Towards Efficient Transport Mechanisms in Mobile Internet : Measurement and Optimization

Furong Yang

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité

Informatique

Laboratoire d'Informatique de Paris 6

Ecole Doctorale Informatique, Télécommunications et Electronique

Présentée par

Furong YANG

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Sujet de la thèse :

Towards Efficient Transport Mechanisms in Mobile Internet: Measurement and Optimization

soutenue le 02 Novembre 2022

devant le jury composé de :

M. Giovanni PAU	Università di Bologna	Directeur de thèse
M. Gaogang XIE	CNIC, Chinese Academy of Sciences	Directeur de thèse
M. Ke XU	Tsinghua University	Rapporteur
M. Kavé SALAMATIAN	Université de Savoie et du Mont Blanc	Rapporteur
M. Zhenyu LI	ICT, Chinese Academy of Sciences	Examinateur
M. Serge FDIDA	Sorbonne Université	Président du jury
Mme. Silvia MIRRI	Università di Bologna	Examinatrice
Mme. Kanchana KANCHANASUT	Asian Institute of Technology	Examinatrice

Curriculum Vitae

First Name/Name: Furong YANG Gender: Man Nationality: China

Education

[1] May. 2018 – Present, Cotutelle Ph.D., Sorbonne Université, France.

[2] Sept. 2014 – Present, Ph.D., Institute of Computing Technology, Chinese Academy of Sciences and University of Chinese Academy of Sciences, China.

[3] Sept. 2010 – June 2014, B.E., Hunan University, China.

Publications

[1] **Yang F**, Ferlini A, Aguiari D, et al. Revisiting WiFi offloading in the wild for V2I applications [J]. Computer Networks, 2022, 202: 108634.

[2] **Yang F**, Wu Q, Li Z, et al. BBRv2+: Towards balancing aggressiveness and fairness with delay-based bandwidth probing [J]. Computer Networks, 2022, 206: 108789.

[3] Zheng Z, Ma Y, Liu Y, **Yang F**, Li Z, et al. XLINK: QoE-driven multi-path QUIC transport in large-scale video services [C]//SIGCOMM 21: Proceedings of the 2021 ACM SIGCOMM 2021 Conference. New York, NY, USA: Association for Computing Machinery, 2021: 418-432.

Manuscripts

[1] **Yang F**, Li Z, Zhou J, et al. Data-driven Dynamic Selection of Multipath Congestion Control Algorithm, Under Preparation.

Patents

[1] CN202111240113.5, "A delay-based BBRv2 bandwidth probing method and system", **Yang F**, Wu Q, Li Z, Xie G (Chinese Patent).

[2] CN201610805579.8, "A cache-coloring based memory allocation method and apparatus for lookup trees", He P, **Yang F**, Xie G (Chinese Patent).

I would like to dedicate this dissertation to my loving parents ...

Acknowledgements

There were many challenges presented in the way of accomplishing my Ph.D. I would like to thank all of the people who have helped me to overcome all the challenges that I have encountered both in my life and in research during my Ph.D.

First, I would like to thank my advisors, Giovanni and Gaogang, for accepting me as a doctoral student. They not only taught me the things related to doing research but also provided a lot of help in my life.

I would like to thank Zhenyu for his guidance and constructive advice on doing research and writing papers. I would like to thank Qinghua and Peng for sharing their technical expertise with me.

I want to thank Serge and Kavé for helping me find accommodation in Paris, which made my life much easier. It is never easy to find an appropriate residence in Paris, especially for a foreigner who comes to the city for the first time.

I would like to thank my colleagues, Émilie, Luca, Andrea, Davide, Wei, Boris, Preechai, Yiu Quan, etc., in the LIP6-NPA team for their help on my life and research project. A special thanks to Émilie, you saved me many times from administrative paperwork. Of course, I will remember the "French apéro time" that we experienced together. It is a part of my nice memories of Paris.

I would like to thank my colleagues, Xian, Xinyi, Ye, Ru, Gerui, Donghui, etc., in the ICT/CAS team. The discussions with you have inspired me in both research and life.

I also want to thank Yanmei for providing me the opportunity to work at Alibaba as a research intern. I have learned so much from this journey.

Last but not least, I would like to thank my family members and my girlfriend who have supported me for years. You were the incentive that motivated me to put myself together when I was down.

After a long period of working on my Ph.D., I have finally arrived at the end. Although the process was not easy at all, luckily, I have been through these days with these helpful people. Therefore, the experience has been great. A special thanks to all of you again. I will miss these days.

Abstract

The transport performance in Mobile Internet is crucial to user experience and directly affects the revenue of providers for Internet applications and services. Measuring the transport characteristics of Mobile Internet and implementing efficient transport mechanisms are essential for improving transport performance. This dissertation focuses on network access and congestion control, two key factors affecting the transport performance in Mobile Internet, and conducts research from three aspects: network access measurement, single-path congestion control, and multipath congestion control. The main innovative work is as follows.

WiFi performance measurement in mobile scenarios: WiFi is widely deployed in cities. Measuring and analyzing WiFi performance in mobile scenarios underpins the optimization of network system design. We designed and implemented a WiFi performance measurement and analysis system based on in-vehicle devices to measure and analyze transport characteristics such as link connectivity, throughput, delay, and packet loss, and carried out measurement experiments in four typical cities including Los Angeles, Macao, Bologna, and Paris. The measurement results include connectivity availability, transport performance, network characteristics, and factors correlated with these metrics. Based on the results, optimization mechanisms such as access point selection and transport protocols are discussed.

An enhanced BBRv2 congestion control algorithm based on delay information: To address the dynamics of network quality such as high packet loss rate, high delay jitter, and fluctuation of available bandwidth in Mobile Internet, an enhanced BBRv2 congestion control algorithm based on delay information is designed and implemented. The algorithm preserves BBRv2's advantages in fairness, improves BBRv2's bandwidth probing mechanism with delay information, and adopts a window compensation mechanism based on jitter to handle the performance degradation caused by high jitter. The experimental results show that the proposed algorithm can improve the throughput by 25% compared with BBRv2 in mobile scenarios.

A data-driven multipath congestion control algorithm selection mechanism: To address the heterogeneity and dynamics of multipath network environments, a data-driven multipath congestion control algorithm selection mechanism is designed and implemented.

The mechanism is implemented based on a lightweight multipath congestion control algorithm selection framework. It mainly uses Machine Learning tools to capture the correlation between network environments and suitable congestion control algorithms. The mechanism dynamically selects the suitable congestion control algorithm for different subflows at runtime, aiming at enhancing transport performance. The experimental results show that compared with state-of-the-art multipath congestion control schemes, the proposed mechanism can improve the average throughput by 19% – 25%, and reduce the average queuing delay by up to 21%.

Keywords: Mobile Internet, Network measurement, Congestion control, BBR, Multipath transport, MPTCP

Résumé

Les performances de transport dans l'Internet mobile sont cruciales pour l'expérience utilisateur et affectent directement les revenus des fournisseurs d'applications et de services Internet. Mesurer les caractéristiques de transport de l'Internet mobile et mettre en œuvre des mécanismes de transport efficaces sont essentiels pour améliorer les performances de transport. Cette thèse se concentre sur l'accès au réseau et le contrôle de la congestion, deux facteurs clés affectant les performances de transport dans l'Internet mobile, et mène des recherches sous trois aspects : la mesure de l'accès au réseau, le contrôle de la congestion à trajet unique et le contrôle de la congestion à trajets multiples. Les principaux travaux innovants sont les suivants.

Mesure des performances WiFi dans les scénarios mobiles: Le WiFi est largement déployé dans les villes. La mesure et l'analyse des performances WiFi dans des scénarios mobiles sous-tendent l'optimisation de la conception du système de réseau. Nous avons conçu et mis en œuvre un système de mesure et d'analyse des performances WiFi basé sur des appareils embarqués pour mesurer et analyser les caractéristiques de transport telles que la connectivité de liaison, le débit, le retard et la perte de paquets, et avons réalisé des expériences de mesure dans quatre villes typiques, dont Los Angeles, Macao , Bologne et Paris. Les résultats de mesure incluent la disponibilité de la connectivité, les performances de transport, les caractéristiques du réseau et les facteurs corrélés à ces métriques. Sur la base des résultats, des mécanismes d'optimisation tels que la sélection du point d'accès et le protocole de transport sont discutés.

Un algorithme de contrôle de congestion BBRv2 amélioré basé sur les informations de retard: Pour répondre à la dynamique de la qualité du réseau, comme le taux de perte de paquets élevé, la gigue de retard élevée et la fluctuation de la bande passante disponible dans l'Internet mobile, un algorithme de contrôle de congestion BBRv2 amélioré basé sur les informations de retard sont conçues et mises en œuvre. L'algorithme préserve les avantages de BBRv2 en matière d'équité, améliore le mécanisme de détection de bande passante de BBRv2 avec des informations de retard et adopte un mécanisme de compensation de fenêtre basé sur la gigue pour gérer la dégradation des performances causée par une gigue élevée.

Les résultats expérimentaux montrent que l'algorithme proposé peut améliorer le débit de 25% par rapport à BBRv2 dans les scénarios mobiles.

Un mécanisme de sélection d'algorithme de contrôle de congestion multi-chemins basé sur les données: Pour répondre à l'hétérogénéité et à la dynamique des environnements de réseau multi-chemins, un mécanisme de sélection d'algorithme de contrôle de congestion multi-chemins basé sur les données est conçu et mis en œuvre. Le mécanisme est implémenté sur la base d'un cadre de sélection d'algorithme léger de contrôle de congestion par trajets multiples. Il utilise principalement des outils d'apprentissage automatique pour capturer la corrélation entre les environnements réseau et les algorithmes de contrôle de congestion appropriés. Le mécanisme sélectionne dynamiquement l'algorithme de contrôle de congestion approprié pour différents sous-flux au moment de l'exécution, dans le but d'améliorer les performances de transport. Les résultats expérimentaux montrent que, par rapport aux schémas de pointe de contrôle de la congestion par trajets multiples, le mécanisme proposé peut améliorer le débit moyen de 19% à 25% et réduire le délai moyen de mise en file d'attente jusqu'à 21%.

Mots-clefs: Internet mobile, Mesure du réseau, Contrôle de la congestion, BBR, Transport multivoie, MPTCP

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With the popularization of mobile broadband networks such as 4G and 5G in recent years, and the continuous development of the computing performance of mobile devices such as mobile phones, tablet computers, smart watches, and in-vehicle devices, Mobile Internet services and applications have been deeply integrated into the various aspects of human society. According to the statistics of investigation agencies [117], the current Mobile Internet traffic has occupied 56.89% of the total Internet traffic share, becoming the most important traffic contributor on the Internet. As an indispensable part of users' access to Mobile Internet services, network transmission will directly affect the quality of experience (QoE) of users, which in turn affects the revenue of the providers of Internet applications and services. For example, Akamai reports that every 100 *ms* increase in the delay of web page access will lead to a 7% decrease in the advertisement conversion rate of e-commerce platforms [6], and a Google survey shows that every 0.5 *s* increase in returning time of search results will cause them to lose 20% of the traffic [33]. Therefore, for the applications and services in Mobile Internet, it is very important to ensure efficient network transmission.

This paper mainly focuses on the two key parts of the transport mechanisms in Mobile Internet, namely, the network access and the congestion control of transport protocols. As shown in Figure 1.1, in order to access Internet applications, users must first select one or several networks to access the Internet from mobile devices, and then use transport protocols such as TCP and QUIC [62] to communicate with the application servers deployed by the application providers in their CDN (Content Delivery Network) or datacenters. Therefore, the choice of the network access affects the upper bound of the transport performance and the transmission cost. Then, under a specific network access, the key component in the transport

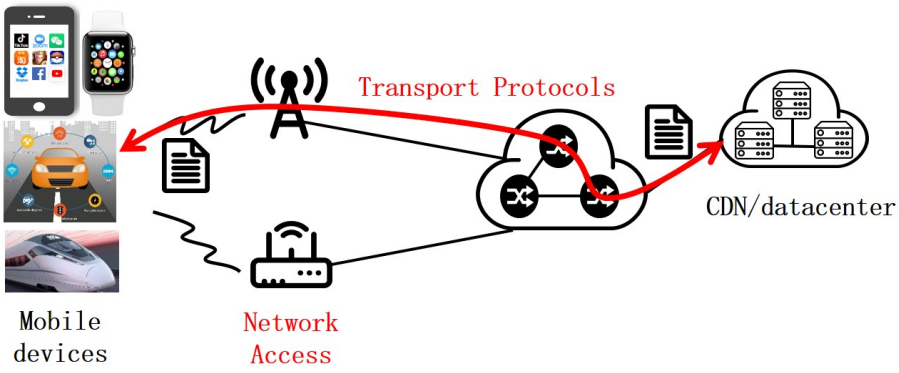


Figure 1.1 Transport mechanisms in Mobile Internet

protocols that determines the sending rate is congestion control, which affects whether the capacity of network can be fully utilized.

In Mobile Internet, there are currently two main types of networks that can be accessed, namely cellular networks including 4G and 5G, and WiFi networks. There is a big difference in the transmission capabilities provided by these two types of networks: **(1)** the wide coverage of the cellular network can provide high network availability and support user mobility in a large area. However, due to the complex infrastructure including base stations and core networks, and the high costs of infrastructure construction and operation, users are also required to pay according to the amount of traffic used; **(2)** the coverage of WiFi networks is usually limited, concentrated indoors or in urban areas, so the support for user mobility is poor. However, the infrastructure of WiFi is relatively simple, so the deployment and operation costs are low. Thus, users are generally not charged according to the amount of traffic. For users, the transmission cost is relatively low. For the selection of network access, there is no silver bullet to solve the problem once-for-all. Users should make full use of all transmission opportunities provided by all available network access types [26, 93]. Therefore, it is very valuable to measure and analyze the transmission capabilities of various network accesses, because the measurement results can provide important guidance for the design of network access strategies, transport protocols and congestion control algorithms (CCAs).

Due to the introduction of user mobility, no matter which network users choose to access, they may face complicated network dynamics, such as delay jitter, packet losses caused by base station handover, frequent changes in available network bandwidth caused by resource competition, and connectivity interruption caused by signal coverage holes. Therefore, in order to make full use of the transmission capacity of these networks, it is of great importance to implement efficient congestion control algorithms.

In addition, many current mobile devices are equipped with multiple network interfaces. For example, smartphones usually have a cellular network interface and a WiFi network interface, connected vehicles can communicate with Internet services through built-in cellular and WiFi network interfaces [72], and the WiFi gateway device on high-speed rails can provide Internet services to high-speed rail passengers via multiple cellular network interfaces [124]. When such devices are in an environment where multiple networks are available at the same time, multipath transport protocols can be used to aggregate bandwidth and improve reliability, thereby optimizing performance. In multipath transport protocols, it is also the congestion control algorithm that determines the transmission rate. However, compared with single-path transport protocols, multipath transport protocols face more complicated network environments: different paths may be heterogeneous, and the network dynamics on different paths may not be synchronized. These bring great challenges to designing efficient multipath congestion control algorithms. Therefore, it is also extremely important to conduct research on multipath congestion control.

1.2 Problems and challenges

In this dissertation, our research work focuses on three important aspects: network access measurement, single-path congestion control, and multipath congestion control in Mobile Internet. Specifically, the research problems to be addressed in this dissertation are as follows.

1) What kind of transmission capability can the in-situ WiFi networks provide in mobile scenarios?

There are many measurement studies in academia [64, 121, 124, 100, 131] showing that, including 5G, cellular networks often face some network dynamics caused by mobility, such as large delay jitter, high packet loss rate, and large bandwidth fluctuations, which prevent users to get consistent high transport performance, thus, in turn affecting QoE. In addition, due to the explosive growth of Mobile Internet traffic, the capacity of the cellular network itself is also under great pressure, and users are also faced with high traffic costs and restrictions on data usage [73, 151].

Currently, in many urban areas, public WiFi networks are readily available, inexpensive to use, and often without traffic restrictions. In this dissertation, such public WiFi networks are referred to as provider-managed WiFi networks. Therefore, a natural thought is whether these provider-managed WiFi networks could be used to complement cellular networks, thereby offering the possibility to improve transport performance and reduce transmission costs. For this reason, it is necessary to measure and analyze the availability, transmission capabilities, and network characteristics of network access based on the provider-managed

WiFi networks. However, the existing measurement studies [9, 13, 34, 116, 115] are mostly based on open WiFi networks[1] more than ten years ago. These open WiFi networks have almost disappeared [72, 103, 108], and are very different from provider-managed WiFi networks in terms of deployment strategies, connection establishment overhead, and transport performance. Therefore, the above measurement results do not reflect the current real situation of using public WiFi networks in mobile scenarios.

In order to fill the above gap, we need to conduct measurement study on the availability, transmission capabilities, and network characteristics of the connectivity based on provider-managed WiFi networks in mobile scenarios. However, due to the lack of measurement tools and the differences in the deployment of provider-managed WiFi networks in different cities, there are certain technical difficulties and it needs a lot of time and manpower to carry out the measurement experiments. How to properly design a measurement tool to avoid underestimating the transmission capabilities provided by provider-managed WiFi networks in mobile scenarios is a technical challenge that needs to be addressed.

2) How well does BBRv2 [18] adapt to the dynamics of network environments? If there is a problem, how can it be improved?

Since the Internet infrastructure is built on packet-switched networks based on statistical multiplexing, congestion control in transport protocols is necessary to avoid congestion collapse on the Internet [63]. Congestion control aims to efficiently utilize available network resources while ensuring that bottleneck bandwidth is fairly shared among multiple flows. However, efficient congestion control in Mobile Internet faces two challenges. First of all, the network quality in Mobile Internet is often very dynamic, which is reflected in large bandwidth changes, high delay jitter, and high packet loss rate. Therefore, congestion control algorithms need to be able to quickly adapt to the above situations. The traditional congestion control algorithms based on packet losses are difficult to meet the demand. Secondly, since there are a large number of traditional congestion control algorithms based on packet loss on the Internet, for the consideration of incremental deployment, any newly designed congestion control algorithm also needs to consider the fairness against traditional algorithms.

More than three decades of congestion control research have produced a large number of congestion control algorithms, including some specifically designed for wireless environments or cellular networks, such as Sprout [131], Westwood [19], Verus [146], and C2TCP [1]. Although the above algorithms provide better performance in cellular or wireless environments than traditional loss-based algorithms such as Cubic [107], due to the problem of fairness and strict assumptions about the network environment, they are hardly deployed on the Internet [89]. In recent years, with the development of Machine Learning (ML) tech-

[1] "Open" means no authentication required.

niques, many scholars have proposed learning-based congestion control algorithms, such as Remy [130], Aurora [65], PCC-Vivace [29], Indigo [138], and Orca [2]. Although this type of learning-based algorithms can quickly adapt to different network environments through self-learning, due to problems in computational overhead, fairness, and generalization ability in unseen network environments [2, 32, 152], they are not mature enough for large-scale deployment on the Internet.

In 2016, Google proposed a new congestion control algorithm BBR [16] (Bottleneck Bandwidth and Round Trip Propagation Time), which alternately measures the bottleneck bandwidth (*BtlBW*) and round trip propagation time (*RTprop*) and actively tries to run at Kleinrock's optimal operating point [69] to maximize throughput and minimize queuing delay. Since BBR abandons the idea of using packet loss as a congestion signal, it can also achieve good performance in a network environment with a high packet loss rate. Due to its advantages in packet loss resilience and low latency, BBR has been widely used in Mobile Internet scenarios [16, 89, 122]. Nevertheless, some recent measurement studies [14, 56, 70, 82, 111, 127] show that BBR has problems in fairness and in dealing with network jitter.

In order to solve these problems in BBR, Google proposed BBRv2 [18], and plans to gradually replace BBR and deploy it in various Google products [144]. Therefore, BBRv2 is promising to cope with the network dynamics in Mobile Internet. Since BBRv2 is still in its early stage, the related works [17, 44, 66, 90, 114] are only aimed at measuring whether it solves the problems of BBR. There is no comprehensive analysis about how BBRv2 deal with the network dynamics.

To fill the above gaps, we firstly conduct a comprehensive measurement analysis of BBRv2. Based on the measurement results of BBRv2, we find that although BBRv2 solves the fairness problem of BBR, it still has some defects, including low packet loss resilience, slow response speed to bandwidth changes, and throughput degradation when network jitter is large. We also find that a part of the defects is the cost of addressing the fairness issue. Then, how to optimize the defects of BBRv2 to make it suitable for Mobile Internet scenarios is a problem to be solved. The key technical challenge here is how to optimize the performance of BBRv2 while retaining its advantage in fairness.

3) How to design a congestion control algorithm selection mechanism to improve the performance of multipath congestion control in Mobile Internet?

In Mobile Internet, the heterogeneous and dynamically changing multipath network environment brings great challenges to multipath congestion control. Multipath congestion control algorithms not only need to be able to adapt to the network environment on each path

but also need to satisfy multipath fairness (i.e. when a multipath connection and a single-path connection share the same bottleneck link, they must be able to share the bandwidth fairly).

However, none of the existing MPTCP congestion control algorithms can cope with the above challenges well. First of all, although any congestion control algorithm in traditional TCP, such as Cubic [107], can be independently applied to the subflow of MPTCP, this type of uncoupled multipath congestion control algorithms can not achieve multipath fairness. Secondly, the classic coupled congestion control algorithms[105, 67, 102, 30, 15, 48, 119] are too conservative when paths do not share bottleneck links [36], and they also do not consider the problem of path heterogeneity. In order to solve the problem that the performance of the above coupled algorithms is too conservative in the case of non-shared bottleneck links, some scholars have proposed some dynamically coupled congestion control algorithms [36, 49, 51, 120, 128, 143]. But, these algorithms are still not carefully designed for the scenario of heterogeneous paths, and can not adapt to the heterogeneous and dynamically changing multipath network environment in Mobile Internet scenarios. In recent years, some researchers have also proposed some learning-based multipath congestion control algorithms [52, 76, 104, 135, 137, 145, 43] to adapt to different network environments. However, these algorithms generally have a series of problems preventing them to be deployed at scale in reality. For example, their ways to implement multipath fairness are either too conservative to cause performance degradation or not guaranteed at all, their performance experiences degradation in unseen network environments, they introduce large system overhead, the convergence speed of them is slow, and the deployment of them is complicated [145].

In terms of traditional single-path TCP congestion control, existing research work [2, 152] has shown that existing congestion control algorithms can often only achieve good transport performance in specific network environments that meet their design assumptions. Therefore, some scholars[20, 94, 152] propose that appropriately choosing congestion control algorithms for TCP connections by sensing network environments is able to improve TCP transport performance. Inspired by this research thread, we consider whether we can design a multipath congestion control algorithm selection mechanism to dynamically select appropriate CCAs for subflows while achieving multipath fairness.

To implement this mechanism, there are three technical challenges. First of all, there are many kinds of congestion control algorithms, and new congestion control algorithms are constantly being proposed. The mechanism must be extensible to facilitate the integration of new congestion control algorithms. In addition, from the perspective of deployability, the implementation overhead of this mechanism should not be too high, and it should be transparent to applications and compatible with legacy MPTCP applications. Finally, because the relationship between the data reflecting the network path quality and the appropriate

congestion control algorithm is very complicated, there is currently no clear guiding principle to explain what rules should be used to map the data to the corresponding CCA. Thus, it is also a technical challenge to select an appropriate congestion control algorithm based on the data of the network environment.

1.3 Research contents

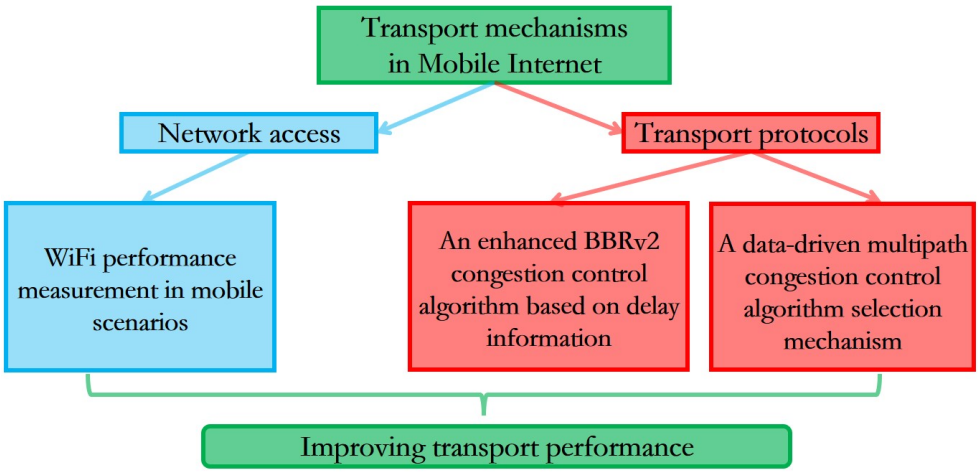


Figure 1.2 The framework of research contents

As shown in Figure 1.2, this dissertation mainly focuses on the important problems and technical challenges in the three aspects of network access measurement, single-path congestion control, and multipath congestion control in Mobile Internet. Specifically, the research contents are as follows.

1) WiFi performance measurement in mobile scenarios

In this dissertation, a measurement system is designed, including: **(1)** a WiFi client that quickly completes the connection establishment to avoid wasting the transmission opportunities offered by provider-managed WiFi networks as much as possible. The client adopts a series of best practice heuristics to accelerate the WiFi association process and DHCP process. Compared with the current open source *wpa_supplicant+dhcpcd*, the client shortens the 90% percentile of the time of WiFi association and DHCP process from 10 *s* to 2 *s*; **(2)** a TCP performance measurement tool. Then, the measurement system is deployed on in-vehicle devices and the measurement experiments are conducted in four typical cities in Europe, Asia and North America, accumulating the measurement data of about 5620 *km* of moving mileage, including tens of thousands of WiFi associations.

Based on the above dataset, the transmission capability provided by provider-managed WiFi networks for mobile users is analyzed in detail from various aspects. The main measurement results and some insights are as follows: **(1)** compared with the measurement results based on open WiFi network [9, 13, 34, 116, 115], the availability of provider-managed WiFi networks does not change significantly. The average duration of connectivity and connectivity holes is around 16 – 29 *s* and 5 – 112 *s* respectively; **(2)** the TCP goodput ($\sim$1 GB/h) offered by provider-managed WiFi networks is an order of magnitude higher than most of the measurement results in open WiFi networks [9, 13, 34, 116, 115]. The average goodput during WiFi connection is 2.97 – 13.36 Mbps (excluding Paris); **(3)** compared with that of open WiFi networks, the connectivity setup overhead of provider-managed WiFi networks is higher, however, it is only a small fraction of the entire WiFi association duration (10% – 18%); **(4)** in terms of AP selection, the existing strategy solely based on signal strength is sub-optimal; as 5 GHz APs usually provide better performance than 2.4 GHz APs in this scenario, from the perspective of AP selection, 5 GHz AP can be given priority; **(5)** provider-managed WiFi networks usually do not guarantee that the IP address can remain unchanged across different APs. Therefore, from the perspective of applications and transport protocols, the ability to support IP handover is necessary; **(6)** during WiFi associations, high latency jitter, high packet loss rates, and large bandwidth changes are likely to occur; **(7)** in terms of congestion control, BBR [16] that provides high loss resilience does not show better transport performance than Cubic [107], the main reason may be high delay jitter affects the transport performance of BBR; **(8)** concurrent TCP flows can provide better goodput than a single TCP flow.

2) An enhanced BBRv2 congestion control algorithm based on delay information

We first measure BBRv2 from various aspects under various network conditions and compare it with BBR. It is found that the improvement of BBRv2 in fairness will bring some additional costs, including low resilience to random packet losses and the response speed to bandwidth changes is slow, which is not conducive to its good performance in the network environment in some Mobile Internet scenarios. In addition, BBRv2 still suffers from high jitter, which can degrade transport performance. For the problem of low packet loss resilience, the measurement results show that the problem can be effectively mitigated by carefully adjusting the packet loss threshold parameter in BBRv2 without losing its fairness advantage.

To address the problems of the responsiveness to bandwidth dynamics and the resilience to jitter, we propose a new congestion control algorithm BBRv2+ based on BBRv2. First, BBRv2+ integrates delay information into its path model, using delay information as feedback from the network environment to guide its aggressiveness in bandwidth probing. Second,

BBRv2+ partially refactors the BBRv2 state machine to use delay information to guide the aggressiveness of BBRv2+ bandwidth probing. By doing this, BBRv2+ strikes a balance between the aggressiveness of bandwidth probing and fairness to loss-based congestion control algorithms. Third, to avoid bandwidth starvation problems caused by the use of delay information when coexisting with loss-based congestion control algorithms, BBRv2+ uses a dual-mode mechanism: it decides whether to enable the new bandwidth probing mechanism based on if there are loss-based CCAs co-exist. Finally, for the issue of jitter resilience, BBRv2+ compensates the Bandwidth Delay Product (BDP) estimation based on the observed delay jitter to reduce the misleading effect of delay jitter on congestion window size so that it can adapt to the network environment with high jitter.

The experimental results show that, compared with BBRv2, BBRv2+ successfully balances the aggressiveness of bandwidth probing and the fairness to loss-based congestion control algorithms and improves the resilience to jitter. Compared with BBRv2, BBRv2+ can achieve a throughput improvement of about 25% in mobile scenarios, and the queuing delay is close to that of BBRv2.

3) A data-driven multipath congestion control algorithm selection mechanism

We design the MPTCP-SSCC (MPTCP Smart Selection of Congestion Control) system based on the de-facto multipath transport protocol, MPTCP [38] (Multipath TCP), to implement a data-driven multipath congestion control algorithm selection mechanism. MPTCP-SSCC continuously perceives the quality and bottleneck link sharing situation of subflows in kernel space, collects data from MPTCP kernel in user space, and then uses a data-driven approach to dynamically adjust the congestion control algorithm for each subflow in user space. This design of the congestion control algorithm selection in user space makes the system have good extensibility.

In order to be compatible with legacy MPTCP applications, efficiently transfer data between kernel and user space, and adjust the congestion control algorithm of each subflow at runtime, we design and implement a lightweight congestion control algorithm selection framework based on eBPF [11, 60] (extended Berkeley Packet Filter). Due to the complex relationship between the network environment data and the suitable congestion control algorithm, it is difficult to map network environment data to a specific CCA with a limited number of human-designed rules. Therefore, the congestion control algorithm selection is abstracted as a classification task. We use the XGBoost [133] algorithm to train an ML model that is combined with some fixed rules to accomplish the CCA selection task in the MPTCP-SSCC system.

The experimental evaluation of MPTCP-SSCC in emulated and real environments shows that the system can dynamically select an appropriate congestion control algorithm for

each subflow and satisfy multipath fairness according to the network conditions of each path. Compared with some mainstream coupled CCAs, such as LIA [105], OLIA [67], BALIA [102], and wVegas [15], or a state-of-the-art learning-based multipath congestion control algorithm, MPCC [43], MPTCP-SSCC system can improve 19% – 25% average throughput and reduce average queuing delay by up to 21%.

1.4 Contributions

This paper makes innovative contributions in three aspects of the Mobile Internet transport mechanism.

1) Network access measurement in Mobile Internet

The core contributions of this part are as follows. We designed and implemented a measurement and analysis system to measure the transmission capabilities provided by provider-managed WiFi networks in mobile scenarios. We conducted measurement experiments in four typical cities to collect data and analyzed the availability, transport performance, and network characteristics of the connectivity based on provider-managed WiFi networks. In terms of optimization, we provide some insights for AP selection, connection establishment process optimization, and the optimization of transport protocols and congestion control algorithms.

2) Single-path congestion control

The core contributions of this part are as follows. We conducted in-depth measurement and analysis of BBRv2, summarized its advantages and disadvantages compared with BBR, and explained the reasons behind the disadvantages. We propose BBRv2+ to mitigate the problems of BBRv2, which improves the transport performance of BBRv2 in mobile scenarios, and hardly sacrifices the advantages of BBRv2.

3) Multipath congestion control

The main contributions of this part are as follows. We designed and immplemented the MPTCP-SSCC system to realize the data-driven multipath congestion control algorithm selection mechanism. MPTCP-SSCC consists of two novel parts: **(1)** an eBPF-based lightweight multipath congestion control algorithm selection framework, which allows to continuously sense the network environment and transport status of subflows in an application-transparent manner in userspace and dynamically adjust the congestion control algorithm for subflows. The framework has good extensibility, which is convenient to integrate various congestion control algorithms; **(2)** based on the above framework, we use Machine Learning with a few artificial rules to capture the relationship between the network environment and the appropriate congestion control algorithm. With this design, we are able to dynamically

select the suitable CCA for subflows to improve transport performance and achieve multipath fairness.

1.5 Struture of chapters

The rest of this dissertation is organized as follows.

Chapter 2 summarizes the background knowledge and related research work of network access measurement, single-path congestion control algorithms and multipath congestion control algorithms in Mobile Internet respectively.

Chapter 3 elaborates on WiFi performance measurement in mobile scenarios. First, the background of this study is introduced. Then, the design of the measurement system is explained. Next, the measurement methodology is described. After that, the summary of the collected dataset is outlined. Thereafter, a detailed analysis is carried out based on the dataset, and the implications of the measurement results are elaborated. Finally, the chapter is concluded.

Chapter 4 focuses on the enhanced BBRv2 congestion control algorithm based on delay information. First, the background and an overview of this chapter is presented. Then, the principles and mechanisms of BBR and BBRv2 are elaborated and analyzed in detail. Next, an in-depth measurement analysis of BBRv2 is presented. Afterwards, the design and implementation of BBRv2+ are described, along with a detailed experimental evaluation of BBRv2+. Finally, the content of this chapter is concluded.

Chapter 5 introduces the data-driven multipath CCA selection mechanism. First, the background and an overview of this part are introduced. Then, the motivation and the technical challenges are analyzed in detail. After that, the core idea of our solution is introduced. Next, the design and implementation of the MPTCP-SSCC system are introduced. Thereafter, the experimental evaluation of the MPTCP-SSCC system is presented. Finally, the content of this chapter is concluded.

Chapter 6 concludes this dissertation and briefly introduces the future research plan.

Chapter 2

State of the art

This chapter mainly introduces the background knowledge and the state quo of the three parts of the research content described in Chapter 1: network access measurement, single-path CCA, and multipath CCA in Mobile Internet.

2.1 Network access measurement in Mobile Internet

There are two main types of network access in Mobile Internet, namely, cellular network access and WiFi network access. In stationary or quasi-stationary scenarios, a large number of scholars [21, 25, 57, 91, 95, 136] have carried out measurement research on these two types of network access. Therefore, in this dissertation, we mainly focus on performance measurement of network access in mobile scenarios.

2.1.1 Cellular access measurement in mobile scenarios

Measuring the transport performance of cellular networks in mobile scenarios has always been a research hotspot. Litjens [78] measured the performance of the UMTS/HSDPA network in scenarios involving terminal mobility and found that transport performance does not simply monotonically decrease as moving speed increases. Yao et al. [142] measured the bandwidth of the HSDPA network in the scenario of vehicular mobility, and found that the network bandwidth is less predictable in the temporal dimension, but has a certain predictability in the spatial dimension. Jang et al. [64] measured the 3G/3.5G cellular network in Hong Kong at the moving speed of cars and high-speed rails of 300 km/h, and found that the transport performance was much lower than that of stationary situations. In addition, spurious retransmissions, delay jitter, and the degree of bandwidth variation are also higher than stationary cases. Tso et al. [121], measured the performance of HSDPA

network at different degrees of mobile speed, and found that mobility would have a great impact on network service quality, even interrupting service. At the same time, they also observed the nonmonotonic relationship between transport performance and moving speed. Merz et al. [86] measured the physical layer performance of LTE network at the moving speed of 100 – 200 km/h, and found that the moving speed will lead to a certain degree of performance degradation. But it is possible to mitigate negative effects by reasonably setting the signal-to-noise coverage. Xiao et al. [134] measured the transport performance of TCP in the LTE network on the Beijing-Tianjin intercity high-speed rail in China, and found that the throughput and RTT (Round Trip Time) were significantly negatively affected, and the fluctuation was higher than that of stationary or driving situations. Liu et al. [79] measured and analyzed the TCP transport performance of 3G/4G networks in the case of high-speed rail mobility, and found that the packet loss rate during timeout retransmission is very high, and the ACK packet loss rate can reach 0.66% leading spurious timeouts to increase. Li et al. [75] also conducted long-term measurements of TCP performance in 3G/4G networks in high-speed rail scenarios, and found that mobility leads to significant packet loss and RTT increases, and greatly reduces throughput. Li et al. [74] measured the TCP performance of cellular networks of multiple operators at the same time in the high-speed rail scenario, and found that the transmission quality degradation between different operators' networks is likely to be asynchronous. Wang et al. [124] measured the LTE network in the high-speed rail scenario and found that the failure of base station handover will have a significant impact on TCP performance, and this impact also has a certain degree of retention. In additioon, high loss rate, high delay jitter, and frequent bandwidth changes are also common. Recently, Pan et al. [100] conducted a long-term measurement study on 5G networks in high-speed rail scenarios. Compared with 4G networks, although 5G increases throughput and reduces access layer latency, the negative impact of base station handover on transport performance is more significant. In addition, there will still be network conditions with high packet loss rate, delay jitter, and large bandwidth changes.

The above research shows that mobility will inevitably cause various issues related to network dynamics, such as high packet loss rate, high latency jitter, connection interruption, and large bandwidth changes. Therefore, it is impossible to rely solely on cellular networks to get consistent high transport performance. It is a promising research direction to use all available network transmission opportunities as much as possible [93] to compensate each other for the performance loss when their own network quality declines.

2.1.2 WiFi access measurement in mobile scenarios

The performance measurement of WiFi in mobile scenarios can be traced back to the Drive-thru project [97]. In this study, the authors placed several APs on the side of the road and then measured whether they can associate with the APs from a moving car and whether TCP or UDP transmission is possible. According to their results: **(1)** the coverage supported by a single AP is quite large (more than 10 s of connection time even when moving at 180 km/h); **(2)** the quality of connections is good in the middle of the connection but degrades at the fringes; **(3)** it is able to transmit data to some extent via UDP and TCP, but the transport performance is worst than that of stationary cases. Since then, Gass et al. [40] and Hadaller et al. [47] also conducted similar measurements in a controlled environment, focusing on the impact of backhaul network quality on the overall transport performance. In addition, Hadaller et al. [47] also pointed out some performance problems in the WiFi protocol stack, and provide some practical guidelines on how to optimize the WiFi protocol stack for mobile scenarios. Mahajan et al. [83] analyzed the characteristics of the link layer of this type of WiFi connections through a self-built small-scale testbed, and found that there are often "bad" periods of poor quality during the connection and it is difficult to predict these periods in the temporal dimension. The above measurement research mainly focuses on the wireless access part.

Another category of measurement studies focuses on the availability and performance of transmissions in mobile scenarios using open WiFi networks already deployed in cities. Typical representatives of this type of research include [9, 13, 116, 34, 115]. According to these measurement studies, the median connection duration that mobile users can obtain each time and the median duration of connection holes are in seconds or tens of seconds level, and the long-term throughput is generally tens of MB per hour. However, currently, open WiFi networks have almost disappeared [72, 103], and they are replaced by provider-managed WiFi networks, which are very different from open WiFi networks in terms of deployment strategies and AP capabilities. Therefore, the conclusions of the previous measurement studies can not represent the current situation of using provider-managed WiFi networks in mobile scenarios.

There are also a few recent research works based on provider-managed WiFi networks [27, 72]. However, their focus is not on measuring the transmission capabilities provided by provider-managed WiFi networks. Thus, only a few metrics are analyzed.

2.1.3 Summary

Fully considering and leveraging available network access opportunities has the potential to opportunistically compensate for the performance problems of cellular networks due to mobility. In urban areas, the in-situ provider-managed WiFi network is one opportunity to consider. But so far, there is no measurement work to measure the availability, transport performance, and network characteristics of such connectivity offered by provider-managed WiFi networks in mobile scenarios.

2.2 Single-path CCAs

Congestion control of transport protocols was introduced in the 1980s. It was originally designed to allow distributed senders to fairly share network resources and avoid the performance of the Internet that is based on statistical multiplexing packet-switched networks to collapse due to congestion [63]. The early congestion control algorithms follow the principle of "Additive Increase Multiplicative Decrease" (AIMD), taking packet loss as the congestion signal and adjusting the congestion window according to Equation (2.1), where $W(t)$ represents the congestion window (in packets) at time t, $\alpha > 0$, and $0 < \beta < 1$. For example, for the most classic Reno [98], α is 1 in the slow start phase (Slow Start) and $1/W(t)$ in the congestion avoidance phase (Congestion Avoidance), and β is $1/2$.

$$W(t) = \begin{cases} W(t) + \alpha, \ if\ a\ packet\ is\ ACked \\ W(t) \times \beta, \ if\ a\ loss\ happens \end{cases} \tag{2.1}$$

With the continuous evolution of network infrastructures, relatively simple early congestion control algorithms such as Reno are gradually unable to fully utilize the available network resources. Therefore, new congestion control algorithms are proposed to adapt to new network environments and to improve transport performance. Among them, the most typical example is Cubic [107], which no longer uses the additive method to increase the congestion window, but uses the cubic function defined by Equation (2.2) to increase the congestion window, where C is a predefined constant, β is the MD (Multiplicative Decrease) coefficient, T_{loss} is the time of the last packet loss, and W_{max} is the congestion window before the last packet loss. This window growth approach enables Cubic to have good scalability and to achieve good bandwidth utilization in high-speed networks. Therefore, Cubic is also adopted as the default congestion control algorithm in Linux, and has become the most widely used congestion control algorithm [89] on the Internet.

$$W(t) = C(t - T_{loss} - \sqrt[3]{\beta \cdot W_{max}/C})^3 + W_{max} \tag{2.2}$$

With the popularization of various wireless broadband networks and the start of the era of Mobile Internet, such wireless networks with dynamic changes in bandwidth, delay, and packet loss pose new challenges to congestion control algorithms. Therefore, researchers have also begun to design the corresponding congestion control algorithms for wireless environments, which will be introduced in § 2.2.1. Recently, due to the prosperity of Machine Learning (ML), some research works have considered using ML methods for congestion control, hoping to make CCAs adapt to network environments, instead of designing specific CCAs for different environments. § 2.2.2 will introduce the status quo of learning-based congestion control algorithms. In addition, § 2.2.3 will introduce the related work of a relatively special congestion control algorithm, BBR [16]. Because BBR can tolerate high packet loss rates and actively avoid queuing to mitigate bufferbloat [41] to reduce queuing delay, it has been widely used in Mobile Internet [16, 89, 122].

2.2.1 CCAs for wireless networks

In a wireless environment, congestion is not the only reason for packet loss. Radio frequency interference, channel errors, and short-term signal interruptions can also lead to packet loss. Usually, this type of packet loss is called random loss, differentiated from congestion loss. Ideally, when encountering random packet loss, the transport protocol only needs to recover the lost packets, and should not actively reduce the sending rate [4]. However, traditional loss-based CCAs, such as Reno and Cubic, are designed for wired networks. They assume that all packet losses are due to congestion, so when random packet loss happens, the congestion window will also be reduced, causing performance degradation.

In order to solve the above problems, Mascolo et al. [19] designed the Westwood algorithm. The core idea is that when packet loss is detected, the sender does not blindly decrease the congestion window according to the MD method, but sets the window to the BDP calculated from the recently measured bandwidth, B. The calculation method of B is shown in Equation (2.3), where B^{-1} is the last estimated B, Δ is the time interval between the current ACK and the last ACK, $\alpha(\Delta)$ is the coefficient function with Δ as the parameter, d represents the data amount of this ACK, and b and b^{-1} represents the bandwidth samples calculated at the current and previous ACK. Because in some environments, there may be severe ACK compression [147], Westwood's bandwidth estimation method will lead to over-estimation. Grieco et al. [46] proposed the Westwood+ algorithm alleviates this problem. Westwood+ uses the amount of confirmed data in a complete RTT to calculate the

bandwidth sample b, and estimates the bandwidth B according to the exponential moving average. The specific calculation method is shown in Equation (2.4), where d represents the amount of data confirmed in one RTT, and α is the smoothing coefficient. For the same motivation as Westwood+, Wang et al. proposed TCP CRB [126] and TCP ABSE [125] to revise Westwood's bandwidth estimation method. In TCP CRB, the author will calculate two kinds of bandwidth estimates, in which the short-term estimates are calculated in the Westwood way, and the long-term estimates are calculated in a similar way to Westwood+, but the predefined time interval T is used instead of Westwood+. Then, when packet loss occurs, if the difference between the BDP calculated based on the long-term estimate and the current congestion window is too large, the TCP CRB will use the short-term bandwidth estimate, otherwise, the long-term estimate will be used. TCP ABSE improves TCP CRB. The author adapts the bandwidth sampling interval T, as shown in Equation (2.5), where VE is the quotient of the congestion window and the minimum RTT, RE is similar to B in Westwood+, and T_{min} is a predefined minimum sampling interval. In addition, TCP ABSE also adapts the smoothing coefficient α for calculating the exponential moving average to react to frequent changes in available bandwidth faster. Yang et al. [140] found that Westwood's packet loss recovery strategy has performance problems when there are many random packet losses, so they propose a more aggressive retransmission recovery strategy when random packet loss is detected. Since Westwood and the above algorithms do not use the MD window reduction method, there is a problem in fairness when coexisting with Reno. For this reason, Shimonishi et al. [112] proposed TCPW BBE to improve the fairness of Westwood. The idea is to add a window reduction coefficient in Westwood's window reduction method, and the coefficient is negatively related to queuing delay. Although all of the above Westwood-like algorithms can effectively improve the transport performance in networks with random packet loss, they also have some problems. For instance, they still use packet loss as the congestion signal, which leads to bufferbloat, and the congestion window growth of AI (Additive Increase) is too slow in Long Fat Networks.

$$\begin{gathered} b = d/\Delta \\ B = \alpha(\Delta) \cdot B^{-1} + [1 - \alpha(\Delta)] \cdot \left(\frac{b + b^{-1}}{2} \right) \end{gathered} \tag{2.3}$$

$$\begin{gathered} b = d/RTT \\ B = \alpha \cdot B^{-1} + (1 - \alpha) \cdot b \end{gathered} \tag{2.4}$$

$$T = \max \left(T_{min}, \frac{RTT \cdot (VE - RE)}{VE} \right) \tag{2.5}$$

In Mobile Internet, there are many interactive applications that require high throughput and low latency, such as real-time audio and video applications such as Skype, Zoom, and live streaming. Their requirements on network transmission pose great challenges to how to design CCAs in cellular networks. Cellular networks have certain particularities. Due to the high retransmission cost, the base station usually maintains a separate buffer for each user to reduce packet loss, and, especially in the scenario where the user moves, the bandwidth of the cellular network changes rapidly [131]. These characteristics make loss-based congestion control algorithms difficult to achieve high performance because they cause bufferbloat and are relatively slow to react to bandwidth changes. To address these issues, Winstein et al. [131] proposed Sprout. Sprout is an active congestion control algorithm that predicts the amount of data that the sender can send without leading the queuing delay to exceed a certain threshold. It calculate the prediction by observing the arrival time of packets at the receiver. Experimental results show that Sprout improves throughput by 2.2× compared to Skype, while reducing latency by 7.9×. However, Sprout has some limitations as follows. First, Sprout makes strong assumptions about the network environment. It requires that the bottleneck link of the network must be the wireless access to the cellular base station and there are no other background flows. Second, it requires the application to have enough data to send and does not support on-off mode transfers. Also, intra-protocol fairness among Sprout flows is not verified. These limitations result in very limited applicable scenarios for Sprout.

Zaki et al. [146] believe that it is very difficult to accurately predict the bandwidth on the wireless access part of a cellular network, because the complex state transition between the base station and the user will affect the channel availability, the packet scheduling strategy in the cellular network will cause the traffic to appear bursty mode, competition for base station resources by different users also affects the available bandwidth, and mobility amplifies the impact of these factors. For the above reasons, Zaki et al. designed the Verus algorithm that does not rely on channel quality prediction, and the goal is to achieve high throughput and low latency transport in cellular networks. Verus is a delay-based congestion control algorithm based on the AIMD framework, replacing AI with a delay-based function. Specifically, it continuously captures the relationship between the end-to-end delay and the congestion window, and then determines how the congestion window changes based on this. In addition, when encountering packet loss, Verus, similar to loss-based algorithms, performs MD window reduction to quickly react to congestion signals. Experimental results show that Verus achieves a throughput improvement of up to 30% with slightly higher latency than Sprout. However, Verus also has strong assumptions about the network environment, that is, the random packet loss rate of the network should be low enough, otherwise, Verus will

frequently respond to random packet loss in the way of MD window reduction, which will greatly reduce throughput.

Abbasloo et al. [1] also proposed a congestion control algorithm, C2TCP, for cellular networks. Unlike Sprout and Verus, C2TCP does not require any complex bandwidth prediction, network state analysis, and rate adjustment mechanisms. C2TCP is designed on the algorithm based on packet loss. The core idea is to maintain a virtual Active Queue Management (AQM) module on the sender side to judge the state of the network, and then to tune the congestion window of the basic algorithm based on this. Specifically, C2TCP will set an explicit delay target according to the network delay information, and then the virtual AQM will judge whether the current network state is good (RTT is lower than the delay target), medium (RTT slightly exceeds the delay target), or bad (RTT continuously exceeds the delay target) according to the comparison between the current RTT and the delay target. The window correction operations corresponding to these states are increasing the congestion window, maintaining the congestion window, and resetting the congestion window to the minimum value (similar to handling timeout). Experiments show that C2TCP trades for lower latency than Sprout and Verus at the cost of lower throughput. However, similar to Sprout and Verus, C2TCP also has some strong assumptions about the network environment. For example, the bottleneck of the network must be on the cellular wireless access side, the path delay must be low enough, and the random packet loss rate must be low enough.

2.2.2 Learning-based CCAs

Considering the congestion control algorithms designed for cellular networks, Sprout, Verus, and C2TCP, it can be seen that there is a trade-off between the performance and generality when designing congestion control algorithms for new network environments. In order to make CCAs have better adaptability, some scholars have introduced some ML tools and methods to design CCAs.

Winstein et al. [130] proposed a congestion control algorithm framework, Remy, based on offline learning. Remy accepts network characteristics, traffic models, and performance objective functions as input, and then uses brute force search to find the most appropriate mapping relationship between the sender's current state and congestion control actions through simulation to optimize the objective function. Dong et al. [29] proposed PCC-Vivace based on online learning, which uses a delay-sensitive utility function framework and an online learning algorithm based on gradient ascent to adjust the transmission speed. Yan et al. [138] proposed Indigo, which uses a Long Short Term Memory (LSTM) neural network to store the relationship between the sender's current state and congestion control actions, and uses Imitation Learning to train the model. Jay et al. [65] proposed the Aurora algorithm,

which uses Deep Reinforcement Learning (DRL) to generate policies that map network states to congestion control actions. Li et al. [77] proposed the AUTO algorithm based on Multi-Objective Reinforcement Learning to improve the generalization ability of the model to different environments, and allow the application to express the performance preferences such as more throughput-oriented or more delay-oriented.

Abbasloo et al. [2] proposed a hybrid learning-based congestion control algorithm, Orca. Orca divides the connection into multiple time slices (control cycles). At the beginning of each time slice, the model trained by DRL is used to predict the basic congestion window value. During the time slice, Orca uses the congestion window adjustment mechanism of a traditional congestion control algorithm (Cubic) to tune the underlying congestion control window. Due to the introduction of the Cubic-based window adjustment mechanism, which adds some disturbance, Orca is easier to get out of some locally optimal equilibrium states than pure learning-based congestion control algorithms. On the basis of Orca, Abbasloo et al. [3] designed the DeepCC framework, which can integrate other types of traditional congestion control algorithms, such as BBR, and can specify the target delay by the application to adjust the performance preferences. Du et al. [32] also proposed a hybrid algorithm, Libra, which combines traditional algorithms and DRL. The differences between Libra and Orca are that Libra is based on the sending rate instead of the congestion window, it uses a different reward function, and it chooses the maximum between the rate calculated by the traditional algorithm and the rate predicted by DRL as the base sending rate at the start of the control cycle.

Although the above algorithms can achieve good performance in scenarios with a certain similarity to the training environment, they also have some general problems, such as fairness problems, slow convergence speed and degraded performance in unseen network environments, the complexity of implementation and deployment, large overhead, and etc. [2, 32, 139, 152]. As such, these algorithms are currently not mature enough to be practically deployed at scale.

2.2.3 Related work of BBR

BBR [16] is a model-based congestion control algorithm proposed by Google in 2016. Unlike traditional loss-based or delay-based congestion control algorithms, BBR does not use packet loss or delay as a signal to passively respond to congestion. Instead, BBR actively measures the BtlBW and RTprop of the network path, operating at the Kleinrock's best operation point [69] to maximize throughput while avoiding the accumulation of packet queues on bottleneck links. Since BtlBW and RTprop can not be measured at the same time, BBR uses a state machine-based method to measure BtlBW and RTprop alternately (for

more details of BBR's operation, refer to § 4.2.1). Since BBR does not respond passively to packet loss, it can also achieve good transport performance on links with some random packet loss. In addition, the implementation of BBR is not complicated, and it only needs to be implemented at the sender. At present, many transport protocols already support BBR. Due to the features of packet loss resilience, low latency, and easy-to-deploy, BBR is widely deployed on the Internet. According to a CCA census [89], BBR currently has a 22% deployment share of Alexa Top 20K websites, second only to Cubic's 36%, and, from the perspective of traffic amount, the BBR traffic on the Internet has exceeded 40%.

Since BBR [16] was released by Google in 2016, due to its promising performance reported in [16], it has attracted lots of researchers and practitioners to evaluate its performance under various network conditions. Hock et al. [56] conducted an extensive experimental evaluation of BBR at high link-speeds ($\geq$1Gbps). They found that BBR is unfair when sharing a bottleneck link with Cubic flows or BBR flows with different RTTs. In addition, they also found that BBR can lead to massive retransmissions when the bottleneck buffer is small. Ma et al. [82] observed that the RTT fairness issue of BBR exist even if the disparity of RTTs of different flows are relatively small, and, revealed the root cause of this issue. Scholz et al. [111] reproduced the results from Hock et al. [56] via the network emulation framework proposed by them. In particular, they found that BBR flows are always able to claim at least 35% of the total bandwidth in deep-buffered networks when competing with Cubic flows. To demystify the aforementioned phenomenon observed in [111], Ware et al. [127] modeled BBR's behaviors when it competes with loss-based CCAs. They found that BBR's share of link capacity only depends on the link capacity and delay, the bottleneck buffer size, and the number of BBR flows. Cao et al. [14] compared BBR with Cubic under various network conditions to understand what are the applicable scenarios for BBR. They identified that the scenarios where BBR outperforms Cubic are also the scenarios where BBR is unfair to Cubic. Kumar et al. [70] observed that the throughput of BBR collapses when network jitters are high in an experimental evaluation of BBR. A similar phenomenon was also observed by Wang et al. [124] in a TCP performance measurement campaign conducted on high-speed rails where the network connectivity is very dynamic.

The issues of BBR identified by these empirical studies motivated many researchers to optimize the performance of BBR from various perspectives. For instance, [70, 124] proposed several modifications in the RTprop estimation of BBR to counter network jitters, [68, 82, 141] enhanced BBR's RTT fairness, [113, 149] improved BBR's inter-protocol fairness with loss-based CCAs, and [84] reduced BBR's aggressiveness in shallow-buffered networks to limit its retranmissions.

To solve the problems of BBR disclosed by the experiences from both academia and industry, Google proposed an upgrade of BBR, BBRv2 [18]. In Google's early tests [17, 18], compared with BBR, BBRv2 showed better inter-protocol fairness with loss-based CCAs and reduced retransmissions in shallow-buffered networks. The tests also showed that BBRv2's resilience to random losses is worse than that of BBR, but it is still better than that of Cubic. Besides the tests done by Google, there are also a few independent measurement studies from academia. Gomez et al. [44] and Nadagiri et al. [90] conducted experiments in emulated environments to study the inter-protocol fairness of BBRv2. Their experiments corroborated Google's results. The evaluation from Gomez et al. [44] also showed that BBRv2 is better than BBR in terms of RTT fairness. Kfoury et al. [66] conducted a more detailed emulation-based evaluation of BBRv2 for wired broadband networks. They also found that BBRv2 eliminates the massive retranmissions in shallow-buffered networks, and improves the inter-protocol fairness and the RTT fairness over BBR. Song et al. [114] compared BBRv2 with BBR to understand the pros and cons of BBRv2. In addition to the aforementioned findings from the previous studies, they found that BBRv2 can not quickly utilize free link capacity when the bottleneck bandwidth increases. However, they did neither disclose the root cause of this phenomenon nor provide means to solve this problem.

2.2.4 Summary

The network environment in Mobile Internet scenarios poses new challenges to congestion control algorithms. The current congestion control algorithms designed for wireless environments have strong assumptions about the network environment, resulting in very limited applicable scenarios. In addition, some algorithms also have fairness problems, which prevents them from being deployed at scale [89]. Although learning-based algorithms have achieved relatively good performance in some environments, these algorithms have some general problems, namely, fairness problems, slow convergence speed and low transport performance in unseen network environments, the complexity of implementation and deployment, and large overhead. These practical issues suggest that they are not yet mature enough for large-scale deployment. Although the new algorithm BBR proposed by Google has been widely used in practice, recent measurement studies have shown that it has issues such as fairness. For this reason, BBRv2 is proposed to optimize the problems of BBR. However, at present, the performance of BBRv2 in Mobile Internet scenarios where high latency jitter, high packet loss rate, and large bandwidth fluctuation are common is still unclear.

2.3 Multipath CCAs

In Mobile Internet scenarios, many mobile devices are usually equipped with multiple network interfaces. Therefore, multipath transport protocols can be used to improve transport performance. Congestion control is also essential in multipath transport protocols. For multipath congestion control, there are two main goals. First, multipath congestion control algorithms need to utilize the available resources on each path as much as possible to maximize transport performance. In addition, the multipath congestion control algorithm should also ensure multipath fairness. That is, when sharing the bottleneck link with a single-path connection, a multipath connection should be able to share the bandwidth with it fairly. In order to achieve the above two goals, a plethora of multipath congestion control algorithms have been proposed in academia. According to the method of realizing multipath fairness, they can be divided into two categories: coupled multipath CCA and dynamically coupled multipath CCA. In addition, ML methods have also been applied in multipath congestion control to improve the ability of multipath congestion control algorithms to adapt to different network environments. Next, the research progress of the three types of multipath congestion control algorithms is introduced respectively.

2.3.1 Coupled multipath CCAs

For the coupled congestion control algorithm, the way to achieve multipath fairness is also called "Network Fair [49]". That is, all multipath flows and single-path flows in the network need to share resources fairly. This strong fairness constraint guarantees that multipath and single-path flows can use bandwidth fairly on shared bottleneck links. The coupled congestion control algorithms in MPTCP are all based on "Network Fair", and their design follows three basic goals [105]: **(1)** "Improve Throughput", a multipath flow should at least provide the same throughput as a single-path flow when using the best path; **(2)** "Do No Harm", the obtained network bandwidth by a multipath flow should not exceed the bandwidth that would be obtained by a single-path flow using the best path, which ensures that the performance of single-path flows on the same bottleneck link with multipath flows will not be compromised under any circumstances; **(3)** "Balance Congestion", multipath flows should try to avoid sending traffic on the most congested paths.

LIA [105] is the earliest proposed multipath congestion control algorithm that satisfies the above three goals. It is based on the NewReno [54] algorithm that follows the AIMD principle, and couples the window growth factor of the AI stage, as shown in Equation (2.6), where W_* represents the congestion window of a subflow, τ_* represents the RTT of a subflow, and R represents the set of subflows.

$$W_r = W_r + \min\left(\frac{\max(W_r/\tau_r^2)}{[\sum_{k\in R}(W_k/\tau_k)]^2}, \frac{1}{W_r}\right) \tag{2.6}$$

Although LIA can provide multipath fairness, it may lead to non-Pareto-optimal problems [67]. That is, after upgrading a TCP flow in the network to a MPTCP flow, no flow will get gains, and some flows may even get penalty on their performance. To this end, Khalili et al. [67] analyzed the problems of LIA and found that in order to respond to changes in available bandwidth, LIA makes a trade-off between balancing congestion and responding to changes in bandwidth, which would lead to sending excessive probing traffic on congested paths. Based on the above analysis, they proposed the OLIA algorithm. The core idea of OLIA is to probe more on good paths and to probe less on poor paths. The specific changes of OLIA only involve the window growth function of the AI part of LIA, as shown in Equation (2.7), where B represents the set of subflows with the most amount of data transmitted between the last two packet losses, and M represents the set of subflows with the largest congestion window.

$$\begin{aligned} W_r &= W_r + \left(\frac{W_r/\tau_r^2}{[\sum_{k\in R}(W_k/\tau_k)]^2} + \frac{\alpha_r}{W_r}\right) \\ \alpha_r &= \begin{cases} \dfrac{1/|R|}{|B-M|} & if\ r \in B-M \neq \emptyset \\ -\dfrac{1/|R|}{|M|} & if\ r \in M\ and\ B-M \neq \emptyset \\ 0 & otherwise \end{cases} \end{aligned} \tag{2.7}$$

Peng et al. [102] found that OLIA has a slower response speed to changes in network bandwidth when paths experience similar RTTs. Therefore, the BALIA algorithm is proposed to achieve a better trade-off between response speed and balancing congestion. Based on theoretical analysis, they modeled the impact of algorithm design parameters on fairness and response speed, and then changed the window adjustment methods of NewReno's AI and MD stages, as shown in Equation (2.8) and Equation (2.9), where $\alpha_r = \max_{k\in R}(W_k/\tau_k)/(W_r/\tau_r)$. Melki et al. [85] based on the BALIA framework, adjusted the relevant parameters of the AI and MD stages to make the algorithm prioritize fairness.

$$W_r = W_r + \frac{W_r/\tau_r^2}{[\sum_{k\in R}(W_k/\tau_k)]^2} + \frac{\alpha_r}{W_r}\left(\frac{1+\alpha_r}{2}\right)\left(\frac{4+\alpha_r}{5}\right) \tag{2.8}$$

$$W_r = W_r - \frac{W_r}{2}\min(\alpha_r, 1.5) \tag{2.9}$$

Cao et al. [15] implemented a delay-based multipath CCA, wVegas, by coupling the delay-based Vegas [12] algorithm. wVegas can achieve more fine-grained load balancing than loss-based multipath congestion control algorithms, but it also faces the problem of Vegas, i.e. bandwidth starvation will happen when coexisting with loss-based algorithms. Han et al. [48] proposed Coupled-BBR that is based on BBR[16]. Specifically, Coupled-BBR modifies the pacing_gain of BBR in the cruise phase from a fixed 1.0 to Equation (2.10) to achieve multipath fairness. Dong et al. [30] coupled the Veno [22] algorithm to design mVeno to improve the performance of MPTCP in a wireless environment with random packet loss. Thomas et al. [119] proposed an eNMCC algorithm based on the window growth rate, which uses the window growth rate instead of the absolute rate to calculate the window adjustment coefficients for the AI and MD stages.

$$pacing\_gain_r = \frac{4\beta_r - 1}{3}, \quad \beta_r = \frac{BtlBW_r \cdot \max_{k \in R}(BltBW_k)}{\sum_{k \in R} BtlBW_k^2} \tag{2.10}$$

2.3.2 Dynamically coupled multipath CCAs

Coupled congestion control algorithms based on "Network Fair" always couple the congestion window of subflows even when there is no shared bottleneck link, resulting in a performance loss of up to about 50% [36]. Therefore, some scholars [36, 49, 51, 120, 128, 143] believe that the multipath fairness should be implmented based on the principle of "Bottleneck Fair" that only couples subflows across a common bottleneck link to improve throughput when subflows are independent.

Hassayoun et al. [49] proposed a DWC algorithm, which detects whether subflows share a bottleneck link through packet loss and delay signals, and then couples the subflows sharing a bottleneck link while allowing other subflows to grow their own congestion window independently. Based on the same idea, Ferlin et al. [36] designed MPTCP-SBD. MPTCP-SBD uses a method based on One Way Delay (OWD) statistical features to detect whether subflows share a bottleneck link, which can improve the accuracy of shared bottleneck link detection compared to DWC. Hayes et al. [51] tried to replace the threshold-based subflow grouping method in MPTCP-SBD with a clustering-based method in their follow-up work, but experiments show that these two methods have no absolute advantages and disadvantages. Ye et al. [143] and Wei et al. [128] both proposed the use of ECN (Explicit Congestion Notification) to achieve a more accurate shared bottleneck link detection. As ECN is mostly supported in proprietary networks, e.g. datacenter networks, it can not be used for multipath congestion control on the Internet. Thomas et al. [120] proposed the NMCC algorithm, which uses network topology information to detect shared bottleneck links. However, the

implementation of this algorithm requires Software Defined Networking (SDN) or Multiple Protocol Label Switching (MPLS). Thus NMCC is not applicable for multipath congestion control on the Internet.

Although the above algorithms achieve multipath fairness more flexibly and improve the performance of multipath congestion control in the case of non-shared bottleneck links, they still perform homogeneous congestion control actions on all paths, without considering the problem of path heterogeneity and the possibility of dynamic change of path quality.

2.3.3 Learning-based multipath CCAs

In order to be able to adapt to the heterogeneous and dynamically changing network environment, some research works have introduced tools and methods in the field of ML to design multipath congestion control algorithms. Gilad et al. [43] proposed a multipath congestion control algorithm, MPCC, based on the PCC [28] online learning framework, using a utility function that satisfies the LMMF (Lexicographic Max-Min Fairness) condition to achieve multipath fairness in "Network Fair" way. Experiments show that MPCC can achieve better performance than traditional algorithms in some highly dynamic environments. However, in short-flow scenarios, MPCC's performance may not be as good as traditional algorithms, because online learning requires a certain startup time. Li et al. [76] designed SmartCC, a multipath congestion control algorithm based on Reinforcement Learning (RL). SmartCC collects the transport performance data of all subflows in each control cycle, and then uses the RL model to predict the control parameters of the window adjustment on each subflow. In terms of model training and deployment, SmartCC adopts an asynchronous method. That is, the model is trained offline through the Q-Learning algorithm, and then the trained model is directly used in the online prediction stage. Xu et al. [137] proposed DRL-CC based on DRL to use a single agent to perform congestion control for all subflows of all MPTCP connections on a host. In order to solve the problem of dynamic changes in the number of MPTCP connections, DRL-CC uses LSTM to learn the state representation of all MPTCP connections, and integrates LSTM into the DDPG (Deep Deterministic Policy Gradient) framework to achieve end-to-end model training. Pokhrel et al. [104] proposed an algorithm DQL-MPTCP based on DQL (Deep Q-Learning) to jointly control congestion control and packet scheduling. He et al. [52] argue that using a single agent for the congestion control of MPTCP connections has some shortcomings, such as performance loss or the need to retrain the model when the number of subflows changes. To tackle the above issues, they proposed DeepCC based on MADRL (Multi-Agent DRL). Different from the previous work, DeepCC uses an agent independently for each subflow, and, in order to correlate with other subflows, DeepCC uses two self-attention mechanisms to relate the states and

reward functions of different subflows. Xu et al. [135] designed an RL-PSD algorithm for Augmented Reality (AR) or Virtual Reality (VR) scenarios. The main idea is to consider the Power Spectrum Density (PSD) of the video stream into the state in the RL-based algorithm. Yu et al. [145] designed a multipath congestion control algorithm, MPLibra, which combines DRL-based algorithms and traditional coupled algorithms, which can be regarded as the coupled version of the learning-based single-path congestion control algorithm, Libra [32]. Similar to MPCC, MPLibra also uses a utility function that satisfies the LMMF condition to achieve multipath fairness.

The above learning-based multipath congestion control algorithms can achieve good performance in scenarios that are similar to the training environment. However, similar to learning-based single-path congestion control algorithms, they also have some general problems, such as unable to achieve multipath fairness, performance degradation and slow convergence speed in unseen network environments, large system overhead, and complicated deployment in reality [145]. In addition, in order to achieve multipath fairness, some algorithms usually use a utility function that satisfies the LMMF condition to implement "Network Fair", which negatively affects the transport performance in the scenario of non-shared bottleneck links.

2.3.4 Summary

Although traditional coupled congestion control algorithms can guarantee multipath fairness, they may face a large performance loss in the case of non-shared bottleneck links, and their homogeneous congestion control actions on all paths can not adapt to heterogeneous and dynamically changing multipath network environments. The dynamically coupled congestion control algorithms can reduce the performance loss caused by coupling congestion windows of all subflows in the case of non-shared bottleneck links, but still do not consider the problem of path heterogeneity. Learning-based multipath congestion control algorithms have the potential to adapt to network environments, but they still have some general problems, which make it difficult to deploy them in production environments.

2.4 Conclusion

This chapter mainly introduces the background knowledge and the status quo related to network access mode measurement, single-path congestion control algorithm, and multipath congestion control algorithm in Mobile Internet, and summarizes some existing problems. In

the next three chapters, this dissertation will elaborate on the work that we have done for the three aspects.

Chapter 3

WiFi performance measurement in mobile scenarios

3.1 Introduction

In the era of Mobile Internet, users usually expect to enjoy good Internet services anytime, anywhere. The cellular network has a wide coverage and high network availability and is generally the first choice for users to access Internet services in mobile scenarios. However, only relying on the cellular network for data transmission cannot fully meet the requirements of transport performance, and there are still many problems. There are many related measurement studies [64, 121, 124, 136, 131] show that, including 5G, cellular networks in mobile scenarios often experience large delay jitter and many packet losses (retransmissions), and large bandwidth fluctuations, resulting in impaired user experience of Internet services. In addition, due to the explosive growth of mobile Internet traffic, the capacity of the cellular network itself is also under great pressure, resulting in the need for operators to invest a lot of costs in infrastructure construction and impose limits of data usage on users [73, 151].

In a real environment, there may be other networks available besides cellular networks. Some research work [93] indicates that users need to fully consider all available network access opportunities to improve transport performance in mobile scenarios. Currently, many operators deploy in-situ public WiFi networks in urban areas, and the economic cost of using these networks is low and usually has no restriction on data usage. For example, public WiFi built by Mobile Network Operators (MNOs) is generally free for the subscribers of MNOs. Therefore, this chapter considers whether these provider-managed WiFi networks can be

used to complement cellular networks in mobile scenarios, thereby offering the possibility to improve transport performance and reduce transmission costs.

Therefore, in mobile scenarios, it is very necessary to measure and analyze the availability, transmission capability, and network quality characteristics of the network access based on the above WiFi network. However, the existing related measurement research [9, 13, 34, 116, 115] are mostly based on open WiFi networks more than ten years ago, and these open WiFi networks have almost disappeared [72, 103, 108] and are very different from the current provider-managed WiFi networks. For example, provider-managed WiFi networks requires identity authentication and encrypts the channel. The authentication server and the DHCP server of provider-managed WiFi networks are usually deployed in core networks. Compared with open WiFi networks, additional connection establishment delay may be introduced. Moreover, with the upgrade of the 802.11 standard, today's access points (APs) should provide better transmission capabilities than APs in out-dated open WiFi networks. To sum up, there is no publicly available measurement data that can indicate the availability, transmission capability, and network quality characteristics of provider-managed WiFi networks in mobile scenarios.

To fill the above gap, we need to conduct large-scale measurement research. However, due to the lack of measurement tools and the differences between the deployment of provider-managed WiFi networks in different cities, there are certain technical difficulties and a lot of time and manpower to carry out the above measurement study. To this end, this chapter designs an automated measurement system based on in-vehicle devices, and then deploys and collects data in four typical cities in Europe, Asia, and North America. The collected measurement data feature about 5620 *km* of moving mileage and contain tens of thousands of WiFi associations.

Based on the collected dataset, we analyze in detail the transmission capability provided by provider-managed WiFi networks for mobile users from the aspects of connection availability, transport performance, network quality characteristics, and the related influencing factors of the above metrics. The main measurement results and observations are as follows: **(1)** compared with the measurement results based on open WiFi network [9, 13, 34, 116, 115], the availability of provider-managed WiFi networks does not change significantly. The average duration of connectivity and connectivity holes is around 16 – 29 *s* and 5 – 112 *s* respectively; **(2)** the TCP goodput ($\sim$1 GB/h) offered by provider-managed WiFi networks is an order of magnitude higher than most of the measurement results in open WiFi networks [9, 13, 34, 116, 115]. The average goodput during WiFi connection is 2.97 – 13.36 Mbps (excluding Paris); **(3)** compared with that of open WiFi networks, the connectivity setup overhead of provider-managed WiFi networks is higher, however, it is only a

small fraction of the entire WiFi association duration (10% – 18%); **(4)** in terms of AP selection, the existing strategy solely based on signal strength is sub-optimal; as 5 GHz APs usually provide better performance than 2.4 GHz APs in this scenario, from the perspective of AP selection, 5 GHz AP can be given priority; **(5)** provider-managed WiFi networks usually do not guarantee that the IP address can remain unchanged across different APs. Therefore, from the perspective of applications and transport protocols, the ability to support IP handover is necessary; **(6)** during WiFi associations, high latency jitter, high packet loss rates, and large bandwidth changes are likely to occur; **(7)** in terms of congestion control, BBR [16] that provides high loss resilience does not show better transport performance than Cubic [107], the main reason may be high delay jitter affects the transport performance of BBR; **(8)** concurrent TCP flows can provide better goodput than a single TCP flow.

To summarize, the main contributions of this chapter are: **(1)** a measurement system for measuring the transmission capability provided by provider-managed WiFi networks in mobile scenarios; **(2)** a unique dataset that presents the availability, transmission capability, and network quality characteristics of the connectivity offered by provider-managed WiFi networks in mobile scenarios; **(3)** the observations extracted from the above dataset, which provide some insights on AP selection, the optimization of the connectivity setup process, the design and optimization of transport protocols and congestion control algorithms. In addition, the measurement system [1] and the dataset [2] are open sourced on Github for further exploration by the research community.

The rest of this chapter is organized as follows. First, § 3.2 describes how the measurement system is designed. Next, the measurement methodology is introduced in § 3.3. Afterwards, § 3.4 presents some overviews of the collected dataset. After that, the measurement data are analyzed in detail in § 3.5 and § 3.6 respectively. Finally, § 3.7 concludes this chapter.

3.2 Measurement system design

The measurement system includes two components — *X-Fi* (§ 3.2.1) and *X-Perf* (§ 3.2.2). *X-Fi* incorporates a set of optimizations to provide fast access to provider-managed WiFi networks in mobile scenarios. *X-Perf* is a measurement tool to study transport performance of intermittent connectivity. Figure 3.1 illustrates the workflow of the toolkit. Auth., assoc., req., resp., ack., and disc. are abbreviations for authentication, association, request, response, acknowledgement, and discovery respectively. MAC auth. represents the authentication with APs, while access auth. means the authentication with the central authentication server via

[1] https://github.com/yangfurong/X-Fi-code
[2] https://github.com/yangfurong/X-Fi_dataset

the EAP protocol [123]. If the connection is interrupted at any stage, the system goes back to the scanning stage.

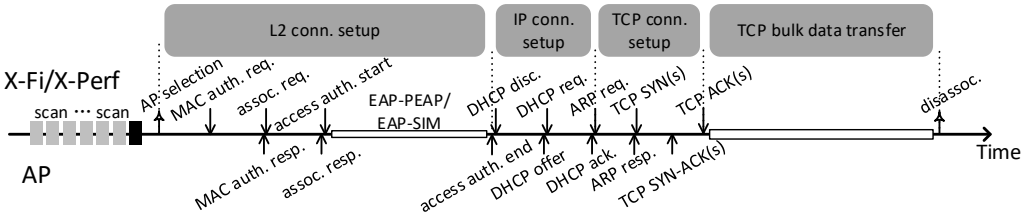


Figure 3.1 Connection procedure handled by *X-Fi*/*X-Perf*

3.2.1 *X-Fi* design

Given that the encounters between moving clients and APs are probably short, a WiFi system that can quickly set up connectivity with encountered APs is of great importance. Although Cabernet [34] proposed a vehicular WiFi client tuned for open WiFi networks via a set of heuristics, it does not support EAP authentication [123] that is prevalent in current provider-managed WiFi networks and is more suitable for mobile scenarios than WEB-based authentication. On the other side, the mainstream WiFi systems for connectivity setup, such as *wpa_supplicant* and *dhcpcd* on Linux, support EAP authentication, but, they are not well-tuned for mobile scenarios, which may waste some transmission opportunities offered by provider-managed WiFi networks. Therefore, we propose *X-Fi* that consolidates the advantages from the two sides above. *X-Fi* consists of two different pieces: **(1)** an enhanced *wpa_supplicant*, which uses a number of best-of-practice heuristics to speed up the WiFi association process; **(2)** an enhanced DHCP client (based on *dhcpcd*) customized for mobile scenarios.

WiFi association setup

The WiFi association process consists of a series of packet exchanges between the WiFi client, the AP, the EAP authentication server, and the DHCP server as shown in Figure 3.1. Given that the vehicular WiFi connectivity has high loss rates in the fringe area of an AP's coverage [47], the WiFi client needs to retransmit the lost packets as fast as possible. As *wpa_supplicant* uses the timeout/retry mechanism to detect and retransmit lost packets, the proper setting of the timeouts is crucial. Therefore, we shortened the timeouts in *wpa_supplicant* to 100 *ms* and the max retry limit to 5 times, as suggested by Cabernet [34].

Besides the changes on timeouts, the AP scanning strategy of *wpa_supplicant* is also optimized via opportunistic active scans. *X-Fi* normally uses passive scans where the WiFi client dwells on each channel for around 80 – 100 *ms* to listen for beacons from nearby APs to get a full list of nearby APs. Then it tries to associate with the AP with the highest Signal-to-Noise Ratio (SNR). If this association fails, *X-Fi* will try to launch an active scan on the channels that have other APs with target ESSIDs[3] (this information was learned from the previous passive scan). The rationale behind this heuristic is that the vehicle is likely still in the range of those APs' coverage, thus, no need to waste time on scanning other channels. If this active scan does not find any AP with a target ESSID, *X-Fi* goes back to a full passive scan. Otherwise, *X-Fi* tries to associate with the AP with the highest SNR. If this association still fails, *X-Fi* can repeat the active scan and association process as long as there still are channels that have untried APs with target ESSIDs and the time since the last passive scan has not exceeded a threshold. In our experiments, we configured this threshold to 5 *s*, considering that the median AP coverage is 50 – 100 *m* (see Figure 3.5b) and 30 km/h is a typical speed of cars running in urban areas.

IP acquisition

Once the supplicant has established link-layer (L2) connectivity to the access point, it is necessary for the client to obtain an IP address in order to communicate with other hosts on the Internet. This is done through the DHCP. The off-the-shelf DHCP client, i.e. *dhcpcd* on Linux, is not tuned for vehicular environments. Specifically, the inefficiency of *dhcpcd* mainly comes from two perspectives: **(1)** the timeouts for DHCP messages are too long for vehicular scenarios as demonstrated by Cabernet [34]; **(2)** the ARP probing mechanism, sending ARP probe packets to check whether the assigned address is in-use by other clients in the same LAN [31], is enabled by default, which may lengthen the IP acquisition process by several seconds. Therefore, we implemented a custom DHCP client based on *dhcpcd* that has shortened timeouts and disables the APR probing mechanism. We note that the timeouts are set to 100 *ms* in our DHCP client, as it is the value recommended by Cabernet [34].

Evaluation of *X-Fi*

The off-the-shelf WiFi system for connectivity setup, *wpa_supplicantdhcpcd*, is compared with *X-Fi* to demonstrate the efficiency of *X-Fi*. The experiments were done as follows. We equipped a car with an on-board computer (see § 3.3.2 for more details about the hardware).

[3]Like *wpa_supplicant*, *X-Fi* allows users to supply a configuration file to specify the ESSIDs of the WiFi networks that it wants to connect to.

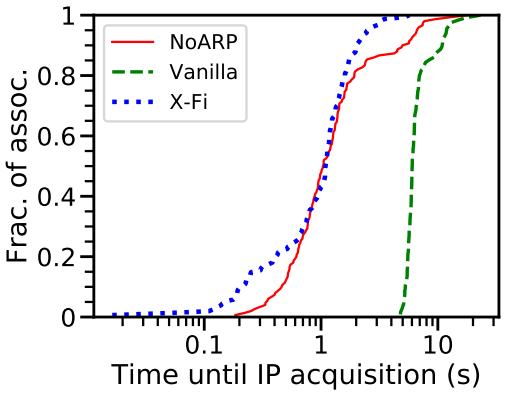


Figure 3.2 *X-Fi* vs vanilla *wpa_supplicant+dhcpcd*

Then, we drove the car for 10 laps following a fixed route in Bologna (the driving time is around 6 hours in total), where we used *X-Fi* for connectivity setup for half of the laps and the *wpa_supplicant/dhcpcd* suite for the other half of the laps.

For every WiFi association, we recorded the "time until IP acquisition" which means the time spent on the WiFi association and DHCP process in total. The results are shown in Figure 3.2: *X-Fi* is significantly faster than the vanilla *wpa_supplicant/dhcpcd* suite with default configurations. The major improvement comes from disabling the ARP probing mechanism in DHCP, while the other heuristics, e.g. shortened timeouts, improve the tail latency of connectivity setup. To sum up, *X-Fi* is able to finish WiFi association setup and IP acquisition within 2 *s* for 90% of the time in our tests.

3.2.2 *X-Perf* design

TCP underpins the vast majority of internet applications. Therefore, studying TCP performance in mobile scenarios is of paramount importance. The popular TCP performance measurement tool, *iPerf*, lacks intermittent networks support[4]. Specifically, *iPerf* has two problems: **(1)** it can not continue the transmission session if an IP handover caused by an AP handoff happens; **(2)** even if the client's IP address does not change after the interruption caused by an AP handoff, *iPerf* can take a very long time to resume the transmission because the TCP RTO timer may increase to a very large value. Therefore, to make full use of the transient transmission window of the intermittent connectivity offered by provider-managed WiFi networks to test the TCP performance without assuming that the WiFi networks support IP roaming, we designed a dedicated performance measurement tool, *X-Perf*.

[4]https://github.com/esnet/iperf/issues/835

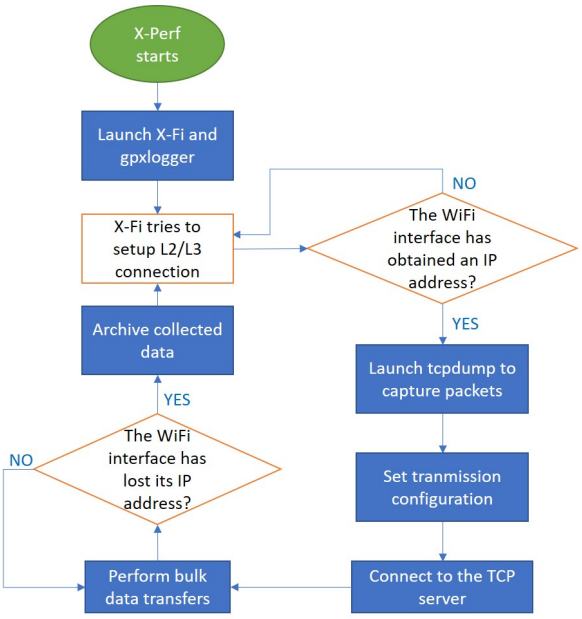


Figure 3.3 The flowchart of *X-Perf*. For the readability of the flowchart, the terminal state is removed from the flowchart.

The working process of *X-Perf* is illustrated in Figure 3.3. Once *X-Perf* is up and running, it executes *gpxlogger* and *X-Fi*, respectively to track GPS coordinates and to start the attempts to associate with WiFi hotspots. Then, *X-Perf* monitors the status of the WiFi interface operated by *X-Fi*. Once the network interface obtains an IP address, *X-Perf* starts to probe the performance of TCP over the wireless link. Specifically, it first decides the transmission configuration for the following TCP bulk transfer, which is randomly selected from a predefined configuration pool. We note that the transmission configuration includes the direction of the transfer (upload or download), the TCP congestion control algorithm (CCA), and the number of concurrent TCP flows (*flow number*). By supplying different configurations to the configuration pool, *X-Perf* allows us to study the impact of different parameters on TCP performance. Thereafter, it measures the TCP throughput by performing data bulk transfers between a remote server and the local onboard computer. During the TCP transmission, all exchanged TCP packets are recorded by *tcpdump* at the client-side. As soon as *X-Perf* finds that the WiFi interface has lost its IP address, which means that the WiFi connectivity is interrupted, *X-Perf* stops the TCP throughput measurement and archives all data collected in this measurement run. The collected data contain a wealth of information from L2 to the application layer including packet-level traces, and all information is also geo-referenced and synced with GPS. *X-Perf* repeats the above process each time *X-Fi* connects to an AP until the end of the experiment or until the system is turned off.

3.3 Measurement methodology

In this section, we detail our the methodology of our measurement study. First, an overview of the cities selected for our measurement activities is presented in § 3.3.1. Next, we describe our measurement apparatus setup in § 3.3.2. Thereafter, the metrics used for data analysis are introduced in § 3.3.3.

3.3.1 Cities selected for measurement

As different traffic patterns and networking infrastructures may present in different cities, we conducted our measurement activities across four different cities in Europe (Bologna and Paris), in Asia (Macao), and in North America (Los Angeles). The diversity of the cities enables us to collect measurements in radically different urban scenarios, traffic patterns, and viability. Specifically, *Bologna* is a good sample of a medium-size touristy medieval city with narrow streets, which have an unpredictable traffic pattern. *Macao* is, again, a medium-size city characterized by a number of high-rise buildings as skyscrapers and casinos. *Paris* is an extremely touristy and crowded metropolis with a well-defined city center. *Los Angeles* (LA) is a very populated megalopolis, with an alternating of residential areas, working districts, and rural regions.

In each city, we subscribed to one or more local providers offering access to urban WiFi APs. This gave us the WPA credentials/SIM cards needed to authenticate and connect while driving. Different cities, and different providers, have different authentication strategies: PEAPv0/EAP-MSCHAPv2 [101] in Bologna, LA, and Macao; EAP-SIM [109] in Paris. Besides, the access point deployment strategies that we encountered during our data collection can be grouped into two main approaches: **(1)** leveraging existing hardware installed in customers' premises; **(2)** installing new dedicated APs. In the first approach, the providers rent the APs to their subscribers. The rented APs present two different ESSIDs when in operation. While one is exclusively used by the customers and secured with WPA/WPA2-PSK [129], the other serves as a WiFi hotspot for public users. Therefore, both the physical capacity of the 802.11 channel and the available bandwidth of the connection, result to be shared between private customers and public users. Quality of Service rules, as token-bucket algorithms, may be applied to prioritize the customers. On the other hand, the second deployment strategy consists of the installation, done by the network providers, of dedicated APs at either indoor or outdoor spots.

3.3.2 Measurement apparatus setup

Hardware: The computer used for the data collection is a Slimpro SP675FP Fanless Mini PC. It features an Intel x64 3^{rd} Gen Core processor, a 3x3 MIMO full-size Mini PCIe 802.11n card (WLE350NX-7A), an u-Blox EVK-7N GPS unit, 16 GBs of RAM, and 300GBs of SSD for storage. While for the deployment in Macao we adopted three dipole antennas, in Bologna, Los Angeles, and Paris, we opted for three multi-polarised antennas. Using different antennas in Macao was not possible due to constraints imposed by the vehicle supplier; it is noteworthy that this limitation slightly affected the *X-Fi* performance in the Macao scenario.

Software: The onboard computer runs Ubuntu 16.04.1 with 4.10.0-33-generic Linux kernel. At the computer boot, *systemd* launches *X-Perf* (§ 3.2.2). Once up, it executes *gpxlogger* and *X-Fi* (§ 3.2.1), respectively to track the GPS coordinates and to start the association and connection attempts with nearby WiFi hotspots. Thereafter, *X-Perf* starts measuring the performance of WiFi offloading and collecting measurement results as explained in § 3.2.2. We configured *X-Perf* to use the following configuration pool (explained in § 3.2.2): CCA $\in$ {BBR, CUBIC}, direction $\in$ {upload, download}, *flow number* $\in$ [1, 16]. BBR [16] and CUBIC [107] are chosen because the two are now the predominant CCAs on the Internet [89]. BBR is a recent work adopted by many Google services and outperforms classic window-based CCAs across various scenarios, especially over lossy links. CUBIC is window based and is the default Linux congestion control. This configuration pool enables us to compare different CCAs and study the impact of concurrent flows. It is noteworthy that system parameters related to Linux TCP stack (e.g. buffer size, etc.) are set by the operating system to the default values.

Long-term deployment in Macao: In Macao, we placed the Slimpro Mini PC under the back seats of a granted van. Two WiFi antennas and the GPS antenna were installed on its roof and one extra WiFi antenna under the back seats for space concerns. The van runs errands on behalf of the granting institution and transport staff. For the whole duration of the experiment, the mobility pattern of the van remained unaltered from its usual one, allowing us to collect in-the-wild samples representative of transporters' real-world mobility. The TCP server was hosted in a virtual machine (VM) based in Singapore to reduce latency.

Short-term trial in Paris: In Paris, we instrumented a vehicle with: three WiFi antennas and one GPS antenna placed on the roof of the vehicle, the Slimpro Mini PC located on the front seat and powered by the vehicle. A *Dekart* SIM card reader is used for the EAP-SIM authentication required by *Free* [39]. We deployed the TCP server on a machine located in Paris. The driving routes covered more than half of the districts in Paris downtown.

Short-term trial in LA and Bologna: The vehicle used in LA was equipped as the one in Paris, except for the SIM card reader. Here, we methodically mapped both residential and commercial areas. We deployed the TCP server used for our tests on a cloud VM in LA. In Bologna, we replicated the LA setup, but with the server in Paris.

3.3.3 Metrics of interests

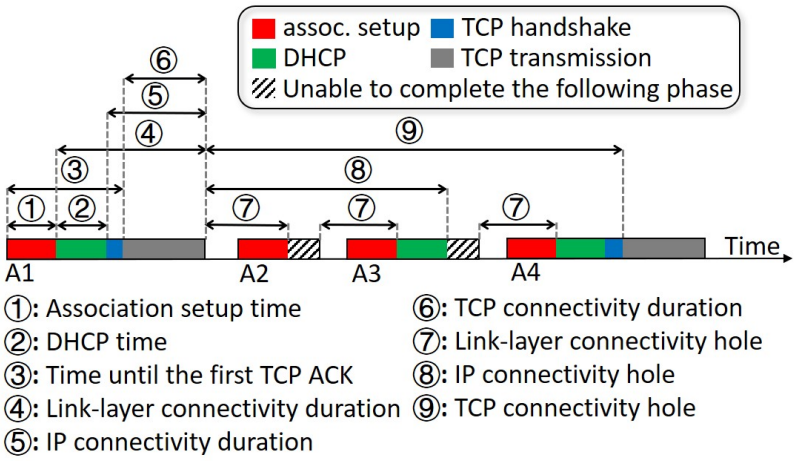


Figure 3.4 The illustration of part of the metrics. In the figure, A1, A2, A3, and A4 stand for four successive associations.

Connectivity: Here now follows the list of metrics used to understand the characteristics of the V2I WiFi connectivity. We note that the first nine metrics are also illustrated via Figure 3.4 to make them more intuitive.

- *Association setup time:* The time needed for the association setup process.
- *DHCP time:* The time for the DHCP process, reflecting the overhead for IP acquisition.
- *Time until the first TCP ACK:* The time between the start of the association and the first TCP ACK received from the server, reflecting the total time spent for establishing end-to-end Internet connectivity.
- *Link-layer connectivity duration:* The time that an association lasts.
- *IP connectivity duration:* The duration that an association has an IP address.
- *TCP connectivity duration:* The time between the start and termination of a TCP session[5]. *X-Perf* stops a TCP session right after the WiFi interface loses its IP. Hence, the end of a TCP session coincides with that of IP connectivity.

[5]A TCP session represent one or multiple TCP flows (*X-Perf* may launch multiple concurrent flows to measure transport performance).

- *Link-layer connectivity hole:* The idle period between two successive associations.
- *IP connectivity hole:* The idle period without an IP address between associations.
- *TCP connectivity hole:* The idle period between two successive TCP sessions.
- *Initial signal strength (ISS):* The signal strength sensed during the WiFi scanning phase. It is the only metric considered by stock WiFi implementations during the AP selection. In a stationary case, selecting the AP with the highest ISS works well. We investigate whether this works as well, for mobile scenarios
- *Average speed per association:* The average vehicle speed during an association, which is calculated based on GPS traces and may affect the transport performance.
- *WiFi frequency:* We seek to understand the impact of 5 GHz and 2.4 GHz APs on transport performance in mobile scenarios.
- *IP address roaming support:* We seek to analyze if clients are able to keep IP address across different APs from a provider-managed WiFi network. This may affect the design and choice of transport protocols.

Transport performance: We now present a list of the metrics used to assess the transport performance.

- *Average TCP goodput:* The aggregated average goodput of a TCP session calculated based on *pcap* traces.
- *Data volume:* The total number of bytes that have been successfully transferred during a TCP session.

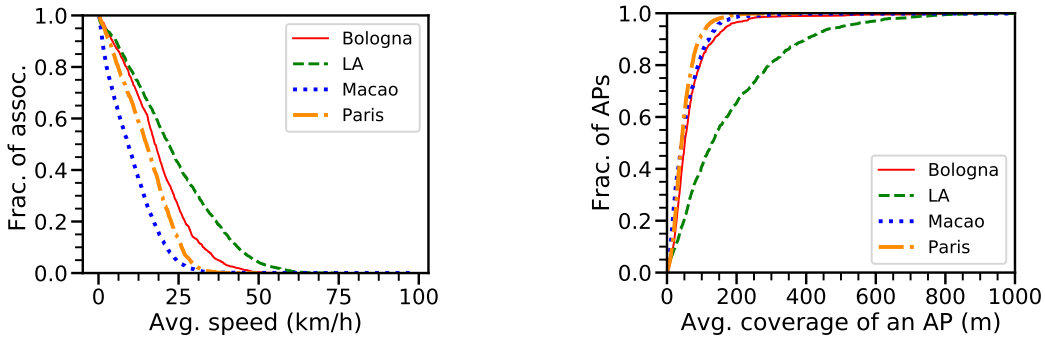
3.4 Dataset Summary

The salient features of our dataset are summarized in Table 3.1. The WiFi networks we connected to in Bologna and Paris (*Fastweb*, *Emilia Romagna regional provider*, *ALMAWIFI* and *Free*) present a higher AP density than the ones in LA and Macao (*Spectrum*, *eduroam*, *TWCPasspoint*, *CableWiFi* and *Companhia de Telecomunicações de Macau* (CTM)). This leads to, on average, more associations per hour: 160, 117, 57, and 17 respectively for Paris, Bologna, LA, and Macao. Remarkably, we observed a lower ratio of IP acquisitions over associations in Paris than in the other cities (43.6% in Paris, 86.7% in Bologna, 85.3% in LA, and 93.1% in Macao). This much lower DHCP success rate in Paris is caused by failures in responding to the DHCP requests, even with good link-layer connectivity ($-40\ dBm$). We confirmed that by connecting to some of these APs which failed to respond to the DHCP requests at locations where their signal strength was decent, between $-40\ dBm$ and $-50\ dBm$.

Table 3.1 Dataset summary

	Paris	Bologna	LA	Macao	Total
Driving time analyzed	11h18m08s	12h48m29s	32h38m06s	311h53m8s	368h37m51s
Driving distance analyzed (km)	144.71	197.12	707.91	4570.89	5620.63
Avg. driving speed (km/h)	12.8	15.39	21.69	14.66	N/A
# of distinct days analyzed	4	5	7	334	350
# of assoc. attempts	4525	5559	6746	21555	38385
# of link-layer assoc.	1812	1501	1856	5412	10581
# of 2.4 GHz assoc.	1812	1218	479	1336	4845
# of 5 GHz assoc.	0	283	1377	4076	5736
# of IP acquisitions.	790	1302	1583	5042	8717
# of TCP tests	664	1179	1490	4736	8069
# of distinct associated APs	1610	917	1385	1013	4925
# of distinct 2.4 GHz APs	1610	796	391	332	3129
# of distinct 5 GHz APs	0	121	994	681	1796
# of WiFi operators	1	3	4	1	9
Total TCP download (GB)	0.7	7	33.1	97.03	137.83
Total TCP upload (GB)	1.3	8.7	13.8	132	155.8

As we have no control nor inside knowledge of the network infrastructure used in Paris, we could not go further explaining the root cause of that. Although the ratios of IP acquisitions over associations are much different across the four cities, the ratios of TCP sessions over IP acquisitions are similar. Therefore, in each of the analyzed cities, once a moving client obtains an IP address it is very likely able to establish TCP connections.



(a) The CCDF of *average speed per assoc.*

(b) The CDF of *AP coverage*

Figure 3.5 The distribution of *average speed per assoc.* and *AP coverage* (failed association attempts are excluded)

The complementary CDF (CCDF) of the *average speed per association* in every city is shown in Figure 3.5a. Given the diversity of APs density and traffic conditions across the four cities, the distributions of *average speed per association* are also dissimilar. In Paris, Macao, as well as in the downtown area of Bologna, where the speed limits are stricter, most of the associations happened at speeds lower than 30 km/h. Instead, outside the center of

Bologna, as in LA, we had the chance to connect with outdoor APs placed along the main roads, driving at higher speed (40 – 62.5 km/h).

Figure 3.5b reports the CDF of *AP coverage*, which is the diagonal length of the smallest bounding box enclosing all GPS points collected during the associations with that AP. The APs in LA present, on average, a much wider coverage (180.17 m) than the ones in the other cities, denoting how the providers placed several outdoor APs along roads and intersections. Our results indicate that the coverage of a single provider-managed AP is generally similar to that of a single open AP [13].

3.5 Connectivity analysis

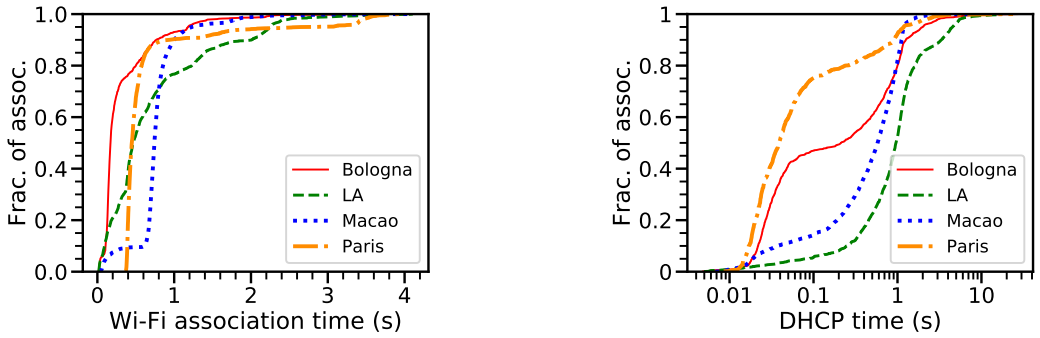
This section presents the measurement results from the connectivity perspective. We first analyze the time needed for connectivity setup in § 3.5.1. Next, the duration of connectivity and holes without connectivity is considered in § 3.5.2. Thereafter, we analyze the impact of moving speed, *ISS*, and *WiFi frequency* on connectivity in § 3.5.3. Finally, the *IP address roaming support* of each provider-managed WiFi network is analyzed in § 3.5.4.

3.5.1 Time for connectivity setup

Association setup time: The CDFs of *association setup time* is reported in Figure 3.6a. In Bologna, Macao, and Paris, 90% of the association setup processes terminate in less than one second. On the other hand, on average, the association setup process in LA takes slightly longer than in the other cities. None of the distributions is heavy-tailed, with the means only lightly skewed from the medians. The data points with relatively longer *association setup time* are due to the re-transmissions of the packets (i.e. authentication packets) necessary for association setup.

DHCP time: The CDF of *DHCP time* is plotted in Figure 3.6b. In Bologna, Macao, and Paris, the 90% of DHCP processes terminiate within 1 s. As the association process, also the DHCP process in LA is slightly longer than in the other cities. The differences between cities might be caused by the ways how operators deploy their DHCP servers.

A more detailed breakdown of the phases prior to gaining IP is shown in Figure 3.7. As expected, the DHCP and EAP authentication account for the major part of the overhead. The reasons for this are two-fold: **(1)** the DHCP and EAP authentication cause more packet exchange between clients and APs than the other two; **(2)** during the DHCP and EAP authentication phase, clients communicate with DHCP and authentication servers normally in core networks, which causes longer latency than communication between clients and APs.



(a) The CDF of *association setup time*

(b) The CDF of *DHCP time*

Figure 3.6 The distribution of *association setup time* and *DHCP time* (failed attempts are excluded).

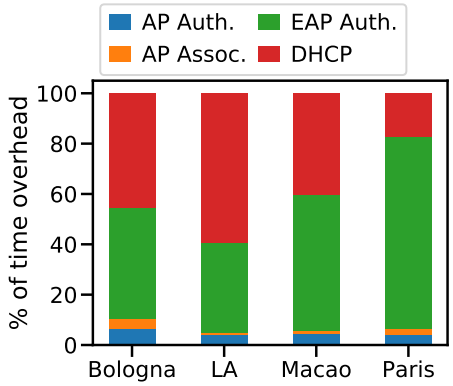


Figure 3.7 The breakdown of the overhead before establishing the IP connectivity. All the values of each phase are averaged over all associations having IP connectivity.

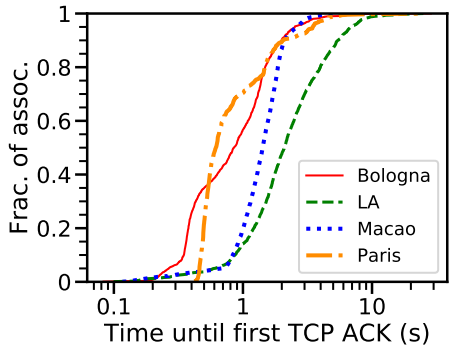


Figure 3.8 The CDF of *time until the first TCP ACK* (failed attempts are excluded).

Time until the first TCP ACK: Figure 3.8 presents the CDF of the *time until the first TCP ACK*. On average, it takes 1.09 s in Bologna, 2.76 s in LA, 1.48 s in Macao, and 1.11 s in Paris. We note that these numbers are larger than the one (0.37s) in open WiFi networks [34]. This is due to the fact that provider-managed WiFi networks require clients to interact with the authentication and DHCP server located in core networks, which leads to lengthened process. Nevertheless, the time to obtain TCP connectivity only counts for a very small part of the entire association (10% – 18%, Figure 3.9), meaning that, for more than the 80% of the time of an association, it is possible to transfer data.

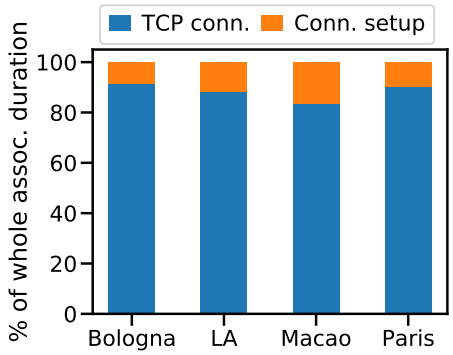


Figure 3.9 The average ratio of *time until the first TCP ACK* over the association duration.

Takeaway: The time spent for establishing end-to-end Internet connectivity in provider-managed networks is larger than the one in open WiFi networks because of the more complicated and lengthened interactions with the authentication and DHCP server in core networks. Therefore, optimizations, such as adopting authentication methods for faster handoffs [5, 153] and granting extended DHCP leases to clients to retain its IP address, may reduce this time. Nonetheless, on average, this overhead only counts for a small part (10% – 18%) of the whole duration of an association.

3.5.2 Connectivity duration and holes

Link-layer connectivity duration: Figure 3.10 shows the CCDF of the *link-layer connectivity duration*. Mean values are 16.74 *s* in Bologna, 29.28 *s* in LA, 21.27 *s* in Macao, and 16.58 *s* in Paris. The distributions are heavy-tailed, with a few long-lasting associations due to traffic lights and traffic jams, typical of urban mobility.

IP connectivity duration: Figure 3.11a depicts the CCDF of the *IP connectivity duration*. Mean values are 17.47 *s* in Bologna, 31.91 *s* in LA, 21.97 *s* in Macao, and 15.84 *s* in Paris. As for the link-layer, also these are heavy-tailed. In Bologna, LA, and Macao, both mean and median *IP connectivity duration* are slightly longer than the ones at link-layer. This is

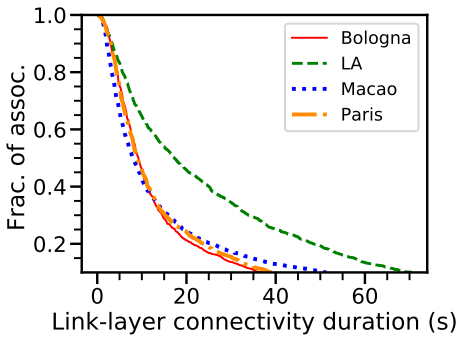
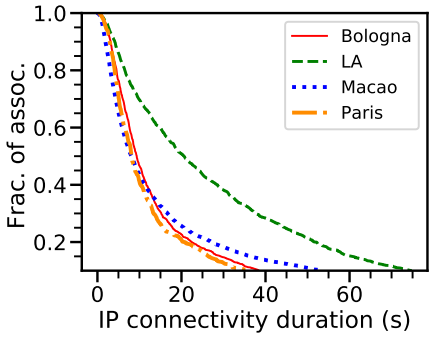
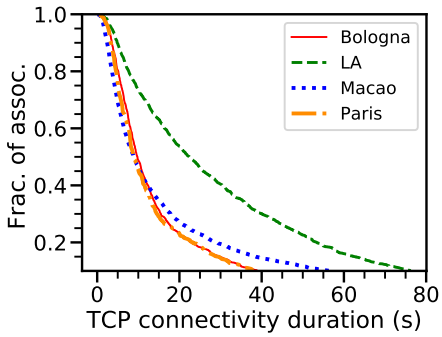


Figure 3.10 The CCDF of *link-layer connectivity duration* (failed attempts are excluded)

due to the exclusion of failed IP acquisitions (Table 3.1), likely to have shortened *link-layer connectivity duration*.



(a) The CCDF of *IP connectivity duration*



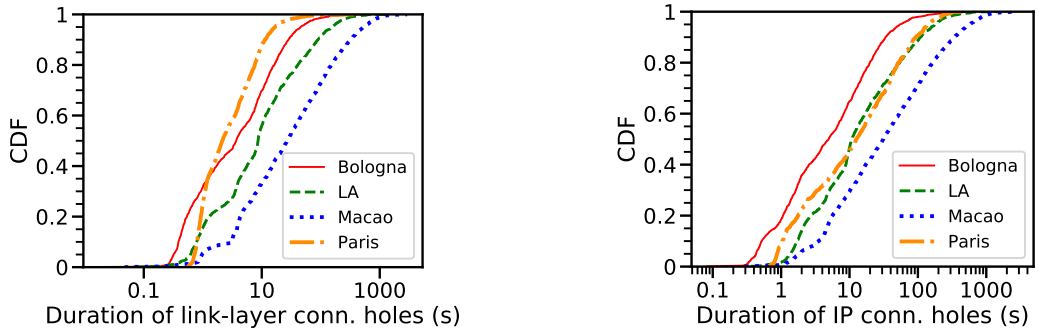
(b) The CCDF of *TCP connectivity duration*

Figure 3.11 The CCDF of *IP connectivity duration* and *TCP connectivity duration* (failed attempts are excluded).

TCP connectivity duration: The CCDF of the *TCP connectivity duration* is reported in Figure 3.11b. Mean values are 18.08 *s* in Bologna, 33.57 *s* in LA, 23.24 *s* in Macao, and 17.24 *s* in Paris. The mean and median values of the *TCP connectivity duration* are slightly higher than the ones of the *IP connectivity duration* due to the exclusion of associations unable to establish TCP connectivity, which tend to have shorter connectivity duration.

Link-layer connectivity holes: Due to the limited coverage of the existing urban WiFi deployments, there are gaps between successive associations. The CDF of the duration of the *link-layer connectivity holes* (Figure 3.12a) reflects the frequency at which vehicles encounter APs, denoting their density in the region. Both in Bologna and in Paris, the AP density is capillary, with a mean inter-arrival time between two consecutive associations of 12.23 *s* and 5.7 *s*, respectively. On the other hand, APs in LA and Macao are rarer, with longer mean inter-arrival time (33.08 *s* and 112.27 *s*). Since in Paris we only drove through

densely populated neighborhoods, the mean is not too far from the median, with no sudden nor significant changes in the AP density across different locations. For all the other cities, the mean values are skewed from their median by a factor of 4x, meaning the holes at the tail of the CDF are particularly long. This denotes how the AP distribution changes significantly across different regions in Bologna, LA, and Macao. The presence of a few extremely short holes (tens of milliseconds) is due to de-authentication packets sent by some APs to force the disassociation from them.



(a) The CDF of *link-layer connectivity holes*

(b) The CDF of *IP connectivity holes*

Figure 3.12 The CDF of *link-layer connectivity holes* and *IP connectivity holes*.

IP connectivity holes: As not every association succeeds in acquiring the IP address (refer to Table 3.1), we expect the duration of *IP connectivity holes* to be enlarged by failed IP acquisitions. The results are shown in Figure 3.12b. The mean values are 15.69 *s*, 40.70 *s*, 120.72 *s* and 34.42 *s* in Bologna, LA, Macao, and Paris respectively. For Bologna, LA, and Macao, the mean duration of *IP connectivity holes* is just marginally longer than that of the *link-layer connectivity holes*. This is because only a small fraction of associations in Bologna, LA, and Macao did not acquire the IP. Though, in Paris, the mean duration of the *IP connectivity holes* is greater than that at the link-layer. This is due to the DHCP issue discussed in § 3.4.

TCP connectivity holes: From Figure 3.13 we observe how the duration of the *TCP connectivity holes* is slightly longer if compared to that of IP. This is because of the exclusion of the fraction of associations that failed to establish a TCP connection, i.e. 15.9% in Paris, 9.4% in Bologna, 5.9% in LA, and 6% in Macao. The mean values are 18.13 *s*, 43.45 *s*, 124.39 *s* and 42.45 *s* in Bologna, LA, Macao, and Paris respectively.

Takeaway: The connectivity offered by provider-managed WiFi networks is generally similar to that provided by open WiFi networks [13, 34]. Mobile clients can expect to offload data for tens of seconds via an encounter with a provider-managed AP, and, are likely to have several such encounters every few minutes.

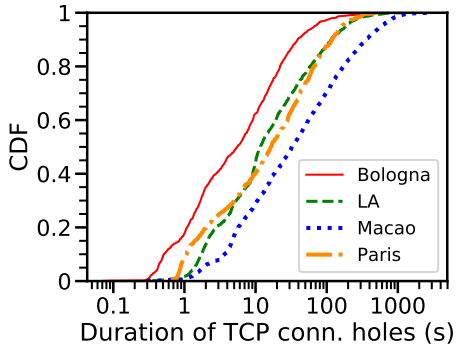


Figure 3.13 The CDF of *TCP connectivity holes*.

3.5.3 Impact of moving speed, ISS, and frequency

Next, we study the factors that may affect WiFi connectivity in mobile scenarios. Specifically, we analyze the impact of moving speed, *ISS*, and *WiFi frequency* on the duration of link-layer connectivity. In the interest of space, we do not present the impacts on the duration of L3/L4 connectivity, as they are similar to the link-layer connectivity's ones.

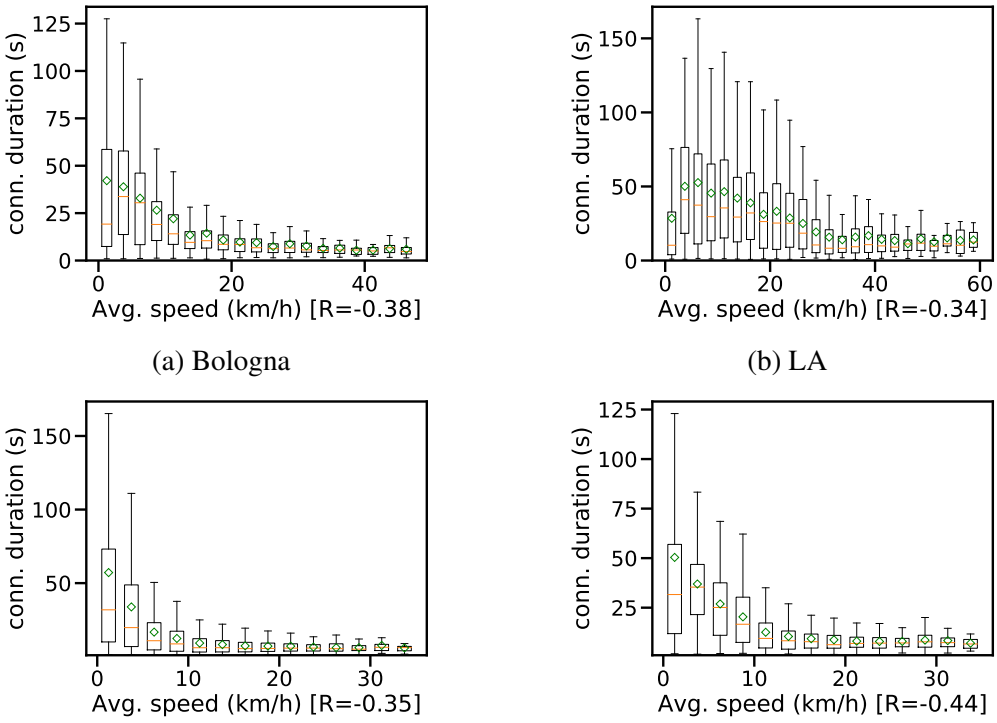
Average speed per association: The first factor that we consider is the moving speed. Intuitively, higher speeds lead to shorter connectivity duration, as the client leaves the area covered by APs quicker. Figure 3.14 confirms the correlation between the moving speed and the *link-layer connectivity duration*.

Initial signal strength: Now, we consider *ISS*. In the stationary case, choosing the AP with the highest *ISS* works well. However, due to the continuous changes of the clients' position caused by mobility, the *ISS* alone is not enough to infer the network performance.

In all the cities, the *ISS* ranges from being very weak (-80 *dBm*) up to an excellent signal level (greater than -50 *dBm*) as shown in Figure 3.15. This allows us to analyze the impact of ISS on connectivity duration.

We plot the correlation between the *ISS* and the link-layer connectivity duration in Figure 3.16. From the figure, we can not evince any obvious pattern confirming how associations with higher *ISS* last longer when mobility is involved. Surprisingly, some associations with a higher *ISS* are shorter than those with a lower *ISS*. This may be due to the client moving from one extremity to the other of the AP coverage when it senses a weak *ISS*; while the vehicle could be already leaving the middle of the AP coverage when it senses a strong *ISS*. Hence, more sophisticated AP selection strategies, based on not only *ISS*, are called for improving connectivity duration.

WiFi frequency: The last factor we take into account is the frequency of the WiFi APs. Since Free has only 2.4 GHz APs available in Paris, we omit it. Table 3.2 reports the



(a) Bologna (b) LA

(c) Macao (d) Paris

Figure 3.14 The correlation between *average speed per association* and connectivity duration: associations are partitioned into 2.5 km/h bins according to their Average Speed. In each bin, red lines, green diamonds, lower box edges, upper box edges, lower whiskers, and upper whiskers are medians, means, $Q1$ (the 1st quartile), $Q3$ (the 3rd quartile), $Q1 - 1.5 \times IQR$, and $Q3 + 1.5 \times IQR$ respectively, where IQR (Interquartile Range) equals to $Q3 - Q1$. R denotes the Pearson's Correlation Coefficient.

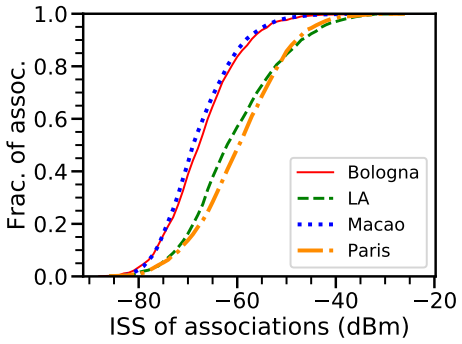
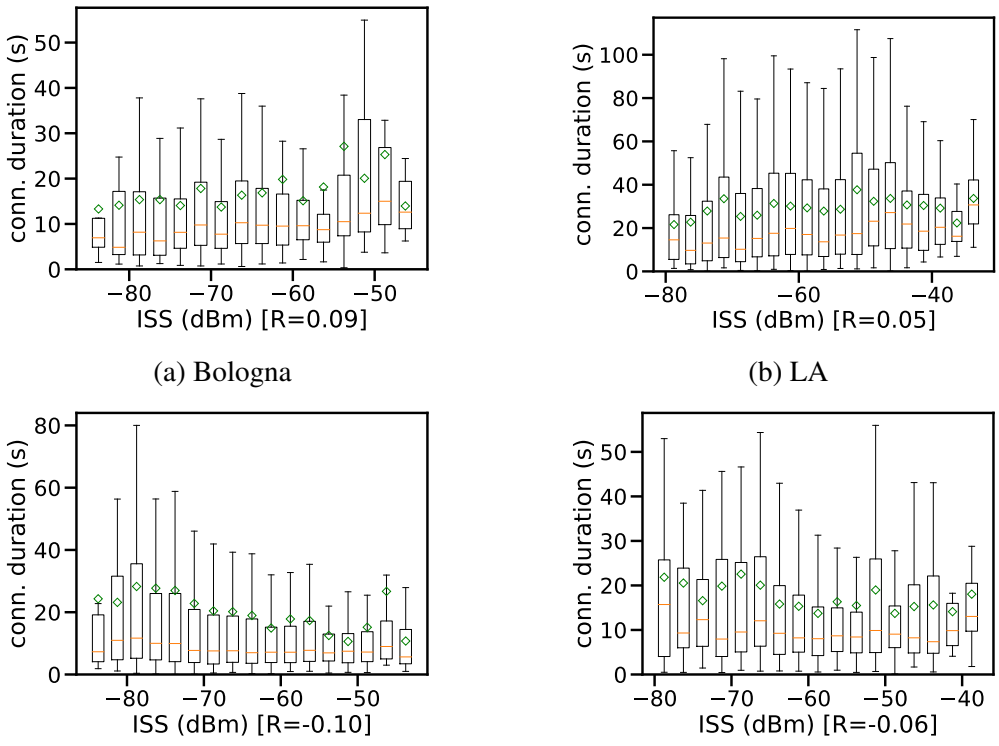


Figure 3.15 The CDF of *ISS*.

correlations between the APs frequency and the connectivity duration. Interestingly, from our results, the duration of the connectivity offered by 5 GHz APs is sometimes better or at least comparable to the one offered by 2.4 GHz APs. This is likely because 2.4 GHz band has fewer independent channels than 5 GHz band, thus, leading to severer radio interference, which further causes bursty beacon losses and interrupts connectivity.

Takeaway: (1) The AP selection algorithm of stock WiFi client implementations, which solely relies on ISS, is not sufficient when it comes to mobility. Applying location-aware AP selection approaches based on GPS history [93] is a good direction, which is likely to achieve better performance as of today since the provider-managed WiFi APs are very likely more stable and predictable than open APs; **(2)** 5 GHz APs often offer longer connectivity than 2.4 GHz APs in mobile scenarios likely due to the less severe radio interference.



(a) Bologna (b) LA

(c) Macao (d) Paris

Figure 3.16 The correlation between *ISS* and connectivity duration: associations are partitioned into 2.5 *dBm* bins according to their ISS. We draw the box-whisker plot as in Figure 3.14. R denotes the Pearson's Correlation Coefficient.

Table 3.2 The correlation between *WiFi Frequency* and connectivity duration.

City	WiFi Frequency	Link-layer duration (s)	
		Mean	Stdev
Bologna	5 GHz	26.22	39.62
	2.4 GHz	14.54	19.56
LA	5 GHz	30.09	33.38
	2.4 GHz	26.94	30.67
Macao	5 GHz	20.22	40.33
	2.4 GHz	24.52	41.57

3.5.4 IP address roaming support

Lastly, we investigated how the IP address (the local IP of the vehicle) assigned by a provider-managed WiFi network changes across different APs during a continuous drive. Usually, the IP address roaming (the ability of a host to retain the same IP address when roaming across APs) is not supported when a WiFi client switches to another operator's WiFi network unless the operators have a particular agreement. Hence, we focus on the IP address roaming inside WiFi networks operated by the same provider. Specifically, we investigate: **(1)** Can a vehicle keep its IP address unchanged during a continuous drive? **(2)** If not, for how long can a vehicle keep the same IP address?

Table 3.3 reports the data collected for every WiFi network[6] used in our measurement campaign. The column IP duration reports the average time during which clients keep an IP address unchanged when moving. The IP duration values shown are formatted as *average* ± *standard deviation*.

Table 3.3 IP Address Roaming Support in different WiFi networks.

City	Wifi network	IP changed?	IP duration (s)
Bologna	ALMAWIFI	✓	687 ± 1153
	Fastweb	✗	N/A
	EmiliaWifi	✓	184 ± 594
LA	Spectrum	✓	1644 ± 2653
	CableWifi	✓	3105 ± 5097
	Eduroam	✓	106 ± 156
Macao	CTM	✗	N/A
Paris	Free	✓	216 ± 384

From our results, as only two networks apparently support it, we could not find any strong evidence of provider-managed WiFi networks supporting, by default, IP address roaming. Hence, either applications or transport layer protocols should take care of it as, otherwise, switching AP may break the connections established by connection-oriented protocol like TCP.

[6] As we only get 12 associations from the TWCPasspoint WiFi network, we are not able to analyze it.

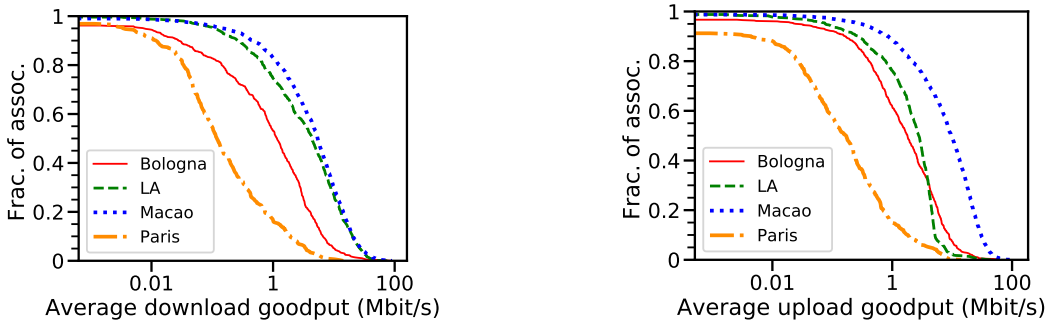
Takeaway: The IP address roaming in provider-managed WiFi networks is not a must implemented by operators by default. Therefore, application designers must bear the caveat in mind if they want to leverage provider-managed WiFi networks in mobile scenarios. That said, transport protocols like QUIC [71, 62], Multipath QUIC [24, 81], and Multipath TCP [38], which supports IP handover, thus, hiding the IP changes from applications, are good options for applications using provider-managed WiFi networks.

3.6 Transport performance analysis

In this section, we first present: **(1)** the average goodput a moving client can experience from an association; **(2)** the amount of data that can be transmitted during a single association. These results in § 3.6.1 demonstrate that today's provider-managed WiFi networks can provide decent throughput in mobile scenarios. Then, in § 3.6.2, we compare the results between different CCAs, and study the impact of *flow number* on TCP goodput. Next, the impact of moving speed, *ISS*, and *WiFi frequency* is analyzed in § 3.6.3. Finally, we analyze the characteristics of the quality of the connectivity offered by provider-managed WiFi networks in § 3.6.4.

3.6.1 Goodput and data volume

Here, we seek to understand the capabilities of today's provider-managed WiFi networks in the V2I context from a high-level point of view. Therefore, we do not distinguish between TCP sessions with different *flow number* and CCA. It is also noteworthy that applications also use various TCP configurations in reality.

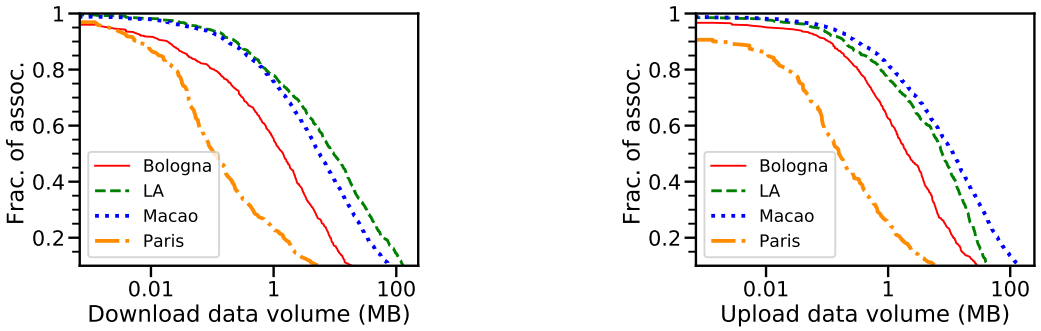


(a) Download (b) Upload

Figure 3.17 The CCDF of *average TCP goodput* (associations without TCP connectivity are excluded).

Average TCP goodput: To support real world applications, a connection with decent goodput is fundamental. In Figure 3.17 we plot the CCDF of the *average goodput per association*, for both upload and download. The results in Macao seems very promising: for every association able to establish TCP connections, the mean goodput is 13.36 Mbps in upload and 8.66 Mbps in download; with peaks up to 103.44 Mbps and 90.48 Mbps, respectively. Peak performances are likely to be achieved when vehicles, due to traffic jams or lights, stop at places with *good* APs. Further, while the results from LA and Bologna are also quite encouraging, TCP does not perform as well in Paris, with 90% of the associations providing less than 2 Mbps for both upload and download. Like in the other cities, also the CCDF of Paris is heavy-tailed, biased by the vehicle stops by good APs. Notice, the maximum average goodput both in upload and download is ∼14 Mbps.

Data volume: To gauge clearer insights on the network capabilities in such a scenario, we consider the actual amount of data uploaded/downloaded per association. Figure 3.18 shows the CCDF of the *data volume* transmitted during each association. In LA, Macao, and Bologna, it is possible to transmit a considerable amount of data during every association (the mean *data volume* downloaded per association is 45.22 MB, 40.31 MB, and 12.46 MB, up to 1.04 GB, 2.26 GB, and 0.81 GB, respectively). On the other hand, on average, the amount of data uploaded per association is 19.15 MB in LA, 55.05 MB in Macao, and 14.67 MB in Bologna, with a maximum of 1.01 GB, 1.86 GB, and 0.5 GB. Once again, in Paris, the results are less attractive, with only 3.88 MB uploaded and 2.39 MB downloaded on average. These figures are skewed by the outliers at the tail of the distribution.



(a) Download

(b) Upload

Figure 3.18 The CCDF of the transfer size per association (associations without TCP connectivity are excluded).

The reasons for the discrepancies between Paris and the other cities could be multi-folds. Firstly, the AP density in Paris is considerably higher than in the other cities and, therefore, we could suffer from higher radio interference in Paris. Secondly, only 2.4 GHz APs are

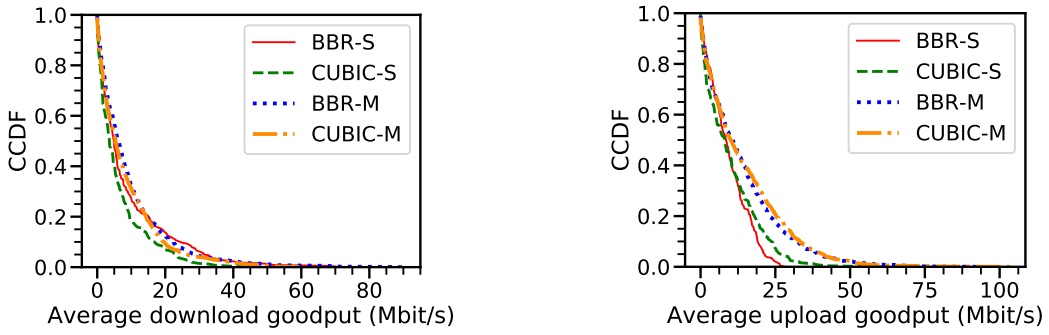
provided by Free. In general, 5 GHz APs offer better performance, as they have more independent channels. Lastly, the APs in Paris are shared between private customers and public users, with customers who are often prioritized by the providers. Besides, in the other cities, there are also dedicated APs, which are with better hardware and friendlier to public users.

From the perspective of long-term goodput, our results from Bologna, LA, and Macao confirm that clients are likely able to download or upload around 1 GB (1.23 GB in Bologna, 1.44 GB in LA, and 0.73 GB in Macao) per hour by using stock TCP implementations, while on-the-go. Furthermore, this figure could improve with more APs coming into operation, and with a more capillary diffusion of 802.11ax APs. Moreover, as TCP is known to be inefficient when it comes to intermittent connectivity, this could get even better by using novel, alternative, transport protocols, such as QUIC [62, 71].

Takeaway: Compared with the numbers provided by open WiFi networks in the past [9, 13, 34, 116, 115], the TCP goodput (∼1 GB/h) offered by provider-managed WiFi networks is an order of magnitude higher, which shows the great potential of using provider-managed WiFi networks to complement cellular networks in mobile scearios.

3.6.2 Impact of CCA and flow number

Next, we compare the average TCP goodput per association of different TCP configurations to investigate the impact of CCA and *flow number*. It is noteworthy that the following analysis is based on the data collected from Macao, as it has more data for us to draw the relationships in a statistically significant way. Figure 3.19 shows the CCDF of average TCP goodput per association under different TCP configurations.



(a) Download (b) Upload

Figure 3.19 The CCDF of *average TCP goodput* under different TCP configurations: S denotes single flow, and M denotes multiple flows.

First, from Figure 3.19, we observe that no obvious differences between BBR and CUBIC, despite the superiority of BBR in conventional scenarios reported by previous studies [16]. The most likely reason is that BBR underestimates the round-trip propagation time (*RTprop*) in such a dynamic environment, which leads to an inappropriate estimation of congestion window size, thus, counterbalancing the merits of BBR. We note that this phenomenon is also reported by a TCP measurement study in high-speed rail scenarios [124].

Another observation made from Figure 3.19 is that multiple TCP flows often deliver better performance than a single TCP flow. The reasons are two folds: **(1)** uncongestional packet losses have a severer impact on single flow than multiple flows as it is likely that only a part of flows is affected in the case of multiple flows; **(2)** multiple flows are more competitive in bandwidth contention than single flow.

Takeaway: (1) BBR and CUBIC perform close to each other in terms of TCP goodput when used for the connectivity offered by provider-managed networks in mobile scenarios; several techniques [70, 124, 139] that compensates the underestimation of *RTprop* may be applied to boost BBR's performance; **(2)** applications can benefit from the use of concurrent TCP flows when offloading traffic via provider-managed WiFi networks.

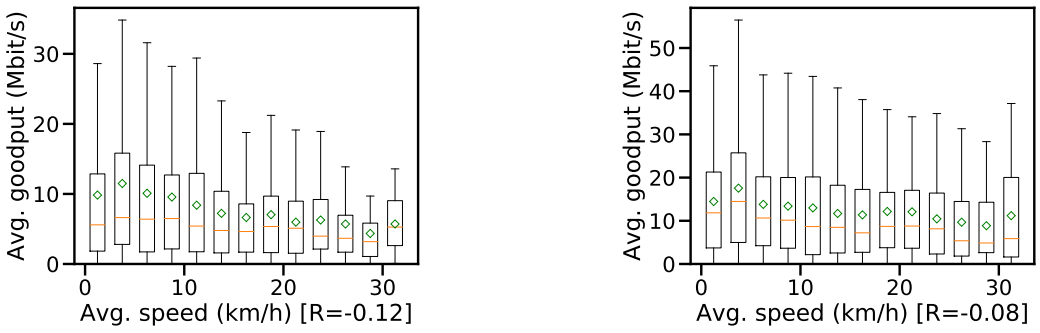
3.6.3 Impact of moving speed, ISS, and frequency

Now, we consider how moving speed, *ISS*, and frequency would impact TCP goodput. As in the previous subsection, we conduct our analysis based on the data collected from Macao due to its' larger scale.

First, the correlation between *average speed per association* and *average TCP goodput per association* is drawn in Figure 3.20. We notice that, unlike the correlation between moving speed and connectivity duration in Figure 3.14, the TCP goodput is not strongly related to the speed: for download and upload, the R between speed and TCP goodput is only -0.12 and -0.08 respectively. On the other hand, we also acknowledge that the maximum goodput achieved in each speed bin tends to be higher when the speed is lower. We note that our results may be limited to the fact that the moving speeds were relatively slow in Macao, $0-35\ km/h$.

Then, we draw the correlation between *ISS* and *average TCP goodput per association* in Figure 3.21. Similar to the correlation between *ISS* and connectivity duration in Figure 3.16, we do not find any evidence indicating that stronger *ISS* leads to higher TCP goodput. The results again call for more sophisticated AP selection algorithms that consider more factors rather than just *ISS*.

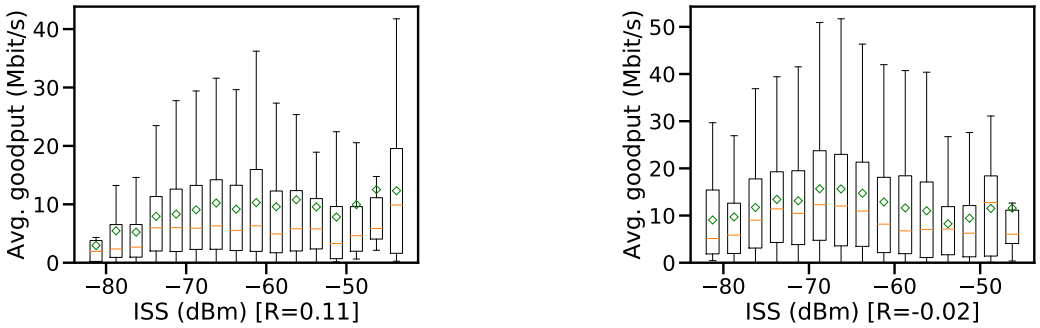
Lastly, we investigate how different APs with different WiFi frequencies could perform. Table 3.4 summarizes the average goodput achieved from 5 GHz and 2.4 GHz APs in both



(a) Download (b) Upload

Figure 3.20 The correlation between *average speed per association* and *average TCP goodput*: associations are partitioned into 2.5 km/h bins according to their average vehicle speed. The box-whisker plot is drawn as in Figure 3.14. R denotes the Pearson's Correlation Coefficient.

upload and download scenarios. We can observe that 5 GHz APs provide higher TCP goodput than 2.4 GHz APs, which comes from the fact that the 5 GHz band has more independent channels, and, therefore, more chances to use 40 MHz channel and less radio interference than the 2.4 GHz band.



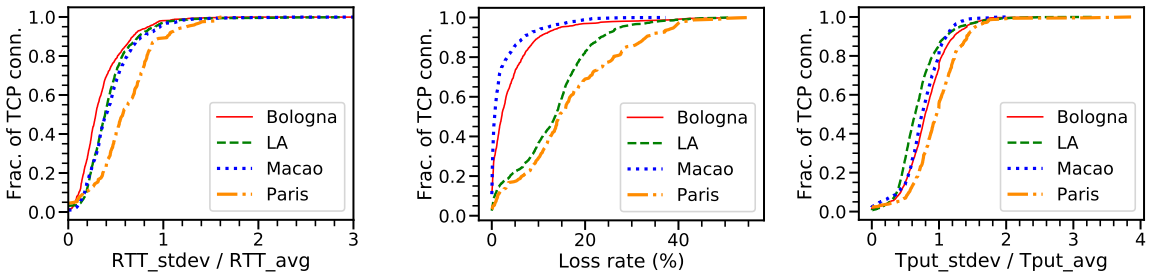
(a) Download (b) Upload

Figure 3.21 The correlation between *ISS* and *average TCP goodput*: associations are partitioned into 2.5 *dBm* bins according to their ISS. The box-whisker plot is drawn as in Figure 3.14. R denotes the Pearson's Correlation Coefficient.

Takeaway: (1) TCP goodput is not correlated with *ISS*, which corroborates that *ISS* alone is not sufficient for the AP selection; **(2)** 5 GHz APs provide notably higher goodput than 2.4 GHz APs. Thus, from the perspective of AP selection, 5 GHz APs can be prioritized to have more chance to get higher performance.

Table 3.4 The correlation between *WiFi Frequency* and *average TCP goodput*.

Trans. direction	WiFi Frequency	Goodput (Mbps)	
		Mean	Stdev
Upload	5 GHz	16.25	13.21
	2.4 GHz	4.33	5.63
Download	5 GHz	10.26	10.92
	2.4 GHz	3.71	4.22



(a) The CDF of jitter (ratio)

(b) The CDF of loss rate

(c) The CDF of throughput standard deviation (ratio)

Figure 3.22 The CDF of jitter, loss rate, and standard deviation of throughput.

3.6.4 The characteristics of connectivity quality

Finally, we analyze the characteristics of the quality of connectivity offered by provider-managed WiFi network in mobile scenarios, which could help to design efficient transport protocols for this use-case. Specifically, three metrics are mainly concerned, i.e., the degree of delay jitter, the packet loss rate, and the degree of bandwidth variation. We calculate the following values based on the pcap file of each TCP flow: the standard deviation of RTT (RTT_stdev), the average value of RTT (RTT_avg), the loss rate (Loss Rate), the standard deviation of the "real-time throughput" over 1 s interval (Tput_stdev), and the average throughput (Tput_avg). Then, the degree of delay jitter is measured by RTT_stdev/RTT_avg, and the degree of bandwidth change is measured by Tput_stdev/Tput_avg. Figure 3.22 shows measurements related to network quality characteristics. In terms of delay jitter, for Bologna, Los Angeles, Macau and Paris, the proportion of TCP connections with a standard deviation of RTT more than half of the average RTT is 27%, 30%, 32%, and 60%. From the perspective of packet loss rate, the ratio of packet loss rate over 5% in the above-mentioned four cities is 26%, 13%, 78%, and 82% respectively. Moreover, the packet loss rate of some connections even exceeds 20%. From the perspective of bandwidth variation, the proportion of TCP connections whose throughput standard deviation exceeds half of the average throughput in the four cities is 85%, 73%, 85%, and 94%, respectively. The above data shows that it is

common in this scenario that the network quality exhibits high latency jitter, high packet loss rate, and large bandwidth changes.

Takeaway: In mobile scenarios, the quality of the connectivity offered by provider-managed WiFi networks often exhibits the characteristics of high latency jitter, high packet loss rate, and large bandwidth changes. Therefore, in the design of transport protocols and congestion control algorithms, these situations need to be considered in order to obtain better transport performance.

3.7 Conclusion

At present, many operators have deployed public WiFi networks in urban areas on a large scale, giving mobile users in urban areas the opportunity to use WiFi networks to complement cellular networks, thereby reducing transmission costs and improving transport performance. We designed a measurement system based on in-vehicle devices to measure and analyze the availability, transport performance, network quality characteristics, and related influencing factors of provider-managed WiFi networks in mobile scenarios in four typical cities in Europe, Asia, and North America. Based on the above measurement results, we provide some guidance and insights for AP selection, connection establishment process optimization, and the optimization of transport protocols and congestion control algorithms.

Chapter 4

BBRv2+: Enhancing BBRv2 with delay information

4.1 Introduction

Transport protocols are the key components for accessing various Internet application services in Mobile Internet. Since the Internet infrastructure is constructed based on the packet-switched network of statistical multiplexing, in order to avoid the network congestion collapse [63], it is necessary to carry out congestion control in transport protocols. Congestion control in transport protocols aims to efficiently utilize available network resources while ensuring that bottleneck bandwidth is fairly shared among multiple flows. Implementing efficient congestion control in Mobile Internet faces two main challenges. First, as shown in the previous chapter and some measurement results [64, 121, 124], whether using cellular or WiFi for network access, due to the introduction of mobility, there are often strong dynamics, such as large bandwidth changes, high delay jitter, and high packet loss rates. Second, since there are a large number of traditional congestion control algorithms (CCAs) based on packet loss on the Internet, for the consideration of incremental deployment, the newly designed congestion control algorithm also needs to consider the TCP fairness to traditional algorithms.

Congestion control has been one of the active research topics in computer networks since it was introduced in the 1980s [63]. More than three decades of research on congestion control have brought us a plethora of congestion control algorithms and TCP variants, aiming at efficient utilization of available bandwidth while fairly sharing the bottleneck bandwidth among multiple flows. For instance, Linux kernel alone has more than 15 different CCAs [2]. While we see many recent proposals on learning-based CCAs (e.g. Remy [130], Aurora [65],

PCC-Vivace [29], Indigo [138], Orca [2]), the widely deployed CCAs today are still classic ones (e.g. Cubic [107], BBR [16]). Among them, BBR has also been widely used in Mobile Internet scenarios due to its advantages in packet loss resilience and low latency [16, 89, 122].

In this chapter, we focus on BBR and its upgrade, BBRv2, as BBR has been used by 22% of the Alexa Top 20K websites [89] and BBRv2 will likely replace BBR in the near future[1]. BBR is a model-based CCA that sets its sending rate based on the measured bottleneck bandwidth (*BtlBW*) and round trip propagation time (*RTprop*). That said, instead of reacting to congestion signals such as packet losses or delay dynamics, BBR tries to actively operate at Kleinrock's optimal operating point [69] to maximize throughput without incurring standing queues at the bottleneck link. Previous empirical studies [14, 56, 70, 82, 111, 127] have disclosed several shortcomings of BBR: **(1)** it introduces excessive retransmissions in shallow-buffered networks[2]; **(2)** it is not fair when competing with flows using loss-based CCAs (e.g. Cubic) and BBR flows with different Round Trip Times (RTTs); **(3)** its performance degrades when network jitter is high.

To address these issues in BBR, Google proposed BBRv2 [18] that inherits most of the design principles from BBR, but reacts to packet losses and Explicit Congestion Notification (ECN) marks for better co-existence with loss-based CCAs and being less aggressive in shallow-buffered networks. Given that BBRv2 may eventually replace BBR, understanding how BBRv2 actually performs is of great importance for its deployment. We have also seen several studies [17, 44, 66, 90, 114] on measuring BBRv2, which have shown that, in comparison with BBR, BBRv2 improves inter-protocol fairness against loss-based CCAs and reduces retransmissions in shallow-buffered networks. Nevertheless, we found in this paper, these improvements come with several costs, including low resilience to random packet losses and slow responsiveness to bandwidth dynamics. However, these studies do not comprehensively consider the network environment that BBRv2 may face in Mobile Internet scenarios (e.g. large bandwidth changes, high delay jitter, and high packet loss rates, and etc.).

Therefore, we first evaluate BBRv2 in various network conditions. We find that the improvements of BBRv2 come with several costs, including low resilience to random packet losses and slow responsiveness to bandwidth dynamics. These costs makes BBRv2 unable to achieve high performance in some Mobile Internet scenarios. Our key observations from the empirical study of BBRv2 are as follows:

[1] As of March 2021, Google had finished the roll-out of BBRv2 for internal TCP traffic [144].

[2] According to previous studies [56, 111, 114], a buffer smaller than 2×BDP (Bandwidth Delay Product) is considered shallow.

- Due to its conservative strategies in bandwidth probing and congestion window (*cwnd*) estimation, BBRv2 achieves better inter-protocol fairness against loss-based CCAs in shallow-buffered networks than BBR. On the other hand, these strategies also make BBRv2 less competitive than BBR in terms of throughput under moderate buffers. BBRv2 also improves fairness among flows of different RTTs, compared with BBR.
- In shallow-buffered networks, BBRv2 significantly reduces retransmissions, compared with BBR. However, the throughput of BBRv2 is 13% – 16% lower than that of BBR. This is because BBRv2 limits its *cwnd* to about 0.85× Bandwidth Delay Product (BDP) for most of the time in these networks.
- BBRv2 is less resilient to random packet losses than BBR. Interestingly, we find that carefully tuning the loss threshold parameter in BBRv2 according to bottleneck buffer sizes can enhance BBRv2's loss resilience without sacrificing its advantages in retransmission and fairness.
- BBRv2 is less responsive to bandwidth dynamics than BBR, which leads to low bandwidth utilization and high queuing delay in networks with bandwidth dynamics. The long bandwidth probing interval and the long expiry time of *BtlBW* estimation are the two major contributors.
- Like BBR, BBRv2's performance still suffers from *cwnd* exhaustion in high-jitter networks that are not rare in wireless scenarios [10, 23, 70, 124, 148].

The results of our empirical study of BBRv2 raise an important question: *are we able to improve BBRv2's performance while keeping its advantages in retransmission and fairness? If so, how do we achieve this goal?*

BBRv2's shortcomings lie in lower loss resilience and slower responsiveness to bandwidth dynamics. The issue of loss resilience can be mitigated by carefully tuning the loss threshold parameter in BBRv2. But, the aggressiveness of BBRv2 in bandwidth probing needs to be adjusted to improve its responsiveness to bandwidth dynamics. However, how to avoid blindly adjusting the aggressiveness of bandwidth probing is a challenging problem. Currently, the aggressiveness of bandwidth probing in BBR and BBRv2 is either hard-coded or pre-configured according to designers' experience without the perception of the network environment. Such unguided aggressiveness may make the bandwidth probing either over-aggressive (like BBR) or over-conservative (like BBRv2) in certain environments. Thus, we argue that BBRv2 needs to consider the feedback from the network environment to dynamically adjust the aggressiveness of bandwidth probing.

To address the above gap, we propose BBRv2+. Firstly, BBRv2+ integrates delay information into its path model, which serves as the feedback to guide its aggressiveness in

bandwidth probing. Secondly, BBRv2+ partially redesigns the state-machine of BBRv2 in order to utilize the delay information to guide the aggressiveness of BBRv2+'s bandwidth probing. In doing so, BBRv2+ balances between the aggressiveness in bandwidth probing and the fairness against loss-based CCAs. Thirdly, to avoid being suppressed when co-existing with loss-based CCAs in deep-buffered networks because of using the delay information, BBRv2+ incorporates a dual-mode mechanism — it dynamically switches between BBRv2+'s state-machine and BBRv2's state-machine depending on whether loss-based CCAs co-exist. Finally, BBRv2+ addresses the *cwnd* exhaustion problem in high-jitter networks by compensating its estimated BDP according to observed jitter; the compensation mechanism improves the estimation accuracy of BDP in high-jitter networks[3].

Extensive experiments in networks emulated by Mininet [88] and Mahimahi [92] with real-world traces are conducted. The results show that compared with BBRv2, BBRv2+ succeeds to balance the aggressiveness of bandwidth probing and the fairness against loss-based CCAs, improves the resilience to network jitter, and achieves 25% higher throughput and comparable queuing delay in mobile scenarios where the bandwidth is very dynamic.

To sum up, the contributions of this paper are three-fold: **(1)** we make a detailed investigation of BBRv2 to examine its benefits and shortcomings in comparison with BBR; **(2)** we propose BBRv2+ to addresses the shortcomings of BBRv2 while barely sacrificing BBRv2's advantages; **(3)** we show through extensive experiments that BBRv2+ meets its design goal.

The remainder of this chapter is organized as follows. We first give an overview of BBR and BBRv2 in § 4.2. Then, a deep dive into BBRv2 is presented in § 4.3. Next, § 4.4 describes the design and implementation of BBRv2+, followed by the evaluation of BBRv2+ in § 4.5. Finally, the chapter is concluded in § 4.6.

4.2 Background

4.2.1 BBR overview

BBR is designed to operate near Kleinrock's optimal point [69]. Operating at this optimal point depends on accurate measurements of $BtlBW$ and $RTprop$ (as $BDP = BtlBW \times RTprop$). Since these two variables cannot be measured simultaneously, BBR introduces a state-machine-based method that alternatively estimates $BtlBW$ and $RTprop$.

The overall architecture of BBR is illustrated in Figure 4.1. BBR takes a set of measurements as the inputs of its state-machine, builds a path model based on the state-machine, and outputs control parameters to a TCP sending engine that controls how much data can

[3]This mechanism can also be applied to BBR and BBRv2

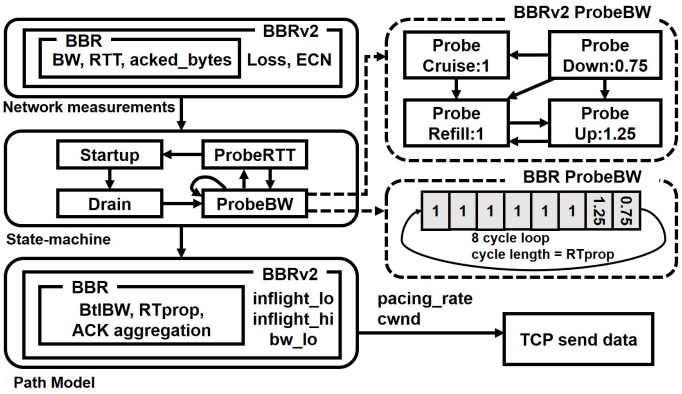


Figure 4.1 The architecture of BBR and BBRv2.

be sent (i.e. setting *cwnd*) and at what speed (i.e. setting *pacing_rate*). The path model of BBR includes *BtlBW* (stored in a 10-RTT-windowed *max_filter* that keeps bandwidth measurements in recent 10 RTTs and returns the maximum), *RTprop* (calculated as the minimum RTT measured in recent 10 *s*), and ACK aggregation degree (for compensating *cwnd* when network paths have severe ACK decimation, e.g. WiFi paths). The output parameters, *cwnd* and *pacing_rate*, are computed based on Equation (4.1). BBR sets its *cwnd_gain* and *pacing_gain* according to which state it operates in to control whether to probe for more available bandwidth, or drain the buffer at a bottleneck link, or cruise at the speed of *BtlBW*.

$$\begin{aligned} cwnd &= cwnd\_gain \times BDP \\ pacing\_rate &= pacing\_gain \times BtlBW \end{aligned} \tag{4.1}$$

As illustrated in Figure 4.2, there are four states in the BBR life-cycle. Firstly, BBR starts with the Startup state, in which it exponentially increases the cwnd size and sending rate by setting *pacing_gain* and *cwnd_gain* to $2/ln(2)$. BBR transits to the Drain state if it is in the plateau of *BtlBW* for three RTTs. In the Drain state, BBR reduces *pacing_gain* to $ln(2)/2$ to drain the standing queue at the bottleneck link induced during Startup. After the above two stages, BBR has successfully built the path model, with *RTprop* measured at the beginning of Startup and *BtlBW* measured at the end of Startup. After that, BBR switches to a steady phase where BBR alternatively runs in the ProbeBW and ProbeRTT state. During the ProbeBW state, BBR sets *pacing_gain* to 1.0 to cruise at the Kleinrock's optimal point for 6 cycles, then sets *pacing_gain* to 1.25 to explore more bandwidth for 1 cycle, and thereafter sets *pacing_gain* to 0.75 to drain the possible standing queue for 1 cycle. It is noteworthy that BBR always sets *cwnd_gain* to 2.0 during the ProbeBW state to allow sending an extra BDP-sized data during loss recovery. If BBR discovers that its *RTprop* has not been updated for 10 s, it enters the ProbeRTT state. During the ProbeRTT state, BBR reduces its inflight

size to 4× MSS (Max Segment Size) and waits for max$\{RTT, 200\ ms\}$ to obtain an updated value of *RTprop*.

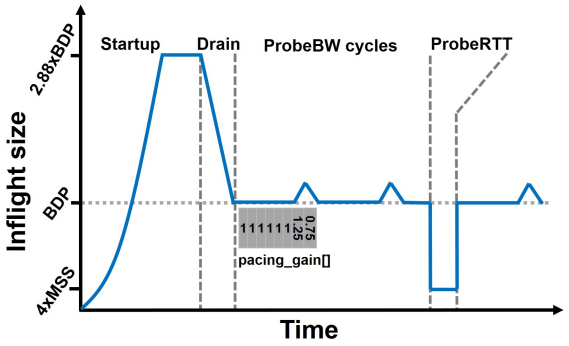


Figure 4.2 The illustration of the BBR life-cycle.

4.2.2 BBRv2 overview

As observed in numerous previous studies [14, 56, 70, 111, 127], BBR has two key issues: unfairly sharing bandwidth with loss-based CCAs and high retransmission rates in shallow-buffered networks. The reasons behind these issues are that BBR is congestion signal agnostic and is over-aggressive when probing for more bandwidth. To mitigate the problems above, Google proposed BBRv2 [18], which inherits most of BBR's design yet redesigns the ProbeBW state, as shown in Figure 4.1.

To avoid being congestion signal agnostic like BBR, BBRv2 reacts to packet loss and DCTCP-style ECN marks [8] for building a tight estimation of the capacity of a bottleneck link. Specifically, BBRv2 adds three variables (*inflight_lo*, *inflight_hi*, and *bw_lo*). *inflight_lo* and *bw_lo* are the short-term lower bounds of inflight size (i.e. *cwnd*) and sending rate respectively, in order to capture the temporary status of network paths (e.g. cross-traffic takes a share of capacity); *inflight_hi*, on the other hand, is the long-term upper bound of *cwnd* to reduce the likelihood of packet losses due to bandwidth probing.

BBRv2 calculates *pacing_rate* and *cwnd* as in Equation (4.2). The reason for having non-zero headroom values is to reserve spare capacity for other flows to quickly grab some bandwidth, improving its friendliness [45]. As bounded by *inflight_lo* and *inflight_hi*, the estimation of *cwnd* in BBRv2 is more conservative than that of BBR.

$$
\begin{aligned}
pacing\_rate &= pacing\_gain \times \min(BtlBW, bw\_lo) \\
cwnd &= \min((1 - headroom) \times inflight\_hi, \\
&\qquad inflight\_lo, cwnd\_gain \times BDP) \\
headroom &= \begin{cases} 0.15, & ProbeCruise\ or\ ProbeRTT \\ 0, & Otherwise \end{cases}
\end{aligned} \tag{4.2}
$$

The BBRv2 life-cycle is shown in Figure 4.3. Firstly, besides the condition that BBR uses, BBRv2 also exits from Startup if the current loss/ECN mark rate is above a predefined threshold (e.g. 2% for the current loss rate). If BBRv2 exits from Startup because of a high loss/ECN mark rate, it sets *inflight_hi* to the current inflight size, otherwise, *inflight_hi* remains at its initial value ($+\infty$). Secondly, BBRv2 decomposes BBR's ProbeBW state into four sub-states, ProbeCruise, ProbeRefill, ProbeUp, and ProbeDown.

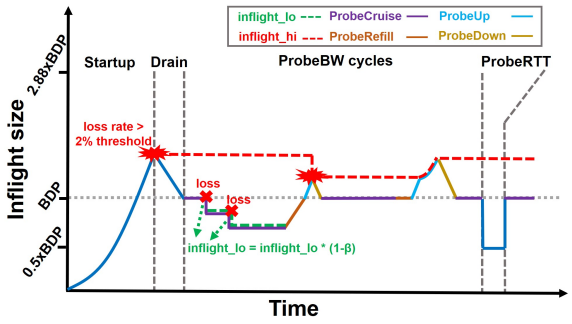


Figure 4.3 The illustration of BBRv2 life-cycle.

ProbeCruise: In the ProbeCruise state, BBRv2 sets *pacing_gain* to 1.0. If any loss or ECN mark occurs during this state, BBRv2 updates *inflight_lo* and *bw_lo* to $\max\{(1-\beta) \times inflight\_lo, BtlBW_{curr}\}$ and $\max\{(1-\beta) \times bw\_lo, inflight_{curr}\}$ respectively, where $BtlBW_{curr}$ and $inflight_{curr}$ are the current measurements of bandwidth and inflight size. It is noteworthy that β is either pre-configured[4] or calculated based on ECN mark rates.

ProbeRefill: When BBRv2 has been in ProbeCruise for a period of T (T is determined in ProbeDown), BBRv2 transits to ProbeRefill, by setting *inflight_lo* and *bw_lo* to $+\infty$ to refill the "pipe" with BDP-sized inflight data, which lasts for one RTT. The goal of this state is to avoid early packet losses before the capacity is fully utilized in shallow-buffered networks since BBRv2 will accelerate in the following ProbeUp state.

ProbeUp: In ProbeUp, BBRv2 sets *pacing_gain* to 1.25 to probe for more available bandwidth. If BBRv2 has increased its inflight size all the way up to *inflight_hi* without

[4]The default value 0.3 is for BBRv2 to co-exist with Cubic [45]

seeing a loss/ECN mark rate exceeding the threshold, it starts to exponentially increase *inflight_hi*. This state ends either when the current loss/ECN mark rate exceeds the threshold, or when the inflight size reaches 1.25× BDP for at least one *RTprop*. In the former case, BBRv2 sets *inflight_hi* to the current inflight size.

ProbeDown: In ProbeDown, BBRv2 drains the potential queue at the bottleneck link by setting *pacing_gain* to 0.75. BBRv2 also sets the duration (T) for the next ProbeCruise state to $\min\{rand(2,3), \frac{BDP}{MSS} \times RTT\}$ s, where $rand(2,3)$ means a number between two and three. The intention of T is to match the interval between loss recovery epochs of Reno for TCP fairness. It is noteworthy that T is usually longer than the bandwidth probing interval of BBR ($8 \times RTprop$). Therefore, BBRv2 probes for bandwidth in a more-conservative way. The ProbeDown state ends when BBRv2 cuts its inflight size below $\min\{BDP, 0.85 \times inflight\_hi\}$. Thereafter, BBRv2 transits to the next ProbeCruise state.

Moreover, BBRv2 no longer uses a 10-RTT-windowed *max_filter* for the estimation of *BtlBW*; rather, it takes the maximum bandwidth measured in the recent two ProbeBW stages as the estimated *BtlBW*.

Summary of BBRv2: BBRv2 is more conservative in *cwnd* estimation and bandwidth probing than BBR, in order to reduce the excessive retransmissions in shallow-buffered networks and improve the fairness of BBR. However, incorporating loss or ECN information in BBRv2 degrades its performance under random losses. More importantly, because of less frequent bandwidth probing in BBRv2, it may not respond to bandwidth dynamics as quickly as BBR. We will quantify the benefits and shortcomings of BBRv2 in the next section.

4.3 A deep dive into BBRv2

In this section, we empirically investigate the improvements and overheads of BBRv2, in comparison with BBR. We especially pay attention to the performance of BBRv2 under bandwidth variations, jitter, and random packet losses that are very common in Mobile Internet scenario. Our key observations include: **(1)** BBRv2 improves inter-protocol fairness and RTT fairness, and also reduces retransmissions in shallow-buffered networks; this observation reaffirms the findings in the previous studies [17, 44, 66, 90, 114]; **(2)** the improvements of BBRv2 come with the cost of low resilience to random loss and slow responsiveness to bandwidth dynamics. That said, it fails to achieve a balance between the aggressiveness in bandwidth probing and the fairness against loss-based CCAs; **(3)** like BBR, BBRv2 experiences low throughput in high-jitter networks because of underestimation of BDP.

4.3.1 Methodology

We rely on Mininet [88] that runs on a server with 8 Intel Xeon Platinum cores and 32GB memory for empirical analysis. The operating system is Ubuntu 18.04.5 with BBR and BBRv2 [45] installed. Figure 4.4 shows the network topology. Linux *tc-netem* [53] is used to emulate different network conditions by varying buffer size, link speed, delay, random loss rates, jitter. We use *iPerf3* [35] to generate TCP traffic between senders and receivers. All packets are captured via *tcpdump* [118] and analyzed by *tcptrace* [96] to get various performance metrics (e.g. RTTs, throughput, retransmissions, inflight bytes). Besides, the values of internal variables (e.g. *cwnd*, *pacing_rate*, *RTprop*, *BtlBW*) in BBR and BBRv2 are recorded in Linux kernel logs. Finally, the queue length at the bottleneck link between R2 and R3 is reported by *tc* [58]. As in previous studies [14, 56, 111] did, each set of experiments is repeated five times and the average results (along with variations if possible) are reported.

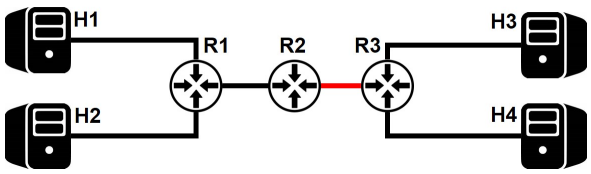


Figure 4.4 Mininet testbed: the bottleneck link is the link between R2 and R3 (marked in red); delay and packet loss are configured on the link between R1 and R2; network jitter is configured at R3's outgoing interfaces towards H3 and H4.

4.3.2 Fairness

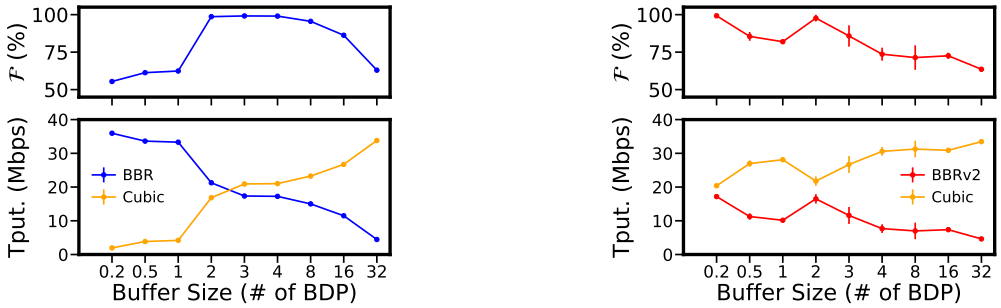
We investigate two types of fairness: the inter-protocol fairness against loss-based CCAs and RTT fairness. In this set of experiments, two flows start simultaneously, one from H1 to H3 and the other from H2 to H4; both flows last for three minutes for the throughput convergence. The bottleneck bandwidth was fixed at 40 Mbps without network jitter or random packet losses.

We use Jain's fairness index [37] ($\mathscr{F}$) of the two flows to evaluate the fairness. The index is calculated as $\mathscr{F} = [(T_1 + T_2)^2]/[2 * (T_1^2 + T_2^2)]$, where T_i is the i-th flow's average throughput over the last 60 s of the flow's lifetime. $\mathscr{F} = 1$ indicates the maximum fairness where two flows have identical average throughput, and $\mathscr{F} = 0.5$ represents that one flow's throughput is zero and the fairness is minimized. We also vary the bottleneck buffer size as it also impacts fairness results.

Inter-protocol fairness

In the experiment, the flow from H1 to H3 uses either BBR or BBRv2, and that from H2 to H4 uses Cubic, which is the default CCA in Linux. The RTTs of the two paths are 40 *ms*.

Figure 4.5 shows the inter-protocol fairness results for both BBR and BBRv2, where we vary the buffer size (expressed in multiples of BDP) at the bottleneck link. We can observe that compared with BBR, BBRv2 significantly improves Jain's fairness index when the buffer is shallow (i.e., less than 2× BDP). This is due to the fact that BBRv2 reacts to losses caused by buffer overflow and bounds the inflight size by using *inflight_hi* and *inflight_lo*. When the bottleneck buffer gets deeper, Cubic obtains more bandwidth than BBR or BBRv2. This is because that the inflight size of both BBR and BBRv2 is upper-bounded by about 2× BDP, while Cubic's inflight size can go beyond this value under deep buffers. As the two flows experience similar RTTs, a larger inflight size means higher throughput. We also note that BBRv2 is less competitive than BBR under moderate buffers. The reason lies in the fact that BBRv2 is more conservative in bandwidth probing and *cwnd* estimation.



(a) BBR vs Cubic

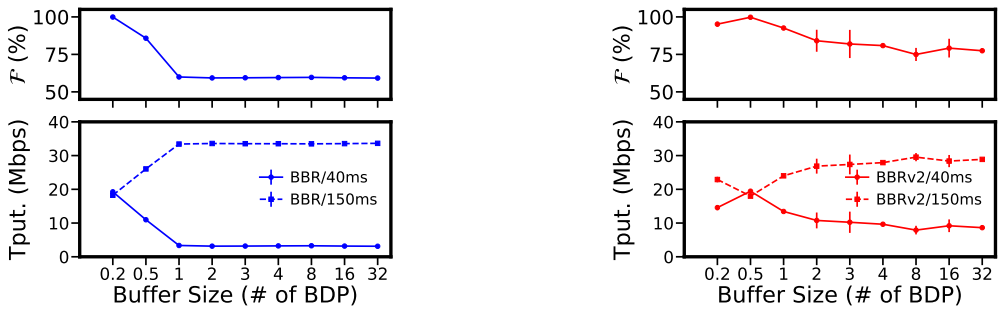
(b) BBRv2 vs Cubic

Figure 4.5 Inter-protocol fairness of BBR and BBRv2 under different buffer sizes (markers: averages, bars: standard deviations).

RTT fairness

In this experiment, two flows use the same CCA, either BBR or BBRv2. The RTT is 40 *ms* for the path between H1 and H3, and 150 *ms* for the path between H2 and H4.

We can see from Figure 4.6 that when the buffer is quite shallow (i.e., 0.2× BDP), BBR has good RTT fairness. However, when the buffer size becomes larger, the BBR flow with longer RTT gradually occupies all the bandwidth and starves the flow with shorter RTT. The reason for the poor RTT fairness of BBR is well documented by the previous studies [56, 82]. Specifically, the aggregated sending rate of two BBR flows exceeds bottleneck bandwidth because of bandwidth probing. As such, a persistent queue forms at the bottleneck link.



(a) BBR (b) BBRv2

Figure 4.6 RTT fairness of BBR and BBRv2 under different buffer sizes (markers: averages, bars: standard deviations).

As the *cwnd* of BBR is proportional to *RTprop*, the flow with longer RTT pours more data into the bottleneck buffer, thus having a larger share of the bottleneck link's capacity. This problem is not severe under shallow buffers, as excess packets are mostly dropped instead of forming a persistent queue.

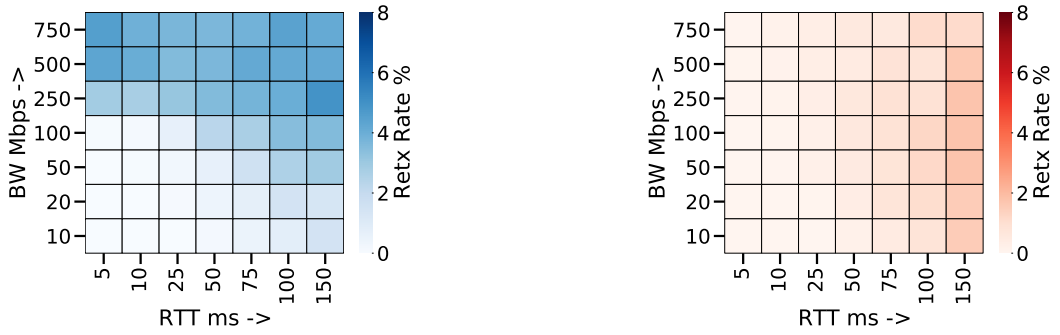
Compared with BBR, BBRv2 has better RTT fairness, especially under deep buffers. The reason is three-fold. First, when the buffer size is moderate, losses are triggered due to buffer overflow, and then both flows reduce their *cwnd* proportional to BDP. As the flow with longer RTT has a larger BDP, it reduces its *cwnd* more than the flow with shorter RTT. Second, when BBRv2 flows are cruising at the speed of *BtlBW*, they always try to leave headroom for other flows to explore the bandwidth. Third, BBRv2 flows yield occupied capacity more frequently because it enters the ProbeRTT state more often than BBR.

4.3.3 Retransmission and throughput

As one of BBRv2's design goals is to reduce unnecessary retransmissions in shallow-buffered networks, next we investigate whether BBRv2 achieves this design goal. The experimental setup is similar to that in the previous work [14], where the bottleneck bandwidth varies in 10 – 750 Mbps and the path RTT varies in 5 – 150 *ms* as these values are commonly employed in modern networks [14, 56, 59]. The buffer size at the bottleneck link is set to 100 KB to emulate shallow-buffered networks[5]. For each setup, we launch a TCP flow for 30s and record its retransmission rate.

The heatmaps in Figure 4.7 show the retransmission rates of BBR and BBRv2 under various network conditions. We can observe that the retransmission rate of BBRv2 is

[5] 100 KB is less than the BDP of most bandwidth-RTT combinations in our setup



(a) BBR (b) BBRv2

Figure 4.7 The heatmap of the retransmission rate of BBR or BBRv2 under various network conditions.

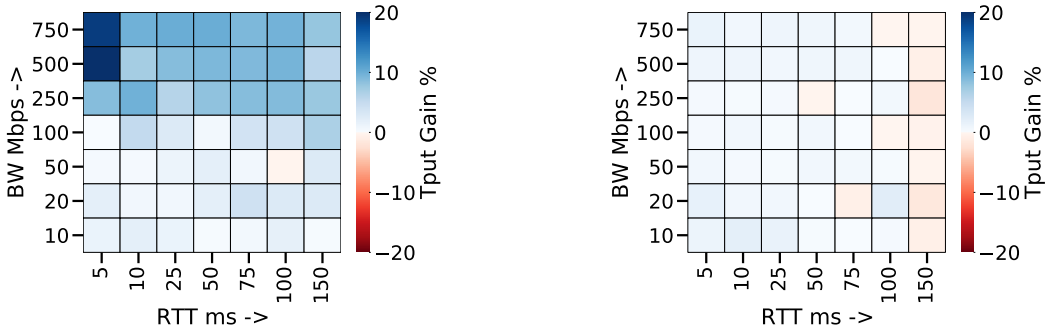
significantly reduced compared with that of BBR, especially when the BDP is larger than 400KB (i.e. the buffer size $\leq 0.25\times$ BDP).

The lower retransmission rate of BBRv2 in shallow-buffered networks stems from the fact that BBRv2 reacts to packet losses, while BBR does not. In the Startup and ProbeUp state, BBRv2 tries to send at a rate higher than the bottleneck bandwidth, which leads to excessive losses (i.e. loss rate $\geq 2\%$). The excessive losses trigger *inflight_hi* to be set as the current inflight size that is likely close to BDP in shallow-buffered networks. As shown in Equation (4.2), BBRv2's *cwnd* is bounded below $0.85\times$ *inflight_hi* in ProbeCruise. Since BBRv2 flows spend most of their time in ProbeCruise, the average throughput of BBRv2 is expected to be 15% lower than the available bandwidth in shallow-buffered networks.

That said, BBRv2 trades off throughput against retransmission in shallow-buffered networks. To verify this, we compute the throughput gain of BBR over BBRv2 (*Tput_Gain*), which is defined in Equation (4.3), where $Tput_{BBR}$ (resp. $Tput_{BBRv2}$) is the average throughput of a BBR (resp. BBRv2) flow over 30 s.

$$Tput\_Gain = \frac{Tput_{BBR} - Tput_{BBRv2}}{Tput_{BBRv2}} \tag{4.3}$$

From Figure 4.8a, we can observe that under the network conditions where BBRv2 reduces the retransmission rate (when BDP exceeds 400 KB), BBRv2 achieves lower throughput than BBR. Specifically, the throughput of BBRv2 is 13% – 16% lower than that of BBR in these cases. In deep-buffered networks, however, packet losses are much less often. It is thus expected that the throughput of BBR and BBRv2 are comparable. This is confirmed by the results in Figure 4.8b, where the buffer size is configured at 10 MB, larger than the BDP of most of the bandwidth-RTT combinations in our setup. The throughput differences between BBR and BBRv2 are indeed marginal in these networks.



(a) 100 KB buffer

(b) 10 MB buffer

Figure 4.8 The heatmap of *Tput_Gain* (in percentage): (a) shallow-buffered network, (b) deep-buffered networks.

4.3.4 Resilience to random losses

Several early tests [17, 66, 114] have shown that BBRv2 is less resilient to random losses than BBR, since BBRv2 limits its *cwnd* by *inflight_lo* and *inflight_hi*, which both react to all types of losses. In BBRv2, there are two parameters that decide how *inflight_lo* and *inflight_hi* react to losses. One is the explicit loss threshold (α) and the other one is the *inflight_lo* reduction factor (β). To study how α and β impact BBRv2's loss resilience, we investigate BBRv2 variants with different α and β under random losses, where each specific BBRv2 variant is referred as BBRv2(α, β). For α, we cap it at 20% to match the maximum loss rate that BBR can tolerate; for β, we only evaluate the difference between the case with (i.e. $\beta = 0.3$ by default) and without it (i.e. $\beta = 0$). In this set of experiments, the bottleneck bandwidth is set to 40 Mbps, and the path RTT is 40 ms. The buffer size is 32× BDP to avoid packet losses due to buffer overflow. The random loss rate ranges from 0% to 30%.

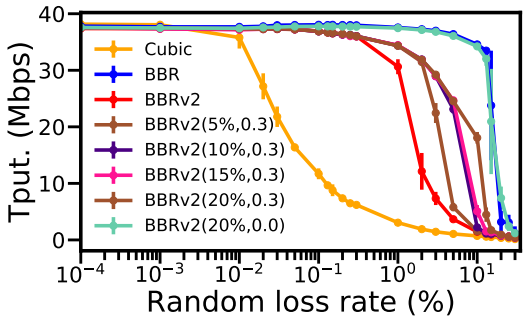


Figure 4.9 Average throughput against different random loss rates (buffer size = 32× BDP).

Figure 4.9 reports the average throughput of each CCA against random loss rates. We observe that the throughput of BBRv2 drops significantly when the random loss rate goes

beyond 2%. Moreover, as α increases, the loss resilience of BBRv2 is improved. For instance, with a 10% random loss rate, BBRv2(20%, 0.3) occupies around half of the maximum bandwidth while the default BBRv2's throughput nearly drops to zero. The impact of β is also remarkable: the loss resilience of BBRv2(20%, 0.3) is lower than that of BBRv2(20%, 0), which performs similar to BBR.

The above results indicate that BBRv2's loss resilience can be improved via raising α. Yet, there is a concern on its effectiveness in alleviating the retransmission issue in shallow-buffered networks, because BBRv2 reduces its *cwnd* when the loss rate exceeds α (see § 4.3.3).

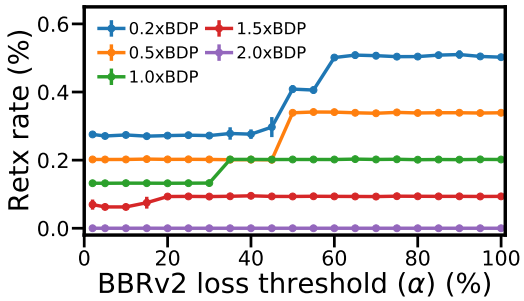


Figure 4.10 Retransmission rates vs α values under networks with different buffer sizes. The errorbars in the figure represent the standard deviations of retransmission rate. β is fixed at 0.3 for all experiments; the default value of α in BBRv2 is 2% (the first data point of every line).

To investigate the aforementioned concern, we extend the experiments by considering more BBRv2 variants ($\alpha \in [2\%, 100\%]$, $\beta = 0.3$) and more configurations on buffer sizes (buffer size $\in \{0.2, 0.5, 1.0, 1.5, 2.0\}\times$ BDP). Figure 4.10 plots the retransmission rates of all considered BBRv2 variants under a random loss rate of 0% (to eliminate the impact of random losses on retransmission rates). Two observations are notable. Firstly, we observe that the retransmission rate increases when α exceeds a certain point, which depends on the bottleneck buffer size. α values beyond the turning points are too high to be reached by temporary loss rates, thus, limiting the effect of *inflight_hi*. Secondly, if the buffer size is large enough (i.e. $2\times$ BDP in our experiments), the retransmissions are eliminated, so the value of α becomes irrelevant.

Another concern about lifting α is the impact on the inter-protocol fairness because the larger α is, the slower reaction of BBRv2 to losses is, which makes BBRv2 more aggressive to loss-based CCAs. To investigate this concern, we test the inter-protocol fairness of BBRv2(20%, 0.3) using the same setup as in § 4.3.2, and plot the results in Figure 4.11. In comparison with Figure 4.5b, the inter-protocol fairness of BBRv2 is indeed worsened in the case of extremely shallow buffer ($0.2\times$ BDP) due to the increased aggressiveness caused by

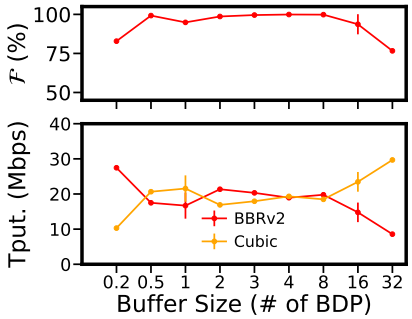


Figure 4.11 Inter-protocol fairness of BBRv2(20%, 0.3) (markers: averages, bars: standard deviations).

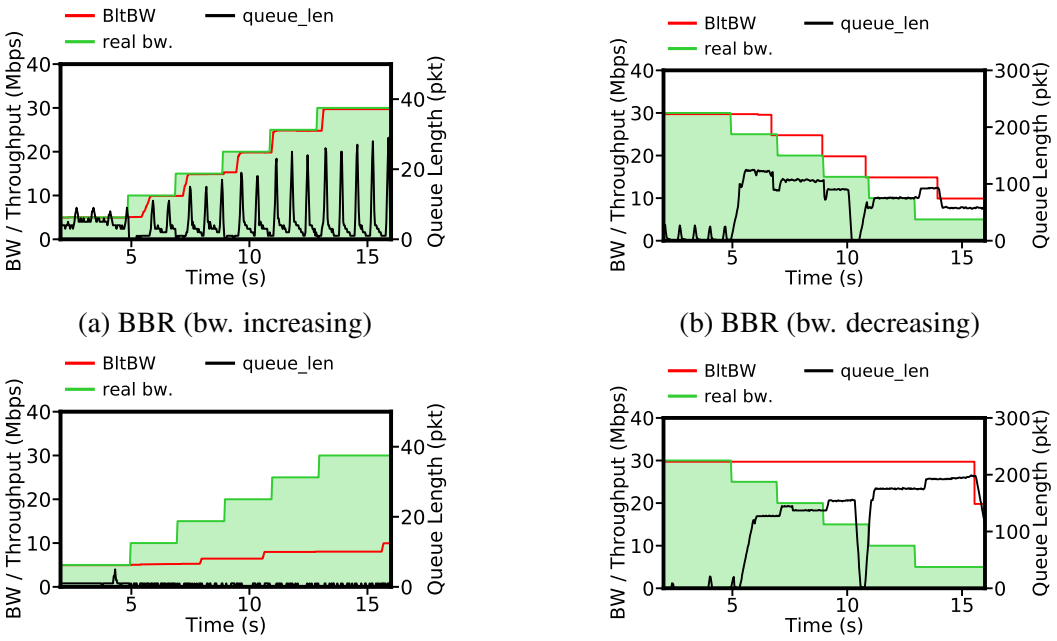
a larger α. Nevertheless, the fairness index is improved under moderate buffers because the increased aggressiveness also makes BBRv2 less vulnerable to Cubic when the bottleneck buffer becomes larger.

Summary of random loss resilience: The loss resilience of BBRv2 can be improved by raising the loss threshold, α. Nevertheless, α should be carefully tuned according to the bottleneck buffer size in order to avoid increasing retransmissions and being too aggressive to loss-based CCAs in extremely shallow-buffered networks.

4.3.5 Responsiveness to bandwidth dynamics

Next, we investigate BBRv2's responsiveness to bandwidth changes. The experiments are designed as follows. The bandwidth of the bottleneck link is configured to increase or decrease 5 Mbps every 2 *s*, the path delay is set to 40 *ms*, and the buffer size is set to 32× BDP. The internal variables during flow transmission (including *pacing_rate* and *BtlBW*, instantaneous throughput, and the queue length at the bottleneck link) are sampled at an interval of 100 *ms*.

Figure 4.12 shows how BBR and BBRv2 adapt to bandwidth increases or decreases respectively. The red line represents the *BtlBW* estimation of BBR/BBRv2, and the green line indicates the real bandwidth of the bottleneck link. The dark line shows the dynamics of the bottleneck link's queue length. Note that the spikes of queue length in Figure 4.12a and Figure 4.12c are caused by the periodical bandwidth probing of BBR and BBRv2, and the sudden drops of queue length around 10 *s* in Figure 4.12b and Figure 4.12d are because of probing minimum RTT values. We can observe that BBRv2 is less effective than BBR in terms of responsiveness to bandwidth dynamics, resulting in the low utilization of bandwidth and long queuing delay.



(a) BBR (bw. increasing)

(b) BBR (bw. decreasing)

(c) BBRv2 (bw. increasing)

(d) BBRv2 (bw. decreasing)

Figure 4.12 Responsiveness to bandwidth increases (a, c) and decreases (b, d).

As we discussed in § 4.2.2, to match the interval between two Reno loss recovery epochs for better inter-protocol fairness, BBRv2 sets its probing interval to $\min\{rand(2,3), \frac{BDP}{MSS} \times RTT\}$ s. This interval can be tens of RTTs, which is too conservative in dynamic environments. That said, BBRv2 improves inter-protocol fairness, at the cost of poorer responsiveness to bandwidth dynamics.

4.3.6 Resilience to network jitter

In Mobile Internet scenarios, high jitter is not uncommon in many widely deployed wireless networks, e.g. WiFi and 5G networks operating in mmWave band [23, 70, 148], and cellular networks [10, 70], especially when high-mobility involves such as high-speed rails [124]. Several works [70, 124] have shown that throughput collapse occurs when BBR operates in high-jitter networks. Therefore, it is interesting to investigate whether BBRv2 operates well in networks with high jitter.

In this experiment, the bottleneck bandwidth is 40 Mbps and the path RTT is 40 *ms*. The bottleneck buffer size is set to 32× BDP to avoid buffer overflow. To emulate jitter, *tc* is used to add jitter following Gaussian distribution at R3's interface that connects to H3. The mean value of the Gaussian distribution varies from 0 – 120 *ms* to emulate different degrees of jitter.

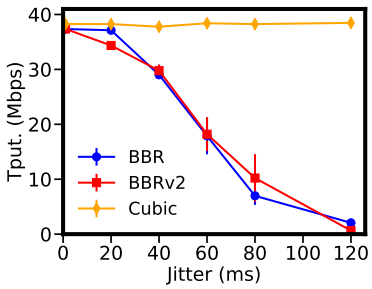


Figure 4.13 Average throughput against different levels of jitter. $X = 0$ is equivalent to no jitter.

Figure 4.13 shows the average value and the standard deviation of throughput of Cubic, BBR, and BBRv2 under various levels of jitter. Compared with Cubic, both BBR and BBRv2 experience low throughput under high jitter. As documented by Kumar et al. [70], BBR underestimates *RTprop* in such networks because it uses the minimum RTT of last 10 s to approximate *RTprop*, which causes *cwnd* exhaustion. This problem still exists in BBRv2, even if BBRv2 updates *RTprop* 2× frequently than BBR (i.e. BBRv2 uses the minimum RTT in recent 5 s to estimate *RTprop*). We also note that the significant throughput degradation starts when the average jitter reaches the path RTT (i.e. 40 *ms*).

4.3.7 Summary and implication

We observe that BBRv2 improves inter-protocol fairness and RTT fairness, and reduces retransmission rates under shallow buffers, at the cost of slow responsiveness to bandwidth dynamics and low resilience to random losses.

First, the root cause for the slow responsiveness is that BBRv2 is over-conservative in bandwidth probing. That said, it fails to balance between the aggressiveness in probing for more bandwidth and the fairness against loss-based CCAs. However, recklessly increasing BBRv2's aggressiveness in bandwidth probing may lead BBRv2 to generate overwhelming retransmissions and unfairly share bandwidth with loss-based CCAs. In the next section, we propose BBRv2+, which incorporates delay information to cautiously guide the aggressiveness of bandwidth probing to avoid reducing the fairness against loss-based CCAs. The challenge is how to effectively use this signal and how to avoid being suppressed by other loss-based CCAs in deep-buffered networks as other delay-based CCAs.

Second, the resilience to random losses can be improved by raising the loss threshold, α, where the value of α needs to be carefully set according to the bottleneck buffer size.

Last, the throughput degradation of BBR and BBRv2 in high-jitter networks is owned to the underestimation of *RTprop*, which in turn leads to a smaller estimation of BDP. We

propose a compensation mechanism of BDP that makes the estimated BDP close to the real BDP.

4.4 Design and implementation of BBRv2+

Table 4.1 BBRv2+'s variables capturing delay information and network jitter.

Variable	Functionality
$\text{MinRTT}_{\text{prev\_rtt}}$	The minimum RTT measured in the previous RTT round.
$\text{MinRTT}_{\text{curr\_rtt}}$	The minimum RTT measured in the current RTT round.
$\text{MinRTT}_{\text{before\_probe}}$	Saving the $\text{MinRTT}_{\text{curr\_rtt}}$ before entering ProbeUp.
$\text{MinRTT}_{\text{curr\_cruise}}$	The minimum RTT measured in the current ProbeCruise state.
$\text{Max}_{4\text{RTT}}(\text{jitter})$	The *max_filter* tracking the maximum jitter in recent four RTT rounds. Here, the jitter is equivalent to the RTT variation maintained by TCP.

Motivated by our measurement results, we design and implement BBRv2+, in order to address the pitfalls of BBRv2 to improve the performance in Mobile Internet scenarios while maintaining its advantages over BBR (i.e. improved fairness and reduced retransmissions in shallow-buffered networks). The basic idea is to incorporate delay information in BBRv2+'s path model to balance between the aggressiveness in probing for more bandwidth and the fairness against loss-based CCAs. BBRv2+ also compensates the estimated BDP in case of high jitter for improved performance.

4.4.1 Overview

The architecture of BBRv2+ is shown in Figure 4.14. BBRv2+ incorporates delay information in its path model. Specifically, the delay information consists of three state variables (the first three variables listed in Table 4.1) of minimum RTTs, which reflect the change of queuing delay over time. The delay information facilitates quick responsiveness to bandwidth dynamics. Particularly, a new sub-state, ProbeTry, is added into the ProbeBW state. In ProbeTry, BBRv2+ slightly speeds up to examine if this acceleration will lead to increased RTTs. In the case of increased RTTs, BBRv2+ quits this probing and moves to the ProbeDown state to drain the queue at the bottleneck link; otherwise, it moves to the ProbeUp state to further explore available bandwidth. BBRv2+ also uses the delay information to quickly adapt to bandwidth decreases; it quickly updates its bottleneck bandwidth estimation to the current bandwidth measurement if an obvious increase of RTT is observed when BBRv2+ is not probing for bandwidth.

Like other CCAs that use delay-based signals, BBRv2+ will be suppressed when co-existing with loss-based CCAs under deep buffers [7], as the loss-based CCAs constantly fill the buffer at a bottleneck link, leading BBRv2+ to falsely yield up obtained bandwidth. BBRv2+ uses a dual-mode mechanism that forces BBRv2+ to use BBRv2's state-machine when loss-based CCAs co-exist.

Finally, BBRv2+ uses a BDP compensation mechanism to address the *cwnd* exhaustion problem caused by network jitter. Our key observation is that in high-jitter networks, BDP will be underestimated because of the underestimation of *RTprop*. The mechanism compensates BDP by taking the recent RTT variations into consideration; this compensation mitigates the underestimation issue significantly.

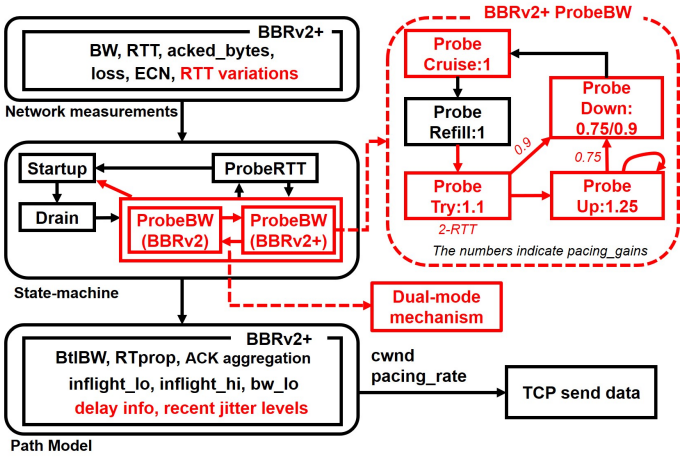


Figure 4.14 BBRv2+ architecture. The parts that differ from BBRv2 are highlighted in red color.

4.4.2 Redesign of the ProbeBW state

In the case of bandwidth increments, BBRv2+ needs to start probing for more bandwidth quickly instead of spending time cruising at the speed of the current estimated *BtlBW*. Thus, the probing interval needs to be reasonably shortened, which is set to approximately match the probing interval of BBR (8 rounds of RTT). However, if BBRv2+ already sends data at the speed close to bottleneck bandwidth, the probing interval above may result in increased packet losses.

BBRv2+ introduces a two-step probing mechanism by incorporating the delay information as shown in Figure 4.14. A new sub-state ProbeTry, which lasts for two RTTs, is inserted before entering ProbeUp in the state machine. In the first RTT of ProbeTry, BBRv2+ slightly increases its *pacing_rate* by increasing *pacing_gain* to 1.1. In the second RTT, BBRv2+

reduces *pacing_gain* to 1.0 and monitors if $\text{MinRTT}_{\text{curr\_rtt}}$ is larger than $\gamma \times \text{MinRTT}_{\text{prev\_rtt}}$, where MinRTT is measured on the ACKs for the packets sent in the previous round (see Table 4.1). The rationale of using $\gamma > 1$ is to introduce a relaxing factor to tolerate noises in RTT measurements, where a small γ may lead BBRv2+ to miss some chances to explore bandwidth while a large γ may make BBRv2+ over-aggressive. It is noteworthy that γ is a design parameter and can be tuned by designers[6]. In our current implementation, we set $\gamma = 1.02$, as such a slight increment (2%) of RTT likely comes from the noises of RTT measurements. If $\text{MinRTT}_{\text{curr\_rtt}} > \gamma \times \text{MinRTT}_{\text{prev\_rtt}}$, BBRv2+ transits to ProbeDown with *pacing_gain* as 0.9 to drain the queue accumulated during the first RTT of ProbeTry. Otherwise, it enters ProbeUp to probe for more bandwidth.

To further boost the speed of bandwidth exploration, BBRv2+ also incorporates a continuous probing mechanism based on the delay information. Specifically, at the end of ProbeUp, if the $\text{MinRTT}_{\text{curr\_rtt}} \leq \gamma \times \text{MinRTT}_{\text{before\_probe}}$, BBRv2+ re-enters ProbeUp. The rationale behind this is that there is possibly more free bandwidth capacity as no significant increment of queuing delay arises in the current ProbeUp sub-state.

Algorithm 1: Advance BtlBW *max_filter*

```
Input  :conn: BBRv2+ TCP connection
1 target_rtt ← θ * conn.RTprop
2 should_advance ← (conn.MinRTT_curr_rtt > target_rtt)
3 if should_advance then
4 |   expire_the_oldest_value(conn.BtlBW)
```

We also leverage the delay information to detect bandwidth decrements. If BBRv2+ is in ProbeCruise or ProbeDown, on the receipt of a new ACK in an RTT round, Algorithm 1 is called[7]. The reason that the algorithm is only applicable in ProbeCruise or ProbeDown is to eliminate the impact on delay variations caused by ProbeTry and ProbeUp sub-states. In Algorithm 1, BBRv2+ expires its current *BtlBW* estimation if $\text{MinRTT}_{\text{curr\_rtt}}$ is larger than the recently measured minimum RTT by θ times. θ is a parameter to balance the speed to converge to new bandwidth and the resistance to noises in bandwidth measurements. A small θ may lead to throughput oscillation, while a large θ may reduce BBRv2+'s responsiveness to bandwidth dynamics. We recommend $\theta \in [1.05, 1.15]$ according to our experiences.

[6] All the design parameters of BBRv2+ in our current implementation are exposed to user-space through the `/sys/module` interfaces, enabling designers to change the parameters without recompiling the kernel module.

[7] It is called once at maximum for every RTT round to avoid expiring the *BtlBW* estimation too frequently.

4.4.3 Dual-mode mechanism

Due to the use of delay information to guide the aggressiveness in bandwidth probing, BBRv2+ will be starved by loss-based CCAs under deep buffers, suffering from the similar problem of delay-based CCAs [7]. The root cause is that loss-based CCAs constantly fill the bottleneck buffer, where BBRv2+ falsely treats the increments of RTT as the signals of bandwidth decrements.

As BBRv2+ periodically drains the bottleneck buffer, $\text{MinRTT}_{\text{curr\_cruise}}$ (see Table 4.1), the measured minimum RTT during ProbeCruise, should be close to *RTprop* if no loss-based competitor exists. By comparing $\text{MinRTT}_{\text{curr\_cruise}}$ with the recorded *RTprop* value, BBRv2+ estimates the existence of loss-based competitors. If loss-based competitors co-exist, BBRv2+ switches to use BBRv2's ProbeBW state, which enables BBRv2+ to co-exist with loss-based CCAs in the same way as BBRv2. Further, if the loss-based competitors no longer exist, BBRv2+ returns back to use the redesigned ProbeBW state. Note that the BBRv2+ does not switch to BBRv2's ProbeBW state if the bottleneck buffer is very shallow, because loss-based CCAs cannot bloat the bottleneck buffer and BBRv2+ will not be starved.

Algorithm 2: The dual-mode mechanism

```
   Input  : conn: BBRv2+ TCP connection
 1 if conn.probe_bw_mode = BBRv2+ then
 2     switch_thld ← λ1 * conn.RTprop
 3     if conn.MinRTT_curr_cruise > switch_thld then
 4         conn.buffer_filling++
 5     else
 6         conn.buffer_filling ← 0
 7     if conn.buffer_filling ≥ η1 then
 8         conn.probe_bw_mode ← BBRv2
 9         restart_from_startup(conn)
10 else
11     switch_thld ← λ2 * conn.RTprop
12     if conn.MinRTT_curr_cruise ≤ switch_thld then
13         conn.buffer_empty++
14     else
15         conn.buffer_empty ← 0
16     if conn.buffer_empty ≥ η2 then
17         conn.probe_bw_mode ← BBRv2+
```

The dual-mode mechanism is detailed in Algorithm 2, which runs at the end of ProbeCruise. If the sender is running in BBRv2+'s ProbeBW state and has not seen RTT samples close to *RTprop* for a number of (η_1) successive ProbeCruise sub-states, it switches to

use BBRv2's ProbeBW state and restarts itself from Startup (line 1 – 9). We note that `restart_from_startup(conn)` in line 9 is a heuristic to quickly regain the bandwidth that has been potentially yielded up to loss-based competitors by BBRv2+ recently. If the sender is using BBRv2's ProbeBW state and it has seen low RTTs for η_2 successive ProbeCruise sub-states, it returns back to use BBRv2+'s ProbeBW state because the competitors are most likely gone. We note that the four parameters in Algorithm 2, λ_1, λ_2, η_1, and η_2, are to control the sensitivity of BBRv2+ to the co-existence of loss-based CCAs. In practice, we used 1.1, 1.05, 2, and 4 for λ_1, λ_2, η_1, and η_2 respectively. Nevertheless, these parameters can be easily tuned in user space in our current implementation.

4.4.4 Compensation for BDP estimation

To boost BBRv2+'s performance under high network jitter, BBRv2+ takes network jitter into account when estimating the BDP of a network path. BBRv2+ compensates the BDP estimation with a component proportional to RTT variations when network jitter is high. The compensation mechanism is detailed in Algorithm 3. As instantaneous RTT variations could be very dynamic, to ensure that BBRv2+ can tolerate jitter up to the maximum extent, we use the recently measured maximum RTT variation, $\text{Max}_{4\text{RTT}}(\text{jitter})$ in Table 4.1, as the estimation of recent jitter. When $\text{Max}_{4\text{RTT}}(\text{jitter})$ exceeds $\mu \times RTprop$, the estimated *RTprop* is increased to $RTprop + \text{Max}_{4\text{RTT}}(\text{jitter})$ to mitigate the underestimation of *RTprop*. We recommend to set μ around 0.5 because the performance of BBRv2 starts to degrade when jitter approaches half of *RTprop* as observed in § 4.3.6.

Algorithm 3: Compensating BDP

```
Input  : conn: BBRv2+ TCP connection
Output : the BDP estimation of BBRv2+
1 Function get_compensated_BDP (conn) :
2     jitter ← conn.Max_4RTT(jitter)
3     threshold ← μ * conn.RTprop
4     fixed_RTprop ← conn.RTprop
5     if jitter > threshold then
6         fixed_RTprop ← fixed_RTprop + jitter
7     return conn.BtlBW * fixed_RTprop
```

4.4.5 Implementation

BBRv2+ is implemented as a Linux kernel module (∼2100 LoCs), based on Google's BBRv2 alpha kernel module [45]. The parameters of BBRv2+ (summarized in Table 4.2) are

Table 4.2 Summary of the parameters of BBRv2+.

Parameter	Functionality	Default value
γ	control the aggressiveness of bandwidth probing	1.02
θ	control the sensitivity to the decrease of bandwidth	[1.05, 1.15]
$\lambda_1, \lambda_2, \eta_1, \eta_2$	control the sensitivity to detect if loss-based CCAs coexist	1.1, 1.05, 2, 4
μ	control the sensitivity to network jitter	0.5

exposed to user-space through the `/sys/module` interfaces, which allows users to change the parameters according to their need without recompiling the kernel module.

4.5 Evaluation of BBRv2+

In this section, we evaluate BBRv2+ based on both Mininet-based emulation and real-world trace driven emulation. First, we describe our experiment setup in § 4.5.1. We then evaluate the benefits of BBRv2+ from the perspectives of the responsiveness to bandwidth dynamics (§ 4.5.2) and the resilience to network jitter (§ 4.5.3). Next, we demonstrate that BBRv2+ is able to keep the advantages of BBRv2 in inter-protocol fairness (§ 4.5.4), RTT fairness (§ 4.5.5), and low retransmission rates (§ 4.5.6). Finally, we evaluate the performance of BBRv2+ through real-world trace driven emulation in § 4.5.7.

4.5.1 Evaluation setup

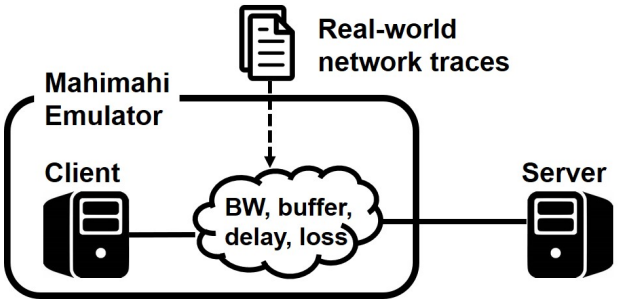


Figure 4.15 Mahimahi testbed: the client connects to the server through the network emulated by Mahimahi.

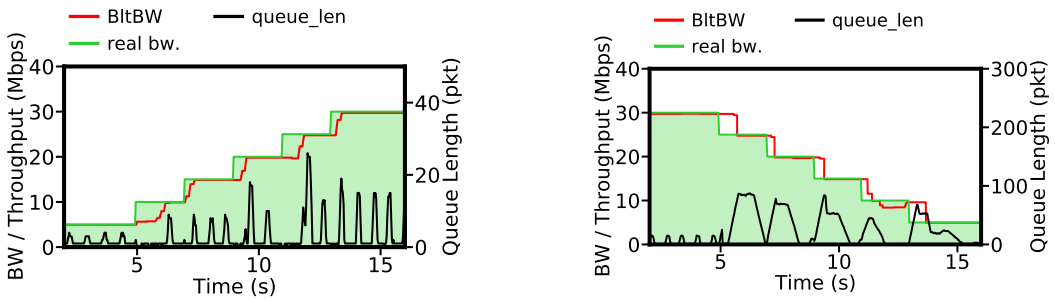
Two testbeds are used for the evaluation of BBRv2+. One is the Mininet-based testbed used in § 4.3, as a controlled environment to evaluate BBRv2+ from various perspectives. The other is based on Mahimahi [92], a trace-driven network emulator that can emulate

networks with dynamic bandwidth by replaying bandwidth traces either collected from real-world networks (e.g. Figure 4.23) or synthesized manually (e.g. Figure 4.17). Other settings of Mahimahi, including delay, packet loss and buffer emulation, are similar to those in Mininet. As shown in Figure 4.15, in the Mahimahi-based testbed, a client connects to a server through the emulated network. The physical server running the two testbeds is the same one as that used to evaluate BBRv2 in § 4.3. The toolset for data analysis and traffic generation is also the same as that in § 4.3.

In all experiments, α (the loss threshold) in BBRv2+ is set to 20% as the value is suitable for most of the buffer sizes according to the results in § 4.3.4. Unless specified otherwise, each set of experiments is repeated five times and the averaged results are reported.

4.5.2 Responsiveness to bandwidth dynamics

We use the same settings as that in § 4.3.5 to have a microscopic view on how BBRv2+ reacts to bandwidth dynamics. Figure 4.16 shows the results.



(a) bw. increasing

(b) bw. decreasing

Figure 4.16 BBRv2+'s responsiveness to bandwidth increases (a) and decreases (b).

In Figure 4.16a, when there is no bandwidth increment, BBRv2+ only enters ProbeTry for a very short duration and finishes bandwidth probing very soon, which leads to an instantaneous short standing queue. However, when the bandwidth is increased, BBRv2+ can timely adapt its *BtlBW* estimation to the real bandwidth. Compared with the results of BBR (Figure 4.12a) and BBRv2 (Figure 4.12c), BBRv2+ is capable of utilizing newly available bandwidth as quick as BBR, while the delay information guided probing strategy reduces queuing delay.

In Figure 4.16b, when bandwidth decreases, BBRv2+ notices that the queuing delay is obviously rising up via increased RTTs (see § 4.4.2). It expires the old *BtlBW* estimation and adapts its *BtlBW* estimation to the available bandwidth. Compared with the results of

BBR and BBRv2 in Figure 4.12, BBRv2+ adapts its sending rate to the decreased bandwidth much faster, which leads to lower queuing delay.

Next, we compare the responsiveness of Cubic, BBR, BBRv2, and BBRv2+ to bandwidth dynamics in the Mahimahi testbed, using five synthesized traces where the bandwidth changes as step functions, as illustrated in Figure 4.17. Following the settings in [2], we set the buffer size to 1.5 MB, the delay to 20 *ms*, and the loss rate to 0%. For each CCA, we conducted experiments in the five emulated networks. In each experiment, a bulk-transfer TCP flow from the server to the client runs for 30 *s* and the flow throughput, as well as the sojourn time of each packet in the buffer at the bottleneck link (denoted as queuing delay), are recorded.

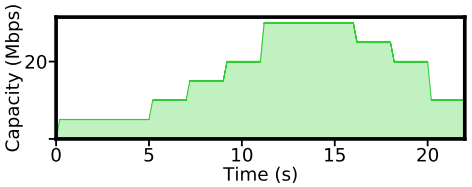


Figure 4.17 An example of synthesized bandwidth traces with bandwidth changing as a step function.

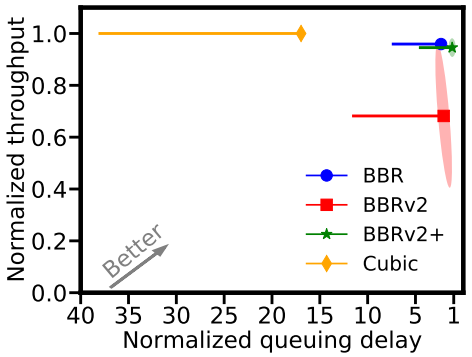


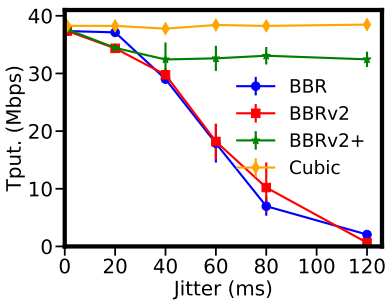
Figure 4.18 Normalized throughput and queuing delay of different CCAs (marker: average throughput and queuing delay, left end of each line: 95%tile of queuing delay, ellipse: standard deviation).

Figure 4.18 compares the performance of all considered CCAs. In the figure, for a CCA, we normalize its average queuing delay (over several runs) to the minimum average queuing delay among all considered CCAs in the same emulated network; we also normalize its average throughput (over several runs) to the maximum average throughput among all considered CCAs. In addition, we report the 95%tile queuing delay of each CCA, which is also normalized to the minimum average queuing delay observed in the same network. Then, the averages and deviations of the normalized values across all emulated networks are plotted. We can see from the figure that, compared with BBRv2, BBRv2+ achieves

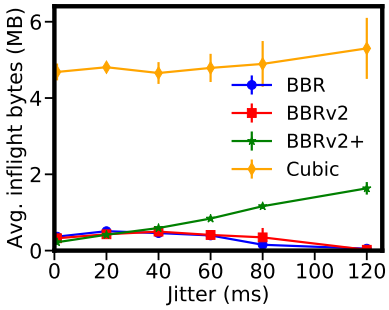
significantly higher throughput and lower queuing delay. On the other hand, compared with BBR, BBRv2+ achieves lower queuing delay at the cost of slightly lower throughput. These observations stem from the facts that: **(1)** BBRv2+ probes for bandwidth at a frequency similar to BBR's one, thus, achieving high bandwidth utilization as BBR does; **(2)** BBRv2+ adapts its sending rate to decreased bandwidth faster than BBR as it quickly updates its *BtlBW* estimation upon increased queuing delay.

4.5.3 Resilience to network jitter

Next, we evaluate how network jitter affects BBRv2+'s performance, using the same settings as in § 4.3.6. The throughput of BBRv2+, Cubic, BBR and BBRv2 under various levels of jitter are shown in Figure 4.19a. Different from BBR and BBRv2, the throughput of BBRv2+ does not degrade when the network jitter becomes larger. Figure 4.19b further plots the average inflight bytes of four CCAs; the results confirm that the BDP compensation mechanism succeeds to compensate *cwnd* in high-jitter networks. Nevertheless, the throughput of BBRv2+ is slightly lower than Cubic. The reason is that our compensation to BDP is conservative; in contrast, Cubic's *cwnd* can grow far beyond the real BDP as it is not affected by network jitter.



(a) Throughput vs jitter



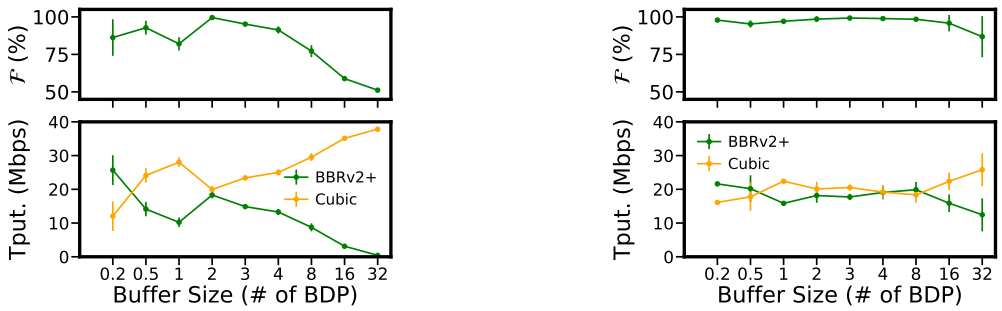
(b) Avg. inflight bytes

Figure 4.19 Resilience to jitter (markers: averages, bars: standard deviations).

4.5.4 Inter-protocol fairness

The inter-protocol fairness of BBRv2+ is evaluated using the same settings as in § 4.3.2. We test BBRv2+ with and without the dual-mode mechanism (see § 4.4.3) to study the impact of this mechanism on inter-protocol fairness. The results are shown in Figure 4.20. Several observations are notable.

Firstly, the results demonstrate the efficacy of the dual-mode mechanism. In Figure 4.20a, we can observe that BBRv2+ without the dual-mode mechanism is starved by Cubic in



(a) W/O the dual-mode mechanism

(b) W/ the dual-mode mechanism

Figure 4.20 BBRv2+: Inter-protocol fairness under different buffer sizes (markers: averages, bars: standard deviations).

deep-buffered cases. This is because BBRv2+ falsely treats the RTT increments caused by Cubic as the signals of bandwidth shrinking. The problem is eliminated by the dual-mode mechanism as shown in Figure 4.20b.

Second of all, compared with the results of BBRv2(20%, 0.3) in Figure 4.11, we can observe that: **(1)** BBRv2+ provides better inter-protocol fairness than BBRv2(20%, 0.3) under an extremely shallow buffer (i.e. 0.2× BDP); **(2)** BBRv2+'s inter-protocol fairness is similar to that of BBRv2(20%, 0.3) under other buffer sizes. The reason for the better inter-protocol fairness of BBRv2+ under an extremely shallow buffer is that BBRv2+ does not enter ProbeUp, which is more aggressive than ProbeTry, due to the two-step probing mechanism (see § 4.4.2) while BBRv2(20%, 0.3) periodically enters ProbeUp.

Lastly, compared with the results of BBR (Figure 4.5a) and BBRv2 (Figure 4.5b), BBRv2+ performs no worse than the better one among BBR and BBRv2 under different buffer sizes. The reasons are three-fold: **(1)** under shallow buffers, BBRv2+ achieves similar inter-protocol fairness to that of BBRv2 thanks to its cautiously aggressive bandwidth probing strategy (see § 4.4.2); **(2)** under moderate buffers, BBRv2+ is close to BBRv2(20%, 0.3) that has better inter-protocol fairness than BBRv2 in these cases as explained in § 4.3.4; **(3)** under deep buffers, the three CCAs perform closely as they all upper bound *cwnd* around 2× BDP, thus, unable to beat loss-based CCAs.

4.5.5 RTT fairness

Next, we evaluate the RTT fairness of BBRv2+ using the same setting as in § 4.3.2. The results are presented in Figure 4.21. Compared with the results of BBR (Figure 4.6a) and BBRv2 (Figure 4.6b), BBRv2+ has better RTT fairness than BBR and behaves close

to BBRv2. The results are expected as the mechanisms in BBRv2 that improves the RTT fairness over BBR (see § 4.3.2) remain unchanged in BBRv2+.

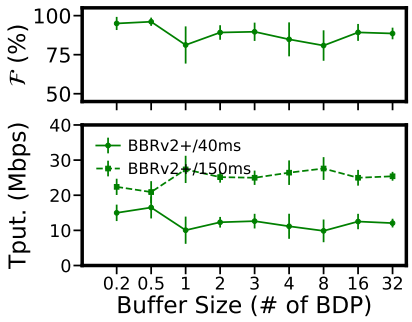


Figure 4.21 BBRv2+: RTT fairness under different buffer sizes (markers: averages, bars: standard deviations).

4.5.6 Retransmissions in shallow-buffered networks

Using the same setting as in § 4.3.3, we evalute the retransmissions in shallow-buffered networks in Figure 4.22. We observe that when the buffer is extremely shallow (e.g. 0.02× BDP when the bandwidth is 500 Mbps and the RTT is 75 *ms*), BBRv2+ incurs more retransmissions than BBRv2. This is because the bandwidth probing frequency of BBRv2+ is higher than that of BBRv2. Although BBRv2+ uses a relatively small *pacing_gain* (1.1), when it starts to probe for more bandwidth in ProbeTry, it still causes buffer overflow if the network buffer is extremely shallow. However, compared with the results of BBR in Figure 4.7a, BBRv2+ reduces retransmissions significantly.

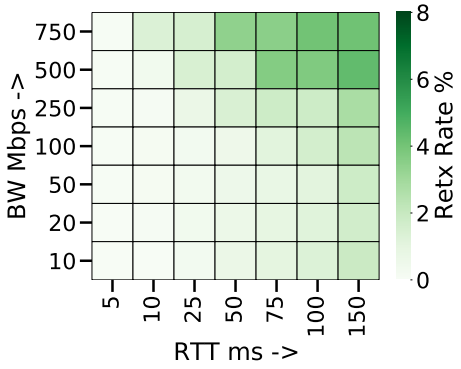
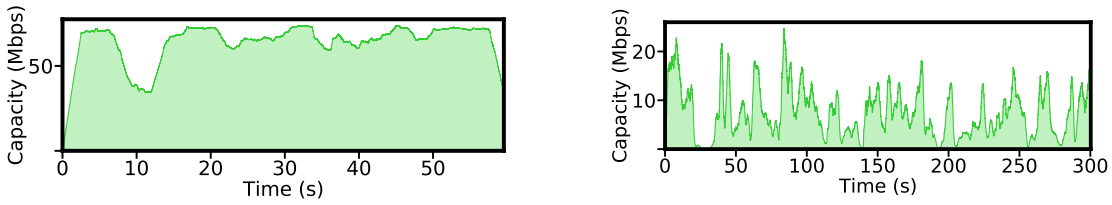


Figure 4.22 BBRv2+: retransmission rates (100 KB buffer).

4.5.7 Real-world trace driven emulation

Finally, we evaluate BBRv2+ in the Mahimahi testbed, where the input bandwidth traces are collected in real-world networks. We compare BBRv2+ with Cubic, BBR, BBRv2, and Orca [2]. Orca[8] is used for comparison as a representative of the state-of-the-art learning-based CCAs.

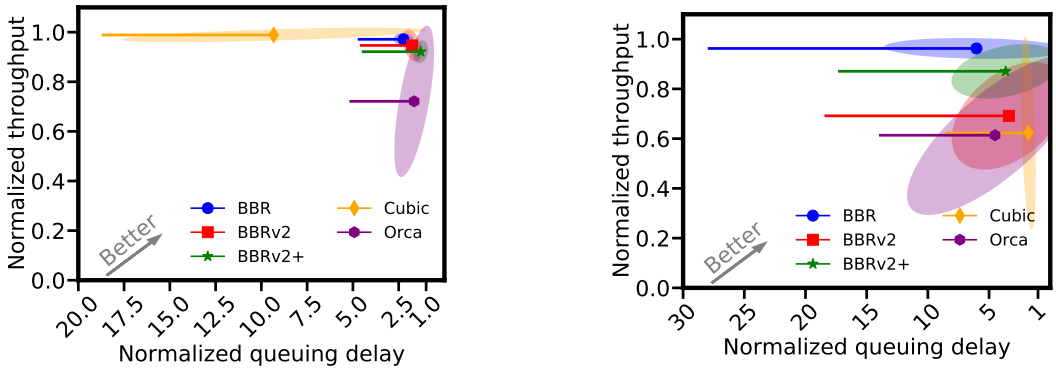
Trace collection: We collected bandwidth traces from WiFi and LTE networks, using *saturatr* [131]. In total, 20 bandwidth traces are collected, half of which were collected when the collector was stationary to the base station (LTE) or the Access Point (WiFi), and the other half were collected when the collector was moving (i.e. in a vehicle or on a high-speed rail). In stationary scenarios, the network bandwidth was usually stable, while in mobile scenarios, the bandwidth fluctuated greatly. Examples of stationary and mobile traces are shown in Figure 4.23. For each bandwidth trace, we also measured the packet loss rate and delay of its corresponding network via *ping*.



(a) An example of stationary traces

(b) An example of mobile traces

Figure 4.23 Examples of bandwidth traces used in our trace-driven evaluation.



(a) Stationary traces

(b) Mobile traces

Figure 4.24 Normalized throughput and queuing delay of different CCAs (marker: average throughput and queuing delay, left end of each line: 95%tile of queuing delay, ellipse: standard deviation of the throughput and queuing delay).

[8]We directly used the model trained by the authors in our experiments.

Experimental setup: Using the collected bandwidth traces along with the information of packet loss rates and delays, we emulate various network environments, where their buffer sizes are fixed at 1.5 MB as the setup in [2]. In each run of the experiments, a bulk-transfer TCP flow from the server to the client runs for 30 *s*.

Results: Using the same metrics as in § 4.5.2 (normalized throughput and queuing delay), the results for the stationary and mobile scenarios are shown in Figure 4.24a and Figure 4.24b respectively.

In stationary scenarios, BBR, BBRv2, and BBRv2+ perform very close to each other because the bandwidth is usually stable. Cubic shows slightly better throughput for most of the time at the cost of high queuing delay. Orca has the lowest throughput probably because the network scenario where the model was trained is different from the emulated network environments in our experiments, which also demonstrates the limitation of learning-based CCAs.

In mobile scenarios, BBR and BBRv2+ achieve the highest and the second-highest throughput respectively. Meanwhile, BBRv2 and Cubic fail to achieve consistently high throughput across different mobile environments. Compared with BBRv2+, BBR achieves higher throughput at the cost of higher queuing delay as it is more aggressive. Orca fails to achieve consistently high throughput and low queuing delay in high-mobility scenarios; the results of Orca raise a concern on the generalization ability of learning-based CCAs.

4.5.8 Summary of experimental results

We can conclude from the above experiments that BBRv2+ succeeds to balance the aggressiveness of bandwidth probing and the fairness against loss-based CCAs. With such a balance, BBRv2+ achieves higher throughput and lower delay than BBRv2 in scenarios where the bandwidth fluctuates, while keeping the advantages of BBRv2 with regard to inter-protocol fairness and reduced retransmissions under shallow buffers. Moreover, the dual-mode mechanism enables BBRv2+ to co-exist with loss-based CCAs under deep buffers and the compensation mechanism for BDP estimation efficiently enhances the performance of BBRv2+ under high network jitter.

4.6 Conclusion

To realize efficient congestion control in Mobile Internet scenarios, it is necessary to comprehensively consider the strong dynamics of network quality brought by mobility (such as large bandwidth changes, high latency jitter, and high packet loss rate) and TCP fairness.

This chapter conducts detailed measurements on the successor of BBR, BBRv2, which is widely used in Mobile Internet scenarios, observes some performance problems, and analyzes the causes of performance problems. On this basis, BBRv2+ is proposed. BBRv2+ successfully balances the aggressiveness of bandwidth probing and the TCP fairness to loss-based congestion control algorithms, resulting in higher throughput than BBRv2 in mobile scenarios, and also retaining BBRv2's advantages in the TCP fairness.

Chapter 5

Data-driven dynamic selection of multipath CCAs

5.1 Introduction

With the continuous development of various mobile devices in Mobile Internet scenarios, more and more devices now support access to the Internet through multiple network interfaces. For example, smartphones usually have both cellular and WiFi network interfaces, connected vehicles can communicate with Internet services via built-in cellular and WiFi network interfaces [72], and the WiFi gateway on high-speed rails can provide Internet access services for passengers through multiple cellular network interfaces [124]. Therefore, for these mobile devices with multiple network interfaces, it is possible to obtain higher bandwidth and better transmission reliability by using multiple network interfaces to access Internet services at the same time.

In order to make full use of the transmission capabilities provided by multiple network interfaces on these mobile devices, multipath transport protocols are indispensable. At present, the most widely used standardized multipath transport protocol is Multipath TCP [38] (MPTCP), which is implemented in iOS and Linux (as the kernel of Android system) operating systems and is widely used in smartphone devices. Currently, MPTCP has been officially merged into the mainstream Linux kernel since Linux 5.6 [61].

As shown in Figure 5.1, in an MPTCP connection, the mobile device and the server establish a few TCP subflows on the multiple network paths existing between them. The sender of the MPTCP connection schedules the data to different subflows for transmission. Then, MPTCP reassembles the data received from different subflows at the receiver. Finally, MPTCP delivers the reassembled data to the application layer.

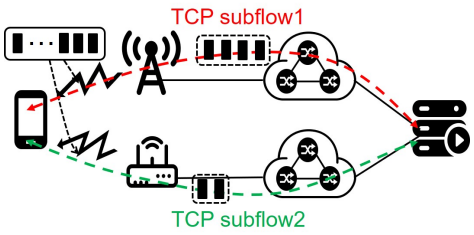


Figure 5.1 An example of data tranmission with MPTCP

In MPTCP, multipath transport involves three key components, namely: path management, packet scheduler, and congestion control. As shown in Figure 5.2, path management determines which paths MPTCP needs to establish TCP subflows on, data scheduling determines how application data is allocated to different subflows, and congestion control determines the rate at which each subflow sends data. Among them, the impact of congestion control on multipath transport performance is of great importance.

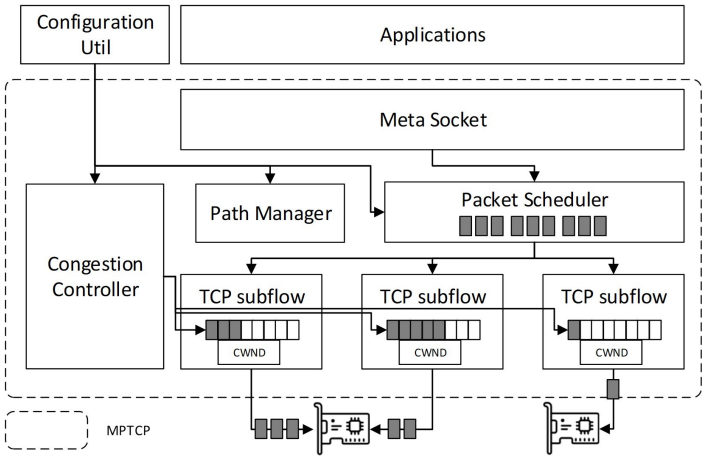


Figure 5.2 The architecture of MPTCP

There are two challenges in designing an efficient congestion control algorithm for MPTCP in Mobile Internet scenarios. First of all, in this scenario, the network environments of different paths may be quite different [132], and due to the introduction of mobility, the quality of these paths could undergo frequent and large dynamic changes [139]. Therefore, running a single type of congestion control algorithm on every subflow can not adapt to the heterogeneous and dynamic multipath network environments, resulting in degraded network transport performance [152]. Secondly, even if the network access types of different paths may be different, there may be shared bottleneck links between them, so MPTCP congestion control needs to implement multipath fairness. That is, when MPTCP connections share

a bottleneck link with traditional TCP connections, the bandwidth is fairly allocated to all types of connections [36].

However, none of the existing MPTCP congestion control algorithms can cope with the above challenges well. First of all, although any congestion control algorithm in traditional TCP, such as Cubic [107], can be independently applied to the subflow of MPTCP, this type of uncoupled multipath congestion control algorithms can not achieve multipath fairness. Secondly, the classic coupled congestion control algorithms, LIA [105], OLIA [67], and BALIA [102], which are based on NewReno's [54] "Additive Increase Multiplicative Decrease" principle, can implement multipath fairness well, but their performance is too conservative when paths do not share bottleneck links [36], and they also do not consider the problem of path heterogeneity. Then, coupled congestion control algorithms based on other types of TCP congestion control algorithms, such as mVeno[30] for wireless packet loss environments, delay-based wVegas [15], model-based Coupled-BBR [48], and eNMCC [119] based on throughput growth rate also have the above problems. In order to solve the problem that the performance of the above coupled algorithms is too conservative in the case of non-shared bottleneck links, some scholars have proposed some dynamically coupled congestion control algorithms [36, 49, 51, 120, 128, 143]. But, these algorithms are still not carefully designed for the scenario of heterogeneous paths, and can not adapt to the heterogeneous and dynamically changing multipath network environment in Mobile Internet scenarios. In addition, in recent years, with the continuous prosperity of ML, some researchers have also proposed some learning-based multipath congestion control algorithms to adapt to different network environments, such as several algorithms based on Reinforcement Learning [52, 76, 104, 135, 137, 145] and MPCC [43] based on online learning. However, these algorithms generally have a series of problems preventing them to be deployed at scale in reality. For example, their ways to implement multipath fairness are either too conservative to cause performance degradation or not guaranteed at all, their performance experiences degradation in unseen network environments, they introduce large system overhead, the convergence speed of them is slow, and the deployment of them is complicated [145].

In view of the shortcomings of existing work, we believe that it is very difficult to design an efficient and easy-to-deploy multipath congestion control algorithm that can adapt to heterogeneous paths and dynamically changing environments while achieving multipath fairness at the same time. In terms of traditional single-path TCP congestion control, existing research work [2, 152] has shown that existing congestion control algorithms can often only achieve good transport performance in specific network environments that meet their design assumptions. Therefore, some scholars[20, 94, 152] propose that appropriately choosing congestion control algorithms for TCP connections by sensing network environments is

able to improve TCP transport performance. Inspired by this research thread, we design the MPTCP-SSCC (MPTCP Smart Selection of Congestion Control) system based on MPTCP to implement a data-driven multipath congestion control algorithm selection mechanism, which can dynamically select an appropriate congestion control algorithm for each subflow according to the network condition of each subflow and also implement multipath fairness. This system enables MPTCP better adapt to the heterogeneous and dynamically changing network environment, thereby improving the transport performance of MPTCP.

MPTCP-SSCC continuously perceives the quality and bottleneck link sharing situation of subflows in kernel space, collects data from MPTCP kernel in user space, and then uses a data-driven approach to dynamically adjust the congestion control algorithm for each subflow in user space. Note, MPTCP-SSCC makes the decision for CCA selection in user space for two reasons. First, there are a large number of available tools and libraries (such as various ML libraries) in user space. On the other hand, this design makes the modification of the decision-making module independent of the kernel, which ensures the extensibility of the system.

In order to be compatible with legacy MPTCP applications, efficiently transfer data between kernel and user space, and adjust the congestion control algorithm of each subflow at runtime, we design and implement a lightweight congestion control algorithm selection framework based on eBPF [11, 60]. Due to the complex relationship between the network environment data and the suitable congestion control algorithm, it is difficult to map network environment data to a specific CCA with a limited number of human-designed rules. Therefore, the congestion control algorithm selection is abstracted as a classification task. We use the XGBoost [133] algorithm to train an ML model that is combined with some fixed rules to accomplish the CCA selection task in the MPTCP-SSCC system. The experimental evaluation of MPTCP-SSCC in emulated and real environments shows that the system can dynamically select an appropriate congestion control algorithm for each subflow and satisfy multipath fairness according to the network conditions of each path.

To sum up, the core contributions of this chapter are as follows:

- We design and implement a lightweight multipath congestion control algorithm selection framework based on eBPF, which allows to continuously perceive the network environment, transport quality, and coupling requirements of subflows in user space in an application-agnostic manner and to dynamically adjust the CCA for each subflow at runtime. As the decision module of CCA selection is in user space, the framework has good extensibility;
- Based on the above framework, MPTCP-SSCC is designed to implement a data-driven multipath congestion control algorithm selection mechanism, in which a combination of

ML models and artificial rules is used to capture the relationship between the network environment data and suitable congestion control algorithms;

- We conducted a large number of experiments based on both emulated and real environments. The results show that MPTCP-SSCC can achieve multipath fairness. At the same time, compared with several popular coupled CCAs, such as LIA [105], OLIA [67], BALIA [102], and wVegas [15], or a state-of-the-art learning-based multipath congestion control algorithm, MPCC [43], MPTCP-SSCC system can improve 19% – 25% average throughput and reduce average queuing delay by up to 21%.

The remainder of this chapter is organized as follows. First, the research motivation and the technical challenges are further described in § 5.2. After that, § 5.3 introduces the core ideas. Then, the design and implementation of the MPTCP-SSCC system and the experimental evaluation results are described in § 5.4 and § 5.5. Finally, § 5.6 concludes this chapter.

5.2 Motivation and challenges

5.2.1 The limitation of existing multipath CCAs

In MPTCP, multipath congestion control determines the sending rate of each subflow by estimating the available bandwidth on each path, which is a key component affecting the transport performance of MPTCP. Different from the congestion control in traditional single-path TCP, multipath congestion control needs to satisfy multipath fairness in addition to efficiently utilizing the available network resources on paths. That is, when multiple MPTCP subflows of a MPTCP connection share the same bottleneck link with a TCP connection, the MPTCP connection needs to fairly share the bandwidth with the TCP connection. Therefore, independently applying a congestion control algorithm of TCP on each subflow can not meet the goal of multipath fairness.

To implement multipath fairness, several coupled multipath CCAs, such as LIA [105], OLIA [67], BALIA [102], and Coupled-BBR [48], are proposed. The key idea of coupled CCAs is to couple the growth coefficient of congestion window or the bandwidth probing gain of subflows as stated in § 2.3.1. However, the fairness constraints implemented by the above CCAs are too strong, causing subflows to be coupled even when they do not share a common bottleneck link, resulting in a performance loss of up to 50% [36]. In order to alleviate this problem, some scholars proposed dynamically coupled multipath CCAs [36, 49, 51, 120, 128, 143]. The core idea of them is to couple subflows only when necessary, which improves the transport performance of MPTCP when subflows do not share

a common bottleneck link. For example, MPTCP-SBD [36] decides to run the coupled algorithm, OLIA [67], or the uncoupled algorithm, NewReno [54], on subflows depending on whether the subflows share a common bottleneck link.

On the premise of satisfying multipath fairness, multipath congestion control needs to be able to efficiently utilize the available network resources on each path for efficient data transmission. In Mobile Internet scenario, the complex network environment poses severe challenges to the realization of efficient multipath congestion control. First of all, whether the multiple network interfaces are homogeneous (e.g. two cellular interfaces) or heterogeneous (e.g. a cellular interface and a WiFi interface), the heterogeneity of path quality is ubiquitous [21, 25, 74, 110]. Secondly, the network environment of these paths could constantly change due to factors such as mobility, routing changes, and the load changes of base stations. However, the aforementioned coupled and dynamically coupled congestion control algorithms perform homogeneous congestion control actions on each path. Thus, they can not adapt to the heterogeneous and dynamic multipath network environments [43, 137, 76], which inevitably cause them to fail to provide high performance in Mobile Internet scenarios.

In order to improve the adaptability of multipath congestion control to the network environment, some scholars have proposed a series of congestion control algorithms based on Machine Learning. Although these learning-based algorithms have achieved a certain degree of performance improvement in some dynamically changing network environments, they still have some problems. Algorithms based purely on Reinforcement Learning [52, 76, 104, 135, 137] (RL) usually do not implement multipath fairness. In addition, their performance and convergence speed have problems in unseen network environments, and the system overhead of them is large [145]. Therefore, it is difficult to deploy them in production environments at scale. The hybrid algorithm, MPLibra [145], based on RL and coupled algorithms adopts similar multipath fairness constraints to the classical coupled algorithms, resulting in its performance being compromised in the scenario of non-shared bottleneck links. MPCC [43] based on online learning also has the problem related to strict multipath fairness constraints. In addition, it also has shortcomings in system overhead and convergence speed.

Therefore, so far, there is no practically deployable multipath congestion control algorithm that can adapt to the complex, heterogeneous and dynamically changing network environment in Mobile Internet scenarios and implement multipath fairness at the mean time.

5.2.2 Motivation and problem

Through the analysis of the existing MPTCP congestion control algorithms in § 5.2.1, we believe that either based on the idea of coupled CCAs or learning-based CCAs, it is very

difficult to design a general multipath congestion control algorithm to satisfy both multipath fairness and adaptability to the complex, heterogeneous and dynamically changing network environment in Mobile Internet scenarios. First, to achieve multipath fairness via coupling the sending rate of subflows, it is usually required to perform homogeneous congestion control on subflows. Therefore, on multiple heterogeneous and dynamically changing paths, the homogeneous congestion control inevitably lead to performance loss. On the other hand, learning-based CCAs lack flexibility when implementing multipath fairness, and also have problems in terms of system overhead, deployment complexity, and generalization ability in unseen network environments.

In recent years, many single-path congestion control algorithms have been proposed. They usually achieve good performance in a network environment that conforms to their own design assumptions [2, 152].Thus, some researchers propose that appropriate congestion control algorithms can be selected for TCP flows by sensing the network environment to improve transport performance [20, 94, 152]. Inspired by this research thread, we consider whether it is possible to implement a multipath congestion control algorithm selection mechanism that can sense the network environment where each subflow is located, and dynamically select the most appropriate congestion control algorithms for subflows to improve transport performance while also achieving multipath fairness. For example, when it is perceived that multiple subflows share a common bottleneck link, a coupled congestion control algorithm is selected for them to satisfy multipath fairness; when subflows are perceived to be in a non-shared bottleneck link situation, they independently select a congestion control algorithm (e.g. BBR [16], Cubic [107], C2TCP [1], and etc.) that best suits their current network environment to improve transport efficiency as much as possible.

5.2.3 Technical challenges

There are several technical challenges to implement the above mentioned congestion control algorithm selection mechanism for MPTCP. First, dynamically adjusting the congestion control algorithms of subflows by sensing their network environment requires a multipath congestion control algorithm selection framework that provides the ability to collect performance statistics for each subflow, to indicate if a subflow shares a common bottleneck link with other subflows within the same MPTCP connection, and to adjust congestion control algorithms for subflows at runtime. Then, considering that there are many kinds of congestion control algorithms, and new congestion control algorithms are constantly being proposed, the framework should have good extensibility to facilitate the integration of new congestion control algorithms. In addition, from the perspective of deployability, the implementation overhead of the framework should not be too high, and it should be transparent to applications

and compatible with legacy MPTCP applications. At present, MPTCP does not provide corresponding tools and interfaces to meet the above requirements. A subflow's information is stored in the corresponding *struct sock* in the kernel. Currently, the only way provided by MPTCP to obtain the information of a subflow is to call the *getsocketopt* API in the application using the MPTCP connection, which does not meet the goal of being transparent to applications. For setting CCAs for subflows, although Hesmans et al. [55] proposed to add socket options to use the *setsocketopt* API to set the congestion control algorithm individually for each subflow, this proposal is not transparent to applications and implemented in any existing MPTCP version. Second, after having the above framework, how to choose an appropriate congestion control algorithm for subflows based on the collected data is also a very challenging problem. The relationship between the network environment reflected by the information of a subflow and its suitable congestion control algorithm is very complicated, and there is currently no clear guidance on the rules to map the subflow information to a suitable congestion control algorithm.

5.3 The core ideas

First, we consider how to design and implement the multipath congestion control algorithm selection framework. A complete multipath congestion control algorithm selection framework should include two parts of functions, namely, collecting the information of subflows to provide input for the selection of CCAs, and providing interfaces for the adjustment of the CCA of a subflow at runtime. To make the framework transparent to applications and extensible, it is necessary to reduce the modification in MPTCP kernel as much as possible, and avoid any modification on the socket interfaces of MPTCP. According to the above requirements, we consider building a multipath congestion control algorithm selection framework based on eBPF, because eBPF allows to monitor various types of information in the kernel safely and efficiently, modify the data structure in the kernel, and flexibly configure the network protocol stack in the kernel [11, 60, 87] without modifying and recompiling the kernel.

For collecting subflow information, the eBPF-based BCC [60] (BPF Compiler Collection) framework can efficiently read the *struct sock* structure corresponding to each subflow in the kernel in user space without any modification to the application. For the adjustment of the congestion control algorithm of a subflow, TCP-eBPF [11] can be used, which is application transparent and only needs to add a small number of hooks in the kernel. Based on the above design, the decision-making part of the congestion control algorithm selection can be completely implemented in user space, enabling it to use a rich software tool library, and the

subsequent modification and expansion of this part can be completely independent of the kernel. It is worth noting that, based on the existing MPTCP implementation in the kernel, the structure of *struct sock* corresponding to each subflow does not include information about whether the subflow shares a common bottleneck link with other subflows. To solve this problem, MPTCP-SBD [36] can be used to implement shared bottleneck link detection of subflows in the kernel, and then save the state in *struct sock*. To sum up, in order to realize the framework, only a small amount of code with fixed functions needs to be added to the kernel, and the rest can be independent of the kernel. Therefore, the framework has good extensibility and deployability.

Secondly, as mentioned above, the relationship between the network environment reflected by the information of a subflow and the congestion control algorithm it is suitable for is very complicated. Therefore, it is considered to introduce ML techniques to capture the relationship between them, and let ML models to choose the appropriate congestion control algorithm for each subflow based on the subflow information provided by the framework. However, simply using ML models leads to the problem that multipath fairness can not be satisfied. Therefore, for situations where multipath fairness needs to be satisfied, that is, when subflows share a bottleneck link, artificial rules are adopted to select coupled congestion control algorithms for them.

Finally, the above two parts are integrated into a complete system MPTCP-SSCC to realize the data-driven multipath congestion control algorithm selection mechanism.

5.4 Design and implementation

This section will elaborate the design and implementation of the MPTCP-SSCC system. First, it explains how the overall architecture of the whole system is designed. Then, in the next few sections, the design of each module in the system is explained respectively. Finally, it is briefly explained how the system is implemented.

5.4.1 System overview

The overall architecture of the entire system is shown in Figure 5.3. The system includes four main modules: **(1)** the Dynamic Coupling Module based on the SBD (Shared Bottleneck Detection) algorithm [36] to detect if subflows share a common bottleneck link; **(2)** the Data Collection Module based on eBPF-BCC [60] to collect the information of subflows; **(3)** the CC (Congestion Control) Selection Module to dynamically predict and select the CCAs of subflows in a data-driven way; **(4)** the CC Switch Module based on TCP-eBPF [11]

to enforce selected CCAs of subflows in kernel. Among them, except for the CC Switch Module, the remaining modules together constitute the eBPF-based multipath congestion control algorithm selection framework.

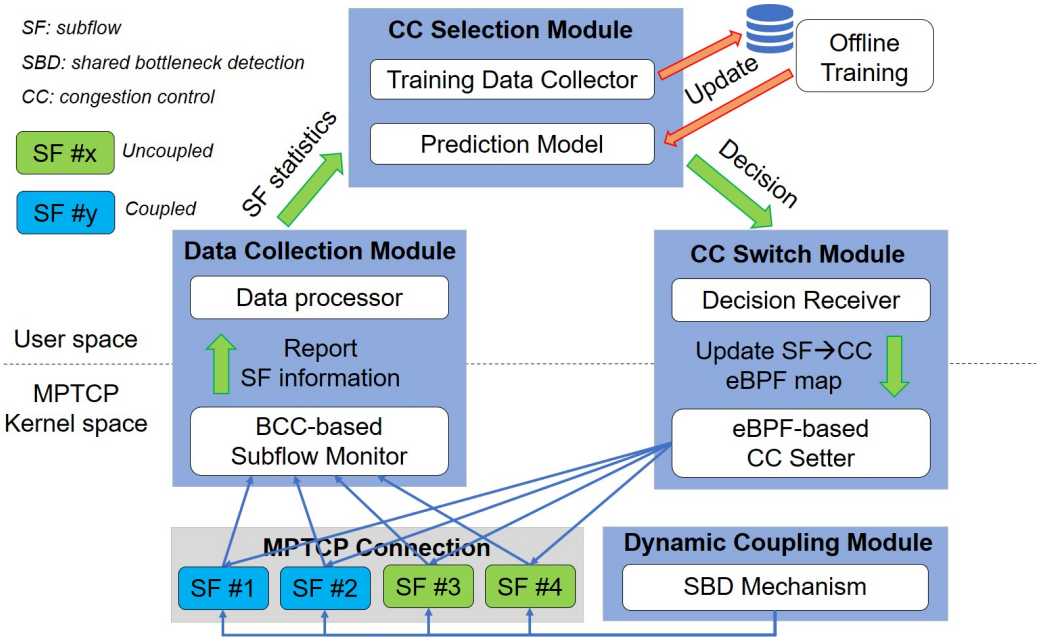


Figure 5.3 The architecture of MPTCP-SSCC

The workflow of the whole system is as follows:

- First, when an MPTCP subflow is established, it belongs to the uncoupled group. During the lifetime of the MPTCP connection, the Dynamic Coupling Module dynamically adjusts each subflow to the coupled or uncoupled group via the SBD algorithm;
- The Data Collection Module periodically reports the grouping state and performance statistics of each subflow to the user space;
- The CC Selection Module reads the data of each subflow, and then predicts the most suitable congestion control algorithm for each subflow in the following period of time through artificial rules and ML models; in addition, this module also stores the data of each subflow for training and retraining ML models;
- The decisions made by the CC Selection Module (i.e. the congestion control algorithm predicted for each subflow) are forwarded to the CC Switch Module, which passes these decisions to kernel space and calls the corresponding eBPF APIs to enforce the selected CCAs on each subflow.

5.4.2 Dynamic Coupling Module

The Dynamic Coupling Module mainly adopts the approach in MPTCP-SBD [36]. The core idea is that if multiple subflows pass through the same bottleneck link, the queuing situation that they experience is similar. As the queuing situation can be reflected by one-way delay (OWD), the temporal shapes of the OWDs of subflows that share a common bottleneck link should have a certain similarity. Then, it is possible to split subflows into different groups based on whether they have the similarity in the temporal shape of OWDs. The flowchart of the entire module is shown in Figure 5.4.

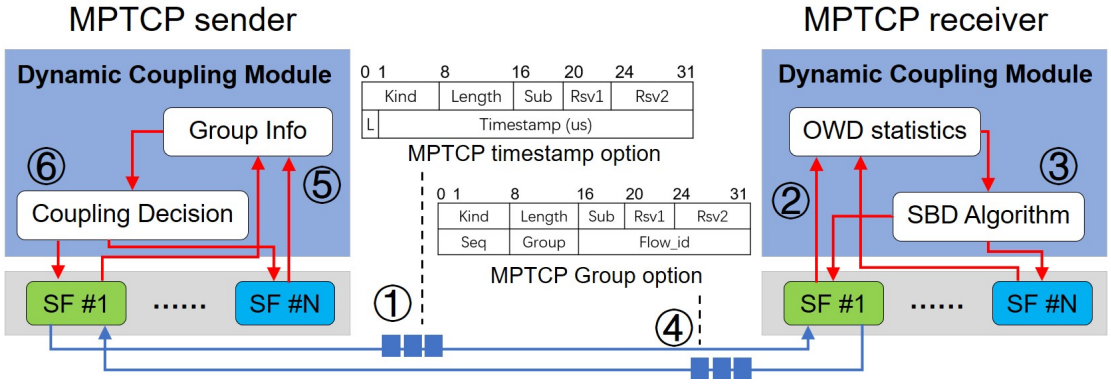


Figure 5.4 The diagram of Dynamic Coupling Module

As the step ① in Figure 5.4 shows, on the sender side, the Dynamic Coupling Module adds the MPTCP Timestamp option that records a packet's sending time to every packet. Therefore, the sending time of every packet can be received by the MPTCP receiver. In addition, the L field in the MPTCP timestamp option incrementally updates the packet loss information from the sender to the receiver, so that the receiver can calculate the sender's loss rate of each subflow.

On the MPTCP receiver side, the Dynamic Coupling Module calculates OWDs by parsing the MPTCP timestamp options carried by packets to get the sending time of packets and computing the difference between the local time and the sending time. Using the collected OWD samples, the statistical features of the OWD of a subflow that reflects the temporal shape of OWDs are calculated (the step ② in Figure 5.4). The way of calculating these statistical features refers to RFC-8382 [50], which is explained by Equation (5.1) – (5.7). In Equation (5.1), OWD_c represents an OWD sample obtained in the k-th measurement interval with duration T, C_k represents the number of samples obtained in this measurement interval, and $\overline{\mathrm{OWD}_k}$ is the average OWD of the k-th measurement interval. By recording the data collected for the last N measurement intervals, Equation (5.2) is used to estimate the long-term average OWD. Next, using the collected OWD-related data, three key statistical

features need to be estimated: *skewness*, *variability*, and *frequency*, which are defined by Equation (5.3) – (5.4), Equation (5.5) – (5.6), and Equation (5.7). Here, the frequency indicates the average times of that the short-term average of OWD ($\overline{\text{OWD}_k}$) deviates from the long-term average ($\overline{\textbf{OWD}}$) by more than a certain threshold (p_v $\times$ **var_est**). Figure 5.5 gives an example of how the frequency is calculated. It should be noted that the Dynamic Coupling Module calculates the relevant statistical features at the end of each measurement interval, and uses the results for the SBD grouping algorithm. That is, a grouping decision is made once at the end of every measurement interval with duration T.

$$\overline{\text{OWD}_k} = \frac{\sum_{c=1}^{C_k} \text{OWD}_c}{\text{C}_k} \tag{5.1}$$

$$\overline{\textbf{OWD}} = \frac{\sum_{k=1}^{N} \overline{\text{OWD}_k}}{N} \tag{5.2}$$

$$\text{skew\_base}_k = \sum_{c=1}^{C_k} [\text{OWD}_c < \overline{\textbf{OWD}}] - \sum_{c=1}^{C_k} [\text{OWD}_c > \overline{\textbf{OWD}}] \tag{5.3}$$

$$\textbf{skew\_est} = \frac{\sum_{k=1}^{N} \text{skew\_base}_k}{\sum_{k=1}^{N} C_k} \tag{5.4}$$

$$\text{var\_base}_k = \sum_{c=1}^{C_k} |\text{OWD}_c - \overline{\text{OWD}_{k-1}}| \tag{5.5}$$

$$\textbf{var\_est} = \frac{\sum_{k=1}^{N} \text{var\_base}_k}{\sum_{k=1}^{N} C_k} \tag{5.6}$$

$$\textbf{freq\_est} = \frac{\text{number\_of\_crossing}}{N} \tag{5.7}$$

After obtaining the OWD-related statistical features of subflows each time, the MPTCP receiver makes a grouping decision for subflows by comparing the similarity of these statistical features of the subflows (the step ③ in Figure 5.4). The specific grouping algorithm adopts the basic algorithm in RFC-8382 [50], as shown in Figure 5.6. The algorithm eventually divides the subflows into several different groups. Among them, the subflows in the non-congested group and the subflows that belong to a group with only one member are considered to not share a common bottleneck link with other subflows. Because the network conditions and bottleneck link congestion levels may fluctuate, the result of a single grouping decision may also change over time. Therefore, the Dynamic Coupling Module records the latest D grouping decision results ($D = 10$), and selects the most popular grouping decision

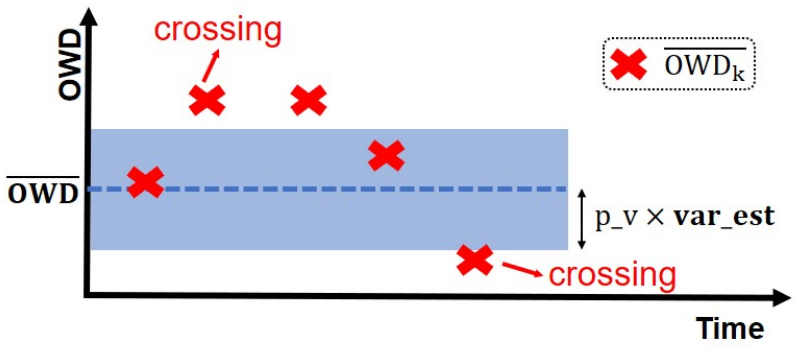


Figure 5.5 An example of frequency calculation: freq_est = 2 / 5

(the decision appeared in more than half of the recent D decisions) as the final subflow grouping result. It should be noted that since the OWD-related statistical features used by the grouping algorithm do not depend on the absolute value of OWD, it is not necessary to perform high-precision clock synchronization on the sender and receiver.

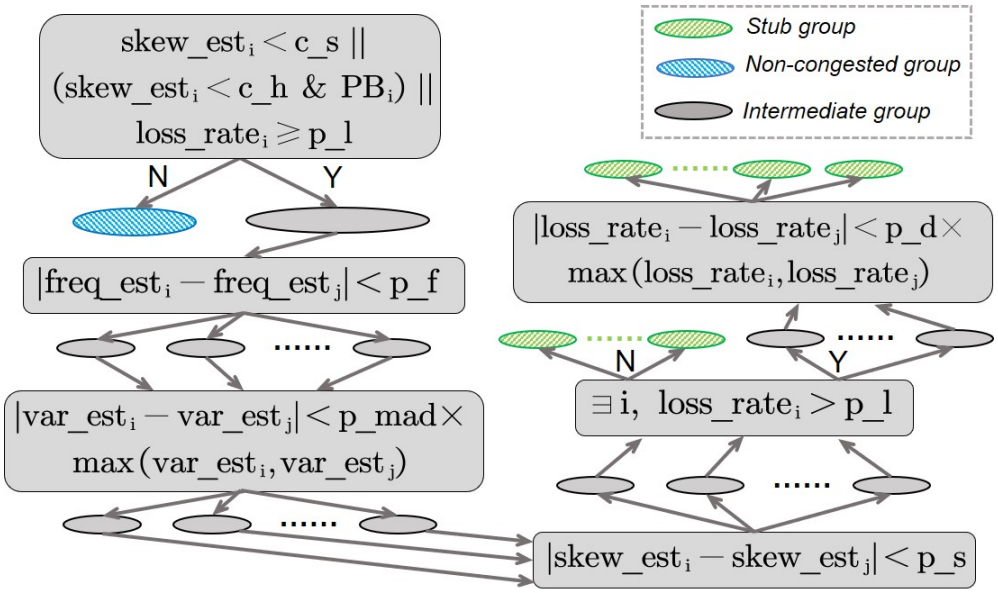


Figure 5.6 The SBD grouping algorithm: p_* are parameters of the algorithm, and PB indicates if a subflow does NOT belong to the non-congested group in a previous grouping decision

After the grouping state is updated, the MPTCP receiver synchronizes the grouping information to the MPTCP sender through the MPTCP Group option (the step ④ in Figure 5.4). The grouping information mainly includes group ID and flow ID, which indicates which

subflow belongs to which group. It should be noted that all the subflows belonging to the non-congested group have the same reserved default group ID.

After the MPTCP sender receives the grouping information, it assigns the subflows in the non-congested group and the subflows that belong to a group with only one member to the uncoupled group, and asssigns all other subflows into the coupled group (the step ⑤ and ⑥ in Figure 5.4).

For the parameters related to the SBD algorithm, we adopt the same values as MPTCP-SBD [36], as shown in Table 5.1.

Table 5.1 The parameters of SBD algorithm [36]

T(ms)	N	c_s	c_h	p_f	p_s	p_v	p_mad	p_d	p_l
350	50	-0.01	0.3	0.1	0.1	0.7	0.1	0.1	0.1

5.4.3 Data Collection Module

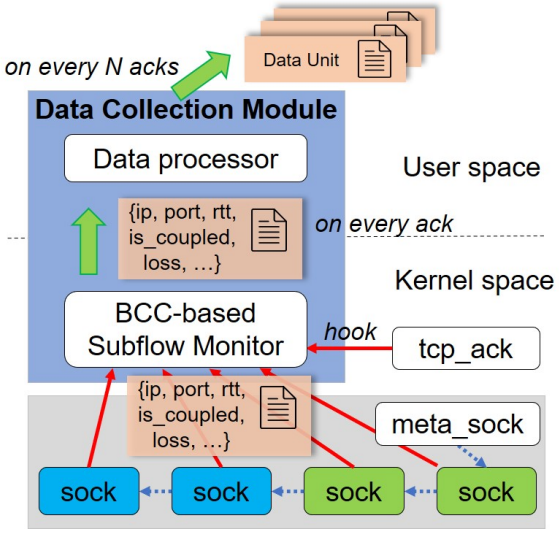


Figure 5.7 Data Collection Module

The Data Collection Module is shown in Figure 5.7. When each subflow of the sender receives a TCP ACK (tcp_ack), it will trigger the eBPF program based on BCC [60] in the kernel to collect relevant information of the corresponding subflow, such as the four-tuple composed of source/destination IP and source/destination port, RTT, the number of lost packets, sending rate, the group that the subflow belongs to (coupled or uncoupled), and etc. The information is then reported to the Data Processor sub-module in user space through eBPF for further processing. The Data Processor records the per-ACK subflow information,

and then aggregates the per-ACK subflow information for N consecutive ACKs to generate a performance summary over the period of N ACKs (Data Unit). The specific information contained in the Data Unit is shown in Table 5.2, including the four-tuple of the subflow, whether the subflow belongs to a coupled group, throughput, packet loss rate, delay, and the timestamp when the Data Unit is generated.

Table 5.2 The structure of Data Unit

Field	Description
SF 4-tuple	The (sip, sport, dip, dport) tuple of a subflow
is_coupled	The latest is_coupled collected in this period
ts	The timestamp of generating this summary
tput	The average delivery rate of this period
loss	The loss rate of this period
delay	The average packet delay of this period
$\texttt{delay}_{\texttt{min}}$	The minimum value of delay so far
$\texttt{tput}_{\texttt{max}}$	The maximum value of delivery rate so far

It should be noted that the reason for using N consecutive ACKs as the time interval for computing the subflow performance summary rather than a fixed time interval is that the RTTs of different subflows may vary greatly. Therefore, a fixed time interval may be too large or too small for different subflows. When the system is actually deployed, the value of N will affect the computational overhead and the timeliness of switching congestion control algorithms. We refer to the settings in Antelope [152] to set the default value of N to 20 for a good balance between computational overhead and the timeliness of switching congestion control algorithms.

5.4.4 CC Selection Module

The CC Selection Module is shown in Figure 5.8, which contains two sub-modules: **(1)** Training Data Collector; **(2)** Prediction Model.

The specific workflow of the Training Data Collector is as follows. First, for the n-th Data Unit of a subflow, DU_n, the Training Data Collector calculates the transport performance score (Reward) of the CCA used by the subflow within the duration of this Data Unit. It should be noted that the Reward calculated according to DU_n reflects the transport performance of the congestion control algorithm selected at DU_{n-1}, so it is R_{n-1} instead of R_n (i.e. the score of the CCA selected at the beginning of the duration of DU_n using the data from DU_{n-1}). Then, the module stores the data in the format of $\{DU_n, CC, R_n\}$ (meaning that according to the current state DU_n, selecting CC as the congestion control algorithm of the subflow for the next period of Data Unit gets the performance score of R_n). It should be noted that if

the subflow corresponding to DU_n belongs to the coupled group at that time, the Prediction Model will inevitably select a coupled congestion control algorithm for it (see below), so there is no need for collecting data related to DU_n.

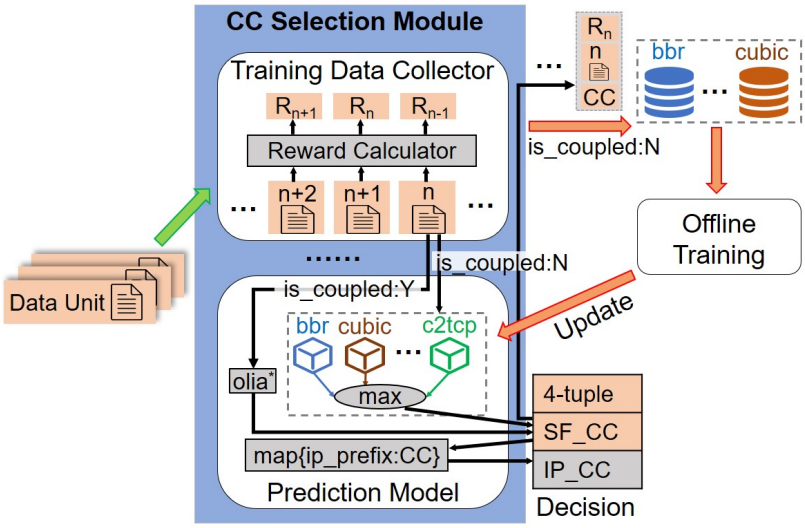


Figure 5.8 Congestion Control Selection Module

The specific way of Reward calculation refers to the literature [2, 152], as shown in Equation (5.8) and Equation (5.9). In Equation (5.8), R is an item based on Power [42]. Its intention is to maximize throughput, minimize packet loss rate (loss) and delay, where R_{max} is the maximum Power achieved by the subflow so far and is used to normalize R. It should be noted that the denominator in the R item does not directly use the delay but uses the delay$'$ defined by Equation (5.9). The main reason is that it allows the delay to exceed the minimum delay a bit for probing the maximum bandwidth. Therefore, in this case, the penalty on Reward caused by this part of the extra delay needs to be excluded [2]. In Equation (5.8) and Equation (5.9), the two parameters ζ and β adjust the sensitivity of Reward to packet loss and delay, respectively. They are set to 5 and 1.25, respectively, based on empirical values in the literature [2].

$$\text{Reward} = \frac{R}{R_{\max}} = \left(\frac{\text{tput} - \zeta \times \text{loss}}{\text{delay}'}\right) / \left(\frac{\text{tput}_{\max}}{\text{delay}_{\min}}\right) \tag{5.8}$$

$$\text{delay}' = \begin{cases} \text{delay}_{\min} & (\text{delay}_{\min} \leq \text{delay} \leq \beta \times \text{delay}_{min}) \\ \text{delay} & \text{otherwise} \end{cases} \tag{5.9}$$

In the Prediction Model, the congestion control algorithm selection of subflows is similar to a multi-classification problem. We use a hybrid approach which combines artificial rules

Algorithm 4: Subflow-level prediction algorithm

```
Input  : a data_unit of a subflow
Output : the predicted CC for the next decision period
Function predict_at_subflow_level (DU_n) :
    if DU_n.is_coupled == true then
        return "olia*"
    reward_max ← 0
    predicted_cc ← None
    for CC in list_of_available_CCs do
        R̂ ← M_CC(DU_n)
        if R̂ > reward_max then
            predicted_cc ← CC
            reward_max ← R̂
    return predicted_cc
```

and ML models to solve the above multi-classification problem. First of all, if the current input DU_n indicates that the subflow currently belongs to a coupled group, it is necessary to select a coupled CCA for it in order to achieve the multipath fairness. Therefore, OLIA [67] is selected as the output for this case[1]. If DU_n indicates that the subflow belongs to an uncoupled group, for the consideration of transport performance, it is necessary to select an uncoupled congestion control algorithm suitable for the current network environment for this subflow. Since there are many types of congestion control algorithms to choose from, and the relationship between network environments and the suitable CCAs is very complicated, we use ML to do the job. Specifically, for each congestion control algorithm candidate, *CC*, a model $\mathscr{M}_{CC}$ needs to be trained to output the predicted Reward, $\hat{R}$, given the input DU_n. Then, in the prediction stage, for a given input, we compare the outputs of all models, and select the *CC* corresponding to the model with the largest output Reward as the prediction result. A pseudocode description of the above process is presented in Algorithm 4. The details of the way to train the ML models will be introduced in 5.4.6.

For the Prediction Model, in addition to the selection of congestion control algorithms during the operation of subflows, it is also necessary to consider the selection of congestion control algorithms during the initialization of subflows, which is crucial for improving the performance of short flows. Since there is no data to generate Data Unit when the subflow is just established, the aforementioned method can not be used to predict which congestion control algorithm should be used. In order to solve this problem, we add a

[1] We made a slight change to the original OLIA algorithm, i.e., only all subflows belonging to the coupled group are considered when calculating the window growth factor. So in Figure 5.8, we use olia* instead of olia.

destination IP prefix[2] based mechanism to predict the initial congestion control algorithms for subflows. This destination IP prefix-based prediction mechanism calculates the weight of each congestion control algorithm for each IP prefix that has appreared based on the historical data generated by the aforementioned ML-based prediction mechanism. Then it outputs the congestion control algorithm with the highest weight for a given IP prefix. The weight update algorithm is shown in Algorithm 5, where α is an attenuation coefficient ($0 < \alpha < 1$) to gradually reduce the proportion of historical weights in the total weight calculation. As such, it is more inclined to assign a higher weight to the most recently selected congestion control algorithm to better adapt to the recent network environment. It should be noted that Zhou et al. [152] adopted a similar approach to predict the initial congestion control algorithm for single-path TCP connections.

Algorithm 5: CC prediction for a given IP prefix

Input : ip_prefix: destination IP masked by 0xFFFFFF00
predicted_cc: the output of *predict_at_subflow_level(DU_n)*
Output : the predicted CC for the given ip_prefix

```
Function predict_at_ip_prefix_level (ip_prefix, predicted_cc) :
    // Initialization for unseen IP prefixes
    if ip_prefix not in cc_weight_map then
        cc_weight_map[ip_prefix] = {CC1:0, CC2:0, CC3:0, ...}
    weight_map ← cc_weight_map[ip_prefix]
    // Reduce the weight of old data
    for CC in list_of_available_CCs do
        weight_map[CC] ← weight_map[CC] × α
    // Increase the weight of the predicted_cc
    weight_map[predicted_cc] ← weight_map[predicted_cc] + 1
    // Return the CC with the maximum weight
    return argmax_CC(weight_map)
```

To briefly summarize the function of the Prediction Model, for each Data Unit, it predicts the congestion control algorithm that should be selected in the next Data Unit period of the subflow (the SF_CC in Figure 5.8), and outputs the recently most used congestion control algorithm (the IP_CC in Figure 5.8) for the destination IP prefix corresponding to the Data Unit. Then, these results will be passed to the CC Switch Module (see § 5.4.5).

[2]If the sender has multiple network interfaces and source IP addresses, we can encode the low bits of source IP addresses in the low bits of IP prefixes to handle the situation where different source IP addresses correspond to one destination IP address. As we deploy the system on servers that usually have only one source IP address, only the IP prefix is used.

5.4.5 CC Switch Module

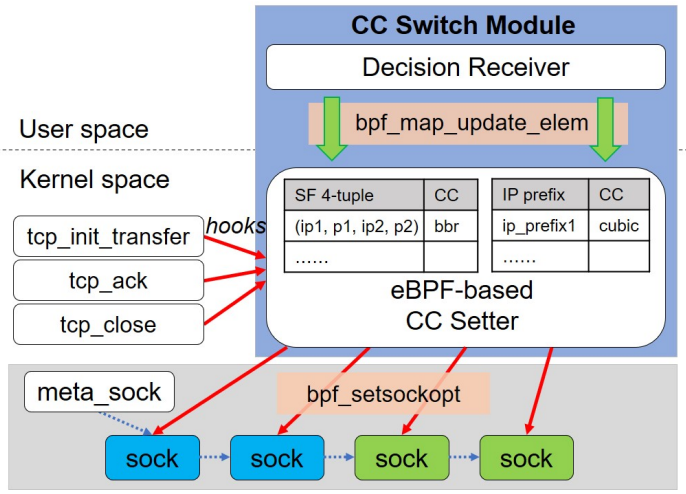


Figure 5.9 Congestion Control Switch Module

The CC Switch Module is shown in Figure 5.9, which mainly includes the Decision Receiver running in user space and the eBPF-based congestion control algorithm setting sub-module running in kernel space (eBPF-based CC Setter). After the Decision Receiver sub-module receives the output of the CC Selection Module (§ 5.4.4), it will pass it to the eBPF-based CC Setter through *ebpf_hash_map*. For each subflow, the eBPF-based CC Setter is called when the subflow is established (tcp_init_transfer). At this time, the eBPF-based CC Setter looks up the IP prefix-based congestion control algorithm table to find the initial congestion control algorithm for the subflow. If the search fails, the default initial congestion control algorithm (e.g. Cubic [107]) is set for the subflow. After the subflow is established, the eBPF-based CC Setter is called every time when it receives a TCP ACK (tcp_ack) to set the CCAs of subflows based on the four-tuple-based CCA table. When a subflow is closed (tcp_close), the eBPF-based CC Setter is also called. At this time, it deletes the entry corresponding to the subflow in the four-tuple-based CCA table.

5.4.6 System implementation

We implement the system on the Linux platform based on MPTCP v0.95 (the kernel version is Linux 4.19). The Dynamic Coupling Module is directly implemented in the kernel. The Data Collection Module is a Python program based on the BCC framework [60]. The CC Switch Module is implemented based on TCP-eBPF [11] and two hooks of *tcp_bpf_call* are added to the two kernel functions, *tcp_ack* and *tcp_close*. The CC Selection Module is implemented in userspace as a Python program.

In the above implementation, except for the Dynamic Coupling Module and the added *tcp_bpf_call* hooks, the modifications and extensions of other parts do not need to recompile the kernel. Since the Dynamic Coupling Module and the added *tcp_bpf_call* hooks are relatively stable, almost no modifications are needed. Therefore, the system has good extensibility. In addition, it should be emphasized that we do not make any changes to the socket interfaces, and the whole system is transparent to applications. Therefore, any legacy applications using MPTCP can use this system without any modification.

To implement the ML models in the CC Selection Module (see § 5.4.4), we use the XGBoost [133] algorithm. The main reason for choosing XGBoost is that the decision tree has advantages in interpretability and lower training complexity compared with some algorithms based on Deep Neural Network, which is more practical for the actual deployment of the system. It should be noted that the congestion control algorithm selection framework also supports replacing XGBoost with other types of ML or Deep Learning (DL) algorithms. In terms of congestion control algorithm selection, six representative congestion control algorithms are selected as candidates, as shown in Table 5.3. It should be noted that this system is not limited to these congestion control algorithms, and any other congestion control algorithms implemented in Linux can be easily integrated into this system.

Table 5.3 The list of congestion control algorithm candidates

CC	Category	Performance Objective
BBR [16]	model-based	high speed, low delay, and resilient to losses
Cubic [107]	loss-based	high speed in long-fat networks (with low loss rates)
Vegas [12]	delay-based	low delay
C2TCP [1]	loss-based	high speed and low delay in cellular networks
Westwood [19]	loss-based	high speed in wireless networks
Illinois [80]	delay/loss-based	high speed and low delay

In terms of training the XGBoost models, an iterative approach is used. When no training data has been collected in the first place, the prediction mechanism in the CC Selection Module is disabled, and the congestion control algorithm to be used by the subflow is randomly selected from Table 5.3 upon the establishment of the MPTCP connection. By this way, the initial training data for a particular congestion control algorithm can be obtained without the Prediction Model. Then, the initial XGBoost models for every CCA is trained with the initial training data. After the initial models are obtained, they are integrated into the CC Selection Module, and the Prediction Model is enabled, allowing the initial models to complete the selection of the congestion control algorithm. At this stage, we continue to collect training data for later retraining of the models. It should be noted that the above-

mentioned iterative training process can be repeated, and the models can be continuously optimized and improved.

During the training of the initial model, a testbed based on the Mahimahi multipath extension [26, 106] (described in § 5.5.1) is used to emulate various network environments. In these environments, all the congestion control algorithms in Table 5.3 are used for the collection of initial training data. Specifically, at this stage, according to the parameters listed in Table 5.4, 200 different network environments are randomly generated. In each environment, for each CCA, a 30 *s* MPTCP connection is repeatedly launched 5 times for data transfer.

Table 5.4 The parameters of network environments used in the initial model training phase

Bandwidth (Mbps)	RTT (ms)	Buffer (MB)	Loss Rate (%)	Background Traffic (MB)
5 – 100	20 – 400	0.1 – 50	0 – 2	0.1 – 50

In the iterative model training phase, 50 network traces in the real environment were collected by using *saturatr* [131] and *ping*. We used these traces with the testbed based on the Mahimahi multipath extension to emulate various real network environments. These network traces include the bandwidth, packet loss rate, and delay of each path when using "cellular + WiFi" or "cellular + cellular" to transmit data. For each network trace, 5 different background flow sizes were randomly generated from 0.1 - 50 MB. Therefore, we finally got 250 configurations. Then, for each configuration, we repeated the MPTCP data transmission of 30 *s* 5 times to obtain new training data. After the training data was acquired, the training data collected in the initial stage and the training data collected in this stage were combined, and the XGBoost model of each congestion control algorithm was retrained. Finally, the model obtained at this stage was used in the system for subsequent experimental evaluation.

5.5 Evaluation

This section describes the evaluation of the system. First, the environment used for the evaluation is introduced. Next, the multipath fairness, congestion control algorithm switching, transport performance, and system overhead are evaluated in detail.

5.5.1 Testbed

In this section, three types of testeds are used to complete the evaluation of different aspects of the system.

The first is a testbed based on Mininet [88]. This testbed can emulate different network topologies, and set the link bandwidth, delay, packet loss rate and routers' buffer size through *tc-netem* [53], so it is mainly used to evaluate multipath fairness. We deployed the first testbed on an Alibaba Cloud server with an 8-core Intel Xeon Platinum CPU, 32 GB memory, a Ubuntu 18.04 that uses a modified Linux MPTCP v0.95 kernel with the Dynamic Coupling Module and *tcp_bpf_call* hooks in *tcp_ack* and *tcp_close* implemented. Hereafter, we refer this MPTCP kernel as MPTCP-SSCC kernel.

The second is a testbed based on the Mahimahi multipath extension [26, 106], as shown in Figure 5.10. This testbed mainly provides an environment for a client to communicate with a server through two emulated independent network paths. In addition to setting fixed network bandwidth, delay, packet loss rate and routers' buffer size, it also supports to replay dynamically changing network bandwidth traces collected from real networks, which can better emulate real network environments. In addition, background traffic can be injected on two paths by using TCP to transmit files of a certain size on a certain path. These background TCP flows use Cubic [107] as the congestion control algorithm. We deployed this testbed on the same cloud server used by the first testbed. We mainly use it to collect training data (see § 5.4.6) and evaluate the performance gain of MPTCP-SSCC in non-shared bottleneck link scenarios.

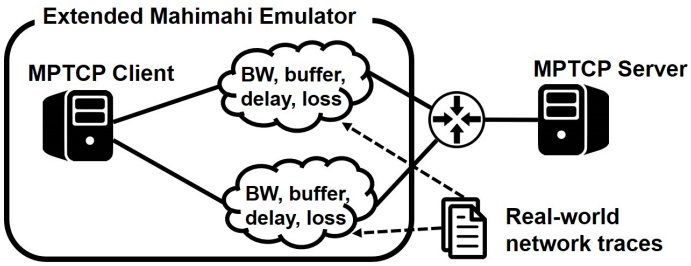


Figure 5.10 The testbed based on the multipath extension of Mahimahi

The third is a testbed based on real devices, as shown in Figure 5.11. In this testbed, the client is a laptop whose operating system is Ubuntu 18.04 using the MPTCP-SSCC kernel, and the server is an Alibaba Cloud server where the first two testbeds are deployed. The client can connect to the MPTCP server in two ways: "cellular + cellular" or "cellular + WiFi". Note, the cellular interfaces of the laptop are provided by tethering smartphones to the laptop via USB and both smartphones support 5G access. We mainly use this testbed to evaluate the performance of the MPTCP-SSCC system in real network environments. Different from the testbed based on the Mahimahi multipath extension, in the real environments, subflows on different paths may share a common bottleneck link.

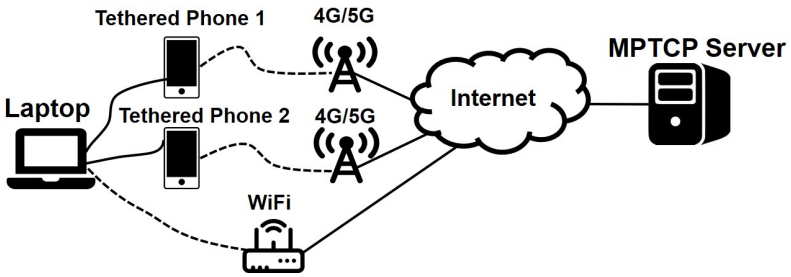


Figure 5.11 The testbed based on real devices

5.5.2 Multipath fairness

First, we evaluate if MPTCP-SSCC is able to implement multipath fairness. Here, two scenarios are considered, namely, the static bottleneck link scenario and the dynamic bottleneck link scenario.

In the static bottleneck link scenario, the network topology is shown in Figure 5.12. In this scenario, an MPTCP client accesses the network through M network interfaces and connects to an MPTCP server with a single network interface, so an MPTCP connection will create M subflows for data transmission. In addition, there are a pair of TCP client and server, the TCP connection between them will share the bottleneck link between R1 and R2 with the MPTCP connection. The MPTCP-SSCC system is deployed on the MPTCP server, and the bandwidth of the bottleneck link is set to 40 Mbps, the RTT is set to 40 ms, the packet loss rate is set to 0%, and the router's buffer size is set to $\eta\times$ BDP.

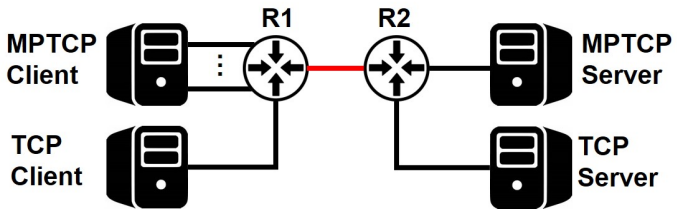


Figure 5.12 The network topology of the static bottleneck link scenario

First, the multipath fairness of the MPTCP-SSCC system is evaluated under the conditions of $M = 2$ and $\eta = 2$. Here, an MPTCP connection is initiated on the MPTCP client to download data from the MPTCP server. The duration of the connection is 180 s. At 20 s, a TCP connection is initiated on the TCP client to download data from the TCP server. In the experiment, the congestion control algorithm of each subflow of MPTCP connection is automatically selected by the MPTCP-SSCC system, and the congestion control algorithm of TCP connection is NewReno [54]. The experimental results are shown in Figure 5.13. It can be observed that after the TCP connection is started, the MPTCP connection is able to give

up bandwidth and fairly share the available bandwidth of 40 Mbps with the TCP connection. This shows that MPTCP-SSCC can successfully detect the situation where subflows share a common bottleneck link, and select the coupled congestion control algorithm, OLIA [67], to achieve multipath fairness.

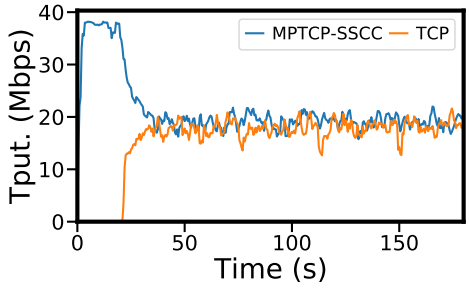


Figure 5.13 The bandwidth sharing result in the static bottleneck link scenario

Further, by changing the values of M and η, the above experiments are repeated to evaluate the influence of the number of subflows and the router's buffer size on multipath fairness. Here, Jain's Fairness Index [37] ($\mathscr{F}$) is used as the measure of multipath fairness. $\mathscr{F}$ is calculated as $\mathscr{F} = (\sum_{i=1}^{n} T_i)^2/(n \times \sum_{i=1}^{n} T_i^2)$, where T_i represents the average throughput of MPTCP connections or TCP connections, and the closer the value of $\mathscr{F}$ is to 1, the better the fairness. It is worth noting that $\mathscr{F}$ is calculated using the average throughput after the bandwidth contention has converged (i.e. 120 – 180 s). The experimental results are shown in Figure 5.14 and Table 5.5. In Figure 5.14, the number of subflows is fixed to 2, and in Table 5.5, the router's buffer size is fixed at 2× BDP. It can be observed that MPTCP-SSCC can correctly detect if subflows share the same bottleneck link under different buffer sizes and different numbers of subflows, and select the OLIA algorithm for subflows to satisfy multipath fairness.

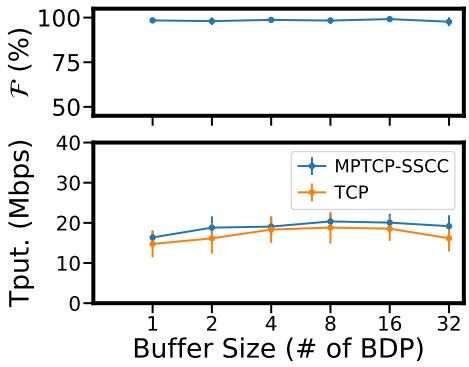


Figure 5.14 The impact of buffer size on multipath fairness in the static bottleneck link scenario

Table 5.5 The impact of subflow number on multipath fairness in the static bottleneck link scenario

# of SF	2	3	4	5	6
$\mathscr{F}(\%)$	96.33	93.55	91.21	92.48	88.78

In the dynamic bottleneck link scenario, the network topology is shown in Figure 5.15. Here, the MPTCP client accesses the network through M paths and communicates with the MPTCP server. On each path, there is a corresponding TCP client and server pair to compete for bandwidth with the subflow on the path. Here, the bandwidth of the LS link is set to 50 Mbps, the RTT is set to 40 *ms*, the packet loss rate is set to 0%, and the router's buffer size is set to $\eta \times$BDP (here, 40 Mbps is used for BDP calculation). For link L1 – LM, the bandwidth is set to $40/M$ Mbps. With the above setup, it is possible to inject UDP traffic on the LS link to shift the bottleneck link of the network from L1 – LM to LS.

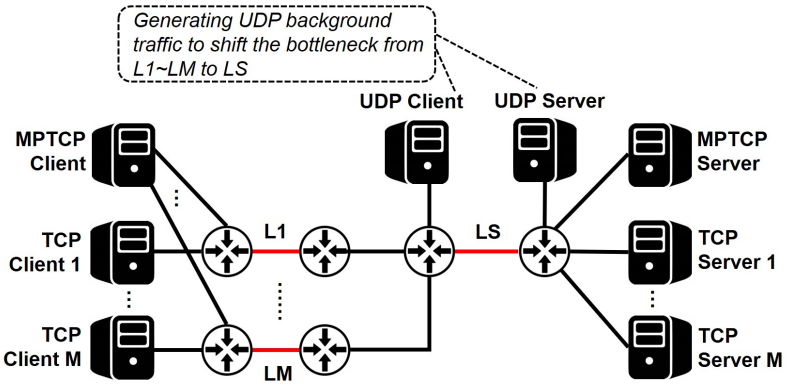


Figure 5.15 The network topology of the dynamic bottleneck link scenario

First, in the dynamic bottleneck link scenario, the multipath fairness of the MPTCP-SSCC system is also evaluated under the conditions of $M = 2$ and $\eta = 2$. In this set of experiments, an MPTCP connection is first initiated on the MPTCP client to download data from the MPTCP server. The duration of the connection is 180 *s*, and then, at 20 *s*, in the download direction of the LS link, 20 Mbps UDP traffic is injected to shift the bottleneck link to LS. Afterwards, at 40 *s*, a TCP connection is created on each TCP client to download data from its corresponding TCP server. In the experiments, the congestion control algorithm of each subflow is automatically selected by the MPTCP-SSCC system, and the congestion control algorithm of TCP connection is NewReno [54]. The experimental results are shown in Figure 5.16. It can be observed that when the bottleneck link changes, the MPTCP-SSCC system can still detect the situation that the subflows share the same bottleneck link, and then select the OLIA congestion control algorithm to achieve multipath fairness.

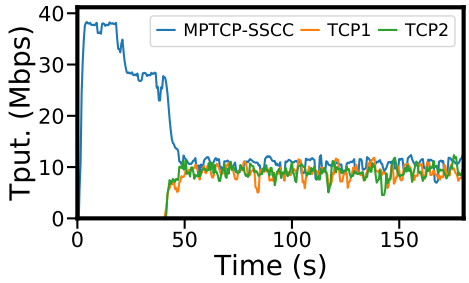


Figure 5.16 The bandwidth sharing result in the dynamic bottleneck link scenario

As in the static bottleneck link scenario, the influence of the number of subflows and the router's buffer size on the multipath fairness in the dynamic bottleneck link scenario is evaluated by changing the values of M and η. The experimental results are shown in Figure 5.17 and Table 5.6. In Figure 5.17, the number of subflows is fixed to 2, and in Table 5.6, the router's buffer size is fixed at $2\times$ BDP. In the dynamic bottleneck link scenario, it can also be observed that MPTCP-SSCC can correctly detect whether subflows share the same bottleneck link under the conditions of different buffer sizes and different numbers of subflows, and select the OLIA algorithm for subflows to satisfy multipath fairness.

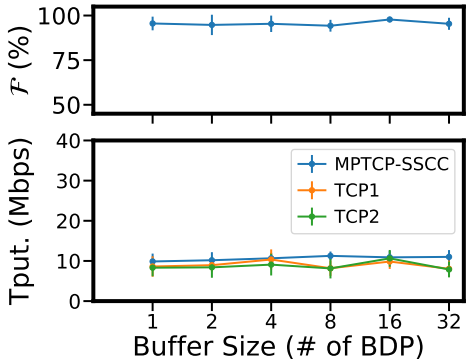


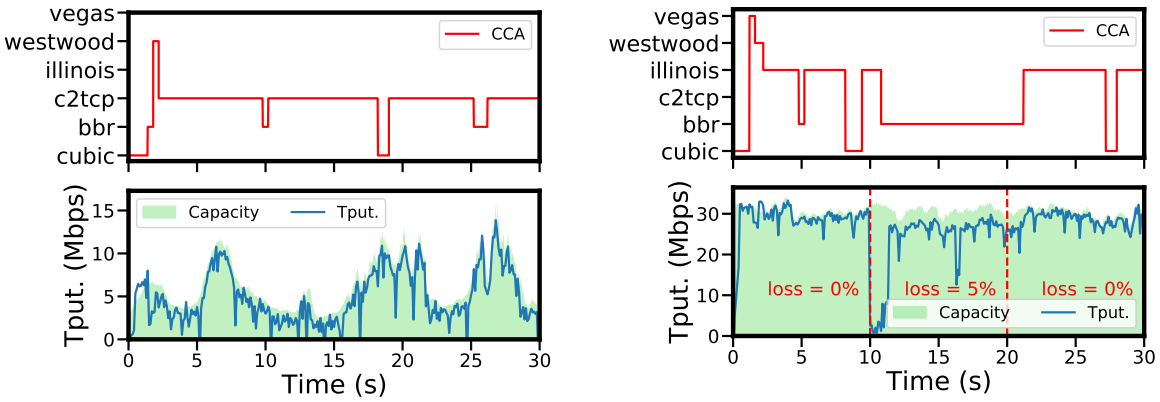
Figure 5.17 The impact of buffer size on multipath fairness in the dynamic bottleneck link scenario

Table 5.6 The impact of subflow number on multipath fairness in the dynamic bottleneck link scenario

# of SF	2	3	4	5	6
$\mathcal{F}(\%)$	92.29	90.31	90.97	87.15	85.44

5.5.3 The demonstration of CCA switching

In this section, on the testbed based on the Mahimahi multipath extension, it is demonstrated that the MPTCP-SSCC system can indeed dynamically select the appropriate congestion control algorithm for the subflow to improve the performance. The specific experimental configuration is as follows. We use *saturatr* [131] to collect the cellular and WiFi bandwidth trace. Then, we input the traces into the testbed to emulate a network environment with both cellular and WiFi network access. The bandwidth trace of the cellular path and the WiFi path is shown in Figure 5.18. For the cellular path, we set its RTT to 40 *ms* and the packet loss rate to 0%. For the WiFi path, we set its RTT to 100 *ms*, and set its packet loss rate to 5% during 10 – 20 *s* and 0% during the rest of time. Then, the MPTCP-SSCC system is deployed on the server, and an MPTCP connection is initiated from the client to download data from the server. During this process, the instantaneous throughput of each subflow and the congestion control algorithm used are recorded.



(a) Cellular path

(b) WiFi path

Figure 5.18 The demonstration of CCA switching

The experimental results are shown in Figure 5.18. First, because the server where the MPTCP-SSCC system is located has not accumulated any historical data before, its IP prefix-based initial congestion control algorithm selection mechanism does not take effect. Therefore, the initial congestion control algorithm for each subflow is the default Cubic algorithm. Then, it can be observed that the MPTCP-SSCC system dynamically selects the appropriate congestion control algorithm for the subflow. For the cellular path, the MPTCP-SSCC system chooses the C2TCP algorithm most of the time because it can maintain high throughput and low queuing delay in the case of frequent bandwidth changes. For the WiFi path, it can be clearly observed that between 10 – 20 *s*, the MPTCP-SSCC system selects

the BBR algorithm that is more resilient to packet losses to avoid transport performance degradation.

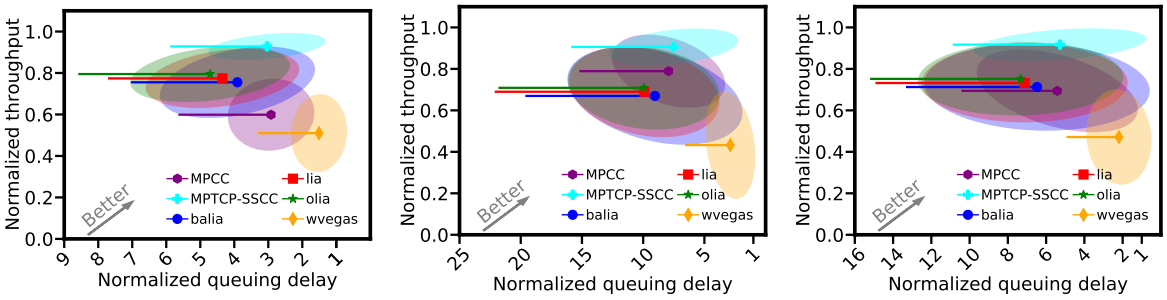
5.5.4 Performance evaluation in emulated networks

Next, we quantitatively evaluate the improvement in performance brought by MPTCP-SSCC on the testbed based on the Mahimahi multipath extension. The CCAs for comparison are coupled MPTCP congestion control algorithms, including LIA [105], OLIA [67], BALIA [102], and wVegas [15], and a state-of-the-art learning-based multipath congestion control algorithm, MPCC [43].

We use *saturatr* [131] and *ping* to collect 50 dual-path network traces ("cellular + WiFi" or "cellular + cellular") from real network environments. These traces include bandwidth time-series, packet loss rate, and delay of each path. It should be noted that the 50 traces are different from the 50 traces used to train the XGBoost models in the § 5.4.6.

In the above 50 network environments, we compare MPTCP-SSCC with LIA, OLIA, BALIA, wVegas, and MPCC in short flow transmission (100 KB – 2 MB), long flow transmission (5 MB – 50 MB), and mixed flow transmission (100 KB – 50 MB) scenarios. Each time we randomly select the amount of data to be transmitted from the short flow or long flow scenarios, and then randomly select the value in the range of 0.1 – 50 MB as the size of the TCP background flow on the two paths. According to the above method, a total of 20 sets of test configurations (flow size and background flow size) in short-flow scenarios and 20 sets of test configurations in long-flow scenarios are generated. Finally, in each network environment, for each test configuration, the transport performance of each congestion control scheme is tested in random order and the process is repeated 5 times. It should be noted that the MPTCP-SSCC system is deployed on the server, and the transmission direction of the flow is the download direction. During the test, the total average throughput per transfer is recorded, as well as the queuing delay of each packet on each path via the log files of the Mahimahi multipath extension.

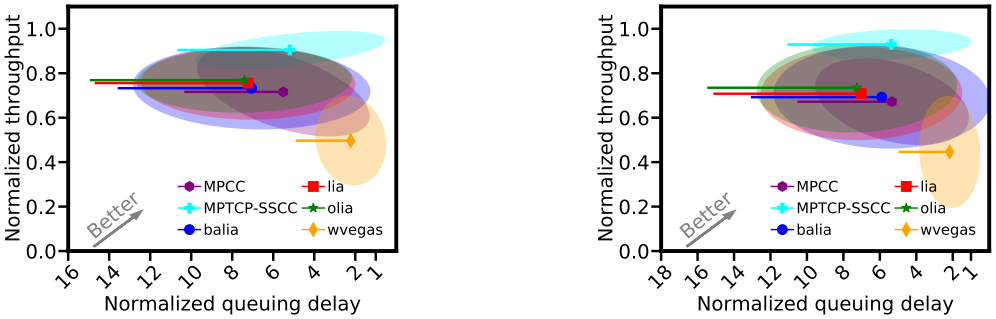
The experimental results are plotted in Figure 5.19, where the data of the mixed flow scenario aggregate the data of the short flow and long flow scenarios. In Figure 5.19, the X-axis shows the normalized packet queuing delay, the Y-axis shows the normalized average throughput, the dots in the figure represent the normalized average throughput and queuing delay, the tail of the line in the figure shows the normalized 95% percentile queuing delay, and the ellipse shadow in the figure shows the standard deviation. The normalization approach of each indicator is as follows. For throughput, under each test configuration in each network environment, the average throughput of 5 tests for each congestion control scheme is normalized to the maximum among them. For the average queuing delay, under



(a) Short flows (b) Long flows (c) Mixed flows

Figure 5.19 The performance comparison results in emulated network environments

each test configuration in each network environment, the average queuing delay (including the queuing delay of all packets on both paths) of 5 tests for each congestion control scheme is normalized to the minimum among them. For the 95th percentile queuing delay, the normalization baseline is the same as that of the average queuing delay. Finally, the normalized metrics of each congestion control scheme under all test configurations in all network environments are averaged to obtain the results in Figure 5.19.



(a) Stationary scenarios (b) Mobile scenarios

Figure 5.20 The performance comparison results in different types of emulated network environments

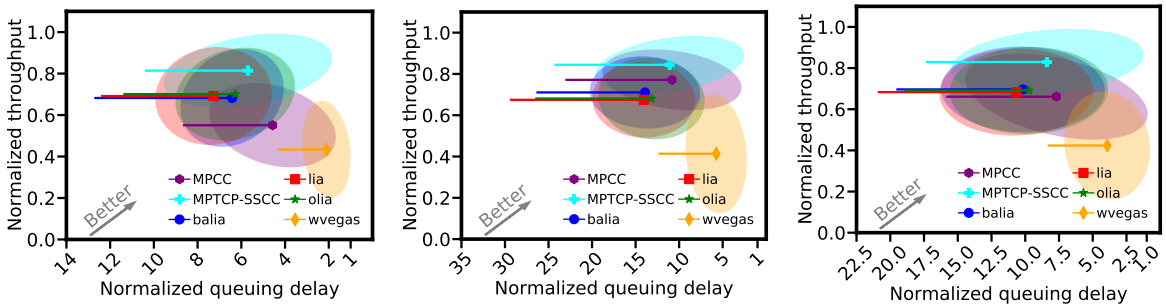
There are several observations from Figure 5.19. First, due to the existence of Cubic background flows, wVegas has disadvantages in competing bandwidth, so the throughput of wVegas is generally low. Second, the performance of LIA, OLIA, and BALIA is close. Third, the throughput MPCC is even lower than LIA, OLIA, and BALIA in short flow scenarios, because the online learning method needs some time to startup [43]. Fourth, no matter in short flow, long flow, or mixed flow scenarios, MPTCP-SSCC achieves good throughput and delay. The main reason for the performance improvement is to decouple the subflows when possible and dynamically select suitable congestion control algorithms for subflows. In the mixed flow scenario, compared with LIA, OLIA, BALIA, and MPCC, the average

throughput of MPTCP-SSCC is increased by 21% – 32%, and the average queuing delay is reduced by 3% – 28%.

Considering the network environment in mobile scenarios is usually more dynamic than that in stationary scenarios, we further compare the results between staionary and mobile scenarios. Figure 5.20a and Figure 5.20b show the mixed flow performance of MPTCP-SSCC in staionary scenarios and in mobile scenarios. In staionary scenarios, compared with LIA, OLIA, BALIA, and MPCC, MPTCP-SSCC improves the throughput by 18% – 26% in the case of mixed flows. In mobile scenarios, compared with LIA, OLIA, BALIA, and MPCC, MPTCP-SSCC improves the throughput by 26% – 38% in the case of mixed flows. The above results show that the improvement of MPTCP-SSCC is more significant when the network environment is more dynamic.

5.5.5 Performance evaluation in real networks

The emulated network environment in the testbed based on Mahimahi multipath extension is the case where subflows do not share the same bottleneck link. In order to better evaluate the performance of MPTCP-SSCC in the real environment where there may be a shared bottleneck link between subflows, we conducted performance evaluation experiments on the testbed based on real devices.



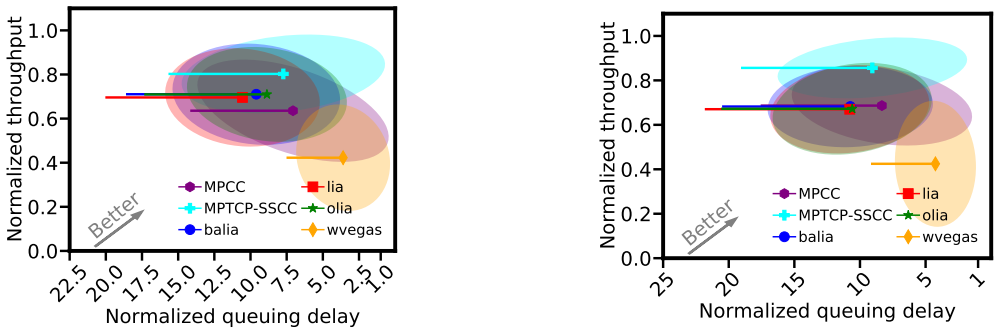
(a) Short flows (b) Long flows (c) Mixed flows

Figure 5.21 The performance comparison results in real network environments

A total of 10 different scenarios were considered in real-world environments including being stationary or moving in the scenarios of residences, restaurants, outdoors, vehicles, and subways. The types of dual-path network access include “cellular + cellular” or “cellular + WiFi”. Since there are background flows in real networks, we did not inject background flows intentionally. The rest of the experimental setup is similar to that in subsection § 5.5.4. It should be noted that in real environments, the queuing delay of each data packet can not be accurately obtained. Therefore, we used *tcpdump* [118] to capture all of the packets during

data transfers, and then use the RTT of each packet minus the minimum RTT of the subflow to which the packet belongs to estimate the queuing delay of each packet.

The experimental results are shown in Figure 5.21. In the figure, a similar conclusion to that in subsection § 5.5.4 can be observed. In the mixed flow scenario, compared with LIA, OLIA, and BALIA, the average throughput of MPTCP-SSCC is increased by 19% – 21%, and the average queuing delay is reduced by 13% – 21%. Compared with MPCC, MPTCP-SSCC improves the average throughput by 25% in the mixed flow scenario, and the average queuing delay is similar. In addition, Figure 5.22a and Figure 5.22b show the performance comparison of mixed flows between MPTCP-SSCC and other schemes in stationary and mobile scenarios, respectively. Compared with LIA, OLIA, BALIA and MPCC, the throughput improvement of MPTCP-SSCC in stationary and mobile scenarios is 13% – 26% and 25% – 28%, respectively. Similar to the experimental results in the emulated network environment in the previous subsection, this part of the experimental results further verifies that MPTCP-SSCC can achieve more performance improvements when the network environments are more dynamic.



(a) Stationary scenarios

(b) Mobile scenarios

Figure 5.22 The performance comparison results in different types of real network environments

5.5.6 System overhead

Finally, the overhead of the MPTCP-SSCC system is evaluated. The specific evaluation method is as follows. First, we recorded the server's system CPU and memory utilization while conducting the performance experiments in § 5.5.4. Then, the average CPU and memory utilization of MPTCP-SSCC and several traditional coupled congestion control algorithms are calculated. Finally, normalized average CPU and memory utilization are displayed in Figure 5.23. Here, the normalization is done by normalizing the average CPU and memory utilization of all schemes to the minimum value among them. In Figure 5.23, it

can be observed that compared to using the traditional coupled congestion control algorithms, MPTCP-SSCC only increases the CPU utilization by 7% at most, and the memory utilization by 5% at most. The main source of extra overhead here is the CC Selection Module in user space, because it needs to save the performance statistics related to subflows and call the trained XGBoost models to predict the congestion used in the next decision cycle for subflows. The performance overhead here has an impact on the maximum number of concurrent MPTCP connections that the server can support. However, considering the improvement in transport performance brought by MPTCP-SSCC, we believe that this part of the system overhead is within an acceptable range.

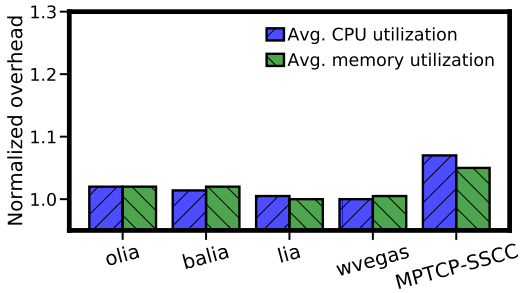


Figure 5.23 The results of system overhead comparison

5.6 Conclusion

There are many opportunities in Mobile Internet to utilize multipath transport protocols to improve transport performance. As an important component in multipath transport protocols, the efficiency of congestion control is crucial to achieve good performance. To address the problem that the existing multipath congestion control algorithms can not adapt to the complicated, heterogeneous and dynamically changing network environment in Mobile Internet scenarios under the premise of satisfying multipath fairness, resulting in performance degradation, we propose a MPTCP-SSCC system to implement a data-driven multipath congestion control algorithm selection mechanism. By sensing the network conditions of subflows, the MPTCP-SSCC system dynamically adjusts the appropriate congestion control algorithm for subflows in a data-driven approach, which can make the congestion control of subflows more adaptable to the network environments while achieving multipath fairness. The experimental results show that the MPTCP-SSCC system can select a coupled congestion control algorithm for subflows when necessary to satisfy multipath fairness, and choose appropriate independent congestion control algorithms for subflows in other cases. Compared

with the state-of-the-art multipath congestion control algorithms, MPTCP-SSCC improves the average throughput by 19% – 25%, and reduces the average queuing delay by up to 21%.

Chapter 6

Conclusion

In the era of Mobile Internet, people hope to enjoy high-quality Internet services anytime and anywhere. To meet this expectation, efficient network transport is essential. This paper conducts an in-depth study of the transport mechanism in Mobile Internet, focusing on two key parts that affect the transport performance, namely, the network access and the congestion control of transport protocols, because the network access affects the upper bound of the transport performance and the congestion control of transport protocols is a key factor in determining whether the transmission capacity provided by the network access can be fully utilized.

From the perspective of network access, users should consider all possible network access opportunities and make full use of all of them to improve transport performance and reduce costs [26, 93]. Since various network access types may be quite different, it is necessary to measure their availability, transport performance, and network characteristics. These measurement results are of great value for network access type selection and the optimization of transport protocols and congestion control algorithms. From the perspective of congestion control, whether single-path transport protocols or multipath transport protocols is used, it is very challenging to implement efficient congestion control when the network quality is highly dynamic due to the introduction of mobility in Mobile Internet. To this end, this dissertation carries out the following three research work from the three aspects of network access measurement, single-path congestion control, and multipath congestion control in Mobile Internet, with the goal of improving transport performance.

1) WiFi performance measurement in mobile scenarios: Mobile users have the opportunity to utilize provider-managed WiFi networks in urban areas to supplement the transport capabilities of cellular networks, thereby reducing transmission costs and improving overall transport efficiency. However, so far, there is no relevant measurement research to measure the availability, transport performance, and network characteristics of this type of access.

In order to fill this gap, we designed a measurement system based on in-vehicle devices to measure and analyze the transport capabilities provided by the provider-managed WiFi networks in mobile scenarios in four typical cities in Europe, Asia, and North America. In addition, our measurement results have certain implications for AP selection, connection establishment process optimization, and the optimization of transport protocols and congestion control algorithms.

2) An enhanced BBRv2 congestion control algorithm based on delay information: In Mobile Internet scenarios, high jitter, high packet loss rates, and large bandwidth changes are inevitable in the network. This dissertation starts from BBRv2 that has good TCP fairness and resilience to packet losses to study how to implement efficient congestion control. For this reason, we firstly compare and analyze BBR and BBRv2 through measurement, find the problems of BBRv2 in dealing with the dynamic network conditions, and explain the reasons for the problems. Then, we propose a new congestion control algorithm BBRv2+ based on BBRv2, which adopts the bandwidth probing based on the delay information to enhance the ability to follow bandwidth changes, and the window compensation mechanism based on jitter to deal with the performance issue caused by high jitter. The experimental results show that BBRv2+ improves the throughput by 25% in the mobile scenarios, compared with BBRv2.

3) A data-driven multipath congestion control algorithm selection mechanism: Many mobile devices currently support simultaneous access to multiple networks to enhance transport performance using multipath transport protocols. However, the multipath network environments usually face the problems of path heterogeneity and network quality dynamics, which challenges the adaptability of congestion control algorithms to the environments. To this end, we design a lightweight system MPTCP-SSCC to implement a data-driven multipath congestion control algorithm selection mechanism, in which a combination of ML and artificial rules is used to dynamically select appropriate congestion control algorithms for subflows on different paths. In this way, MPTCP-SSCC can enhance transport performance on the premise of satisfying multipath fairness. Compared with the traditional multipath congestion control algorithms or the latest learning-based multipath congestion control algorithm, the system can improve the average throughput by 19% – 25%, and reduce the average queuing delay by up to 21%.

On the basis of the above research work, we will try to conduct research from the following aspects in the future.

1) WiFi and cellular cooperative transmission optimization in mobile scenarios: In this dissertation, we mainly measure the transmission capability provided by provider-managed WiFi networks in mobile scenarios. The future plan is to study how to use

multipath transport protocols such as MPTCP and MPQUIC to use cellular and WiFi for cooperative transmission. In this scenario, the connectivity provided by WiFi is highly dynamic. Therefore, how to make multipath transport protocols to quickly utilize the transmission capacity when WiFi is available and how to properly schedule data on cellular and WiFi paths are challenging research problems.

2) Design and optimization of transport protocols with cross-layer information: In this dissertation, we mainly optimize the transport performance from the perspective of the transport layer. With the wide adoption of userspace protocol stacks such as QUIC [62] in Mobile Internet scenarios, it has become a research trend to tightly integrate the requirements of applications and the design of transport protocols [99, 150]. Taking application requirements into consideration is helpful for the on demand fine-grained control in transport protocols, such as choosing aggressive or conservative congestion control strategies, different packet loss recovery strategies, different network coding strategies, and different packet scheduling strategies based on application requirements.

Bibliography

[1] Abbasloo, S., Xu, Y., and Chao, H. J. (2019). C2TCP: A Flexible Cellular TCP to Meet Stringent Delay Requirements. *IEEE Journal on Selected Areas in Communications*, 37(4):918–932.

[2] Abbasloo, S., Yen, C.-Y., and Chao, H. J. (2020). Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 632–647, Virtual Event USA. ACM.

[3] Abbasloo, S., Yen, C.-Y., and Chao, H. J. (2021). Wanna Make Your TCP Scheme Great for Cellular Networks? Let Machines Do It for You! *IEEE Journal on Selected Areas in Communications*, 39(1):265–279.

[4] Afanasyev, A., Tilley, N., Reiher, P., and Kleinrock, L. (2010). Host-to-Host Congestion Control for TCP. *IEEE Communications Surveys & Tutorials*, 12(3):304–342.

[5] Aissaoui, H., Urien, P., and Pujolle, G. (2013). Low Latency of Re-authentication during Handover - Re-authentication using a Signed Token in Heterogeneous Wireless Access Networks:. In *Proceedings of the 10th International Conference on Signal Processing and Multimedia Applications and 10th International Conference on Wireless Information Networks and Systems*, pages 248–254, Reykjavík, Iceland. SCITEPRESS - Science and and Technology Publications.

[6] Akamai (2017). Akamai Online Retail Performance Report: Milliseconds Are Critical. https://www.akamai.com/newsroom/press-release/akamai-releases-spring-2017-state-of-online-retail-performance-report.

[7] Al-Saadi, R., Armitage, G., But, J., and Branch, P. (Fourthquarter 2019). A Survey of Delay-Based and Hybrid TCP Congestion Control Algorithms. *IEEE Communications Surveys Tutorials*, 21(4):3609–3638.

[8] Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and Sridharan, M. (2010). Data Center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 63–74, New York, NY, USA. ACM.

[9] Balasubramanian, A., Mahajan, R., and Venkataramani, A. (2010). Augmenting mobile 3G using WiFi. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services - MobiSys '10*, page 209, San Francisco, California, USA. ACM Press.

[10] Beshay, J. D., Nasrabadi, A. T., Prakash, R., and Francini, A. (2017). Link-Coupled TCP for 5G networks. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pages 1–6.

[11] Brakmo, L. (2019). TCP-BPF: Programmatically tuning TCP behavior through BPF. Technical report.

[12] Brakmo, L. and Peterson, L. (Oct./1995). TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480.

[13] Bychkovsky, V., Hull, B., Miu, A., Balakrishnan, H., and Madden, S. (2006). A measurement study of vehicular internet access using in situ Wi-Fi networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking - MobiCom '06*, page 50, Los Angeles, CA, USA. ACM Press.

[14] Cao, Y., Jain, A., Sharma, K., Balasubramanian, A., and Gandhi, A. (2019). When to use and when not to use BBR: An empirical analysis and evaluation study. In *Proceedings of the Internet Measurement Conference*, pages 130–136, Amsterdam Netherlands. ACM.

[15] Cao, Y., Xu, M., and Fu, X. (2012). Delay-based congestion control for multipath TCP. In *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–10, Austin, TX, USA. IEEE.

[16] Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., and Jacobson, V. (2016). BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue*, 14(5):20–53.

[17] Cardwell, N., Cheng, Y., Yeganeh, S. H., Jha, P., Seung, Y., Swett, I., Vasiliev, V., Wu, B., Mathis, M., and Jacobson, V. (2019a). BBR v2: A Model-based Congestion Control IETF 105 Update.

[18] Cardwell, N., Cheng, Y., Yeganeh, S. H., Swett, I., Vasiliev, V., Jha, P., Seung, Y., Mathis, M., and Jacobson, V. (2019b). BBR v2 A Model-based Congestion Control.

[19] Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M., and Wang, R. (2002). TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks. *Wireless Networks*, 8(5):467–479.

[20] Chen, K., Shan, D., Luo, X., Zhang, T., Yang, Y., and Ren, F. (2020). One Rein to Rule Them All: A Framework for Datacenter-to-User Congestion Control. In *4th Asia-Pacific Workshop on Networking*, APNet '20, pages 44–51, New York, NY, USA. Association for Computing Machinery.

[21] Chen, Y.-C., Lim, Y.-s., Gibbens, R. J., Nahum, E. M., Khalili, R., and Towsley, D. (2013). A measurement-based study of MultiPath TCP performance over wireless networks. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, pages 455–468, Barcelona Spain. ACM.

[22] Cheng Peng Fu and Liew, S. (2003). TCP Veno: TCP enhancement for transmission over wireless access networks. *IEEE Journal on Selected Areas in Communications*, 21(2):216–228.

[23] Chitimalla, D., Kondepu, K., Valcarenghi, L., Tornatore, M., and Mukherjee, B. (2017). 5G fronthaul-latency and jitter studies of CPRI over ethernet. *IEEE/OSA Journal of Optical Communications and Networking*, 9(2):172–182.

[24] De Coninck, Q. and Bonaventure, O. (2017). Multipath QUIC: Design and Evaluation. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies - CoNEXT '17*, pages 160–166, Incheon, Republic of Korea. ACM Press.

[25] Deng, S., Netravali, R., Sivaraman, A., and Balakrishnan, H. (2014). WiFi, LTE, or Both?: Measuring Multi-Homed Wireless Internet Performance. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 181–194, Vancouver BC Canada. ACM.

[26] Deng, S., Sivaraman, A., and Balakrishnan, H. (2016). Delphi: A Software Controller for Mobile Network Selection. Technical report.

[27] Deshpande, P., Hou, X., and Das, S. R. (2010). Performance comparison of 3G and metro-scale WiFi for vehicular network access. In *Proceedings of the 10th Annual Conference on Internet Measurement - IMC '10*, page 301, Melbourne, Australia. ACM Press.

[28] Dong, M., Li, Q., Zarchy, D., Godfrey, P. B., and Schapira, M. (2015). PCC: Re-architecting congestion control for consistent high performance. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 395–408, Oakland, CA. USENIX Association.

[29] Dong, M., Meng, T., Zarchy, D., Arslan, E., Gilad, Y., Godfrey, B., and Schapira, M. (2018). PCC Vivace: Online-Learning Congestion Control. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 343–356, Renton, WA. USENIX Association.

[30] Dong, P., Wang, J., Huang, J., Wang, H., and Min, G. (2016). Performance Enhancement of Multipath TCP for Wireless Communications With Multiple Radio Interfaces. *IEEE Transactions on Communications*, 64(8):3456–3466.

[31] Droms, R. (1997). Dynamic Host Configuration Protocol. Technical Report RFC2131, RFC Editor.

[32] Du, Z., Zheng, J., Yu, H., Kong, L., and Chen, G. (2021). A unified congestion control framework for diverse application preferences and network conditions. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '21, pages 282–296, New York, NY, USA. Association for Computing Machinery.

[33] Einav, Y. (2022). Amazon found every 100ms of latency cost them 1% in sales. https://www.gigaspaces.com/blog/amazon-found-every-100ms-of-latency-cost-them-1-in-sales.

[34] Eriksson, J., Balakrishnan, H., and Madden, S. (2008). Cabernet: Vehicular content delivery using WiFi. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking - MobiCom '08*, page 199, San Francisco, California, USA. ACM Press.

[35] Esnet (2021). Esnet/iperf. https://github.com/esnet/iperf.

[36] Ferlin, S., Alay, O., Dreibholz, T., Hayes, D. A., and Welzl, M. (2016). Revisiting congestion control for multipath TCP with shared bottleneck detection. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, San Francisco, CA, USA. IEEE.

[37] Floyd, S. (2008). Metrics for the Evaluation of Congestion Control Mechanisms. Request for Comments RFC 5166, Internet Engineering Task Force.

[38] Ford, A., Raiciu, C., Handley, M., and Bonaventure, O. (2013). TCP Extensions for Multipath Operation with Multiple Addresses. Technical Report RFC6824, RFC Editor.

[39] Free (2022). Free. https://www.free.fr/freebox/.

[40] Gass, R., Scott, J., and Diot, C. (2006). Measurements of In-Motion 802.11 Networking. In *Seventh IEEE Workshop on Mobile Computing Systems & Applications (WMCSA'06)*, pages 69–74, Orcas Island, WA, USA. IEEE.

[41] Gettys, J. and Nichols, K. (2012). Bufferbloat: Dark buffers in the internet. *Communications of the ACM*, 55(1):57–65.

[42] Giessler, A., Hänle, J., König, A., and Pade, E. (1978). Free buffer allocation — An investigation by simulation. *Computer Networks (1976)*, 2(3):191–208.

[43] Gilad, T., Rozen-Schiff, N., Godfrey, P. B., Raiciu, C., and Schapira, M. (2020). MPCC: Online learning multipath transport. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, pages 121–135, Barcelona Spain. ACM.

[44] Gomez, J., Kfoury, E., Crichigno, J., Bou-Harb, E., and Srivastava, G. (2020). A Performance Evaluation of TCP BBRv2 Alpha. In *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, pages 309–312, Milan, Italy. IEEE.

[45] Google (2021). Google/bbr. https://github.com/google/bbr/tree/v2alpha.

[46] Grieco, L. A. and Mascolo, S. (2004). Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. *ACM SIGCOMM Computer Communication Review*, 34(2):25–38.

[47] Hadaller, D., Keshav, S., Brecht, T., and Agarwal, S. (2007). Vehicular opportunistic communication under the microscope. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services - MobiSys '07*, page 206, San Juan, Puerto Rico. ACM Press.

[48] Han, J., Xue, K., Xing, Y., Li, J., Wei, W., Wei, D. S. L., and Xue, G. (2021). Leveraging Coupled BBR and Adaptive Packet Scheduling to Boost MPTCP. *IEEE Transactions on Wireless Communications*, 20(11):7555–7567.

[49] Hassayoun, S., Iyengar, J., and Ros, D. (2011). Dynamic Window Coupling for multipath congestion control. In *2011 19th IEEE International Conference on Network Protocols*, pages 341–352, Vancouver, AB, Canada. IEEE.

[50] Hayes, D., Ferlin, S., Welzl, M., and Hiorth, K. (2018). Shared Bottleneck Detection for Coupled Congestion Control for RTP Media. Request for Comments RFC 8382, Internet Engineering Task Force.

[51] Hayes, D. A., Welzl, M., Ferlin, S., Ros, D., and Islam, S. (2020). Online Identification of Groups of Flows Sharing a Network Bottleneck. *IEEE/ACM Transactions on Networking*, 28(5):2229–2242.

[52] He, B., Wang, J., Qi, Q., Sun, H., Liao, J., Du, C., Yang, X., and Han, Z. (2021). DeepCC: Multi-Agent Deep Reinforcement Learning Congestion Control for Multi-Path TCP Based on Self-Attention. *IEEE Transactions on Network and Service Management*, 18(4):4770–4788.

[53] Hemminger, S. (2021). Tc-netem(8) - Linux manual page. https://www.man7.org/linux/man-pages/man8/tc-netem.8.html.

[54] Henderson, T., Floyd, S., Gurtov, A., and Nishida, Y. (2012). The NewReno Modification to TCP's Fast Recovery Algorithm. Technical Report RFC6582, RFC Editor.

[55] Hesmans, B., Bonaventure, O., and Duchene, F. (2018). A socket API to control Multipath TCP. Internet Draft draft-hesmans-mptcp-socket-03, Internet Engineering Task Force.

[56] Hock, M., Bless, R., and Zitterbart, M. (2017). Experimental evaluation of BBR congestion control. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–10.

[57] Huang, J., Qian, F., Guo, Y., Zhou, Y., Xu, Q., Mao, Z. M., Sen, S., and Spatscheck, O. (2013). An in-depth study of LTE: Effect of network protocol and application behavior on performance. *ACM SIGCOMM Computer Communication Review*, 43(4):363–374.

[58] Hubert, B. (2021). Tc(8) - Linux manual page. https://man7.org/linux/man-pages/man8/tc.8.html.

[59] Huffaker, B., Fomenkov, M., Plummer, D. J., Moore, D., Claffy, K., et al. (2002). Distance metrics in the Internet. In *Proc. of IEEE International Telecommunications Symposium (ITS)*.

[60] IO-Visor-Project (2022). BPF Compiler Collection (BCC). https://github.com/iovisor/bcc.

[61] IP-Networking-Lab-UCLouvain (2022). MultiPath TCP. https://multipath-tcp.org/.

[62] Iyengar, J. and Thomson, M. (2021). QUIC: A UDP-Based Multiplexed and Secure Transport. Request for Comments RFC 9000, Internet Engineering Task Force.

[63] Jacobson, V. (1988). Congestion avoidance and control. In *Symposium Proceedings on Communications Architectures and Protocols - SIGCOMM '88*, pages 314–329, Stanford, California, United States. ACM Press.

[64] Jang, K., Han, M., Cho, S., Ryu, H.-K., Lee, J., Lee, Y., and Moon, S. B. (2009). 3G and 3.5G wireless network performance measured from moving cars and high-speed trains. In *Proceedings of the 1st ACM Workshop on Mobile Internet through Cellular Networks - MICNET '09*, page 19, Beijing, China. ACM Press.

[65] Jay, N., Rotman, N., Godfrey, B., Schapira, M., and Tamar, A. (2019). A Deep Reinforcement Learning Perspective on Internet Congestion Control. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3050–3059. PMLR.

[66] Kfoury, E. F., Gomez, J., Crichigno, J., and Bou-Harb, E. (2020). An emulation-based evaluation of TCP BBRv2 Alpha for wired broadband. *Computer Communications*, 161:212–224.

[67] Khalili, R., Gast, N., Popovic, M., and Le Boudec, J.-Y. (2013). MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution. *IEEE/ACM Transactions on Networking*, 21(5):1651–1665.

[68] Kim, G.-H. and Cho, Y.-Z. (2020). Delay-Aware BBR Congestion Control Algorithm for RTT Fairness Improvement. *IEEE Access*, 8:4099–4109.

[69] Kleinrock, L. (1979). Power and deterministic rules of thumb for probabilistic problems in computer communications. In *Proceedings of the International Conference on Communications*, volume 43, pages 1–43.

[70] Kumar, R., Koutsaftis, A., Fund, F., Naik, G., Liu, P., Liu, Y., and Panwar, S. (2019). TCP BBR for Ultra-Low Latency Networking: Challenges, Analysis, and Solutions. In *2019 IFIP Networking Conference (IFIP Networking)*, pages 1–9.

[71] Langley, A., Iyengar, J., Bailey, J., Dorfman, J., Roskind, J., Kulik, J., Westin, P., Tenneti, R., Shade, R., Hamilton, R., Vasiliev, V., Riddoch, A., Chang, W.-T., Shi, Z., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., and Swett, I. (2017). The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication - SIGCOMM '17*, pages 183–196, Los Angeles, CA, USA. ACM Press.

[72] Lee, H., Flinn, J., and Tonshal, B. (2018). RAVEN: Improving Interactive Latency for the Connected Car. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking - MobiCom '18*, pages 557–572, New Delhi, India. ACM Press.

[73] Lee, K., Lee, J., Yi, Y., Rhee, I., and Chong, S. (2013). Mobile Data Offloading: How Much Can WiFi Deliver? *IEEE/ACM Transactions on Networking*, 21(2):536–550.

[74] Li, L., Xu, K., Li, T., Zheng, K., Peng, C., Wang, D., Wang, X., Shen, M., and Mijumbi, R. (2018). A measurement study on multi-path TCP with multiple cellular carriers on high speed rails. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication - SIGCOMM '18*, pages 161–175, Budapest, Hungary. ACM Press.

[75] Li, L., Xu, K., Wang, D., Peng, C., Zheng, K., Mijumbi, R., and Xiao, Q. (2017). A Longitudinal Measurement Study of TCP Performance and Behavior in 3G/4G Networks Over High Speed Rails. *IEEE/ACM Transactions on Networking*, 25(4):2195–2208.

[76] Li, W., Zhang, H., Gao, S., Xue, C., Wang, X., and Lu, S. (2019). SmartCC: A Reinforcement Learning Approach for Multipath TCP Congestion Control in Heterogeneous Networks. *IEEE Journal on Selected Areas in Communications*, 37(11):2621–2633.

[77] Li, X., Tang, F., Liu, J., Yang, L. T., Fu, L., and Chen, L. (2021). AUTO: Adaptive congestion control based on multi-objective reinforcement learning for the satellite-ground integrated network. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 611–624. USENIX Association.

[78] Litjens, R. (2005). HSDPA flow level performance and the impact of terminal mobility. In *IEEE Wireless Communications and Networking Conference, 2005*, volume 3, pages 1657–1663, New Orleans, LA, USA. IEEE.

[79] Liu, Q., Xu, K., Wang, H., Shen, M., Li, L., and Xiao, Q. (2016). Measurement, Modeling, and Analysis of TCP in High-Speed Mobility Scenarios. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 629–638.

[80] Liu, S., Başar, T., and Srikant, R. (2006). TCP-Illinois: A loss and delay-based congestion control algorithm for high-speed networks. In *Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools - Valuetools '06*, page 55, Pisa, Italy. ACM Press.

[81] Liu, Y., Ma, Y., Huitema, C., An, Q., and Li, Z. (2021). Multipath Extension for QUIC. Internet Draft draft-liu-multipath-quic-04, Internet Engineering Task Force.

[82] Ma, S., Jiang, J., Wang, W., and Li, B. (2017). Fairness of Congestion-Based Congestion Control: Experimental Evaluation and Analysis. *arXiv:1706.09115 [cs]*.

[83] Mahajan, R., Zahorjan, J., and Zill, B. (2007). Understanding wifi-based connectivity from moving vehicles. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement - IMC '07*, page 321, San Diego, California, USA. ACM Press.

[84] Mahmud, I., Kim, G.-H., Lubna, T., and Cho, Y.-Z. (2020). BBR-ACD: BBR with Advanced Congestion Detection. *Electronics*, 9(1):136.

[85] Melki, R., Mansour, M. M., and Chehab, A. (2018). A fairness-based congestion control algorithm for multipath TCP. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, Barcelona. IEEE.

[86] Merz, R., Wenger, D., Scanferla, D., and Mauron, S. (2014). Performance of LTE in a high-velocity environment: A measurement study. In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges - AllThingsCellular '14*, pages 47–52, Chicago, Illinois, USA. ACM Press.

[87] Miano, S., Bertrone, M., Risso, F., Tumolo, M., and Bernal, M. V. (2018). Creating Complex Network Services with eBPF: Experience and Lessons Learned. In *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–8.

[88] Mininet (2021). Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet. http://mininet.org/.

[89] Mishra, A., Sun, X., Jain, A., Pande, S., Joshi, R., and Leong, B. (2019). The Great Internet TCP Congestion Control Census. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–24.

[90] Nandagiri, A., Tahiliani, M. P., Misra, V., and Ramakrishnan, K. K. (2020). BBRvl vs BBRv2: Examining Performance Differences through Experimental Evaluation. In *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN*, pages 1–6, Orlando, FL, USA. IEEE.

[91] Narayanan, A., Zhang, X., Zhu, R., Hassan, A., Jin, S., Zhu, X., Zhang, X., Rybkin, D., Yang, Z., Mao, Z. M., Qian, F., and Zhang, Z.-L. (2021). A variegated look at 5G in the wild: Performance, power, and QoE implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 610–625, Virtual Event USA. ACM.

[92] Netravali, R., Sivaraman, A., Das, S., Goyal, A., Winstein, K., Mickens, J., and Balakrishnan, H. (2015). Mahimahi: Accurate record-and-replay for HTTP. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, pages 417–429, Santa Clara, CA. USENIX Association.

[93] Nicholson, A. J. and Noble, B. D. (2008). BreadCrumbs: Forecasting mobile connectivity. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking - MobiCom '08*, page 46, San Francisco, California, USA. ACM Press.

[94] Nie, X., Zhao, Y., Li, Z., Chen, G., Sui, K., Zhang, J., Ye, Z., and Pei, D. (2019). Dynamic TCP Initial Windows and Congestion Control Schemes Through Reinforcement Learning. *IEEE Journal on Selected Areas in Communications*, 37(6):1231–1247.

[95] Nikravesh, A., Guo, Y., Qian, F., Mao, Z. M., and Sen, S. (2016). An in-depth understanding of multipath TCP on mobile devices: Measurement and system design. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 189–201, New York City New York. ACM.

[96] Ostermann (2021). Tcptrace(1): TCP connection analysis tool - Linux man page. https://linux.die.net/man/1/tcptrace.

[97] Ott, J. and Kutscher, D. (2004). Drive-thru internet: IEEE 802.1 1b for "automobile" users. In *IEEE INFOCOM 2004*, volume 1, pages 362–373, Hong Kong, PR China. IEEE.

[98] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J. (2000). Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145.

[99] Palmer, M., Appel, M., Spiteri, K., Chandrasekaran, B., Feldmann, A., and Sitaraman, R. K. (2021). VOXEL: Cross-layer optimization for video streaming with imperfect transmission. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '21, pages 359–374, New York, NY, USA. Association for Computing Machinery.

[100] Pan, Y., Li, R., and Xu, C. (2022). The First 5G-LTE Comparative Study in Extreme Mobility. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(1):1–22.

[101] PEAP (2021). Protocol/EAP PEAP. https://wiki.freeradius.org/protocol/EAP-PEAP.

[102] Peng, Q., Walid, A., Hwang, J., and Low, S. H. (2016). Multipath TCP: Analysis, Design, and Implementation. *IEEE/ACM Transactions on Networking*, 24(1):596–609.

[103] Pesavento, D., Grassi, G., Pau, G., Bahl, P., and Fdida, S. (2015). Demo: Car-Fi: Opportunistic V2I by Exploiting Dual-Access Wi-Fi Networks. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 173–175, Paris France. ACM.

[104] Pokhrel, S. R. and Walid, A. (2021). Learning to Harness Bandwidth with Multipath Congestion Control and Scheduling. *IEEE Transactions on Mobile Computing*, pages 1–1.

[105] Raiciu, C., Handley, M., and Wischik, D. (2011). Coupled Congestion Control for Multipath Transport Protocols. Technical Report RFC6356, RFC Editor.

[106] Ravinet (2022). Mpshell. https://github.com/ravinet/mahimahi/tree/old/mpshell_scripted.

[107] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and Scheffenegger, R. (2018). CUBIC for Fast Long-Distance Networks. Technical Report RFC8312, RFC Editor.

[108] Rodrigues, A. (2019). Improving Vehicular WiFi Performance with Data-driven Brokerage. Technical report, Carnegie Mellon University, Pittsburgh, PA 15213.

[109] Salowey, J. A. and Haverinen, H. (2006). Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). Request for Comments RFC 4186, Internet Engineering Task Force.

[110] Sathyanarayana, S. D., Lee, J., Lee, J., Grunwald, D., and Ha, S. (2021). Exploiting Client Inference in Multipath TCP Over Multiple Cellular Networks. *IEEE Communications Magazine*, 59(4):58–64.

[111] Scholz, D., Jaeger, B., Schwaighofer, L., Raumer, D., Geyer, F., and Carle, G. (2018). Towards a Deeper Understanding of TCP BBR Congestion Control. In *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–9.

[112] Shimonishi, H., Sanadidi, M., and Gerla, M. (2005). Improving efficiency-friendliness tradeoffs of TCP in wired-wireless combined networks. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 5, pages 3548–3552 Vol. 5.

[113] Song, Y.-J., Kim, G.-H., and Cho, Y.-Z. (2020). BBR-CWS: Improving the Inter-Protocol Fairness of BBR. *Electronics*, 9(5):862.

[114] Song, Y.-J., Kim, G.-H., Mahmud, I., Seo, W.-K., and Cho, Y.-Z. (2021). Understanding of BBRv2: Evaluation and Comparison with BBRv1 Congestion Control Algorithm. *IEEE Access*, pages 1–1.

[115] Soroush, H., Banerjee, N., Corner, M., Levine, B., and Lynn, B. (2012). A retrospective look at the UMass DOME mobile testbed. *ACM SIGMOBILE Mobile Computing and Communications Review*, 15(4):2–15.

[116] Soroush, H., Gilbert, P., Banerjee, N., Levine, B. N., Corner, M., and Cox, L. (2011). Concurrent Wi-Fi for mobile users: Analysis and measurements. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies on - CoNEXT '11*, pages 1–12, Tokyo, Japan. ACM Press.

[117] Statista (2022). Mobile internet usage worldwide - statistics & facts. https://www.statista.com/topics/779/mobile-internet/#dossierKeyfigures.

[118] The-Tcpdump-Group (2021). TCPDUMP/LIBPCAP public repository. https://www.tcpdump.org/.

[119] Thomas, Y., Karaliopoulos, M., Xylomenos, G., and Polyzos, G. C. (2020a). Low Latency Friendliness for Multipath TCP. *IEEE/ACM Transactions on Networking*, 28(1):248–261.

[120] Thomas, Y., Xylomenos, G., and Polyzos, G. C. (2020b). Multipath congestion control with network assistance. *Computer Communications*, 153:264–278.

[121] Tso, F. P., Teng, J., Jia, W., and Xuan, D. (2012). Mobility: A Double-Edged Sword for HSPA Networks: A Large-Scale Test on Hong Kong Mobile HSPA Networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(10):1895–1907.

[122] Vargas, S., Drucker, R., Renganathan, A., Balasubramanian, A., and Gandhi, A. (2021). BBR Bufferbloat in DASH Video. In *Proceedings of the Web Conference 2021*, pages 329–341, Ljubljana Slovenia. ACM.

[123] Vollbrecht, J., Carlson, J. D., Blunk, L., Aboba, B. D., and Levkowetz, H. (2004). Extensible Authentication Protocol (EAP). Request for Comments RFC 3748, Internet Engineering Task Force.

[124] Wang, J., Zheng, Y., Ni, Y., Xu, C., Qian, F., Li, W., Jiang, W., Cheng, Y., Cheng, Z., Li, Y., Xie, X., Sun, Y., and Wang, Z. (2019). An Active-Passive Measurement Study of TCP Performance over LTE on High-speed Rails. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, Los Cabos Mexico. ACM.

[125] Wang, R., Valla, M., Sanadidi, M., and Gerla, M. (2002a). Adaptive bandwidth share estimation in TCP Westwood. In *IEEE Global Telecommunications Conference, 2002. GLOBECOM '02*, volume 3, pages 2604–2608 vol.3.

[126] Wang, R., Valla, M., Sanadidi, M., Ng, B., and Gerla, M. (2002b). Efficiency/friendliness tradeoffs in TCP Westwood. In *Proceedings ISCC 2002 Seventh International Symposium on Computers and Communications*, pages 304–311.

[127] Ware, R., Mukerjee, M. K., Seshan, S., and Sherry, J. (2019). Modeling BBR's Interactions with Loss-Based Congestion Control. In *Proceedings of the Internet Measurement Conference*, pages 137–143, Amsterdam Netherlands. ACM.

[128] Wei, W., Xue, K., Han, J., Wei, D. S. L., and Hong, P. (2020). Shared Bottleneck-Based Congestion Control and Packet Scheduling for Multipath TCP. *IEEE/ACM Transactions on Networking*, 28(2):653–666.

[129] Wikipedia (2021). Wi-Fi Protected Access. https://en.wikipedia.org/w/index.php?title=Wi-Fi_Protected_Access&oldid=1031381932.

[130] Winstein, K. and Balakrishnan, H. (2013). TCP ex machina: Computer-generated congestion control. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, pages 123–134, Hong Kong China. ACM.

[131] Winstein, K., Sivaraman, A., and Balakrishnan, H. (2013). Stochastic forecasts achieve high throughput and low delay over cellular networks. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 459–471, Lombard, IL. USENIX Association.

[132] Wu, H., Alay, Ö., Brunstrom, A., Ferlin, S., and Caso, G. (2020). Peekaboo: Learning-Based Multipath Scheduling for Dynamic Heterogeneous Environments. *IEEE Journal on Selected Areas in Communications*, 38(10):2295–2310.

[133] XGBoost (2022). XGBoost Documentation — xgboost 1.5.2 documentation. https://xgboost.readthedocs.io/en/stable/index.html.

[134] Xiao, Q., Xu, K., Wang, D., Li, L., and Zhong, Y. (2014). TCP Performance over Mobile Networks in High-Speed Mobility Scenarios. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 281–286, Raleigh, NC, USA. IEEE.

[135] Xu, C., Qin, J., Zhang, P., Gao, K., and Grieco, L. A. (2021). Reinforcement Learning-based Mobile AR/VR Multipath Transmission with Streaming Power Spectrum Density Analysis. *IEEE Transactions on Mobile Computing*, pages 1–1.

[136] Xu, D., Zhou, A., Zhang, X., Wang, G., Liu, X., An, C., Shi, Y., Liu, L., and Ma, H. (2020). Understanding Operational 5G: A First Measurement Study on Its Coverage, Performance and Energy Consumption. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 479–494, Virtual Event USA. ACM.

[137] Xu, Z., Tang, J., Yin, C., Wang, Y., and Xue, G. (2019). Experience-Driven Congestion Control: When Multi-Path TCP Meets Deep Reinforcement Learning. *IEEE Journal on Selected Areas in Communications*, 37(6):1325–1336.

[138] Yan, F. Y., Ma, J., Hill, G. D., Raghavan, D., Wahby, R. S., Levis, P., and Winstein, K. (2018). Pantheon: The training ground for Internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 731–743, Boston, MA. USENIX Association.

[139] Yang, F., Wu, Q., Li, Z., Liu, Y., Pau, G., and Xie, G. (2022). BBRv2+: Towards balancing aggressiveness and fairness with delay-based bandwidth probing. *Computer Networks*, 206:108789.

[140] Yang, G., Wang, R., Sanadidi, M., and Gerla, M. (2003). TCPW with bulk repeat in next generation wireless networks. In *IEEE International Conference on Communications, 2003. ICC '03.*, volume 1, pages 674–678 vol.1.

[141] Yang, M., Yang, P., Wen, C., Liu, Q., Luo, J., and Yu, L. (2019). Adaptive-BBR: Fine-Grained Congestion Control with Improved Fairness and Low Latency. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6.

[142] Yao, J., Kanhere, S. S., and Hassan, M. (2008). An empirical study of bandwidth predictability in mobile computing. In *Proceedings of the Third ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization - WiNTECH '08*, page 11, San Francisco, California, USA. ACM Press.

[143] Ye, J., Liu, R., Xie, Z., Feng, L., and Liu, S. (2019). EMPTCP: An ECN Based Approach to Detect Shared Bottleneck in MPTCP. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–10, Valencia, Spain. IEEE.

[144] Yeganeh, S. H., Jha, P., Seung, Y., Hsiao, L., Mathis, M., and Jacobson, V. (2021). BBR Updates: Internal Deployment, Code, Draft Plans.

[145] Yu, H., Zheng, J., Du, Z., and Chen, G. (2021). MPLibra: Complementing the Benefits of Classic and Learning-based Multipath Congestion Control. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pages 1–11, Dallas, TX, USA. IEEE.

[146] Zaki, Y., Pötsch, T., Chen, J., Subramanian, L., and Görg, C. (2015). Adaptive Congestion Control for Unpredictable Cellular Networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 509–522, London United Kingdom. ACM.

[147] Zhang, L., Shenker, S., and Clark, D. D. (1991). Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. *ACM SIGCOMM Computer Communication Review*, 21(4):133–147.

[148] Zhang, M., Polese, M., Mezzavilla, M., Zhu, J., Rangan, S., Panwar, S., and Zorzi, M. (2018a). Will TCP work in mmWave 5G Cellular Networks? *arXiv:1806.05783 [cs]*.

[149] Zhang, Y., Cui, L., and Tso, F. P. (2018b). Modest BBR: Enabling Better Fairness for BBR Congestion Control. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00646–00651.

[150] Zheng, Z., Ma, Y., Liu, Y., Yang, F., Li, Z., Zhang, Y., Zhang, J., Shi, W., Chen, W., Li, D., An, Q., Hong, H., Liu, H. H., and Zhang, M. (2021). XLINK: QoE-driven multi-path QUIC transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, pages 418–432, New York, NY, USA. Association for Computing Machinery.

[151] Zhou, H., Chen, X., He, S., Chen, J., and Wu, J. (2020). DRAIM: A Novel Delay-Constraint and Reverse Auction-Based Incentive Mechanism for WiFi Offloading. *IEEE Journal on Selected Areas in Communications*, 38(4):711–722.

[152] Zhou, J., Qiu, X., Li, Z., Tyson, G., Li, Q., Duan, J., and Wang, Y. (2021). Antelope: A Framework for Dynamic Selection of Congestion Control Algorithms. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pages 1–11, Dallas, TX, USA. IEEE.

[153] Zrelli, S., Okabe, N., and Shinoda, Y. (2012). EAP-Kerberos: A Low Latency EAP Authentication Method for Faster Handoffs in Wireless Access Networks. *IEICE Transactions on Information and Systems*, E95-D(2):490–502.