POLYCORN: Data-driven Cross-layer Multipath Networking for High-speed Railway through Composable Schedulerlets

Yunzhe Ni, *Peking University;* Feng Qian, *University of Minnesota – Twin Cities;* Taide Liu, Yihua Cheng, Zhiyao Ma, and Jing Wang, *Peking University;* Zhongfeng Wang, *China Railway Gecent Technology Co., Ltd;* Gang Huang and Xuanzhe Liu, *Key Laboratory of High Confidence Software Technologies, Ministry of Education, Peking University;* Chenren Xu, *Zhongguancun Laboratory and Key Laboratory of High Confidence Software Technologies, Ministry of Education, Peking University*

https://www.usenix.org/conference/nsdi23/presentation/ni

This paper is included in the Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation.

April 17–19, 2023 • Boston, MA, USA

978-1-939133-33-5

Open access to the Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation is sponsored by

POLYCORN: Data-driven Cross-layer Multipath Networking for High-speed Railway through Composable Schedulerlets

Yunzhe Ni[ℙ], *Feng Qian*[𝕄], *Taide Liu*[ℙ], *Yihua Cheng*[ℙ], *Zhiyao Ma*[ℙ], *Jing Wang*[ℙ]
Zhongfeng Wang[𝔾], *Gang Huang*[ℙℍ], *Xuanzhe Liu*[ℙℍ], *Chenren Xu*[ℙℤℍ✉*]
[ℙ]*Peking University* [𝕄]*University of Minnesota – Twin Cities* [𝔾]*China Railway Gecent Technology Co., Ltd*
[ℤ]*Zhongguancun Laboratory* [ℍ]*Key Laboratory of High Confidence Software Technologies, Ministry of Education (PKU)*

Abstract – Modern high-speed railway (HSR) systems offer a speed of more than 250 km/h, making on-board Internet access through track-side cellular base stations extremely challenging. We conduct extensive measurements on commercial HSR trains, and collect a massive 1.79 TB GPS-labeled TCP-LTE dataset covering a total travel distance of 28,800 km. Leveraging the new insights from the measurement, we design, implement, and evaluate POLYCORN, a first-of-its-kind networking system that can significantly boost Internet performance for HSR passengers. The core design of POLYCORN consists of a suite of composable multipath schedulerlets that intelligently determine what, when, and how to schedule user traffic over multiple highly fluctuating cellular links between HSR and track-side base stations. POLYCORN is specially designed for HSR environments through a cross-layer and data-driven proactive approach. We deploy POLYCORN on the operational LTE gateway of the popular Beijing-Shanghai HSR route at 300 km/h. Real-world experiments demonstrate that POLYCORN outperforms the state-of-the-art multipath schedulers by up to 242% in goodput, and reduces the delivery time by 45% for instant messaging applications.

1 Introduction

High-speed railway (HSR) systems, which offer a speed of 250+ km/h, revolutionize inter-city travel. Internet services on HSR are typically provided by track-side cellular base stations and an on-board proxy relaying data between WiFi APs and the cellular infrastructure [1–7]. However, the ultra-fast speed of HSR poses unprecedented challenges in bringing seamless Internet service to passengers because of the intermittent link connectivity characteristic – handover happens every less than 10 seconds [8] and the handover failure may cause a "blackout" period of up to 10 seconds [9], as reported by previous studies.

It is known that MPTCP [10] (or multipath transport in general) can leverage path diversity (with each path associated with a cellular carrier or mobile network operator) to improve link/connection robustness, as demonstrated in low mobility scenarios [11–14]. However, applying multipath transport to HSR networking is very challenging. Under such extreme mobility, the network performance fluctuates subsecond level [15], leading to 1636x higher variance in RTT than in static or low mobility scenarios [16] – this is a sharp contrast to the common assumption that a relatively stable network condition may last for several RTTs, which is leveraged by the state-of-the-art MPTCP schedulers [17, 18] for making scheduling decisions. Indeed, as to be demonstrated in §6.2, the inaccurate link quality estimation and scheduling decision informed by inaccurate throughput and RTT observations often lead to poor performance under extreme mobility.

In this paper, we argue that in contrast to the state-of-art multipath schedulers that rely on instantaneous performance measurement for making reactive scheduling decisions, we should carefully mine the networking features specific to the extreme mobility scenario, identify the main events leading to predictable failures, and design proactive scheduling strategies accordingly. For this purpose, we conduct real-world measurements on the popular Beijing-Shanghai route in China traveling at an average speed of 300 km/h. Over 24 trips spanning three weeks, we collected 1.79 TB data covering a total travel distance of 28,800 km. Our study differs from all prior HSR networking measurement studies [8, 15, 19, 20]: through collaborating with the on-board HSR ISP, we obtain precise location (from the GPS receiver mounted on the carriage roof) for every collected network performance sample. This allows us to statistically correlate the train's physical location with various network performance events, enabling many key analyses and henceforth the design of our system.

Leveraging our unique dataset, we identify three key aspects that guides our system design. First, handover failures, which incur several seconds of link disconnection, can be reasonably predicted from the train's moving trajectory. Second, the multipath heterogeneity (the relative performance ranking across paths, *i.e.,* carriers) is highly dynamic, oftentimes changing on a per-RTT basis. Third, transport-layer packet retransmission is much more common than typical cellular links. We find that on HSR, 1.8% of the TCP packets experience retransmission timeout (RTO) – among them, 24% are retransmitted more than once.

Inspired by the above findings, we develop POLYCORN, a practical networking system that significantly boosts the Internet performance for HSR passengers. It is to our best knowledge the first full-fledged system that specifically optimizes for Internet services on extreme mobility ground trans-

*✉: chenren@pku.edu.cn

portation. Leveraging multipath heterogeneity, POLYCORN distributes passengers' traffic over multiple cellular carriers for bandwidth aggregation, reduced delay, and improved reliability. In its core, POLYCORN is equipped with a novel multipath scheduler called HSRSCH, which judiciously determines transmitting which data over which path(s) in real time despite the highly dynamic network conditions. HSRSCH is tailored to HSR environments through a proactive and cross-layer approach. Its design consists of four optimizations that could be either individually or jointly applied to existing "baseline" schedulers such as the minRTT shipped with MPTCP.

• **Handover-failure-aware Path Rejection** (§4.3) has an intuitive idea: once an imminent *handover failure event* is predicted, HSRSCH temporarily disables the corresponding path so packets will not be scheduled over it to avoid the blackout period (and inter-subflow out-of-order delay). To realize this idea, we apply robust and lightweight machine learning to predict handover failures. We use two features carefully derived from our measurements: location and cell ID, which lead to an overall prediction accuracy of 80.6%.

• **Tail-aware Path Rejection** (§4.4) determines whether to use path(s) when their link conditions deteriorate. Its basic idea is to avoid scheduling tail packets, which belong to the end of a flow (*i.e., upon an end-of-flow indicator*) on slow path(s). Since HSR traffic is dominated by short-lived flows (*e.g.,* web browsing, instant messaging, mini videos), this optimization can significantly accelerate short flows' completion time. We instantiate the idea by modeling the queuing and transmission process of tail packets to guide path selection.

• **Extended Reinjection** (§4.5) detects vulnerable packets when losses occur, and retransmit them early over other paths in a batched manner. The idea stems from the bursty nature of wireless losses and consequent *excessive RTO events*, which we find to be even more prominent on HSR: one single packet loss will introduce more subsequent ones. Our approach differs from MPTCP's default packet-by-packet reinjection mechanism that is far too conservative for HSR. We carefully determine the reinjection aggressiveness based on real data to avoid putting too much burden on other paths.

• **Opportunistic Redundant Traffic Injection** (§4.6) proactively leverages idle path(s) to transmit redundant data. It not only provides extra resiliency to link quality fluctuation, but also enables the transport layer to continuously probe the path for important metrics such as RTT and RTO, which are highly dynamic when probed from HSR.

We integrate the above components through a composable scheduling framework, which treats a complex multipath scheduler as a pipeline of modularized *schedulerlets* as described above. Compared to a monolithic scheduler, our schedulerlet-based approach decouples the multipath scheduling logic, and thus significantly reduces the system complexity and development overhead, through the unified interface of schedulerlets designed by us. It also makes the system extensible and open to future optimizations (§4.2). The scheduling framework is then integrated with a multi-user/multi-path data transport mechanism (also developed by us), leading to the full-fledged POLYCORN system (§4.7).

Utilizing various system-level techniques including multipipe multiplexing [21, 22] and user-level packet interception, our implementation runs completely in the user space while maintaining full user/server transparency and high packet I/O performance. It is deployed as two proxy modules, one running on the HSR train and the other running on a cloud server, that schedule uplink and downlink traffic respectively over multiple cellular paths. POLYCORN requires no hardware or firmware modifications, and is orthogonal to HSR-specific PHY/MAC layer innovations [23–26] for cellular networks. Our implementation consists of 24K lines of code.

Through our three-year collaboration efforts with the HSR ISP's operational department, we managed to deploy POLYCORN on real HSR trains by instrumenting their onboard LTE gateways. We evaluate POLYCORN on the popular Beijing-Shanghai route at 300 km/h, with the key results as follows.

• On HSR, POLYCORN outperforms state-of-art multipath schedulers (*e.g.,* ECF [27], STMS [18], MuSher [17], BLEST [28] and MPTCP' default scheduler [10]) by 43% to 242% when downloading files with different sizes.

• POLYCORN consistently outperforms MPTCP by 61.5%, 30.6%, 64.2% on the three HSR route segments (Beijing-Jinan, 406 km; Jinan-Nanjing, 617 km; Nanjing-Shanghai, 301 km) respectively, in terms of the file download time. This indicates that POLYCORN could boost the networking performance under different HSR track-side environments.

• POLYCORN reduces the delivery time by 45% for an instant messaging application in a multi-user setting, compared to the current operational deployment of HSR Internet access.

Note that although the above results are obtained from LTE, POLYCORN is compatible with 5G networks that are being deployed along the HSR tracks [29]. We elaborate the applicability of POLYCORN on 5G in §7.

The Contributions of this paper is summarized as follows.

• New insights of extreme mobility networking characteristics derived from a massive, GPS-labeled TCP-LTE dataset covering 28,800 km travel distance.

• The design of cross-layer, proactive multipath scheduling algorithms tailored to extreme mobility networking, and their integration through a composable scheduling pipeline.

• The development of POLYCORN, the first-of-its-kind network system boosting the mobile Internet performance for all the HSR passengers.

• Deployment and extensive evaluations of POLYCORN on commercial HSR trains in the wild.

This work does not raise any ethical issues.

2 Networking Performance Measurement

To motivate the design of POLYCORN, we conduct real-world measurements of mobile networking performance on HSR. Our study is unique in two aspects. First, all our measured network performance samples have precise GPS coordinates, as opposed to coarse-grained cell IDs used in previous studies [8, 15]. Second, leveraging the unique GPS-labeled data, we offer new insights such as the predictability of failed LTE handovers on HSR.

2.1 Data Collection Methodology

A major challenge of collecting fine-grained location on HSR is a lack of GPS signal in the carriage due to electromagnetic shielding. To overcome it, we collaborate with the China Railway Gecent Technology operating China's HSR WiFi platform. We next describe our data collection setup in detail.

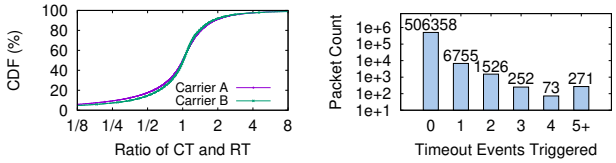
Onboard LTE Gateway. It is deployed by China's HSR WiFi carrier in the server room on each train. This gateway serves two purposes: in the upstream, it connects to track-side LTE base stations for Internet access; in the downstream, it connects to the WiFi access points (802.11ac APs) serving passengers in each carriage through a wired local area network (LAN). The gateway is equipped with multiple SIM cards of two major cellular carriers, as well as a 2×2 MIMO antenna mounted on top of the server room carriage. The GPS receiver is also mounted on the carriage roof, allowing precisely tracking the location and speed of the train. We are permitted to access the GPS data and use two of the LTE interfaces exclusively (*i.e.,* there is no other user traffic over these interfaces during data collection). We conduct data collection using iPerf [30], tshark [31], and Quectel LTE QLog[1] to measure the available bandwidth, capture packet traces, and record LTE control-plane messages, respectively. No prior study to our knowledge has leveraged such a unique infrastructure for HSR network measurement and optimization.

Measurement Server. We deploy two co-located servers (4× Intel Xeon Skylake 6133 2.5 GHz CPU with 8 GB RAM) in a major cloud service provider in China. Each server serves measurement requests for one LTE carrier. The servers are located in Shanghai that is 20 to 1,300 km away from our studied HSR route. We conduct wired experiments from several hosts near the HSR route to the two servers, and the throughput (RTT) are measured to be ≥50 Mbps (≤ 33 ms), which is far above (below) the corresponding metric measured on HSR. This indicates that the Internet is not the performance bottleneck for the end-to-end (*i.e.,* HSR-to-server) path.

Route and Duration. We carried out experiments on the "Fuxing" high-speed trains between Beijing and Shanghai, the top-two cities in China. This 1,318 km route is one of the busiest railway routes in China, with an annual passenger volume of 215 million. The average train speed is 300 km/h. Over 24 trips spanning three weeks, we collected 1.79 TB data covering a total travel distance of 28,800 km. Our dataset consists of the following cross-layer records: (1) The *GPS location* of the train updated every second; (2) The *packet traces and downlink TCP Throughput* collected by a long-lived iPerf session running on the LTE gateway; the server uses the BBR congestion control [32] that is known to yield a more accurate bandwidth estimation compared to widely deployed TCP CUBIC [33]; (3) The *LTE lower-layer information* including cell ID, signal strength (RSRP), and the LTE signaling messages collected by Quectel LTE QLog and parsed by MobileInsight [34]. We made the dataset publicly available in: https://soar.group/projects/hsrnet/dataset.html.

2.2 Throughput & Latency Characterization

Leveraging our large dataset, we begin with basic characterizations of throughput and latency. Across the entire dataset, the 25th, 50th, and 75th-percentile downlink TCP throughput of Carrier A (B)[2] are measured to be 2.56 (4.60), 6.48 (10.67), and 12.42 (19.73) Mbps, respectively. Regarding the latency, the 25th, 50th, and 75th-percentile RTT of carrier A (B) are 123 (147), 185 (191), and 315 (348) ms, respectively. We observe that both throughput and latency exhibit high temporal fluctuations. To quantify them, we compute the ratio between the average throughput in the current time window $[t_0 - \Delta t, t_0]$, denoted as CT, and the average throughput in the previous window $[t_0 - 6\Delta t, t_0 - \Delta t]$, denoted as RT, where t_0 is the current time. Δt is empirically chosen as 0.2 seconds, the median RTT when HSR travels at 300 km/h. Fig. 1a plots the distribution of the throughput ratio defined above. As shown, in 26.4% (Carrier A) or 22.6% (Carrier B) of the cases, CT/RT is lower than 50% or higher than 200%, confirming the high throughput fluctuation. The latency variation is also prominent (figure not shown). This leads to frequent TCP Retransmission Timeout events (RTOs), which are experienced by 1.8% of the packets. We even observe that the transmissions of many packets experience more than one RTO, as shown in Fig. 1b ("0" indicates no RTO is triggered).



(a) Throughput. (b) RTT.

Figure 1: TCP performance temporal variation.

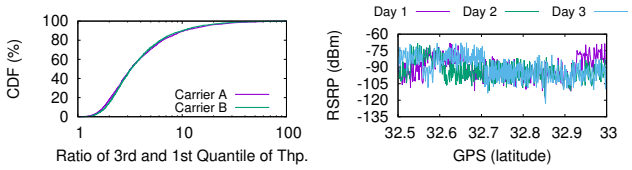
2.3 Predictability of Networking Performance

Since trains move along fixed rail tracks, it is anticipated that the networking performance is predictable, similar to what has been reported for lower-speed vehicles [35]. On HSR, however, the predictability may be affected by the ultra-high speed. No prior study to our knowledge has studied the predictability of HSR networking performance.

[1] A proprietary tool offered by the gateway vendor for collecting LTE log data from their LTE modems.

[2] China Mobile and China Unicom respectively.

We begin with exploring a straightforward approach of predicting the TCP throughput using the train's trajectory. For each <location, direction> pair, we compute the average throughput near the location ($\pm$ 1 km) in all trips to smooth out the temporal variation, aggregate all those samples and calculate the ratio between the 75-th and the 25-th percentile values. As shown in Fig. 2a, the median ratio is 3.21 and 3.24 for Carrier A and B, respectively, and it may reach up to 100[3]. The results indicate that, unlike low-speed transportation, throughput prediction in HSR is very challenging. The main reason is that the extreme mobility introduces complex stochastic channel fading, which can cause significant temporal variation in received signal strength and henceforth high data rate fluctuation at the same location. This is exemplified in Fig. 2b, which plots the RSRP values of Carrier A over a 40 km route measured on three different days. As shown, the link quality not only exhibits randomness across the three trips but also lacks spatial locality during the same trip. We also would to note that this unpredictable pattern is jointly determined by the highly dynamic channel condition and its complex interaction with TCP congestion control – a small difference in wireless channel condition may result in a big difference in future TCP performance. In addition, the prediction result also depends on a specific congestion control (CC) algorithm, which makes the design space for throughput prediction even more challenging.



(a) Variability of Throughput. (b) Location vs. RSRP.

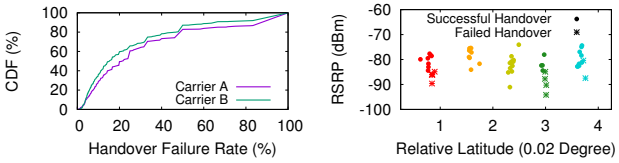
Figure 2: Measurement study of location-aware TCP throughput repeatability and predictability.

Given the difficulty of throughput prediction for HSR, POLYCORN takes a unique approach of predicting *handover failures*. Due to its high speed, HSR experiences much more frequent handovers compared to low-speed vehicles. More importantly, in HSR, handovers are more likely to fail. A *failed handover* occurs when a UE disconnects from or loses connection to the current base station but is not yet connected to the new base station. Failed handovers bring negative performance impact, including packet losses and their incurred retransmission timeout (RTO) that force TCP to enter a slow start. In our dataset, the performance impact is measured to be at least 1 second and can last as long as 10 seconds. In contrast, a successful handover usually incurs shorter than 100 ms of TCP throughput disruption.

We next explore the predictability of failed handovers given their importance. Our dataset records 32,231 and 45,656 handovers for Carrier A and B, respectively. Among them, the fraction of failed handovers is 6.22% and 5.47%, respectively, significantly higher than those experienced by low-speed vehicles. We observe that 47.33% (40.35%) of the source cells (from which the handover initiates) of Carrier A (Carrier B) experiences at least one handover failure in our three-week measurement. Fig. 3a plots the handover failure rate, defined as the ratio of failed handovers, across the cells[4] experienced at least one handover failure. As shown, for both carriers, more than 30% of the cells have a failure rate between 20% and 80%, indicating that it is infeasible to predict failed handovers only using cell ID. Nevertheless, we identify two noticeable features that can facilitate prediction of handover failures. First, the RSRP and failure rate are found to be negatively correlated, for handover messages are more likely to be lost under lower SNR. Second, a handover that is triggered late (compared to historically recorded handovers at the same location) is more likely to fail: due to HSR's high speed, there is simply not enough time for a late handover to complete. This is confirmed by the statistics shown in Tab. 1, which plots the successful rates across all handovers under four scenarios. (1) Handovers starting within 200 m of the Earliest Handover Positions (EHP) of their source cells. (2) Handovers starting at least 200 m beyond EHP; (3) Handovers with $\geq$-95dB RSRP when they start; (4) Handovers with $<$-95dB RSRP when they start. Here the EHP of a cell s is defined as the earliest geographic location (w.r.t. the train's moving direction) of all the handovers with a source cell s when they start. As shown in Tab. 1, handovers that start early or have high RSRP values have higher chances of success compared to late or low-RSRP handovers, respectively. We also plot some individual handovers versus signal strength and location in Fig. 3b where a dot (star) represents a successful (failed) handover, and the colors correspond to different cells.

Scenario	Carrier A	Carrier B
Distance to EHP < 200m	75.4%	83.7%
Distance to EHP $\geq$ 200m	65.9%	71.8%
RSRP $\geq$ -95dBm	89.2%	83.8%
RSRP < -95dBm	52.9%	67.7%

Table 1: Handover success rate.



(a) Failure rate. (b) Pattern snapshot.

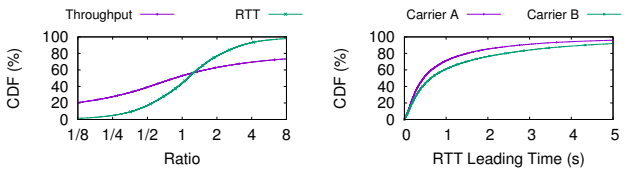
Figure 3: Handover pattern study.

[3]The ratio could be even higher with finer-grained location granularity due to the bursty TCP performance on HSR.

[4]Unless explicitly mentioned, *cell* means <cell, direction> pair.

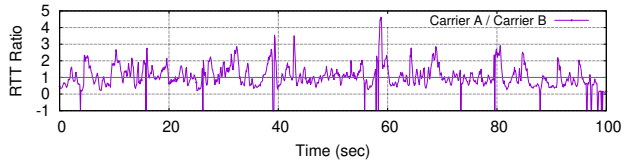
2.4 Multipath Heterogeneity

Recall that the on-board LTE gateway (§2.1) is equipped with SIM cards of two carriers. We next explore whether the performance of the two carriers are correlated or not. Specifically, we compute the throughput and RTT ratios between the two carriers in a synchronous manner. We find that 39.2% (16.9%) of the computed throughput (RTT) ratios are lower than 0.5, and 37.0% (23.2%) of the throughput (RTT) ratios are higher than 2.0, as plotted in Fig. 4a. This suggests that the two carriers' performance is indeed heterogeneous when accessed from HSR, and the two carriers can performance-wise complement each other. We further quantify at what time granularity one carrier can consistently outperform the other. Specifically, we define the *RTT Leading Time* as the longest consecutive period during which one carrier always has a lower RTT than the other. As shown in Fig. 4b, the median RTT leading time for Carrier A and B are 457 ms and 632 ms respectively. This indicates that the multipath heterogeneity on HSR is highly dynamic, changing every 2 to 4 RTTs (see also Fig. 4c), attributed to HSR's extreme mobility.



(a) Throughput/RTT ratio.

(b) RTT leading time.



(c) Snapshot of RTT ratio time series.

Figure 4: Measurement study of path diversity.

2.5 Implications on System Design

We summarize key findings of our measurements and their implications on system design.

• As the mobile networking performance is highly fluctuating on HSR, one needs to continuously probe the cellular link to get fresher link quality estimations and use it strategically;

• One may tackle the link dynamics by proactive reinjection, as the RTO-based retransmission is often inefficient (§2.2);

• One could leverage the predictability of handover failures to take early actions before losing the connectivity (§2.3);

• One can further leverage the path heterogeneity to mitigate the volatility on individual paths. The path selection requires judicious decisions based on traffic patterns, real-time link quality monitoring, and handover failure prediction (§2.4).

3 Handover Failure Prediction

In this section, we present how to leverage the available yet reliable information to predict handover failures, which is crucial to improving networking performance in our frequently disconnected networking environment.

Handover Success/Failure Determination Methodology. Our measurement in §2.3 provides evidence that HSR handover failures, which disrupt TCP performance for several seconds, are potentially predictable – they are more likely to fail if happened at a latter location in the overlap zone (for handover) and/or if RSRP is lower. Herein, we formulate handover success/failure determination task as a classification problem, and adopt SVM, a lightweight supervised machine learning algorithm fed with location (*i.e.,* longitude, latitude) and RSRP values when UE disconnects from the source cell as features and handover result as labels. We log all these relevant information into a database called LinkDB deployed on both mobile relay and remote proxy (see Fig. 7) and train the SVM with linear kernel function and L2 loss function for each source cell. By using location, the percentage of the linearly separable (successful and failed) handovers is 72.8% and 71.4% for carrier A and B respectively; by using RSRP, this number is reported as 73.5% and 67.1%. When jointly use location and RSRP, this number raises up to 92.6% and 89.2%. This data shows that for most cells, the handover result could be accurately inferred from the location and/or RSRP when the handover starts. Another implication is that if a fresh handover event is close to the historical handover failure data in the feature space, it is very likely to fail.

Handover Failure Prediction in POLYCORN. Although the aforementioned offline analysis shows promising results in determining whether the handover is successful or failed based on location and RSRP data, it is not straightforward to turn it into a practical online handover failure predictor. The main challenge is that the handover failure has to be predicted *in advance* so as to be useful for guiding interface scheduling, especially for downlink traffic. In other words, the time advance needs to take into account the tens or hundreds of milliseconds of delay for mobile relay to deliver the handover failure precaution signal to the remote proxy. In our data analysis, we also find that the LTE chipset can delay the RSRP log reporting to userspace for up to 200 ms. This fact together with RSRP's highly fluctuating nature (Fig. 2b) makes RSRP an error-prone feature for SVM to use for online handover failure prediction. Therefore, POLYCORN has to rely on location information only since it is truly predictable given the reliable train speed. In practice, the mobile relay sends its associated cell ID, train's speed and location, and current time to the remote proxy. On the remote proxy, LinkDB provides information about all historical handovers from this cell and calculates the predicted location of handover failure $\hat{L_{HOF}}$, defined as the average location of all handover failures. Then, LinkDB predicts $\hat{t_{HOF}}$, the time that train passes $\hat{L_{HOF}}$. If the current moment is approaching $\hat{t_{HOF}}$ (to be elaborated in §4.3), POLYCORN predicts that the handover in this cell would fail because it is too late to initiate it even from now.

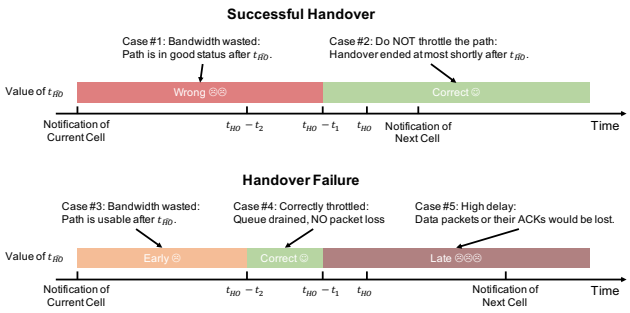


Figure 5: Types of Handover Prediction Results.

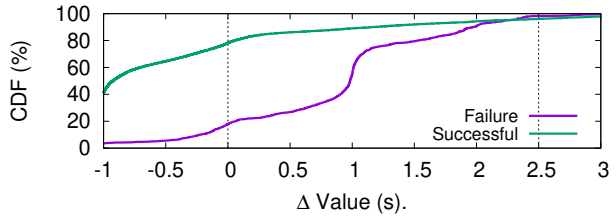


Figure 6: Handover Prediction Validation.

Prediction Validation. We consider the case where we throttle the path from $t_{\hat{HOF}}$ to the time when the UE connects to the next cell. The point is, if the handover failed, we should throttle the path right before the handover to drain the queue on it. Otherwise, we should only throttle the path for a very short period, or optimally do not throttle it. As illustrated in Fig. 5, let t_{HO} be the groundtruth value of handover time, $t_1 < t_2$ where t_1 and t_2 are two non-negative numbers, and hence there are five combinations of $\Delta = t_{HO} - t_{\hat{HOF}}$ and handover results. We set t_1 to 0 seconds where $t_{\hat{HOF}}$ exactly matches t_{HO}, and set t_2 to 2.5 seconds, beyond which the side effect brought by early prediction overweighs its benefit. Fig. 6 plots the distribution of Δ, where $t_{\hat{HOF}}$ is predicted using leave-one-trip-out cross-validation over the entire dataset. For handover failure, 80.6%, 1.3%, and 18.1% of the prediction results are correct, late, and early, respectively. For successful handovers, the correct rate is 78.2%. Here we note that POLYCORN seeks for a conservative approach towards handovers prediction and path throttling decision – it prioritizes avoiding the penalty of a handover failure misprediction and considers it acceptable to waste available bandwidth in the cases that successful handover predicted as failure (Case 1) or successfully predicting the handover failure but in a earlier moment (Case 3) – in both cases the networking performance already starts to degrade when approaching the handover point anyway. As to be shown in §6.2, such coarse-grained results is adequate for our system given the high speed and the GPS system errors.

4 System Design of POLYCORN

We now present the design of POLYCORN, a software solution for high-performance Internet access for ultra-high-speed transportation. The design goals of POLYCORN include the following: (1) *Be resilient to extreme mobility environment.*

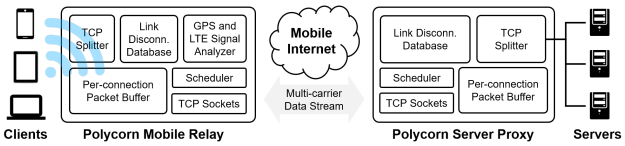


Figure 7: POLYCORN Architecture Overview.

POLYCORN should survive highly fluctuating network performance and inaccurate link quality estimations. (2) *Faster flow completion.* As opposed to bandwidth-intensive such as bulk data transfer [8], networked applications used by passengers, such as instant messaging and web browsing, typically have short or small sessions. It is therefore important to reduce the flow completion time. (3) *Effectively use multiple cellular carriers*. The multipath heterogeneity revealed in §2.4 should be leveraged for robust traffic delivery. (4) *Be practical for real-world deployment.* POLYCORN should be easy to deploy and ideally, be transparent to client and server applications and require no infrastructure modifications.

4.1 Overall Architecture

The high-level architecture of POLYCORN is illustrated in Fig. 7. As shown, POLYCORN leverages the dual-proxy architecture [22]. One proxy deployed at the on-board mobile relay (*i.e.,* cellular gateway) multiplexes passengers' uplink traffic over the paths of multiple cellular carriers; another proxy deployed at a cloud server performs the reverse operation of demultiplexing the traffic and delivering them to the destination servers. Each in-cloud proxy is paired with one on-board mobile relay, and more pairs can be flexibly set up for scaling up the service for larger HSR network. Downlink traffic is handled in a symmetric way: the cloud-side and on-board proxy perform multiplexing and demultiplexing, transparently. The two proxies are essential for providing transparent transport-layer multipath to off-the-shelf hosts.

The dual proxies offer a centralized place for multipath scheduling, *i.e.,* deciding which traffic should be transmitted over which path(s). One path is one subflow that exclusively uses one network interface (*i.e.,* SIM card). The scheduler for uplink and downlink traffic resides on the on-board LTE gateway and the in-cloud proxy, respectively. Multipath scheduling is one of the most critical components of a multipath transport system, in particular for HSR networking where the paths' performance is individually fluctuating and collectively heterogeneous. We are unaware of any multipath scheduler specifically designed for extreme mobility transportation, and POLYCORN's design fills this gap.

4.2 Composable Multipath Scheduler

Our measurement study in §2.5 suggests that multipath transport for HSR networking needs to consider multiple dimensions: performance fluctuation, predictable handover failures, and path heterogeneity, *etc.* To tackle such complexity, we adopt a novel framework that treats a multipath scheduler as a pipeline of modularized *schedulerlets*. Each schedulerlet encapsulates a multipath scheduling functionality that ma-

Performance Issue	Mitigation Strategy	Schedulerlet
Disruption due to handover failure	Filter paths facing imminent handover failures based on prediction	§4.3
Suboptimal scheduling due to path heterogeneity	Filter paths that lengthen TCP flow completion due to tail packets	§4.4
Single packet experiencing multiple RTOs	Proactively reinject packet clusters facing excessive retransmissions	§4.5
Stale performance metrics on idle paths	Opportunistically deliver redundant data over unselected paths	§4.6

Table 2: Logic flow from performance issues to designs.

nipulates three sets of paths: a *selected set* $\mathbb{S}$ containing the currently selected path(s) for data transmission, a *candidate set* $\mathbb{C}$ containing the candidate paths that can be selected, and an *unavailable set* $\mathbb{U}$ containing the unavailable paths that by default cannot be selected. The purpose of having $\mathbb{U}$ is to restrict path selection to only a subset of all paths. Initially, all the paths belong to $\mathbb{C}$. Depending on which set(s) to manipulate, we classify the schedulers into different categories: (1) a *candidate filter* that moves path(s) from $\mathbb{C}$ to $\mathbb{U}$; (2) a *selection filter* that moves paths from $\mathbb{S}$ to $\mathbb{C}$; and (3) a *soft selector* that moves path(s) from $\mathbb{C}$ to $\mathbb{S}$; (4) a *hard selector* that moves path(s) from $\mathbb{C}$ or $\mathbb{U}$ back to $\mathbb{S}$. The purpose of having a hard selector is to provide a mechanism that can "revive" any path. This is useful when paths' conditions are highly dynamic; it also ensures the completeness of the framework.

The schedulerlets are then strategically arranged to form the overall scheduling pipeline. Compared to a monolithic scheduler, our schedulerlet-based approach decouples the multipath scheduling logic, and thus significantly reduces the system complexity and development overhead, through the unified interface of schedulerlets (modifying $\mathbb{S}$, $\mathbb{C}$, and/or $\mathbb{U}$). Note that we do not claim that our design can achieve any optimality, since the "local" optimalities achieved by individual schedulerlets do not necessarily translate into a "global" optimality. Nevertheless, from a practical perspective, formulating a global optimization problem and solving it through a monolithic scheduler is extremely challenging due to the large solution space, real-time requirement, and volatile network dynamics. Therefore, we believe our "decouple-then-integrate" design achieves a right balance among practicality, simplicity, and performance, as to be thoroughly evaluated in §6.

We now consider how to instantiate the above generic framework into the concrete design of HSRSCH, the multipath transport scheduler for POLYCORN. The high-level design principle is to identify scenarios where vanilla MPTCP performs poorly, based on our extensive field studies, and improve them through judiciously designed schedulerlets. Specifically, Tab. 2 lists our identified performance issues, mitigation strategies, and the corresponding schedulerlets of HSRSCH, which will be detailed in the rest of §4. Here we describe the high-level scheduling pipeline. As shown in Fig. 8, the pipeline begins with a *candidate filter* schedulerlet that removes "bad" paths facing an imminent handover failure (§4.3), followed by a *soft selector* that performs initial, rough path selection. We find that the default minRTT scheduler

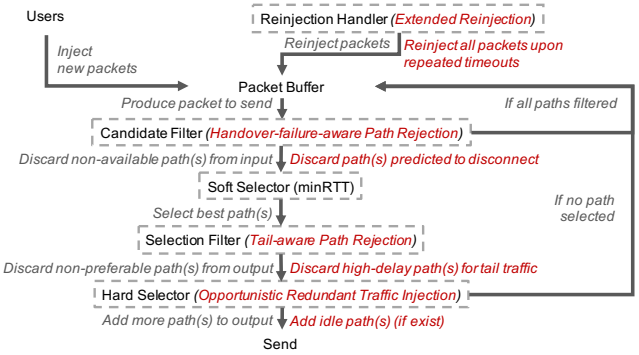


Figure 8: Composable scheduler in POLYCORN.

used by MPTCP can be properly leveraged as a soft selector, because favoring a low-latency path (when congestion window permits) is also desirable in HSR networking. Subsequently, we employ a *selection filter* to further remove certain selected paths, shortening the flow completion time (§4.4). Next, we apply a *hard selector* to make use of the remaining paths (in $\mathbb{C}$ and $\mathbb{U}$) – we use them for delivering redundant data and probing the bandwidth (§4.6). Finally, due to the high network condition volatility, performance degradation or even outage may still appear on a selected path despite the above schedulerlets. To tackle this, we also introduce a *reinjection handler* that dynamically redistributes scheduled packets to other paths. This mechanism tolerates other schedulerlets' errors and further improves the overall robustness (§4.5). Note that HSRSCH is designed to be scalable, *i.e.,* they could work with any number of available paths.

The aforementioned taxonomy of schedulerlets based on which sets ($\mathbb{S}$, $\mathbb{C}$, $\mathbb{U}$) they manipulate also helps decide the order of the schedulerlets. For example, the candidate filter is invoked first since it does not depend on other schedulerlets; the selection filter needs to examine the output of the soft selector so the former comes after the latter in the pipeline. The hard selector tries to make use of any unselected paths; it is therefore situated at the end of the pipeline.

4.3 Handover-failure-aware Path Rejection

Recall from §2.3 that handover failures incur considerable performance impact. Our first optimization has an intuitive idea: predict imminent handover failures using LinkDB (§3), and apply a schedulerlet (candidate filter) to disable the path(s) facing handover failure(s). The rationale is that, sending traffic over a path experiencing a blackout of several seconds caused by a failed handover will significantly lengthen the flow com-

pletion time. Therefore, it should be avoided at all costs. In our design, for downlink traffic scheduling[5], the gateway continuously sends the collected features (GPS reading and cell ID) to the in-cloud proxy where LinkDB runs. Those features are sent with top priority using a redundant scheduler to ensure that the proxy would receive the features in time. The proxy then predicts t_{HOF}, the interval between the current time and the next expected handover (§3). If t_{HOF} is predicted to be less than a threshold L, the proxy-side scheduler disables the path. The path will be re-enabled when the connectivity to the new cell is established. The threshold L incurs a tradeoff between bandwidth waste (occurs when disabling the path early) and performance degradation (occurs when disabling the path late). We configure L as: $L = RTT + \frac{E_{GPS}}{V_{HSR}}$. The first term is the estimated RTT between the in-cloud proxy and the LTE gateway. RTT is the lower bound of L because at least one round trip is needed for any in-flight data to be delivered with confirmation; if we send any data after $t + t_{HOF} - RTT$ (t is the current time), then we run into risks where the data/ACK delivery is affected by a failed handover. The second term $\frac{E_{GPS}}{V_{HSR}}$ accounts for the GPS localization inaccuracy, where E_{GPS} and V_{HSR} are the maximum GPS error (in meter) and the train's current speed (in m/s) respectively. Due to potentially erroneous GPS reading, the train's actual location may be ahead of the reported GPS location. Therefore, we need the second term for an additional safety margin. We conservatively set E_{GPS} to 20 m, and V_{HSR} is estimated from the recent GPS trajectory. We would also like to mention that in the corner case when all paths are predicted to experience handover failure soon, all of them would be disabled – this would effectively block all pending traffic until the mobile relay notifies the remote proxy of a new cell.

4.4 Tail-aware Path Rejection

The HSR networking performance is not only affected by handover failures, but also by the highly fluctuating channel quality due to HSR's high speed. When a path's quality deteriorates, HSRSCH needs to make a key decision: should the path be temporarily disabled? Here the tradeoff is bandwidth utilization vs. latency: skipping the path misses the opportunity of utilizing its (albeit low) bandwidth, while using the path can possibly lengthen the flow completion time compared to sending the data over a faster path.

To balance the above tradeoff, since our goal is to accelerate short flows dominating the traffic pattern on HSR, HSRSCH detects scenarios where a flow is about to end, and employs a schedulerlet (selection filter) that rejects sending *tail packets* over low-performance paths. Here tail packets reside at the end of a flow, whereas non-tail packets are at the beginning or in the middle of a flow. The rationale is that, sending a tail packet over a low-performance path is very likely to delay the flow completion. In contrast, transmitting a non-tail packet over a poor-quality path usually only incurs packet out-of-order with a negligible or small impact on the flow completion time, provided that (1) all the paths are fully utilized (*i.e.,* there is no idle period), and (2) the receiver has a large enough buffer to accommodate out-of-order packets (which we can ensure as we have control over both multiplexing proxies). To transparently identify tail packets without the knowledge of flow size, HSRSCH keeps monitoring TCP FIN or RST packets. Once a TCP FIN or RST is observed, HSRSCH marks all the packets in the send buffer and all future outgoing packets of the same flow as tail packets. We leave more sophisticated tail packet identification methods (*e.g.,* based on application semantics) as future work.

To cope with highly fluctuating network conditions, HSRSCH employs a new way of deciding whether flow f has low performance over path i. Specifically, HSRSCH compares two scenarios. In the first scenario, we schedule some packets of f on path i. The packet delivery time and henceforth the flow completion time of f is *at least*:

$$T^{-}_{i,f} = owd_i + \frac{buf_i}{bw_i}$$

where owd_i, buf_i, and bw_i are the estimated one-way delay, the send buffer occupancy level, and the estimated bandwidth of path i, respectively. $T^{-}_{i,f}$ estimates the time taken to drain the FIFO send buffer of path i prior to sending any new packet belonging to f. It is therefore a *lower bound* of f's completion time if any of its packets are scheduled on path i. In the second scenario, we do not use path i at all to schedule f. In this scenario, the optimal flow completion time of f is *at most*:

$$T^{+}_{i,f} = \min_{j \neq i} \left\{ \left(owd_j + \frac{buf_j + remain_f}{bw_j} \right) (1 + \eta_j) \right\}$$

where $remain_f$ denotes the remaining bytes of f yet to be sent, η_j is a relaxation parameter empirically defined as the ratio between path j's RTT variance and RTT, and j iterates over all the paths except path i. $T^{+}_{i,f}$ quantifies the time of transmitting all the remaining bytes of f over the best path (except i), considering the paths' bandwidth and current send buffer occupancy levels. It is a loose *upper bound* of the optimal (single-path) completion time of f, which can in fact be further reduced (albeit difficult to quantify) by distributing f over multipath. If $T^{-}_{i,f} > T^{+}_{i,f}$, it implies that there exists a better scheduling strategy of not using path i compared to any strategy of using path i; we thus determine that path i is too slow to schedule tail packets of f. See example in §A.

4.5 Extended Reinjection

Multipath transport needs to simultaneously manage multiple paths. To handle individual paths' failure, MPTCP has a built-in reinjection mechanism (also known as Opportunistic Retransmission [36]): upon RTO events, the oldest unacknowledged packet on the same path will be retransmitted (reinjected) over another path as determined by performing

[5]For brevity, we only describe downlink traffic scheduling. Uplink traffic scheduling is performed at the on-board gateway in a similar manner.

the scheduling again. We find that such a reinjection policy is too conservative since it handles reinjection on the basis of *individual packets*. In contrast, in HSR networks, packet delivery failures often occur in a *bursty* manner: if a packet experiences an RTO, the probability that the subsequent packets are delayed or lost becomes much higher. Bursty losses are common in wireless networks in general. Nevertheless, on HSR, the bursty pattern of packet losses is much more prominent as the UE is frequently disconnected from base stations due to failed handovers or low signal strength.

Inspired by the above observation, we design an Extended Reinjection mechanism for HSRSCH, whose basic idea is to detect scenarios where packets are undergoing multiple RTOs, and reinject packets in a *proactive and batched* manner to match the bursty packet loss pattern for HSR cellular access. Specifically, in our approach, when any packet experiences the k-th RTO, HSRSCH reinjects *all* the unacknowledged (*i.e.,* in-flight) packets on the same path to other path(s). Instead of executing the reinjection(s) *immediately* by invoking the scheduling algorithm multiple times (similar to what MPTCP does when executing each single reinjection operation), HSRSCH performs *lazy* reinjection: the to-be-reinjected packets are thrown back to their origin connection-level send buffers, and the actual reinjection will take place later when these packets are (re)scheduled according to their connection- and user-level priorities (at that time that destination path will also be determined[6]). The purpose of lazy reinjection is to maintain priority and fairness, *i.e.,* the reinjected packets are regarded as newly arrived so they do not bear an unfairly high priority over the reinjected path. This is particularly important when many packets belonging to the same connection are reinjected. Also note that for each reinjection, the receiver will receive at least two identical copies: the original packet and at least one reinjected packet; only the first arrived copy will be consumed by the receiver.

In the above algorithm, the parameter k incurs a tradeoff: a large k provides fewer reinjection opportunities, potentially worsening the performance on the current path where losses occur, whereas a small k makes reinjection more aggressive, adding more traffic burden on other paths. We take a data-driven approach to select the appropriate k value: in our dataset, setting k to 1, 2, and 3 incurs 47%, 15%, and 0.3% more redundant traffic, respectively. We therefore set $k = 3$ in our evaluation (§6.2).

4.6 Opportunistic Redundant Traffic Injection

The soft selector along with both filters intend to find the best path for each packet, leaving the remaining unselected path(s) idle. This will cause two major issues: (1) the available bandwidth on idle paths, despite their high latency, is wasted; (2) no performance measurement can be carried out on idle paths without traffic, and previous measurement quickly becomes stale due to the high path dynamics. To address both issues, we design a schedulerlet (hard selector) that opportunistically schedules redundant data over idle paths. This not only allows passive measurement to be conducted, but also provides extra resiliency to channel quality fluctuation, leading to further reduced flow completion time, *i.e.,* when one path experiences unexpected performance degradation, the receiver can still receive extra copies of the data delivered over other path(s). In HSRSCH, the idleness of a path is determined when no traffic has been scheduled over it for either α seconds, or β bytes worth of data, whatever occurs first. Once a path becomes idle, HSRSCH duplicates the next τ scheduled packets and transmits their duplicated copies over the idle path. If multiple idle paths are available, the duplicate copies will be transmitted over all the idle paths. We empirically set α = 1 second, β = 8 KB, and τ = 16, which were found to well balance the tradeoff between the performance and bandwidth overhead based on our on-board controlled experiments.

Besides fostering reliability under performance degradation, injecting traffic over an idle path brings another benefit: it allows the transport layer to keep the path performance statistics up-to-date. As shown in Fig. 1a, on HSR, a cellular link's quality changes almost every RTT. If no packet is sent over an idle path, TCP's built-in probing mechanism, which is piggybacked with user traffic, will be paused, and TCP will thus lose track of important metrics such as RTT and RTO. Our proactive injection design addresses this issue.

4.7 Putting Everything Together

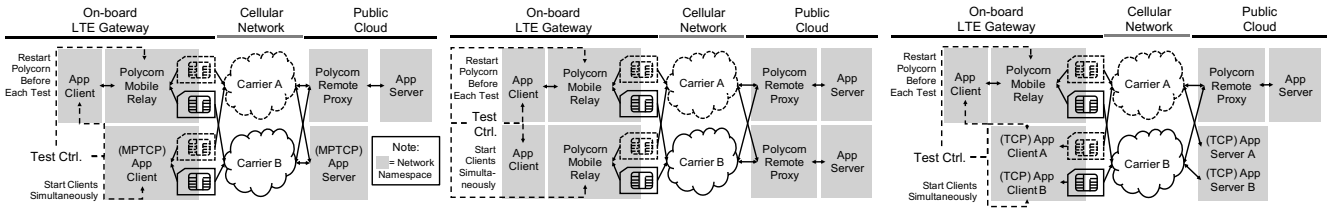
We integrate the above four schedulerlets into the composable framework introduced in §4.2. We next describe the detailed scheduling logic on both in-cloud proxy (for downlink traffic) and on-board LTE gateway (for uplink traffic).

Recall that POLYCORN is a multi-user system. In each scheduling round, HSRSCH begins with selecting a user to serve, and then picking a flow belong to the selected user. Our current implementation uses the standard proportional fair (PF) scheduling [37] for user selection, and round-robin for flow selection. More sophisticated scheduling algorithms can be plugged into our framework. Once the to-be-served flow is determined, HSRSCH schedules the flow's next packet, which is the untransmitted packet (including to-be-reinjected packets) with the smallest sequence number, as described in Fig. 8. Note that the reinjection handler runs in parallel with the scheduling thread that invokes the candidate/selection filters and soft/hard selectors.

5 Implementation

This section details the implementation of POLYCORN. Our high-level design goals consist of the following. First, POLYCORN should be practical. The required changes on clients' mobile devices should be minimized, or ideally none. Also, POLYCORN should be able to deploy on the HSR LTE gateway and keep its running components unmodified. Second, the data transport scheme should be able to schedule traffic

[6] We use a bitmap field to ensure that the same packet is not reinjected to its previously scheduled path.



(a) POLYCORN vs. MPTCP. (b) POLYCORN microbenchmark. (c) POLYCORN vs. TCP.

Figure 9: Experimental setup for different comparative evaluation.

with awareness of multiple users and connections.

Working with Unmodified Users and LTE Gateway. Based on TM^3 [21] and MPFlex [22] frameworks, POLYCORN uses TCP splitting to achieve transparency towards both clients and servers. A brief summary of TCP splitting is: when communicating with servers, POLYCORN acts as a forward proxy; when communicating with clients, POLYCORN acts as a reverse proxy. To avoid kernel modification which is not allowed by the LTE gateway vendor, we use raw socket instead of `netfilter`-based [38] kernel modules (which is used by MPFlex and TM^3) or high-performance packet I/O frameworks like DPDK [39] to capture user traffic. After capturing user traffic, we drop all user packets using `iptables` [40] to prevent the kernel from forwarding them. In this way, we implement POLYCORN completely in userspace. As for the well-known performance issue of raw sockets, our experiments show that raw sockets could operate at 300 Mbps on the LTE gateway, which is significantly higher than the peak aggregated throughput of the LTE interfaces. Moreover, POLYCORN works in a separated network namespace (netns) to avoid conflicts with runtime kernel configuration used by other programs and mitigate potential security issues. For instance, POLYCORN disables reverse-path filtering in its own netns to forward packets generated by POLYCORN with any source IP. Our design allows sensitive system configurations to be preserved in the original netns, thus isolate the security risk from normal runtime programs.

Multiuser Multipath Traffic Multiplexing. Similar to TM^3 and MPFlex, POLYCORN multiplexes user traffic onto off-the-shelf sockets to implement multipath data transfer. POLYCORN uses TCP sockets as its subflows. Although QUIC is better than TCP in terms of Head-of-Line blocking mitigation, especially in multiuser settings, we choose the "fallback" TCP sockets primarily because we encountered extensive rate limitation cases when launching QUIC flows in our measurements. This observation agrees with the findings in [41] to reveal that UDP traffic will most likely be treated as malicious flow by cellular carriers when sending in large volume, which situation might not disappear soon in most developing and under-developed countries. Therefore, we believe our choice is better for long-term real-world deployment. As for multiuser traffic scheduling, POLYCORN maintains a separated send/reinject buffer and a metadata set including user source IP, amount of sent traffic, etc, for each flow. With those metadata, in operation POLYCORN first determines which user/flow to serve, then checks the flow's send buffer to choose a packet to send. Finally, POLYCORN runs HSRSCH to choose interfaces to send the packet, as described in §4.2.

6 Evaluation

6.1 Experimental Setup

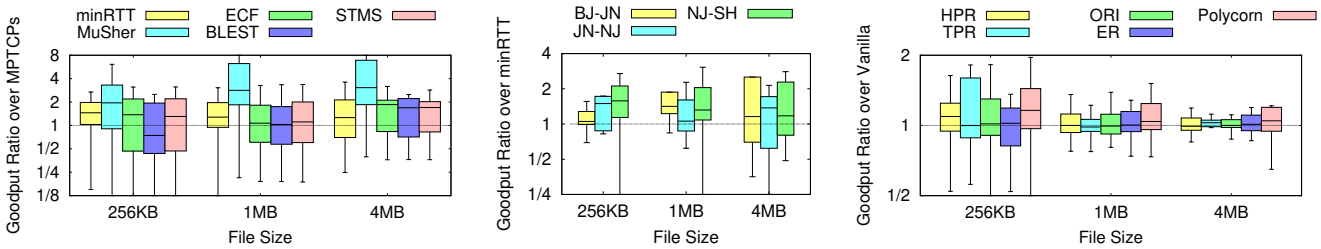
We carried out the experiments on Beijing-Shanghai HSR route, the one carrying most HSR passengers in the country.

Deployment on Operational System. We deployed POLYCORN mobile relay and server proxy on the high-speed train LTE gateways and public cloud servers respectively, with the same hardware configuration described in §2.1. More specifically, the mobile relay runs CentOS 7.3 with MPTCP kernel 0.94 (Linux 4.14) – we adhere to CentOS for POLYCORN deployment on the LTE gateway because it runs other mission-critical train-ground communication services developed by the operators and third-party IT service providers.

Fairness in Comparative Study. We made the following efforts to improve fairness in evaluating POLYCORN:

• *MPTCP Baseline* is configured in decoupled rather than default coupled congestion control mode (used in [15]) because the relay-proxy suite acting as an end-user traffic aggregation and delegation point should harness more wireless bandwidth from multiple cellular carriers instead of treating itself as a single user or session – it makes the baseline stronger and comparative evaluation fairer.

• *Pairwise study.* We carried all the experiments in the side-by-side concurrent test setting for 50 times with different software configuration tailored for fairness in different scenario, including comparing POLYCORN with MPTCP variants (Fig. 9a) and its own variants for microbenchmark (Fig. 9b) in single session bulk data download (§6.2), and comparing POLYCORN with SPTCP (Fig. 9c) and MPTCP (Fig. 9a) in multi-user instance messaging settings (§6.3). Specifically, we managed to obtain exclusive access to four SIM cards with two for each carrier respectively from the HSR WiFi service division, pair each transport software solution (*e.g.,* SPTCP, MPTCP and POLYCORN) with two cards from different carriers, and perform all the experiments on the operational HSR with a speed of 300+ km/h. Note that the scheduler and system design of POLYCORN can easily scale to more than two



(a) Comparing with state-of-art MPTCPs. (b) On different route segments. (c) Microbenchmark.

Figure 10: Bulk data downloading time comparative evaluation.

interfaces in LTE, 5G, *etc.* We choose to perform evaluations with two cards primarily because that is the maximum number we can obtain permission to access presently.

6.2 Bulk Data Download Performance

We first evaluate the performance of POLYCORN in comparison to the state-of-the-art MPTCP solutions and its own variants for microbenchmark from a single session perspective. We use fixed-size flows of 256 KB, 1 MB, 4 MB to examine POLYCORN's performance in a pairwise manner. Specifically, we choose to use average goodput ratio of test object to its opponent as the primary metric instead of showing their absolute value. This is because the cellular link quality and the associated mobile networking performance differs significantly from one trackside location to another – the large variance for a single performance profile prevents comparative quantitative illustration and analysis between the two solution in the same testing environment.

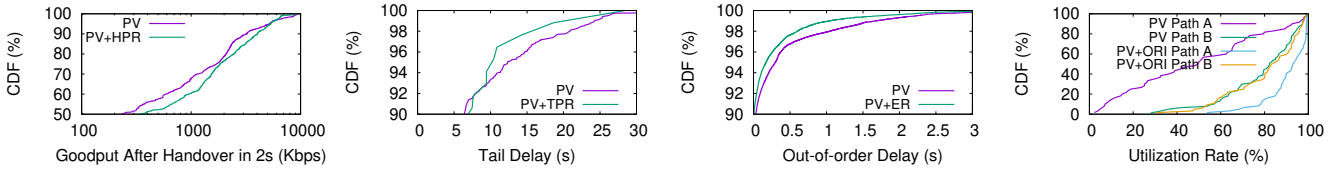
Comparison with MPTCP Schedulers. We examine the efficacy of POLYCORN by showing its goodput ratio relative to the state-of-the-art MPTCP schedulers in Fig. 10a. We make three key observations: First, POLYCORN wins in almost all the cases, demonstrating that the four data-driven scheduler designs collectively and successfully improve the networking performance under different corner cases unique to HSR that are not well handled by all the state-of-the-art MPTCP scheduling strategies. Second, minRTT (as the default MPTCP scheduler) performs similarly compared with BLEST [28] (1 MB and 4 MB), ECF [27], and STMS [18], and outperforms MuSher [17]. This indicates that the strategies assuming predictable path heterogeneity are not advantageous over the simplest (and generic) one when encountering the highly dynamic networking environment. Specifically, POLYCORN outperforms minRTT by 1.45x, 1.28x, and 1.26x for 256 KB, 1 MB, and 4 MB respectively with overall 1.31x. In general, all the state-of-the-art schedulers trust and exclusively rely on their estimations of network condition to make interface scheduling decisions accordingly, while the fluctuating network nature on HSR makes the estimations error-prone and degrade the accuracy of the scheduling decisions. This is also why MuSher performs worse than others in our case: it assigns traffic to interfaces according to the quickly varying ratio of throughput on each interface and failed to catch up with the changes; others who rely more on RTT performed better simply because RTT is relatively less variable. POLYCORN chooses to employ coarse-grained but more reliable event information and achieves better performance. Third, POLYCORN performs worse than BLEST in the shortest flow. Unlike POLYCORN that tries to improve bandwidth utilization (*i.e.,* Tail-aware Path Rejection), BLEST does not schedule packets on the path that would potentially cause head-of-line blocking, and hence achieves zero tail delay. This benefit comes at the cost of reduced bandwidth utilization, and will not continue to stay in a long flow.

Different HSR Route. We also examine the robustness of POLYCORN in different segments on the Beijing-Shanghai HSR route with different cellular coverage and terrain patterns of different channel characteristics [42]. As shown in Fig. 10b, POLYCORN consistently outperforms minRTT – the performance gain of POLYCORN on Beijing-Jinan (plain/rural), Jinan-Nanjing (hills/rural), Nanjing-Shanghai routes (urban/plain) are 19.4%, 25.2%, 44.9%, respectively.

Microbenchmark. We further study the performance gain of HSRSCH and our four individual scheduler designs over POLYCORN Vanilla (*i.e.,* POLYCORN with minRTT scheduler). We plot the goodput ratio of the aforementioned five multipath transmission schemes and POLYCORN Vanilla in Fig. 10c. The four proposals all positively improve POLYCORN's performance by 7.0%, 18.8%, 4.2%, and 2.9% on average for the three different file sizes, and they cumulatively contribute to 16% goodput gain.

• *Handover-failure-aware Path Rejection* is designed to mask the impact of packet loss during the disconnected period and the consequent RTO to the TCP (*e.g.,* slow start) by receiving or predicting handover from explicit signals from LTE real-time analytic and/or our LinkDB information and take action accordingly. Specifically, by sending redundant cross-flow copies during the period any interface encountering disconnectivity, as shown in Fig. 11a, POLYCORN can recover from the disconnection much faster, *i.e.,* achieve 1.2x and 1.5x as mean and median values within 2 seconds after handover failure, which typically lasts a few seconds or longer.

• *Tail-aware Path Rejection* is used to avoid the out-of-order delay caused by the slow paths by refusing to inject data on the interface that may increase the flow completion time. As shown in Fig. 11b, the tail delay was reduced by 5.6% on average and 15.6% in 95 percentile compared to POLYCORN



(a) HPR. (b) TPR. (c) ER. (d) ORI.

Figure 11: POLYCORN microbenchmark comparative evaluation.
(PV: Polycorn Vanilla; HPR: Handover-failure-aware Path Rejection; TPR: Tail-aware Path Rejection;
ER: Extended Reinjection; ORI: Opportunistic Redundant Traffic Injection.)

Vanilla. This mechanism is useful especially for those tail delay greater than 10 seconds, which is due to the high packet loss rate and prolonged retransmission time of the interfaces with abnormal high RTT.

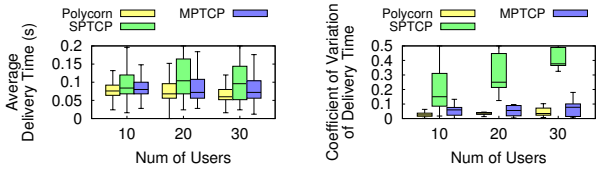
• *Extended Reinjection* mainly focuses on reducing extremely high retransmission time and leads to a significant reduction of out-of-order delay, which is shown in Fig. 11c. It reduces out-of-order delay by 23% among all the out-of-order packets, and 4% out-of-order delay among all the packets.

• *Opportunistic Redundant Traffic Injection* aims to proactively update the performance metrics of interfaces that have been idle for a while due to higher measured RTT. This helps HSRSCH more quickly discover recovered paths and improve path utilization. As shown in Fig. 11d, POLYCORN Vanilla simply ignores the path with much higher RTT. By employing opportunistic probing mechanism, the utilization rate of the path appears to be worse is increased by more than 60%, which allows better bandwidth utilization from all the paths.

Remarks. We note that POLYCORN exhibits non-trivial variation in its performance, and sometimes it falls behind the counterpart solutions. There are two major reasons: 1) It is difficult to exactly *repeat* tests on HSR: minor difference in test location results large difference in network condition – given the fluctuating network delay, the remote proxy (*i.e.,* sender) cannot accurately learn about the location of the train; 2) POLYCORN works with inaccurate handover failure predictions and TCP performance metrics. Our schedulerlets could tolerant minor errors in the context data, *e.g.,* comparative operators tolerates minor error in RTT. However, there exist cases where other solution has the proper information to make right scheduling decisions while POLYCORN does not.

6.3 Multi-user Instant Messaging Performance

We next evaluate POLYCORN in a multi-user setting, and choose instant messaging, the most popular application of HSR passengers as a representative use case for a case study. To best emulate the application behavior, we establish a long-lived TCP connection (adopted by many instant messaging application including WeChat, the most popular one in China) between POLYCORN mobile relay and server proxy for each user. We let each user sends 100 messages concurrently with pre-generated intervals following the exponential distribution. Each messaging event includes a 100-byte uplink message and an immediate 4-byte downlink one. Note that POLYCORN adopts a symmetric scheduler design for both downlink and uplink, which makes our data-driven interface scheduling work with uplink without extra effort. We perform concurrent pairwise experiments for POLYCORN vs. SPTCP and POLYCORN vs. MPTCP-minRTT respectively – SPTCP with round-robin scheduling cross different SIM card is the current operational solution used by the HSR WiFi systems due to its simplicity and interface-level fairness.



(a) Performance. (b) User fairness.

Figure 12: Multi-user instant messaging evaluation.

Experimental Results. Benefiting from the multi-stage data-driven scheduler design, POLYCORN outperforms SPTCP and MPTCP in both performance and user fairness. As shown in Fig. 12a, POLYCORN consistently improves aggregated instant messaging performance when the number of users ranges from 10 to 30. On average, POLYCORN reduces delivery time by 45% and 16% in comparison to SPTCP and MPTCP respectively, and tail delay, *e.g.,* 90 percentile, by 34% and 14%. In terms of user fairness, we use the coefficient of variation to quantify the variance of message delivery time regardless of the mean value across tests on different route segments with diverse networking conditions. As we can see in Fig. 12b, POLYCORN reduces the coefficient of variation by 86% and 49% on average when compared with SPTCP and MPTCP respectively, which significantly improves fairness across different on-board users.

7 Discussion

POLYCORN for 5G. Our evaluation does not cover 5G because the 5G CPE (Customer Premise Equipment) is not available on our HSR WiFi system yet. However, we believe POLYCORN's techniques remain applicable to 5G. For example, a recent measurement reveals that on 5G HSR, the handover failure rate is comparable to LTE [20], and multiple studies suggest that 5G suffers from higher bandwidth fluctuation and packet losses compared to 4G [20, 26].

Fairness. We discuss two fairness issues here. First, the fairness among POLYCORN users is ensured by POLYCORN's multi-user scheduling algorithm (we use proportional fair scheduling, see §4.7). Second, the fairness between POLYCORN users and non-POLYCORN users (who use their own cellular data plan) is typically guaranteed by the LTE base station. Also, POLYCORN uses unmodified TCP congestion control for each multiplexed long-lived TCP connection established for each sim card. This further minimizes the risk of POLYCORN being overly aggressive.

Other Mobile Applications. We have evaluated POLYCORN on bulk download (with different sizes) and instant messaging (with different number of users). Other applications popular on HSR include web browsing and short videos. Short video traffic may resemble bulk download that POLYCORN can effectively handle, whereas web browsing further involves client-side processing overhead, which may reduce the effectiveness of POLYCORN that only optimizes content delivery.

Scalability. The POLYCORN architecture natively supports adding more wireless interfaces (*e.g.,* SIM cards) and pairs of onboard mobile relay & in-network server proxy to meet the scalability requirement. We leave larger-scale evaluations of POLYCORN as our future work.

Head-of-Line (HoL) Blocking Issue in TCP Reuse. POLYCORN uses one multipath connection formed with TCP subflows to transmit all user traffic. The use of TCP will inevitably introduces the HoL blocking at both intra-connection and inter-user level due to due to its byte-level ordering guarantee, which is an overkill in POLYCORN's multi-user multiplexing context. We envision that this problem will be solved by (MP)QUIC with its support of out-of-order delivery and cross-path acknowledgment [43, 44].

8 Related Work

Mobile Networking Performance Improvement. A plethora of research efforts have been devoted to improve network performance (under high mobility) through developing robust handover schemes [26], simplifying cellular control plane [45, 46], and fixing base station-side policy configuration bugs [47, 48]. Unlike POLYCORN, all the above approaches require modifications to the cellular infrastructure. There are also studies at upper layers, *e.g.,* designing customized single-path transport protocol [49] and optimizing congestion control algorithms [50–54]. POLYCORN instead proposes a holistic multipath solution with new optimization dimensions.

Performance-enhancement Proxy (PEP). In vehicular systems, PEPs are often deployed on mobile relays, to leverage carrier diversity and UDP encapsulation for bandwidth aggregation and mitigating link failure, *e.g.,* through striping [11, 55], opportunistic erasure coding [12], and flow splicing [13]. Specifically, the work [35, 55] present the idea of location-aware link characteristics (*e.g.,* throughput and availability) prediction and packet scheduling. PEPs can also be deployed in fixed locations in the Internet [56–58]. POLYCORN synthesizes all the ideas above and presents four multipath scheduling strategies dedicated to addressing the unique networking challenges in extreme mobility.

Multipath Transport Architecture. Transmitting data over multiple paths can be realized at different layers, *e.g.,* WNIC driver [59, 60], in-kernel transport layer [36, 61], light kernel modification [21, 22], and UDP encapsulation [62–64]. Differing from the above, POLYCORN is an entire userspace solution reusing Linux TCP for the benefits of OS/middlebox compatibility, application transparency, and good runtime performance, with multipath, multi-user multiplexing support.

Scheduling over Heterogeneous Paths. Several generic multipath transport schedulers have been proposed to mitigate the head-of-line blocking and out-of-order delay incurred by imbalanced subflows, such as opportunistic declining [28] and migrating [27], intra-chunk opposite scheduling [65], out-of-order transmission [18], and reactive bandwidth probing [17]. None of them considers HSR-specific aspects, and many of them [27, 28, 65] only work for two paths. Horde [66] and miDRR [67] allows user/app to specify their QoS requirement and perform packet scheduling accordingly. RAVEN [14] achieves low latency by extensively leveraging redundant transmission over multiple paths. HSRSCH brings new optimization dimensions integrated through a composable schedulerlet pipeline, and strikes a balance where overall throughput is not harmed by large amount of redundant traffic and latency of short flows are preserved.

9 Conclusion

The popularity of HSR systems brings the requirement of high-performance data networking under extreme mobility more tangible than ever. In this work, we have addressed the challenge of bringing seamless Internet service to passengers on HSR by synthesizing multipath transmission and data-driven scheduling techniques into a practical and readily deployable system design. Extensive experimental results have demonstrated the effectiveness of our system design dedicated to extreme-mobility. We believe that our upper layer optimization solution can seamlessly cooperate with the ongoing 5G/NextG(-Unlicensed) evolution.

Acknowledgment

We are grateful to the reviewers for their constructive critique, and our shepherd Keith Winstein in particular, for his valuable comments, all of which have helped us greatly improve this paper. We also thank Dina Katabi, Songwu Lu, Kun Tan and Yong Cui for their thoughtful input based on an early version of the work. This work was supported by National Key Research and Development Plan, China (Grant No. 2020YFB1710900), National Natural Science Foundation of China (Grant No. 62022005 and 62172008) and Microsoft Research Asia. Chenren Xu is the corresponding author.

References

[1] China launches upgraded high-speed trains, with wi-fi. https://gbtimes.com/china-launches-upgraded-high-speed-trains-wi-fi.

[2] Jr to launch free wi-fi on bullet trains from may. https://mainichi.jp/english/articles/20180303/p2a/00m/0na/009000c.

[3] South korea's brand-new olymic bullet train will make americans jealous. https://mic.com/articles/187809/south-koreas-brand-new-olympic-bullet-train-will-make-americans-jealous.

[4] Spain's high speed trains introduce high speed wifi. https://www.thelocal.es/20161104/spains-high-speed-trains-introduce-high-speed-wifi.

[5] Eurostar - on-board entertainment server launched. https://nomad-digital.com/customer-story/eurostar-on-board-entertainment-server-launched.

[6] Enjoy the standard experience. https://www.thalys.com/nl/en/info-services/enjoy-the-standard-experience.

[7] Deutsche bahn launches 'wifi @ db' wlan network. https://www.globalrailwayreview.com/news/109948/deutsche-bahn-launches-wifi-db-wlan-network.

[8] Jing Wang, Yufan Zheng, Yunzhe Ni, Chenren Xu, Feng Qian, Wangyang Li, Wantong Jiang, Yihua Cheng, Zhuo Cheng, Yuanjie Li, Xie Xiufeng, Yi Sun, and Zhongfeng Wang. An active-passive measurement study of tcp performance over lte on high-speed rails. In *ACM MobiCom*, 2019.

[9] Chenren Xu, Jing Wang, Zhiyao Ma, Yihua Cheng, Yunzhe Ni, Wangyang Li, Feng Qian, and Yuanjie Li. A first look at disconnection-centric tcp performance on high-speed railways. *IEEE Journal on Selected Areas in Communications*, 38(12), 2020.

[10] Multipath tcp - linux kernel implementation. https://multipath-tcp.org.

[11] Pablo Rodriguez, Rajiv Chakravorty, Julian Chesterfield, Ian Pratt, and Suman Banerjee. Mar: A commuter router infrastructure for the mobile internet. In *ACM MobiSys*, 2004.

[12] Ratul Mahajan Jitendra Padhye Sharad Agarwal and Brian Zill. High performance vehicular connectivity with opportunistic erasure coding. In *USENIX ATC*, 2012.

[13] Joshua Hare, Lance Hartung, and Suman Banerjee. Transparent flow migration through splicing for multi-homed vehicular internet gateways. In *IEEE VNC*, 2013.

[14] HyunJong Lee, Jason Flinn, and Basavaraj Tonshal. Raven: Improving interactive latency for the connected car. In *ACM MobiCom*, 2018.

[15] Li Li, Ke Xu, Tong Li, Kai Zheng, Chunyi Peng, Dan Wang, Xiangxiang Wang, Meng Shen, and Rashid Mijumbi. A measurement study on multi-path tcp with multiple cellular carriers on high speed rails. In *ACM SIGCOMM*, 2018.

[16] Qingyang Xiao, Ke Xu, Dan Wang, Li Li, and Yifeng Zhong. Tcp performance over mobile networks in high-speed mobility scenarios. In *IEEE ICNP*, 2014.

[17] Swetank Kumar Saha, Shivang Aggarwal, Rohan Pathak, Dimitrios Koutsonikolas, and Joerg Widmer. Musher: An agile multipath-tcp scheduler for dual-band 802.11 ad/ac wireless lans. In *ACM MobiCom*, 2019.

[18] Hang Shi, Yong Cui, Xin Wang, Yuming Hu, Minglong Dai, Fanzhao Wang, and Kai Zheng. Stms: Improving mptcp throughput under heterogeneous networks. In *USENIX ATC*, 2018.

[19] Li Li, Ke Xu, Dan Wang, Chunyi Peng, Kai Zheng, Rashid Mijumbi, and Qingyang Xiao. A longitudinal measurement study of tcp performance and behavior in 3g/4g networks over high speed rails. *IEEE/ACM Transactions on Networking*, 25(4), 2017.

[20] Yueyang Pan, Ruihan Li, and Chenren Xu. The first 5g-lte comparative study in extreme mobility. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(1), 2022.

[21] Feng Qian, Vijay Gopalakrishnan, Emir Halepovic, Subhabrata Sen, and Oliver Spatscheck. Tm 3: flexible transport-layer multi-pipe multiplexing middlebox without head-of-line blocking. In *ACM CoNEXT*, 2015.

[22] Ashkan Nikravesh, Yihua Guo, Feng Qian, Z Morley Mao, and Subhabrata Sen. An in-depth understanding of multipath tcp on mobile devices: measurement and system design. In *ACM MobiCom*, 2016.

[23] Fumihiro Hasegawa, Akinori Taira, Gosan Noh, Bing Hui, Hiroshi Nishimoto, Akihiro Okazaki, Atsushi Okamura, Junhwan Lee, and Ilgyu Kim. High-speed train communications standardization in 3gpp 5g nr. *IEEE Communications Standards Magazine*, 2(1), 2018.

[24] Bo Ai, Andreas F Molisch, Markus Rupp, and Zhang-Dui Zhong. 5g key technologies for smart railways. *Proceedings of the IEEE*, 108(6), 2020.

[25] Study on international mobile telecommunications (imt) parameters for 6.425 - 7.025 ghz, 7.025 - 7.125 ghz and 10.0 - 10.5 ghz. https://www.3gpp.org/DynaReport/38921.htm.

[26] Yuanjie Li, Qianru Li, Zhehui Zhang, Ghufran Baig, Lili Qiu, and Songwu Lu. Beyond 5g: Reliable extreme mobility management. In *ACM SIGCOMM*, 2020.

[27] Yeon-sup Lim, Erich M Nahum, Don Towsley, and Richard J Gibbens. Ecf: An mptcp path scheduler to manage heterogeneous paths. In *ACM CoNEXT*, 2017.

[28] Simone Ferlin, Özgü Alay, Olivier Mehani, and Roksana Boreli. Blest: Blocking estimation-based mptcp scheduler for heterogeneous networks. In *IFIP Networking*, 2016.

[29] China's high-speed rail links winter olympics cities. http://english.cctv.com/2019/12/30/ARTIITvoMUF29MZmtX5y4t9m191230.shtml.

[30] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs. Iperf: The tcp/udp bandwidth measurement tool. http://dast.nlanr.net/Projects.

[31] Gerald Combs. Tshark-the wireshark network analyser. http://www.wireshark.org.

[32] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, et al. Bbr: congestion-based congestion control. *Communications of the ACM*, 60(2), 2017.

[33] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review*, 42(5), 2008.

[34] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. Mobileinsight: Extracting and analyzing cellular network information on smartphones. In *ACM MobiCom*, 2016.

[35] Jun Yao, Salil S Kanhere, and Mahbub Hassan. Improving qos in high-speed mobility using bandwidth maps. *IEEE Transactions on Mobile Computing*, 11(4), 2011.

[36] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be? designing and implementing a deployable multipath tcp. In *USENIX NSDI*, 2012.

[37] Raymond Kwan, Cyril Leung, and Jie Zhang. Proportional fair multiuser scheduling in lte. *IEEE Signal Processing Letters*, 16(6), 2009.

[38] The netfilter.org project. https://www.netfilter.org/.

[39] Linux Foundation. Data plane development kit (DPDK), 2015.

[40] The netfilter.org "iptables" project. https://www.netfilter.org/projects/iptables/index.html.

[41] Korian Edeline, Mirja Kühlewind, Brian Trammell, and Benoit Donnet. copycat: Testing differential treatment of new transport protocols in the wild. In *ACM ANRW*, 2017.

[42] Cheng-Xiang Wang, Ammar Ghazal, Bo Ai, Yu Liu, and Pingzhi Fan. Channel measurements and models for high-speed train communication systems: A survey. *IEEE communications surveys & tutorials*, 18(2), 2015.

[43] J Iyengar and M Thomson. Rfc 9000 quic: A udp-based multiplexed and secure transport. *Omtermet Emgomeeromg Task Force*, 2021.

[44] https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/.

[45] Zafar Ayyub Qazi, Melvin Walls, Aurojit Panda, Vyas Sekar, Sylvia Ratnasamy, and Scott Shenker. A high performance packet core for next generation cellular networks. In *ACM SIGCOMM*, 2017.

[46] Yuanjie Li, Zengwen Yuan, and Chunyi Peng. A control-plane perspective on reducing data access latency in lte networks. In *ACM MobiCom*, 2017.

[47] Yuanjie Li, Haotian Deng, Jiayao Li, Chunyi Peng, and Songwu Lu. Instability in distributed mobility management: Revisiting configuration management in 3g/4g mobile networks. In *ACM SIGMETRICS*, 2016.

[48] Zengwen Yuan, Qianru Li, Yuanjie Li, Songwu Lu, Chunyi Peng, and George Varghese. Resolving policy conflicts in multi-carrier cellular access. In *ACM MobiCom*, 2018.

[49] Hongke Zhang, Wei Quan, Jiayang Song, Zhongbai Jiang, and Shui Yu. Link state prediction-based reliable transmission for high-speed railway networks. *IEEE Transactions on Vehicular Technology*, 65(12), 2016.

[50] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. Adaptive congestion control for unpredictable cellular networks. In *ACM SIGCOMM*, 2015.

[51] Wai Kay Leong, Zixiao Wang, and Ben Leong. Tcp congestion control beyond bandwidth-delay product for mobile cellular networks. In *ACM CoNEXT*, 2017.

[52] Shinik Park, Jinsung Lee, Junseon Kim, Jihoon Lee, Sangtae Ha, and Kyunghan Lee. Exll: An extremely low-latency congestion control for mobile cellular networks. In *ACM CoNEXT*, 2018.

[53] Ke Liu, Zhongbin Zha, Wenkai Wan, Vaneet Aggarwal, Binzhang Fu, and Mingyu Chen. Optimizing tcp loss recovery performance over mobile data networks. *IEEE Transactions on Mobile Computing*, 19(6), 2019.

[54] Soheil Abbasloo, Yang Xu, and H Jonathan Chao. C2tcp: A flexible cellular tcp to meet stringent delay requirements. *IEEE Journal on Selected Areas in Communications*, 37(4), 2019.

[55] Joshua Hare, Lance Hartung, and Suman Banerjee. Beyond deployments and testbeds: experiences with public usage on vehicular wifi hotspots. In *ACM MobiSys*, 2012.

[56] Rajiv Chakravorty, Sachin Katti, Ian Pratt, and Jon Crowcroft. Using tcp flow-aggregation to enhance data experience of cellular wireless users. *IEEE Journal on Selected Areas in Communications*, 23(6), 2005.

[57] Kyu-Han Kim and Kang G Shin. Prism: Improving the performance of inverse-multiplexed tcp in wireless networks. *IEEE Transactions on Mobile Computing*, 6(12), 2007.

[58] Jiasi Chen, Rajesh Mahindra, Mohammad Amir Khojastepour, Sampath Rangarajan, and Mung Chiang. A scheduling framework for adaptive video delivery over cellular networks. In *ACM MobiCom*, 2013.

[59] Srikanth Kandula, Kate Ching-Ju Lin, Tural Badirkhanli, and Dina Katabi. Fatvap: Aggregating ap backhaul capacity to maximize throughput. In *USENIX NSDI*, 2008.

[60] Anthony J Nicholson, Scott Wolchok, and Brian D Noble. Juggler: Virtual networks for fun and profit. *IEEE Transactions on Mobile Computing*, 9(1), 2010.

[61] Alexander Frömmgen, Amr Rizk, Tobias Erbshäußer, Max Weller, Boris Koldehofe, Alejandro Buchmann, and Ralf Steinmetz. A programming model for application-defined multipath tcp scheduling. In *ACM Middleware*, 2017.

[62] Luiz Magalhaes and Robin Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *IEEE ICNP*, 2001.

[63] Cheng-Lin Tsao and Raghupathy Sivakumar. On effectively exploiting multiple wireless interfaces in mobile hosts. In *ACM CoNEXT*, 2009.

[64] Quentin De Coninck and Olivier Bonaventure. Multipath quic: Design and evaluation. In *ACM CoNEXT*, 2017.

[65] Yihua Ethan Guo, Ashkan Nikravesh, Z Morley Mao, Feng Qian, and Subhabrata Sen. Accelerating multipath transport through balanced subflow completion. In *ACM MobiCom*, 2017.

[66] Asfandyar Qureshi and John Guttag. Horde: separating network striping policy from mechanism. In *ACM MobiSys*, 2005.

[67] Kok-Kiong Yap, Te-Yuan Huang, Yiannis Yiakoumis, Sandeep Chinchali, Nick McKeown, and Sachin Katti. Scheduling packets over multiple interfaces while respecting user preferences. In *ACM CoNext*, 2013.

A Example for Tail-aware Path Rejection

Fig. 13 exemplifies how tail-aware path rejection works on HSR. In this example, the traffic consists of a flow whose FIN packet was received by POLYCORN (so all the subsequent packets are tail packets). There are two paths A and B, whose estimated RTT and send buffer occupancy levels are plotted in the top and bottom subfigure, respectively. Path A has a low RTT and its send buffer is almost full, whereas Path B has a high RTT and its buffer occupancy level is low. MPTCP's default minRTT scheduler frequently schedules tail packets to Path B because Path A is busy (congestion window being full). However, our algorithm usually rejects Path B because the flow completion time will reduce if we wait for Path A to become available and send tail packets over A. This results in a large number of path rejection instances.

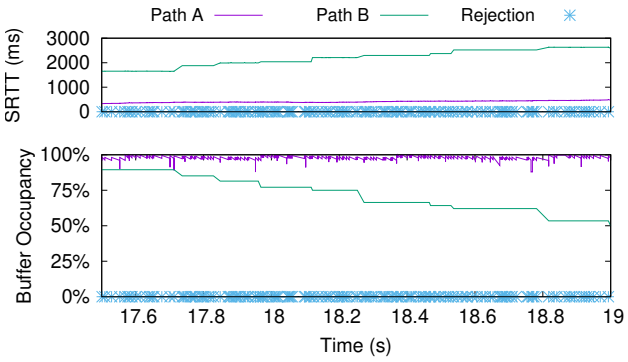


Figure 13: Reject Trace.