Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8863420, 17 pages
https://doi.org/10.1155/2020/8863420

Research Article

Learn-ing-Based On-AP TCP Performance Enhancement

Shirong Lin and Shouxu Jiang

School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

Correspondence should be addressed to Shouxu Jiang; jsx@hit.edu.cn

Received 18 March 2020; Revised 11 July 2020; Accepted 28 July 2020; Published 13 August 2020

Academic Editor: Mauro Femminella

Data transmissions suffer from TCP's poor performance since the introduction of the first commercial wireless services in the 1990s. Recent years have witnessed a surge of academia and industry activities in the field of TCP performance optimization. For a TCP flow whose last hop is a wireless link, congestions in the last hop dominate its performance. We implement an integral data sampling, network monitoring, and rate control software-defined wireless networking (SDWN) system. By analysing our sampled data, we find that there exist strong relationships between congestion packet loss behaviors and the instant cross-layer network metric measurements (states). We utilize these qualitative relationships to predict future congestions in wireless links and enhance TCP performance by launch necessary rate control locally on the access points (AP) before the congestions. We also implement modeling and rate control modules on this platform. Our platform senses the instant wireless dynamic and takes actions promptly to avoid future congestions. We conduct real-world experiments to evaluate its performance. The experiment results show that our methods outperform the bottleneck bandwidth and RTT (BBR) protocol and a recently proposed protocol Vivace on throughput, delay, and jitter performance at least 16.5%, 25%, and 12.6%, respectively.

1. Introduction

Mobile traffic and Wi-Fi traffic will account for more than 70.6% of all the IP traffics by 2022 [1]. At the same time, the 5G and 802.11 ax standards will be launched to the market soon. Moreover, machine learning and software-defined networking (SDN) have achieved great successes in practical applications. Among the 70.6% of data, the multimedia traffic, the virtual reality, and augmented reality data have stringent Quality of Service (QoS) constraints. It is apparent that today's prevalent congestion control algorithms, the TCP family, fall short of the critical performance requirement, especially on the last (wireless) hop [2–8]. In addition, current transport layer protocols are no longer suitable for the mmWave communication in 5G and Wi-Fi 6 [2, 9, 10]. As the primary functions of TCP protocols, congestion control algorithms are designed to probe and utilize the network capacity efficiently. However, they perform poorly in wireless networks whose capacity varies rapidly with the channel conditions, contentions, interferences, and mobilities.

Researchers of the traditional end-to-end TCP protocols focus on exploring suitable parameters and algorithms to infer the instant link capacity. BBR [7], performance-oriented congestion control (PCC) [8], and Vivace [4] are such protocols. In addition, research works in [9–12] point out that TCP benefits from a shorter control loop, congestion management should be pushed to the network edge, where the server can react faster to link impairments. Furthermore, some research works are focused on the explorations of the dominating factors that affect TCP performance. The correlations between wireless network measurements and TCP throughput are revealed in [13]. Research works in [14, 15] reveal the relationships between network latencies and wireless network measurements.

However, these research results cannot be used directly in TCP congestion avoidance, even though they do reveal the relationships between TCP performance and some of the

metrics, mainly due to the following four issues. Firstly, their researches are not about congestion, so information such as queue-related metrics are not utilized. Secondly, the congestion packet losses and noncongestion packet losses are not differentiated in their researches. There are two kinds of packet losses: noncongestion losses and congestion ones. The noncongestion losses happen in the medium access control (MAC) layer, while the congestion ones happen when the queues are overloaded. Data rates should be limited only when there are congestion packet drops. Thirdly, they do not have a complete and general system that contains data sampling modules, data processing modules, optimization modules, and control modules. Fourthly, their researches are built on low-frequency datasets, but high-frequency real-time data sampling is essential for congestion avoidance. Our research works in this article try to answer the following three questions:

(a) What does the network states look like when congestion happens?

(b) Which factors affect the congestions the most in wireless networks?

(c) How to avoid Wi-Fi congestions?

We explore the correlations between the cross-layer (transport layer, network layer, and MAC layer) network metrics and the congestions to enhance the TCP performance on commercial APs. The cross-layer network states before congestions happen are sampled to learn their qualitative relationships with the congestions. Then, we propose a learning-based model to predict future congestions based on the discovered correlations, and we also propose a receive window-based rate control method to avoid the congestions. Moreover, we implement the whole system (includes data sample, learning, control) in our SDWN platform and evaluate its performance.

To summarize, our main contributions are the following:

(1) We finish an enterprise-scale measurement study on Wi-Fi congestion losses. Our results reveal the correlations between congestion packet losses and the cross-layer network metrics in detail. We believe these results have deep implications for all participants in the mobile Internet ecosystem

(2) We model the rate allocation on wireless nodes as a utility maximization problem which yields optimal proportional fair data rates. The rates are then transformed into windows to control the TCP sending rates

(3) We solve the congestion window regression problem by exploring the correlations between potential network metrics and the congestion windows. The congestion window regression problem is crucial for TCP rate control on intermediate nodes

(4) We implement the whole system in our real-world SDWN platform, and this work is novel in TCP optimization. Moreover, we conduct practical experiments to evaluate its performance

The paper is organized as follows: the related works are introduced in Section 2. The motivations are introduced in Section 3. Our SDWN-based platform is introduced in Section 4. The congestion prediction is introduced in Section 4. The congestion prediction processes are introduced in Section 5. In Section 6, we model the rate allocation as a utility maximization problem, solve it, and obtain an optimal window. In Section 7, we introduce how we solve the congestion window regression problem on intermediate wireless node. In Section 8, we evaluate the performance of our platform. At last, we conclude our work in Section 9.

2. Related Works

Our works are focused on the congestion packet loss prediction and the implementations of SDWN-based TCP rate allocation. To this end, we introduce the related works from two aspects: the development of TCP protocols and the researches on the correlations between TCP performance and network states in wireless networks.

2.1. The Development of the TCP Protocols. We classify current TCP protocols into loss-based, delay-based, capacity-based, hybrid-based, proxy-based, ECN, learning-based, and cross-layer ones.

2.1.1. The Loss-Based and Delay-Based Protocols. Tahoe, Reno, New Reno, and CUBIC are typical loss-based TCP protocols. Loss-based protocols cannot differentiate the congested packet loss from the noncongested packet loss. Moreover, loss-based protocols are aggressive, especially to the delay-based protocols. TCP Vegas, New Vegas, and Verus are delay-based TCP protocols. Delay-based protocols are more stable and perform better on the retransmission and latency performance than the loss-based ones. Also, they can infer the congestions happen on the wireless links by monitoring the round-trip time (RTT). However, their adoption has been blocked for the aggressiveness of existing protocols.

2.1.2. The Hybrid Protocols. Compound TCP for Windows systems and BBR are hybrid protocols. Compound TCP contains both the loss-based mechanism and the delay-based mechanism. BBR estimates both the RTT and the link capacity; it is a hybrid delay-based and capacity-based protocol. BBR uses recent RTT and delivery rate to model the bandwidth-delay product (BDP) and varies pacing rate to keep inflight near the BDP (for the full pipe but small queue). BBR exhibits high rate variance and a high packet loss rate upon convergence. Also, it is aggressive towards other TCP algorithms.

2.1.3. The Learning-Based Protocols. TCP Remy [16], PCC [8], Vivace [4], Copa [5], TCP-RL [17], and QTCP [18] are learning-based protocols. Machine-learning methods will be more and more important for the TCP protocol designers. The Vivace model utility of all TCP flows that share the same bottleneck link as a function of delays and delivery rates. The authors prove there exists a Nash equilibrium for their

optimization, and they design an online learning algorithm by computing the gradient of the utility function to get the optimal pacing rate. Remy is an off-line machine learning method that aims to produce TCP protocols by mapping observed congestion signals to sender actions. Remy is a cooperative game method, and it needs intense off-line computation, and the performance of Remy depends on the accuracy of the network and traffic models. PCC and Vivace perform online noncooperative game optimizations. PCC is rate-based, and its performance depends on the accuracy of the clocking.

2.1.4. The Protocols for Wireless Networks. Some of the capacity-based protocols are designed to enhance the TCP performance in wireless and lossy links. TCP Westwood, Sprout are two such protocols. However, researches in [19] show that the throughput and delay performance of TCP Westwood is nearly the same as TCP CUBIC. Sprout seems to be unable to efficiently estimate the capacity of practical wireless links, even without cross-traffic [20].

2.1.5. The ECN and Proxy Approaches. ECN is an effective way to feedback the congestion reasons to the TCP sender to improve the performance. Passive support in the most popular websites has increased to over 70% in 2017. Forecasting the congestion and control the data rates with ECNs is one of our initial ideas. This idea seems perfect, the congestion prediction modules predict the congestions, and ECN is used to control the data rates, and it corresponds to fewer developing works and troubles. However, the problem is when congestion is predicted, how much should the rates be reduced to? A method in DCTCP [21] is able to gauge the extent of congestion, but it is implemented on the sender side. Proxy solutions implement protocol optimizations in an intermediate network device. A window regulator [22] is used to prevent buffer overflow at the link through modification of the receive window size. This solution works well in wireless networks with small buffers but is no longer suitable, as the buffer size of current devices is often larger than even the maximum receives window size.

2.2. Wi-Fi Measurements and Mathematic Modeling

2.2.1. Wi-Fi Measurements. An autonomic management system which also considers user personalization of wireless networks is proposed in [23]. The architecture of our platform is similar to this platform. Our platform could be regarded as a simplified SDN version of this platform. Network optimizations such as load-balance can be implemented on this platform; also, our platform could be extended to support the load-balance applications. Research work [14, 15, 24–28] focuses on Wi-Fi measurements. The author of PIE [24] captures and records wireless frames and analyzes whether two wireless links are mutually interfering if the packets of two links can be transmitted at the same time. However, this analysis requires that all the APs have the same time clock, and the computation complexity is high. The authors of [25] find that TCP performance degrades significantly in a dense usage scenario, even with 20-30 clients per access point, this discovery is contrary to previous research results [29, 30], and they find that the uplink data transmission causes contradicting results. The author of [13] finds that TCP's performance is related to the Wi-Fi measurement metrics. They define WiTT (Wi-Fi-based TCP throughput) as the wireless experience indicator, then regress WiTT with some of the Wi-Fi measurement metrics. Similarly, the authors of [14, 15, 26] predict page load time, packet latency (RTT for ping command) with some of the Wi-Fi metrics. The authors in [27] used a machine learning method to explore the relationship between AP sensing behaviors and the wireless network states (AP sensing processes are launched by clients to find potential APs, and they are time-consuming and can harm the throughputs). The authors in [31] utilize machine learning algorithms to explore the correlations between network metrics and network loads. The authors in [28] used a machine learning method to identify the ongoing TCP protocols.

2.2.2. Mathematic Modeling. Research work [12, 32–37] focuses on model the throughput performance in wireless networks. The interference caused by rogue APs are modeled in [32], the authors find that hidden terminals cause about 30% MAC layer loss rate increase on average, and the carrier sense interference due to rogue APs causes only 5% access delay to increase at the MAC layer. A two-layer credit-based rate control algorithm for wireless networks is proposed in [12]. A mathematical model of throughput in LTE networks is proposed in [33]. Similarly, the authors in [34] use measurements in the physical layer to model the capacity of the wireless links. The measurements in wireless networks are spatio-temporal data; the authors in [35] define a spatio-temporal distance based on which spatio-temporal neighbors. Then, the spatio-temporal load is defined; at last, the authors use machine learning methods to regress the cell loads. Research works in [27] aim to model the packet delay caused by interference, and the author proposed an algorithm to minimize the interference and decrease the packet latencies. The compound TCP in the Internet of Thing (IoT) is modeled as several subprocesses in [37]; these subprocesses are congestion window, throughput, and packet loss.

3. Motivations

TCP data transmissions over wireless links are full of congestion packet losses. TCP protocols' performance of throughput, delay, and jitter degrades significantly when congestion losses happen. This is why more and more delay and jitter constraint applications abandon TCP protocols. There should be an algorithm to guarantee TCP protocols stable throughput, delay, and jitter; in addition, the algorithm should run on a device that can provide complete and real-time network state information. No other devices know more details about the data transmission over wireless links than the APs, and an SDWN that controls all the APs can provide complete statistics for optimization algorithms. We experiment to show the congestions over wireless links and our ideas.

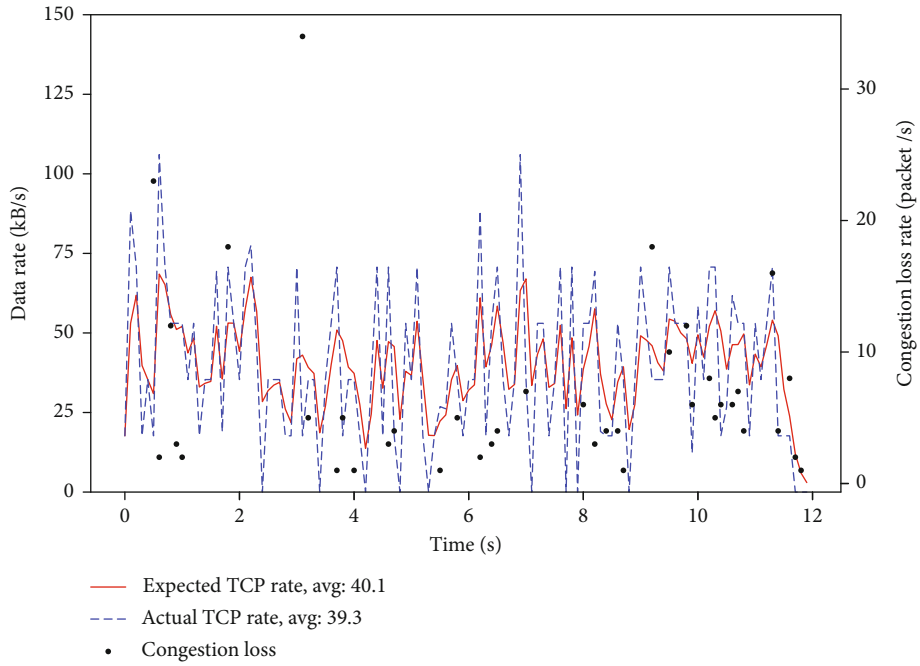

Figure 1: TCP data rates and congestion loss rate over wireless links. The left-side y-label corresponds to the data rate, and the left-side y-label corresponds to the loss rate.

We performed the measurements on a desktop computer, an AP, and a laptop computer. The desktop computer and the laptop were associated with the AP via wired Ethernet and 802.11n protocol, respectively, and the laptop moved with a person. We implemented a TCP client on the desktop computer to send data to the TCP server was running on the laptop. Wireshark was used to monitor the TCP states on the desktop computer and the laptop; tcpdump was used to monitor the TCP states on the AP. The statistics of data rates and congestion losses could be obtained from logs of Wireshark and tcpdump. We draw the actual data rate and the congestion loss rate in Figure 1, and the results show that the TCP data rates fluctuate sharply, especially when congestions happen. The congestions will not only decrease the throughputs but also enlarge the packet delays and the variations of delay time (i.e., the jitter). Delay and jitter performance are fatal for the applications that have stringent corresponding requirements. Is there an implementation to predict and avoid the congestions, so that the data rates in wireless networks are stable as in wired networks?

We propose to develop a such platform and try to achieve this goal. Our idea is to predict future congestion and avoid it by decreasing the data rates in advance to get more stable rates. Our expected rates will be similar to the ones in wired networks, just as the red curve in Figure 1. The logic contained in the red curve in Figure 1 is exactly our idea. Suppose the number of all the time points is N, the set of all the time points is $TP = \{t_1, \cdots, t_N\}$. Let CG denote the time points when congestion happen, and $CG = \{t_{c1}, \cdots t_{cK}\}$, $CG \subset TP$. Let $\text{IN} = \{c1, \cdots, cK\}$ to be the index set. $\forall k \in \text{IN}$, obtain

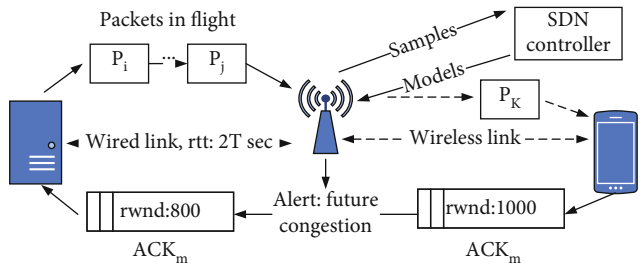

Figure 2: On-AP TCP performance enhancement.

the data rates r_{t_k-2}, r_{t_k-1}, r_{t_k}, and r_{t_k+1} from the blue curve. Then, we perform the rate decreasing operations before each congestion. $r_{t_k-1} = (r_{t_k-2} + r_{t_k-1})/2$, $r_{t_k} = (r_{t_k-1} + r_{t_k})/2$, and $r_{t_k+1} = (r_{t_k+r_{t_k+1}})/2$. $\forall i \in TP, i \notin \text{IN}, (i+1) \notin \text{IN}$ and $(i-1) \notin \text{IN}$Obtain data rate r_i for i from the blue curve. At last, these r_{t_k-1}, r_{t_k}, r_{t_k+1} and r_i will form the red curve, i.e., our expected rates. These "stable" data rates (the red curve) may have lower mean values than the blue ones (the origin data rate) sometimes, but the most important thing is, they will be friendly to the applications that have delay or jitter related constraints.

The processes of our TCP performance optimization are described in Figure 2. Remote TCP senders and the local SDN controller are associated with the APs via Ethernet, and the wireless devices associate with the AP via 802.11 protocols. Network statistics are sampled on the APs and transmitted to the controller. The controller generates and updates forecasting models for the APs, and the APs predict congestions based on these models. As

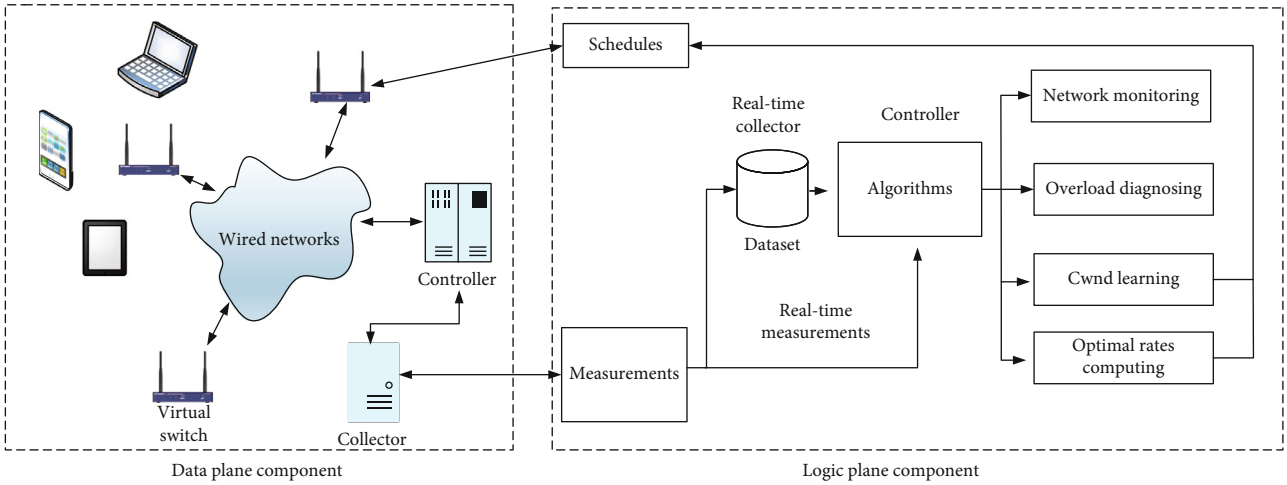

Figure 3: Overview of our SDN-based TCP optimization platform.

shown in the figure, downlink packets such as P_i, P_j, and P_k are transmitted from the remote sender to the mobile device, and the corresponding acknowledgments (ACK) are transmitted back to the sender. Suppose when ACK_m reaches the AP, the forecast model runs on the AP tells there will be congestion 2 T time later. Then, the data rates will be decreased to an empirical value via our data control modules. A rate-to-window transformation module is also implemented in our system, and suppose the output of this module for the shown TCP flow is 800. Then, the rest of the work is to replace the origin size (1000 here) to 800. This replacing process is implemented by our custom OpenFlow actions. The receive window (rwnd) in ACK_m will become 800 after the replacement, and ACK_mwill reach the server in T seconds. The rwnd in ACK_m will limit the send window in TCP stacks of the server (the TCP rate are dominated by cwnd and rwnd), and the rate will be decreased to the empirical value immediately. Then, T seconds later, the data rate on the AP is decreased before the queue is overloaded and congestion is happening. At last, the congestion is avoided.

Two problems are essential for the TCP enhancing processes aforementioned. When and how to act, i.e., the congestion prediction based on cross-layer network statistics, and the optimal window set in ACK packets. We describe our works on congestion prediction and rate control in Section 5, and Sections 6 and 7, respectively.

4. Overview of Our SDN-Based TCP Optimization Platform

The design of our SDWN platform is briefly introduced in this section, with the emphasis on the modules of data sampling, data receiving, preprocessing, and model learning.

The platform is shown in Figure 3. Since the platform is designed in an SDN fashion, it contains a data plane part and a logic plane part. Six function modules are implemented on these planes, and they are data sampling, network monitoring, congestion window (cwnd) learning, optimal rates computing, overload diagnosing, and rate control. Rate control, network monitoring, and data sampling work on the data plane part, and the other modules work on the logic plane part. Custom messages are implemented to transmit data between the data plane and the logic plane. Custom actions are implanted to process the messages, maintain the forecasting models from the controller, and control the data rates. We introduce these modules in the following paragraphs.

4.1. Data Plane Part. The commercial APs (NetGear WNDR 3800, 4300 in our project) are firstly converted into virtual switches by utilizing Open vSwitch [38]. Open vSwitch is an OpenFlow implementation that turns a device into the data plane of OpenFlow protocols; also, actions, custom messages, and modules can be implemented on the Open vSwitch framework to support performance optimization. We also need the support of OpenWRT (https://openwrt.org/), which is a Linux operating system that runs on commercial Aps, and Open vSwitch runs on OpenWRT. Most of the statistics listed in Table 1 run in OpenWRT kernel, but the packet statistics are sampled in the datapath of Open vSwitch.

4.2. Data Sampling Module. The network statistics are sampled in the data sampling module. We utilize supervised learning to get the forecasting model, with the goal of analysing the congestion losses, and cross-layer network metrics are utilized as the inputs of the prediction algorithms.

Congestion losses cannot be obtained directly from the sampled packet statistics. There are two kinds of packet losses on the APs, the congestion losses in queue processes and the noncongestion ones in the driver (can be obtained in Mac80211). The number of packet losses minus the number of noncongestion ones is the number of congestion ones. The number of packet losses can be obtained by checking the sequence and payload of each packet. The number of noncongestion losses can be obtained by directly counting the data samples from Mac80211.

Table 1: Categories, sources, and sample rates of sampled metrics.

Module	Location	Layer	Rate
Packet	Open vSwitch, kernel	Transport and network	Each pkt
Link	Mac80211, kernel	Mac layer	250 Hz
Channel	Mac80211, kernel	Mac layer	250 Hz
Queue	Sch_generic.c, kernel	Kernel	250 Hz
Beacon	Mac80211, kernel	Mac layer	10 Hz
Drops	Mac80211, codel.h	Mac layer	Each drop

Table 2: The collected metrics and their Kendall and IG index with congestion packet loss ratios.

Radio factor	Kendall index	IG index
Backlog of queue 3	0.3307	0.1405
Channel utilization	0.5413	0.1694
All dropped in queues	0.3109	0.1838
All dropped in queue 3	0.3057	0.182
Received bytes in queue 3	0.4082	0.1504
Received packets in queue 5	0.0138	0.1403
Requeues in queue 3	0.3128	0.0869
All the transmission retries over the links	-0.02	0.0157
All the transmission bytes over the links	0.2898	0.0091
All the received bytes over the links	0.2557	0.0058
All the transmission failed over the links	-0.007	0.0032
Noise	-0.1386	0.003
Data rate counted in Open vSwitch datapath	0.8	0

The related factors are the cross-layer metric measurements before congestions happen. The data are sampled from the transport layer, network layer, and MAC layer. Therefore, we call them cross-layer metrics. Most of them are sampled at 250 Hz; the sample rate for beacon statistics is 10 Hz. Statistics of all the arrival packets and all the dropped packets are sampled. The details of the sampling are listed in Table 1, and most of the cross-layer network metrics sampled and utilized in this article are listed in Table 2.

The IG index and Kendall index are also listed in Table 2. IG is a general index to quantify the predictable relationships, and the Kendall index is a general index to quantify the monotonic relationships. The larger the absolute value of the IG index, the stronger the predictable relationship. Moreover, a large absolute value of the Kendall index also means there exists a remarkable correlation between two vectors.

The sampled network metrics are listed in Table 2. We notice that some metrics belong to queue 3, and there is one that belongs to queue 5. This is because the OpenWrt maintains five queues on our APs (NetGear WNDR 3800/4300). Nearly all the data are transmitted in queue 3, but when data bursts occur, some of them are transmitted in queue 5. The queue statistics are sampled because the congestions happen in queue algorithms, and they have significant correlations with congestions, as the indexes show in Table 2. Because researches in [15] show that the channel-related statistics (channel utility and noise in Table 2) are directly related to packet latencies, we sample them to evaluate their correlations with congestions. Links are units of wireless communications, so their statistics are sampled. We also sample the statistics of all the data packets that arriving Open vSwitch datapath, and the overall data rates, the number of all the packet losses. In addition, the receive windows are obtained from these statistics. The correlation indexes in Table 2 will be discussed in subsequent paragraphs.

The sampling modules sample the real-time statistics, remove their redundancies, and then transmit them to the data receiving module that runs on the controller. The receiver preprocesses the real-time statistics into training data items for learning algorithms, and it also utilizes application programming interfaces (API) to provide inputs to the optimization algorithms.

4.3. Custom Messages and Actions. Custom messages are implemented to transmit data samples and algorithm outputs between the APs and the controller. There are three kinds of algorithm outputs that need to be transmitted from the controller to APs in our platform: the forecasting models generated by the machine learning algorithms, the congestion windows of the window regression algorithms, and the receive windows of the aforementioned state machines. We encode these messages into predefined formats and transmit them to corresponding APs. Custom actions are implemented to decode and maintain them on the Open vSwitch. The network statistics of other APs are also needed to be transmitted to each AP. At last, as most of the statistics are sampled in kernel space, but most of the computations happen in userspace, we implement some modules with Netlink communication sockets to transmit statistics between kernel space and userspace.

4.4. Logic Plane Part. The data plane runs on the controller. RYU [39] is utilized to turn an HP-Z420 work station into an SDN controller. Modules of the congestion window learning, virtual rwnd computing, overload diagnosing work on this plane. We will introduce these functional modules in the following paragraphs.

4.4.1. Data Receiving and Processing. Real-time network statistics are needed on the data plane to compute the forecasting model, and they are also required for real-time congestion window regressions. Furthermore, APIs are

implemented to provide necessary inputs for optimization algorithms. The data receiving and processing module does these works. The challenge for this module is the real-time data amount. The raw data samples should be counted and computed to yield meaningful data items. Much of the developing works were focused on this part. Nearly, all the codes are written in C language, and the APIs are programmed with both C and Python to guarantee efficiency and flexibility.

We introduce some details about the receive windows here. The advertisement windows contained in ACK packets are not the real window for the TCP receivers. Both sides of a TCP communication will announce their window scaling factor to each other when TCP is creating. The real window equals the window in ACK multiply 2^x, where x is the window scaling factor, and 2^x is 2 to the power of x. Window scaling factors are contained in the syn packets only. They are created when TCP flows are created and destroyed when TCP flows are ended. To get these windows in real-time, we implement state machines in the data receiving module, and these state machines monitor the dynamics of TCP flows. The obtained receive windows are used in the rate control module.

4.4.2. Optimal Rates Computing. This module is utilized to allocate data rates among the data flows that run on the AP, which will be explained in detail later.

4.4.3. Congestion Window Learning. As introduced in the previous section, congestion windows are necessary for rate-to-window regression. We design a window regression on the controller, the inputs are the sampled network statistics, and the outputs are the congestion windows of the running TCP flows. Details of the regression will be described in subsequent sections.

5. Congestion Prediction

We argue that the model should be tailored for the specific scenario, rather than building a single unified congestion loss model. The reasons come from the following four aspects. Firstly, more than nine factors can affect packet congestions as described in Figure 4. Furthermore, the relationships between congestion losses and each of the factors are not linear nor monotonic. Secondly, some of the related factors are mutually correlated. For example, channel busy utilization, receiving utilization, and transmitting utilization are mutually related. Thirdly, it is a challenge to model the wireless channels that dominate wireless communication processes, even when several channel models are proposed. Fourthly, it is also difficult to model the queue algorithms, MAC protocols, and the wireless drivers that affect packet congestions. In view of this, we choose learning-based methods in this article. We sample all suspect metrics, learn their correlations with congestion packet loss, mark the dataset, and build and evaluate a corresponding supervised learning model.

5.1. Large-Scale Measurement Study. We deployed 12 APs on which the sampling modules ran in the library of H university. We provided Internet access services to encourage potential users to utilize our free APs. We also replaced the old APs of our laboratory center with four of our APs. The sampling processes lasted for three months, and we collected about 1060.9GB (600.9 GB TCP headers, and 460 GB cross-layer statistics) raw data. Our dataset contains hundreds of millions of records; it even contains statistics of more than 3300 devices that had ever accessed our APs. This dataset is unprecedented, and no such datasets were collected before, based on our surveys. We will deploy more APs in the campus to obtain more representative data.

We get several high-level findings from the dataset. (1) Public wireless networks are crowded and ill configured. Our APs in the laboratory can detect more than 200 active APs. The data sampled from the library shows that more than 1200 APs (includes Wi-Fi Hotpoints) appear in a 30-days' dataset. The statistics show that 16.1%, 25.7%, 48.2%, and 10% of the APs utilize channels 1, 6, 11, and other channels, respectively. (2) TCP congestion loss ratio is high and reaches 6.7677% for our network as shown in Figure 5; (3) RTT is stable in the wired links. (4) There are far more congestion loss packets than the noncongestion ones. (5) Congestion loss ratio has a long-tail distribution.

We compute the congestion loss ratios and draw their cumulative distribution function (CDF) in Figure 5. The median of the ratios is 3.5%. The corresponding ratio for the 50th percentile, 90th percentile, and 99th percentile is 3.57%, 33%, and 77%, respectively. We can conclude that this congestion loss ratio has a typical long-tail distribution, which indicates that some users or applications are suffering from severe congestions.

5.2. Measurement Study. We observe that the metrics have strong or weak correlations with congestion loss ratios. Furthermore, the significant IG and Kendall indexes between congestion loss ratios and the related factors indicate that congestion loss forecasting models built on these metrics will be potential excellent algorithms.

5.2.1. Observations. We observe that several cross-layer network metrics, especially the queue-related factors, have strong qualitative relationships with the congestion loss ratios. We choose nine of them and draw the correlations in Figure 4. The black lines in the figures are the distributions of the related cross-layer network metrics. The red dotted curve in each subfigure is the congestion loss ratio curve for the values of the corresponding metric, it is not a distribution. We explain these subfigures here. Subfigure (a) shows that about 75% of the arrival rates in queue 3 are slightly higher than zero, the congestion loss ratios are also high for these low arrival rates. People may think that congestions only happen when the queue arrival rates are high, as shown in the right part of the subfigure (a). However, low data rates may cause congestions when queues are nearly saturated; we notice that more than 80% of the data rates are near zero in subfigure (b); this is because queue 5 is empty unless data bursts happen. Subfigure (c) is similar to (a) and (b); these big zigzag curves indicate that the relationships between data rates and congestions are complex when queue buffers exist; subfigures (d), (e), and (f) are easier to be understood. When the values of backlogs in queue 3, values of packet dropping rate in queue 3, or the retransmission rate are high, there is

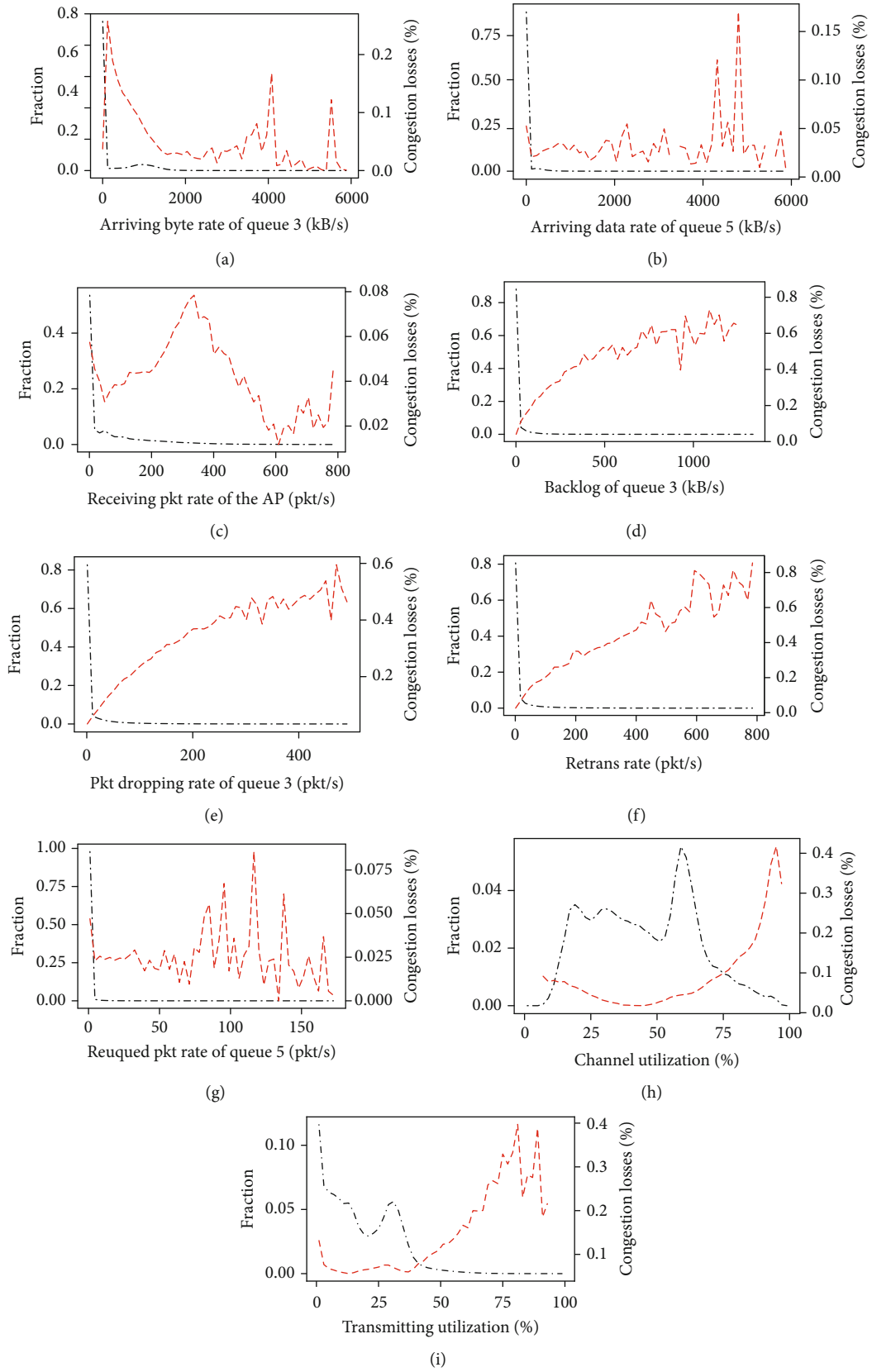

Figure 4: Distribution of related factors, and their relationships with Wi-Fi congestion loss ratios.

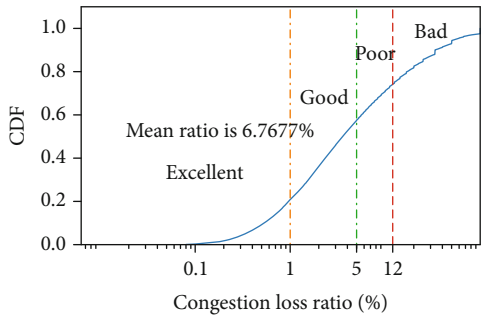

Figure 5: Wi-Fi congestion loss ratio distribution.

no reason that the congestion loss ratios will be low. Furthermore, when the retransmission rate is high, the congestion loss ratio is high. The curve in the subfigure (g) is complicated. The reasons may be data items for the nonzero distributions are not enough. As the requeue events in queue 5 rarely happen. Subfigures (h) and (i) have similar curves. The curves indicate congestion loss ratios are low when the channels are under suitable utilizations. High channel utilization indicates that the channel is busy. Low utilization indicates that few data are transmitting, but the queue backlog values are unknown. High congestion loss ratios are common in these two scenarios.

5.2.2. Qualitative Correlations. As explained in previous sections, the IG index and Kendall index can be utilized to quantify the correlations between object metrics and the related metrics [14, 32]. The IG indexes and Kendall indexes between congestion loss ratios and the related factors are listed in Table 2. We can learn from this table that backlog of queue 3, channel utilization, drops in queues, and arriving rates have strong qualitative relationships with congestion loss ratios. They are not surprising results; these metrics are shown to have strong correlations with congestions in Figure 4.

5.3. Classification. To assist in interpreting the losses in terms of their impact on well-known applications, the authors of [40] categorize the losses into six classes. Labels "bad," "very poor," and "poor" are utilized to describe the negative congestion states, and "acceptable," "good," and "excellent" are utilized to describe the positive congestion states. Only two labels are needed in our current algorithms to control the data rates: "bad" and "good." However, we reclassify these six classes into four classes for our future works, two for positive states, and the other two for negative states; details are listed in Table 3. Finally, the training data will be labelled with these four labels, and utilized in supervised learning algorithms.

Supervised learning algorithms are powerful methods to generate forecasting models. In this section, we explore the congestion prediction performance of the supervised learning algorithms.

We choose several typical supervised learning methods to model the congestion loss ratio. They are decision tree (DT), random forest (RF), support vector machine (SVM), k-nearest neighbor (KNN), and neural network (NN). Their performance is evaluated and then represented with the

Table 3: Classes of Wi-Fi congestion loss ratios [40].

Wi-Fi loss class	Loss range	Our classes
Excellent	<0.1%	Good
Good	0.1% and <1%	Good
Acceptable	1% and <2.5%	Acceptable
Poor	2.5% and <5%	Acceptable
Very poor	5% and <12%	Poor
Bad	12%	Bad

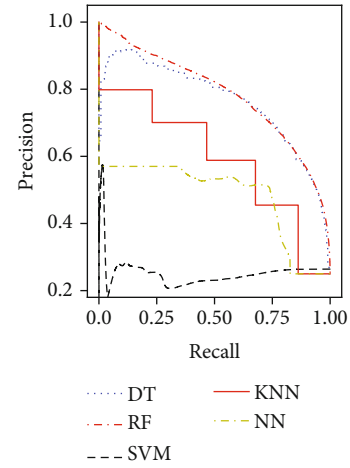

Figure 6: The precision-recall curve for different classification algorithms.

Table 4: Performance of the 4-class classifiers, values in each 4-tuple are the accuracies for the 4 classes.

Items	Decision tree	Random forest
Precision	(0.680, 0.431, 0.485, 0.612)	(0.680, 0.504, 0.627, 0.795)
Recall	(0.677, 0.432, 0.485, 0.626)	(0.829, 0.446, 0.489, 0.594)
F-score	(0.679, 0.432, 0.485, 0.622)	(0.747, 0.423, 0.549, 0.680)

precision-recall curve (PR-curve) [41]. The curves are shown in Figure 6, and they indicate that decision tree and random forest methods outperform the others. We choose these two kinds of algorithms and learn their performance on our dataset. Their performance listed in Table 4 indicate that the accuracies are high for "good" and "bad" classes, and low for "acceptable" and "poor" ones. This may imply that network metrics show some cluster tendencies, that is, network metrics trend to positive when the congestion loss ratio is low, and trend to negative when the congestion loss ratio is high. The lowest accuracy is 0.423, but it is acceptable based on the results in [14]. The authors of [14] claim that the accuracy of 0.42 for their 4-classes classifier is reasonable.

The decision tree model and the random forest model perform better than other models, and their performance is nearly the same. We claim that the decision tree model is more suitable for network management, because it is easier to be understood and is facilitate to diagnose the

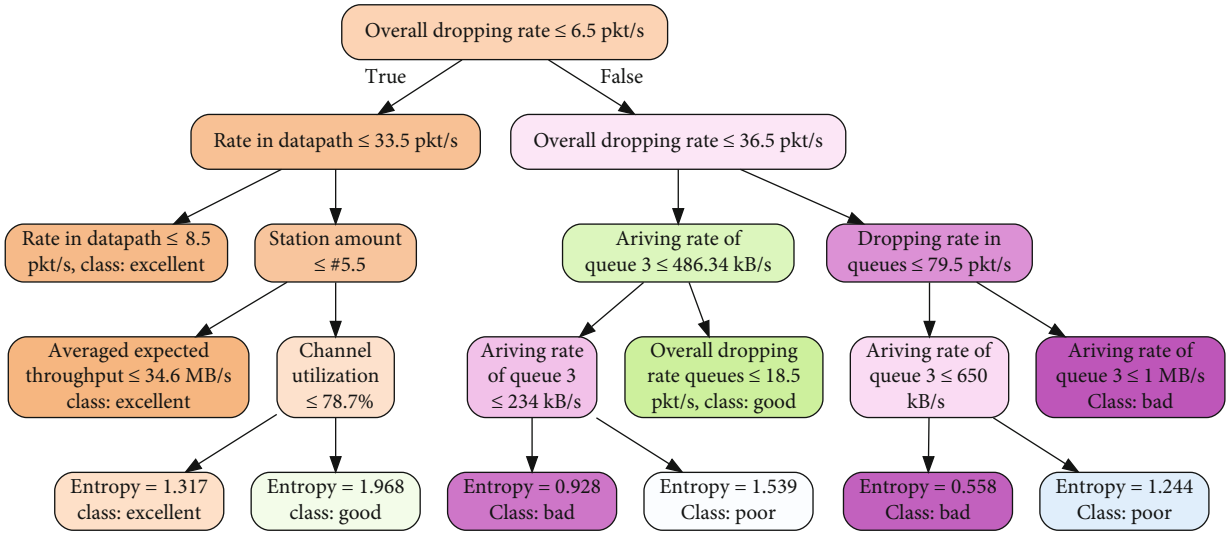

Figure 7: This 5-depth decision tree is an example of our congestion forecasting models. Some of the metrics are not shown because of the shallow depth.

faults. Therefore, we choose it as the forecasting model in this article. We draw a 5-depth decision tree example in Figure 7. Not all the metrics are shown in this figure because the max_depth parameter is set to be 5. The scales of the practical decision trees that run on our APs are larger. Essential metrics that have more effect on congestion losses will be put near the root.

The controller generates a decision tree every second by using the latest 10000 training items. The new trees are then transmitted to the corresponding APs to replace the older trees. A deep neural network (DNN)-based framework is proposed in [42]; decision trees are generated by utilizing a deep neural network method. We are also developing our DNN-based decision tree generation framework, and this framework will be utilized in the congestion loss ratio forecasting in the future.

5.4. Loads of Tree-Based Congestion Prediction. Loads of computation and communication are vital for a system although the performance of commercial APs is powerful. We introduce these two kinds of loads of our congestion prediction algorithm that runs on APs, respectively.

The first one to introduce is the communication load. We sample a large amount of high-precision and high-frequency network metrics, as described in Table 1. For each AP, suppose the number of associated devices is z; and it has y neighbor APs. r is the packet arriving rate, and η is the packet loss rate. The (item, length) pairs of these metrics are (link, 64B), (channel, 64B), (beacon, 32B), (drop, 48B), (queue, 48B), (packet, 64B). The sampling rates are listed in Table 1. Therefore, the data rate of sampling data r_s computed as the following:

$$r_s = \frac{250 * (64 * z + 64 + 48 * x) + 32 * 10 * y + 48 * \eta * r + r * 64}{1024} = 15.625\,z + 15.625 + 11.718x + 0.316y + 0.0468r\,\eta + 0.0625r. \quad (1)$$

Suppose $z = 10$, $x = 1$, $y = 20$, $r = 1500$, and $\eta = 0.1$, then $r_s = 283.5$ kbyte/s. The practical r_s is much lower than 283.5 kB/s, as the sampled data are full of redundancies, and we implement modules to reduce these redundancies. Different elements of a sample data structure have different varying frequencies. For example, the eth addresses and IP addresses of a link will remain the same most of the time, while its signal strength and noise vary sharply. Furthermore, the channel coherent time is longer than 4 ms. Coherence time is a statistical measure of the time duration over which the channel impulse response is essentially invariant. Researches in [43] indicate the coherence time for 802.11 WLAN is approximately 25.3885 ms when the velocity of the client is 1 m/s. Thus, there are no many data samples that really needed to be transmitted, and we only need to transmit the data items that have different values with the ones in the last transmitted sample.

The second to introduce is the computation load which comes from tree-related operations: tree creation, network states acquisitions, and tree searching. First of all, the load of the decision tree creation is affordable. The controller encodes each node of the generated decision tree to be a 5-tuple: (father_id, itself_id, name, value, class_id), where value is the decision value compared in the tree-searching algorithm, and the class id is which congestion class it belongs. When an AP receives this encoded text sequence, it decodes the nodes and creates a corresponding tree. The loads for this sequence decoding and tree creation processes are low because the scale of the decision tree is finite. Secondly, the load of network states acquisitions is also affordable. To create and maintain a decision tree in an AP's kernel space is tricky, every potential bug could halt the AP. Therefore, we create and maintain the tree in userspace. However, most of the network statistics should be obtained from our custom APIs that run in the kernel space, as shown in Table 1. Therefore, we implement communication applications to transmit messages between kernel space and userspace with Netlink socket, and some basic computations are utilized to decrease

the data amount that needed to be transmitted as the float computations are not supported in kernel space. In addition, statistics of other APs are transmitted from the controller to local AP, but the data amount is limit. Thirdly, tree searching is not a time-consuming operation. Each of the congestion prediction processes is a tree searching operation, and the class id of the found leaf node is the prediction result. The time complexity of our Binary Search algorithm is $O(\log_2 n)$, where n is the number of tree nodes, and the tree depth is limit, so it could not cause high latency.

Overall, these processes are complicated but not time-consuming as they have not much computation load. However, the communication loads will be a problem in large scale wireless networks, and we will focus our research works on communication loads in our future works.

6. Utility-Based Window Modelling

As described in the previous section, a decision tree observes network states and yields the congestion state prediction. The prediction decides whether we should launch a new rate control. When the predicted state is "poor" or "bad," the rates of TCP flows should be recomputed and controlled in a proportional fair manner. We model this rate allocation as a utility maximization problem to obtain optimal solutions.

Suppose the set of all the APs is AP_k, i is a flow, and the set of all the flows is $\mathscr{L}_k$, the data rate for flow i is x_i. For $i \in \mathscr{L}_k$, the congestion packet dropping model is:

$$l_i = \begin{cases} 1 - \dfrac{C(t)}{\sum_{i\in\mathscr{L}_k} x_i} & \text{if } \sum_{i\in\mathscr{L}_k} x_i > C(t), \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where C(t) is the capacity of links associated with the AP. C(t) is time-varying in wireless networks, and we obtain it from historical data. C(t) is set to the data rate of its nearest noncongested time slot in this article.

We model a utility for each data flow; this utility is based on its data rate x_i and congestion packet dropping probability l_i. The utility for data flow i is as the following:

$$u(x_i, l_i) = c_i \log x_i - x_i\left(\frac{1}{1-l_i} - 1\right), \tag{3}$$

where c_i is the weight of flow i, and $\sum\sqrt{c_i} = \sqrt{C(t)}$. The network utility for AP_k is $U_k = \sum_{i\in\mathscr{L}_k} u(x_i, l_i)$. The TCP optimization problem turns into an optimization maximization problem.

Taking the first derivative of the utility function, we obtain the followings:

$$\frac{du}{dx_i} = \frac{c_i}{x_i} + 1 - \frac{\sum_{j\in\mathscr{L}_k} x_j}{C(t)} - \frac{x_i}{C(t)} = \frac{c_i}{x_i} - \frac{x_i}{C(t)} \tag{4}$$

Which, by utilizing the condition $\sum_{j\in\mathscr{L}_k} x_j = C(t)$, is:

$$x_i = \sqrt{C(t)c_i} \tag{5}$$

The obtained data rates should be mapped to the receive windows and embedded them in uplink ACKs. Two things should be known in advance for this process: current cwnds and historical statistics of the cwnd-rate pairs. The historical mapping statistics are available on the controller, but it is a challenge to obtain cwnds on intermediate wireless devices. We introduce how we solve this congestion window problem in the following section.

7. Congestion Window Regression

As introduced in the sections above, the data rates should be transformed into receive windows to control the TCP sending rates. As the sending rate for a TCP flow is dominated by the small one of its congestion window and its received receive window, the congestion window is necessary when computing a new receive window on an intermediate device.

Some research works have considered the regression of congestion windows on intermediate nodes [44–46]. As the TCP senders control the data rates based on TCP state machines maintained in TCP stacks, shadow-state machines are implemented on intermediate nodes to follow the dynamics of the real state machines on the sender sides [45, 46]. Recently, research works in [44] indicate that these state-machine-based methods have performance and compatibility problems. Furthermore, the authors of [44] utilize the number of unacknowledged packets (UNA) to regress the congestion windows on intermediate nodes. Their evaluations show that UNA can guarantee a more than 0.9 forecasting accuracy in wired networks. However, our experiments show that typical machine learning algorithms can only achieve a forecasting accuracy of 0.2911 when using UNA only.

As UNA only is not enough for congestion window regressions in wireless networks, there should be several other network metrics that could enhance the forecasting accuracies. We try to explore these metrics and their correlations with congestion windows. We sample abundant data for regression researches, and the sampling processes are introduced in the following paragraph.

As the object metric for learning, the congestion window data items are sampled on the TCP sender sides. We develop a TCP statistics sniffer to monitor the TCP real-time dynamics. The outputs of this sniffer are similar to the outputs of the "ss" command in Linux systems; they contain system time, congestion window, round trip time, and TCP ports. This sniffer utilizes the Netlink socket communication to obtain TCP dynamics from the kernel space. The related network metrics that used to regress congestion windows are sampled on intermediate nodes. These related metrics come from our data sampling modules. We utilize seaborn.heatmap (https://seaborn.pydata.org/generated/seaborn.heatmap.html) to learn the correlation matrix for all the metrics, and we refer this matrix to decide which metric should be chosen. Furthermore, we learn each metric's impaction on the regression and decide whether it should be kept. At last, the chosen metrics are listed in Table 5.

There are some details about time synchronization. Because the window values utilized in regressions are sampled

Table 5: The chosen metrics, and the correlations computed with seaborn.heatmap.

Metrics	Instructions	Correlations
UNA	Computed from the sampled packet statistics	0.52
Data rate for all the links	Rates from link statistics	0.5
Channel transmit utility	Time for transmitting in last 1000 ms.	0.49
Signal strength	Signal strength from channel statistics	0.27
Channel busy utility	Time channel is busy in last 1000 ms.	0.23
Retransmission ratio	Packet loss statistics	0.2
Noise	Noise from channel statistics	-0.13

Table 6: The regression algorithms and their performance when using UNA only and using our metrics.

Algorithms	Only UNA	Our metrics
GradientBoostingRegressor	0.2911	0.5362
RandomForestRegressor	0.2902	0.5135
LinearRegression	0.2713	0.4354
VotingRegressor	0.2892	0.5246
AdaBoostRegressor	0.2715	0.3900
BaggingRegressor	0.2902	0.4685
ExtraTreesRegressor	0.2902	0.5045
XGBRegressor	0.2911	0.5374

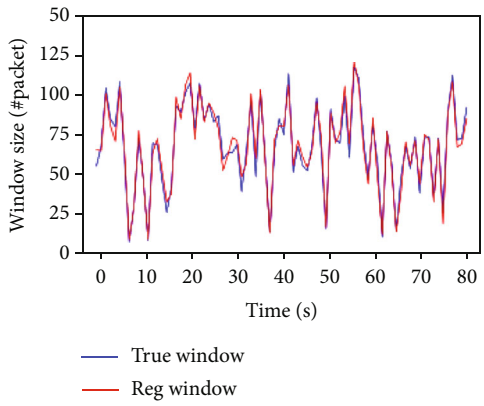

Figure 8: The regression results, the blue ones are the actual windows, and the red ones are the windows obtained with the XGBregressor.

at the sender sides, and the other metrics are sampled on intermediate nodes. Therefore, we set a common time server with the network time protocol for time synchronization.

Several typical regression algorithms are utilized on our data set of the chosen metrics. The results are listed in Table 6. The results show that the regression accuracy is only 23.12% when merely using UNA, and the results increase to 55% when utilizing our chosen metrics in Table 5. Then, we utilize the XGBregressor for window regression on our chosen metrics and draw the actual windows and the regress windows in Figure 8 from which we could observe that the two curves fit well. The data and codes are available in GitHub (https://github.com/lingerlilac/cwnd).

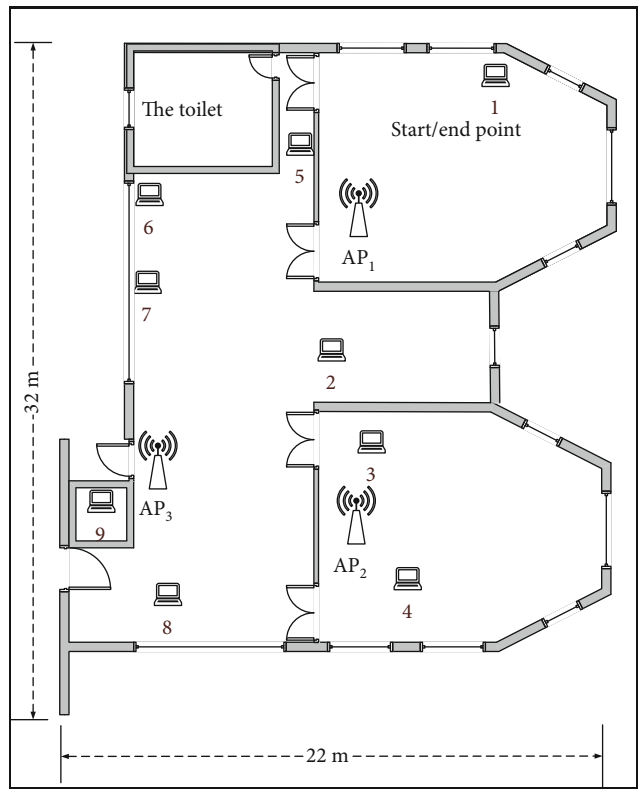

Figure 9: Deployment of APs and clients in the experiments.

8. Evaluation

We conduct evaluations to characterize the throughput, delay, jitter, fairness, and retransmission performance of our method. Our method is an on-AP learning method, so we name it as OAL (on-AP learning). OAL is not a protocol, and it is a module that runs on Cubic protocol; it helps to enhance the performance of Cubic. We compare the OAL method with the BBR protocol and Vivace [4] method. The evaluating objects are throughput, delay, jitter, retransmission ratios, and the corresponding fairness.

8.1. Experimental Setup. We carry out the evaluations on our real-world testbed. The deployment of evaluations is shown in Figure 9. Our system contains a server (runs the controller), three APs, and nine wireless stations, and this

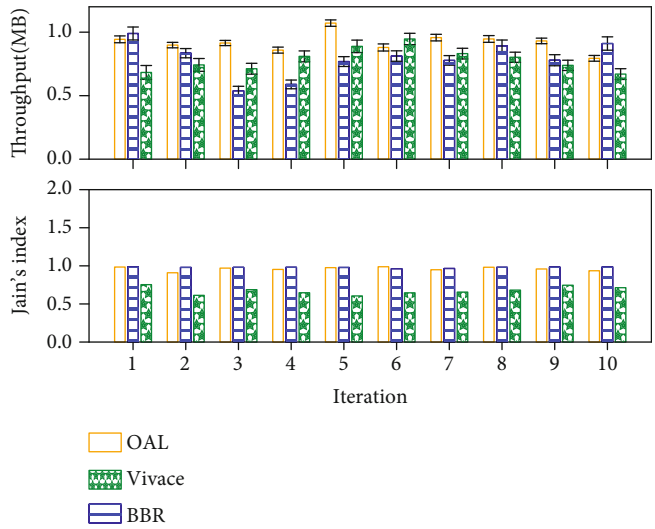

Figure 10: Network-wide throughput performance for the algorithms.

experiment scale is the same as the one in [47]. The server is an HP Z420 work station that runs the Debian system. The Ryu controller runs on the server. The APs are Netgear 4300 routers that run OpenWRT 15.05.01(openwrt-lede) and Open vSwitch 2.8.2. Nine Laptops act as the wireless stations, and they run the Debian system. All the evaluations were running on the 2.4 GHz ISM band.

8.1.1. The Evaluation Processes. During the experiment, the controller also acts as iperf3 servers, we open nine TCP flows on it, and each flow corresponds to a client. For each of the evaluated methods, we run experiments for ten iterations, and each iteration lasts 600 seconds. The outputs of iperf3 are stored in both server sides and client sides to calculate throughput performance. We obtain the delay and jitter performance from the outputs of the sniffer introduced in the previous section.

8.2. Experimental Results. In this paper, we utilize the Jain fair index to quantify the fairness of throughput, delay, and jitter performance. The Jain fair index is denoted by the following equation:

$$Jain = \frac{\left(\sum_{k\in A} J_k\right)^2}{m\sum_{k\in A} J_k^2}, \tag{6}$$

where J_k represents the metric of AP k, A is the set of APs, and m is the amount of APs.

We analyze the log files of iperf3 to obtain the throughput and the corresponding fairness performance for each iteration. The results are shown in Figure 10. For each iteration, the throughput of an algorithm is defined as the mean throughput of the three APs; the throughput fairness of an algorithm is defined as the Jain's index of the three APs' throughputs. These definitions are also applicable to delay and jitter. Figure 10 indicates that OAL achieves the best throughputs in seven of the ten iterations, and the average throughput of OAL is 17.5% more than BBR and 16.5% than Vivace. Furthermore, its fairness throughput performance is acceptable. The throughput enhancements must because OAL achieves better congestion performance, as shown in Figure 11, which shows that congestion losses of OAL are especially excellent in iterations 3, 4, and 5, and it achieves much more throughputs in these iterations.

We get the delay performance from the outputs of the TCP dynamic sniffer, and the variation of delay time is the jitter. The delay, jitter, and the corresponding fairness over the ten iterations are drawn in Figures 12 and 13 , respectively. The average delay of OAL is 30% lower than the value of BBR, and 26% lower than the value of Vivace, and the delay and jitter performance is fairer than the others. The average jitter of OAL is 2% lower than the value of Vivace, and 16% lower than the value of BBR. The performance enhancements on delay and jitter agree with our research motivations, and they are the evidence that OAL works. Recall that OAL is designed to enhance the delay and jitter performance by avoiding unnecessary congestions. The congestion performance shown in Figure 11 is another evidence. The congestion loss ratio is computed with the sampled packet statistics (include the packet loss statistics) as described in Table 1. The congestion loss ratio of OAL is 12.6% less than Vivace, and 25.2% than BBR. It is shown in Figure 11 that OAL achieves better congestion performance in eight of the ten iterations. It should achieve the best congestion performance in all the ten iterations, and we will improve our system in our future works.

9. Discussion

We choose TCP BBR as it will be the default TCP protocol for Linux since kernel 4.9. However, surprisingly, we find in our evaluation processes that its delay and throughput performance is sometimes weaker than TCP Cubic or TCP Reno. This may because that BBR exhibits

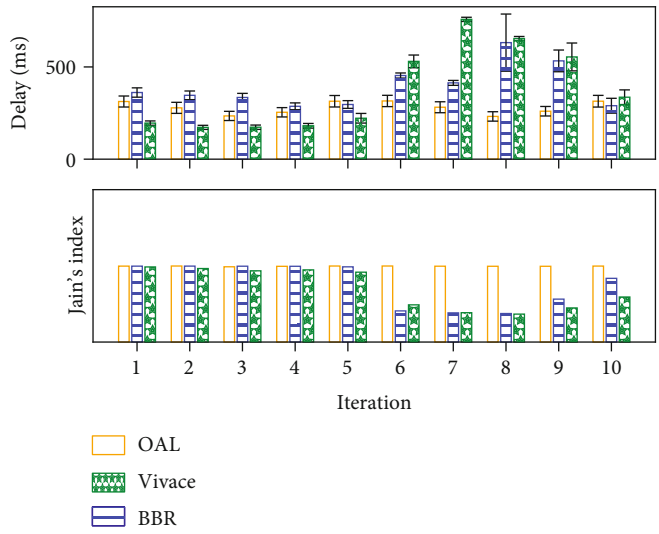

Figure 11: Network-wide congestion packet loss ratio for the algorithms.

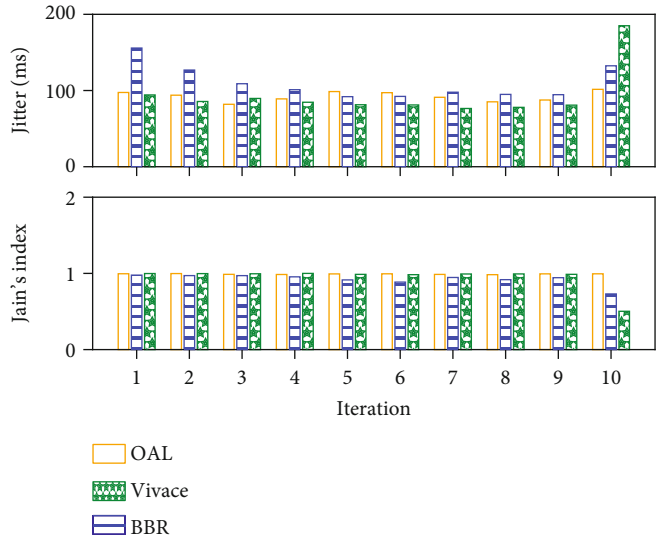

Figure 12: Network-wide delay performance for the algorithms.

high rate variance and high packet loss rate upon convergence, and the convergences of wireless links are lengthy processes. This may be the reason for BBR's poor congestion performance in iteration 3, 4, and 5, and the congestions dominate its transmission performance. Vivace tries to achieve the Nash equilibrium among data flows; it is a learning-based method that estimates the capacity of the link by monitoring delay and rate. However, the relationships between congestion and these two factors are complex as shown in our research works. Also, the congestion avoidance algorithms are designed as cooperative games, i.e., they decrease their data rates whenever congestions happen. Furthermore, research in [16] points out that the Nash equilibrium is inefficient unless the endpoints are restricted to run TCP Reno over a drop-tail buffer, in which case the equilibrium is unfair but not inefficient. With this interpretation, it is not difficult to explain why Vivace performs ordinary during the evaluations with this discovery.

Our platform is designed and implemented to forecast and avoid congestions by getting current network states instantly and acting immediately. We aim to forecast and avoid all the congestions. However, the results in Figure 11 show that some congestions happened. This is not difficult to explain, and our optimization processes include a congestion prediction process and a rate control process. We know that the congestion window regressions play an essential role in rate control processes, but the regression accuracies are still weak. Furthermore, congestion prediction accuracies are also not good. Therefore, OAL still has the potential to be a good algorithm, and more future works will be focused on it.

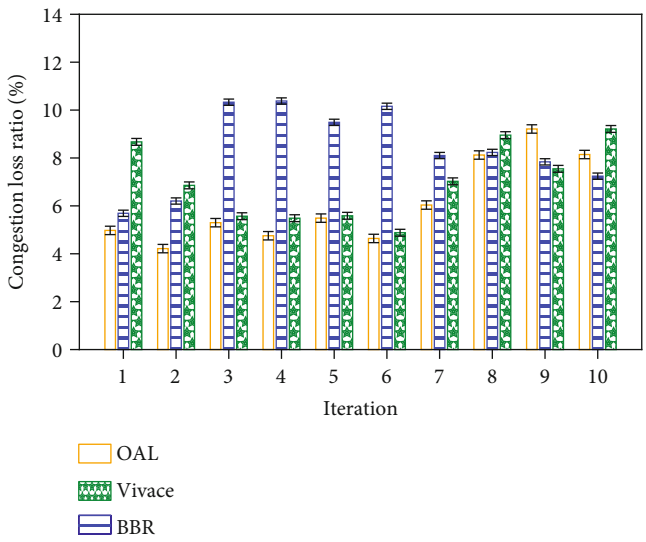

Figure 13: Network-wide jitter performance for the algorithms.

10. Conclusion

We implement an SDWN platform, and several sampling modules to sample instant network metric measurements. The obtained dataset can comprehensively reveal the qualitative relationships between congestion losses and the cross-layer network metrics. Especially the queue-related metrics that are not used in previous researches, and they have significant correlations with congestions. Our platform has several advantages. Firstly, it is prompt. It is implemented on the APs and can get the instant network states. Moreover, the prediction model (decision tree in our project) is also implemented on the APs, so the network states can instantly trigger its actions. Secondly, it is comprehensive. The data samples (training set) come from the transport layer, the network layer, and the MAC layer. Thirdly, it is scalable. We implement our methods in SDWN, which has excellent compatibility. These three advantages guarantee a better performance for our platform.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

There are no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61370214 and Grant 61300210.

References

[1] Index, "Cisco visual networking index: global mobile data traffic forecast update, 2017–2022," Cisco white paper, 2019.

[2] M. Polese, F. Chiariotti, E. Bonetto, F. Rigotto, A. Zanella, and M. Zorzi, "A Survey on Recent Advances in Transport Layer Protocols," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3584–3608, 2019.

[3] W. Sun, L. Xu, S. Elbaum, and D. Zhao, "Model-Agnostic and Efficient Exploration of Numerical State Space of Real-World TCP Congestion Control Implementations," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pp. 719–734, USENIX Association, Boston, MA, 2019.

[4] M. Dong, T. Meng, D. Zarchy et al., "PCC Vivace: Online-Learning Congestion Control," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pp. 343–356, USENIX Association, Renton, WA, 2018.

[5] V. Arun and H. Balakrishnan, "Copa: Practical Delay-Based Congestion Control for the Internet," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pp. 329–342, USENIX Association, Renton, WA, 2018.

[6] F. Y. Yan, J. Ma, G. D. Hill et al., "Pantheon: the training ground for Internet congestion-control research," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pp. 731–743, USENIX Association, Boston, MA, 2018.

[7] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[8] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting Congestion Control for Consistent High Performance," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pp. 395–408, USENIX Association, Santa Clara, CA, 2015.

[9] M. Zhang, M. Mezzavilla, J. Zhu, S. Rangan, and S. Panwar, "TCP Dynamics over Mmwave Links," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–6, Sapporo, Japan, 2017.

[10] M. Polese, M. Mezzavilla, M. Zhang et al., "milliProxy: A TCP proxy architecture for 5G mmWave cellular systems," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pp. 951–957, Pacific Grove, CA, USA, 2017.

[11] M. Zhang, M. Polese, M. Mezzavilla et al., "Will TCP Work in mmWave 5G Cellular Networks?," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 65–71, 2019.

[12] F. M. F. Wong, C. Joe-Wong, S. Ha, Z. Liu, and M. Chiang, "Mind Your Own Bandwidth: An Edge Solution to Peak-hour Broadband Congestion," 2013.

[13] A. Patro, S. Govindan, and S. Banerjee, "Observing home wireless experience through WiFi APs," in *Proceedings of the 19th annual international conference on Mobile computing & networking - MobiCom '13*, pp. 339–350, Miami, Florida, USA, 2013/September, 2013.

[14] K. Sui, M. Zhou, D. Liu et al., "Characterizing and Improving WiFi Latency in Large-Scale Operational Networks," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '16*, pp. 347–360, Singapore, 2016.

[15] C. Pei, Y. Zhao, G. Chen et al., "WiFi Can Be the Weakest Link of Round Trip Network Latency in the Wild," in *IEEE INFOCOM 2016 - the 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, San Francisco, CA, USA, 2016.

[16] K. Winstein and H. Balakrishnan, "TCP Ex machina," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 123–134, 2013.

[17] X. Nie, Y. Zhao, Z. Li et al., "Dynamic TCP Initial Windows and Congestion Control Schemes Through Reinforcement Learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1231–1247, 2019.

[18] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, "QTCP: Adaptive Congestion Control with Reinforcement Learning," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 3, pp. 445–458, 2019.

[19] K. Ong, D. Murray, and T. McGill, "Large-sample comparison of tcp congestion control mechanisms over wireless networks," in *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 420–426, Crans-Montana, Switzerland, 2016.

[20] F. Chiariotti, S. Kucera, A. Zanella, and H. Claussen, "Analysis and Design of a Latency Control Protocol for Multi-Path Data Delivery With Pre-Defined QoS Guarantees," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1165–1178, 2019.

[21] M. Alizadeh, A. Greenberg, D. A. Maltz et al., "Data center TCP (DCTCP)," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM '10*, vol. 40, Conference, New Delhi, India, 2010no. 4.

[22] M. C. Chan and R. Ramjee, "Improving TCP/IP Performance over Third-Generation Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 4, pp. 430–443, 2008.

[23] N. Blefari-Melazzi, D. Di Sorte, M. Femminella, and G. Reali, "Autonomic control and personalization of a wireless access network," *Computer Networks*, vol. 51, no. 10, pp. 2645–2676, 2007.

[24] V. Shrivastava, S. Rayanchu, S. Banerjee, and K. Papagiannaki, "PIE in the Sky: Online Passive Interference Estimation for Enterprise WLANs," *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, , , pp. 337–350, USENIX Association, 2011.

[25] M. Maity, B. Raman, and M. Vutukuru, "TCP Download Performance in Dense WiFi Scenarios: Analysis and Solution," *IEEE Transactions on Mobile Computing*, vol. 16, no. 1, pp. 213–227, 2017.

[26] D. N. da Hora, R. Teixeira, K. van Doorselaer, and K. van Oost, "Predicting the Effect of Home Wi-Fi Quality on Web QoE," in *Proceedings of the 2016 workshop on - Internet-QoE '16*, pp. 13–18, Florianopolis Brazil, 2016.

[27] D. Jaisinghani, V. Naik, S. K. Kaul, and S. Roy, "Sniffer-Based Inference of the Causes of Active Scanning in WiFi Networks," in *2017 Twenty-Third National Conference on Communications (NCC)*, pp. 1–6, Chennai, India, 2017.

[28] X. Chen, S. Xu, X. Chen, S. Cao, S. Zhang, and Y. Sun, "Passive TCP identification for wired and wireless networks: a long-short term memory approach," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 717–722, Tangier, Morocco, Morocco, 2019.

[29] M. A. Ergin, K. Ramachandran, and M. Gruteser, "An experimental study of inter-cell interference effects on system performance in unplanned wireless LAN deployments," *Computer Networks*, vol. 52, no. 14, pp. 2728–2744, 2008.

[30] G. Kuriakose, S. Harsha, A. Kumar, and V. Sharma, "Analytical models for capacity estimation of IEEE 802.11 WLANs using DCF for internet applications," *Wireless Networks*, vol. 15, no. 2, pp. 259–277, 2009.

[31] S. Lin, N. Che, F. Yu, and S. Jiang, "Fairness and Load Balancing in SDWN Using Handoff-Delay-Based Association Control and Load Monitoring," *IEEE Access*, vol. 7, pp. 136934–136950, 2019.

[32] K. Sui, Y. Zhao, D. Pei, and L. Zimu, "How bad are the rogues' impact on enterprise 802.11 network performance?," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 361–369, Kowloon, Hong Kong, 2015.

[33] U. P. Moravapalle, S. Sanadhya, A. Parate, and K.-H. Kim, "Pulsar: Improving Throughput Estimation in Enterprise LTE Small Cells," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, pp. 1–7, Heidelberg, Germany, 2015.

[34] D. Neves da Hora, K. Van Doorselaer, K. Van Oost, R. Teixeira, and C. Diot, "Passive Wi-Fi Link Capacity Estimation on Commodity Access Points," in *Traffic Monitoring and Analysis Workshop (TMA)*, Louvain-la-Neuve, Belgium, 2016.

[35] E. Lovisotto, E. Vianello, D. Cazzaro et al., "Cell Traffic Prediction Using Joint Spatio-Temporal Information," in *2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1–4, Thessaloniki, Greece, 2017.

[36] C. Pei, Y. Zhao, G. Chen et al., "How Much Are Your Neighbors Interfering with Your WiFi Delay?," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, Vancouver, BC, Canada, 2017.

[37] S. R. Pokhrel and C. Williamson, "Modeling Compound TCP Over WiFi for IoT," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 864–878, 2018.

[38] B. Pfaff, J. Pettit, T. Koponen et al., "The Design and Implementation of Open vSwitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pp. 117–130, USENIX Association, Oakland, CA, 2015.

[39] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced Study of SDN/OpenFlow Controllers," in *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, pp. 1–6, Moscow, Russia, 2013.

[40] S. McKee, L. Cottrell, and M. Babik, "ICFA SCIC Network Monitoring Report," in *International Committee for Future Accelerators (ICFA)*, Geneva, Switzerland, 2016.

[41] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[42] G. Ke, Z. Xu, J. Zhang, J. Bian, and T.-Y. Liu, "DeepGBM: A Deep Learning Framework Distilled by GBDT for Online Prediction Tasks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 384–394, Anchorage, AK, USA, 2019.

[43] H. Jung, T. T. Kwon, K. Cho, and Y. Choi, "REACT: Rate Adaptation using Coherence Time in 802.11 WLANs," *Computer Communications*, vol. 34, no. 11, pp. 1316–1327, 2011.

[44] D. H. Hagos, P. E. Engelstad, A. Yazidi, and Ø. Kure, "A Machine Learning Approach to TCP State Monitoring from Passive Measurements," in *2018 Wireless Days (WD)*, pp. 164–171, Dubai, United Arab Emirates, 2018.

[45] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP connection characteristics through passive

measurements," in *IEEE INFOCOM 2004*, vol. 3, pp. 1582–1592, Hong Kong, China, 2004.

[46] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 54–66, 2007.

[47] E. Coronado, R. Riggio, J. Villalón, and A. Garrido, "Wi-balance: Channel-aware user association in software-defined Wi-Fi networks," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, Taipei, Taiwan, 2018.