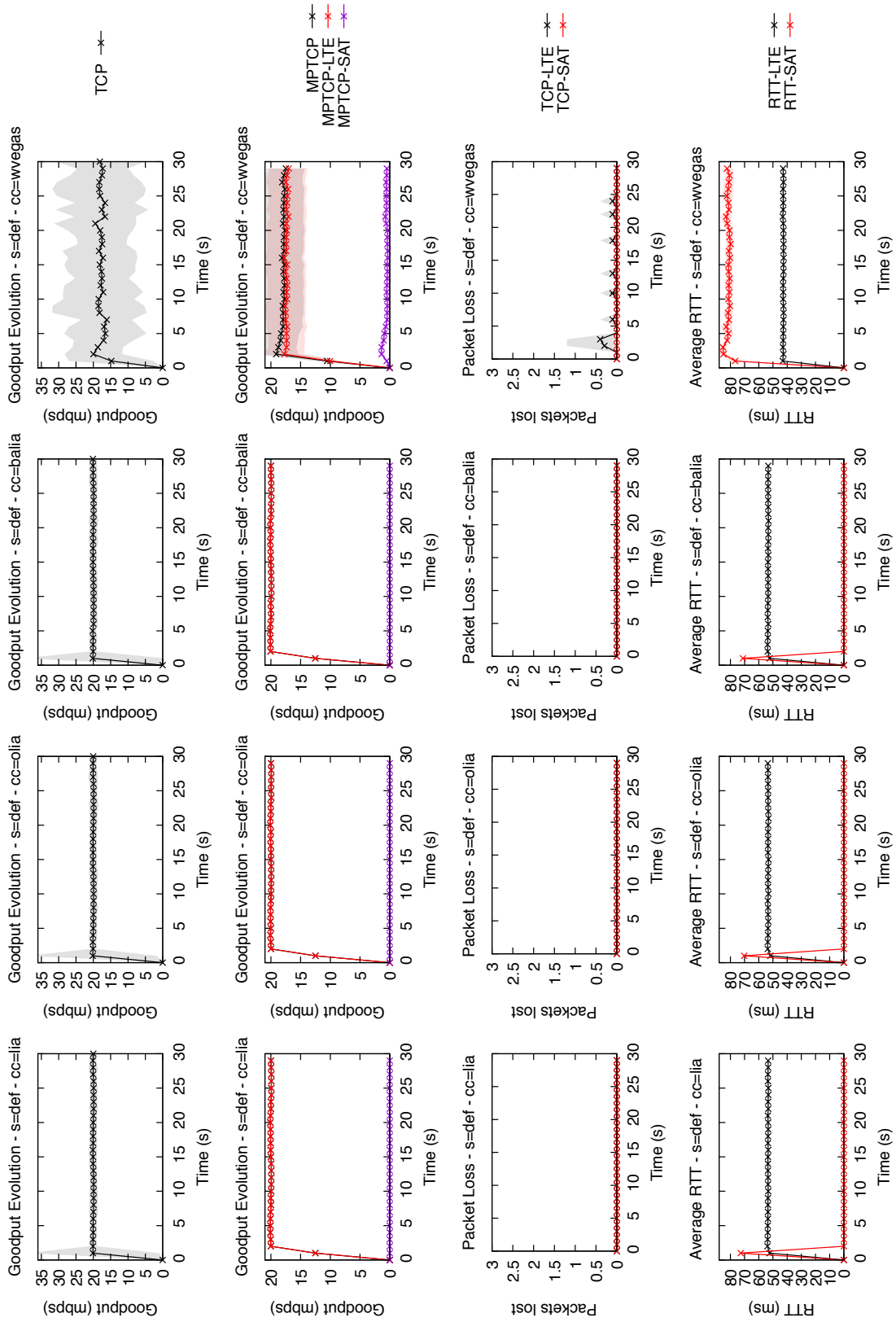


Figure 5.6: MPTCP baseline performance of the default LRF scheduler with the LIA, OLIA, BALIA, and wVegas CCs (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

5.3 Realistic Performance

Multipath TCP should fulfill the defined application-level QoS parameters to support remote piloting aerial vehicles in the realistically emulated scenario. The protocol should keep up with the prescribed application send rate of **20 Mbps** to carry video and telemetry traffic. We assume that by doing so, it can sustain the control traffic modeled as a CBR traffic of 5 Mbps. As mentioned previously, modeling the bidirectional flows is left for future work to simplify the analysis.

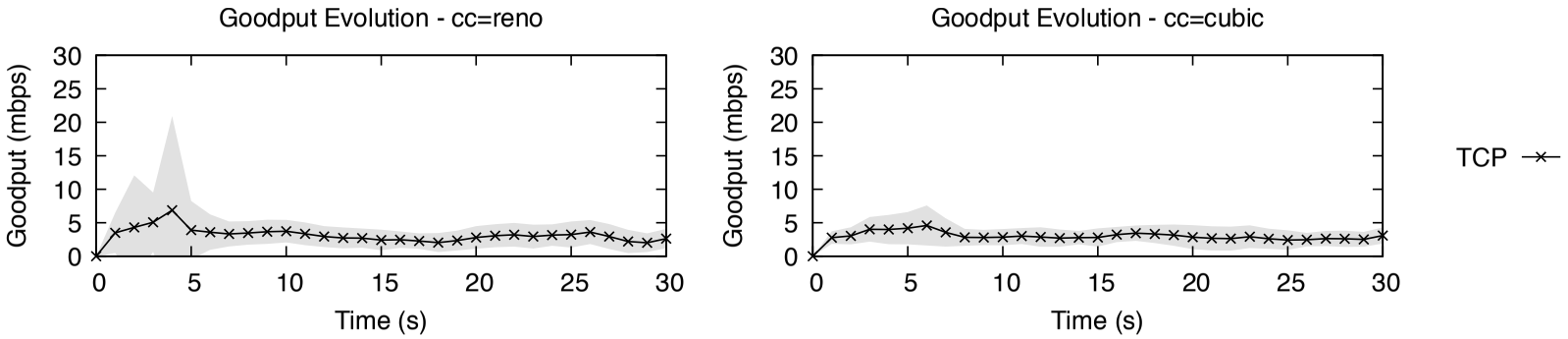
The PER threshold is 10^{-3} for the bulk of the UAS operation and 10^{-4} for telemetry traffic that stems from UAS during take-off and landing. The modeled cellular LTE links with a PER of $6X10^{-5}$ in principle meets the threshold. However, the LEO Satcom link with PERs modeled at $1.6X10^{-3}$ does not meet it for all stages of the operation. Note that the PER estimation for the emulation is abstracted and derived from the transport layer losses due to the constraints as mentioned earlier and may not represent the actual losses at the link layer.

MPTCP should fulfill the latency threshold of **250 ms** defined in the QoS requirements. Any configuration that exceeds this value on either path is deemed unfit for our use case.

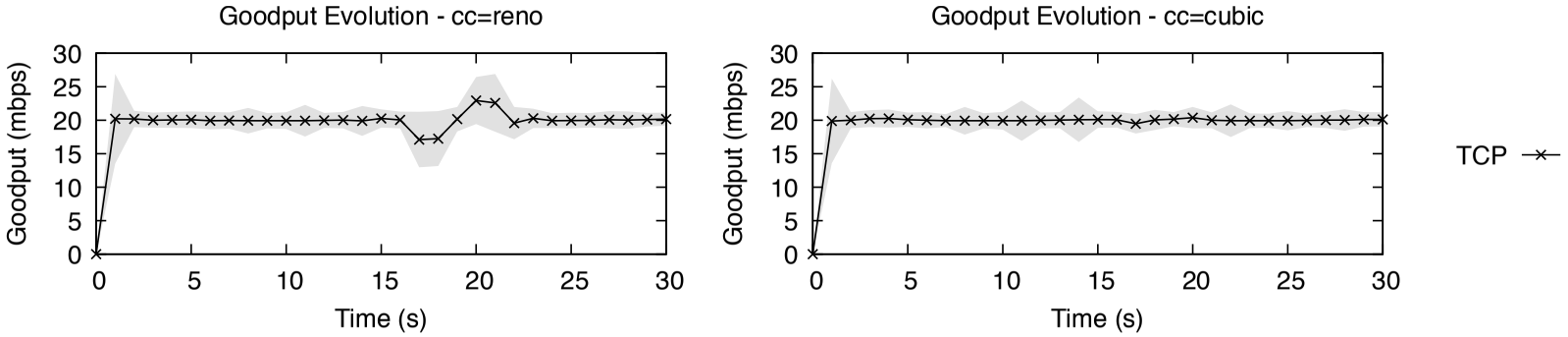
We now look at the performance of SPTCP and MPTCP with emulated packet losses. The packet error rates (PER) on the Satcom link are an order of magnitude greater than the LTE link, and hence the share of bandwidth on the LTE link is higher than Satcom with MPTCP. Before we start the analysis, the performance of both links under SPTCP is to be noted, as described below. The underlying TCP subflows limit MPTCP performance to be compatible with existing network infrastructure and prevent protocol ossification.

Single-path TCP performance

Figure 5.7 shows the SPTCP results with an emulated loss of **0.00006** and concealed loss of **0.005** on the LTE link and **0.00166** on the Satcom link. With low loss probabilities and loss recovery on the link layer, LTE can keep up with the application send rate and offer higher goodput with both congestion control algorithms. However, TCP suffers on the Satcom link due to a high amount of packet loss and resulting retransmissions. A higher latency also impedes loss recovery, and it is unable to keep up with the application sending rate. [Pavu 20] evaluates the performance of TCP under adverse conditions GEO links using the OpenSAND emulator. The authors employ short-lived flows and measure TCP goodput under varying loss rates logarithmically distributed from 1×10^{-9} (rare packet losses) and 1 (all packets lost). They achieve similar degradation in performance for the packet loss rates emulated in our scenario with and without PEPs.



(a) SPTCP goodput on the Satcom link with TCP NewReno and Cubic CC and emulated losses.



(b) SPTCP goodput on the LTE link with TCP NewReno and Cubic CC and emulated losses.

Figure 5.7: SPTCP goodput measurements with emulated losses on the LTE and Satcom links employing TCP NewReno and Cubic CCs.

Multipath TCP performance

We classify the analysis into three parts and study the MPTCP uncoupled and coupled CCs with each scheduler configuration and the schedulers with each CC configuration. We employ time series and cumulative distribution functions (CDFs) to analyze and compare the performance of the CC and schedulers.

The aggregate goodput is measured at the MPTCP connection-level using the iPerf3 tool. The individual subflow link utilization is extracted from the captured packet traces using the Captcp tool. Both are measured at the client (receiver). We estimate transport-level RTTs for each subflow (path latencies) and from the captured packet traces at the server (sender).

The MPTCP connection-level goodput should be close to the application send rate of **20 Mbps**. The CDFs are from a measurement session lasting 30 seconds with values averaged over one second for ten runs. As mentioned in the previous Chapter, [Stra 20] accounts for the IP and Ethernet headers in their throughput regulation which is close to two percent. Additionally, they observe that concealed losses on the link layer add to variance in the measured throughput. **Figure 4.8** shows a small gap between the configured and observed throughput due to the mentioned overhead. Hence for the goodput measurements, we set a threshold of **19 Mbps** close to the send rate of **20 Mbps** to account for the additional overhead and variance. MPTCP connection should perform as well as the best single-path link. Hence we benchmark the achieved MPTCP goodput with the SPTCP goodput on the best path, which reaches a value above **19 Mbps for 70 percent** of data points.

Similarly, MPTCP should fulfill the latency threshold of **250 ms** in each direction as defined in the QoS requirements. Any configuration that exceeds this value on either path is deemed unfit for our use case.

5.3.1 Scheduler comparison with various CCs

Goodput

We compare different multipath TCP schedulers with different MPTCP CC algorithms. **Table 5.2** shows the percentage of data points that achieve goodput higher than the defined 19 Mbps threshold for each set of configurations. The table is symmetric and can be referred to for all comparisons.

BLEST

BLEST offers better goodput aggregation and outperforms all other compared schedulers when paired with uncoupled and loss-based CC algorithms, with at least 75 percent of

Table 5.2: The percentage of data points achieving MPTCP connection-level goodput > 19 Mbps.

[%]	lia	olia	balia	wvegas	reno	cubic
Default	60	70	70	5	75	80
BLEST	75	75	80	5	80	80
RR	60	60	60	5	70	75
Redundant	55	60	55	5	60	55

data points achieving a goodput higher than 19 Mbps, as shown in **Table 5.2 and Figure 5.10**. It also fulfills the MPTCP's aim of offering goodput at least as good as the best available single-path.

BLEST is an MPTCP scheduler designed for heterogeneous links. It estimates if scheduling a segment on a given subflow causes HoL-blocking at the receiver. It takes a proactive approach and skips a slow subflow to wait for a faster subflow to be available. Hence it lowers the risk of HoL-blocking and the number of retransmissions. This behavior is critical for supporting remote-piloting operations in real-time by lowering bufferbloat, reducing jitter, and increasing application goodput.

Default LRF

The default LRF scheduler has the second-best performance in achieving goodput higher than 19 Mbps for 70 percent data points with OLIA, BALIA, NewReno, and Cubic CCs. The primary reasons for the slight deterioration in performance are due to the scheduler's retransmission, and penalization (RP) scheme and the effect of ACK-clocking on the scheduling decision [Paas 14]. It suffers when paired with the default LIA CC which trades in responsiveness for fairness.

The LRF scheduler takes a reactive approach to scheduling and chooses the subflow with the lowest RTT estimation that has space available in its CWND. When the MPTCP send window fills up, the LRF scheduler retransmits segments over the fast subflow and penalizes the slow subflow responsible for the blocking, reducing its CWND and contributing to the connection as an aim to prevent HoL-blocking. However, the penalization is short-lived as the eventual CWND growth on the slow subflow re-triggers the blocking. The retransmission and penalization (RP) scheme keeps the CWND of the slow subflow artificially low, and its share of bandwidth [Ferl 16].

Figure 5.8 shows the degradation in the aggregate goodput when compared to BLEST between 10 to 20 seconds. The LRF scheduler utilizes the slow subflow in the face of congestion. The scheduler employs the RP scheme when the MPTCP send window fills up.

Round-Robin

The round-robin scheduler archives a goodput of 19 Mbps for only 60 percent of data points when paired with the loss-based coupled CC algorithms, LIA, OLIA, and BALIA. And slightly higher throughput with uncoupled CCs. The behavior stems from how congestion is balanced and is explained when comparing CCs. [Paas 14] states that in the case of bulk data transmission, the application fills up congestion windows of all subflows. The scheduling decision then depends on the received ACKs freeing up space in the subflow's CWND, i.e., it experiences the ACK-clock effect, which results in scheduling that is not in a true round-robin fashion.

Additionally, the MPTCP Linux implementation of the round-robin scheduler has a tunable parameter known as *cwnd-limited*, which defaults to true. It fills up the congestion windows of the subflows without leaving any space. Meanwhile, when set to false, it prefers to leave open space in CWND to achieve actual round-robin scheduling. The parameter defaults to true in our experiments. **Figure 5.9** shows the achieved goodput for a round-robin scheduler with LIA CC. For the reasons mentioned above, we do not observe true round-robin behavior.

Redundant

The redundant class of schedulers aims to transmit data redundantly over both inflows to achieve the desired goodput while keeping retransmissions to a minimum. By doing so, the application's perceived latency is as good as the latency on the best path. They provide the best performance when the aggregate link capacities of the underlying subflow are higher than the application sending rate [Vu 19], which is valid for our setup.

However, the redundant scheduler fails to meet its goal and achieves the worst application goodput compared to other schedulers in both the lossless and scenarios with emulated losses. Also, the added redundancy is at the cost of higher link utilization on the underlying subflows. As explained previously, redundancy is expensive. The application sending data at 20 Mbps with redundancy utilizes up to twice the underlying link capacity, i.e., 40 Mbps [From 16]. Although it is shown that the coupled CC algorithms can reduce the amount of duplication by way of resource pooling compared to uncoupled CC, we observe similar results. The slow and lossy Satcom link cannot support a throughput greater than seen on the single path. In such cases, the redundant segments scheduled on slow subflow fail to transmit the link. Hence, the scheduler reschedules the redundant packets eventually sent on the fast LTE subflow. **Figure 5.9** shows the majority of bandwidth share on the LTE subflow with high packet losses in the slow-start phase as the link utilization nears the emulated bandwidth leading to congestion.

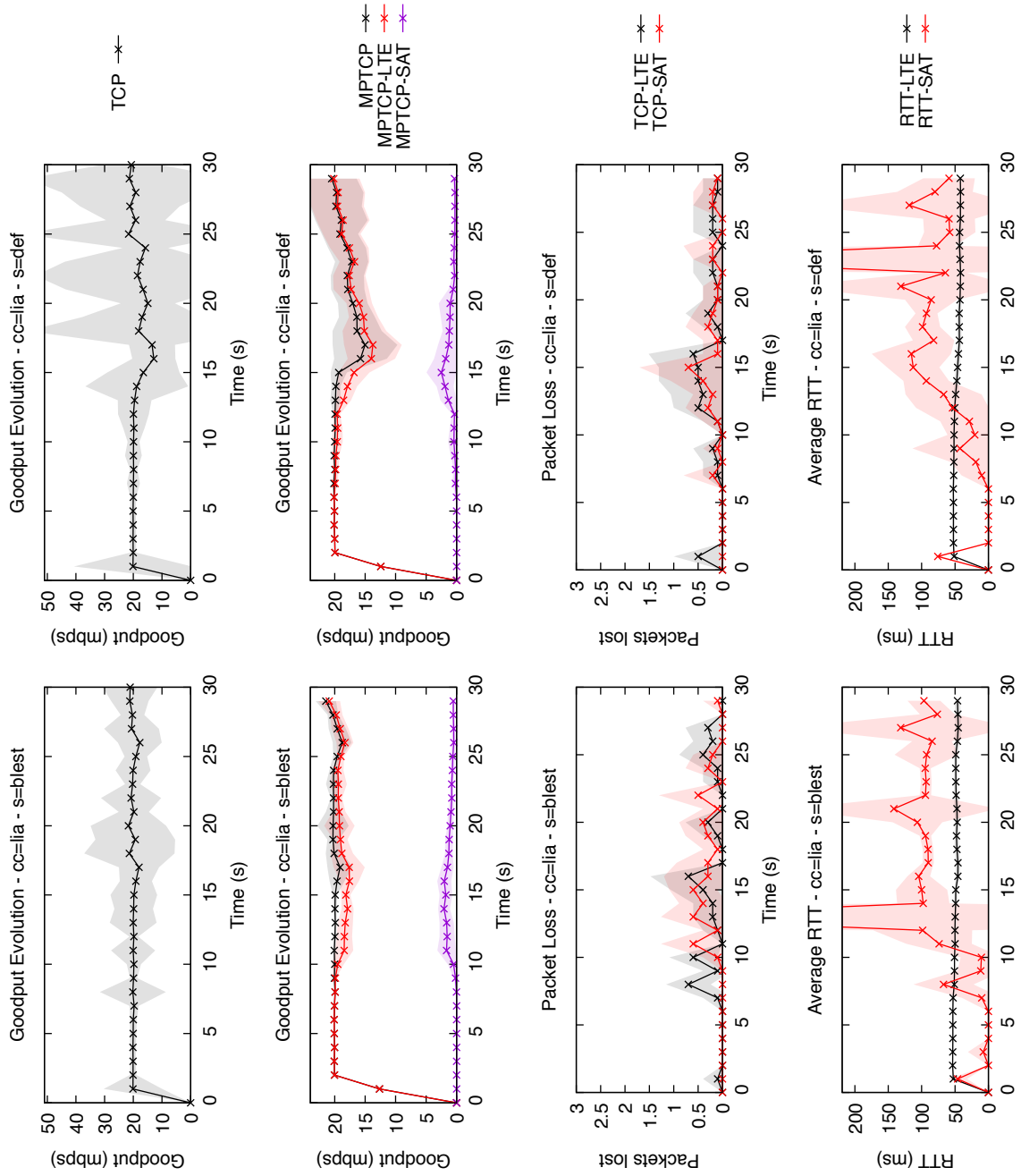


Figure 5.8: MPTCP realistic performance of the LIA CC with the BLEST and default schedulers (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

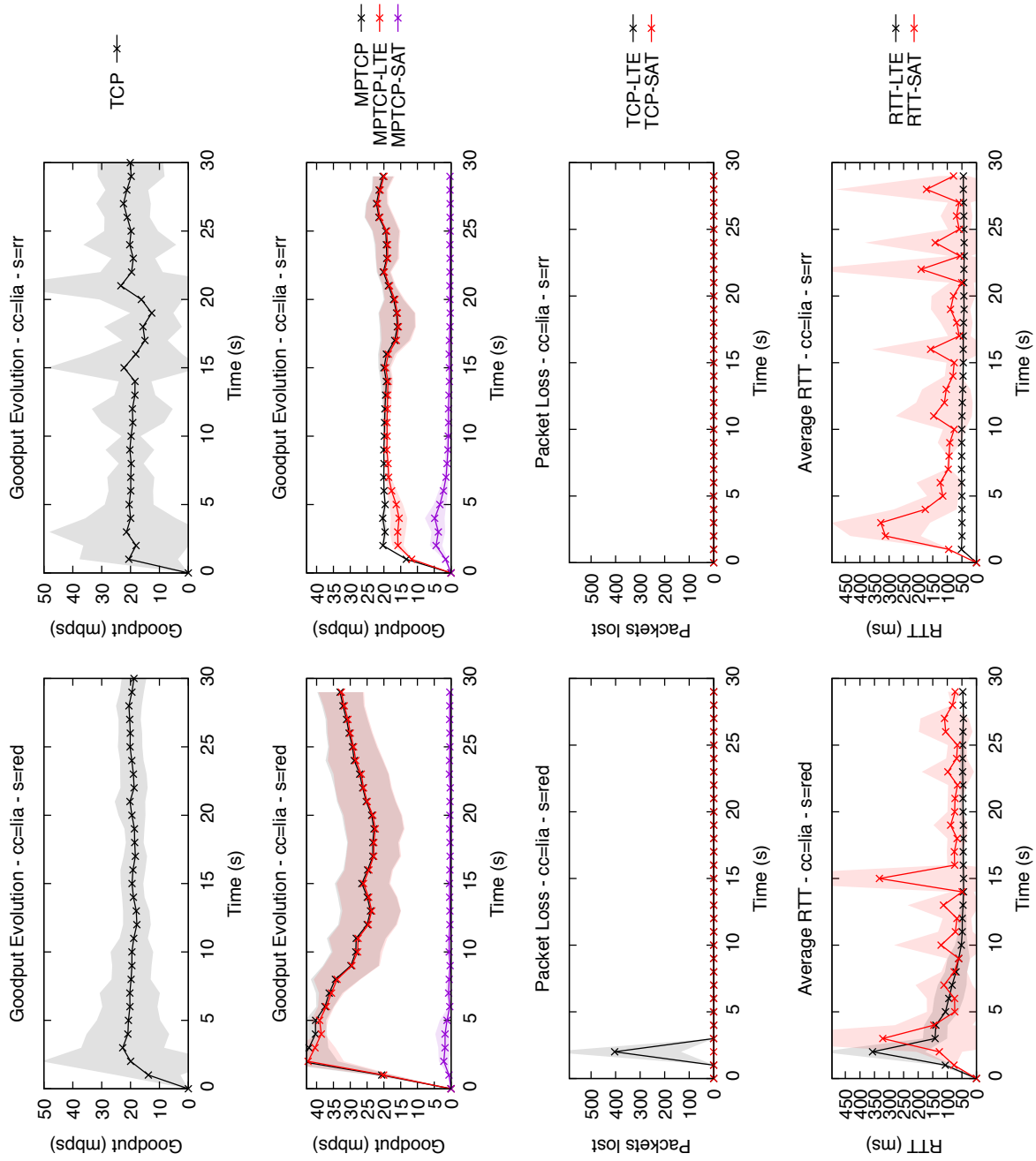


Figure 5.9: MPTCP realistic performance of the LIA CC with the Redundant and RR schedulers (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

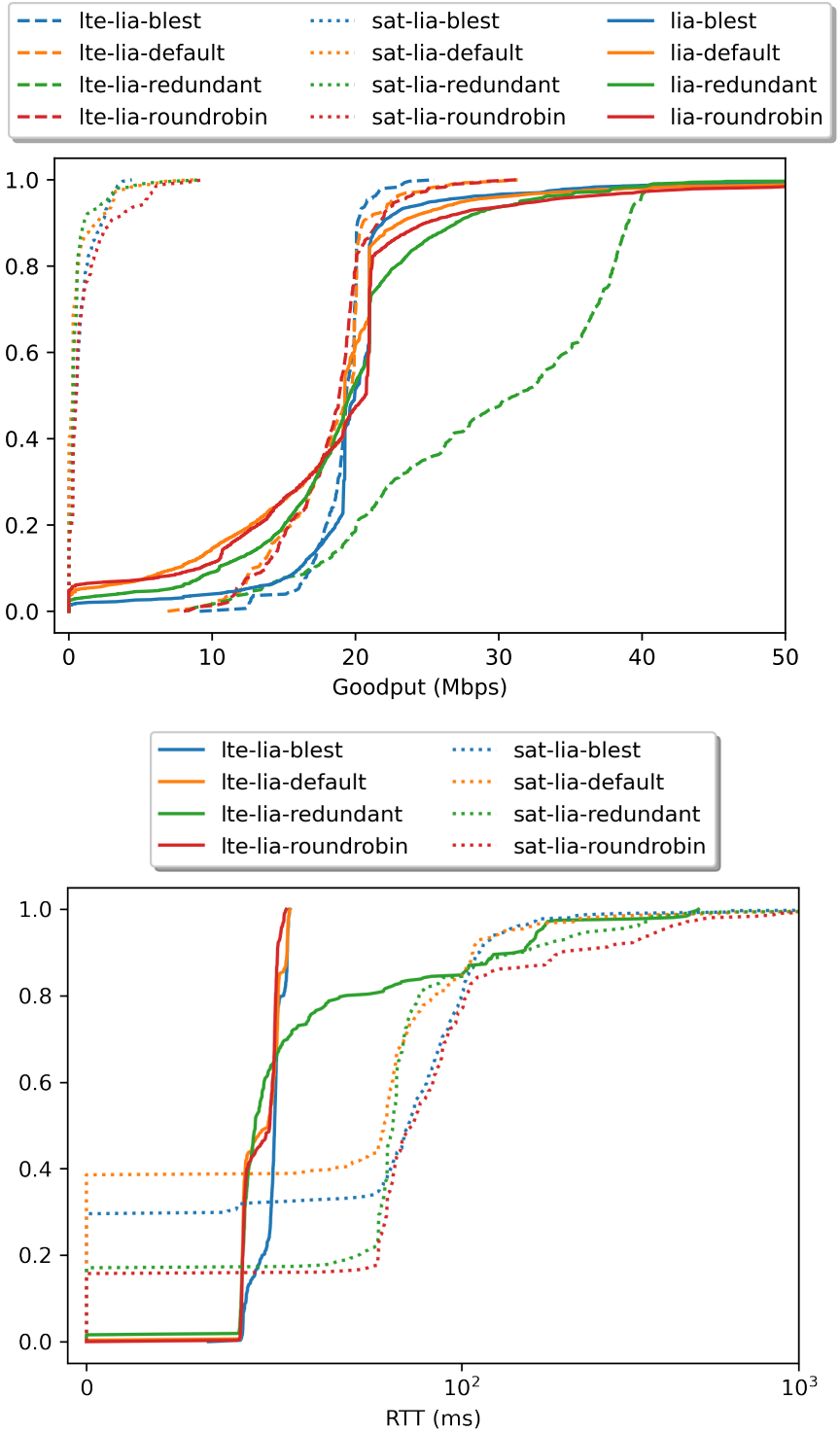


Figure 5.10: CDF of the MPTCP connection level goodput and flow level link utilization (top) and path latencies (bottom) with the LIA CC and employed schedulers.

Latency

When analyzing latency to support remote-piloting operations, the primary consideration is to avoid crossing the latency threshold of **250 ms** in both directions. As mentioned before, we estimate transport-level RTTs on each Satcom and LTE subflow using packet traces. We first compare the RTTs at the **90 percentile** to avoid outliers as shown in **Tables 5.3 and 5.4**. However, due to the strict latency requirements for our use case, no configuration should result in RTTs above **500 ms** in either link for the entire measurement session. Later, we deem the permutations crossing this threshold infeasible to support our use case. **Figure 5.10** shows the RTT CDF for the different schedulers with the default LIA CC.

BLEST and LRF

BLEST and the LRF scheduler achieves a latency which is similar for all loss-based CCs. The primary utilization of the LTE subflow along with low packet losses and retransmissions leads to the observed value. Both schedulers avoid congesting the links.

Redundant

The redundant scheduler experiences significantly higher RTT values at the 90 percentile, especially on the LTE subflow, compared to the other scheduler configurations. The duplicate packets congest both subflows. The LTE subflow is more congested than the Satcom subflow, leading to packet losses, unnecessary retransmissions of original and redundant segments, and increased per-packet latencies. As mentioned earlier, the redundant scheduler fails to achieve latency improvements on heterogenous links where the loss on the Satcom subflow is an order of magnitude higher than the LTE. The behavior is undesirable for our use case.

Round-Robin

The round-robin scheduler equally spits the segments across subflows in the lossless scenario. We observe a similar behavior with emulated losses, albeit with ACK-clocking. The segments do not experience congestion on the LTE subflow and have RTTs similar to

Table 5.3: MPTCP path latencies at 90 percentile on the LTE link.

[ms]	lia	olia	balia	wvegas	reno	cubic
BLEST	53.57	53.54	53.69	42.46	53.47	50.80
Default	53.51	53.48	53.70	42.61	53.50	50.95
Redundant	144.44	159.68	101.67	42.38	157.48	173.60
RR	50.96	50.89	51.99	42.17	51.46	50.56

Table 5.4: MPTCP path latencies at 90 percentile on the Satcom link.

[ms]	lia	olia	balia	wvegas	reno	cubic
BLEST	112.82	108.82	108.98	94.00	111.92	112.78
Default	108.11	112.51	112.42	95.88	108.22	110.43
Redundant	148.69	124.79	120.75	92.03	131.80	94.38
RR	195.06	171.45	210.26	92.10	131.89	113.86

the configuration with BLEST and LRF schedulers. However, the Satcom link utilization is higher under the RR scheduler than the rest, and hence, it operates under congestion with significantly higher RTTs.

Summary

We observe that for all schedulers paired with the delay-based CC, wVegas consistently offers a latency closest to the emulated latencies on both the LTE and the Satcom links at the 90 percentile. However it crosses the RTT threshold with all schedulers configurations on the Satcom link.

All schedulers except the redundant scheduler meet the defined latency threshold of 500 ms on the LTE link when considering the outliers. Meanwhile, only the BLEST and the default LRF scheduler meet the threshold on the Satcom link.

We focus on the BLEST and the LRF schedulers for the task of fulfilling our requirements in terms of latency and goodput.

5.3.2 Uncoupled CCs with various Schedulers

We compare the MPTCP uncoupled congestion control algorithms with different scheduler configurations.

Goodput

The uncoupled CC algorithms provide a goodput which is in most cases higher than coupled CC since they have independent CC loops on the underlying subflows adapting their CWNDs. Except for the redundant scheduler, the uncoupled CC meets the defined goodput threshold as shown in **Table 5.2**. TCP NewReno and TCP Cubic, when used with MPTCP, are said to be unfair to other flows at the shared bottleneck [Beck 12]. They are more aggressive than coupled CCs, which take a fairer approach to link utilization.

The uncoupled CC does not balance congestion among subflows. The behavior can be seen with the RR scheduler, where the congested Satcom path is overutilized throughout the run compared to the coupled CCs as shown in **Figures 7.1 and 7.2**. Meanwhile, coupled CCs balance congestion by moving traffic away from the most congested paths to the least.

TCP Cubic archives slightly better goodput than NewReno with LRF, BLEST, and RR schedulers, and it converges fastest towards the most recent bandwidth estimate before packet loss. The CWND evolution is independent of path RTTs, and hence it is more resilient to random packet losses than NewReno, as shown in **Figure 5.11** with the default scheduler. However, its gooput suffers when paired with a redundant scheduler, as shown in **Table 5.2**, due to the inability of uncoupled CC to adapt the rate of packet duplication with independent CC loops. The large send gap on the send queue, and the failure to transmit segments on the Satcom suflow due to high losses results in MPTCP connection-level rescheduling and overutilization of the LTE link [Vu 19].

Latency

Cubic and NewReno experience similar RTTs at 90 percentile when paired with the LRF, BLEST, RR schedulers on the LTE subflow. However, with redundant scheduler segments experience slightly higher RTTs due to the reasons mentioned above, shown in **Table 5.3**.

Cubic and NewReno have similar RTTs on the Satcom subflow with LRF BLEST. We cannot explain lower RTTs for Cubic with a redundant scheduler. Both achieve lower RTTs than coupled CC with RR due to the steady link utilization. With outliers, both meet the defined RTT threshold for the LTE link. But for the Satcom link, only TCP Cubic paired with the default LRF scheduler meets the threshold.

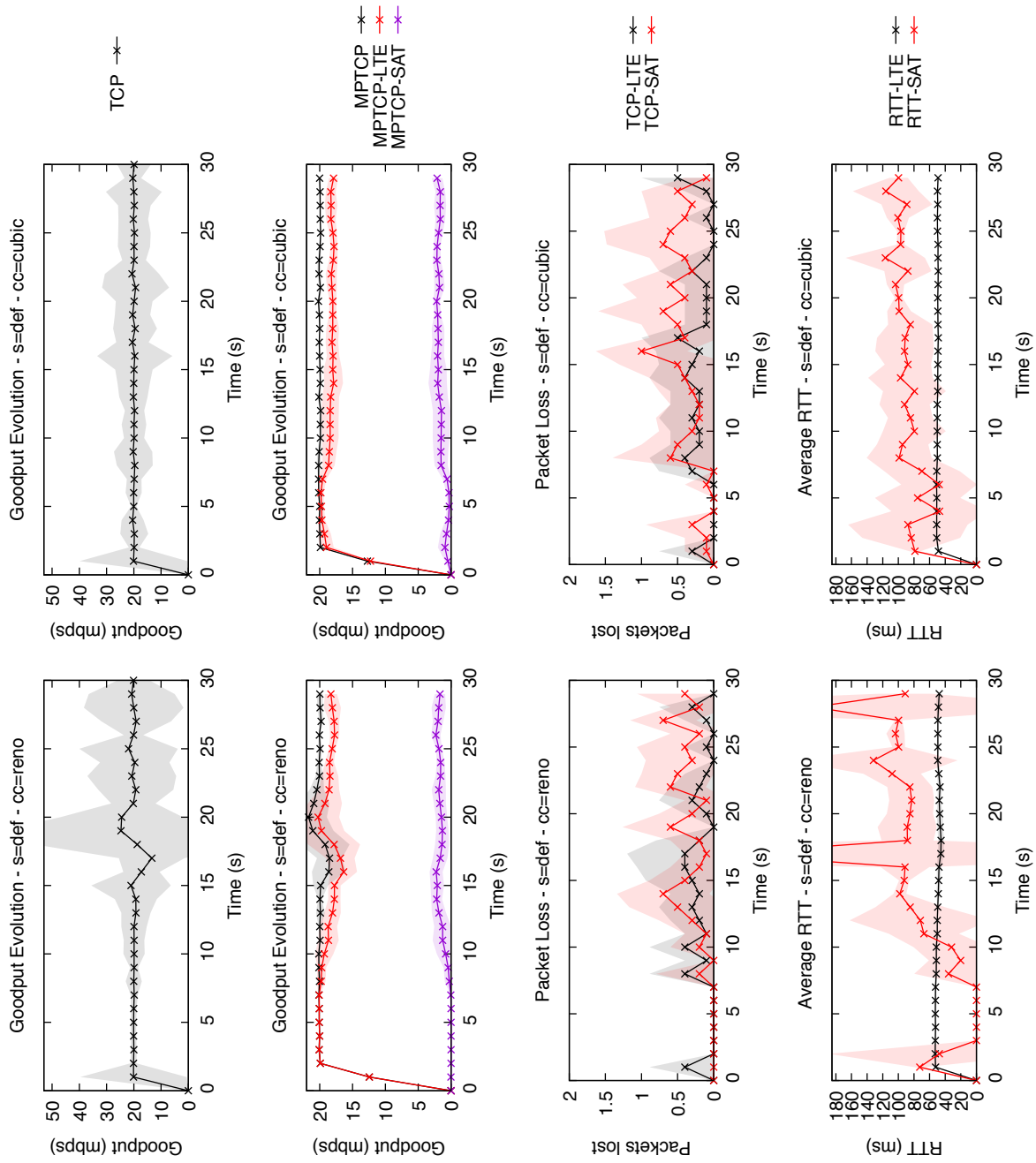


Figure 5.11: MPTCP realistic performance of the default LRF scheduler with the NewReno and Cubic CCs (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

5.3.3 Coupled CCs with various Schedulers

Next, we compare the MPTCP coupled congestion control algorithms with different scheduler configurations. Coupled CC adjusts the CWND of the underlying subflows in a combined manner to achieve objectives like friendliness, responsiveness, throughput improvement, and congestion balance [Kimu 18].

There are two types of CC employed in the study. LIA, OLIA and BALIA are loss-based CCs while wVegas is a delay-based CC.

Goodput

Coupled CCs, in general, achieve a goodput that is less than or equal to the uncoupled CCs with the defined scheduler configurations as shown in **Table 5.2**.

LIA

To ensure fairness, LIA increases its CWND defined by the minimum multipath window increase, which depends on the multipath aggressiveness factor, a sum of all the CWNDs sizes, and a window increase an SPTCP would get in the same situation. Hence MPTCP with LIA will not occupy more than a fair share of bandwidth at the bottleneck, and we see a reduced goodput with LIA compared to coupled CC. However, the achieved goodput is comparable to OLIA and BALIA in specific configurations with BLEST, RR, and redundant schedulers and is less with the default scheduler, as shown in **Table 5.2**.

LIA forces a tradeoff between load balancing and responsiveness [Khal 13] which is detrimental to heterogeneous links like the LTE-Satcom link where the load balancing suffers.

Only the BLEST scheduler paired with LIA achieves a goodput that meets our threshold. The primary reason is that the employed scheduler waits for a more promising path rather than scheduling segments over the slow path, reducing HoL-blocking at the receiver and packet retransmissions.

OLIA

OLIA delivers responsiveness to changes in the subflow's CWNDs while providing optimal congestion balancing. For this reason, it is better able to utilize both links. It achieves a higher goodput than LIA with default and the RR schedulers and similar performance with BLEST and redundant, as shown in **Table 5.2**.

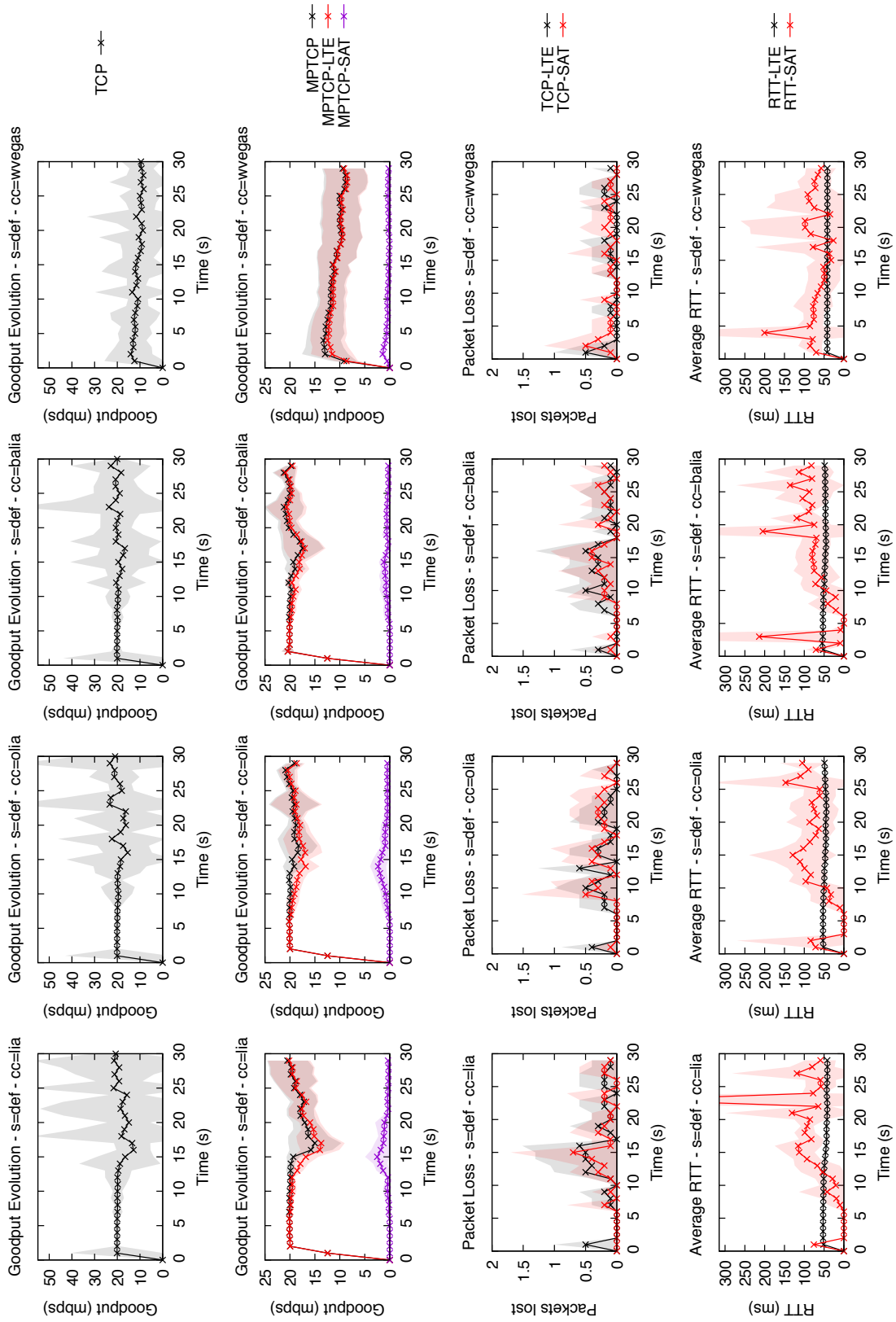


Figure 5.12: MPTCP realistic performance of the default LRF scheduler with the LIA, OLIA, BALIA, and wVegas CCs (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

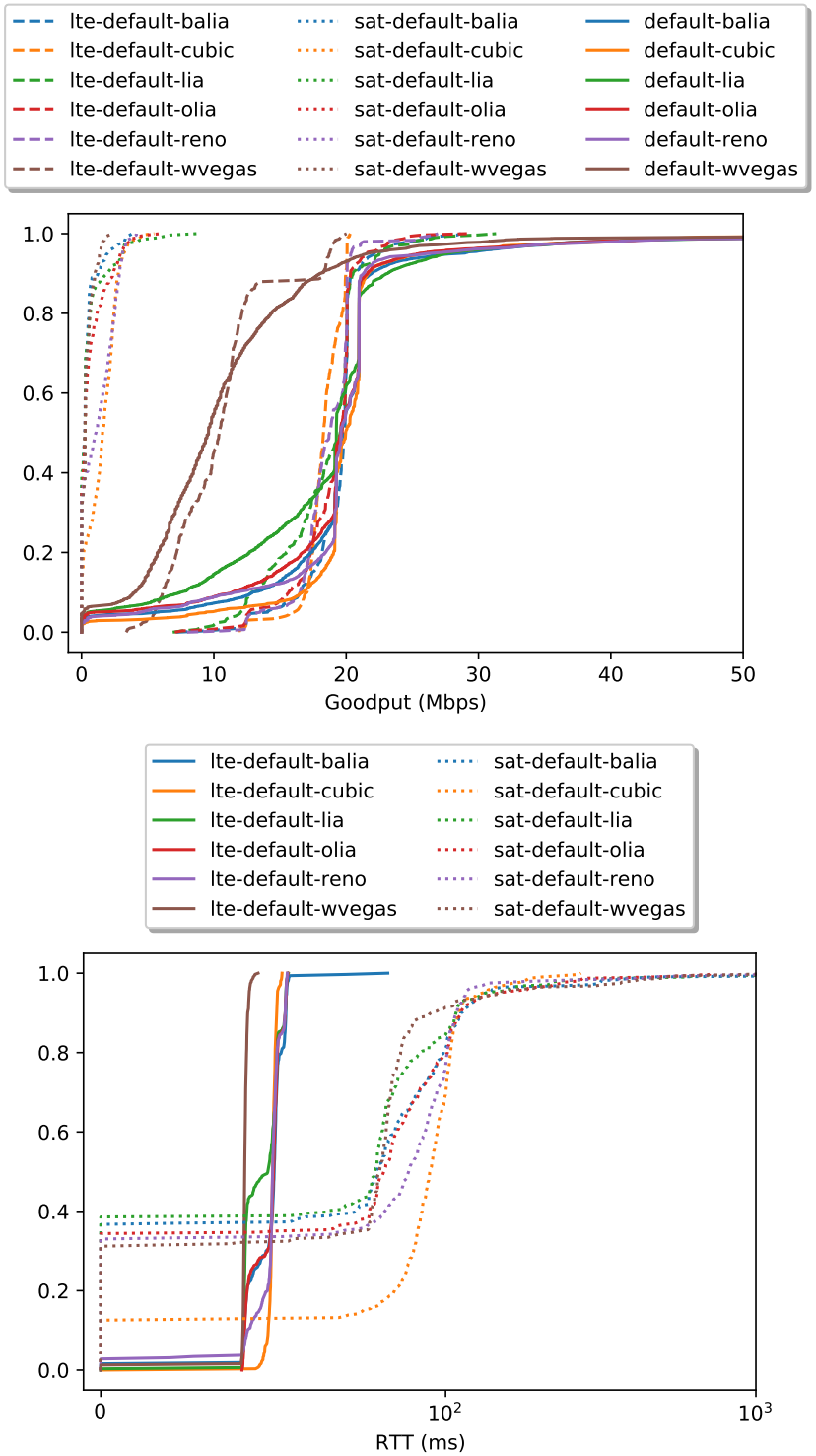


Figure 5.13: CDF of the MPTCP connection level goodput and flow level link utilization (top) and path latencies (bottom) with the default LRF scheduler and employed CCs.

BALIA

BALIA allows for oscillation in the CWND of the subflows to an ideal level to provide a good balance between fairness to other subflows and responsiveness to network changes.

BALIA offers the best performance in terms of the achieved goodput compared with other coupled CCs, as seen in **Table 5.2**, and is closest to the one achieved with uncoupled CCs. It is TCP friendly and offers throughput stability, and stikes a balance between OLIA and LIA characteristics.

wVegas

wVegas estimates the link queuing delay to realize path congestion and then proactively adapts the congestion windows of the subflows. By detecting the variation in the link queues and proactively backing-off by reducing the CWND of the subflow, it occupies fewer link buffers, reduces packet loss, and mitigates bufferbloat. The congestion avoidance phase is more sensitive to network changes than loss-based algorithms. Hence the reduced application goodput when paired with all compared schedulers. It archives the goodput threshold for only 5 percent of data points, with the median concentrated at around 11 Mbps, as shown in **Table 5.2, Figures 5.12 and 5.13**.

The sensitive nature of wVegas is observed with all configuration, where it shifts the traffic from the more congested Satcom link to the LTE link within the first five seconds.

Latency

OLIA and LIA

OLIA and LIA fail to meet the latency threshold for our application. They offer similar performance except the case with a redundant scheduler where OLIA performs better than LIA due to a reduction in packet duplication and hence congestion and retransmissions, as shown in **Tables 5.3 and 5.4**.

BALIA

BALIA offers comparable latencies on both links compared to LIA and OLIA with the default LRF and the BLEST, and RR schedulers, as shown in **Tables 5.3 and 5.4**.

The redundant scheduler performs the best with BALIA achiveding RTTs significantly lower than other loss-based CCs by reducing the amount of duplication.

With outliers, BALIA meets the latency threshold on the LTE link with all scheduler configuration. On the Satcom link, it only manages to meet it with the BLEST scheduler.

wVegas

Among the CCs studied, wVegas offers the least estimated RTTs closest to the emulated RTTs on both the subflows, as shown in **Tables 5.3 and 5.4**. Additionally, it achieves the RTTs up to 10 ms less than the other CCs at the 90 percentile. This behavior makes it very interesting for our use case—however, wVegas trades bandwidth for latency with a median goodput close to half the application sending rate. In such cases the bitrate for video traffic needs adaptation to meet the QoE targets.

With outliers considered, wVegas fails to meet our latency threshold on the Satcom link, most likely due to queueing effects. Therefore, we do not consider the CC for our use case.

We conclude that BALIA with BLEST and default LRF with Cubic offer the best balance between the achieved goodput and latency while meeting the defined application thresholds.

Packet Loss

We study packet losses for the selected permutations to validate the obsevations regarding the goodput and path latencies.

BLEST-BALIA

Figure 5.15 shows the packet loss with the BALIA CC. The BLEST scheduler has the least amount of losses when compared to other congestion control on both the LTE and the Satcom paths. As concluded, the BLEST scheduler is able to reduce the packet losses and retransmissions resulting in increased goodput and path latencies.

Default-Cubic

Figure 5.15 shows the packet losses with the Cubic CC. The default scheduler has comparable performance to BLEST on both the paths. The plot validates our choice of the permutation. While the default scheduler employs the low RTT path, the BLEST estimates HoL-blocking from the received congestion signals and avoids the high RTT path.

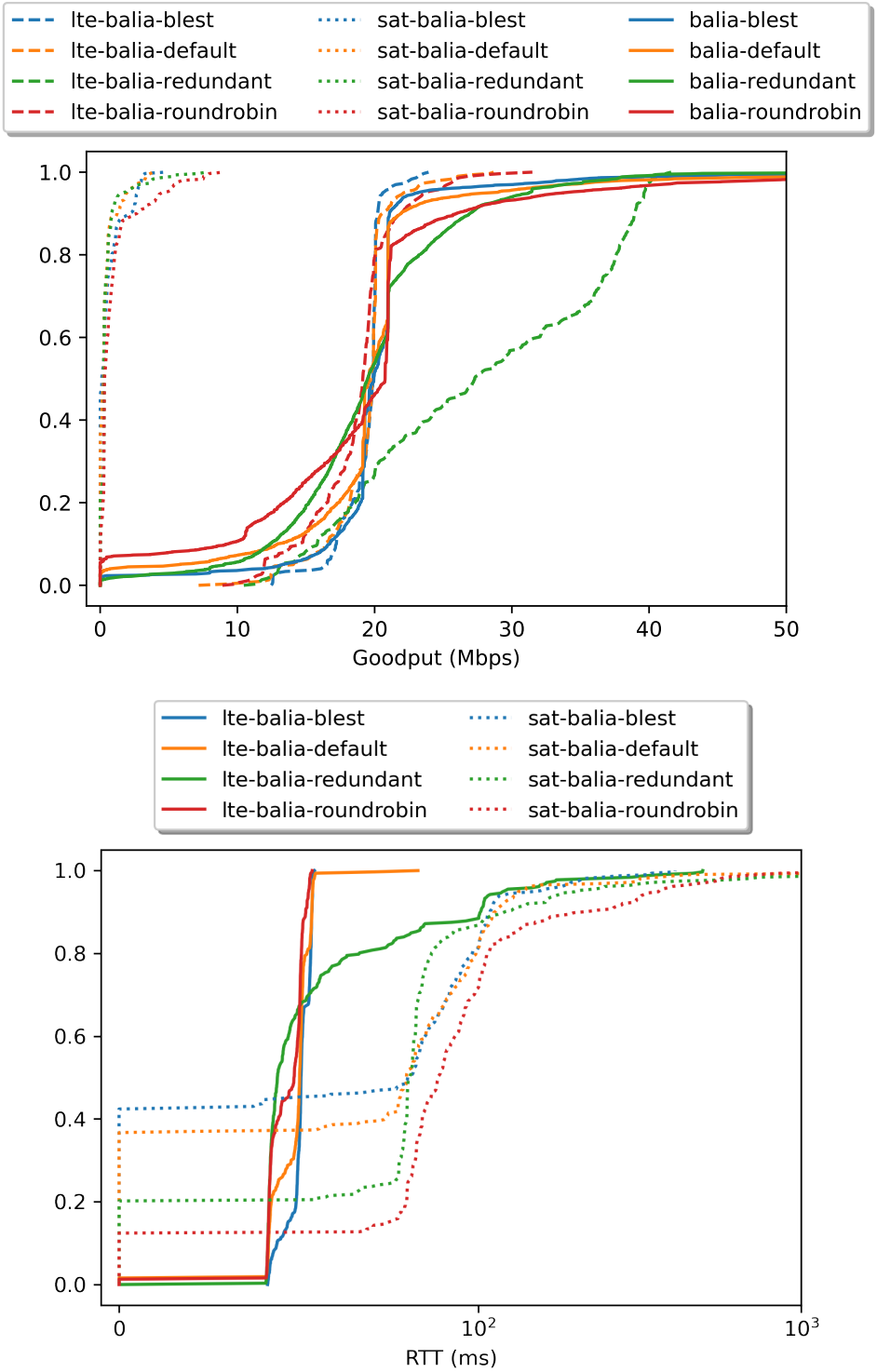


Figure 5.14: CDF of the MPTCP connection level goodput and flow level link utilization (top) and path latencies (bottom) with the BALIA CC and employed schedulers.

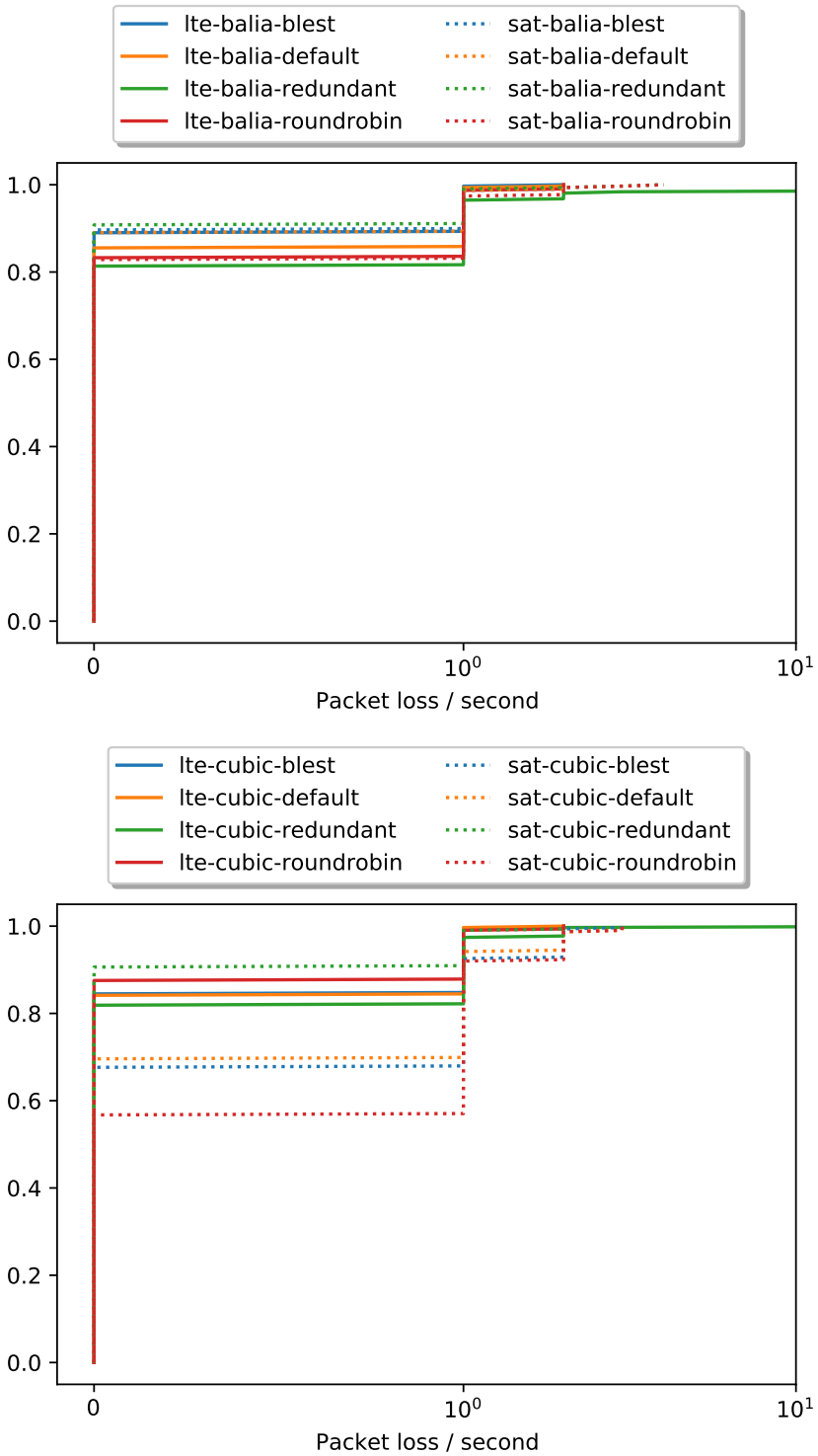


Figure 5.15: CDF of the MPTCP packet losses with BALIA CC (top) and Cubic CC (bottom) and employed schedulers.

Summary

We observe that MPTCP meets its design goals and offers a goodput equal to the best path single-path with the right choice of congestion control algorithms and schedulers in the emulated scenario. The BLEST scheduler with the BALIA CC and the LRF scheduler with the Cubic CC meet our stringent QoS requirements in terms of the achieved goodput and path latencies with the emulated PERs.

We notice that the schedulers that take a proactive approach to scheduling and employ heuristics, such as BLEST, perform better than the reactive class of schedulers like the LRF, RR, Redundant. The proactive schedulers estimate the impact of their scheduling decision on the connection's performance. Hence, future work should focus on schedulers such as the Out-of-Order Transmission for In-Order Arrival (OTIAS) scheduler, which aims to optimize real-time application performance by reducing variability, delay, and jitter [Yang 14]. It also effectively schedules retransmissions on the subflow, not used for the original transmission.

The redundant scheduler encounters an edge case with our experiment and congests the low RTT link with duplicate packets without offering the promised latency benefits. This behavior needs further investigation. Meanwhile the round-robin scheduler experiences ACK-clock which affects its scheduling decisions and prevents it from distributing traffic equally in a true round-robin fashion.

A critical consideration in estimating the one-way delay is its computation from the Linux kernel's SRTT estimation in links where the segments do not experience symmetric OWDs.

Uncoupled CC like NewReno and Cubic better utilize the high RTT, high loss link than coupled CC, resulting in a higher goodput. Cubic achieves a higher goodput than NewReno since its CWND evolution is independent of path RTT. Hence both paths grow their congestion windows at the same rate. The Cubic growth function is suitable for networks with high BDP like the Satcom [Ha 08]. However, an MPTCP connection with multiple subflows is not fair to other single-path TCP flows at the shared bottleneck.

In the multipath scenarios, loss-based coupled congestion controls that better balance responsiveness and fairness like OLIA and BALIA perform better than those not, like LIA.

Delay-based CCs like the wVegas significantly reduces RTTs experienced by segments on the best path compared to loss-based CC but at the expense of achieved goodput. However they fail to meet the performance threshold when the outliers are considered on the high delay link. Additionally, delay-based CCs are not fair to loss-based CC on a

shared bottleneck [Rodr 11].

We deployed heterogeneous LTE-Satcom links rather than the homogenous LTE-LTE or Satcom-Satcom connections to support remote piloting operations with MPTCP. One reason is the way both networks execute handovers. When the UE moves from one eNB to another in the LTE network, it follows the break-before-make paradigm, resulting in a Handover Interruption Time (HIT), as described in Chapter 4.

An interruption in communication is detrimental to critical applications such as ours. On the other hand, the ST switches from one satellite to another in a Satcom network, following the make-before-break paradigm without terminating the transport connection. The heterogeneous links employing the MPTCP aim to provide continuity in operations.

Additionally, both networks have their advantages and disadvantages. For example, the LTE is optimized for resource-constrained devices with features such as DRX to save battery, while the Satcom's satellite terminal is bulky and has higher energy requirements. The cellular networks are patchy in remote locations, while the Satcom network consists of satellite constellations designed to provide coverage even to most distant regions.

[ChU 21] conduct measurements on the Starlink LEO network and observe improved availability of up to 99.8 percent. As the LEO technologies mature, they are a promising candidate for our use case.

Chapter 6

Conclusion and Future Work

We conduct a feasibility analysis of the MPTCP protocol to support remote piloting aerial vehicles. To achieve this goal, we first study the application QoS requirements and identify latency, goodput, and packet error rates as key performance metrics. We chose to employ the terrestrial cellular LTE and the LEO SATCOM networks for the study. When paired together, they provide resilience in the face of challenging network conditions like handovers as they employ different handover mechanisms. We chose to emulate the links for the performance analysis rather than simulation or real-world testing. Emulation offers the best of both worlds; the experiments run on actual network equipment with real protocol implementations. It provides reasonable control over the experiments with moderate cost.

We propose a novel multipath emulation testbed employing the MoonGen LTE emulation script and OpenSAND Satcom emulator, which offers the flexibility to conduct automated network experiments with a host of configurable options. The testbed runs in a containerized environment using Linux network namespaces on a single physical machine, offering convenience and flexibility to enhance the testbed according to future needs. It enables transport measurements with TCP and UDP-based QUIC on a single path and MPTCP over multiple heterogeneous links. We integrate measurement tools such as ICMP ping, iPerf3, and qperf. An accompanying evaluation script analyses the captured traces and generates relevant plots. Additionally, we model the handover interruption time (HIT) on the LTE link after real-world measurements conducted under mobility on air. We also interface traces from an LEO simulator to emulate variable delays on the Satcom link.

MPTCP's performance largely depends on the selected schedulers and congestion controls. Hence a holistic approach is taken to study MPTCP protocol performance with various

scheduler and congestion control permutations. The LRF, RR, Redundant, and BLEST schedulers are analyzed with the uncoupled Reno and Cubic CC and coupled LIA, BALIA, OLIA, wVegas CCs. As a first step towards evaluating the MPTCP's performance to support our use case, we run unidirectional traffic on the testbed with emulated conditions modeled after real-world measurements. We compare the results of experiments with a total of 24 permutations and discover two which fulfill the stringent application QoS requirements in terms of the achieved goodput and path latencies.

The BLEST-OLIA and the LRF-Cubic CC are the most promising candidates. BLEST is a proactive class of schedulers that employs heuristics to estimate HoL-blocking at the receiver and reduces packet retransmissions. The default LRF scheduler maximizes goodput and reduces path latencies by primarily scheduling segments over the low-RTT LTE subflow. With a growth independent of path RTTs and a fast recovery towards the bandwidth at the most recent packet loss, the Cubic CC provides high path utilization.BALIA balances the tradeoff between responsiveness and fairness and achieves higher goodput when compared to other coupled-CCs.

The future work includes emulating the link with bidirectional traffic. We simplify the analysis and disable the LTE handover mechanism. A study with emulated mobile handovers is a promising next step. We employ CBR traffic for the survey, and the protocol analysis with VBR traffic is an interesting next step. Scope for improvement lies in improving the tools used for multipath evaluations. We estimate path latencies in our measurements, and however, MPTCP maintains an additional connection-level sequence space. Tools need to be developed that can evaluate RTTs at the connection level. Finally, Multipath QUIC transport is a promising candidate for future research.

List of Figures

Glossary

3G Third Generation. 3

ACKs Acknowledgments. 46

ATC Air Traffic Controller. 4, 6

AWS Amazon Web Services. 46

BVLoS beyond the visual line-of-sight. 3, 4, 7

C2 Command and Control. 3, 4, 5, 6, 7

C3 Control, Command, and Communication. 6

CBR constant bit rate. 46

CC congestion control. 6, 46, 47

CNPC Control and Non-payload Communication. 7

DL downlink. 44, 46

FPS frames per second. 7

GEO geostationary orbit. 4

GW gateway. 45

ICAO International Civil Aviation Organization. 4

ICMP Internet Control Message Protocol. 43

LEO low Earth orbit. 6, 4, 6, 43, 44, 45, 46, 47

LRF Lowest-RTT first. 46

LTE Long-Term Evolution. 6, 3, 6, 43, 44, 45, 47

MEO medium Earth orbit. 4

MHz megahertz. 44

MODCOD Modulation and Coding. 44

MPTCP Multipath TCP. 6, 12, 43, 46, 47

ms milliseconds. 7

PANS Procedures for Air Navigation Services. 4

PER Packet Error Rates. 7

QoS Quality of Service. 6, 1, 3, 4, 6, 7

RPA remotely-piloted aircraft. 4, 5, 6

RPAS remotely-piloted aircraft system. 4, 5, 78

RTCA Radio Technical Commission for Aeronautics. 7

RTT round-trip time. 43, 45

s seconds. 46, 47

SARPs Standards and Recommended Practices. 4

Satcom satellite communication. 6, 4, 6, 43, 44, 45, 47

SPTCP single-path TCP. 47

ST satellite terminal. 45

TCP Transmission Control Protocol. 43, 46, 47

UAS Unmanned Aircraft Systems. 3, 4, 7

UL uplink. 44, 46

Chapter 7

Appendix

7.1 Code Repositories

- The augmented MoonGen LTE Emulation script is available at https://gitlab.lrz.de/ge36xuy/moongen-lte-emulator.
- The emulation scripts for the LTE-SATCOM emulator are available at https://gitlab.lrz.de/cm/lte-satcom-emulator.
- The evaluation scripts for the LTE-SATCOM emulator are available at https://gitlab.lrz.de/cm/lte-satcom-evaluation.

7.2 Additional Plots

- **Figure 7.1**. MPTCP baseline performance of the RR scheduler with the NewReno and Cubic CCs.
- **Figure 7.2**. MPTCP baseline performance of the RR scheduler with the LIA, OLIA, BALIA, and wVegas CCs.

7.3 Mathematical Background

Mathematically, we define the model in terms of latent variables and observables. The latent variables, denoted by z, are never observed, and their correct values are unknown in advance. The observables, represented by x, are always perceived. The HIT duration is observable in our case, and the latent variable isn't defined. It is due to the 'black-box' nature of the LTE handovers.

The latent variables correspond to the mixture component in mixture models and take values in a discrete set denoted by $1, ..., K$.

To sample a data point from the mixture model, we first sample z. Then we sample the observables x from the distribution that depends on z.

$$p(z, \boldsymbol{x}) = p(z)p(\boldsymbol{x}|z). \tag{7.1}$$

$p(z)$ is always a multinomial distribution in a mixture model. $p(\boldsymbol{x}|z)$ can take a multitude of parametric forms. In our case, we use Gaussians hence the model is a mixture of Gaussians.

The GMM can be defined by the following equation,

$$z \sim Multinomial(\pi) \tag{7.2}$$

$$x|z = k \sim Gaussian(\mu_k, \sigma_k) \tag{7.3}$$

Where $\boldsymbol{\pi}$ is a vector of probability known as the mixing properties, the vector values are non-negative and sum up to 1.

$$\pi = \sum_k p_k \tag{7.4}$$

The probability density function (PDF) is computed over x by marginalizing out, or summing out, z.

$$p(x) = \sum_z p(z)p(\boldsymbol{x}|z) \tag{7.5}$$

$$p(x) = \sum_{k=1}^{K} Pr(z = k)p(\boldsymbol{x}|z = k) \tag{7.6}$$

where, $K = 4$ and

$$p(\boldsymbol{x}|z = k) = Gaussian(\mu_k, \sigma_k) \tag{7.7}$$

Expectation-Maximization (E-M) Algorithm

Expectation-Maximization algorithm consists of two steps. The first step computes the responsibilities or expectations of the latent variables, and the second applies maximum likelihood update with those responsibilities. A responsibility describes how strongly a data point "belongs" to each component or cluster. The two steps repeat until convergence. The output is the weights assigned to each cluster and their Gaussian parameters, i.e., the mean and variance.

Considerations

[Gros 22] point out two significant considerations when modeling a mixture model.

The first is the number of clusters (K) defined in the GMM. Too many clusters can lead to overfitting the dataset, and too few clusters can lead to underfitting. A solution to the problem is treating K as a hyperparameter that can be tuned.

The second consideration relates to the initialization of the clusters. A bad initialization can lead to the E-M converging to a single-mode or a component "die-out" because it doesn't assign a high likelihood to any data points. An ideal initialization strategy assigns clusters with random means and broad standard deviations.

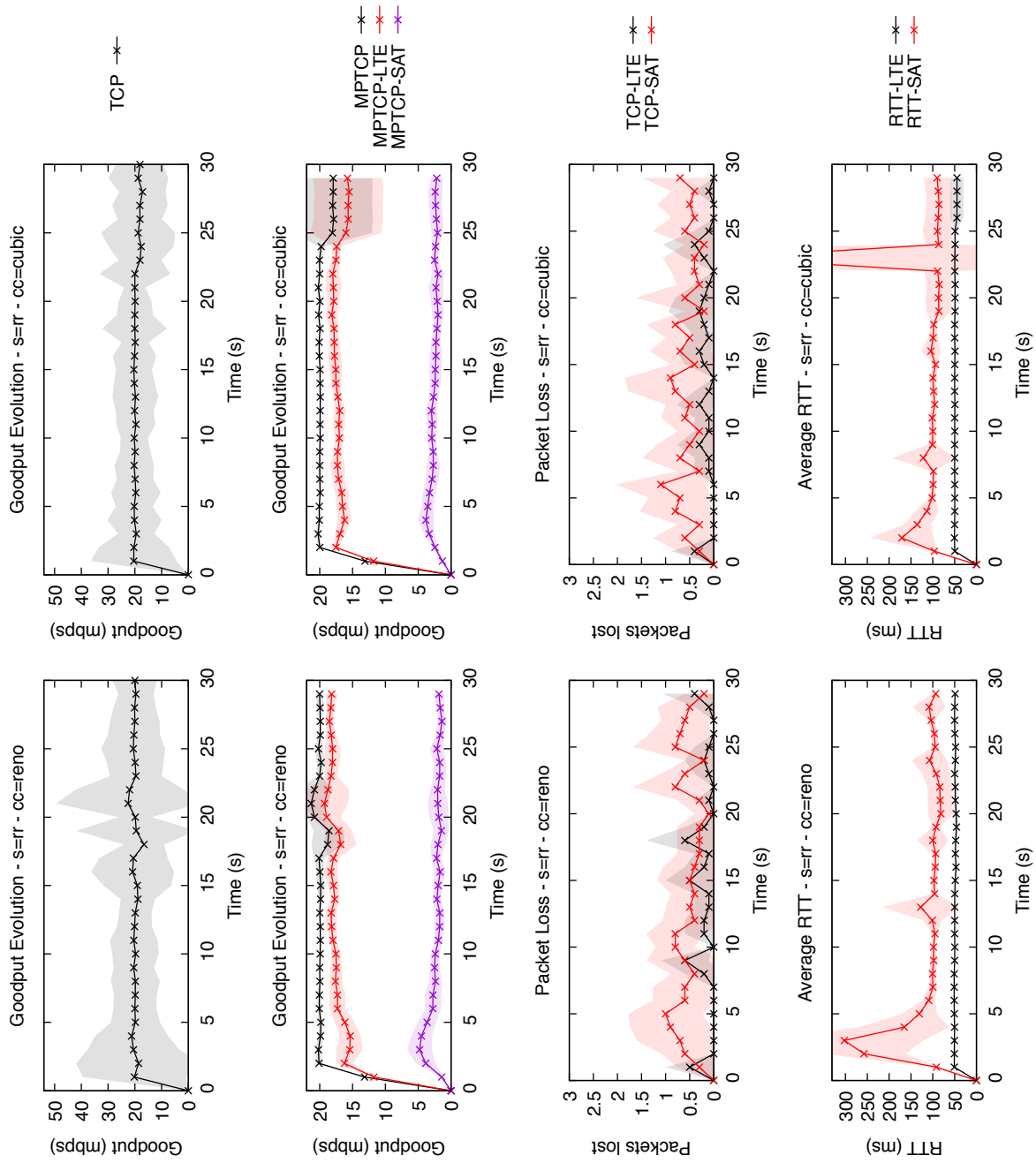


Figure 7.1: MPTCP baseline performance of the RR scheduler with the NewReno and Cubic CCs (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

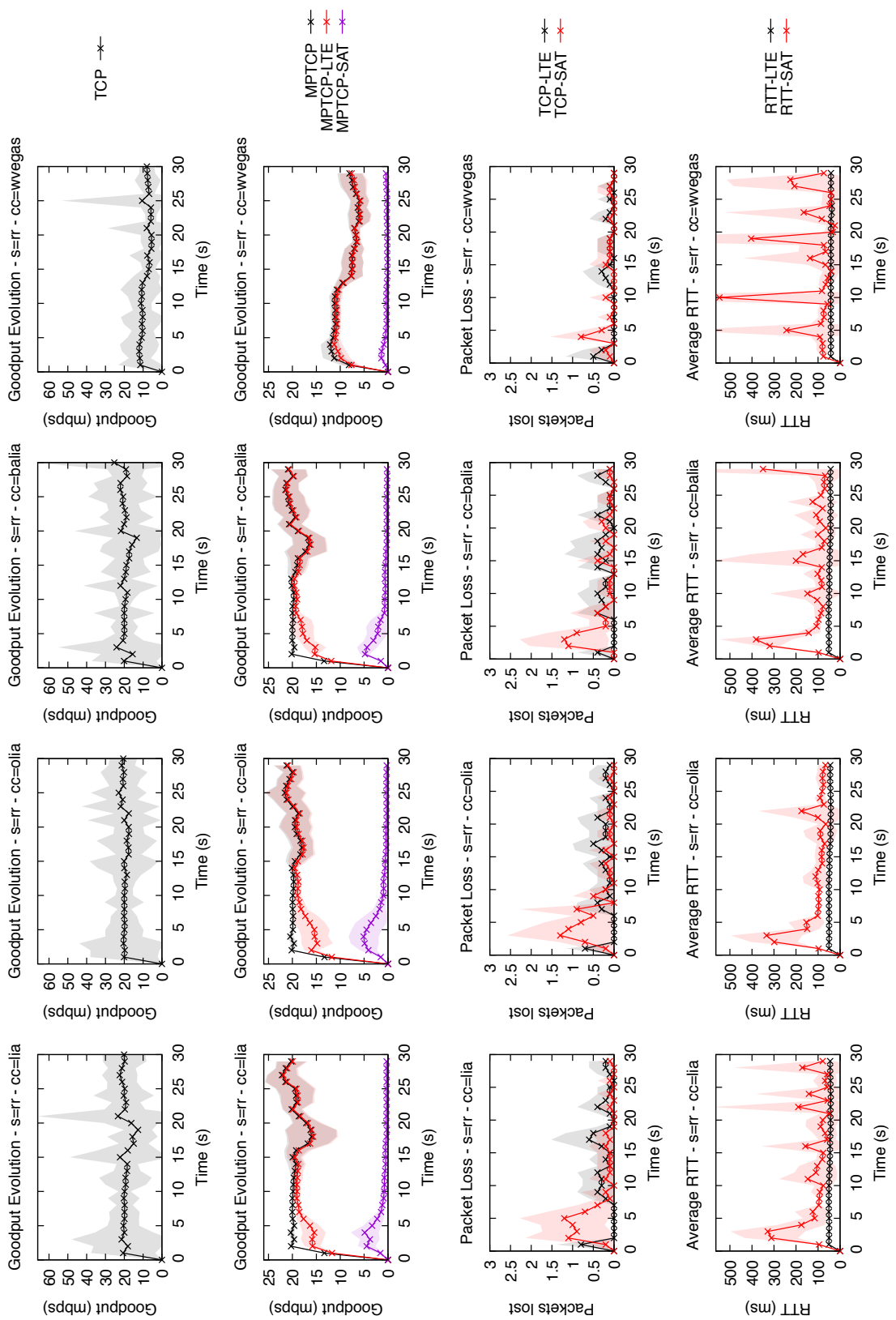


Figure 7.2: MPTCP baseline performance of the RR scheduler with the LIA, OLIA, BALIA, and wVegas CCs (left to right). The metrics shown are; the connection-level goodput, the subflow link utilization, the packet losses, and the path latencies (top to bottom).

Bibliography

[Balt 14] D. Baltrunas, A. Elmokashfi, and A. Kvalbein. "Measuring the Reliability of Mobile Broadband Networks". In: *Proceedings of the 2014 Conference on Internet Measurement Conference*, p. 45–58, Association for Computing Machinery, New York, NY, USA, 2014.

[Balt 21] A. Baltaci, M. Klügel, F. Geyer, S. Duhovnikov, V. Bajpai, J. Ott, and D. Schupke. "Experimental UAV Data Traffic Modeling and Network Performance Analysis". In: *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pp. 1–10, 2021.

[Barr 11] S. Barré, C. Paasch, and O. Bonaventure. "MultiPath TCP: From Theory to Practice". In: *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, p. 444–457, Springer-Verlag, Berlin, Heidelberg, 2011.

[Beck 12] M. Becke, T. Dreibholz, H. Adhari, and E. P. Rathgeb. "On the fairness of transport protocols in a multi-path environment". In: *2012 IEEE International Conference on Communications (ICC)*, pp. 2666–2672, 2012.

[Beck 14] N. Becker, A. Rizk, and M. Fidler. "A measurement study on the application-level performance of LTE". In: *2014 IFIP Networking Conference*, pp. 1–9, 2014.

[Beil 21] F. Beil. *Evaluating QUIC over Long Delay Networks*. Master's thesis, Technical University of Munich, Germany, 2021.

[Beur 13] R. Beuran. *Introduction to network emulation*. Pan Stanford, 2013.

[Brun 19] R. Brünig. *Performance Considerations for QUIC over Long Delay Networks*. Master's thesis, Technical University of Munich, Germany, 2019.

[Cain 06] C. Caini, R. Firrincieli, and D. Lacamera. "PEPsal: a Performance Enhancing Proxy designed for TCP satellite connections". In: *2006 IEEE 63rd Vehicular Technology Conference*, pp. 2607–2611, 2006.

[Cao 12] Y. Cao, M. Xu, and X. Fu. "Delay-based congestion control for multipath

TCP". In: *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, 2012.

[Cech 21] H. L. Cech. *Evaluating Transport Protocols for Remote Piloting of Aerial Vehicles*. Master's thesis, Technical University of Munich, Germany, 2021.

[ChU 21] ChU. "Comparison of Starlink performance between June and September". Nov 2021.

[Drei 13] T. Dreibholz and E. G. Gran. "Design and Implementation of the NORNET CORE Research Testbed for Multi-homed Systems". In: *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pp. 1094–1100, 2013.

[Drei 15] T. Dreibholz, X. Zhou, and F. Fa. "Multi-path TCP in Real-World Setups – An Evaluation in the NORNET CORE Testbed". In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pp. 617–622, 2015.

[Emme 15] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle. "MoonGen: A Scriptable High-Speed Packet Generator". In: *Internet Measurement Conference 2015 (IMC'15)*, Tokyo, Japan, Oct. 2015.

[Eric 22] Eric W. Biederman and Nicolas Dichtel. "ip-netns". Accessed on March 2022.

[Ferl 16] S. Ferlin, z. Alay, O. Mehani, and R. Boreli. "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks". In: *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 431–439, 2016.

[Floy 04] S. Floyd. "Limited Slow-Start for TCP with Large Congestion Windows". RFC 3742, March 2004.

[Ford 11] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. "Architectural Guidelines for Multipath TCP Development". RFC 6182, RFC Editor, March 2011.

[Ford 13] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. "TCP Extensions for Multipath Operation with Multiple Addresses". RFC 6824, RFC Editor, January 2013.

[Ford 20] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch. "TCP Extensions for Multipath Operation with Multiple Addresses". RFC 8684, RFC Editor, March 2020.

[From 16] A. Frommgen, T. Erbshäußer, A. Buchmann, T. Zimmermann, and K. Wehrle.

"ReMP TCP: Low latency multipath TCP". In: *2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2016.

[Fu 15] F. Fu, X. Zhou, T. Dreibholz, K. Wang, F. Zhou, and Q. Gan. "Performance comparison of congestion control strategies for multi-path TCP in the NORNET testbed". In: *2015 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–6, 2015.

[Gros 22] R. Grosse and N. Srivastava. "Lecture Notes : Mixture models". University of Toronto, Accessed on March 2022.

[Gurt 12] A. Gurtov, T. Henderson, S. Floyd, and Y. Nishida. "The NewReno Modification to TCP's Fast Recovery Algorithm". RFC 6582, Apr. 2012.

[Ha 08] S. Ha, I. Rhee, and L. Xu. "CUBIC: A New TCP-Friendly High-Speed TCP Variant". *SIGOPS Oper. Syst. Rev.*, Vol. 42, No. 5, p. 64–74, jul 2008.

[Han 15] D. Han, S. Shin, H. Cho, J.-m. Chung, D. Ok, and I. Hwang. "Measurement and stochastic modeling of handover delay and interruption time of smartphone real-time applications on LTE networks". *IEEE Communications Magazine*, Vol. 53, No. 3, pp. 173–181, 2015.

[Hurt 19] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, z. Alay, and N. Kuhn. "Low-Latency Scheduling in MPTCP". *IEEE/ACM Transactions on Networking*, Vol. 27, No. 1, pp. 302–315, 2019.

[Inte 12] International Civil Aviation Organization. "Unmanned Aircraft Systems (UAS)". Report Circular 328-AN/190, International Civil Aviation Organization, 2012.

[Kass 20] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla. "Exploring the "Internet from Space" with Hypatia". In: *Proceedings of the ACM Internet Measurement Conference*, p. 214–229, Association for Computing Machinery, New York, NY, USA, 2020.

[Katt 08] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. "XORs in the Air: Practical Wireless Network Coding". *IEEE/ACM Transactions on Networking*, Vol. 16, No. 3, pp. 497–510, 2008.

[Kell 05] F. Kelly and T. Voice. "Stability of End-to-End Algorithms for Joint Routing and Rate Control". *SIGCOMM Comput. Commun. Rev.*, Vol. 35, No. 2, p. 5–12, apr 2005.

[Khal 13] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec. "MPTCP is Not Pareto-Optimal: Performance Issues and a Possible Solution". *IEEE/ACM Trans. Netw.*, Vol. 21, No. 5, p. 1651–1665, oct 2013.

[Kimu 18] B. Y. L. Kimura and A. A. F. Loureiro. "MPTCP Linux Kernel Congestion Controls". *CoRR*, Vol. abs/1812.03210, 2018.

[Kval 14] A. Kvalbein, D. Baltrūnas, K. Evensen, J. Xiang, A. Elmokashfi, and S. Ferlin-Oliveira. "The Nornet Edge platform for mobile broadband measurements". *Computer Networks*, Vol. 61, pp. 88–101, 2014. Special issue on Future Internet Testbeds – Part I.

[Larm 09] A. Larmo, M. Lindström, M. Meyer, G. Pelletier, J. Torsner, and H. Wiemann. "The LTE link-layer design". *IEEE Communications Magazine*, Vol. 47, No. 4, pp. 52–59, 2009.

[Linu 21] Linux Foundation. "Data Plane Development Kit (DPDK)". Nov 2021.

[Neme 13] F. Németh, B. Sonkoly, L. Csikor, and A. Gulyás. "A Large-Scale Multipath Playground for Experimenters and Early Adopters". In: *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, p. 481–482, Association for Computing Machinery, New York, NY, USA, 2013.

[nsna 22] nsnam. "ns-3 — a discrete-event network simulator for internet systems". Accessed on March 2022.

[Paas 12] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. "Exploring Mobile/WiFi Handover with Multipath TCP". In: *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, p. 31–36, Association for Computing Machinery, New York, NY, USA, 2012.

[Paas 14] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure. "Experimental Evaluation of Multipath TCP Schedulers". In: *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, p. 27–32, Association for Computing Machinery, New York, NY, USA, 2014.

[Pavu 20] J. Pavur, M. Strohmeier, V. Lenders, and I. Martinovic. "QPEP: A QUIC-Based Approach to Encrypted Performance Enhancing Proxies for High-Latency Satellite Broadband". *arXiv:2002.05091 [cs]*, Feb. 2020. arXiv: 2002.05091.

[Pola 22] C. Polack. *Evaluating QoE Parameters using QUIC over Long Delay Networks*. Master's thesis, Technical University of Munich, Germany, 2022.

[Raic 11] C. Raiciu, M. Handley, and D. Wischik. "Coupled Congestion Control for Multipath Transport Protocols". RFC 6356, RFC Editor, October 2011.

[Raic 12] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. "How Hard Can It Be? Designing and

Implementing a Deployable Multipath TCP". In: *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pp. 399–412, USENIX Association, San Jose, CA, Apr. 2012.

[Rhee 18] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger. "CUBIC for Fast Long-Distance Networks". RFC 8312, Feb. 2018.

[Rodr 11] M. Rodríguez-Pérez, S. Herrería-Alonso, M. Fernández-Veiga, and C. López-García. "Common problems in delay-based congestion control algorithms: a gallery of solutions". *European Transactions on Telecommunications*, Vol. 22, No. 4, pp. 168–178, 2011.

[Saxe 20] P. Saxena, T. Dreibholz, H. Skinnemoen, z. Alay, M. A. Vazquez-Castro, S. Ferlin, and G. Acar. "Resilient Hybrid SatCom and Terrestrial Networking for Unmanned Aerial Vehicles". In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 418–423, 2020.

[Scha 06] M. Scharf and S. Kiesel. "NXG03-5: Head-of-line Blocking in TCP and SCTP: Analysis and Measurements". In: *IEEE Globecom 2006*, pp. 1–5, 2006.

[Scha 13] M. Scharf and A. Ford. "Multipath TCP (MPTCP) Application Interface Considerations". RFC 6897, March 2013.

[Sook 19] K. Sookocheff. "How does LTE work?". 2019.

[Star 21] Startlink Services, LLC. "Petition of Starlink Services, LLC for designation as an Eligible Telecommunications Carrier". Federal Communications Commission, 2021.

[Stew 07] R. Stewart. "Stream Control Transmission Protocol". RFC 4960, RFC Editor, September 2007.

[Stra 20] L. Stratmann, B. Walker, and V. A. Vu. "Realistic Emulation of LTE With MoonGen and DPDK". In: *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental Evaluation &; Characterization*, p. 87–94, Association for Computing Machinery, New York, NY, USA, 2020.

[Thal 22] Thales Alenia Space and CNES. "OpenSAND". Accessed on March 2022.

[UCLo 22] UCLouvain. "Configure MPTCP". Accessed on March 2022.

[Vu 19] V. A. Vu and B. Walker. "Redundant Multipath-TCP Scheduling with Desired Packet Latency". In: *Proceedings of the 14th Workshop on Challenged Networks*, p. 7–12, Association for Computing Machinery, New York, NY, USA, 2019.

[Wali 16] A. Walid, Q. Peng, J. Hwang, and S. H. Low. "Balanced Linked Adaptation Congestion Control Algorithm for MPTCP". Internet-Draft draft-walid-mptcp-congestion-control-04, Internet Engineering Task Force, Jan. 2016. Work in Progress.

[Wang 19] J. Wang, Y. Zheng, Y. Ni, C. Xu, F. Qian, W. Li, W. Jiang, Y. Cheng, Z. Cheng, Y. Li, X. Xie, Y. Sun, and Z. Wang. "An Active-Passive Measurement Study of TCP Performance over LTE on High-Speed Rails". In: *The 25th Annual International Conference on Mobile Computing and Networking*, Association for Computing Machinery, New York, NY, USA, 2019.

[Wisc 08] D. Wischik, M. Handley, and M. B. Braun. "The Resource Pooling Principle". *SIGCOMM Comput. Commun. Rev.*, Vol. 38, No. 5, p. 47–52, sep 2008.

[Yang 14] F. Yang, Q. Wang, and P. D. Amer. "Out-of-Order Transmission for In-Order Arrival Scheduling for Multipath TCP". In: *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 749–752, 2014.

[Yap 12] K.-K. Yap, T.-Y. Huang, M. Kobayashi, Y. Yiakoumis, N. McKeown, S. Katti, and G. Parulkar. "Making Use of All the Networks around Us: A Case Study in Android". *SIGCOMM Comput. Commun. Rev.*, Vol. 42, No. 4, p. 455–460, sep 2012.