future internet

MDPI

Article

Performance Impact of Nested Congestion Control on Transport-Layer Multipath Tunneling

Marcus Pieska [1], Andreas Kassler [1,2], Anna Brunstrom [1], Veselin Rakocevic [3,*] and Markus Amend [4]

[1] Department of Mathematics and Computer Science, Karlstad University, 65188 Karlstad, Sweden; andreas.kassler@th-deg.de (A.K.); anna.brunstrom@kau.se (A.B.)
[2] Intelligent Networks and Systems, Deggendorf Institute of Technology, 94469 Deggendorf, Germany
[3] Department of Engineering, City, University of London, London EC1V 0HB, UK
[4] Deutsche Telekom AG, Deutsche-Telekom-Alle 9, 64295 Darmstadt, Germany; markus.amend@telekom.de
* Correspondence: veselin.rakocevic.1@city.ac.uk

Abstract: Multipath wireless access aims to seamlessly aggregate multiple access networks to increase data rates and decrease latency. It is currently being standardized through the ATSSS architectural framework as part of the fifth-generation (5G) cellular networks. However, facilitating efficient multi-access communication in next-generation wireless networks poses several challenges due to the complex interplay between congestion control (CC) and packet scheduling. Given that enhanced ATSSS steering functions for traffic splitting advocate the utilization of multi-access tunnels using congestion-controlled multipath network protocols between user equipment and a proxy, addressing the issue of nested CC becomes imperative. In this paper, we evaluate the impact of such nested congestion control loops on throughput over multi-access tunnels using the recently introduced Multipath DCCP (MP-DCCP) tunneling framework. We evaluate different combinations of endpoint and tunnel CC algorithms, including BBR, BBRv2, CUBIC, and NewReno. Using the Cheapest Path First scheduler, we quantify and analyze the impact of the following on the performance of tunnel-based multipath: (1) the location of the multi-access proxy relative to the user; (2) the bottleneck buffer size, and (3) the choice of the congestion control algorithms. Furthermore, our findings demonstrate the superior performance of BBRv2 as a tunnel CC algorithm.

Keywords: 5G; ATSSS; heterogeneous wireless access; MP-DCCP; multipath; real-time traffic; transport layer; unreliable traffic

Citation: Pieska, M.; Kassler, A.; Brunstrom, A.; Rakocevic, V.; Amend, M. Performance Impact of Nested Congestion Control on Transport-Layer Multipath Tunneling. *Future Internet* **2024**, *16*, 233. https://doi.org/10.3390/fi16070233

Academic Editor: Franco Davoli

Received: 6 May 2024
Revised: 21 June 2024
Accepted: 21 June 2024
Published: 28 June 2024

1. Introduction

The demand for mobile network capacity is increasing rapidly, with Ericsson forecasting an increase in monthly data transmitted from 130 exabytes (EB) per month in 2023 to 403 EB per month by 2029 [1]. The fifth generation of cellular networks (5G) aims to address this escalating demand, in part by facilitating the simultaneous usage of multiple access technologies by the user equipment (UE)—thereby enhancing the efficiency—for both its proprietary cellular access network and non-trusted wireless access, such as WiFi. Notably, the introduction of Access Traffic Steering, Switching, and Splitting (ATSSS) in 3GPP Rel. 16 serves the purpose of enabling multi-access aggregation seamlessly, without necessitating alterations to applications or Internet servers. To realize ATSSS, a multipath proxy can be deployed, e.g., in the operator core network, transparently forwarding packets coming from the Internet over multiple paths, each path connecting a different wireless interface on the UE, effectively implementing multipath connectivity between the UE and the tunnel endpoint in the proxy. Apart from capacity aggregation, multipath offers additional advantages including fault tolerance and potential for load balancing, thus shifting traffic away from congested wireless access according to UE needs and congestion states of the different tunnels.

A considerable amount of research and standardization has been conducted to develop multipath protocols like MPTCP [2], MP-QUIC [3], MP-DCCP [4], and CMT-SCTP [5], with MPTCP emerging as the de facto standard for multipath support for TCP traffic. In the context of ATSSS, end-to-end TCP connections may be split at a TCP proxy and transformed into MPTCP spanning multiple paths between the proxy and the user. However, handling unreliable traffic poses a distinct challenge, e.g., by a multipath tunnel that may not guarantee reliable delivery. Moreover, any reliable protocol for which the connection cannot be split, such as end-to-end encrypted QUIC, would also require a multipath tunnel within the ATSSS architecture. Recently, MP-QUIC and MP-DCCP have emerged as viable candidates for providing multipath support for various transport-layer traffic, including UDP, QUIC, and TCP, as well as lower layer protocols such as IP or Ethernet. The unique capabilities of MP-DCCP, notably its ability to resolve packet scrambling, make it an attractive option for tunneling any type of traffic including TCP, UDP, or Ethernet, mitigating concerns regarding multipath head-of-line blocking.

In this paper, we employ both network emulations and a physical test bed (for a brief overview, see Figure 1) to assess the impact of different CC algorithms (CCA) on capacity aggregation within a multipath tunnel context. For network emulations, we use a user-space multipath tunneling framework based on MP-DCCP, where we evaluate the achievable performance when using both TCP and DCCP as tunneling protocols. A scheduler of particular interest in the ATSSS context is a strict priority scheduler [6], which prioritizes the cheaper path whenever available, known as the Cheapest Path First scheduler (CPF). Such cost-based scheduler, which uses the free WiFi connection whenever available, is likely to be preferred by telecom operators and users alike, as it can divert traffic from expensive cellular networks. We identify several aspects that influence achievable aggregation when using CPF and demonstrate that this impact is specific to the scheduler in use.

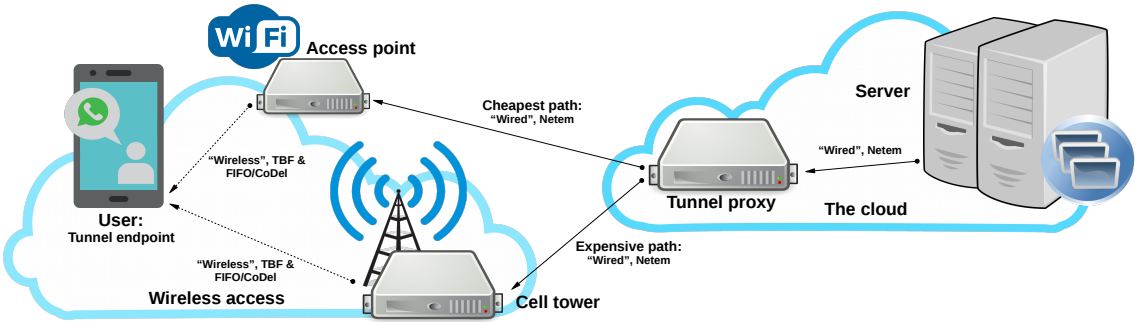

Figure 1. The topology used in our evaluation, with the corresponding TC queuing disciplines used.

Our results demonstrate both the gains when using multipath tunnels to increase the throughput over using just a single wireless access and the challenges to achieve effective capacity aggregation. We focus on the impact of nested congestion control, with one CC instance in the server and another CC instance for each path in the tunnel proxy. The challenges for nested CC are numerous, stemming from ATSSS proxy positioning, end-to-end RTT, tunnel reliability in the face of packet loss, bottleneck buffer size, flow size, and a combination of server and tunnel CCA. To illustrate the scale of the challenges, Figure 2 presents the download time when a multipath tunnel and the CPF scheduler are utilized, normalized against the download time when single-path end-to-end (E2E) transport is exclusively utilized. Ideally, the multipath solution should reduce the download time by 50%, as each path between the proxy and the UE is provided with the same capacity. A relative download time exceeding 100% demonstrates a worse throughput compared to using single path. The four bars in Figure 2 compare aggregation for far and near proxy deployments, when using either UDP or TCP end to end. The top two bars represent end-to-end UDP, showing a download time at approximately 50% for both proxy deployments. The bottom two bars depict end-to-end TCP, showing a download time which is cut in half when the proxy is situated close to the user but unchanged relative to single-path

download when the proxy is far from the user. In other words, the addition of an end-to-end congestion control mechanism may have a very negative effect on the aggregation, contingent on some other parameter, such as the proxy deployment.

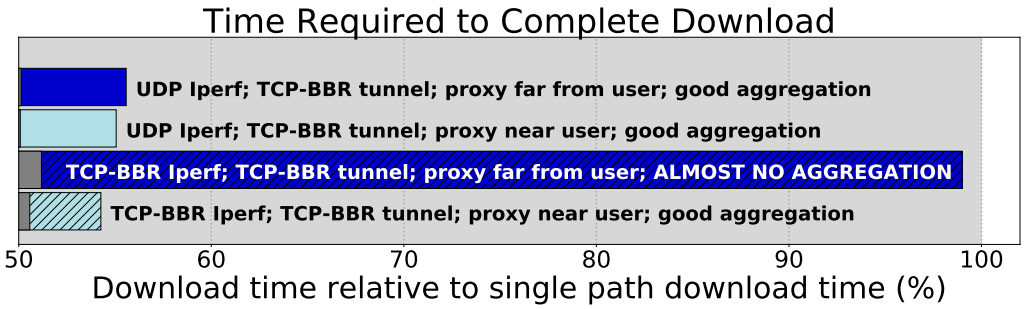

Figure 2. Experimental results illustrating the impact of nested CC under different configurations; gray bars show the standard deviation.

Our findings in this paper demonstrate several key insights: (i) BBRv2 emerges as the preferred choice for a tunnel congestion control algorithm (CCA), demonstrating its ability to effectively circumvent congestion and buffer overflows; (ii) partial reliability for the tunnel framework may be useful to mask loss between the UE and the proxy as some unreliable traffic may still use a CCA that reacts to loss; (iii) CPF performs poorly and is often unable to achieve aggregation; (iv) while our primary focus is on CPF, we also highlight that the commonly used Shortest Round-Trip Time First (SRTT) scheduler encounters fewer challenges within our multipath tunnel framework with respect to throughput aggregation.

The remainder of this paper is outlined as follows: Section 2 presents background information on MP-DCCP and our user-space multipath framework based on MP-DCCP. Section 3 describes the setup of our test environment and the evaluation scenarios used. Section 4 presents our results. Section 5 introduces related work. The paper concludes and outlines future work in Section 6.

2. The Multi-Access Framework

This section outlines the multipath solutions employed in our research. Section 2.1 provides an overview of MP-DCCP as implemented in the Linux kernel, while Section 2.2 discusses a user-space framework that supports MP-DCCP. Both frameworks serve as transport-layer multipath tunnels rather than end-to-end solutions, with shared schedulers and receiving side reordering modules utilized in our evaluation. Of particular relevance is the CPF scheduler, described in Section 2.3. We chose MP-DCCP over MP-QUIC, as the latter is primary-reliable. In addition, MP-DCCP is simpler to implement and offers modular packet reordering support.

2.1. The MP-DCCP Protocol

The Linux kernel MP-DCCP framework [7] represents a novel IP-compatible multipath architecture designed to support 5G-ATSSS mobile scenarios where multiple network interfaces are available at the user equipment (UE), typically WiFi and 5G. Additionally, it can be deployed to support home access network scenarios, where the access gateway contains both fixed access links (e.g., an ADSL uplink), and wireless access links (e.g., 5G cellular). Built upon DCCP [8], MP-DCCP inherits modular congestion control (CC) support similar to TCP, together with unreliable UDP-like transport services. MP-DCCP encapsulates packets and transmits them across distinct DCCP tunnels serving two main objectives: (i) the integrated DCCP CC facilitates path quality estimation in terms of loss rate, bandwidth, and round-trip time (RTT); (ii) the tunnels ensure traffic redirection to the aggregation termination point or proxy. Layer 3 compatibility is ensured via virtual network interfaces, facilitating seamless integration into both UE and home gateway environments. Packets entering the virtual network interface are distributed across individual DCCP tunnels by a

scheduler, which selects for the transmission of each packet an appropriate DCCP socket that is bound on a specific interface routing over a specific wireless access.

Figure 3 depicts the placement of scheduling, reordering, and congestion control instances within an MP-DCCP-based framework assuming downlink traffic (from the server on the Internet towards the UE). Here, the scheduling component is located in the proxy, which runs two separate CC instances over the two different paths spanning over the different access networks. Apart from CPF, other schedulers such as Shortest Round-Trip Time First (SRTT) and Out-of-Order Delivery for In-Order Arrival (OTIAS) [9] are available within the MP-DCCP framework. At the UE (in the donwstream direction) or at the proxy (for upstream packets, not shown in the Figure), the MP-DCCP framework can optionally reorder packets. Here, the reordering module processes the packets after it receives them from the respective tunnel endpoints. When a flow is distributed across multiple paths, packet scrambling may occur due to differences in path delays, leading to packets arriving out of sequence. The reordering module can adjust the timing of packets (denoted as Wait in the Figure) arriving earlier than expected, up to a predefined limit. This limit may either be static or adapt to variations in path delay, known as adaptive or dynamic reordering. Packet encapsulation with both global and path-specific sequence numbers facilitates this reordering process. Additionally, a mechanism for fast loss detection analyses received sequence numbers to identify lost packets, enabling the early release of delayed packets. Another strategy to mitigate packet scrambling and jitter involves enforcing delay equalization [10], which artificially delays packets received via the faster path by the difference in path delay experienced. For instance, if two paths exhibit delays of 10 and 30 ms, packets that need to be sent over the shorter path are delayed by an additional 20 ms. This kind of flexible release mechanism, when tuned properly, minimizes both packet scrambling and multipath head-of-line blocking, thereby making MP-DCCP an excellent candidate for tunneling traffic.

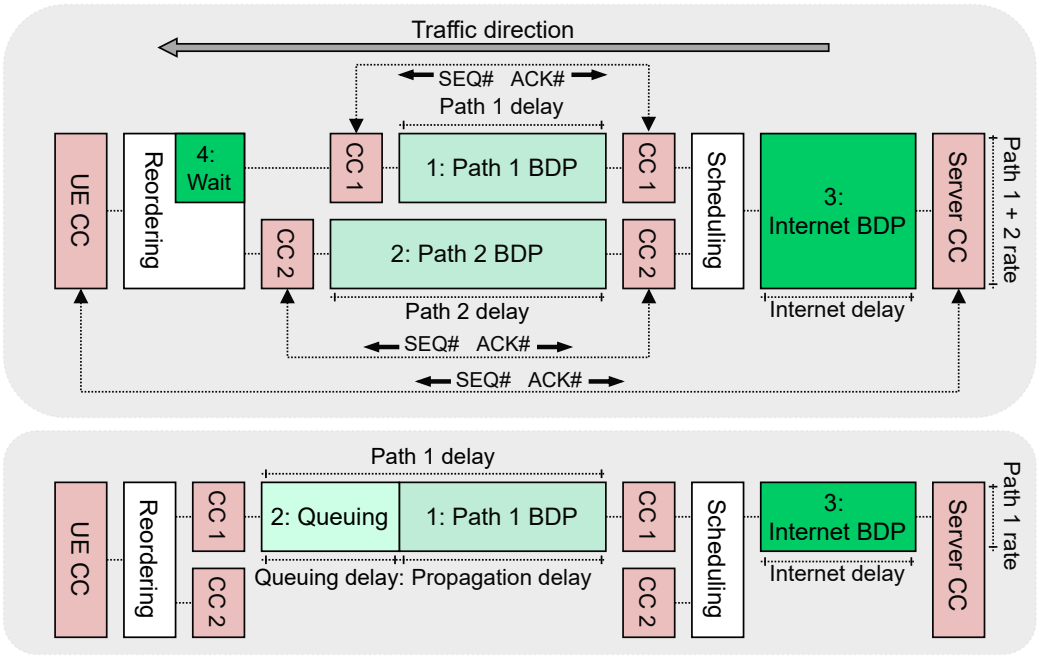

Figure 3. Nested Congestion control with 3 instances.

2.2. Protocol-Agnostic Multipath Framework

For flexible experimentation, we also implemented a protocol-agnostic user-space multipath framework supporting MP-DCCP. This framework operates as a multi-threaded application, receiving packets from a Linux TUN device and encapsulating them in a custom header, mirroring the function of kernel-space MP-DCCP headers. The header includes a global sequence number, a path sequence number, and a timestamp. The application

schedules packets across multiple single-path sockets, with separate threads used for sending and receiving for each traffic flow. A separate thread is used at the proxy side to listen for connection requests and establish new tunnels. This framework incorporates the CPF scheduler and offers the same reordering functionality as MP-DCCP. The user-space implementation permits the use of DCCP-based sockets per tunnel to resemble MP-DCCP functionality but also supports, e.g., TCP sockets. Consequently, it allows us to compare the performance when using multiple TCP tunnels (e.g., using TCP BBR) versus multiple DCCP tunnels when combined with the schedulers and reordering modules of MP-DCCP.

2.3. Multipath Scheduling

Our evaluation focuses on the Cheapest Path First (CPF) scheduler, a particular implementation of the more general strict-priority scheduler [6]. CPF prioritizes available paths based on user-defined preferences, such as favoring WiFi interfaces over others. For CPF, a path is considered available if its path congestion window (CWND) has sufficient room to send a packet. Consequently, CPF always schedules packets over the highest priority path if it has available capacity and only schedules packets to the next available path once the CWND of the highest priority path is filled up. This mechanism establishes a clear preference for certain paths, a preference that can also be set by the operator. We concentrated on CPF due to its significance in 5G ATSSS scenarios, where the utilization of cost-effective or free-access links, such as WiFi, is preferred compared to more expensive and volume-restricted cellular access.

2.4. Nested Congestion Control

The utilization of multipath tunnels, as introduced in Section 1, with throughput enhancement by path aggregation as the main performance objective, raises several research questions. The primary objective of employing multipath tunnels is to facilitate the simultaneous efficient utilization of multiple paths. Previous studies have predominantly focused on single-path TCP-over-TCP encapsulation [11–13] or end-to-end (E2E) transport-layer multipath protocols like MPTCP. However, employing multipath tunnels for E2E traffic introduces the complexity of parallel congestion control (CC) across different paths nested within the server CC loop.

Figure 3 demonstrates different scenarios. The server congestion control operates end to end and in addition to this, there is a per-path congestion control necessary for the operation of multipath tunnels. Full aggregation over multipath requires that the in-flight traffic maintained by the server CC is *equal to or greater than* the sum of the four (green) bandwidth-delay-product (BDP) components. Subtle issues arise when you consider that, e.g., the path 1 BDP also contains the buffering at the path 1 bottleneck. Thus, aggregation might depend on the quantity of data the path 1 CC maintains in the bottleneck buffer. This is illustrated in the bottom figure in Figure 3, where there is an added BDP component on path 1, but path 2 is not aggregated.

Nested congestion control results in unforeseen side effects, and a detailed identification and analysis of these side effects is essential for the development of an optimal multipath solution. This paper offers detailed insight into this problem. The multipath scheduler, responsible for determining each packet's path and potential packet reordering, adds further complexity to the scenario.

3. Related Work

The work described in this paper rests at the intersection of (i) transport-layer congestion control, (ii) multipath, and (iii) tunneling. Although there is a plethora of related research, previous work has focused only on some specific aspects. While there is work on TCP-in-TCP single-path tunneling, the effects investigated in this paper do *not* appear *without* a multipath packet scheduler and having multiple paths. Similarly, much of the work on multipath scheduling does not consider the performance if there is a congestion control algorithm running on top of the scheduler, *which is the case in the tunneling context*.

Traditionally, CC in multipath networks has been considered an important challenge since the early works on integrated routing and CC in multipath networks [14] and in the general case of wireless mesh networks [15]. MPTCP, the dominant multipath transport-layer protocol, establishes a connection composed of multiple sub-flows, each one managing its own individual CC and re-transmission states. A known design problem with CC in MPTCP is presented in [16], and MPTCP CC approaches are summarized in [17]. Originally, each MPTCP sub-flow CC instance was independent, i.e., uncoupled. However, due to unfairness when run in parallel with single-path TCP flows [18], new semi-coupled and closely coupled algorithms emerged that limited the increase in the congestion window variable on some paths. Although the aforementioned research did address *parallel* congestion control, it did not address the interactions between *nested* congestion control approaches which arise when a transport-layer tunnel is deployed; the latter is the focus of this paper, and is made highly relevant by, e.g., the 5G ATSSS architecture.

Other work [19] has shown the limitations of coupled CC in wireless networks, where the *head-of-line* (HoL) blocking problem of MPTCP is further increased due to the volatility of wireless links, and packet loss due to transmission errors and re-transmissions negatively affect both the average throughout and its stability. MP-QUIC [3] can overcome HoL blocking with QUIC's stream-based semantic, demonstrating an advantage over TCP, which lacks streams. However, multipath HoL blocking remains a potential issue as long as a flow or a stream is *split* over multiple paths and is particularly pronounced if the protocol is reliable and thereby *never* release packets out of order. Other avenues that have been explored in CC in multipath environments include the application of reinforcement learning [20], or deep-Q learning [21]. While this is interesting from our perspective, our attempt in this paper is primarily to identify performance problems. Another approach is using forward error correction (FEC) to help with the head-of-line blocking [22]. FEC or network coding has been combined with rate allocation on different paths to maximize the performance in some of the works [23,24]. Finally, solutions have been proposed for application-layer congestion management for video streaming and real-time video traffic in multipath [25,26], utilizing delay information to address congestion earlier compared to when using loss-based algorithms.

Multipath solutions may be deployed at different levels in the network stack. A multipath tunnel solution at the *network layer* [10,27,28] has the advantage that the Internet Protocol (IP) is universally used, and that *any* transport-layer protocol can lay on top, i.e., *a single architecture can serve all traffic types*. The main advantage of splitting or converting at the *transport layer* [29,30] is that many transport-layer protocols include CC and therefore also path state detection; however, any such solution can only handle a single transport-layer protocol; for instance, a solution that converts between TCP and MPTCP [31,32] can only serve TCP traffic. The above examples of multipath proxies and anchors are either at a layer *other than* the transport layer or are based on *converting* a single-path protocol to its multipath analogue at a proxy, rather than tunneling at an anchor. If a multipath transport-layer solution is implemented *as a tunnel rather than a protocol converter*, the tunnel may serve *all* traffic types, but as demonstrated in this paper, issues of adverse interactions between nested CC instances arise.

When transport-layer tunnels have been considered, they have typically been *single-path* tunnels. Issues explored include things like how to use TCP tunnels to force CC upon UDP [13], the usage of TCP-tunnels with network coding over high loss links [33], the performance difference between TCP and UDP tunnels [12], or when TCP-over-TCP is helpful and when it is harmful [11]. Obviously, that research lacks the multipath component which is a key aspect of this paper. A very recent development is *Multiplexed Application Substrate over QUIC Encryption* (MASQUE) [34,35], whereby multiple end-to-end connections may use a single QUIC tunnel. One possible use case is for end-to-end QUIC; as end-to-end QUIC traffic is encrypted, a tunnel is required to support network functionality that has historically been implemented as a TCP proxy [36]. With a future multipath QUIC, MASQUE could serve a function very much similar to MP-DCCP in this paper.

There are several CC algorithms (CCA) in common use today. These typically regulate the sending rate by limiting the quantity of in-flight data to be no more than the congestion window (CWND). NewReno increases its CWND linearly and is thereby often bad at utilizing *high BDP* paths, *i.e., paths having a high throughput or a long RTT*. Cubic [37], on the other hand, increases its CWND using a cubic function. The rationale behind this design choice is that a cubic function helps the CCA to plateau at a CWND that ensures the full utilization of the path, without consistently filling the bottleneck buffer. Using a cubic function also more rapidly utilizes high BDP paths. BBR is designed based on a run-time model of the path that BBR maintains [38]. This requires BBR to monitor the achievable throughput and the minimum round-trip time. By multiplying these two into a BDP, BBR may achieve the full utilization of the link by having a number of bytes in-flight which is equal to or greater than that of the BDP. BBR version 2 (BBRv2) is currently being developed as an improvement on the original BBR and will, among other things, react to packet loss, *unlike BBR*. To understand the issues that arise in the paper, and how to solve them, it becomes necessary to understand and analyze the CC algorithms used. The particulars of a given tunnel CCA may well drive certain problems, as well as constitute a solution in other circumstances; as our results demonstrate. Although we do not take this step in the paper, it is fairly clear that the problems we have illustrated could be solved by using a CCA designed for the transport-layer multipath tunneling context.

Packet scheduling algorithms are actively being researched. It is known that a bad traffic scheduling strategy cannot fully utilize multipath resources, and that too many data sent over a long-RTT path may cause blocking on that path [39]. The min RTT scheduler [40] is the default scheduler for MPTCP and prioritizes the path with the shortest estimated RTT. However, min RTT does not estimate how much traffic should be sent over each path and can be improved, for example, by using a Blocking Estimation-based MPTCP scheduler algorithm [41], which adds such an estimation. More recent work consider how to avoid blocking for streams in QUIC [42]. Other scheduling algorithms include Earliest Completion First [43], Peekaboo [44], DEMS [45], and QSAT [46]. Our work here focus on the CPF scheduler, which, much like nested CC, is made particularly relevant given the 5G ATSSS architecture, where a cheap path, e.g., WiFi, is often available.

4. Evaluation Setup

The user-space framework described in Section 2.2 was integrated with Mininet to set up the evaluation topology, comprising a multipath tunnel over emulated WiFi and cellular access links having constant capacities. The evaluation topology was identical to the topology provided in Figure 1. The topology consisted of a server running end-to-end congestion control, the tunnel proxy point which served as an end point of the multipath framework, the two independent access points representing different access networks (WiFi and cellular), and finally, user equipment, which worked as the end point for both end-to-end congestion control and also for the multipath. In the Mininet configuration and for all our testing, we used static wireless links. While a more realistic deployment of ATSSS would include dynamic links, we performed our evaluation over static links to isolate and fully understand the underlying performance issues. The WiFi path was considered to be cheaper and was designated as the preferred path by the CPF scheduler. We used TC-Netem to modify the link delays between the server and the proxy and between the proxy and the access point/cell tower, allowing us to evaluate the impact of the proxy placement (e.g., in the operator network versus in the cloud). The MAC-layer bottleneck buffers were located at the access point and at the cell tower. Bottleneck links were created using TC-TBF. We also explored the impact of active queue management (AQM), in which case CoDel was enabled as the AQM at the bottleneck links. We used Iperf to create TCP traffic that emulated greedy downloads from the server to the UE. Our rationale was that TCP successfully represented both TCP and QUIC traffic. In the tunnel framework, we compared the achievable performance when using DCCP vs. TCP tunnels over the access networks, primarily to evaluate the impact of having reliability over each tunnel path

(when using TCP), while retaining the MP-DCCP sender (scheduler) and receiver modules (reordering) as shown in Figure 3.

4.1. Congestion Control Algorithms

Our evaluation focused on the performance impact of employing different congestion control algorithms (CCA) within a multi-access multipath tunneling framework. We used NewReno or BBR as tunnel CC when TCP tunnels were deployed, and DCCP's CCID2 [47] or CCID5 [48] when using DCCP tunnels. This enabled the comparison of TCP and DCCP, as CCID2 and CCID5 are implementations of NewReno and BBR, respectively. We also used BBRv2 as tunnel CCA, which had not yet been implemented for DCCP. Furthermore, we used Cubic or BBR as server CC for TCP-Iperf. More details on all CCAs can be found in Section 5. Considering that the focus of this study was to analyze the challenges of nested congestion control, coupled congestion control in the multipath layer was not included in the evaluation. Rather, the tunnel CCs worked independently from each other. Results obtained using TCP E2E are applicable to other E2E protocols, like QUIC, that use similar CCAs as TCP. Figure 3 illustrates where each CC instance was located.

4.2. Baseline Configuration

As we have seen in the presentation of the problem of nested congestion control, the actual distribution of delay is highly likely to have a significant influence on the performance of nested congestion control. We therefore needed to perform a detailed evaluation using a realistic range of RTT values. To create realistic scenarios, we measured actual RTT values to establish benchmarks for detailed evaluation. Over WiFi, we measured a baseline RTT of 32.6 ms to a national server and 83.8 ms to an international server. Doing the same over cellular access resulted in an RTT of 75.1 ms to a national and 84.0 ms to an international server. Based on this, we chose the default RTT values as presented in Table 1. The default RTT corresponded roughly to the RTT values received when pinging local and national servers, with the tunnel proxy assumed to be located halfway between the server and the user. We then performed evaluations on a range of RTT values, as presented in greater detail in Section 5. Table 1 also presents other parameters for the default configuration used in our Mininet-based emulation.

Table 1. Default Mininet configuration.

Parameter	Value
Server link RTT	20 ms
Path 1 RTT	20 ms
Path 2 RTT	40 ms
Path 1 and 2 bandwidth	50 Mbps
Path 1 and 2 MAC buffer	500 pkt
Path 1 and 2 queuing discipline	FIFO
Path 1 and 2 drop rate	0%
Scheduler	CPF
Scheduler buffer	1500 pkt
Reordering	Adaptive
Reorder buffer	1500 pkt
Moving average weight	95%
Added margin	10 ms

The MAC buffer size in Table 1 refers to those buffers in each path that act as bottleneck buffers, where packets queue up when the tunneling system is transmitting packets at a rate higher than what the bottleneck can deliver (i.e., the WiFi or cellular link); the size of these buffers were set to the BDP produced when multiplying the path throughput with an RTT of 100 ms—meaning that the maximum queuing delay was also 100 ms. When an extra-large or extra-small buffer was used, the size was changed to 50% or 200% of the default size, respectively. The size of the scheduler buffer was three times the size of the

bottleneck buffer. When estimating the RTT at the receiving side, we used an exponentially weighted moving average, with each new value determining 95% of the estimation. We also padded the reorder wait limit with 5 ms, and thereby erred on the side of solving packet scrambling at the expense of the E2E delay.

4.3. Evaluation Metrics and Parameters

We assessed throughput as the key performance metric, determined by analyzing the end-to-end TCP ACK numbers returned by the receiver. Additionally, we measured download completion time (DLCT) for a 300 MB file download. When calculating the throughput of individual paths, we used the in-built tracing of the user-space framework, which logs packet transmissions and receptions. To evaluate the factors that impact the ability of our multipath solution to achieve throughput aggregation and reduce download times using a particular combination of CC algorithms, we analyzed the impact of the following: (i) end-to-end RTT; (ii) the position of the tunnel proxy; (iii) the size of the MAC-layer bottleneck buffer; (iv) asymmetry in the path delay; (v) the presence of CoDel at the bottleneck.

4.4. Additional Scenarios

In addition to the default configuration given in Table 1, Table 2 provides an overview of our evaluation scenarios and highlights how they differ from the default configuration. If a parameter remains unchanged from Table 1, it is not explicitly mentioned in Table 2. Each scenario was given a descriptive name, along with a shorthand key, like SHORT or ASYM200. Note that some of these could be combined, for instance, a short RTT with the proxy located close to the user (denoted as SHORT and NEAR). Some scenarios were relative, such as the proxy's proximity to the user, or the presence of a large asymmetry. These scenarios were used to create a standard set of tests.

Table 2. Scenarios used in the evaluation.

Name	Key	RTT	Path 1 RTT	Path 2 RTT
Short RTT	SHORT	40	20	40
Long RTT	LONG	80	40	60
Proxy near user	NEAR	n	0.2n	0.2n + 20
Proxy far from user	DIST	n	0.8n	0.8n + 20
No asymmetry	ASYM0	n	0.5n	0.5n
Large asymmetry	ASYM40	n	0.5n	0.5n + 40
Excessive asymmetry	ASYM200	n	0.5n	0.5n + 200

4.5. Testbed Configuration

We set up a testbed using the MP-DCCP Linux kernel implementation (available from https://multipath-dccp.org, accessed on 5 May 2024) to verify certain results obtained in the user-space framework. While the topology remained the same, there were some variations: the proxy connected to the Internet and downloaded files from real servers. As the entire topology was not under our control, scenarios within this context deviated slightly from those listed in this section in terms of, e.g., RTT. One key difference was that the testbed links between the proxy and the UE had a lower rate (10 Mbps each instead of 50 Mbps). To maintain the same maximum queuing delay, the MAC-layer bottleneck buffer was scaled down by a factor of 5.

5. Results

In our evaluation, we aimed to determine which factors impacted the capacity aggregation of our multi-access system, for different CCA combinations. We pick up from the results in Section 1, where aggregation varied from nearly optimal to non-existent. In particular, in our evaluation, we aimed to answer the following questions:

- How does the proxy deployment impact the performance of the multipath framework (cf. Section 4.1)?
- What is the impact of bottleneck buffer sizes for different congestion controls (cf. Sections 4.2 and 4.3)?
- How do different nested congestion control algorithm combinations impact download completion times (cf. Section 4.4)?

To present our results, we plotted examples of achievable throughput over time, with the per-path throughput stacked in graphical plots. Additionally, we plotted the average throughput for multiple tests using bar diagrams. For a more comprehensive presentation, we used contour plots to illustrate the impact of the proxy placement and end-to-end RTT. This involves varying both the end-to-end RTT and the percentage of that RTT within the tunnel itself. For instance, if the end-to-end RTT was 40 ms and the tunnel RTT accounted for 25%, then the tunnel RTT was 10 ms and the RTT from proxy to server was 30 ms. We organized the results in batches of 400 tests, where both the end-to-end RTT and the tunnel RTT were randomly selected. The former ranged from 0 to 100 ms, and the latter was determined as a percentage of the end-to-end RTT, ranging from 0 to 100. To distribute the points more evenly, any point too close to a previous point was discarded and regenerated. The same distribution of points was used for all batches. To convey the information effectively, we used color coding. A deep blue color indicates good aggregation, with throughput reaching 95 Mbps—double the single-path throughput. A deep red color indicates no aggregation at all, with throughput being as low as 40 Mbps—less than the single-path throughput.

In Sections 4.1–4.3, we look primarily at one single combination of tunnel and end-to-end CCA, namely, the combination most impacted by the observed phenomenon. Finally, in Section 4.4, we present the findings from the original set of tests, which prompted the deeper evaluation presented in Sections 4.1–4.3.

5.1. Impact of Proxy Deployment When Using BBR over BBR as Congestion Control

As illustrated in the introduction, achievable performance is heavily influenced by the proxy's location relative to the user (see Figure 2). This is mainly due to nested congestion control and buffer bloat, which does not push enough packets so that the CPF scheduler would use the available capacity on the second path. We expected that any solution capable of mitigating buffer bloat would likely improve performance. We focused here on BBR over BBR for the CCs, using both TCP with BBR and BBRv2 (end to end) and two DCCP tunnels between the proxy and the UE using BBR (CCID5) for the DCCP tunnel.

Figure 4 shows the per-tunnel throughput over time for four experimental tests when BBR was the end-to-end congestion control algorithm applied. Each of the two paths offered a maximum of 50 Mbps, and it can be seen from Figure 4 that different scenarios offered very different levels of traffic aggregation over multiple paths. In the top case, when RTT was short and the server was positioned close to the proxy (the starting point of the multipath tunnels running BBR), full aggregation was achieved. In the second case, also using BBR tunnels but with the server now being far away from the proxy, there was no aggregation and only one path was used. The CCID5 tunnels—plots 3 and 4—resulted in partial aggregation. The reasons for the results presented in Figure 4 are explained in more detail below, using a larger set of results given in Figure 5. In that Figure, the tunnel server-side endpoint location is specified as the tunnel RTT, as a percent of the end-to-end RTT; a high percent means the proxy (PRX) is close to the server.

The following key take-away messages highlight our observations: **A proxy near the user increases the aggregation ratio**. According to Figure 5, minimal aggregation is achieved when approximately 20% or less of the end-to-end RTT is spent between the server and proxy, as indicated near the upper x-axis. The color scheme uses white to indicate a secondary path utilization of approximately 50% or an aggregated throughput of around 70 Mbps. The deep red indicates no aggregation at around 45 Mbps throughput. Poor

aggregation results from the server and tunnel BBR instances measuring a similar minimum RTT, leading to similar timing and poor interaction between the two CC control loops.

Aggregation does not occur if the end-to-end RTT is short. Figure 5 shows minimal aggregation when the end-to-end RTT is less than approximately 20 ms, as demonstrated near the left y-axis. This result is particularly disappointing, given the general expectation that a short RTT would typically be advantageous for aggregation in a distributed system. Interestingly, this issue applies to a lesser extent when tunneling BBR over CCID5, although tunnels shorter than about 10 ms notably restrict aggregation. This problem appears to be driven by BBR exhibiting an effective gain higher than two for low RTTs, and for low RTTs, the actual time difference between the two CC control loops is small, even for larger relative differences.

BBRv2 tunnels achieve better aggregation that BBR tunnels. Figures 6 and 7 illustrate aggregation achieved over BBRv2, normalized against (left) not using a tunnel and (right) using BBR tunnels. In details, Figure 6 compares the aggregate throughput of BBRv2 tunnels against single-path transport, while Figure 7 compares BBRv2 against multipath transport over BBR. The latter shows that BBRv2 achieves higher aggregation than BBR when either (i) the tunnel RTT is really long, or (ii) CoDel is used at the bottleneck, or (iii) the bottleneck buffer is relatively small. This is not a surprising result, considering that BBRv2—unlike BBR—reacts to loss, and all the aforementioned scenarios will generate proportionally greater packet loss, considering the buffer gets smaller in terms of BDP.

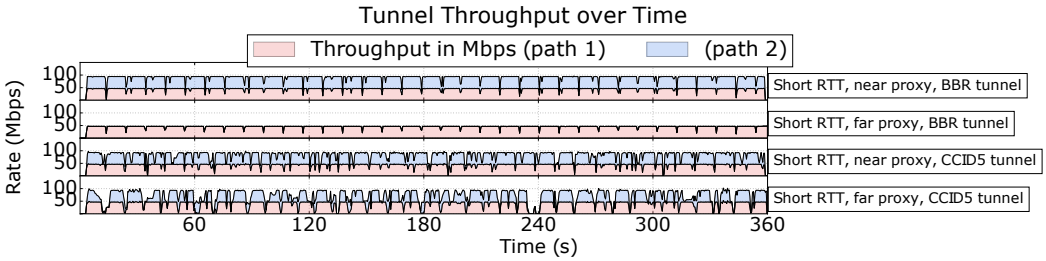

Figure 4. Tunneling BBR over BBR or CCID5. Red and blue are for paths 1 and 2, respectively.

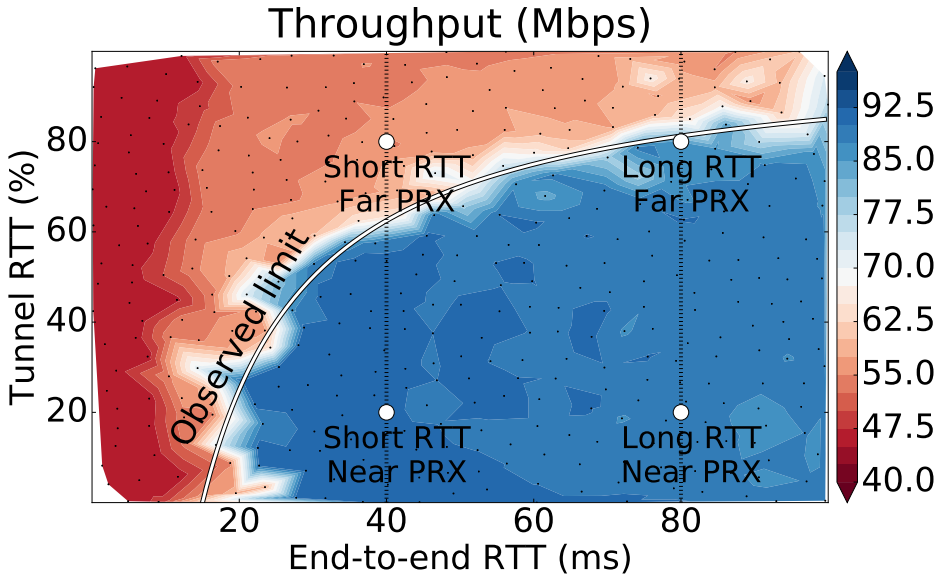

Figure 5. Tunneling BBR over BBR or CCID5. No asymmetry. PRX is short for proxy.

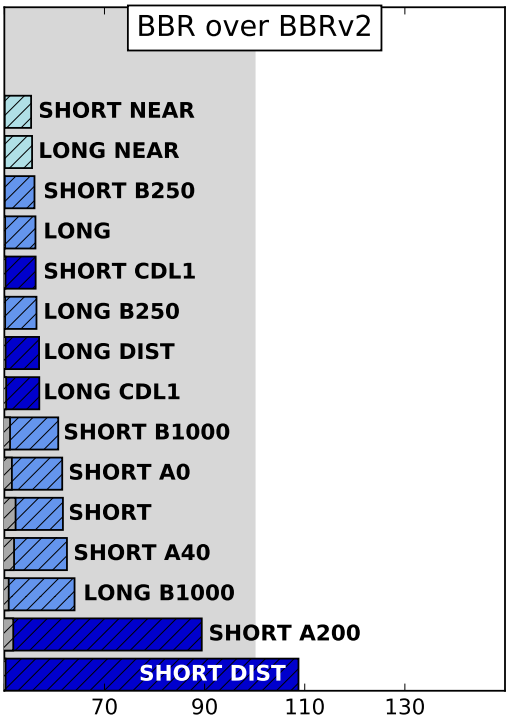

Figure 6. Norm. vs. not using a tunnel.

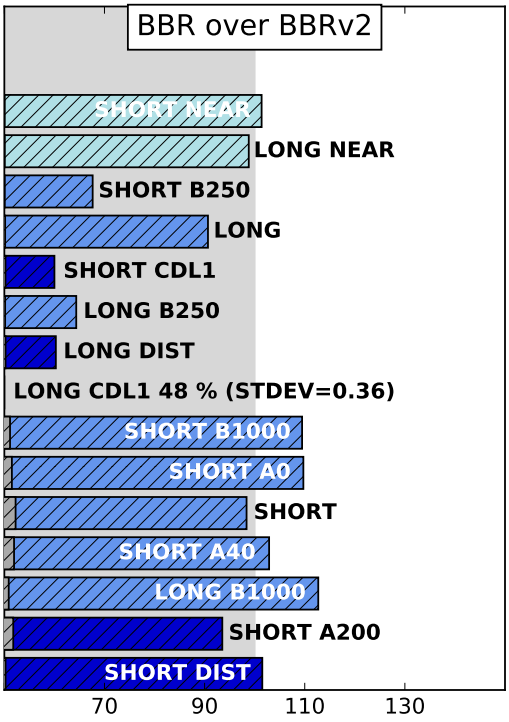

Figure 7. Norm. vs. using BBR tunnel.

5.2. Cubic over BBR and Inadequate Bottleneck Buffer Size

In this section, we analyze how the degree of the achieved aggregation depends on the bottleneck buffer size relative to the number of packets that BBR attempts to keep within that buffer. It is worth noting that this number is typically around one bandwidth-delay product.

Figure 8a shows four examples of throughput over time. As with the plots given in Figure 4, in Figure 8a, we observe the level of aggregation for four different testing

scenarios. This time CUBIC end-to-end congestion control is applied over different tunnels and with a different location of the server related to the multipath proxy. We can see again that only in the top case is full aggregation with a 100 Mbps rate achieved, with the other three scenarios offering much lower rates of aggregation. The reasons for this are explained in detail below, using the results from Figure 8b.

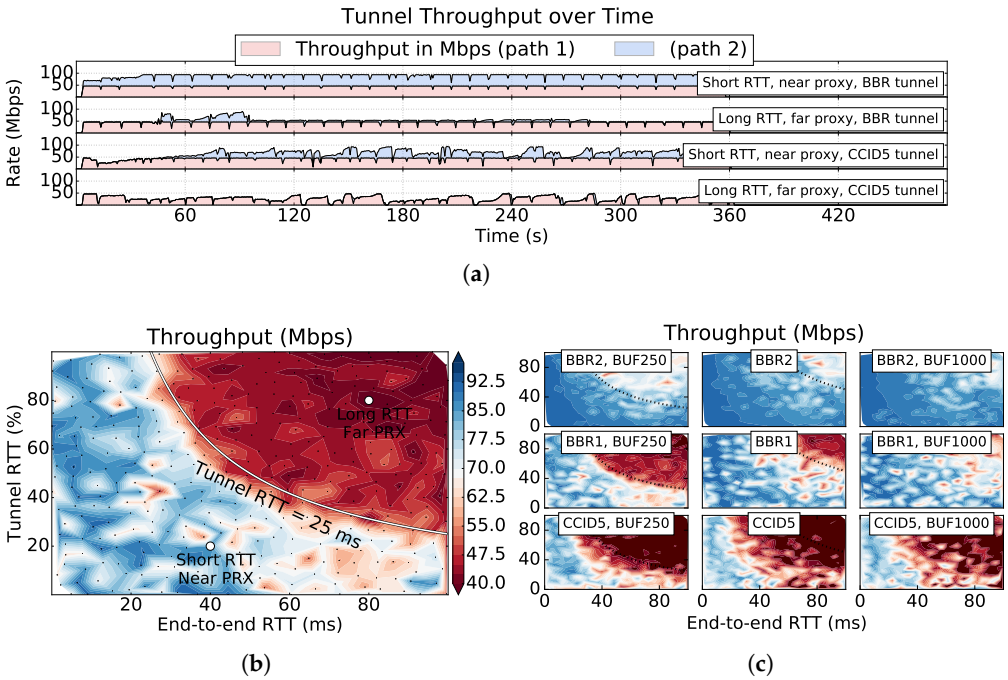

Figure 8. Tunneling Cubic over BBR or CCID5. (**a**) Tunneling Cubic over BBR/CCID5. Blue and red are for paths 1 and 2, respectively; (**b**) Path asymmetry: 10 ms, buffer size: 250; (**c**) Variable bottleneck buffer sizes and congestion control algorithms.

Figure 8b,c show contour plots of a large set of results; the tunnel server-side endpoint location is specified as the tunnel RTT, as a percent of the end-to-end RTT; a high percent means the proxy is near the server. For each plot, the path asymmetry was set to 20 ms. The circle segment lines indicate a constant tunnel RTT corresponding to the maximum queuing delay, for the following bottleneck buffer sizes: a 25 ms tunnel RTT when the buffer was 250 packets large; 50 ms when the buffer was 500 packets large; no line when the buffer size was 1000 packets.

A long tunnel RTT reduces aggregation. In Figure 8b, a circular segment line serves as a reference, representing all points where the tunnel RTT is constant at 25 ms. Clearly, a steep slope extends from the origin to the top-right corner, suggesting a significant influence of the tunnel RTT on aggregation. This is consistently observed across all nine plots in Figure 8c, albeit not as steep. As we move along this gradient towards the top-right corner, BBR/CCID5 becomes more likely to overflow the bottleneck buffer. Figure 8a illustrates this issue using time-series plots—the longer RTT results is less aggregation.

A small bottleneck buffer reduces aggregation. Examining the middle row in Figure 8c, we observe the red circle segment shifting towards the upper-right corner as the bottleneck buffer size increases. A larger bottleneck buffer reduces the risk of the buffer overflowing when tunneling Cubic over BBR. Given the sharp shift from blue to red in the middle row, it is possible to notice a hard limit to how small the buffer may become before it has a severe impact on the aggregation. This pattern reproduces itself in the other two rows, highlighting the necessity for the bottleneck buffer to be large relative to the tunnel RTT.

BBRv2 tunnels enable more aggregation than BBR tunnels. Examining the top row of Figure 8c, we observe that BBRv2 yields more aggregation compared to BBR or CCID5.

This is not surprising as BBRv2 reacts to loss to some degree, whereas the other two CCIDs do not. Therefore, we may expect that BBRv2 will avoid buffer overflows to a greater extent than BBR or CCID5. Poor aggregation is in this case caused by BBR or CCID5 trying to maintain more in-flight packets than the bottleneck buffer can hold.

A receiver wait limit that allows for re-transmissions is beneficial. When using BBR as tunneling protocol across a set of scenarios, we found that (i) an added margin of 150 ms decreased the download completion time (DLCT) to 68.7% of the single-path DLCT, and 76.2% of the multipath DLCT with the standard margin of 5 ms; when (ii) using BBRv2 as tunneling protocol, the corresponding numbers were 65.4% and 74.0%, respectively.

5.3. BBR over NewReno and Excessive Bottleneck Buffer Size

Figure 9a shows four examples of throughput over time generated in the user-space framework, while Figure 9b depicts results from the MP-DCCP testbed. Figure 9c,d show contour plots of a large set of results; the tunnel server-side endpoint location is specified as the tunnel RTT, as a percent of the end-to-end RTT; a high percent means the proxy is near the server. In the figures, the prefixes *A* and *B* denote the asymmetry in ms and buffer size in packets. RTT values obtained in the kernel-space testbed are labeled * while the RTT values set in the user-space emulation are labeled **.

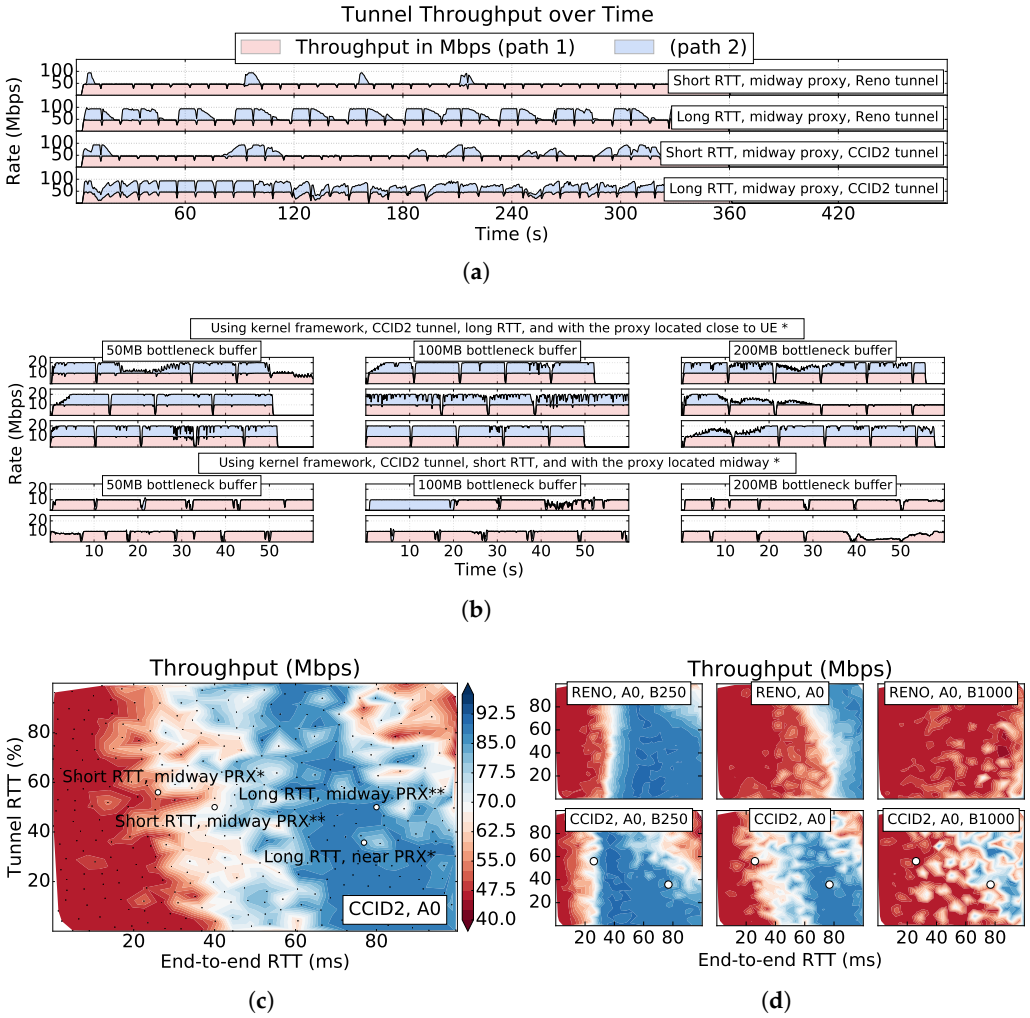

Figure 9. Tunneling BBR over NewReno or CCID2. (**a**) Downloads with BBR over NewReno or CCID2. Blue and red are for paths 1 and 2, respectively; (**b**) Downloads with BBR over CCID2, the RTT is *long*. Blue and red are for paths 1 and 2, respectively; (**c**) Contour plot of a large set of results; (**d**) Contour plots for different scenarios.

This section examines an issue that becomes particularly evident when tunneling BBR over NewReno. Rather counter-intuitively, the challenge here is that the RTT has to be long enough so that BBR at the server retains more in-flight traffic than tunnel-NewReno can sustain over the primary path. It is important to highlight that this presents an opposite problem to the one described in section IV-B.

A short RTT inhibits aggregation. In Figure 9c, we see a nearly vertical red-blue divide where the end-to-end RTT is around 40 ms. Figure 9 plots the throughput for different results, while Figure 9a provides further real-world testbed results obtained using the Linux kernel implementation of MP-DCCP. Matching these results with the corresponding points in Figure 9d demonstrates that these examples are consistent with the results in Figure 9c.

Excessive bottleneck buffer size inhibits aggregation. The vertical red-blue divide in Figure 9c is reproduced in all six plots in Figure 9d, albeit not at the same RTT. The key pattern that can be observed is the shifting of the vertical line to the right as the bottleneck buffer size increases. It is well known that NewReno tends to fill bottleneck buffers, leading to the buffer size determining the tunnel RTT. As a result of this, a larger bottleneck buffer necessitates a longer end-to-end RTT when the average number of packets kept in-flight end to end by BBR exceeds the number kept in-flight by NewReno over the primary path tunnel.

5.4. Performance Comparison of Nested CCA Combinations

This section presents the achieved aggregation results for all CCA combinations and scenarios described in Section 4.4. Each scenario was repeated 10 times for every combination of server and tunnel CCA. Results are presented as the download completion time (DLCT) when downloading a 300 MB file, relative to the DLCT without a tunnel. Recall that the optimal DLCT is around 50% when using the multipath framework as this would allow to double the available capacity. A relative DLCT of more than 100% indicates that the DLCT increases when using multipath compared to only single path, which clearly should be avoided.

Figure 10 shows the average DLCT over a multipath tunnel relative to the average DLCT without a tunnel for a set of 15 scenarios, repeated 10 times for each CCA combination. In the figure, *SHORT* and *LONG* refer to the RTT; *BUF* denotes the bottleneck MAC layer buffer size, in packets, in both paths; the prefix *ASYM* refers to the difference in path RTT; *NEAR* and *DIST* refer to the proxy being near to, or far from the user; *CDL*0 and *CDL*1 refers to whether CoDel was used at the bottleneck or not. The *bright* and *dark* highlights help to indicate the RTT; each result is paired with the result of the opposite scenario for easy comparison. The results in Figure 10a,b are a subset of those in Figure 10c,f.

Figure 10a,b arrange these results by scenario for a subset of all scenarios, while Figure 10c,f arrange the same results by CCA combination. Here, we can see how some configurations particularly impact some CCA combinations.

BBR over BBR/CCID5: The four top bars in Figure 10c demonstrate that when tunneling BBR over BBR the performance improves when the proxy is deployed near the user, as explained in Section 4.1. Figure 10c shows that a small bottleneck buffer or presence of CoDel also inhibits aggregation, likely due to BBR's limited reaction to loss. A similar pattern is observed when tunneling BBR over CCID5, albeit with a more favorable outcome for CCID5. Our conclusion is that CCID5 performs better, in part because it bases its congestion window (CWND) on packets rather than bytes. While it is plausible that the ability of DCCP and CCID5 to avoid single-path head-of-line blocking contributed to this performance disparity, this explanation remains speculative and requires further validation.

Cubic over BBR/CCID5: The six top bars in Figure 10d demonstrate that a short tunnel and sufficiently large bottleneck buffer are beneficial when tunneling Cubic over BBR/CCID5. As explained in Section 4.2, these scenarios experience fewer packet losses. CoDel is another source of packet losses, additionally deteriorating the performance. A similar pattern is seen when tunneling BBR over CCID5, but the performance is worse for CCID5. The visibly longer minimum-RTT probing periods of CCID5 have an adverse

impact on the more sensitive Cubic congestion control algorithm. In addition, Section 4.2 already illustrated that both using BBRv2 instead of BBR and tuning the tunnel receiver to wait longer for re-transmitted packets could improve the performance significantly.

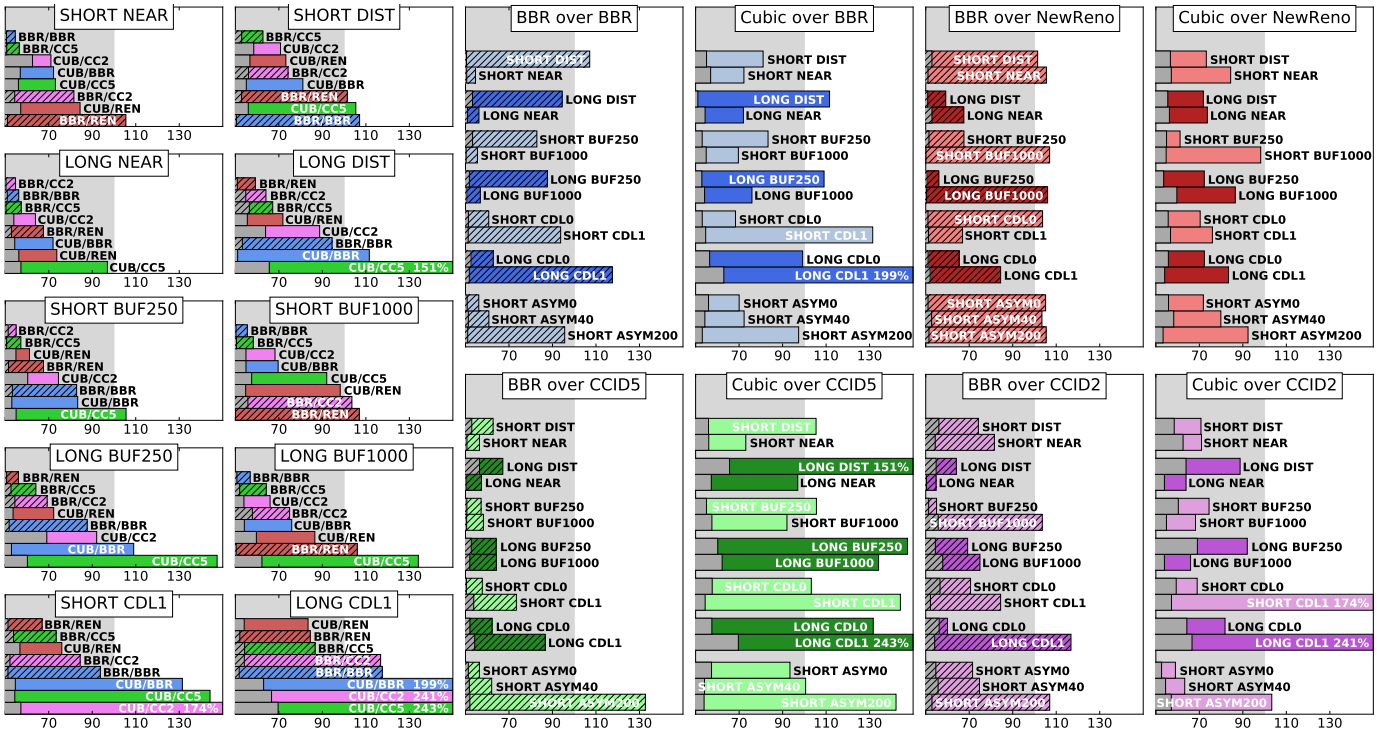

Figure 10. Average download completion times (DLCTs) over a multipath tunnel relative to the average DLCT without a tunnel

BBR over NewReno/CCID2: In Figure 10e, the top eight bars demonstrate that when tunneling BBR over NewReno/CCID2, a long RTT or a small bottleneck buffer results in very good performance. This was previously analyzed in Section 4.3. With congestion being one of the fundamental challenges, CoDel at the bottleneck may also improve aggregation, although not when the RTT is long. While a long end-to-end RTT improves the performance, this is not something that is in itself desirable or feasible. Our conclusion, given these results, is that NewReno and CCID2 are bad candidates for a tunneling CCA.

Cubic over NewReno/CCID2: This combination was not dealt with in previous sections. Part of the reason we omitted a deeper evaluation of this combination was that we were unable to pose a good single hypothesis as to which factors determined the performance. As when tunneling BBR over NewReno/CCID2, we conclude that NewReno and CCID2 are in themselves bad candidates for a tunneling CCA. As packet loss is part of the congestion control cycle of NewReno/CCID2, recovering from these losses within the tunnel should improve the performance when tunneling Cubic over NewReno/CCID2. This is only possible using NewReno, as CCID2 and DCCP are unreliable. Although these results were omitted for lack of space, when tuning the receiver wait limit such that re-transmissions arrived before a packet was forcibly released, we found that performance improved in almost all scenarios, and most of all in those scenarios where CoDel was used. We see in Figure 10f that CoDel is very detrimental to the DLCT when CCID2 is used, but much less so when NewReno is used. This is because the packet loss that CoDel generates is partially hidden from the E2E Cubic sender. We also see in Figure 10f that a big buffer is very detrimental to performance for NewReno, but preferable for CCID2. A smaller buffer makes packet loss more frequent, so this seeming contradiction makes sense, i.e., when using CCID2, we prefer scenarios where packet loss is infrequent.

Summary: Finally, Table 3 offers a comprehensive overview of the average DLCT across various scenarios, highlighting each combination of congestion control algorithm (CCA) and tunnel protocol, relative to the baseline scenario when using a single path without using a tunnel. This presentation allows us to make some determination as to which tunnel CCA is actually preferable. The results are sorted by DLCT for CPF, with lower values indicating better performance. This order is maintained for the SRTT results. It is worth noting that an optimal DLCT hovers around 50%, assuming that paths have identical capacity. A DLCT above 100% indicates worse performance than using a single path. Notably, BBRv2 emerges as the top-performing tunnel CCA, irrespective of whether BBR or Cubic is employed end to end. Alternatively, if NewReno/CCID2 is the preferred CCA, reasonably good performance is achievable by pairing it with the static reordering module alongside NewReno—excluding CCID2—and using the reliability inherent in TCP. Figure 10b–f show that deploying the proxy near the user is only beneficial when the tunnel CCA is BBR/CCID5, regardless of which end-to-end CCA is in use. The same is true for a large bottleneck buffer. Conversely, a small bottleneck buffer is seen to be preferable only if the tunnel CCA is NewReno/CCID2, regardless of the end-to-end CCA. The exception to this is tunneling Cubic over CCID2, in which case, a large buffer is preferable. In all these cases, the preferred configuration is that which leads to the least amount of congestion and packet loss. With that in mind, it becomes clear why tunneling Cubic over CCID2 shows performance problems, as a small buffer leads to less congestion but also more frequent loss. In contexts like ATSSS, where deploying a transport-layer multipath tunnel is envisioned, adjusting the proxy deployment might be more feasible than modifying the bottleneck buffer size, thereby favoring the adoption of BBR/BBRv2/CCID5 as the tunnel CCA.

Table 3. Average DLCT relative to no tunnel.

CCA Combination	DLCT for CPF	DLCT for SRTT
Cubic over CCID5	124.3%	101.6%
Cubic over BBR	94.2%	75.7%
Cubic over CCID2	92.4%	90.6%
BBR over NewReno	87.0%	65.4%
BBR over CCID2	77.5%	66.1%
Cubic over NewReno	77.8%	67.5%
BBR over BBR	76.2%	65.8%
Cubic over NewReno (static)	70.2%	60.0%
BBR over CCID5	67.9%	66.4%
Cubic over BBRv2 (static)	65.6%	61.1%
BBR over BBRv2	59.9%	58.9%

Table 3 also includes results obtained using the SRTT scheduler. When comparing to CPF, SRTT performed better in all scenarios. One reason for this is that SRTT often switches to the secondary path earlier than CPF, namely, at the point at which the congestion on the primary path has caused the RTT of the two paths to equalize. As mentioned earlier, SRTT is, however, often less preferable than CPF in that it makes more heavy use of the expensive path.

6. Conclusions and Future Work

In this paper, we examined the viability of employing transport-layer multipath tunnels alongside the CPF scheduler. The driving scenario behind this investigation was the aspiration to efficiently utilize both cost-effective and expensive wireless access, such as wireless LAN and cellular networks. Our analysis showed various performance challenges associated with such a scenario. These challenges existed for both of the server CCAs assessed in this study and can be expected to exist for other algorithms as well. Specifically, when the server used BBR, aggregation occurred more or less perfectly or not at all. When the server used Cubic, the performance tended to improve gradually over time, with aggregation occurring if the transfer time or file size was sufficiently large.

A helpful conceptual framework for understanding these issues is to consider the distribution of the server's congestion window over multiple paths. A larger server CWND or a smaller CWND over the primary path both increase the likelihood that the server CWND is distributed over the secondary path. Therefore, it is advantageous to inflate the server CWND whenever possible and manage congestion on the primary path. Strategies that may help in this regard include: (i) adjusting the deployment of the proxy relative to the user; (ii) adjusting the bottleneck buffer size; (iii) deploying an AQM at the bottleneck buffer; (iv) changing the CCA used at the tunnel; (v) using a reliable tunnel protocol to allow for re-transmissions of packets within the tunnel; (vi) changing CPF such that it becomes aware of congestion and considers congestion states when making packet scheduling decisions over the different paths.

Related to (v), a potential solution could involve implementing semi-reliability at the scheduler level, which is a planned area of future work. Finally, with respect to (vi) efforts have already begun to adjust CPF [49,50]. Specifically, work has been conducted using BBR in the tunnel to enable CPF to leverage the bandwidth-delay product knowledge, which in turn allows for a more intelligent scheduling to limit congestion. Future work on this enhanced CPF scheduler will include enhancing its ability to handle dynamic paths with fluctuating capacity.

Author Contributions: Conceptualization, M.P., A.K., A.B., V.R. and M.A.; Methodology, M.P., A.K., A.B. and M.A.; Software, M.P.; Validation, M.P.; Formal analysis, V.R.; Investigation, V.R.; Resources, A.K. and M.A.; Writing—original draft, M.P.; Writing—review & editing, M.P., A.K., A.B., V.R. and M.A.; Visualization, M.P.; Supervision, A.K. and A.B.; Project administration, A.K. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: The work has been partly funded by the Knowledge Foundation of Sweden grant 20220072-H-01, and the Bavarian State Ministry for Science and Art through the Hightech Agenda (HTA).

Data Availability Statement: No data is being made available for the work presented in the paper.

Conflicts of Interest: Markus Amend is an employee of Deutsche Telekom (Germany). The authors declare no conflicts of interest.

References

1. Ericsson. *Mobility Report November 2023*; *Ericsson White Paper*; 2023. Available online: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2023 (accessed on 5 May 2024).
2. Ford, A.; Raiciu, C.; Handley, M.; Bonaventure, O. *TCP Extensions for Multipath Operation with Multiple Addresses*; Document RFC 6824; 2013. Available online: https://www.rfc-editor.org/rfc/rfc6824.html (accessed on 5 May 2024).
3. Coninck, Q.D.; Bonaventure, O. Multipath QUIC: Design and Evaluation. In Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, Incheon, Republic of Korea, 12–15 December 2017.
4. Amend, M.; Brunstrom, A.; Kassler, A.; Rakocevic, V.; Johnson, S. DCCP Extensions for Multipath Operation with Multiple Addresses, IETF Draft. May 2024. Available online: https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-multipath-dccp-16 (accessed on 5 May 2024).
5. Iyengar, J.R.; Amer, P.D.; Stewart, R. Multipath Transfer Using SCTP Multihoming over Independent end-to-end Paths. *IEEE/ACM Trans. Netw.* **2006**, *14*, 951–964. [CrossRef]
6. Bonaventure, O.; Piraux, M.; Coninck, Q.D.; Baerts, M.; Amend, C.; Amend, M. *Multipath Schedulers*; Working Draft; 2020. Available online: https://www.ietf.org/archive/id/draft-bonaventure-iccrg-schedulers-02.html (accessed on 5 May 2024).
7. Amend, M.; Bogenfeld, E.; Cvjetkovic, M.; Rakocevic, V.; Pieska, M.; Kassler, A.; Brunstrom, A. A Framework for Multiaccess Support for Unreliable Internet Traffic using Multipath DCCP. In Proceedings of the 2019 IEEE 44th Conference on Local Computer Networks (LCN), Osnabrueck, Germany, 14–17 October 2019; pp. 316–323.
8. Floyd, S.; Handley, M.J.; Kohler, E. Datagram Congestion Control Protocol (DCCP), IETF RFC 4340, March 2006. Available online: https://www.rfc-editor.org/rfc/rfc4340.html (accessed on 5 May 2024).
9. Yang, F.; Wang, Q.; Amer, P. Out-of-order transmission for in-order arrival scheduling policy for multipath TCP. In Proceedings of the 2014 28th International Conference on Advanced Information Networking and Applications Workshops, Victoria, BC, Canada, 13–16 May 2014.
10. Evensen, K.; Kaspar, D.; Engelstad, P.; Hansen, A.F.; Griwodz, C.; Halvorsen, P. A network-layer proxy for bandwidth aggregation and reduction of IP packet reordering. In Proceedings of the 2009 IEEE 34th Conference on Local Computer Networks, Zurich, Switzerland, 20–23 October 2009; pp. 585–592.

11. Honda, O.; Ohsaki, H.; Imase, M.; Ishizuka, M.; Murayama, J. Understanding TCP over TCP: Effects of TCP tunneling on end-to-end throughput and latency. In *Performance, Quality of Service, and Control of Next-Generation Communication and Sensor Networks III*; International Society for Optics and Photonics: Bellingham, WA, USA, 2005; Volume 6011, pp. 138–146.
12. Coonjah, I.; Catherine, P.C.; Soyjaudah, K.M.S. Experimental performance comparison between TCP vs UDP tunnel using OpenVPN. In Proceedings of the 2015 International Conference on Computing, Communication and Security (ICCCS), Pointe aux Piments, Mauritius, 4–5 December 2015; pp. 1–5.
13. Lee, B.P.; Balan, R.K.; Jacob, L.; Seah, W.K.G.; Ananda, A.L. Avoiding congestion collapse on the Internet using TCP tunnels. *Comput. Netw.* **2002**, *39*, 207–219. [CrossRef]
14. Han, H.; Shakkottai, S.; Hollot, C.V.; Srikant, R.; Towsley, D. Multi-path TCP: A joint congestion control and routing scheme to exploit path diversity in the Internet. *IEEE/ACM Trans. Netw.* **2006**, *14*, 1260–1271. [CrossRef]
15. Radunovic, B.; Gkantsidis, C.; Gunawardena, D.; Key, P. Horizon: Balancing TCP over multiple paths in wireless mesh network. In Proceedings of the 14th ACM International Conference on Mobile Computing and Networking, San Francisco, CA, USA, 14–19 September 2008, pp. 247–258.
16. Wischik, D.; Raiciu, C.; Greenhalgh, A.; Handley, M. Design, implementation and evaluation of congestion control for multipath TCP. In Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation, Boston, MA, USA, 30 March–1 April 2011.
17. Raiciu, C.; Handly, M.; Wischik, D. *Coupled Congestion Control for Multipath Transport Protocols*; Document RFC 6356; 2011. Available online: https://www.rfc-editor.org/rfc/rfc6356.html (accessed on 5 May 2024)
18. Khalili, R.; Gast, N.; Popovic, M.; Boudec, J.-Y.L. MPTCP is not Pareto-optimal: Performance issues and a possible solution *IEEE/ACM Trans. Netw.* **2013**, *21*, 1651–1665.
19. Noda, K.; Ito, Y.; Muraki, Y. Study on congestion control of multipath TCP based on web-QoE under heterogeneous environment. In Proceedings of the 6th IEEE Global Conference on Consumer Electronics (GCCE), Nagoya, Japan, 24–27 October 2017; pp. 1–3.
20. Xu, Z.; Tang, J.; Yin, C.; Wang, Y.; Xue, G. Experience-Driven Congestion Control: When Multi-Path TCP Meets Deep Reinforcement Learning. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1325–1336. [CrossRef]
21. Pokhrel, S.R.; Walid, A. Learning to Harness Bandwidth with Multipath Congestion Control and Scheduling. *IEEE Trans. Mob. Comput.* **2021**, *22*, 996–1009. [CrossRef]
22. Li, M.; Lukyanenko, A.; Tarkoma, S.; Cui, Y.; Ylä-Jääski, A. Tolerating path heterogeneity in Multipath TCP with bounded receive buffers. *ACM SIGMETRICS Perform. Eval. Rev.* **2013**, *41*, 375–376. [CrossRef]
23. Cui, Y.; Wang, L.; Wang, X.; Wang, H.; Wang, Y. FMTCP: A fountain code-based multipath transmission control protocol. *IEEE/ACM Trans. Netw.* **2015**, *23*, 465–478. [CrossRef]
24. Wu, J.; Yuen, C.; Cheng, B.; Wang, M.; Chen, J. Streaming High-Qual. Mob. Video Multipath TCP Heterog. *Wirel. Netw. IEEE Trans. Mob. Comput.* **2016**, *15*, 2345–2361. [CrossRef]
25. Carlucci, G.; Cicco, L.D.; Holmer, S.; Mascolo, S. Analysis and design of the google congestion control for Web real-time communication (WebRTC). In Proceedings of the 7th International Conference on Multimedia Systems, MMSys 2016, Klagenfurt, Austria, 10–13 May 2016; p. 13.
26. Johansson, I.; Sarker, Z. Self-Clocked Rate Adaptation for Multimedia, IETF RFC 8298, Internet Requests for Comments, December 2017. Available online: https://www.rfc-editor.org/rfc/rfc8298.txt (accessed on 5 May 2024).
27. Leymann, N.; Heidemann, C.; Zhang, M.; Cullen, M. Huawei's GRE Tunnel Bonding Protocol, IETF RFC 8157, May 2017. Available online: https://www.rfc-editor.org/rfc/rfc8157 (accessed on 5 May 2024).
28. Bednarek, M.; Kobas, G.B.; Kühlewind, M.; Trammell, B. Multipath bonding at Layer 3. In Proceedings of the 2016 Applied Networking Research Workshop, Berlin, Germany, 16 July 2016; pp. 7–12.
29. Detal, G.; Paasch, C.; Bonaventure, O. Multipath in the middle (box). In Proceedings of the 2013 Workshop on Hot Topics in Middleboxes and Network Function Virtualization, Santa Barbara, CA, USA, 9 December 2013; pp. 1–6.
30. Hampel, G.; Rana, A.; Klein, T. Seamless TCP mobility using lightweight MPTCP proxy. In Proceedings of the 2013 11th ACM International Symposium on Mobility Management and Wireless Access, Barcelona, Spain, 3–8 November 2013; pp. 139–146.
31. Bonaventure, O.; Boucadair, M.; Gundavelli, S.; Seo, S.; Hesmans, B. 0-RTT TCP Convert Protocol, IETF RFC 8803, July 2020. Available online: https://www.rfc-editor.org/rfc/rfc8803.html (accessed on 5 May 2024).
32. Kim, D.; Moon, Y.; Hwang, J.; Park, K. FlexCP: A Scalable Multipath TCP Proxy for Cellular Networks. In Proceedings of the ACM on Networking, Paris, France, 5–8 December 2023.
33. Ha, N.V.; Kumazoe, K.; Tsukamoto, K.; Tsuru, M. Masking lossy networks by TCP tunnel with network coding. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017; pp. 1290–1295.
34. Schinazi, D. *The MASQUE Protocol*; Working Draft; 2024. Available online: https://datatracker.ietf.org/doc/html/draft-schinazi-masque-proxy-02 (accessed on 5 May 2024).
35. Kühlewind, M.; Sarker, Z.; Fossati, T.; Pardue, L. *Use Cases and Requirements for QUIC as a Substrate*; Working Draft; 2020. Available online: https://www.ietf.org/archive/id/draft-kuehlewind-masque-quic-substrate-00.html (accessed on 5 May 2024).
36. Krämer, Z.; Kühlewind, M.; Ihlar, M.; Mihály, A. Cooperative performance enhancement using QUIC tunneling in 5G cellular networks. In Proceedings of the Applied Networking Research Workshop, Virtual, 24–30 July 2021; pp. 49–51.
37. Ha, S.; Rhee, I.; Xu, L. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Oper. Syst. Rev.* **2008**, *42*, 64–74. [CrossRef]

38. Cardwell, N.; Yuchung, C.; Gunn, S.C.; Yeganeh, S.H.; Jacobson, V. BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* **2016**, *15*, 20–53. [CrossRef]
39. Paasch, C.; Ferlin, S.; Alay, O.; Bonaventure, O. Experimental evaluation of multipath TCP schedulers. In Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop, Chicago, IL, USA, 18 August 2014; pp. 27–32.
40. Raiciu, C.; Paasch, C.; Barre, S.; Ford, A.; Honda, M.; Duchene, F.; Bonaventure, O.; Handley, M. How hard can it be? designing and implementing a deployable multipath TCP. In Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 25–27 April 2012.
41. Ferlin, S.; Alay, O.; Mehani, O.; Boreli, R. BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks. In Proceedings of the 2016 IFIP Networking Conference (IFIP Networking) and Workshops, Vienna, Austria, 17–19 May 2016.
42. Xing, Y.; Xue, K.; Zhang, Y.; Han, J.; Li, J.; Wei, D.S.; Li, R.; Sun, Q.; Lu, J. A stream-aware mpquic scheduler for HTTP traffic in mobile networks. *IEEE Trans. Wirel. Commun.* **2022**, *22*, 2775–2788.
43. Lim, Y.-S.; Nahum, E.M.; Towsley, D.; Gibbens, R.J. ECF: An MPTCP path scheduler to manage heterogeneous paths. In Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies, Incheon, Republic of Korea, 12–15 December 2017.
44. Wu, H.; Alay, O.; Brunstrom, A.; Ferlin, S.; Caso, G. Peekaboo: Learning-Based Multipath Scheduling for Dynamic Heterogeneous Environments. *IEEE Sel. Areas Commun.* **2020**, *38*, 2295–2310. [CrossRef]
45. Guo, Y.E.; Nikravesh, A.; Mao, Z.M.; Qian, F.; Sen, S. Accelerating Multipath Transport Through Balanced Subflow Completion. In Proceedings of the 23nd Annual International Conference on Mobile Computing and Networking, Snowbird, UT, USA, 16–20 October 2017.
46. Nguyen, T.T.; Vu, M.H.; Nguyen, P.L.; Do, P.T.; Nguyen, K. A q-learning-based multipath scheduler for data transmission optimization in heterogeneous wireless networks. In Proceedings of the 2023 Proceedings of IEEE 20th Consumer Communications Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2023; pp. 573–578.
47. Floyd, S.; Kohler, E. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control, IETF RFC 4341, March 2006. Available online: https://datatracker.ietf.org/doc/rfc4341/ (accessed on 5 May 2024).
48. Moreno, N.R.; Amend, M.; Brunstrom, A.; Kassler, A.; Rakocevic, V. CCID5: An implementation of the BBR Congestion Control algorithm for DCCP and its impact over multi-path scenarios. In Proceedings of the Applied Networking Research Workshop, Virtual, 24–30 July 2021; pp. 52–58.
49. Pieska, M.; Rabitsch, A.; Brunstrom, A.; Kassler, A.; Amend, M.; Bogenfeld, E. Low-delay cost-aware multipath scheduling over dynamic links for access traffic steering, switching, and splitting. *Comput. Netw.* **2024**, *241*, 110186. [CrossRef]
50. Pieska, M.; Rabitsch, A.; Brunstrom, A.; Kassler, A.; Amend, M. Adaptive cheapest path first scheduling in a transport-layer multi-path tunnel context. In Proceedings of the Applied Networking Research Workshop, Virtual, 24–30 July 2021; pp. 39–45.