MPTCP meets FEC: Supporting Latency-Sensitive Applications over Heterogeneous Networks

Simone Ferlin*, Stepan Kucera†, Holger Claussen†, Özgü Alay*
*Simula Research Laboratory, Norway
{ferlin, ozgu}@simula.no
†Nokia Bell Labs, Dublin, Ireland
{stepan.kucera, holger.claussen}@nokia-bell-labs.com

***Abstract*—Over the past years, TCP has gone through numerous updates to provide performance enhancement under diverse network environments. However, with respect to losses, little can be achieved with legacy TCP detection and recovery mechanisms. Both *fast retransmission* and *retransmission timeout* take at least one extra round trip time to perform, and this might significantly impact performance of latency-sensitive applications, especially in lossy or high delay networks. While forward error correction (FEC) is not a new initiative in this direction, the majority of the approaches consider FEC inside the application. In this paper, we design and implement a framework, where FEC is integrated within TCP. Our main goal with this design choice is to enable latency sensitive applications over TCP in high delay and lossy networks, but remaining application agnostic. We further incorporate this design into multipath TCP (MPTCP), where we focus particularly on heterogeneous settings, considering the fact that TCP recovery mechanisms further escalate head-of-line blocking in multipath. We evaluate the proposed framework and show that such a framework can bring significant benefits compared to legacy TCP and MPTCP for latency-sensitive application traffic such as video and web transfers.**

***Index Terms*—TCP, MPTCP, forward error correction, XOR, multipath, congestion control, wireless networks**

I. Introduction

THE enormous growth in mobile wireless devices and mobile traffic led to increased dependency on mobile infrastructures. Today, mobile operators are expected to deliver high capacity and reliable networks to meet the demand from many stakeholders. One approach to increase both reliability and capacity is to better foster network resources through multi-connectivity. For example, smartphones can leverage both cellular and WLAN, or air-to-ground communications can utilise both mobile satellite terminals and cellular networks. While the number of use-cases and applications vary and steadily grow, the choices of transport protocols to address these demands do not evolve at the same pace, with UDP and TCP being the main options in the Internet, and, in their original designs, unable to explore multiple networks simultaneously.

In light of this diversity for network access technologies, we find multipath TCP (MPTCP). MPTCP maintains applications unchanged, where *subflows* running on different networks are unnoticed by the application. In Figure 1, we illustrate the smartphone scenario with cellular and WLAN. Two main advantages with MPTCP are envisioned: Capacity aggregation and the ability to maintain the connection, if at least one network path remains active. Capacity aggregation is, however, quite challenging with heterogeneous paths, in particular due to delay and loss heterogeneity. This heterogeneity results in packet reordering, leading to head-of-line (HoL) blocking, increased out-of-order (OFO) buffer at the receiver and, ultimately, reduced goodput.

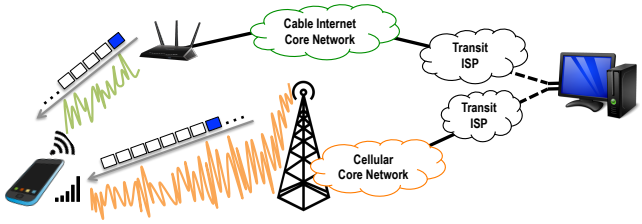


Fig. 1: System Building Blocks: Overview

Furthermore, since MPTCP is closely tied to TCP, TCP's time-dependent loss recovery can, in turn, also be a bottleneck for high delay and lossy networks. In TCP, both fast recovery (FR) and retransmission timeout (RTO), are strictly tied to round trip time (RTT). Hence, regardless of how the network capacities grow, the required RTTs recovery remain the same. For MPTCP, while the scheduler is commonly the place to improve scenarios with network heterogeneity, little can be achieved with legacy TCP loss recovery if, in addition to delay, the *subflows* have heterogeneous loss characteristics[1].

In this paper, we address TCP's loss recovery mechanism in order to improve TCP's performance in high delay and lossy networks as well as improve MPTCP's performance in heterogeneous settings. The main contributions of this paper can be summarised as:

1) We integrate forward error correction (FEC) in TCP in order to provide *zero-RTT* loss recovery for latency-sensitive applications. To achieve this, we propose TCP with dynamic FEC (TCP-dFEC) building on top of TCP instant recovery (TCP-IR) [1], [2] that uses XOR-based FEC within TCP. However, TCP-dFEC extends TCP-IR in two major ways: *(i) making it fair to regular TCP* and *(ii) designing a dynamic FEC mechanism to better cope with changing channel conditions*.

[1]MPTCP's default recovery mechanism for FR resends a packet on the same *subflows*, whereas, for an RTO, it reschedules a packet on the *subflows* with space in its congestion window (CWND) and the next lowest RTT.

2) We further extend this framework and propose MPTCP with dynamic FEC (MPTCP-dFEC) where each TCP *subflow* runs TCP-dFEC. We follow an *intra-subflow* FEC approach, in order to better understand the interaction of FEC within MPTCP, without considering its interaction with the scheduler and congestion control algorithms. The proposed MPTCP-dFEC works seamlessly with MPTCP's connection-level management signalling without sacrificing resources of good subflow(s) with FEC for other(s).
3) The proposed TCP-dFEC and MPTCP-dFEC algorithms are implemented in the Linux kernel. This enables the proposed framework to be application agnostic. Our evaluations show that the proposed TCP-dFEC and MPTCP-dFEC significantly improve the completion times for HTTP/2 web traffic and the frame rate for video streaming with H.264.

The remainder of this paper is organised as follows: Section II motivates our work putting the features provided by TCP and MPTCP protocols, FEC mechanisms and latency-sensitive application requirements into perspective. Section III explains our dynamic FEC (dFEC) algorithm design as well as the necessary system building blocks to integrate it into TCP and MPTCP. Section IV explains our measurement setup with different applications, network settings and the end-host configuration. Section V presents the results of proposed algorithms compared to regular TCP and MPTCP. Section VI puts our work into perspective with other proposals to integrate FEC into either applications or into the transport layer. Finally Section VII concludes our work, hinting to the future directions with dFEC design and evaluation setups.

II. Motivation and Background

The performance of the Transport Control Protocol (TCP) over wireless high delay and lossy networks is known to be suboptimal [3], [4], with one of the main limiting factors being TCP's loss recovery time. In such scenarios, it is often an option to replace TCP by UDP, at the expense of compromising benefits such as flow and congestion controls. When MPTCP [5] emerged, enabling simultaneous use of multiple network paths by a single data-stream, it had to take operability and deployment in the Internet into account, hence, making MPTCP look like regular TCP from the network's perspective. Although this integration brings many benefits, it also comes with challenges that hinder MPTCP's evolution. Particularly, when the underlying network paths are heterogeneous, MPTCP often underperforms TCP especially with certain latency-sensitive applications [6], [7].

Next, we will first put MPTCP in perspective with TCP, also mentioning its performance challenges under certain network environments. We will then discuss our motivation for designing a dynamic XOR-based Forward Error Correction (dFEC) inside TCP, and how this can aid multipath transport for heterogeneous paths. Finally, we will summarise the applications chosen for the evaluations and their requirements.

A. Transport Protocols

When TCP and the Internet Protocol (IP) were specified more than 35 years ago, end-hosts were typically connected to the Internet via a single network interface, and TCP was built around the notion of a single connection between them. Nowadays, the picture has been changing with end-hosts commonly accommodating multiple interfaces, e.g., smartphones with cellular and WLAN interfaces. Standard TCP is not able to efficiently explore the multi-connected infrastructure as it ties applications to source and destination IP addresses and ports. MPTCP emerged to close this gap, by allowing a single data-stream to use multiple network paths simultaneously, providing a great potential for higher application throughput and resilience against network path failure [8].

Although MPTCP enables better utilisation of network resources, scenarios with heterogeneity remain a challenge: Delay and loss heterogeneity result in packet reordering, which lead to Head-Of-Line (HoL) blocking, increased out-of-order (OFO) buffer and, ultimately, reduced overall throughput, causing MPTCP at times perform worse than TCP [6], [7].

Not only MPTCP-specific elements, e.g., the scheduler, are critical to enhance multipath performance with heterogeneity [7], but also TCP specific elements should be addressed. For example, TCP's performance in certain scenarios, e.g., high delay or lossy networks, is suboptimal [3], [4]. Focusing on TCP first, one of the limiting factors is the loss recovery time, with its legacy loss detection and recovery mechanisms being strictly tied to time: A Retransmission Timeout (RTO) after the timer expiration, or Fast Retransmission (FR) after three duplicated acknowledgements (`DupACK`) arrive from the receiver[2] to detect a loss, and at least one extra RTT is required for a retransmission to perform. For multipath, MPTCP *subflows* belonging to different technologies with distinct delay and loss profiles, can result in a situation where one of the *subflows* stalling the multipath connection [6].

B. Forward Error Correction (FEC)

One approach addressing the challenges of TCP loss recovery mechanisms is proposed in TCP-Tail Loss Probe (TLP) [9], focusing on reducing web latency for short flows, duplicating packets at the flow's tail to avoid RTOs[3].

To improve reliability in wireless networks, a common approach has been the use of FEC, where block erasure codes are used to correct transmission errors with redundant information added to the data stream. For example, in an (n, k) block erasure code, there are a total of n packets, with k source packets and $(n - k)$ redundant parity packets. The parity packets are generated in such a way that any k of the n encoded packets are sufficient to reconstruct the k source packets, resulting in a overhead of $m/(k + m)$. For many applications of block erasure codes, encoding and decoding complexity is the key concern behind the choice of the codes. Here, XOR-based codes are very beneficial with pure XOR operation, efficient in both hardware and software

[2]The standard value in Linux TCP stacks.

[3]In addition to TCP-TLP, TCP-RACK [10] is under study to change TCP's hardcoded `DupACK` loss detection threshold.

implementations, although limited to single losses within a block of packets, see Figure 2.

Taking the fact that the average link loss rate is generally unknown, and loss can be manifested through actual network congestion or be the result of random channel effects, medium access control or schedulers; there has been interest to integrate FEC-like approaches into transport protocols. In particular, we observe initiatives for TCP [9], [1], and, recently, for QUIC [11]. The reasons for that are multifold: First, improve performance by decoupling loss detection from recovery to better scale bandwidth with latency. Second, improve performance over wireless networks when recovered loss by link layer mechanisms can arrive too late at the transport layer, being discarded and retransmitted.

Along this line of thoughts, TCP-Instant Recovery (TCP-IR) [1], [2] aims to reduce TCP's loss recovery to *zero-RTT*, by applying XOR-based FEC injecting encoded packets within TCP to provide N+1 redundancy. However, XOR-based FEC can be disadvantageous if more than one packet per FEC *block* is lost and a fixed-rate FEC that always reserve Congestion Window (CWND) for FEC, as it is the case in TCP-IR, wastes link capacity if FEC is not used for recovery. In QUIC, the details about its earlier experiments with FEC are not publicly available, although unofficial reports state that applications such as YouTube performed worse with FEC, which may have been the reason to its deprecation in QUIC's current development. Later, in Section V-A, we show some shortcomings of TCP-IR, which, at this stage, can be only source of speculation, whether QUIC's experiments with FEC could not have suffered from similar issues.

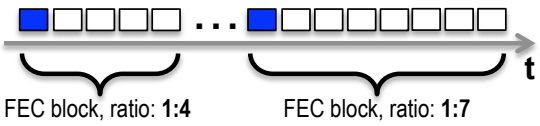


Fig. 2: XOR-based FEC block size: View from the wire.

Summary: FEC within the transport layer has been proposed, but it has been prohibitively complex inside TCP, with most of the proposals focusing on application layer FEC or simulations. The goals are multifold to improve TCP's performance over wireless networks and to decouple loss detection and recovery mechanisms. TCP-TLP and TCP-IR initiated the work to reduce TCP's loss detection and recovery to *zero-RTT*, however, both approaches do not respect TCP's CWND nor specify adaptiveness for FEC at run-time. In order to support latency sensitive applications, with the benefits that TCP provides, there is a strong need for dynamic FEC (dFEC) adaptation that also respects TCP's CWND. This should be further propagated to MPTCP, where link heterogeneity can amplify the problem with heterogeneous networks in terms of delay and loss.

C. Latency-Sensitive Application Traffic

Although web traffic still constitutes a large fraction of today's Internet [12], video is becoming the most dominant and bandwidth intensive application. Recent reports [13] show that more than 53% of North America's downstream traffic is already video streaming. Forecasts [14] also point that Internet video will continue to grow. Even though web and video differ in many ways, they are both sensitive to latency in a way that users have a better experience when web pages are loaded faster and when video has a more fluid delivery. In this paper we use video and web traffic to assess whether MPTCP with FEC can be suitable for latency-sensitive applications. The remainder of this section describes the main characteristics of the applications and discusses their requirements.

1) HTTP: HTTP/1.1 has now served the Internet for more than 15 years, being the dominant application protocol for web requests. However, loading web pages efficiently nowadays is more resource intensive, with HTTP/1.1 allowing only one outstanding request per TCP connection, hence, leaving data splitting to applications themselves. This has shown very quickly to lead to self-inflicted congestion, hurting performance. For this reason, HTTP/2 if becoming to be de-facto substitute, addressing such shortcomings, e.g. HTTP/2 is fully multiplexed, allowing multiple requests within a single TCP connection and using a single connection for parallelism.

Requirements: The quality of user experience when accessing a website is highly linked to the download completion time. For example, [15] reports that "an additional 500 ms to compute (a web search) resulted in a 25% drop in the number of searches done by users.". Although the download completion time may not be the most relevant metric for modern browsers, as they often start rendering pages before completion, it is the most suitable metric to use when evaluating transport protocols as it is browser agnostic.

2) Video: We consider non-adaptive live video streaming with H.264 in our experiments, with frames that are not delivered on time being dropped by the receiving application. In such applications, users' good quality of experience watching a live video delivered over networks that have high base delay induced or not by *bufferbloat* [16], e.g. cellular or satellite air-to-ground networks, is the ability to receive data as early and as complete as possible. Here, retransmissions caused by full or partial frame loss are hardly affordable, resulting in frames being dropped at the receiving application. Similarly, video delivery over TCP, e.g. Skype uses TCP as a backup transport protocol, would be penalised in such scenarios.

Requirements: The quality of user experience for video considered in our experiments is related to latency, however, we quantify it as the average fully received frames for non-adaptive H.264, i.e. the more complete and early received frames, the better will be the video delivery.

Finally, in addition to HTTP and non-adaptive video, we also include bulk traffic in the evaluation, since this is the most common application assessed with MPTCP.

III. System Design and Implementation

We illustrate different proposals to integrate FEC within transport protocols in Table IV. We observe that the majority of these proposals opt for an application layer approach, simplifying deployability at the expense of implementation complexity and maintenance. However, by doing so, they compromise a generic application agnostic scheme as well as sacrifice the benefits of kernel space operations such as

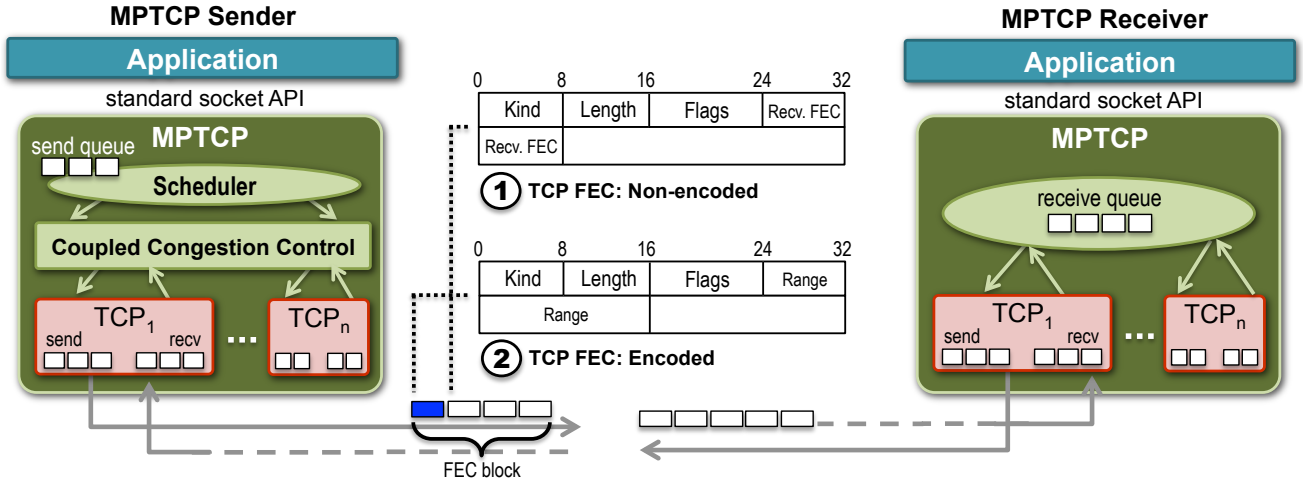


Fig. 3: **System Overview:** During TCP's three-way handshake, the FEC option is negotiated between sender and receiver. Right after it, shown by ①, the sender includes the FEC option in *all* subsequent packets of the same connection to be able to discern encoded from non-encoded packets. In ②, one FEC packet that encodes a certain number of preceding unencoded packets is sent from the sender to the receiver, on the link it is the last packet in what we call a *FEC block*, see Figure 2.

high granularity about the connection state, e.g., RTT, flow and congestion controls. Also, some of the proposals use MPTCP solely as a multipath protocol, not taking the underlying subflows' characteristics directly into account inside FEC.

In this paper, we opt for a pure transport layer XOR-based FEC within TCP to aid MPTCP with heterogeneous networks. Our goal with this design choice is to provide a clearer interface to MPTCP to manage FEC on each of its subflows independently. This is particularly relevant in the presence of heterogeneity, where MPTCP subflows have different delay and loss rates. In other words, we aim at not sacrificing capacity with FEC on low loss subflow, while avoiding HoL-blocking, due to FEC sent on a path with higher delay.

We illustrate the system building blocks in Figure 3. During TCP's three-way handshake, the FEC option is negotiated between both end-hosts. Afterwards, as depicted in both ① and ②, the sender include the FEC option in all subsequent packets, marked inside the *Flags* field, allowing the receiver to distinguish between encoded and non-encoded packets. Likewise, the receiver keeps the same format, signalling inside the *Flags* field, whether FEC failed to recover or not. As one can see, FEC signalling takes place entirely in the TCP-level at this stage, which raises questions related to deployment, e.g., if FEC options are removed or not successfully negotiated. In this case, the connection is terminated as stated in [1].

In the remainder of this section, we explain in detail how we departed from TCP-IR towards dynamic FEC (dFEC) and how it is finally integrated into MPTCP.

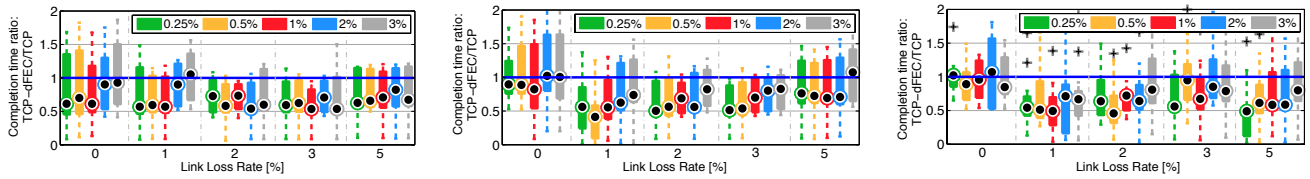
A. FEC within TCP

Both TCP-TLP and TCP-IR approaches duplicate data at a fixed rate, not respecting TCP's CWND, even though TCP-IR integrates FEC into the congestion control [1]. Also, both focus on web latency, although other latency-sensitive applications can profit from such a mechanism, e.g., video. However, these applications have a different behaviour, e.g., application-limited, bursty or greedy traffic, and must be also taken into evaluation for a generic FEC scheme. Hence, our proposed TCP dynamic FEC (dFEC), similar to TCP-IR, chooses a XOR-based FEC scheme due to its low computational overhead and implementation simplicity. However, TCP-dFEC extends TCP-IR framework in two major ways: *(i) respect CWND* and *(ii) Dynamic FEC adaptation*. TCP-dFEC aims at being fully compatible to TCP's congestion control, application agnostic, adjusting FEC dynamically at run-time. By adopting a XOR-based FEC, we send FEC systematically every X TCP segments, see Figure 2.

We argue that a pure transport layer FEC implementation is necessary mainly due to two factors: First, due to TCP's small CWNDs in very lossy environments, FEC segments may not be guaranteed every RTTs at times, depending mostly on the FEC adaption rate. Hence, a dynamic FEC is strictly necessary, see Part III-B. Second, and a particular corner-case: When FEC ratio = 10 and packet number #7 within this block is lost, the receiver will send `DupACKs` back, as it is expected with TCP, telling the sender that #7 is missing, triggering a Fast-Retransmission (FR) after three `DupACKS`[4], whereas, meanwhile, the FEC belonging to this block could have arrived and recovered #7. As a rule of thumb, in our implementation, the `DupACK` threshold should be changed to the current `FEC block size` to avoid early retransmissions[5]. We would like to point out our implementation, with the exception of adjusting the `DupACK` threshold, does not change how much data is sent into the network, which is originally TCP's congestion control task. The XOR-FEC implementation changes, however, only what is sent in terms of the ratio between data and FEC packets covering the data. Hence, there is no impact on congestion control rather than adapting the `DupACK` threshold, which has shown to be a source of concern in [17], [18].

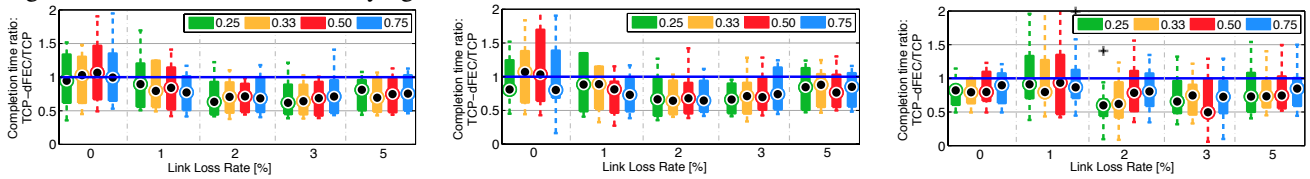
[4]This TCP's default behaviour to recover before the timer expires. There is however controversy, whether the `DupACK` threshold should be hard-coded as it is set to 3 in Linux-TCP stacks [17] and TCP-RACK [10].

[5]Note that when TCP triggers a FR, the CWND is reduced, e.g., CWND/2 with a Reno-based congestion control over one RTT, according to TCP's Proportional Rate Reduction (PRR).



(a) **25 ms** (b) **100 ms** (c) **400 ms**

Fig. 4: **TCP-dFEC vs TCP:** Varying the tolerance between 0.25, 0.5, 1, 2 and 3% with fixed correction rate δ = 0.33.



(a) **25 ms** (b) **100 ms** (c) **400 ms**

Fig. 5: **TCP-dFEC vs TCP:** Varying the correction rate δ between 0.25, 0.33, 0.50 and 0.75 with fixed tolerance = 1%.

B. The dynamic FEC (TCP-dFEC) algorithm

(i) Respect CWND: The sender tracks FEC packets in order to determine whether they successfully recovered data at the receiver or not. If more than one packet is lost within a block, FEC will then fail. If FEC is successful, then XOR-based FEC provides single loss recovery avoiding retransmissions. This, in turn, provides *zero-RTT* loss recovery.

In TCP-IR, both successful and failed FECs are used in the feedback, and integrated into the congestion control. However, it is not clear whether single losses in TCP should be treated in the same way by the congestion control [3] in all settings. Hence, in our design, the sender accounts for FEC in the congestion control, i.e., reducing the CWND[6], only if FEC is lost or it fails. Similarly, acknowledged FEC triggers a CWND increase in congestion avoidance. Furthermore, TCP-dFEC only sends FEC if the CWND has space, remaining compliant to TCP's congestion control. There is, however, a trade-off between the FEC block size, probability of multiple losses within a block and, hence, the chance of FEC to fail.

(ii) Adjust FEC ratio: TCP Dynamic FEC (TCP-dFEC): TCP-IR addresses many beneficial aspects of FEC within TCP, however, they do not specify a ratio between TCP and FEC segments, but rather use a hard-coded approach by sending a FEC packet every 0.25 RTT. We introduce the ability to set after how many TCP segments FEC should be sent, e.g., ratio = 4 means that after 4 TCP segments 1 FEC is generated. However, FEC should adapt to link changes and be application agnostic. TCP-dFEC's adaptivity is based on steering *residual losses*, with residual loss being packets that need retransmission due to FEC failing to recover, hence, triggering TCP's default loss detection and recovery behaviour.

The *residual loss* is computed as the fraction of retransmitted to first-time transmitted packets:

$$\text{Residual}_i = \frac{Retransmit}{Total - Retransmit} \tag{1}$$

where i identifies a particular Residual Loss measurement over interval T. The average residual loss is computed as:

$$\overline{\textbf{Residual}} = \frac{\sum_{n=1}^{N} \text{Residual}_n}{N} \tag{2}$$

where N is the average Residual Loss period. Then, the average residual loss is compared against a *target* residual loss rate. If the average residual loss is higher than the target, the FEC ratio is reduced, otherwise, increased as:

$$\begin{aligned} &\textbf{if} \quad \overline{\textbf{Residual}} > \texttt{target} \quad \textbf{then} \\ &\qquad \text{ratio'} = \text{ratio} \times (1-\delta) \\ &\textbf{else} \\ &\qquad \text{ratio'} = \text{ratio} \times (1+\delta) \\ &\textbf{end if} \end{aligned} \tag{3}$$

where the algorithm can update the FEC ratio, following the *target*, with a correction rate δ. Here, `target` and δ are configurable, and determine the tolerance to FEC recovery fail and correction rate, respectively.

We choose T = 3 RTTs as a minimal period during which we can capture how TCP recovers with loss. If during one RTT a loss occurs, retransmissions will be performed during the second RTT and, possibly, concluded during the third.

With a start FEC ratio = 9, N = 2 and δ = 0.33, the algorithm includes one FEC in TCP's Initial Window (IW), updating FEC in short N intervals at δ rate. On low-loss links, we expect FEC block to grow quickly reducing overhead, while on high-loss links FEC block will oscillate between low values[7]. Note that we restrict the ratio to be not smaller than 4, which corresponds to a maximum overhead of 20%. We also enforce an upper bound of 256, hence, limiting the amount of buffering at the receiver. Both values can be, however, set by the user.

The TCP-dFEC's adaptation rate depends on RTT, i.e., the adaptation rate is slower in connections with higher RTT. For short flows, this might be suboptimal, and, as a remedy, end-hosts could cache the FEC ratio per connection or per interface, just like TCP does with `ssthresh`[8]. In Figures 4, 5 and 6, we show preliminary TCP-dFEC results with NewReno, and due to its simplicity to understand, run in the measurement setup described in Section IV, including background traffic. The same way we have later run our emulation experiments.

[6]With a FR, the CWND is reduced, e.g., CWND/2 with a Reno-based congestion control over one RTT, according to TCP's Proportional Rate Reduction (PRR).

[7]We evaluated the algorithm with several N and δ, e.g., δ = 0.25 and 0.50, where δ = 0.33 yield best results.

[8]As a rule of thumb: FEC should be approximately twice the amount of the average link loss: 2% FEC for 1 to 2% random loss.

Figure 4 illustrates TCP-dFEC with different tolerance values, which indicate the FEC fail occurrences (%) before the FEC ratio is changed. There, we show tolerance values between 0.2, 0.5, 1, 2 and 3% with a fixed δ, with δ being the correction rate for the FEC ratio in the next RTT, i.e., the lower the value the milder the correction. One can observe that *tighter* tolerance values up to 1%, although mild, yield in general better results, regardless of the RTT or link loss rates. In Figure 5, we present preliminary results keeping tolerance fixed at 1%, but varying δ between 0.25, 0.33, 0.50 and 0.75. Finally, Figure 6 shows the first 10 s of a TCP bulk transfer with 25 ms RTT and how the FEC ratio changes over time for different link losses.

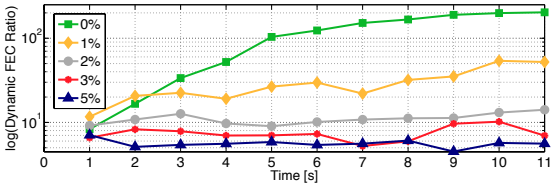
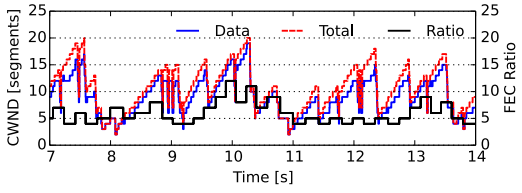
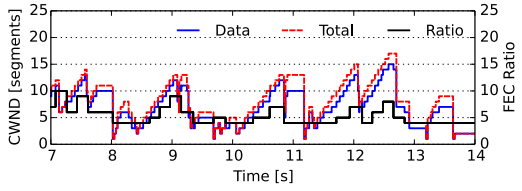


Fig. 6: **TCP-dFEC:** 10 s of a TCP bulk with RTT 25 ms and loss between 0, 1, 2, 3 or 5%, see ③ in Section III-A.

*1) **TCP-dFEC adaptation**:* Figure 7 shows a snapshot over 15 s of the CWND evolution accompanied by the respective FEC ratio stepwise adaptation for 3% loss rate and we included some preliminary results with Gilbert-Elliot burst loss model [19] with 3% loss rate and average burst size of 2 packets measured in the scenario described in Section IV. Herewith, we would like to illustrate how the FEC ratio stably adapts over time with different injected loss rates and pattern, i.e., random and burst. We use these results to demonstrate the stability of the FEC adaptation algorithm, also because it is more controlled compared to real experiments. We would like to point out in Figure 7 that the CWND increase is not altered with TCP-dFEC, rather the amount of FEC over a new set of new packets to be sent, forming a FEC block size, is changed according to the calculate FEC ratio. For the burst loss Figure 7(b) the CWNDs are slightly smaller compared to Figure 7(a), where we also observe lower FEC ratios, hovering above its minimum of 1:4.



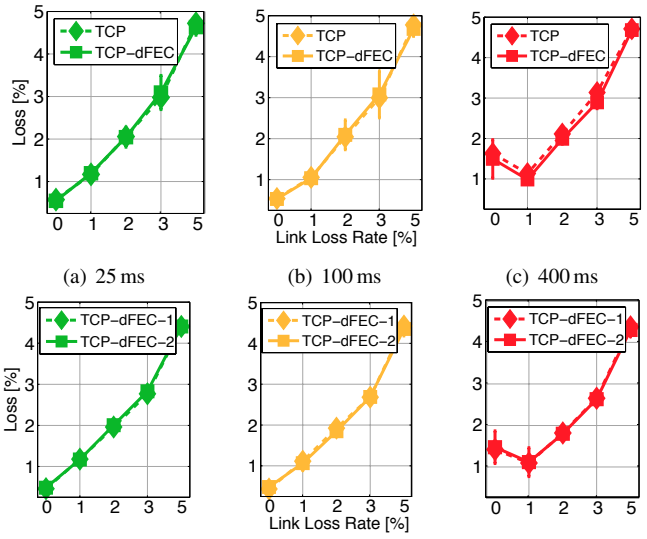
(a) **Random loss 3%**



(b) **Gilbert-Elliot average 3% loss and burst size = 2**

Fig. 7: **TCP-dFEC:** CWND evolution and its respective FEC ratio with 25 ms RTT, 3% random loss and Gilbert-Elliot loss model [19] with 3% loss rate and average burst size of 2.

*2) **TCP-dFEC fairness**:* Finally, we also consider fairness in the bottleneck against regular TCP and against TCP-dFEC itself, because of how dFEC is implemented into TCP's congestion control might be interpreted as "loss masking" by QUIC [11]. However, Figure 8 shows that TCP-dFEC does not introduce losses on a concurrent TCP and TCP-dFEC, running against further TCP bulk flows in the bottleneck.



(a) 25 ms (b) 100 ms (c) 400 ms

(d) 25 ms (e) 100 ms (f) 400 ms

Fig. 8: **TCP-dFEC fairness to TCP:** Figures 8(a), 8(b) and 8(c) and **TCP-dFEC fairness to TCP-dFEC:** Figures 8(d), 8(e) and 8(f) with 0, 1, 2, 3 or 5% injected losses and RTTs between 25, 100 and 400 ms.

The loss ratio in both experiments in Figure 8 is bound, regardless of the injected loss % and the FEC block sizes. Remember that values less than 1 are in favour of TCP-dFEC. We relate these results to the FEC algorithm congestion control management in Section III-A, part ①.

C. Dynamic FEC and MPTCP

To finally achieve our goal to integrate dynamic FEC into MPTCP, the XORed packets on the TCP level, i.e., subflow level, need to be mapped onto MPTCP's connection level signalling and management. In MPTCP, data is multiplexed on all subflows belonging to the same MPTCP connection according to mainly the scheduler, but also the couple congestion control, e.g., via load balancing. However, at the receiver, data on the different subflows need to be reconstructed in the MPTCP level, before the application can read it. This is achieved through MPTCP's Data Sequence Signal (DSS), which in a non-FEC connection, normally maps subflow data directly to connection-level window. Therefore, in a FEC connection, relevant parts of the DSS option had to be also XORed so that the receiver can reconstruct the data and packets are not dropped on the MPTCP level for this reason.

The choice of applying FEC on the subflow level rather than direct on the MPTCP (connection-level) is dictated by three main reasons: Firstly, we wanted to guarantee compatibility and seamless operation to TCP, regardless of a single path or a multipath connection with or without FEC. Secondly, adding FEC directly to MPTCP would require major integration with

MPTCP's scheduler and congestion control. Finally, and most importantly, considering the fact that our focus in this paper is paths with heterogeneous characteristics, carrying FEC on a well-performing subflow, e.g. lower latency and loss rates, to compensate for other subflows' performance, e.g. with higher loss rates, could dismiss the advantage of FEC.

Summary: This section presented dFEC, a *dynamic* FEC for TCP, which is application agnostic and adapts FEC dynamically according to the network condition. We also described how the algorithm extends TCP-IR and integrates into MPTCP to aid multipath transport with heterogeneous networks.

IV. Measurement Setup

Throughout this section, we explain our experiment setup, with the respective network settings and the applications.

A. Experiment Setup

We use CORE network emulator [20], which enables the use of real protocols and applications with emulated network links, making the evaluation easy to control and replicate. We use the MPTCP v0.90 Linux kernel implementation, so this setup also allowed us to use most of the features[9]. We use the default options of MPTCP, including e.g. receive buffer optimisation, and the socket buffer size adjustment to improve MPTCP's aggregation as suggested in [21]. To guarantee independence between experiments, we flushed all TCP-related cached metrics after each run. The network characteristics are shown in Table I and the topology is illustrated in Figure 9.

TABLE I: Emulation Network Characteristics.

	WLAN	3G/4G	Satellite
Capacity [Mbps]	20	5 – 10	0.5 – 1.5
End-to-end delay [ms]	20 – 30	50 – 85	250 – 500
Loss [%]	0 – 5	0	5 – 10

To create a more realistic emulation environment, the experiments are run with background traffic modelled as a synthetic mix of TCP and UDP generated with D-ITG [22]. The TCP traffic is composed of greedy and rate-limited TCP flows with exponential distributed mean rates of 150 pps. The UDP traffic was composed of flows with exponentially distributed mean rates varying between 395 and 995 pps and Pareto distributed on and exponentially distributed off times with on/off intervals between 1 s and 5 s. The UDP and TCP generated flows have packet sizes with a mean of 1000 Bytes and RTT between 25 and 1000 ms. Note that, Bottlenecks 1 and 2 have different capacities and the UDP background load was adjusted accordingly. Also, for the video experiments with H.264, since both video files are encoded at 3.4 Mibps and maximum of 4 Mibps, we also increased the UDP background traffic load to keep the congestion levels comparable to the other experiments. We would like to motivate the configuration of the background traffic to control the load as well as the *burstiness* in the bottleneck. The choice of the traffic distribution is based on earlier studies we have made and followed [23], [24]. In general terms, the bottleneck is loaded with bulk TCP flows and oscillations are caused with few bursty UDP flows with different average rates and distribution, which also represent popular Internet applications.

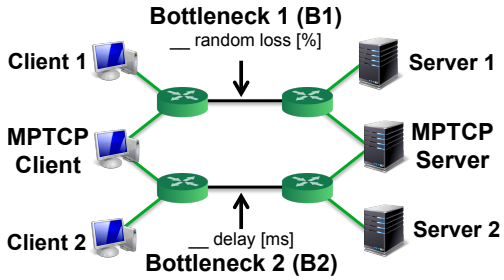


Fig. 9: Non-shared bottleneck emulation scenario

B. Applications

We perform experiments in a static scenario evaluating bulk, non-adaptive video with H.264 and `ffmpeg` and web traffic via HTTP/2 with real application traffic, see Table II.

For the non-adaptive H.264 streaming, we use `ffmpeg`[10] with one minute of the *Big Buck Bunny* video H.264 encoded at 3.4 MiBps with 25 Frames Per Second (FPS). For the HTTP/2 experiments, Table II shows the websites with their corresponding number of objects and total transferred size in KiByte, where these are downloaded using a combination of different tools, such as `nghttp2`.

TABLE II: Web Traffic Generation.

Domain name	Objects	Transfer Size [KiB]
http://www.google.com	6	1,080
http://www.youtube.com	26	3,204
http://www.espn.go.com	111	6,072

C. Experiment Configuration

To emulate a multipath scenario in the topology shown in Figure 9, we select a list of different path Bandwidth-Delay Products (BDP) and loss rates, mimicking different networks, such as cellular, WLAN and satellite. Following the settings from Table III, we keep $B1$'s RTT fixed at 25 ms, varying the loss rate between 0, 1, 2, 3 and 5%, while, in $B2$, only the RTT is changed to 25, 100 and 400 ms[11]. Hence, the scenarios under evaluation can be read as: 1) Loss heterogeneity, e.g., $B1$'s RTT is 25 ms but loss rate > 0% relative to $B2$ and 2) loss and RTT heterogeneity when $B1$'s loss rate > 0% and $B2$'s RTT > 25 ms, e.g., $B1$'s RTT is 25 ms and loss rate 1, 2, 3 or 5% and $B2$'s RTT is 100 or 400 ms. A more comprehensive summary is shown in Table III.

V. Evaluation and Discussion

This section is dedicated to present our results from the implementation described in Section III with all settings from Section IV, starting with TCP and following with MPTCP.

[9]Linux MPTCP: http://www.multipath-tcp.org

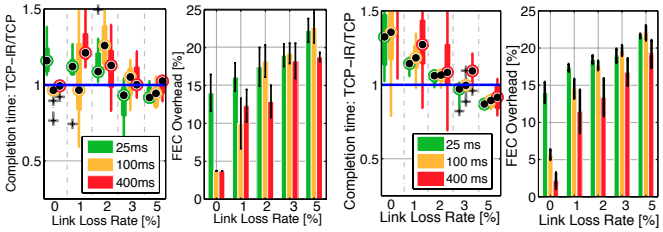
[10]https://ffmpeg.org/ffplay.htm

[11]Although there is evidence that LTE networks maintain buffer sizes larger than the path's BDP, best known as *bufferbloat* [16], we adjusted the buffers to be the value of one BDP in our experiments.

A. Dynamic FEC and TCP

In Section V-A1, we show some of the results from [25] of the original TCP-IR algorithm compared to regular TCP with bulk and web traffic, and then we compare TCP-dFEC to regular TCP with bulk, H.264 and HTTP/2 in Section V-A2. Note that the following results evaluate TCP-dFEC and TCP-IR under the same conditions, using the same measurement setup from Section IV and parameters from Table I. More discussion and evaluation results about TCP-IR with web transfer of different sizes can be found in [25].

1) TCP-IR vs TCP: In Figure 10(a) we show the completion time and FEC overhead for TCP-IR compared against regular TCP with Google web traffic. TCP-IR provides no benefit in terms of completion times regardless of RTTs or loss rates also with a relatively high FEC overhead.



(a) **TCP-IR:** Google (b) **TCP-IR:** Bulk

Fig. 10: **TCP-IR:** Google web traffic and bulk transfers. Completion time ratio (TCP-dFEC/TCP) and FEC Overhead.

In Figure 10(b), with 5% injected loss, the CWND becomes small and hence the FEC overhead increases to approximately 20%, still providing no gains in terms of completion time for bulk transfers. Also, for low link loss rate, TCP-IR increases the completion time compared to regular TCP. The results from [25] with TCP-IR provided in this section illustrate how a FEC algorithm can turn out to negatively impact applications. We emphasize the point here that a stronger integration into TCP and an adaptation mechanism to the link conditions at run time are strictly necessary.

2) TCP-dFEC vs TCP: **Bulk:** Figure 11 shows the completion time and FEC overhead of TCP-dFEC with bulk transfer against regular TCP. The left-hand side figure shows the completion times for 25, 100 and 400 ms, while the right-hand side figure shows the FEC overhead (%). We observe that regardless of the RTTs, adjusting FEC dynamically to the link characteristics brings a clear benefit of up to 40%. The benefit is lower with 25 ms, because the *price*, i.e., the time, for a retransmission to perform is lower compared to 400 ms RTT. We further observe a small FEC overhead (%) with low link losses, increasing with the average link losses. With 0% injected loss, the FEC overhead is about 1% for 25 and 100 ms and 3.5% for 400 ms, whereas it reaches up to 9% with 5% link loss. Since TCP-dFEC adapts the FEC-ratio with the feedback from the receiver, the FEC overhead for 25, 100 and 400 ms scenarios is, hence, also distinct with same average link loss rates. To compare TCP-dFEC performance with the original TCP-IR directly under the same conditions, see Figure 10(b).

TABLE III: Bottlenecks ($B1$ and $B2$) link capacity, RTT and average link loss (%), see Figure 9.

Capacity B1 and B2 [Mibps]	RTT B1 and B2 [ms]	Loss B1 and B2 [%]
20 and 10	25 and 25 25 and 100 25 and 400	0, 1, 2, 3, 5 and 0

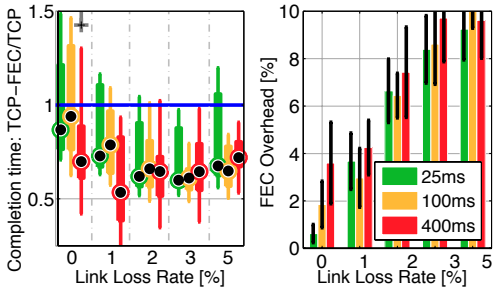


Fig. 11: **TCP-dFEC for bulk:** Completion time ratio (TCP-dFEC/TCP) and FEC Overhead.

H.264: Figures 12 shows TCP-dFEC with H.264 with the left-hand side figure presenting fully received frames ratio compared against regular TCP for 25, 100 and 400 ms and the right-hand side figure showing the FEC overhead (%). TCP-dFEC brings a steady benefit, although more variable compared to bulk. We observe a constant benefit of ca. 20% with 25 ms, and a larger benefit of up to 30 to 40% with 100 and 400 ms.

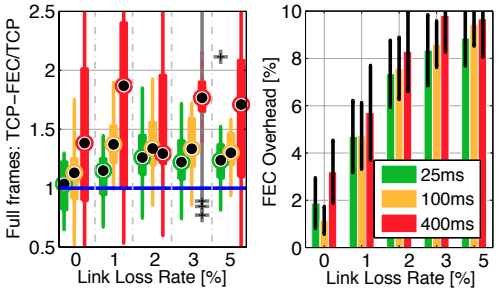
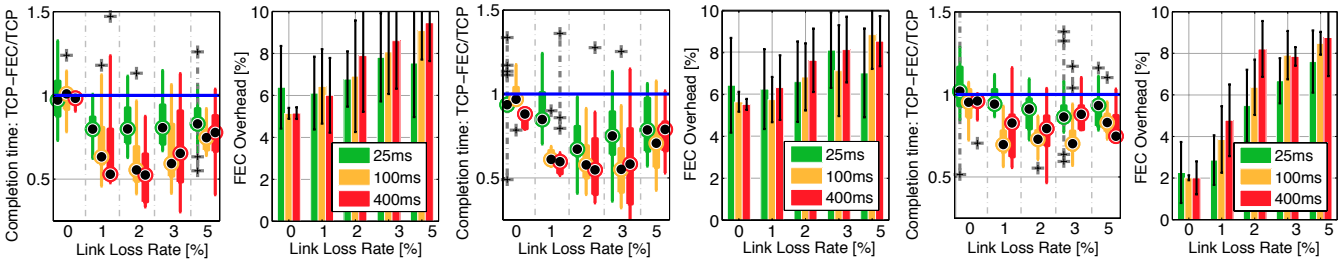


Fig. 12: **TCP-dFEC for H.264: 1 min. of *Big Buck Bunny* encoded at 3,4 Mibps:** Average dropped frame (TCP-dFEC/TCP) and FEC Overhead.

HTTP/2: Figure 13 shows TCP-dFEC with HTTP/2 and different websites sizes, see Table II, with the left-hand side figures showing the completion time ratio to regular TCP and the right-hand side figures depicting the FEC overhead (%). We observe that TCP-dFEC brings a clear benefit as the link gets lossier for Google and YouTube. The experiments with injected loss of 0% show little benefit and a relatively high FEC overhead with ca. 5 to 6% for 25 ms with Google. This is due to the FEC dynamic ratio starting with 1 FEC each 9 non-FEC packets, i.e., we can send 1 FEC packet within TCP's IW. Since the Google's website is relatively small, finishing within few RTTs, dFEC does not have the time to substantially reduce the overhead. However, with all other RTTs, loss rates and website sizes, the dynamic FEC ratio reduces the completion time by more than 30% with YouTube and 20% with ESPN. To compare TCP-dFEC from Figure 13(a) under the same conditions directly against TCP-IR, see Figure 10(b).



(a) **Google** (b) **YouTube** (c) **ESPN**

Fig. 13: **TCP-dFEC for HTTP/2:** Completion time ratio (TCP-dFEC/TCP) and FEC Overhead with background traffic, losses between 0, 1, 2, 3 or 5% and RTT between 25, 100 and 400 ms, see Table III.

B. Dynamic FEC and MPTCP

In this section we first explain how dynamic FEC can be beneficial in a multipath scenarios with heterogeneous links for bulk transfers. To emulate these links we use the topology shown in Figure 9, applying the configuration for $B1$ and $B2$ as shown in Table III. Next we present the results for MPTCP with bulk, H.264 and HTTP/2.

Bulk: Figure 14(a) shows MPTCP-dFEC's completion time compared to MPTCP, with $B2$'s RTT varying between 25, 100 and 400 ms and $B1$'s loss rate between 0, 1, 2, 3 and 5%. One can see that regardless of the RTTs and loss rates, MPTCP-dFEC brings a benefit of more than 20% for 100 ms and 400 ms as the link gets lossier. Figures 14(c) and 14(e) show per subflow FEC overhead for $B1$ and $B2$, respectively. We observe that there is a consistent higher utilisation of the $B1$ subflow compared to non-FEC subflows, the gains are up to 40% across all RTTs and loss rates. Note that $B1$'s settings emulate the WLAN path, meaning that, MPTCP-dFEC better utilises the lossy WLAN subflow compared to regular MPTCP, shifting traffic away from the higher delay and commonly paid cellular $B2$ subflow. Finally, Figures 14(c) and 14(e) depict the FEC overhead for $B1$ and $B2$. We observe that the FEC overhead in Figure 14(e) for 400 ms is high due to dynamic FEC adjustment based on the response from the receiver. Although this is an optimisation aspect of dFEC, in Figures 14(d), FEC was not sent in vain, improving $B2$'s utilisation up to 20% with 100 ms RTT, reduced to ca. 10% with 400 ms. Following the same setup for the experiments, we now comment on H.264 and HTTP/2.

H.264: Figures 15 illustrates MPTCP-dFEC's performance compared to regular MPTCP with H.264, where the left-hand side figure shows the ratio of number of full frames, i.e., I, P and B, and the right-hand side figure shows the FEC overhead. We observe that MPTCP-dFEC brings a smaller benefit compared to bulk, but constant of ca. 10 to 20% for 25 ms and more than 20% for 100 ms and 400 ms.

HTTP/2: Figure 16 illustrates MPTCP-dFEC's performance compared to MPTCP with HTTP/2 for the websites from Table II. The left-hand side figures show the completion time ratio and the right-hand side figures the FEC overhead. We observe that FEC brings benefit improving the completion time in all cases. The experiments with loss rates of 0% show less benefit and a FEC overhead of up to 5%. This is due to the FEC dynamic ratio starting a 10% rate in TCP's IW. For Google, finishing within few RTTs, dFEC does not have the necessary time, i.e., RTTs, to reduce the overhead significantly. However, with all other RTTs for $B2$, link losses for $B1$ and website sizes, dFEC can bring a benefit of more than 40% with YouTube and a constant 10 to 20% benefit with ESPN.

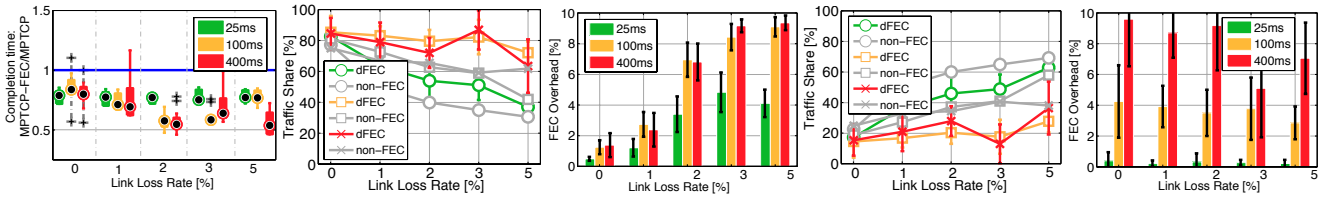
C. Real-Network Experiments

Finally, we validate the performance of the dFEC algorithm with real-network experiments within the topology as shown in Figure 17, i.e. non-shared bottleneck, constructed over NorNet [26]. We also use consumer hardware connected to a 20 Mbps DSL via WLAN and 10 Mbps via cellular 3.5G. We hold the same network capacities as shown in Table III, however, for the delays, we select two different settings: 25 ms and 100 ms and 400 ms and 100 ms for WLAN and 3.5G. While the first scenario is very close to a smartphone, the second aims at experimenting with a path with a higher delay and loss rates, e.g. a satellite terminal, combined with a cellular network. For the loss rates, we set 0.5% and 1% for 3.5G and WLAN networks, respectively.

We introduced losses with `netem` on the client side, in addition to non-influenceable concurrent traffic from other users to have some control over the experiment. The introduced losses have the goal to create a form of *ground truth*, checking how the average loss rate combinations affect dFEC compared to non-FEC experiments[12]. In addition, we also evaluated the effect of *bufferbloat* on dFEC, when excessive network buffering reduce the ability of TCP loss-based congestion control to be responsive. Hence, we aim at showing dFEC's performance under more realistic conditions in a constructed non-shared bottleneck scenario. Figures 18, 19 and 20 show bulk, H.264 and HTTP/2, respectively, for MPTCP and MPTCP-dFEC in this setup.

Bufferbloat: We run bulk transfers, removing the injected losses from both networks. The 3.5G network buffered up to 8 MiB data at times without any packet loss, we compared some of the results using the receive and the congestion windows to compare. We observed no penalty with respect to completion time compared to MPTCP without FEC in any of the two scenarios. The traffic distribution is also similar in both scenarios compared to MPTCP without FEC. In Figure 18, we show that dFEC with *bufferbloat* does not lose performance. We counted less than 0.005% loss on both networks and the 3.5G path with up to 8 MiB without loss.

[12]The expected average loss rate is set using `netem`, and were monitored during each experiment; counting the total amount of retransmissions (RTX) over the total traffic amount sent.



(a) **Completion time** (b) $B1$ **utilisation** (c) $B1$ **FEC overhead** (d) $B2$ **utilisation** (e) $B2$ **FEC overhead**

Fig. 14: **MPTCP-dFEC for bulk:** MPTCP Completion time, Subflow $B1$ and $B2$ utilisation and FEC overhead.

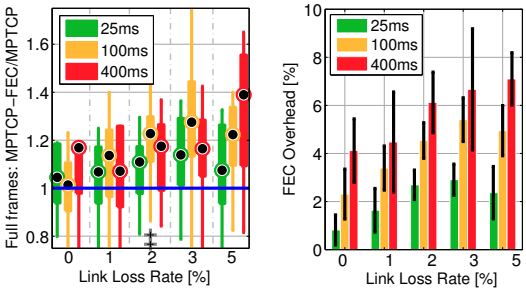


Fig. 15: **MPTCP-dFEC for H.264:** Background traffic with 0, 1, 2, 3 or 5% loss in $B1$ and 25, 100 and 400 ms RTT in $B2$, see Table III. 1 min. of *Big Buck Bunny* encoded at 3,4 Mibps. Goodput (MPTCP-FEC/MPTCP) and FEC Overhead.

Figure 19 shows the full frames for H.264 with MPTCP with and without FEC. dFEC maintains its benefit close to 22% compared to MPTCP, also in the *bufferbloat* scenario.

Figure 20 shows the completion times for HTTP/2 with the websites from Table II for MPTCP with and without FEC in the *bufferbloat* scenario. dFEC reduces completion times of up to 30% with a relatively low FEC overhead of less than 5% on average in all cases.

D. Dynamic FEC and System Performance

In this section, we analyse how dFEC affects the end-host in terms of memory usage. We take, as a measure, data being queued in the Out-Of-Order (OFO) queue, waiting for missing packets to be in-order delivered to either MPTCP-level or to the application. We performed tests with bulk and measured all changes in the Out-Of-Order (OFO) queue sizes on both subflows and on the MPTCP level. Figure 21 shows the OFO queue sizes for 25, 100 and 400 ms RTTs. We observe that dynamic FEC does not improve MPTCP's OFO queue occupancy when the subflows are homogeneous in terms of RTTs, see Figure 21(a), and it even worsen the scenario with heterogeneous RTTs, see Figures 21(b) and 21(c). To better understand this, we looked under MPTCP, into the subflows: Figure 22 shows the OFO queue size for the subflow on $B1$, see Table III, hence the subflow with only the average loss rate changed. Here, one can hardly see a difference compared to regular MPTCP. In Figures 23, however, the OFO queue occupation of the subflow on $B2$ is considerably lower compared to default MPTCP. We explain this by showing the effect of dynamic FEC into default MPTCP's congestion control and scheduler algorithms: The subflow on $B1$ is the lossier subflow compared to the subflow on $B2$, where we only vary its RTT between 25, 100 and 400 ms. dFEC improves the utilisation on subflow on $B1$ on average by 15% meaning that the average multipath rate is increased on $B1$'s subflow. In such case, MPTCP's couple congestion control exposes a larger CWND for the min-RTT scheduler, which, in turn, prefers $B1$'s subflow over $B2$, due to its minimum RTT scheduling policy.

Hence, while dynamic FEC increases the average utilisation on the lossy $B1$ subflow by shifting traffic from $B2$'s subflow and, consequently, increases the overall multipath throughput, it cannot guarantee improvements in the system resource's utilisation. At this stage, a tighter integration between FEC and MPTCP's scheduler and congestion control algorithms is required to minimise the impact on the receiver. This will be considered as part of our future research.

VI. Related Work

This section is divided in two parts: VI-A comments on TCP and MPTCP over wireless networks and VI-B comments on FEC strategies within the transport layer. Finally, Table IV summarises the key aspects of the systems in VI-B.

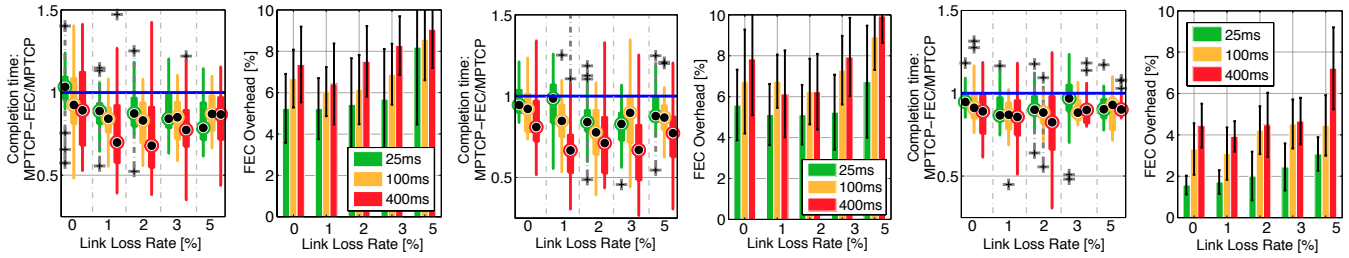
A. TCP and MPTCP over Wireless Networks

In its initial design, TCP was not meant to operate in wireless environments with links that often face random effects and, depending on the congestion control, make TCP drastically reduce its sending rate; having a long-term detrimental impact [27], [28], [29], [30], [31], [32], [33]. For example, TCP's Additive Increase and Multiplicative Decrease (AIMD) mechanism reduces its sending rate by 50% over one Round-Trip-Time (RTT)[13] and, increase, roughly, one packet per RTT.

Hence, TCP's performance over wireless networks, such as cellular or satellite, is suboptimal [3], [4], where one of the main limiting factors is the necessary recovery time. Besides many ratifications, TCP's legacy loss detection and recovery mechanisms remained mostly unchanged and tied to time: A Retransmission Timeout (RTO) is applied after a timer expires, or a Fast Retransmission (FR) is performed after three duplicated acknowledgements (`DupACK`) arrive from the receiver to detect a loss, and a retransmission requires at least one extra RTT to perform. Hence, in such scenarios, TCP is often combined with FEC, where there has been interest in the interplay of FEC and TCP with more recent initiatives being TCP-Tail Loss Probe (TLP) [9], TCP-Instant Recovery (IR) [1] or QUIC [11].

More recently, Multipath TCP (MPTCP) emerged to address the necessity of exploiting network resources with multi-connected devices, e.g., smartphones via WLAN and cellular

[13] With a Fast Retransmission (FR), the rate is halved, according to TCP's Proportional Rate Reduction (PRR).



(a) **Google** (b) **YouTube** (c) **ESPN**

Fig. 16: **MPTCP-dFEC for HTTP/2:** Completion Time (MPTCP-FEC/MPTCP) and FEC Overhead with background traffic: losses between 0, 1, 2, 3 and 5% in $B1$ and RTT between 25, 100 and 400 ms in $B2$, see Table III.

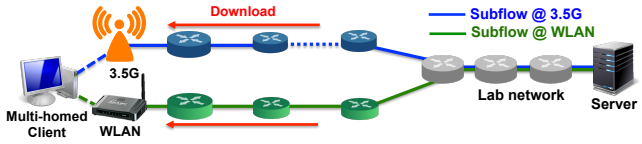


Fig. 17: Real-network experiment scenario.

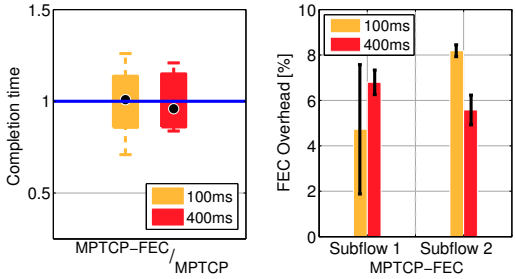


Fig. 18: **MPTCP-dFEC/MPTCP with Bufferbloat:** Completion time and FEC overhead with real-network experiments.

networks [21], IPv4 and IPv6 addresses [34] and mobility resilience support [35], [36], [37]. However, having to remain deployable and operable in today's Internet, required MPTCP to be built on top of regular TCP [38], [5].

MPTCP started off by focusing on bandwidth aggregation in scenarios with homogeneous network paths [39], [38], e.g., data center networks. However, there was an increasing interest to explore MPTCP capabilities with applications that have tighter Quality Of Service (QoS) requirements, such as web traffic and video streaming, in mobile wireless Internet. These scenarios showed to be more critical, especially when the underlying network paths are heterogeneous in terms of delay and loss rate [40], [7], [6].

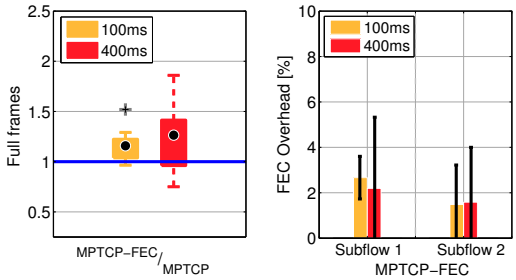


Fig. 19: **MPTCP-dFEC/MPTCP for H.264: 1 min. of *Big Buck Bunny* encoded at 3,4 Mibps:** Full frame ratio with real-network experiments.

B. Forward Error Correction (FEC)

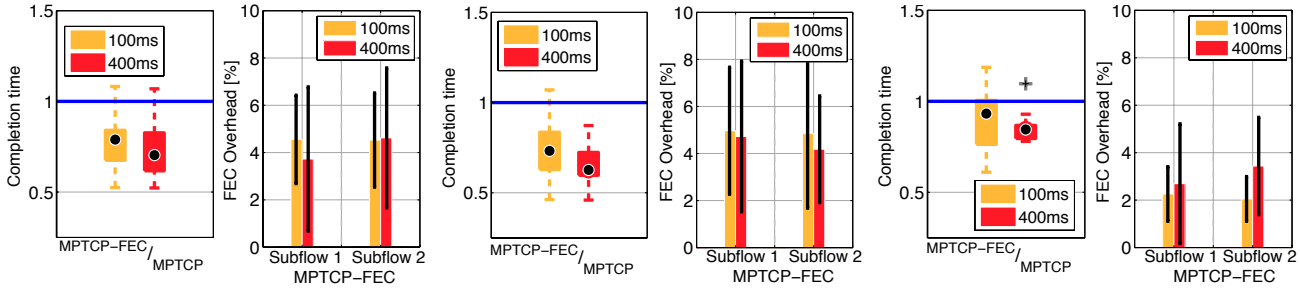
FEC is commonly used to improve reliability, with an active interest to integrate it into the transport layer [9], [1], [11]. However, this has been prohibitively complex, with most of the proposals focusing on system simulations. Here we comment on the closest and most recent proposals:

MPLOT [41], is a new transport protocol built in `ns-2` to exploit multipath diversity with erasure codes. The paper, however, does not mention transport protocol-related details, e.g., congestion control and signalling. MPLOT focus solely on bandwidth aggregation aspects in both homogeneous and heterogeneous settings with delay and loss rates, comparing its approach against regular TCP. For homogeneous scenarios MPLOT shows gains of over 20% and 37% in heterogeneous scenarios the larger the number of paths. For heterogeneous RTTs, the gains are reduced the larger the delay heterogeneity. The evaluation was done with only bulk traffic.

Coded TCP (C-TCP) [42] built in user-space, uses multiple homogeneous paths, i.e., two WLAN paths, with systematic block codes, transmitting data over UDP through a self-designed system based on delay and loss to realise congestion control and signalling. C-TCP shows its results against regular TCP using file transfer with FTP. C-TCP outperforms TCP with artificially injected losses up to 5% by ca. 69%, however, C-TCP implements an hybrid congestion control mechanism with both loss and delay signals, realised through tokens, whereas regular TCP is using CUBIC, a loss based congestion control. With two homogeneous paths, C-TCP outperforms regular TCP by 98% with 5% injected loss.

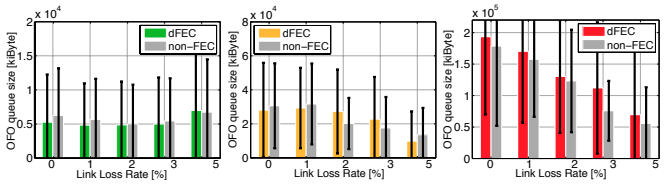
FMTCP [43] proposes a fountain code-based system in `ns-2` to help mitigate path heterogeneity in MPTCP. FMTCP focuses on evaluating heterogeneous settings by varying both RTTs between 25, 50, 100 and 150 ms and loss rates from 2 up to 15% in one of the subflows, while the other subflow's characteristics were kept constant with 100 ms and no injected loss. They experiment with a non-shared bottleneck scenario, but it is not evident, which congestion control FMTCP adopts, since they claim to focus only on data distribution. They claim gains of more than 50% in aggregation over MPTCP.

SC-MPTCP [44] uses linear systematic encoding within `ns-3` and focuses on bandwidth aggregation for heterogeneous settings, varying loss and RTTs between 1 and 5% and 20 to 60 ms, respectively. They analyzed the impact of losses on the performance of both SC-MPTCP and



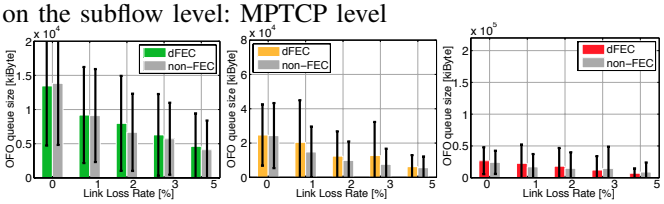
(a) **Google** (b) **YouTube** (c) **ESPN**

Fig. 20: **MPTCP-dFEC for HTTP:** Completion time for MPTCP-dFEC/MPTCP with real-network experiments.



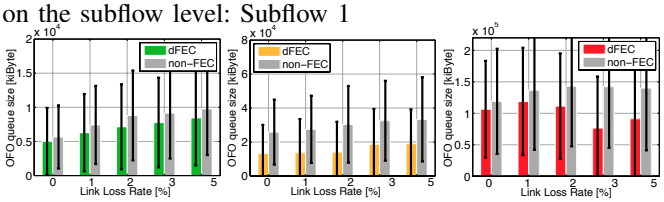
(a) **25 ms** (b) **100 ms** (c) **400 ms**

Fig. 21: **MPTCP-dFEC:** End-host average OFO queue size on the subflow level: MPTCP level



(a) **25 ms** (b) **100 ms** (c) **400 ms**

Fig. 22: **MPTCP-dFEC:** End-host average OFO queue size on the subflow level: Subflow 1



(a) **25 ms** (b) **100 ms** (c) **400 ms**

Fig. 23: **MPTCP-dFEC:** End-host average OFO queue size on the subflow level: Subflow 2

regular Multipath TCP. They demonstrated that SC-MPTCP aggregated bandwidth with losses ca. 10% compared to MPTCP with 5% loss, and it uses much less buffering, e.g., 270 kiB against 8 MiB in MPTCP.

ADMIT [45] is a multipath system focusing on real-time high definition H.264 using a MPTCP-model within the `Exata` emulator. It applies systematic Reed-Solomon codes, describing also an adaptive FEC algorithm. It focuses on goodput, end-to-end delay and PSNR metrics, since it is designed for video. Goodput gains are ca. 20% compared to regular MPTCP. Similarly, Bandwidth-Efficient Multipath Streaming (BEMA) [46], similar to ADMIT [45], is built for H.264 video over multiple paths, hence, it uses metrics such as goodput, end-to-end delay and PSNR. BEMA uses UDP and TCP with T-FRC [47] within the `Exata` emulator, applying systematic Raptor codes with FEC adaptivity.

Stochastic Earliest Delivery Path First (S-EDPF) [48] is a user-space system built to stream video over multiple paths with MPTCP using low delay random linear codes. They reuse CTCP's framework [49], however, not considering FEC adaptivity. S-EDPF (no coding) performs as good as MPTCP's low-RTT, whereas S-EDPF-8 and 16 can improve goodput aggregation by 40%, while halving the end-to-end delay.

Table IV summarises the different FEC schemes, methodology and metrics of all proposals showed in VI-B. One can notice that all systems design their own FEC inside the application, using MPTCP underneath only to distribute data over multiple paths, and they are evaluated with simulations or protocol models inside a network emulator. Although interesting, this cannot capture the interaction of such a design with a complex framework such as TCP and MPTCP. Hence, we depart from the proposed systems, proposing an entire XOR-based adaptive FEC implementation inside TCP and MPTCP.

In such case, the advantages of a XOR-based FEC approach are low computational overhead and simple implementation, where TCP's original segment structure can be maintained. However, the obvious disadvantage is that it can only recover one segment per block, e.g., if two or more packets are lost within a block, see Figure 2, the FEC packet is sent in vain, and the missing packets have to be retransmitted with Fast Retransmission (FR) of after a Retransmission Timeout (RTO).

Summary: Although there has been interest in adding FEC to the transport layer, in particular to TCP due to its loss detection and recovery mechanisms being tied to time, it has been prohibitively complex and it has been implemented with some simplifications. Most of the proposals suggest implementations in user-space, where applications have to be modified and FEC is, thus, application-specific. Also, in the application layer, the knowledge about network conditions is less granular, e.g., delay and loss rates. Therefore, in this paper we aim for a XOR-based FEC implementation within TCP, to aid multipath transport with heterogeneity with MPTCP.

VII. Conclusion

The performance of TCP over wireless high delay and lossy networks is known to be suboptimal [3], [4], with one of the main limiting factors being the loss recovery time. In such scenarios, it is often the option to replace TCP by UDP, even it compromises on benefits. However, another option is to tackle long loss recovery time of TCP by adding FEC to TCP, even though this has shown so far to be prohibitively complex. Since MPTCP is closely tied to regular TCP, this brings many benefits when it comes to deployability, but it also comes with challenges hindering MPTCP, in particular, when the underlying network paths are heterogeneous [6], [7].

TABLE IV: Some key characteristics of the systems described in Section VI.

Systems	Implementation Layer	Transport Protocol	FEC Algorithm	FEC Adaptivity	Evaluation	Application(s) and Evaluation Metrics
MPLOT	Transport	MPLOT	Erasure	✔	`ns-2`	Goodput
C-TCP	Application	UDP	Systematic block	✔	Emulation	Throughput of each path
FMTCP	Transport	TCP	Rateless Fountain	✔	`ns-2`	Goodput, delivery delay and jitter
SC-MPTCP	Transport	MPTCP	Linear systematic	✔	`ns-3`	Goodput and buffer size and delay
ADMIT	Transport	MPTCP-model	Syst. Reed-Solomon	✔	`Exata` emulator	PSNR, e2e delay and goodput
S-EDPF	Application	MPTCP	Random linear	×	Real-network	Goodput, e2e and reordering delay
BEMA	Transport	UDP and TCP	Systematic raptor	✔	`Exata` emulator	PSNR, e2e delay and goodput

In this work, we designed and implemented a XOR-based dynamic FEC scheme for TCP and MPTCP. We showed that with the proposed framework, for links having low loss rates, the FEC overhead is relatively small and for lossy links, significant performance gains can be achieved for different applications, such as HTTP/2 with different website sizes, adaptive video with HTTP-DASH, non-adaptive video with H.264 and bulk transfers.

For future study we plan to investigate FEC in MPTCP across subflows where the interaction with the scheduler and congestion control need to be taken into consideration. We will than be able to compare an intra-subflow FEC with an inter-subflow FEC approach, under different network settings. Also, real experiments with static and mobility scenarios will help to shed light of real network effects.

REFERENCES

[1] T. Flach, N. Dukkipati, Y. Cheng, and B. Raghavan, "TCP Instant Recovery: Incorporating Forward Error Correction in TCP," Internet Engineering Task Force, Internet-Draft draft-flach-tcpm-fec-00, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-flach-tcpm-fec-00

[2] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing Web Latency: the Virtue of Gentle Aggression," in *Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM '13)*, August 2013.

[3] J.-P. Martin-Flatin and S. Ravot, "TCP congestion control in fast long-distance networks," Citeseer, Tech. Rep., 2002.

[4] I. Johansson, "Congestion control for 4G and 5G access," Internet Engineering Task Force, Internet-Draft draft-johansson-cc-for-4g-5g-02, Jul. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-johansson-cc-for-4g-5g-02

[5] A. Ford, C. Raiciu, M. Handley, S. Barré, and J. R. Iyengar, "Architectural Guidelines for Multipath TCP Development," IETF, Informational RFC 6182, Mar. 2011, ISSN 2070-1721.

[6] K. Yedugundla, S. Ferlin, T. Dreibholz, Ö. Alay, N. Kuhn, P. Hurtig, and A. Brunström, "Is multi-path transport suitable for latency sensitive traffic?" *Computer Networks (COMNET)*, vol. 105, pp. 1–21, 08/2016 2016.

[7] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *IFIP Networking*. IEEE, 2016.

[8] C. Raiciu, D. Wischik, and M. Handley, "Practical Congestion Control for Multipath Transport Protocols," University College London, London/United Kingdom, Tech. Rep., Nov. 2009.

[9] N. Dukkipati, N. Cardwell, Y. Cheng, and M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses," Internet Engineering Task Force, Internet-Draft draft-dukkipati-tcpm-tcp-loss-probe-01, Oct. 2015, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-dukkipati-tcpm-tcp-loss-probe-01

[10] Y. Cheng and N. Cardwell, "RACK: a time-based fast loss detection algorithm for TCP," Working Draft, Internet-Draft draft-cheng-tcpm-rack-01, 2016.

[11] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-based secure and reliable transport for HTTP/2," *IETF, draft-tsvwg-quic-protocol-02*, 2016.

[12] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir, "Atlas Internet Observatory 2009 Annual Report," in *47th NANOG*, 2009. [Online]. Available: http://www.nanog.org/meetings/nanog47/presentations/Monday/Labovitz_ObserveReport_N47_Mon.pdf

[13] Sandvine Intelligent Broadband Networks, "Global Internet Phenomena Report," Jul. 2013. [Online]. Available: https://web.archive.org/web/20141216103806/https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/sandvine-global-internet-phenomena-report-1h-2013.pdf

[14] Cisco Visual Networking Index, "Forecast and Methodology, 2012 to 2017," 2013. [Online]. Available: https://web.archive.org/web/20141216103933/http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

[15] O3b Networks and Sofrecom, "Why Latency Matters to Mobile Backhaul," Apr. 2013. [Online]. Available: https://web.archive.org/web/20141216102926/http://www.o3bnetworks.com/media/45606/o3b_latency_mobile\%20backhaul_130417.pdf

[16] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling bufferbloat in 3G/4G networks," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, ser. IMC '12. ACM, 2012, pp. 329–342. [Online]. Available: http://doi.acm.org/10.1145/2398776.2398810

[17] S. Bhandarkar, A. L. N. Reddy, M. Allman, and E. Blanton, "Improving the robustness of TCP to non-congestion events," Internet Requests for Comments, RFC Editor, RFC 4653, August 2006.

[18] N. Dukkipati, M. Mathis, Y. Cheng, and M. Ghobadi, "Proportional rate reduction for tcp," in *Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement 2011, Berlin, Germany - November 2-4, 2011*, 2011. [Online]. Available: http://conferences.sigcomm.org/imc/2011/program.htm

[19] G. Haßlinger and O. Hohlfeld, "The Gilbert-Elliott model for packet loss in real time services on the internet," in *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference-*. VDE, 2008, pp. 1–15.

[20] J. Ahrenholz, "Comparison of CORE Network Emulation Platforms," in *Military Communications Conference (MILCOM)*, San Jose, California/U.S.A., Oct. 2010, pp. 166–171.

[21] C. Paasch, "Improving multipath TCP," Ph.D. dissertation, UCLouvain / ICTEAM / EPL, November 2014.

[22] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[23] D. Hayes, D. Ros, L. Andrew, and S. Floyd, "Common TCP evaluation suite," IRTF, Internet Draft draft-irtf-iccrg-tcpeval-01, Jul. 2014. [Online]. Available: http://tools.ietf.org/html/draft-irtf-iccrg-tcpeval-01

[24] D. A. Hayes, S. Ferlin, and M. Welzl, "Practical passive shared bottleneck detection using shape summary statistics," in *39th Annual IEEE Conference on Local Computer Networks*. IEEE, 2014, Conference.

[25] S. Ferlin and Ö. Alay, "TCP with dynamic FEC for high delay and lossy networks," in *ACM CoNEXT Student Workshop*, 2016.

[26] E. G. Gran, T. Dreibholz, and A. Kvalbein, "NorNet Core – A Multi-Homed Research Testbed," *Computer Networks, Special Issue on Future Internet Testbeds*, 2013.

[27] C. Parsa and J. Garcia-Luna-Aceves, "Differentiating congestion vs. random loss: a method for improving TCP performance over wireless links," in *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE*, vol. 1. IEEE, 2000, pp. 90–93.

[28] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 2, pp. 216–228, 2003.

[29] Y. Tian, K. Xu, and N. Ansari, "TCP in wireless environments: problems and solutions," *Communications Magazine, IEEE*, vol. 43, no. 3, pp. S27–S32, 2005.

[30] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to TCP throughput prediction," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 1. ACM, 2007, pp. 97–108.

[31] K. Winstein and H. Balakrishnan, "TCP ex machina: computer-generated congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 123–134, 2013.

[32] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan, "An experimental study of the learnability of congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 479–490, 2015.

[33] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 509–522.

[34] I. Livadariu, S. Ferlin, O. Alay, T. Dreibholz, A. Dhamdhere, and A. Elmokashfi, "Leveraging the IPv4/IPv6 identity duality by using multi-path transport," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2015, pp. 312–317.

[35] M. Bagnulo, P. Eardley, A. Ford, A. García-Martínez, A. Kostopoulos, C. Raiciu, and F. Valera, "Boosting mobility performance with multipath TCP," in *Future Network and Mobile Summit, 2010*. IEEE, 2010, pp. 1–8.

[36] 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*, ser. CellNet '12. ACM, 2012, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/2342468.2342476

[37] B. Oh and J. Lee, "Feedback-based path failure detection and buffer blocking protection for MPTCP," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3450–3461, 2016. [Online]. Available: http://dx.doi.org/10.1109/TNET.2016.2527759

[38] S. Barré, C. Paasch, and O. Bonaventure, "MultiPath TCP: From Theory to Practice," in *Proceedings of the 10th International IFIP Networking Conference*, Valencia/Spain, May 2011, pp. 444–457, ISBN 978-3-642-20756-3.

[39] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving Datacenter Performance and Robustness with Multipath TCP," in *Proceedings of the ACM SIGCOMM*, Toronto/Canada, Aug. 2011.

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

[41] V. Sharma, S. Kalyanaraman, K. Kar, K. K. Ramakrishnan, and V. Subramanian, "MPLOT: A transport protocol exploiting multipath diversity using erasure codes," in *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*, 2008, pp. 121–125.

[42] M. Kim, A. ParandehGheibi, L. Urbina, and M. Meedard, "CTCP: Coded TCP using multiple paths," *arXiv preprint arXiv:1212.1929*, 2012.

[43] Y. Cui, X. Wang, H. Wang, G. Pan, and Y. Wang, "FMTCP: A fountain code-based multipath transmission control protocol." in *ICDCS*. IEEE Computer Society, 2012, pp. 366–375.

[44] M. Li, A. Lukyanenko, S. Tarkoma, Y. Cui, and A. Ylä-Jääski, "Tolerating path heterogeneity in multipath TCP with bounded receive buffers," *Computer Networks*, vol. 64, pp. 1–14, May 2014.

[45] J. Wu, C. Yuen, B. Cheng, M. Wang, and J.-L. Chen, "Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks," *IEEE Transactions on Mobile Computing*, September 2016.

[46] J. Wu, C. Yuen, B. Cheng, Y. Yang, M. Wang, and J. Chen, "Bandwidth-efficient multipath transport protocol for quality-guaranteed real-time video over heterogeneous wireless networks," *IEEE Transactions on Communications*, vol. 64, no. 6, pp. 2477–2493, June 2016.

[47] J. Padhye and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC 3448, Mar. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc3448.txt

[48] A. Garcia-Saavedra, M. Karzand, and D. J. Leith, "Low delay random linear coding and scheduling over multiple interfaces," *arXiv preprint arXiv:1507.08499*, 2015.

[49] M. Kim, J. Cloud, A. ParandehGheibi, L. Urbina, K. Fouli, D. Leith, and M. Médard, "Network coded tcp (CTCP)," *arXiv preprint arXiv:1212.2291*, 2012.

Simone Ferlin received her Dipl.-Ing. degree in Information Technology with major in Telecommunications from Friedrich-Alexander Erlangen-Nuernberg University, Germany in 2010 and her PhD degree from the University of Oslo, Norway in 2017. Her research interests lie in the areas of computer networks, transport protocols, congestion control, network measurements and data analysis. Her dissertation focuses on improving robustness in multipath transport for heterogeneous networks.

Stepan Kucera Dr. Stepan Kucera (S'14 M'06) is a senior research scientist in the Access Lab of Nokia Bell Laboratories in Ireland. In this role, he serves as a principal investigator and technical lead of multiple large projects aiming to create novel disruptive technologies for wireless networking with current focus on multi-connectivity and gigabit wireless technologies. His research innovates Nokia product portfolio in both fixed and mobile access as well as triggered commercialization interactions with leading global operators such as Verizon Wireless, China Mobile and British Telecom. His expertise lies mainly in the area of wireless and IP networking technology, including both 3GPP/IEEE/IETF standards as well as proprietary solutions. Stepan has filed over 50 patents and published over 30 book chapters, transactions and conference papers in peer-reviewed IEEE venues. He is also the (co)-recipient of several professional awards. Between 2008 and 2011, he was a research scientist at the New Generation Wireless Communications Research Center at the Keihanna Research Laboratories, NICT, Japan. He received his Ph.D. degree in Informatics from the Graduate School of Informatics at Kyoto University, Kyoto, Japan, in 2008. He is a Senior Member of IEEE, and actively serves on technical boards of major IEEE journals and conferences.

Holger Claussen Dr. Holger Claussen is leader of the Small Cells Research Department of Nokia Bell Labs located in Ireland and the US. In this role, he and his team are innovating in all areas related to future evolution, deployment, and operation of small cell networks to enable exponential growth in mobile data traffic. His research in this domain has been commercialised in Nokia's (formerly Alcatel-Lucent's) Small Cell product portfolio and continues to have significant impact. He received the 2014 World Technology Award in the individual category Communications Technologies for innovative work of "the greatest likely long-term significance". Prior to this, Holger was head of the Autonomous Networks and Systems Research Department at Bell Labs Ireland, where he directed research in the area of self-managing networks to enable the first large scale femtocell deployments from 2009 onward. Holger joined Bell Labs in 2004, where he began his research in the areas of network optimisation, cellular architectures, and improving energy efficiency of networks. Holger received his Ph.D. degree in signal processing for digital communications from the University of Edinburgh, United Kingdom in 2004. He is author of more than 100 publications and 120 filed patent families. He is Fellow of the World Technology Network, IEEE senior member, and member of the IET.

Ozgu Alay Dr. Ozgu Alay received the B.S. and M.S. degrees in Electrical and Electronic Engineering from Middle East Technical University, Turkey, and Ph.D. degree in Electrical and Computer Engineering at Tandon School of Engineering at New York University. Currently, she is a senior research scientist at Networks Department of Simula Research Laboratory, Norway and Associate Professor in University of Oslo, Norway. Her research interests lie in the areas of mobile broadband networks, multipath protocols and robust multimedia transmission over wireless networks. She is author of more than 50 peer-reviewed IEEE and ACM publications and she actively serves on technical boards of major conferences and journals.