Fast Packet Loss Inferring via Personalized Simulation-Reality Distillation

Wenchao Xu, *Member, IEEE*, Haodong Wan, Haozhao Wang, Nan Cheng, *Member, IEEE*, Quan Chen, *Member, IEEE*, Haibo Zhou, *Senior Member, IEEE*, and Song Guo, *Fellow, IEEE*

***Abstract*—Packet loss inferring can enable a transceiver to distinguish between channel impairment and collision for transmission failures, and thus can improve the network performance by exclusively performing rate adaptation or adjusting the medium access parameter. Machine learning methods from literature have shown great potential in producing models that can detect the loss causes over various network trace, however haven't considered accurate data-driven loss inferring on resource-constrained devices that cannot accommodate deep models. In this paper, we propose a novel packet loss inferring framework that can train lightweight models to distinguish between channel losses and collisions by learning the data trace from both simulation and real devices. Specifically, we first train a sophisticated teacher model based on extensive simulation datasets, whose knowledge is then transferred to a small student model that can be deployed on tiny device. The simulation-reality distillation is conducted via personalized trace from each client correspondingly, whose performance bound is analytically guaranteed. We have implemented our method on real testbed and show that the network access performance can be significantly improved, especially for sudden network variations.**

***Index Terms*—Medium access control, rate adaptation, knowledge distillation, simulation-reality gap, loss diagnosis.**

I. Introduction

PACKET loss inferring has been a long-standing challenge for multi-user wireless access systems, which is required to track the cause of transmission failures, i.e., either due to insufficient channel conditions or collision caused by multi-user concurrent transmission [1], [2]. Wireless transceivers, e.g., IEEE 802.11 distributed coordination function (DCF) based WiFi interfaces, can benefit from such tracking and fundamentally improve the link performance, especially in dense or mobile conditions, such as airport, meeting room, vehicular communication, etc., where frequent signal blocking or intensive collisions could cause extremely low medium access control (MAC) efficiency and degraded throughput performance [3], [4]. That is because, for contention based distributed wireless access systems, e.g., Wireless PAN, WiFi, etc., the transceivers can only determine if the transmission is successful by checking the feedback ACK frame, while cannot infer the packet loss cause if no ACK is received. For example, a WiFi transceiver would simultaneously double the back-off window and apply degraded modulation and coding scheme (MCS) upon packet loss, leading to even more crowded channel and thus excessive time for the following re-transmission, which can further degrade the network performance.

Existing literature has intensively investigated the loss inferring to differentiate the collision from channel loss, e.g., to improve the rate adaptation [5], [6], [7], [8], or to optimize the random access parameters [9], [10]. Methodologies include protocol re-design, signal monitoring and transmission statistics collection, etc. For example, Sen et al. propose to add an extra function to notify the transmitter of the collision happened during the transmission process [11], [12], [13] propose to monitor the trend of the signal strength to infer the loss from channel cause, and [14] observe the statistical error patterns, e.g., error rate and bit-level or symbol level error distribution to differentiate the collision from channel loss. However, there schemes often require non-trivial efforts to revise the protocol, or suffer from low loss inferring accuracy with large delays due to inaccurate channel tracking or outdated statistical results [15], and thus are not always effective in practice.

As a solution, machine learning (ML) paradigms are applied to infer the packet loss from various network traces in a model-free way. Hermans et al. propose a support vector machine (SVM) based classifier to classify the interference via mining the corrupted packets [16]. Yi et al. utilize a deep neural network to successfully infer loss from different interference source for wireless sensor networks [17]. Chen et al. also employ a neural network to distinguish between wireless and congestion loss to improve the link management [18]. Compared with traditional model-based methods that explicitly construct packet inferring functions following the protocol standards [19], ML-based

Manuscript received 13 December 2022; revised 26 April 2023; accepted 18 May 2023. Date of publication 31 May 2023; date of current version 4 April 2024. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, China under Grant PolyU15222621, in part by the National Natural Science Foundation of China under Grants U1836204, U1936108, and 62206102, in part by the Science and Technology Support Program of Hubei Province under Grant 2022BAA046, and in part by the Research Grants Council under the Areas of Excellence scheme under Grant AoE/E-601/22-R. Recommended for acceptance by Y. Fang. *(Corresponding author: Haozhao Wang.)*

Wenchao Xu and Song Guo are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, SAR, China (e-mail: wenchao.xu@polyu.edu.hk; song.guo@polyu.edu.hk).

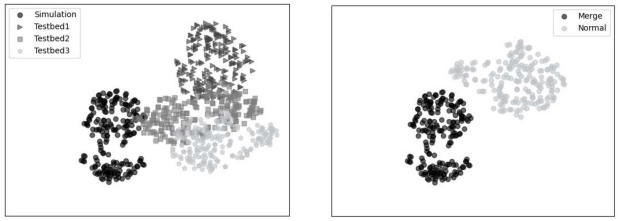
Haozhao Wang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China (e-mail: hz_wang@hust.edu.cn).

Haodong Wan and Nan Cheng are with the School of Telecomm. Engineering, Xidian University, Xian, Shanxi 710126, China (e-mail: hdwan@stu.xidian.edu.cn; dr.nan.cheng@ieee.org).

Haibo Zhou is with the School of Electronic Science and Engineering, Nanjing University, Nanjing, Jiangsu 210093, China (e-mail: haibozhou@nju.edu.cn).

Quan Chen is with the School of Computers, Guangdong University of Technology, Guangdong, Guangzhou 510006, China (e-mail: quan.c@gdut.edu.cn).

Digital Object Identifier 10.1109/TMC.2023.3281725



(a) Simulation-reality gap (b) Data shift when users merge

Fig. 1. Visualizations produced by t-SNE from the MAC parameters time series data. (Reducing the high dimension data to two-dimensional vectors) (a) There are difference between simulation data and real testbed. Different testbed also have gaps with each other. (b) Simulation data are shifted when users merge, i.e., we can draw a boundary between the collision and normal points.

methods can infer the network status based on the transmission traces, e.g., collected from network stack log, etc., and thus are easier to implement and can be generalized to practical scenarios [14], [20].

However, there remains crucial challenges to implement the ML methods for loss inferring. First, an accurate model often requires large training datasets, which need non-negligible efforts to collect from real network systems. Besides, deep model inference can consume significant computing power that are often beyond the capacity of most tiny Internet of things (IoT) devices for packet-level loss inferring. To save the efforts for building the training datasets, some learning-based solutions utilize the simulation trace for model training [21], [22]. Nonetheless, it is pointed out that there may be non-negligible gap between the simulation trace and the reality, which would compromise the performance when applying the simulation-trained model to real hardware [23]. In our preliminary experiment results that shown in Fig. 1(a), such simulation-reality gap can also be observed from the visualization produced from the time series of the back-off window during the node merge event (i.e., significant collisions would happen) [24]. Fig. 1(a) also show that the gaps among different transceivers cannot be neglected, which may come from their diverse external environments and internal network stack executive conditions, e.g., clock shift, catching effect. So simply training a neural network for all network devices cannot always well infer the loss for different clients.

In this paper, we present a brand-new ML design to achieve fast packet loss inferring that can accurately distinguish between collision and channel impairment for classical multi-user wireless access scenario where carrier-sense multiple access with collision avoidance (CSMA/CA) scheme is employed. We identify two key scenarios where accurate loss inferring is crucial to the network performance, i.e., sudden channel deterioration and massive users merge,[1] which can avoid unnecessary back-off waiting time and conservative rate selection upon transmission failures. Specifically, we generate the network traces from the multi-access simulation of the protocol to train a sophisticated teacher model, whose knowledge is distilled to a small model for each user based on a small set of the personalized trace from corresponding devices. We have theoretically shown the distillation efficiency from simulation to reality and implement the proposed method on real devices. Experimental results show that the student model can support effective loss inferring on real hardware and significant performance improvement can be achieved.

The contribution of the paper can be summarized as follows.

- we propose a novel ML based loss inferring framework to train effective model from both simulation and real data.
- We design a two-phase training method that can transfer the knowledge from the protocol simulation to the real devices so that a lightweight neural model can support accurate loss inferring on resource-constrained devices.
- We theoretically analyze the distillation efficiency from simulation to reality regarding different sizes of the distillation dataset.
- We have implemented our method on real devices, which experimentally show that the proposed methods can significantly improve the network performance, especially for sudden channel deterioration and users merge cases.

The remainder of the paper is organized as follows. In Section II we present the related works to this paper. Section III shows the detailed design of the proposed framework. We provide our theoretical analysis about the simulation-reality distillation in Section IV, and show the performance evaluation in Section V. And finally Section VI concludes the paper.

II. Related Work

Loss inferring aims to differentiate the cause of packet drop between channel impairment and collision. In this section, we introduce the related works regarding the medium access and rate adaptation (RA) schemes that can benefits a lot from loss inferring, and then we also introduce related machine learning schemes for MAC and RA.

A. CSMA/CA Protocol and Rate Adaptation

The CSMA/CA MAC scheme has been widely applied in popular IoT network protocols, e.g., IEEE 802.11, IEEE 802.15.4. The main principal is that before send the packet to air, each user should listen to the channel status and would perform back-off process, i.e., maintain a back-off counter which decreases only if the channel is idle, only when the packet would be sent. If a transmission fails, the back-off window would be increased to alleviate the channel load and thus the collision probability [25].

RA is to determine the MCS level according to the channel status. Unlike cellular networks that employ dedicated control channels to track the channel status and set the MCS correspondingly, WiFi or WPAN rely on the statistical transmission results (e.g., Auto Rate Fallback [26]) or channel probing (e.g., Minstrel [27]) to determine the MCS level for the egress packets.

The back-off adjustment and MCS selection should be conducted according to different packet loss causes as they are designed for collision alleviation and channel adaptation purposes respectively. Accurate loss inferring can avoid unnecessary

[1]Such as connected vehicles arrive simultaneously at road intersection, or massive unmanned aerial vehicles merge.

oversize back-off size or unneeded conservative MCS selection comparing with current program running on most IoT devices, which would by default conduct MAC and MCS adjustment simultaneously.

Existing literature and experiments have well shown that in static or quasi-static conditions, current RA schemes, e.g., Minstrel, can well track the slow channel variations and thus minimize the channel loss, and thus most packet loss can be approximately attributed to collision [27]. However, in highly dynamic conditions where sudden users merge or signal attenuation, e.g., vehicles merge at intersection, it is necessary to identify the loss cause and perform the right operation correspondingly.

B. Loss Inferring for Rate Adaptation (RA)

It is shown that inaccurate channel estimation can cause significant throughput loss [28]. [29] show that differentiate the loss can greatly improve the network access efficiency. [5] proposes to examine the loss pattern in preambles to resolve the concurrent transmission, and thus improve the RA accuracy. [6] employ the TXOP function in IEEE 802.11e to distinguish collision from channel noise, and thus optimize the rate selection. [7], [8] leverage the RTS/CTS probing to identify the collision. Zhou et al. propose to infer the collision types by examining the channel idle time status and the active neighbors [30]. Pang et al. propose to verify the loss reason by checking if the packet header can be successfully received, which can indicate if collision happened and thus optimize the rate selection for next packets [31]. Pefkianakis et al. investigated the 'loss pattern' from the frame aggregation and BlockACK, and infer if it is caused by collision which is then used to improve the performance of the RA [32]. [33] utilized the relationship between noisy error and subsequent fragments packet loss and distinguish it from collision. These methods require explicit network modelling or protocol knowledge to construct the loss inferring function, and haven't considered the gap between protocol implementation and device discrepancies.

C. Machine Learning for MAC

To coordinate the transmission for multiple users, reinforcement learning (RL) has been applied to maximize the network access performance. [34] propose to employ Multi-Agent RL to guide the channel access for WiFi users. Kumar et al. propose to apply the deep Q-learning to adaptively adjust the contention window to improve the MAC performance [35].Bayat-Yeganeh et al. train multi-state RL agents to track the optimal probability of transmission for P-persistent CSMA networks [36]. However, existing works have attributed the packet loss to the simultaneously transmission collision only, and have yet considered the channel error. In this paper, we use the recent network logging data to infer the packet loss for CSMA networks. It is also possible to apply the mechanism to other multi-user access networks, e.g., TDMA networks, where there also exist transmission conflicts. Identifying the conflicts from channel error can help to avoid unnecessary time slot adjustment or rate degradation, and can achieve the network performance following the same principle of the paper.

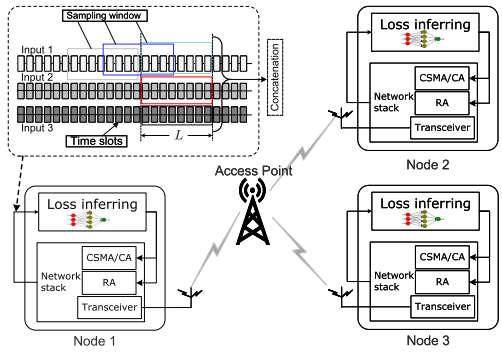


Fig. 2. System Model: multiple user with CSMA/CA scheme for AP access.

D. Machine Learning for RA

ML enabled methods are extensively applied in RA and MAC by conducting model-free design, i.e., conduct neural network training and inference based on network trace without prior network protocol knowledge. Chen et al. apply the reinforcement learning to capture the channel condition online and adjust the rate selection for IEEE 802.11ac devices. [37]. Xu et al. propose to map the signal strength time series to the future channel status, which can optimize the rate selection for vehicle users [38]. [39] utilizes neural networks to predict the throughput of different rate options for optimal WiFi MCS selection. [40] experimentally shows that machine learning methods can help to find a suitable MCS for IEEE 802.11c even in mobile conditions.

E. Simulation-Reality Gap

Accurate loss inferring should consider both the protocol specification and the actual discrepancies of actual devices. Current ML methods for smart networking are often based on simulated datasets [41], [42], and have yet to consider the simulation-reality discrepancy, which is also shown in Fig. 1(b) in our case. [43] utilize both the simulation trace and real-world test to optimize the learning-based network configuration. [23] consider to apply the teacher-student learning to adapt the simulation knowledge to reality, which has inspired us to transfer the protocol knowledge to the real device.

III. Fast Packet Loss Inferring Design

Our objective is to train a neural model that can accurately infer the loss cause while restrict the computing power for each inference. We first train a teacher model using the abundant simulated trace, and then for each user, conduct knowledge distillation between the teacher model to a student model using a few real traces collected from corresponding device.

A. System Model

We consider a time-slotted multi-user wireless access system, where N clients follow the CSMA/CA protocol to compete for the channel resource for transmitting data packets to an access point (AP). As shown in Fig. 2(a) dedicated loss inferring thread read the time series sampling from MAC layer parameters, of length L. When a data packet is scheduled for transmission,

the transceiver should determine the back-off size and the MCS level before it is sent to air. For simplicity, we consider the basic IEEE 802.11 DCF [25] and the original ARF scheme [26] for the MAC and RA protocols respectively on a real device to show the performance gain from accurate loss inferring.

The loss inferring thread collects L samplings of the network stack parameters at recent L time slots when there is a packet loss (ACK timeout), including back-off, send and ACK status at each time slot, and concatenate into a vector (a data sample) v_t. The purpose is to train model $\mathbf{w}$ so that the classifier $h_{\mathbf{w}}(\cdot) : x \rightarrow y$ can map v_t to the loss cause, i.e., $y = (y_1, y_2)$, y_1 indicates channel impairment and y_2 indicates concurrent collision.

Client devices run the original CSMA/CA and RA schemes. When an ACK timeouts and re-transmission is scheduled. We consider the saturated case and no limits for the re-transmission tries. The classifier results will be used to control the corresponding MAC and RA components. For channel loss, only the RA parameters would be updated and the back-off window would stay unchanged. For collision loss, user will not change the MCS level and only increase the back-off window size.

We examine the effectiveness of loss inferring by examining two typical scenarios, users merge and sudden channel deterioration. In users merge case, a set of new users will join in the channel competition, which will significantly increase the collision probability; in sudden channel deterioration case, the number of users remains does not change and the channel gain is manually reduced for some users, which means that their packets will likely be dropped for too aggressive MCS selection before the RA could react and adjust to proper level.

B. Data Collection

According to CSMA/CA protocol, when a data packet is scheduled for transmission and then the back-off counter is initialized, if the channel is sensed busy, the counter will be frozen until when the channel become idle. Thus the channel variation trend can be indicated by monitoring the back-off status of the device by a series of Boolean variable b at each time slot, i.e., $b = 1$ indicates that back-off counter is non-zero while $b = 0$ means that back-off counter is zero. If a data packet is under transmission, then b is set to 0. The historical back-off status can show the channel crowdedness variations and thus can reflect the trend of the collision probability.

For previous transmission attempts, their starting moments show that the channel has been idle at least for DIFS duration, which can also reflect the level of channel competition. If the transmission is successful, then the moment when receiving the ACK frame can show there is no collision in this transmission duration. The send and ACK status can also be represented by Boolean variables, i.e., $s = [0, 1]$ indicates there is a transmission starts at this time slot while $a = [0, 1]$ indicates there is an ACK received at this time slot.

The time slot for the MAC scheme is denoted by σ. Assume that at time t, an ACK frame timeouts. Then we sample N values at time moments $[t - (N-1)\sigma, t - (N-2)\sigma..., t-1, t]$ for all the above-mentioned indicators, i.e., the back-off status, the transmission starting and ACK receiving status. Then these time series data are pre-processed to a status vector v, i.e., fist compress by summing up the consecutive K data and then concatenate them to v.

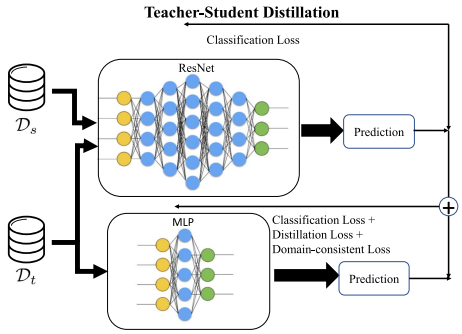


Fig. 3. Teacher-Student distillation for adapting simulation to reality.

We have created users merge event in simulation environment where five clients are competing the channel to access to an AP and another five clients suddenly join in the channel access competition. We have collected these three indicators series and the visualization is shown in Fig. 1(b). We can observe that there exists clear boundary between normal time and the merge moment, which show potential that training a neural model can also differentiate the collision-caused loss.

Except for the simulation trace, we also conduce the users merge and channel deteriorate test on our testbed and collect the trace with corresponding labels, i.e., approximately attribute the packet loss to collision for merge case and to channel loss for the latter. Fig. 1(a) shows data trace from different testbeds are different, and thus when training the student models for different clients, we use corresponding testbed trace from that client.

C. Simulation-Testbed Knowledge Distillation

Deep model with more layers that trained over large datasets often can provide higher accuracy. However, IoT devices are often restrained in computing resource and memory space, and has difficulties in accommodate sophisticated models. In quick variation conditions like users merge or sudden channel deterioration, frequent packet loss may happen and corresponding model inference can consume significant computing power and cause excessive inference delay and outdated results. Besides, it it time-consuming to collect the trace from real devices, especially the labelling process.

To achieve accurate loss inferring over tiny IoT devices, we design a teacher-student knowledge distillation method that can transfer the knowledge from abundant simulation data to real devices with limited ground truth trace. The framework is shown in Fig. 3. The datasets from simulation and the testbed are denoted by $\mathcal{D}_s$ and $\mathcal{D}_t$ respectively. We use a relatively large model with parameters $\mathbf{w}_s$ to capture the priors from the simulation model, and then perform knowledge distillation from the simulation model to the testbed model $\mathbf{w}_t$. The goal of the simulation model training is to learn parameters that minimize

the following loss over $\mathcal{D}_s$:

$$\min_{\mathbf{w}_s} \mathcal{L}(\mathbf{w}_s) := - \mathop{\mathbb{E}}_{x \in \mathcal{D}_s} y \log(h_{\mathbf{w}_s}(x)). \tag{1}$$

For the testbed model training, the loss function we optimize over $\mathcal{D}_t$ is determined by $\mathcal{D}_t$'s loss, distillation loss $\mathcal{L}_{dis}$ and the domain consistence loss $\mathcal{L}_{dc}$:

$$\mathcal{L}(\mathbf{w}_t) = \mathcal{L}_t + \alpha \mathcal{L}_{dis} + \beta \mathcal{L}_{dc}. \tag{2}$$

The testbed model classification loss $\mathcal{L}_t$ is to learn from the real trace and defined as

$$\mathcal{L}_t = - \mathop{\mathbb{E}}_{x \in \mathcal{D}_t} y \log(h_{\mathbf{w}_t}(x)), \tag{3}$$

where y is the label from users merge and channel deterioration experiments and $h_{\mathbf{w}_t}(x)$ is the inference result of the testbed model.

The distillation loss $\mathcal{L}_{dis}$ is set to

$$\mathcal{L}_{dis} = - \mathop{\mathbb{E}}_{x \in \mathcal{D}_t} q log(h_{\mathbf{w}_t}(x)), \tag{4}$$

where q is the softmax probability from the simulation model over the testbed trace:

$$q = \frac{exp(\frac{m_i}{T})}{\sum\limits_{j \in [y_1, y_2]} exp(\frac{m_j}{T})} \tag{5}$$

T is the distillation temperature [23] to generate soft probability from the output of the simulation model, i.e., m_i, whereby the simulation knowledge can be transfer to the testbed model.

The domain consistence loss $\mathcal{L}_{dc}$ is defined as

$$\mathcal{L}_{dc} = || \mathop{\mathbb{E}}_{x \in \mathcal{D}_s} h_{\mathbf{w}_t}(x_s) - \mathop{\mathbb{E}}_{x \in \mathcal{D}_t} h_{\mathbf{w}_s}(x_t)|| \tag{6}$$

It is to regulate the simulation and the testbed datasets in a latent embedding space by minimizing their distribution difference.

The detailed training procedure is shown in Algorithm 1, which has two phases. In the first phase, the simulation model w_s is trained only on the simulation datasets $\mathcal{D}_s$. In the second phase, the testbed model w_t is trained with the loss function in (2) based on the testbed datasets $\mathcal{D}_t$ under the guidance of the simulation model. Following the above proceed, the knowledge from massive simulation trace can be transferred to the limited-size testbed model.

Algorithm 1: Simulation-Testbed Distillation.

Initialize w_t, w_s to random values
Set Batch size = 64;

Phase 1 - Simulation Model

procedure Simulation Networkw_s
  for all mini batches in D_s **do**
    Pick mini batch i $D_{s,i}$
    Compute loss $\mathcal{L}(\mathbf{w}_s)$
    Update network model with mini batch i
  end for
end procedure

Phase 2 - Testbed Model

procedure Knowledge Distillation$w_s \rightarrow w_t$
  for All Testbed in B **do**
    Pick Testbed b
    for all mini batches in $\mathcal{D}_{t,b}$ $\mathcal{D}_s$ **do**
      Pick mini batch i $\mathcal{D}_{s,i}$, j $\mathcal{D}_{t,b,j}$
      if Teacher Network **then**
        $q_{soft,j} \leftarrow$ soft targets for mini batch j
        $\mathcal{L}_{dis} \leftarrow - \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}_t} \mathbf{q_{soft,j}} \log(h_{\mathbf{w}_t}(j))$
      end if
      if Student Network **then**
        $\mathcal{L}_t \leftarrow - \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}_t} \mathbf{y} \log(h_{\mathbf{w}_s}(j))$
        $\mathcal{L}_{dc} \leftarrow || \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}_s} h_{\mathbf{w}_t}(i) - \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}_t} h_{\mathbf{w}_s}(j)||$
        $loss = \mathcal{L}_t + \alpha \mathcal{L}_{dis} + \beta \mathcal{L}_{dc}$
        Update testbed model with mini batch j
      end if
    end for
  end for
end procedure

IV. Simulation-Reality Distillation Analysis

We assume that the simulation and testbed models can be approximated as linear classifiers [44], [45]. We analytically investigate the efficiency of the proposed simulation-reality knowledge transfer between the simulation (teacher) and the reality (student). From the analytical results, we can summarize the relationship between the size of the testbed trace and the performance of testbed model.

We establish the theory for the error bound of distillation on classifiers $h \in \mathcal{H}$ which is defined by $h(\mathbf{x}) = \mathbb{1}\{\mathbf{w}^T \mathbf{x} \geq 0\}$ with parameters $\mathbf{w}$ on any data sample $\mathbf{x} \in \mathbb{R}^d$, where $\mathcal{H}$ is the set of all classifiers. We denote D_s the distribution of the simulation dataset on which the teacher linear classifier $h_s \in \mathcal{H}$ is learned with parameters $\mathbf{w}_*$, and D_t the distribution of the testbed dataset with n samples on which the student linear classifier $\hat{h}_t \in \mathcal{H}$ is learned. Besides, our theory relies on the following reverse cdf function:

$$p(\theta) = P_{\mathbf{x} \sim D_t} \left[\cos^{-1} \left(\frac{|\mathbf{w}_*^T \mathbf{x}|}{\|\mathbf{w}_*^T\| \cdot \|\mathbf{x}\|} \right) \geq \theta \right], \tag{7}$$

for any $\theta \in [0, \pi/2]$. We use $d_{\mathcal{H}\Delta\mathcal{H}}$ to measure the domain discrepancy between two distributions. By denoting $\mathcal{L}_D(h)$ the risk of classifier h on the data distribution D, we define $\lambda = \mathcal{L}_{D_t}(h_*) + \mathcal{L}_{D_s}(h_*)$ with $h_* = \arg\min_{h \in \mathcal{H}} \mathcal{L}_{D_s}(h) + \mathcal{L}_{D_t}(h)$ corresponding to idea joint classifier that minimizes the combined error. Besides, our theorems are built on the following existing theorems.

Proposition 1: (Theorem 1, Ben et al. [46]). Considering the distributions D_s and D_t, for every $h \in \mathcal{H}$ and any $\sigma \in (0, 1)$, with probability at least $1 - \sigma$ (over the choice of the samples), there exists

$$\mathcal{L}_{D_t}(h) \leq \mathcal{L}_{D_s}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) + \lambda, \tag{8}$$

where $\lambda = \mathcal{L}_{D_s}(h_*) + \mathcal{L}_{D_t}(h_*)$ with $h_* = \arg\min_{h\in\mathcal{H}} \mathcal{L}_{D_s}(h) + \mathcal{L}_{D_t}(h)$.

Proposition 2: (Theorem 3, Mary et al. [47]). For any training set $\hat{D} = \{\mathbf{x}_i \in \mathbb{R}^d | i = 1, \ldots, n\}$ sampled from the distribution D, let $\hat{h}(\mathbf{x}) = \mathbb{1}\{\mathbf{w}\mathbf{x} \geq 0\}$ be the linear classifier learned by distillation from the teacher h with weight vector $\mathbf{w}_*$. Then, when $n > d$, it holds that

$$\mathbb{E}_{\hat{D}\sim D}[R(\hat{h})] = 0. \tag{9}$$

For $n < d$, it holds for any $\beta \in [0, \pi/2]$ that

$$\mathbb{E}_{\hat{D}\sim D}[R(\hat{h})] \leq p(\beta) + p(\pi/2 - \beta)^n, \tag{10}$$

where $R(h) = P_{\mathbf{x}\sim\mathbf{D}}[\hat{h}(\mathbf{x}) \neq h(\mathbf{x})]$ denotes the probability of student $\hat{h}$ predicting differently from teacher h.

Now, we have the following theory for the error bound of the $\hat{h}_t$ distilled on the empirical dataset $\hat{D}_t$.

Theorem 1: By distilling from the simulation classifier h_s on any testbed dataset $\hat{D}_t \in \mathbb{R}^{d\times n}$, the error of student classifier $\hat{h}_t$ on the empirical distribution D_t is bounded:

$$\mathcal{L}_{D_t}(\hat{h}_t) \leq \mathcal{L}_{D_s}(h_s) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) + \lambda, \tag{11}$$

when $n \geq d$, and

$$\mathcal{L}_{D_t}(\hat{h}_t) \leq \mathcal{L}_{D_s}(h_s) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) + \lambda + p(\beta) + p(\pi/2 - \beta)^n, \tag{12}$$

for every $\beta \in [0, \pi/2]$, when $n < d$.

Proof: We start from the relationship of error between the classifier on the distribution of the simulation dataset and that of the testbed dataset. According to Proposition 1, with defining $h_t \in \mathcal{H}$ as the learned classifier on the distribution of testbed dataset D_t, we have

$$\mathcal{L}_{D_t}(h_t) \leq \mathcal{L}_{D_s}(h_s) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) + \lambda. \tag{13}$$

Next, we present the error bound of the distillation. According to Proposition 2, we can obtain the relationship of the error bound between $\hat{h}_t$ and h_t as distilled on testbed dataset $\hat{D}_t$ consisting of n samples with d dimensions on each sample. Specifically, when $n \geq d$, it holds that

$$\mathcal{L}_{D_t}(\hat{h}_t) = \mathcal{L}_{D_t}(h_t). \tag{14}$$

For $n < d$, it holds for any $\beta \in [0, \pi/2]$ that

$$\mathcal{L}_{D_t}(\hat{h}_t) \leq \mathcal{L}_{D_t}(h_t) + \lambda + p(\beta) + p(\pi/2 - \beta)^n. \tag{15}$$

Considering (14) and (15) with (13) together yields the theorem. ■

Remark 1: The size of testbed dataset is of great importance to the error of learned classifier. The error decreases with the growing of the size of testbed dataset. Nevertheless, the error keeps unchanged after the size of testbed dataset exceeding a certain value.

Remark 2: The domain discrepancy between the distribution of simulation dataset and testbed dataset is positively related to the distillation error. The error decreases with the reducing of the domain discrepancy, and vice versa.

Furthermore, we compare the error between the classifier obtained by distillation and the original classifier trained on testbed dataset only, to present when to adopt the distillation. By defining the $h_{\hat{D}_t} \in \mathcal{H}$ as the original classifier obtained by only training on testbed dataset, we have the following theorem.

Theorem 2: Assuming that the size of simulation dataset is infinite and the capacity of teacher classifier is strong enough, i.e., $\mathcal{L}_{D_s}(h_*) = \mathcal{L}_{D_t}(h_*) = 0$, the error of classifier $\hat{h}_t$ obtained by distillation on the empirical distribution D_t is smaller than $h_{\hat{D}_t}$ with probability at least $1 - \sigma$ for any $\sigma \in (0, 1)$. Specifically, $\mathcal{L}_{D_t}(\hat{h}_t) \leq \mathcal{L}_{D_t}(h_{\hat{D}_t})$ holds when

$$d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) \leq 2\mathcal{L}_{\hat{D}_t}(h_{\hat{D}_t}) + \sqrt{\frac{2\log\frac{2}{\sigma}}{n}}, \tag{16}$$

for $n \geq d$, and

$$\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) + p(\beta) + p(\pi/2 - \beta)^n \leq \mathcal{L}_{\hat{D}_t}(h_{\hat{D}_t}) + \sqrt{\frac{\log\frac{2}{\sigma}}{2n}}, \tag{17}$$

for $n < d$ with any $\beta \in [0, \pi/2]$.

Proof: By the definition of classifier, we know that $h(\mathbf{x}) \in [0, 1]$. Our proof for the error bound of classifier $h_{\hat{D}_t}$ trained only on testbed dataset is based on Hoeffding inequality. Specifically, considering n independent variables $\mathbf{u}_i \in [a_i, b_i]$, $i = 1, \ldots, n$, with the average $\bar{\mathbf{u}} = \frac{1}{n}\sum_{i=1}^n \mathbf{u}_i$, then the Hoeffding inequality holds $P(|\bar{\mathbf{u}} - \mathbb{E}\bar{\mathbf{u}}| \geq \epsilon) \leq 2exp(-\frac{2n^2\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2})$ for any $\epsilon > 0$.

Accordingly, we can bound the error of classifier $h_{\hat{D}_t}$ trained only on testbed dataset:

$$P\left(\left|\mathcal{L}_{D_t}(h_{\hat{D}_t}) - \mathcal{L}_{\hat{D}_t}(h_{\hat{D}_t})\right| > \epsilon\right) \leq 2\exp(-2n\epsilon^2), \tag{18}$$

with the probability at least $1 - \sigma$. Re-organizing (18) obtains

$$\mathcal{L}_{D_t}(h_{\hat{D}_t}) \leq \mathcal{L}_{\hat{D}_t}(h_{\hat{D}_t}) + \sqrt{\frac{\log\frac{2}{\sigma}}{2n}}. \tag{19}$$

Similarly, denoting by $\hat{D}_s$ the empirical distribution of simulation dataset on which the teacher model h_s is trained, then

$$\mathcal{L}_{D_s}(h_s) \leq \mathcal{L}_{\hat{D}_s}(h_s) + \sqrt{\frac{\log\frac{2}{\sigma}}{2m}}, \tag{20}$$

where m is the size of the simulation dataset. Considering that the capacity of h_s is strong enough to enable $\mathcal{L}_{\hat{D}_s}(h_s) \approx 0$, and the size of simulation dataset is infinite to enable $\sqrt{\frac{\log\frac{2}{\sigma}}{2m}} \approx 0$, we have

$$\mathcal{L}_{D_s}(h_s) \approx 0. \tag{21}$$

Since $\lambda = \mathcal{L}_{D_t}(h_*) + \mathcal{L}_{D_s}(h_*)$, the similar derivation as $\mathcal{L}_{D_s}(h_s)$ can be obtained, i.e.,

$$\lambda \approx 0. \tag{22}$$

when the size of simulation dataset is infinite and the capacity of teacher model is strong enough.

Now, joint considering (19), (21), (22), and (11) in Theorem 1 together derives the (16) in Theorem 2. Putting (19), (21), (22), and (12) in Theorem 1 together obtains the (17) in Theorem 2.■

From the above analysis, we can conclude from Theorem 2 that the distilled testbed model can theoretically approach to the performance of the simulation model when the size of simulation dataset is large and the capacity of teacher model is strong. Normally, these two conditions can be satisfied in our framework since the simulation dataset can be generated from the protocol standard, i.e., can have as much as we want. The simulation model is not deployed on device and thus its size is not restricted. So we can infer that the dominant factor to affect the testbed model performance is the discrepancy between the distribution of the simulation dataset and the testbed dataset. Besides, from (11)–(12), we can learn that the accuracy of the testbed model will increase with the testbed trace until the testbed trace size reaches certain threshold, i.e., be bounded. Such conclusion is also demonstrated in our experiments in next Section. Thus we not only use the personalized testbed trace from corresponding clients, we also employ different size of the datasets used for distillation, i.e., depends on when the loss of the testbed model achieves the threshold.

V. Performance Evaluation

We conduct experimental evaluation of the proposed framework in real devices. First, we implement the IEEE 802.11 DCF MAC scheme in both simulation and testbed to generate corresponding traces. For the simulation model, we use the ResNet34, and for the testbed we use a multi-layer classifier with two hidden layers that can cost 2 ms for one model inference operation at Raspberry Pi 4B. Each testbed model is trained following Algorithm 1 and deployed to test the loss inferring performance.

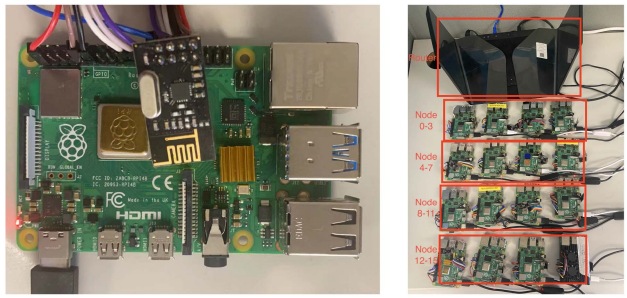
A. Trace collection

It is straightforward to have both the data and corresponding labels from the simulation environment as the collision and channel loss can be artificially generated. For the testbed, it is not practical to generate pure collision or channel loss events and label the corresponding data, so we create the users merge and channel deterioration events, where the dominant reason for packet loss is the inter-user collision and channel loss respectively. The data we use in this paper can be obtained from network driver program of most devices[2]. In the experiment, we set N to 100, i.e., 100 historical samples for the backoff status b, ACK status s and the transmission starting moments, and fed to the machine learning models.

B. Network Implementation

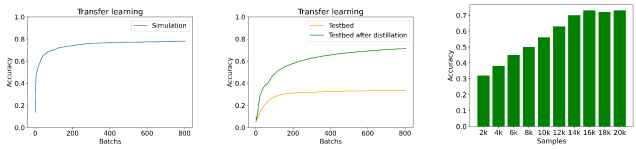
We use an wireless transceiver module *nRF24L01* working in 2.4 GHz unlicensed frequency with three MCS levels (250 kbps/1 Mbps/2 Mbps). The entire MAC and ARF protocols are implemented using the transceiver interface that run on Raspberry Pi 4B, as shown in Fig. 4.

[2]E.g., the back-off parameter can be obtained from the 'back-off timer Counter' from the WiFi driver in Linux kernel.



(a) nRF24L01 module driven by Raspberry Pi (b) Multi-testbed MAC test

Fig. 4. Testbed: nRF24L01 driven by Raspberry Pi.



(a) Simulation model accuracy (b) Testbed model accuracy (c) Testbed model accuracy vs. distillation trace size

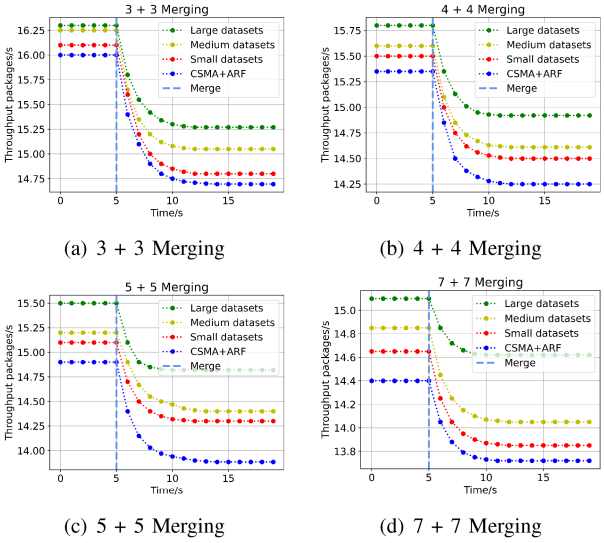
Fig. 5. Model accuracy during training.

For multi-testbed test, all client nodes compete the 2.4 GHz channel send uplink packets to an AP node. All Raspberry Pi systems are connected to a logging script on a computer via the 5 GHz WiFi through a NETGEAR router rax200. We use Pytorch to train the models, which is also installed on Raspberry Pi to perform the model inference. We implement the network protocols (MAC and RA) in python so that we can directly apply the inference results to control the access behavior.

C. Experiment Results

To evaluate the loss inference results and eliminate other factors, we let the testbed follow the basic IEEE 802.11 DCF without RTS/CTS scheme and the ARF rate selection scheme when boot up. For any packet drop, if the loss cause is collision, then the rate selection for the next transmission attempt will not change; if it is channel loss, then the back-off window for next attempt will keep the same as the previous value. We show that such simple scheme can effectively improve the throughput and reduce the transmission latency for each packet.

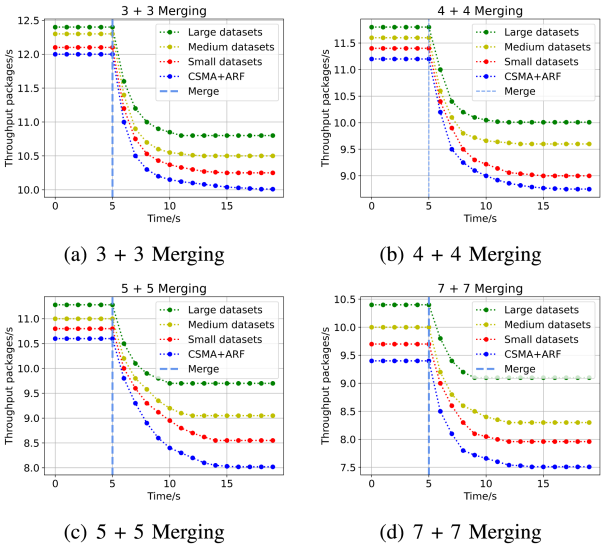
1) Testbed Model Accuracy: During the training phase, we measure the accuracy of a simulation model trained on simulation trace (in Fig. 5(a)), a testbed model trained solely on testbed trace and trained by distillation (in Fig. 5(b)). It can be observed that the testbed model accuracy can be greatly improved by distillation than just training over testbed trace, which show that the knowledge from large simulation trace can be effectively transferred to small testbed models that consume limited inference time, which is critical for resource constrained devices.



(a) 3 + 3 Merging (b) 4 + 4 Merging

(c) 5 + 5 Merging (d) 7 + 7 Merging

Fig. 6. Throughput performance for users merge in strong SNR.



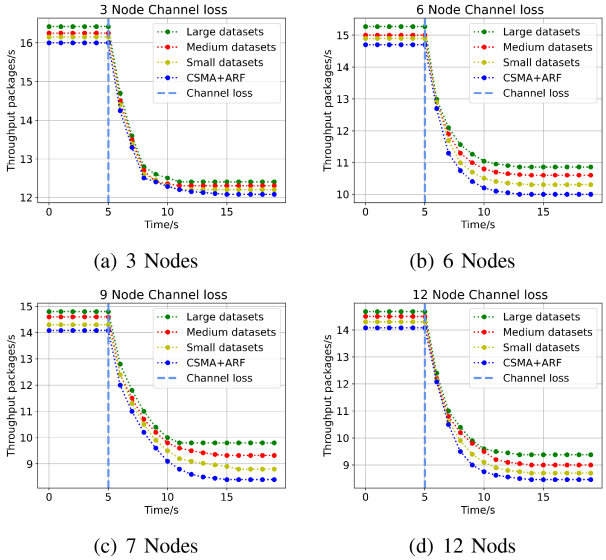
(a) 3 + 3 Merging (b) 4 + 4 Merging

(c) 5 + 5 Merging (d) 7 + 7 Merging

Fig. 7. Throughput performance for users merge in weak SNR.

Fig. 5(c) shows the relationship between a testbed model accuracy and the size of the testbed trace used for the distillation. The accuracy will first increase to a certain value and then stop rising, which is consistent with the conclusion in Section IV and motivates us to determine the proper size of testbed trace.

2) Network Performance Metric: The distillation-trained testbed models over corresponding personalized testbed datasets are deployed on them respectively. We have measured the network performance for two typical scenarios, i.e., users merge and channel deterioration to verify how the access performance can be improved by loss inferring. Figs. 6 and 7 show the network throughput in a time duration when users merge happens at different network scales. It can be observed that during both static period or users merge moments, the distilled testbed model can always outperform than the default standard (CSMA + ARF)



(a) 3 Nodes (b) 6 Nodes

(c) 7 Nodes (d) 12 Nods

Fig. 8. Throughput performance for sudden channel deterioration.

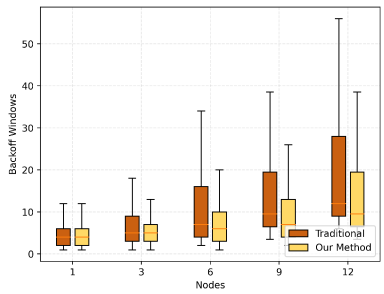
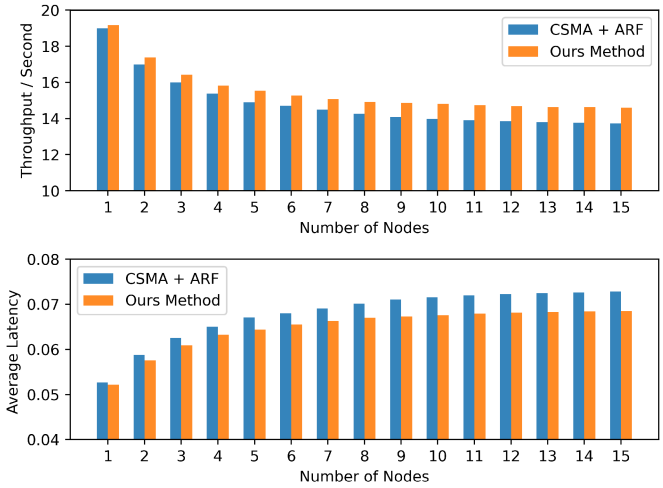


Fig. 9. Statistics of back-off windows.

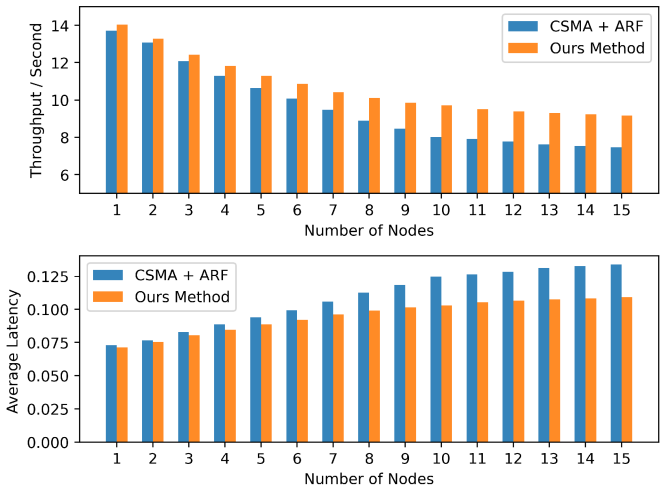
that has no loss inferring guidance for the network access. Besides, large distillation datasets from reality can bring significant performance gain by comparing high (15 k samples), medium (10 k samples) and small (5 k samples) amounts of testbed trace data. In addition, we observe that the loss inferring results from our method can help the device to quickly adapt to the new contention condition after users merge event.

Fig. 8 shows the throughput performance when there is a channel deterioration event where the signal strength between each client and the AP is manually turned down at a certain moment. We can have similar observation that the loss inferring can improve the network performance during and after the sudden channel deterioration event by detecting the cause of the packet dropping.

Fig. 10 shows that our method scale well over different numbers of clients in the experiment duration (10 minutes) at static conditions where no manual events are conducted. The aggregated throughput of the whole network can be improved in both strong and weak signal-to-noise ratio (SNR) conditions. It is observed that higher performance gain can be achieved for more clients, showing the importance of the loss inferring as there are much more collisions for large size network, and thus are useful for future massive IoT network access.



(a) Network throughput at stable condition: strong SNR



(b) Network throughput at stable condition: weak SNR

Fig. 10. Aggregated throughput and average latency performance for different network size.

TABLE I
THROUGHPUT - ABLATION EXPERIMENT

	3	6	9	12
Our Method	16.42	15.27	14.86	14.66
CSMA + ARF	16.02	14.70	14.08	13.86
Simulation Data Only	16.40	15.16	14.57	14.32
Testbed Data Only	16.02	14.93	14.26	14.08
Classification Loss Only	16.18	15.05	14.28	14.10

We also record the sizes of the back-off windows in users merge event duration. It can be shown in Fig. 9 that for a collision-dominant scenario, our method can significantly reduce the back-off window size, i.e., avoiding unnecessary back-off and shortening the waiting time for egress packets and thus improve the throughput.

3) Ablation study: We have conducted the ablation study by comparing the overall network throughput of different configurations, as listed in Table I.

VI. CONCLUSION

In this article, we have proposed a novel loss inferring framework that can transfer the knowledge learned from network simulation to real testbed via a teacher-student distillation method to diagnose the packet loss cause. We have analyzed the effectiveness of such simulation-to-reality knowledge distillation and tailored personalized training for different clients based on their own practical trace. We have implement the proposed method on real devices, which show that the network performance can be significantly improved by explicitly guiding the MAC and RA process from accurate loss inferring.

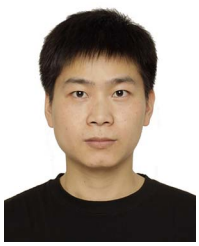
REFERENCES

[1] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee, "Diagnosing wireless packet losses in 802.11: Separating collision from weak signal," in *Proc. 27th Conf. Comput. Commun.*, 2008, pp. 735–743.
[2] I. E. Khayat, P. Geurts, and G. Leduc, "Enhancement of TCP over wired/wireless networks with packet loss classifiers inferred by supervised learning," *Wirel. Netw.*, vol. 16, no. 2, pp. 273–290, 2010.
[3] L. Ho and H. Gacanin, "Design principles for ultra-dense Wi-Fi deployments," in *Proc. IEEE Wirel. Commun. Netw. Conf.*, 2018, pp. 1–6.
[4] Z. Zhong, P. Kulkarni, F. Cao, Z. Fan, and S. Armour, "Issues and challenges in dense WiFi networks," in *Proc. Int. Wirel. Commun. Mobile Comput. Conf.*, 2015, pp. 947–951.
[5] L. Kong, Y. Cao, L. He, G. Chen, M.-Y. Wu, and T. He, "Multi-rate selection in ZigBee," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1055–1068, 2019.
[6] K. D. Huang, K. R. Duffy, and D. Malone, "H-RCA: 802.11 collision-aware rate control," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1021–1034, 2012.
[7] J.-S. Kim, S.-K. Kim, and S.-H. Choi, "CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs," *J. Korean Inst. Commun. Inf. Sci.*, vol. 31, no. 2A, pp. 154–167, 2006.
[8] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *Proc. 12th Annu. Int. Conf. Mobile Comput. Netw.*, 2006, pp. 146–157.
[9] T.-Y. Lin, C.-Y. Tsai, and K.-R. Wu, "EARC: Enhanced adaptation of link rate and contention window for IEEE 802.11 multi-rate wireless networks," *IEEE Trans. Commun.*, vol. 60, no. 9, pp. 2623–2634, Sep. 2012.
[10] B.-J. Chang and S.-P. Chen, "Cross-layer-based adaptive congestion and contention controls for accessing cloud services in 5G IEEE 802.11 family wireless networks," *Comput. Commun.*, vol. 106, pp. 33–45, 2017.
[11] S. Sen, R. R. Choudhury, and S. Nelakuditi, "CSMA/CN: Carrier sense multiple access with collision notification," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 544–556, 2011.
[12] P. Kulkarni, B. Motz, T. Lewis, and S. Quadri, "Inferring loss causes to improve link rate adaptation in wireless networks," in *Proc. IEEE Int. Conf. Adv. Inf. Netw. Appl.*, 2011, pp. 659–666.
[13] Y. Zhu and Y. Sun, "Packet-level failure classification by characterizing failure patterns in wireless sensor networks," in *Proc. IEEE Glob. Commun. Conf.*, 2015, pp. 1–6.
[14] M. Wu, X. Hu, R. Zhang, and L. Yang, "Collision recognition in multi-hop IEEE 802.15. 4-compliant wireless sensor networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8542–8552, Oct. 2019.
[15] L. Zhang, W. Xiang, and X. Tang, "An adaptive anti-collision protocol for large-scale RFID tag identification," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 601–604, Dec. 2014.
[16] F. Hermans, O. Rensfelt, T. Voigt, E. Ngai, L.-Å. Nordén, and P. Gunningberg, "SoNIC: Classifying interference in 802.15. 4 sensor networks," in *Proc. 12th Int. Conf. Inf. Process. Sensor Netw.*, 2013, pp. 55–66.
[17] S. Yi et al., "Interference source identification for IEEE 802.15. 4 wireless sensor networks using deep learning," in *Proc. IEEE 29th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2018, pp. 1–7.
[18] Y. Chen, J. Yan, Y. Zhang, and K. A. Hummel, "Differentiating losses in wireless networks: A learning approach," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2022, pp. 1–2.
[19] R. Anwar, K. Nishat, M. Ali, Z. Akhtar, H. Niaz, and I. A. Qazi, "Loss differentiation: Moving onto high-speed wireless LANs," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 2463–2471.

[20] T. Huang, H. Chen, L. Cui, and S. Li, "An effective discriminator for differentiating the root causes of packet transmission failures in indoor WSNs," *Trans. Emerg. Telecommun. Technol.*, vol. 28, no. 7, 2017, Art. no. e3135.
[21] K. Tekbıyık, A. R. Ekti, A. Görçin, G. K. Kurt, and C. Keçeci, "Robust and fast automatic modulation classification with CNN under multipath fading channels," in *Proc. IEEE 91st Veh. Technol. Conf.*, 2020, pp. 1–6.
[22] A. Oliveira and T. Vazão, "Generating synthetic datasets for mobile wireless networks with sumo," in *Proc. 19th ACM Int. Symp. Mobility Manage. Wirel. Access*, 2021, pp. 33–42.
[23] J. Shi, M. Sha, and X. Peng, "Adapting wireless mesh network configuration from simulation to reality via deep learning based domain adaptation," in *Proc. 18th USENIX Symp. Networked Syst. Des. Implementation*, 2021, pp. 887–901.
[24] L. V. der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.
[25] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, Mar. 2000.
[26] M. Lacage, M. H. Manshaei, and T. Turletti, "IEEE 802.11 rate adaptation: A practical approach," in *Proc. 7th ACM Int. Symp. Model. Anal. Simul. Wirel. Mobile Syst.*, 2004, pp. 126–134.
[27] D. Xia, J. Hart, and Q. Fu, "Evaluation of the minstrel rate adaptation algorithm in IEEE 802.11 G WLANs," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 2223–2228.
[28] J. Wang et al., "3D beamforming technologies and field trials in 5G massive MIMO systems," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 362–371, 2020.
[29] R. G. Purandare, S. P. Kshirsagar, and S. M. Koli, "Loss differentiated channel aware rate adaptation for IEEE 802.11n wireless links," *Wirel. Pers. Commun.*, vol. 107, no. 4, pp. 2211–2230, 2019.
[30] S. Zhou, H. Lu, A. Mukherjee, and Z. Zhang, "Locora: Practical Wi-Fi packet loss diagnosis with only local observations," in *Proc. IEEE Wirel. Commun. Netw. Conf.*, 2018, pp. 1–6.
[31] Q. Pang, V. C. M. Leung, and S. C. Liew, "A rate adaptation algorithm for IEEE 802.11 WLANs based on MAC-layer loss differentiation," in *Proc. 2nd Int. Conf. Broadband Netw.*, 2005, pp. 659–667.
[32] I. Pefkianakis, Y. Hu, S. H. Y. Wong, H. Yang, and S. Lu, "MIMO rate adaptation in 802.11 n wireless networks," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw.*, 2010, pp. 257–268.
[33] D. Giustiniano, D. Malone, D. J. Leith, and K. Papagiannaki, "Measuring transmission opportunities in 802.11 links," *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1516–1529, 2010.
[34] Z. Guo, Z. Chen, P. Liu, J. Luo, X. Yang, and X. Sun, "Multi-agent reinforcement learning-based distributed channel access for next generation wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 5, pp. 1587–1599, May 2022.
[35] A. Kumar, G. Verma, C. Rao, A. Swami, and S. Segarra, "Adaptive contention window design using deep Q-learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 4950–4954.
[36] H. Bayat-Yeganeh, V. Shah-Mansouri, and H. Kebriaei, "A multi-state Q-learning based CSMA MAC protocol for wireless networks," *Wirel. Netw.*, vol. 24, pp. 1251–1264, 2018.
[37] S.-C. Chen, C.-Y. Li, and C.-H. Chiu, "An experience driven design for IEEE 802.11 ac Rate adaptation based on reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
[38] W. Xu, S. Guo, S. Ma, H. Zhou, M. Wu, and W. Zhuang, "Augmenting drive-thru internet via reinforcement learning-based rate adaptation," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3114–3123, Apr. 2020.
[39] S. Khastoo, T. Brecht, and A. Abedi, "NeuRA: Using neural networks to improve WiFi rate adaptation," in *Proc. 23rd Int. ACM Conf. Model. Anal. Simul. Wirel. Mobile Syst.*, 2020, pp. 161–170.
[40] C.-Y. Li, S.-C. Chen, C.-T. Kuo, and C.-H. Chiu, "Practical machine learning-based rate adaptation solution for Wi-Fi nics: IEEE 802.11 ac as a case study," *IEEE Trans. Veh. Technol*, vol. 69, no. 9, pp. 10264–10277, Sep. 2020.
[41] J. Shi and M. Sha, "Parameter self-configuration and self-adaptation in industrial wireless sensor-actuator networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 658–666.
[42] H. Gupta, J. Chen, B. Li, and R. Srikant, "Online learning-based rate selection for wireless interactive panoramic scene delivery," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1799–1808.
[43] Z. Xia, Y. Zhou, F. Y. Yan, and J. Jiang, "Automatic curriculum generation for learning adaptation in networking," 2022, *arXiv:2202.05940*.
[44] M. Phuong and C. Lampert, "Towards understanding knowledge distillation," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5142–5151.
[45] G. Ji and Z. Zhu, "Knowledge distillation in wide neural networks: Risk bound, data efficiency and imperfect teacher," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 20823–20833.
[46] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach.. Learn.*, vol. 79, no. 1–2. pp. 151–175, 2010.
[47] M. Phuong and C. Lampert, "Towards understanding knowledge distillation," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, 2019, pp. 5142–5151.

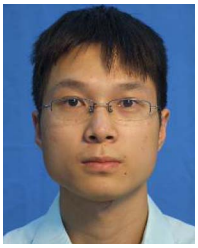
Wenchao Xu (Member, IEEE) received the BE and ME degrees from Zhejiang University, Hangzhou, China, in 2008 and 2011, respectively, and the PhD degree from the University of Waterloo, Canada, in 2018. He is a research assistant professor with the Hong Kong Polytechnic University. In 2011, he joined Alcatel Lucent Shanghai Bell Company Ltd., where he was a software engineer for telecom virtualization. He has also been an Assistant Professor with the School of Computing and Information Sciences, Caritas Institute of Higher Education, Hong Kong. His research interests includes wireless communication, Internet of Things, distributed computing, and AI enabled networking.

Haodong Wan received the ME degree from the Xidian University, Xi'an, China, in 2023. He has worked as a research assistant with the Department of Computing, The Hong Kong Polytechnic University from 2021 to 2022. His research interests include wireless networking, distributed machine learning, and federated learning.



Haozhao Wang received the bachelor's degree from the University of Electronic Science and Technology, and the PhD degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, in Dec. 2021. He is currently holds a postdoctoral position with the School of Computer Science and Technology, Huazhong University of Science and Technology. He worked as a research assistant with the Department of Computing, The Hong Kong Polytechnic University from 2018 to 2021. His research interests include distributed machine learning, federated learning, and AI security.

Nan Cheng (Member, IEEE) received the BE and MS degrees from the Department of Electronics and Information Engineering, Tongji University, Shanghai, China, in 2009 and 2012, respectively, and the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, in 2016. He worked as a Post-doctoral fellow with the Department of Electrical and Computer Engineering, University of Toronto, from 2017 to 2019. He is currently a professor with State Key Laboratory of ISN and with School of Telecommunications Engineering, Xidian University, Shaanxi, China. He has published more than 90 journal papers in IEEE Transactions and other top journals. He serves as associate editors for *IEEE Transactions on Vehicular Technology*, *IEEE Open Journal of the Communications Society*, and *Peer-to-Peer Networking and Applications*, and serves/served as guest editors for several journals. His current research focuses on B5G/6G, AI-driven future networks, and space-air-ground integrated network.



Quan Chen (Member, IEEE) received the BS, master's and PhD degrees from the School of Computer Science and Technology, Harbin Institute of Technology, China. He is currently an Associate Professor with the School of Computers, Guangdong University of Technology. In the past, he worked as a postdoctoral research fellow with the Georgia State University, USA. He is the receipt of Hong Kong Scholar award, the ACM SIGCOM CHINA Excellent Doctoral Dissertation and the CCF Excellent Doctoral Dissertation Nomination award. His research interests include IoT networks, wireless communication, and edge AI. He has served as the Technical Program Committe members for several referred conferences, and as the technical reviewer for several IEEE transactions.

Haibo Zhou (Senior Member, IEEE) received the PhD degree in information and communication engineering from Shanghai Jiao Tong University, Shanghai, China, in 2014. From 2014 to 2017, he was a postdoctoral fellow with the Broadband Communications Research Group, Department of Electrical and Computer Engineering, University of Waterloo. He is currently a full professor with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China. He was a recipient of the 2019 IEEE ComSoc Asia–Pacific Outstanding Young Researcher Award, 2023-2024 IEEE ComSoc Distinguished Lecturer, and 2023-2025 IEEE VTS Distinguished Lecturer. He served as Track/Symposium Co-Chair for IEEE/CIC ICCC 2019, IEEE VTC-Fall 2020, IEEE VTC-Fall 2021, WCSP 2022, IEEE GLOBECOM 2022, IEEE ICC 2024. He is currently an associate editor of the *IEEE Transactions on Wireless Communications*, *IEEE Internet of Things Journal*, *IEEE Network Magazine*, and *Journal of Communications and Information Networks*. His research interests include resource management and protocol design in B5G/6G networks, vehicular ad hoc networks, and space-air-ground integrated networks.

Song Guo (Fellow, IEEE) is a full professor with the Department of Computing, The Hong Kong Polytechnic University. He also holds a Changjiang chair professorship awarded by the Ministry of Education of China. He is a fellow of the Canadian Academy of Engineering. His research interests include mainly in big data, edge AI, mobile computing, and distributed systems. He published many papers in top venues with wide impact in these areas and was recognized as a Highly Cited Researcher (Clarivate Web of Science). He is the recipient of over a dozen Best Paper Awards from IEEE/ACM conferences, journals, and technical committees. Prof. Guo is the editor-in-chief of *IEEE Open Journal of the Computer Society* and the chair of IEEE Communications Society (ComSoc) Space and Satellite Communications Technical Committee. He was an IEEE ComSoc distinguished lecturer and a member of IEEE ComSoc Board of Governors. He has served for IEEE Computer Society on Fellow Evaluation Committee, and been named on editorial board of a number of prestigious international journals like *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Emerging Topics in Computing*, etc. He has also served as chairs of organizing and technical committees of many international conferences.