BeaconRider: Opportunistic Sharing of Beacon Air-Time in Densely Deployed WLANs

Hyunjoong Lee†, Jungjun Kim†, Changhee Joo‡, and Saewoong Bahk†
†Department of Electrical and Computer Engineering and INMC, Seoul National University, Seoul, South Korea
‡Department of Computer Science and Engineering, Korea University, Seoul, South Korea
Email: {hjlee, jjkim}@netlab.snu.ac.kr, changheejoo@gmail.com, sbahk@snu.ac.kr

***Abstract*—The explosion of mobile traffic volume has led to dense deployment of IEEE 802.11 WLANs. As a consequence, periodic beacon transmissions can overwhelm the air-time, leading to significant air-time depletion for data transmissions. In this work, we develop an opportunistic air-time sharing scheme, named *BeaconRider*, that facilitates simultaneous data and beacon transmissions aimed at improving spectrum efficiency in dense network environments. The proposed method works for downlink communication and allows access points (APs) to coordinate with each other in a distributed manner to exploit opportunities provided by the capture effect. Our protocol is backward compatible with legacy 802.11 APs. Through experiments with a prototype implementation using off-the-shelf IEEE 802.11n dongles as well as extensive ns-3 simulation, we show that the proposed method achieves substantial performance gains that increase with the number of APs.**

***Index Terms*—beacon frame, air-time sharing, capture effect, interference mitigation, high-density WLANs**

I. Introduction

The proliferation of mobile devices has resulted in explosive traffic growth which is expected to reach 49 exabytes per month globally [1]. As cellular spectra and network resources become taxed, wireless local area networks (WLANs) have been utilized to offload mobile traffic [2], [3]. It is reported that 60% of global mobile data traffic has been offloaded to WLANs in 2016 [1].

The upsurge in IEEE 802.11 WLAN traffic has led to dense deployment of access points (APs) in urban areas including office buildings, residential housing, shopping malls, public spaces, among many others. As density of APs increases, the number of APs that a wireless station (STA) can access increases, leading to interference between nearby APs. As a consequence, interference induced by management frames (e.g., beacon frame) also increases, which can significantly degrade system performance.

Changhee Joo is the corresponding author.

This work was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT and Future Planning (MSIT) (No. 2017R1E1A1A01074358), in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-11-1864, Scalable Spectrum Sharing for Beyond 5G Communication), and in part by the NRF grant funded by the Korea government (MSIT) (No. NRF-2017R1E1A1A03070524).

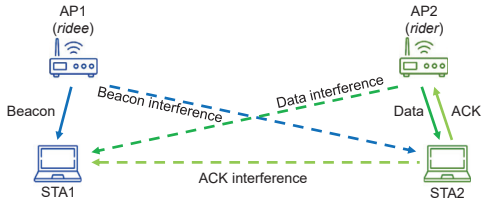


Fig. 1. Example of simultaneous transmission of beacon and data frames.

In this paper, we focus on the problem of air-time depletion due to beacon transmissions which are essential for the operation of a basic service set (BSS)[1]. Beacon frames are transmitted at the lowest basic transmission rate to ensure reliable delivery to STAs associated with a BSS. This implies that each beacon frame occupies considerable air-time despite its short length. If a STA is within the range of multiple APs, it will be exposed to a large number of beacon frames which can take up a significant portion of its air-time. Our experiments indicate that up to 43.9% can be taken up by beacon frames, which leads to significant reduction of air-time for data transmissions.

To address the air-time depletion problem, we introduce a new approach of air-time sharing that allows simultaneous data transmission during the air-time of beacon transmission of neighboring APs. Fig. 1 depicts an example of two pairs of AP and STA, where one of the APs (AP2) initiates data transmission while the other (AP1) is transmitting a beacon frame. The dotted lines indicate potential interference across the two BSSs that cause frame decoding failure at the receiving STA of the other BSS. However, the interference does not always lead to the decoding failure thanks to the capture effect[2] [4]–[6]. Since a beacon frame is transmitted at the lowest transmission rate, the beacon decoding at STA1 may succeed if data/ACK transmission interference from AP2/STA2 is not sufficiently strong. In addition, whether the capture effect leads to successful decoding of the data frame at STA2 will depend on its transmission rate and strength of beacon interference.

[1]A BSS consists of an AP and multiple associated STAs.

[2]When two nodes simultaneously transmit different frames toward the same receiver on the same channel, the both frames may be garbled due to mutual interference. In practice, however, it is often observed that one frame with considerably higher signal strength is successfully decoded at the receiver despite the interference. This phenomenon is named *capture effect*.

Our goal is to exploit the signal diversity of wireless communication by opportunistically engaging in simultaneous data transmission in the presence of beacon transmission. To this end, we develop a distributed protocol that explicitly exploits the capture effect based on real-world measurement characteristics and coordinates air-time sharing in an efficient and reliable manner. The protocol selects two appropriate APs such that the beacon frame is sufficiently protected from the interference of the data-ACK transmissions, and at the same time, data-ACK transmissions are likely to be successful under beacon interference.

Our main contributions are as follows.

- Based on beacon frame traces in real-world environments, we analyze the characteristics of beacon frames and their impact on system performance.
- We develop an opportunistic air-time sharing protocol, *BeaconRider*, that allows simultaneous data transmission during the beacon period of neighboring APs. The proposed method provides distributed coordination between APs without incurring additional control message exchange.
- We implement the protocol on an off-the-shelf IEEE 802.11n dongle using the *ath9k-htc* open-source firmware [7] which operates as an AP. The modified APs coexist with legacy APs and do not require any changes at STAs. These properties facilitate partial deployment of *BeaconRider*.
- Through experiments with a prototype implementation and extensive simulation using ns-3 [8] under a variety of network environments, we demonstrate that *BeaconRider* significantly improves throughput performance.

To the best of our knowledge, this is the first work that shares beacon air-time for data transmission by exploiting the capture effect.

The rest of this paper is organized as follows. We summarize background and related work in Section II. Section III describes motivation of our work, and Section IV presents a detailed design of *BeaconRider*. In Section V, based on measurements and analysis, we gauge reliability of beacon frames under riding data interference. We evaluate the performance of *BeaconRider* via experiment and simulation in Section VI. Finally, we conclude the paper in Section VII.

II. Background and Related work

A. Conditions for the Capture Effect

To trigger the capture effect and successfully receive a certain frame (say target frame), high signal-to-interference-ratio (SIR) is required at the receiver. The transmission result depends on the selected modulation and coding scheme (MCS) level of the target frame as well as the arrival order of the two frames. Let the sender denote the AP that transmits the target frame. According to the arrival order of the target frame, the conditions for the capture effect are given as follows [9].

- *Sender-first capture*: In this case, the target frame arrives first and the second frame acts as interference. For the target frame of MCS level x to be captured at the receiver, it requires that the SIR should be greater than threshold θ_x. It is known that θ_x increases with x.
- *Sender-last capture*: In this case, the target frame arrives later. The receiver has already started decoding the first frame when the target frame arrives. It can detect the target frame only when the SIR is greater than ψ. This value is often denoted as *capture threshold* and known to be 10 dB for *ath9k* chipsets [9]. After the receiver detects the target frame, it is necessary for the SIR to be greater than θ_x again for successful decoding of data transmitted with MCS level x. As a result, we have the SIR threshold for the sender-last capture of MCS level x as $\max\{\theta_x, \psi\}$.

B. Related Work

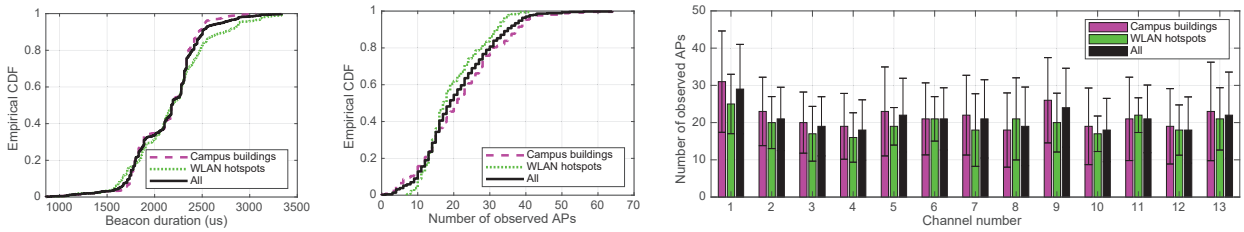
We summarize previous works in beacon congestion mitigation and air-time sharing relevant to *BeaconRider*.

1) Beacon congestion mitigation: There have been many efforts that aim to address the air-time depletion problem due to excessive beacon frames. A direct approach is to adjust the beacon interval to reduce beacon congestion [10]–[12]. In [10], the authors propose using double hundred beacon interval (2HKBI) for opportunistic device-to-device communication to reduce energy consumption and bandwidth overhead. For wireless HTTP clients, a dynamic beacon interval adaptation method based on round-trip time (RTT) is proposed [11]. In [12], an AP adjusts the beacon interval according to client traffic. However, an enlarged beacon interval induces longer delay for power save mode (PSM) enabled devices, thus it is not suitable for latency-sensitive traffic or real-time applications such as VoIP applications.

Instead of changing the beacon interval, air-time for beacon transmissions can be reduced by increasing transmission rate [13], [14]. A beacon congestion control algorithm that adjusts beacon transmission rate for vehicular ad-hoc networks is proposed in [13]. A beacon transmission rate control algorithm for cooperative vehicular networks is advanced in [14]. Changing beacon transmission rate is not backward compatible, a key feature of *BeaconRider*.

A third class of methods aims to mitigate interference from beacon transmissions through power management [15]–[17]. In [15], a method for adjusting beacon transmission power based on traffic forecast is proposed. Beacon transmission power control that reduces collision rate under high beacon load is advanced in [16], [17]. Both beacon transmission rate and power control affect the transmission range of APs. Time-varying coverage may also lead to frequent deauthentication of boundary STAs. Beacon interval, transmission rate, and power adjustment are control dimensions complementary to our approach of exploiting capture effect for simultaneous data and beacon transmissions.

2) Air-time sharing: Several studies have targeted spectrum efficiency by allowing simultaneous transmission through scheduled air-time [5], [18], [19]. In [5], the authors propose a MAC frame scheduler called *Shuffle* that coordinates frame



(a) Distribution of beacon air-time duration

(b) Distribution of the number of observed APs per channel

(c) The number of observed APs at each channel in 2.4 GHz band

Fig. 2. Statistics of beacon frames in practical WLANs. Total 3,218 beacon frames are collected from 20 campus buildings and 14 commercial WLAN hotspots.

TABLE I
SUMMARY OF BEACON FRAME STATISTICS

Parameters		Campus buildings	WLAN hotspots	All
Air-time duration of one beacon frame (μs)	Min.	856	864	856
	Max.	3,064	3,336	3,336
	Avg.	2,120	2,182	2,140
The number of observed APs per channel	Min.	0	7	0
	Max.	64	41	64
	Avg.	22	20	21

transmission orders and supports simultaneous transmission from multiple APs. In [18], a fine-grained network coordination called *CENTAUR* that centrally schedules hidden and exposed terminals is presented. Both *Shuffle* and *CENTAUR* require a centralized controller to maintain precise time synchronization across multiple APs. From a deployability perspective, these approaches are limited in WLANs that are part of a single administrative domain. In [19], the authors propose a distributed protocol, *CMAP*, that constructs a conflict map and supports simultaneous transmission from multiple exposed terminals. This requires modifications at the MAC frame structure that does not warrant backward compatibility.

There have been efforts to recover collided frames after simultaneous transmission by using data coding techniques [20]–[22]. Although these techniques can provide significant throughput improvement in interference prone WLANs, they require changes at both AP and STAs, and do not provide backward compatibility. Our technique can coexist with most of these coding techniques, resulting in further performance improvement.

Air-time sharing in heterogeneous Bluetooth and WLAN has been studied in [23], [24]. The authors in [23] and [24] have shown that a dual-stack device with Bluetooth and WLAN modules can opportunistically improve spectral efficiency of one protocol using the other's unused periods. Our approach falls in the air-time sharing category that exploits capture effect to facilitate simultaneous beacon and data transmissions in a backward compatible manner without requiring centralized control.

III. BEACON CONGESTION PROBLEM

In IEEE 802.11 WLANs, APs periodically transmit beacon frames to announce their network information. In this paper, we consider an infrastructure BSS where only APs transmit beacon frames. Beacon frames are typically transmitted at an interval of 102.4 ms [2], at the lowest basic (mandatory) transmission rate for reliability, e.g., at 1 Mbps in 2.4 GHz band and at 6 Mbps in 5 GHz band. As a consequence, beacon frames often take up more air-time than data frames despite their short length. As the density of BSSs increases, a large portion of air-time is consumed by beacon frames, which is referred to as the *beacon congestion problem*.

For advanced network services, vendors may tag additional information onto beacon frames as optional fields, which aggravates the beacon congestion problem. The problem is more severe in the 2.4 GHz band due to lower beacon transmission rate and smaller number of available channels. Moreover, a large portion of WLAN devices still support 2.4 GHz band only, and dual-band devices that support both 2.4 GHz and 5 GHz bands may prefer 2.4 GHz band due to propagation advantages of the 2.4 GHz band [25]–[27]. In this paper, we focus on mitigating the beacon congestion problem in the 2.4 GHz band since it is more pronounced.

To quantify the impact of beacon congestion, we used *Wireshark* [28] to collect beacon frame traces (total 3,218 distinct APs) in several real-world WLAN networks across campus buildings, commercial WLAN hotspots at shopping malls, subway stations, coffee shops, and movie theaters. The trace results are shown in Fig. 2. The distribution of beacon air-time duration in Fig. 2(a) shows the impact of a single beacon frame on air-time depletion. The minimum, maximum, and average of the beacon air-time duration are summarized in Table I. Considering the default beacon interval of 102.4 ms, average duration of a single beacon frame accounts for about 2.1% of total air-time. When coverage spans multiple APs, beacon congestion deteriorates depending on the number of APs observed by a STA.

The distribution of the number of observed APs in a single channel is shown in Fig. 2(b), and the values are summarized in Table I. A single STA observes 21 beacon frames on average in a typical beacon interval of 102.4 ms, where the average duration of a single beacon frame is 2,140 μs. This results in 43.9% air-time consumption from beacon sensing, which significantly limits air-time available for data transmissions.

Fig. 2(c) shows the number of observed APs at each channel in the 2.4 GHz band. Although the number of observed APs

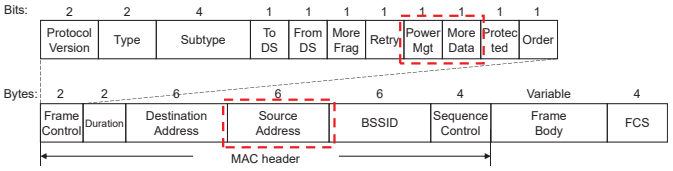


Fig. 3. Frame structure of the IEEE 802.11 beacon frame. Marked fields are of active use under *BeaconRider* for the coordination between APs.

in non-overlapped channels (i.e., channels 1, 5, 9, and 13) is larger than that in other channels, a significant number of APs are also observed in other channels as well. This shows the beacon congestion problem is not limited to the non-overlapped channels. In this sense, we observed that beacon transmission in one channel can be detected in partially overlapped channels [29]. This further exacerbates beacon congestion.

IV. Design of *BeaconRider*

In this section, we propose *BeaconRider*, an air-time sharing scheme that allows simultaneous data transmission during the air-time of beacon transmission, by explicitly exploiting the capture effect. We first provide the overview of *BeaconRider*, and explain its operation in detail.

A. Overview of BeaconRider

We implement *BeaconRider* at the AP side, and denote the AP with our scheme by *BeaconRider* AP, or simply BAP. When we use the term AP, it is either a BAP or a legacy AP. In Fig. 1, we denote the left AP transmitting the beacon frame by *ridee* and the right AP transmitting data by *rider*. In our design, we let the ridee select the rider AP to transmit data frames while transmitting beacon frames.

Successful riding and beacon reception need coordination between BAPs. To avoid additional control message exchange for the coordination, we redefine three fields of the IEEE 802.11 beacon frame: *power management (PM)*, *more data (MD)* subfields, and *source address (SA)* field, marked in Fig. 3. Note that the PM and MD subfields are reserved (i.e., not used), and the SA field is a duplicate of the BSSID field[3]. We use the PM and MD subfields as the BAP indicator and the buffer status indicator, respectively. The BAP sets the PM bit for each beacon frame, and sets the MD bit when it has data traffic to transmit. The SA field plays a key role in coordination. The ridee uses this field to grant riding to the rider. We explain the overall operation through the following three actions.

1) **Listening:** A BAP listens to beacon frames from neighboring (B)APs, and collects information needed to determine whether it becomes a ridee or not. Using the information, it selects one of neighboring BAPs as its corresponding (target) rider. The selection may change over time according to the channel state.
2) **Informing rider selection:** When transmitting beacon frames, each BAP (as a ridee) informs its neighboring APs of the latest decision on rider selection through using the SA field of the beacon frame.
3) **Riding:** While receiving a beacon frame, the BAP quickly checks the SA field in the beacon frame. If the receiving BAP is indicated as the target rider of the beacon frame, it immediately stops receiving the beacon frame and starts transmitting a data frame, thus attempting to ride the beacon frame.

[3]According to the 802.11 standard [2], the PM subfield is used to indicate the PSM status of a STA, and the MD subfield is valid only in individually addressed data frame or management frame. For the SA field, the implementation cases that we tested, e.g., *ath9k* and *android* [30], use only the BSSID field of the received beacon frame to identify the beacon transmitter and does not refer to the SA field. There may be some legacy implementations that access the SA field for this purpose, which may need further investigation.

B. Operation of Ridee

The important task of the ridee is to select the right riding BAP, enabled to share its beacon air-time. The transmissions of beacon and riding data-ACK frames should not interfere with each other. Moreover, AP coordination should proceed without a central entity or additional control message exchange, conforming to the distributed nature of the WLAN protocol and backward compatibility.

In our scheme, the BAP continues listening to beacon frames from its neighboring APs, and collecting the information of beacon length, received signal strength (RSS), BSSID, buffer status (i.e., MD bit), and BAP indication (i.e., PM bit). If the beacon transmitter is a BAP, it computes average RSS through exponential weighted moving average (EWMA). By default, we set the weight factor of EWMA to 0.125.

At every beacon transmission, the ridee selects the BAP with the lowest average RSS as the rider, among the list of neighboring BAPs with non-empty buffer. The ridee informs neighboring BAPs of the decision on rider selection by setting the SA field of the beacon frame to the MAC address of the target rider.

C. Operation of Rider

Upon receiving a beacon frame, the BAP begins to decode its SA field. If the MAC address in the SA field matches with its own MAC address, the BAP operates as the rider of the beacon frame, i.e., immediately stops receiving the beacon frame and starts to transmit a data frame.

Note that the beacon reception is protected from the interference of data transmission through the sender-first capture: the beacon frame transmitted at the lowest MCS level (i.e., 1 Mbps) always precedes a riding data frame. In this scenario, the capture effect is easily triggered since a low SIR threshold (-10 dB as shown in Section V) is required and the RSS from the selected rider is the lowest among neighboring BAPs.

On the other hand, the beacon reception may not be protected from the interference of ACK transmission if the riding data receiver is close to the beacon receiver. Also, it is possible that the riding data transmission fails due to the beacon interference. To address these problems, we carefully choose the data frame length, the MCS level, and the final decision on a riding attempt at the rider.

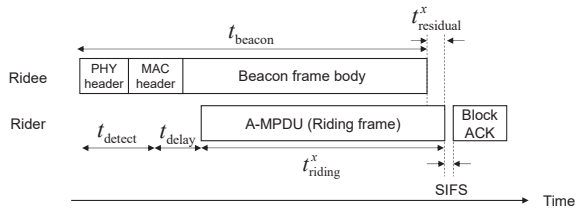


Fig. 4. Timeline example of riding data-ACK transmissions.

1) Selection of the data frame length for riding: Unlike the interference from riding data transmission, the interference due to ACK transmission paired with riding data transmission can be significant and disrupt the beacon reception. The interference level highly depends on the location of the receiver. To avoid such uncertainties, *BeaconRider* controls the length of the riding data frame by aggregating several subframes such that the transmission of aggregated riding data frame lasts longer than the beacon transmission, which ensures no transmission overlap between the beacon frame and the block ACK (BlockACK) frame paired with the riding data frame[4].

An example is shown in Fig. 4. Let t_{detect} and t_{delay} denote the times for the rider to detect the beacon frame and to prepare for riding, respectively, and t_{beacon} be the beacon duration[5]. For given MCS level x for riding transmission, we easily obtain the smallest number of subframes such that the riding data transmission is not completed earlier than the beacon transmission (i.e., $t_{\text{detect}} + t_{\text{delay}} + t^x_{\text{riding}} \geq t_{\text{beacon}}$), and set it as the length of the riding data frame.

2) Selection of the MCS level for riding: Another important decision that the rider BAP should make is about the MCS level for data frame transmission. From the viewpoint of the receiver of the riding data frame, on-going beacon transmission appears as interference. As a result, the rider may need to lower its MCS level than usual.

To find the right MCS level, we define a utility function for each MCS level using subframe error rate (SFER). To elaborate, suppose that the MCS level for a (non-riding) data transmission is m. The level is usually determined by a built-in rate adaptation algorithm according to the channel condition of the receiver, given by signal-to-noise-ratio (SNR) or frame error rate (FER). When MCS level x is used for riding when m is given, after a riding attempt, the rider BAP counts the number of non-acknowledged[6] subframes in the corresponding BlockACK and computes average SFER $\tilde{e}^i_{m,x}$ for ridee i through EWMA with weight factor of 0.125. The utility $U^i_{m,x}$ of MCS x given m is updated as

$$U^i_{m,x} = r_x \cdot \left(1 - \tilde{e}^i_{m,x}\right) \cdot t^x_{\text{riding}} - r_m \cdot t^x_{\text{residual}}, \qquad (1)$$

where r_x denotes the transmission rate under MCS x, and t^x_{riding} and t^x_{residual} denote the riding duration and the residual time under MCS x, respectively, as shown in Fig. 4. Roughly, the utility represents the effective gain of riding transmission under MCS x over a (non-riding) data frame transmission under MCS m. The first term can be considered as a gain of air-time sharing and the second term as a loss in the case of reception failure. In the next riding, the rider chooses the MCS level x^* with the highest utility, i.e., $x^* = \arg\max_x U^i_{m,x}$. We break a tie arbitrarily.

Under the disturbance in signal strength (e.g., fading), it may occur that our MCS selection remains at low levels since SFER is only updated after a riding attempt. To avoid this, we conditionally explore the next higher MCS level if the same MCS level has been used multiple times in a row. Let $p^i_{\text{win}}\ (> 0)$ and $p^i_{\text{cnt}}\ (\geq 0)$ as the window size of exploration back-off and the remaining number of riding trials before the rider explores the next higher MCS level for ridee i, respectively. Both are set to 1 initially. When the BAP tries a riding transmission with the same MCS (m, x) as the previous riding, it decreases p^i_{cnt} by one. Otherwise, it sets $p^i_{\text{cnt}} = p^i_{\text{win}}$. If p^i_{cnt} reaches 0, the BAP explores the next higher MCS level at the next riding. If the exploration is successful, the BAP halves p^i_{win} and sets $p^i_{\text{cnt}} = p^i_{\text{win}}$. Otherwise, it doubles p^i_{win} and sets $p^i_{\text{cnt}} = p^i_{\text{win}}$.

3) Adaptive riding (AR): Despite the utility-based selection of the MCS level, success of a riding transmission highly depends on the interference from the ridee's beacon transmission. If the riding transmission fails, we lose t^x_{residual} amount of air-time. Hence, in case that the interference from the beacon transmission is significant, it may be better not to attempt riding. To this end, we develop a simple adaptive riding-attempt (AR) algorithm. We define two parameters $g^i_{\text{win}}\ (> 0)$ and $g^i_{\text{cnt}}\ (\geq 0)$ as the window size of riding back-off and the remaining number of riding back-offs before the rider resumes riding for ridee i, respectively. We initially set g^i_{win} and g^i_{cnt} as 1 and 0, respectively. When the rider has $U^i_{m,x} < 0$ for all x due to high SFER, it sets $g^i_{\text{cnt}} = g^i_{\text{win}}$ and stops riding. Upon receiving a beacon frame from ridee i, the rider decreases g^i_{cnt} by one. When g^i_{cnt} reaches 0, the rider resumes riding with the lowest MCS level. If the riding attempt fails again, the rider doubles g^i_{win}, sets $g^i_{\text{cnt}} = g^i_{\text{win}}$, and repeats the g^i_{cnt}-decreasing procedure. On any riding attempt success, the rider halves g^i_{win} (down to 1) and sets $g^i_{\text{cnt}} = 0$.

V. Reliability of Beacon Transmission

An interesting feature of *BeaconRider* is that riding permission is granted by the ridee. Specifically, if the ridee suffers difficulty in beacon frame delivery, it easily stops allowing riding by filling the SA field with its own MAC address. Nevertheless, it is important to estimate the impact of riding on the beacon reception. In this section, we investigate reliability

[4]Recent WLAN standards (IEEE 802.11n/ac) [2], [3] support *frame aggregation*, under which multiple MAC protocol data unit (MPDU) frames are aggregated into a single aggregate MPDU (A-MPDU) frame, and the receiver positively/negatively acknowledges all individual subframes (i.e., MPDUs) by using a BlockACK, which allows selective retransmission of subframes.

[5]t_{detect} and t_{delay} can be measured online by using a local time function in the firmware in our implementation, which are around 400 μs and 100 μs, respectively. We expect that these delays decrease further in recent devices. Finally, t_{beacon} can be easily obtained from the beacon length.

[6]If the BlockACK does not arrive, we consider that all the subframes are not acknowledged.

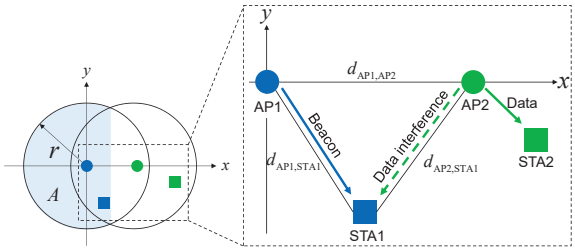


Fig. 5. Example of two BSSs. Each STA is associated with the closest AP in terms of received signal strength (RSS), and the shaded area (A) is the coverage of AP1.

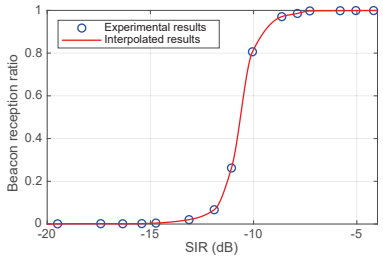


Fig. 6. Measurements on the beacon reception ratio (BRR) according to the SIR value. Beacon frames of 280 bytes are transmitted at rate 1 Mbps. We averaged 5 measurements, and each iteration lasts for 3 minutes. The red line is obtained by interpolating experimental results.

of the beacon reception under riding scenarios. We first model the beacon frame error rate (BFER) according to the location of the STA. Then, using the measured beacon reception ratio (BRR), we numerically calculate the expected BFER within the coverage of an AP. Lastly, we propose a simple algorithm to detect a beacon loss at STAs to eliminate corner cases.

A. Beacon Frame Error Rate (BFER)

We consider two BSSs as shown in Fig. 5, where AP1 (ridee) transmits a beacon frame to STA1, and AP2 (rider) transmits a riding data frame to STA2. We denote $d_{i,j}$ the distance between node i and node j. Without loss of generality, AP1 and AP2 are located at the origin (0,0) and ($d_{\mathrm{AP1,AP2}}$, 0), respectively. Suppose that STAs are randomly located in the space, and associated with the closest AP in terms of RSS. A STA communicates with an AP if they are within distance r.

We model the wireless channel gain between a STA and an AP according to their distance. Let $L_{i,j}$ denote the channel gain from node i to node j, where a node is either an AP or a STA. The channel gain is given as

$$L_{i,j} = \Gamma_{i,j} \cdot L_0 \cdot d_{i,j}^{-\alpha}, \tag{2}$$

where α and L_0 denote the pathloss exponent and a fixed loss, respectively, and $\Gamma_{i,j} \sim \mathrm{Exp}(1)$ is a Rayleigh fast fading component with a unit average power. Assuming mutually independent $\Gamma_{i,j}$ and the channel reciprocity for any pair of nodes, the signal power from AP1 to STA1 can be written as $S_{\mathrm{AP1}\to\mathrm{STA1}} = P_{\mathrm{AP1}} \cdot G_{\mathrm{AP1}} \cdot G_{\mathrm{STA1}} \cdot L_{\mathrm{AP1,STA1}}$, where P_i and G_i denote the transmission power and the antenna gain at node i, respectively. The data interference from AP2 to STA1 can be similarly obtained as $I_{\mathrm{AP2}\to\mathrm{STA1}} = P_{\mathrm{AP2}} \cdot G_{\mathrm{AP2}} \cdot G_{\mathrm{STA1}} \cdot L_{\mathrm{AP2,STA1}}$.

Assuming the noise is negligible compared to the interference and $P_{\mathrm{AP1}} \cdot G_{\mathrm{AP1}} = P_{\mathrm{AP2}} \cdot G_{\mathrm{AP2}}$, the SIR at STA1 can be written as

$$\mathrm{SIR}_{\mathrm{STA1}} = \frac{S_{\mathrm{AP1}\to\mathrm{STA1}}}{I_{\mathrm{AP2}\to\mathrm{STA1}}} = \frac{\Gamma_{\mathrm{AP1,STA1}}}{\Gamma_{\mathrm{AP2,STA1}}} \left(\frac{d_{\mathrm{AP1,STA1}}}{d_{\mathrm{AP2,STA1}}}\right)^{-\alpha}. \tag{3}$$

Note that the probability density function (PDF) of a random variable Z, defined as the ratio of two independent and identically distributed (i.i.d.) exponential random variables, is known as $f_Z(z) = \Pr(Z = z) = \frac{1}{(z+1)^2}$ [31]. Since $\Gamma_{\mathrm{AP1,STA1}}$ and $\Gamma_{\mathrm{AP2,STA1}}$ are i.i.d. exponential random variables with a unit variance, we obtain the PDF of the SIR which depends on $\gamma = \left(\frac{d_{\mathrm{AP1,STA1}}}{d_{\mathrm{AP2,STA1}}}\right)^{-\alpha}$ as

$$f_\gamma(\rho) = \frac{1}{\gamma} \cdot \frac{1}{\left(\frac{\rho}{\gamma} + 1\right)^2}, \tag{4}$$

where ρ represents the SIR value. Given the STA's location (x, y), we obtain the expected BFER as

$$\overline{\mathrm{BFER}}(x, y) = \int_0^\infty \mathrm{BFER}(\rho) \cdot f_{\gamma_{xy}}(\rho) d\rho, \tag{5}$$

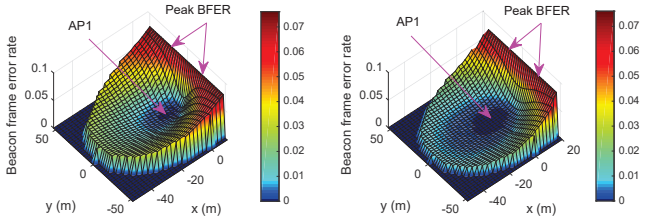
where $\mathrm{BFER}(\rho)$ is the BFER for given ρ.

B. Measurement-based Calculation of Beacon Reception Ratio (BRR)

There have been several studies to model the FER, but they did not take the sender-first capture into consideration, thus leading to an overestimation of BFER in our riding scenario [32], [33]. To design a precise model of $\mathrm{BFER}(\rho)$, we conduct experiments in our testbed as follows.

We deploy a couple of AP-STA pairs as shown in Fig. 5. We set the distance between AP1 and AP2 as 5 m. While AP1 transmits beacon frames periodically, AP2 generates saturated UDP downlink traffic toward STA2 for 3 minutes. Whenever AP2 detects a beacon frame of AP1, it immediately transmits a riding data frame. In the experiments, we count the number of successfully received beacon frames at STA1 and the number of total transmitted beacon frames from AP1, and take their ratio to obtain the BRR. We repeat the measurements under different SIR configurations at STA1 by changing the transmission power of AP1 from 0 to 15 dBm, while fixing that of AP2 to 20 dBm.

Fig. 6 shows the BRR of STA1 according to the SIR value. The BRR sharply rises from zero to one around the SIR of -11 dB. This is because, below the threshold, the sender-last capture occurs, and STA1 receives a data frame from AP2, not a beacon frame from AP1. From the results, we observe that the required SIR threshold to capture the beacon frame is very small (around -10 dB), which implies that the beacon frame can be easily captured when the STA associates with the AP of the strongest signal strength. From $\mathrm{BFER}(\rho) = 1 - \mathrm{BRR}(\rho)$, we use the measurement results to compute $\overline{\mathrm{BFER}}(x, y)$ in (5).



(a) r = 50, $d_{\mathrm{AP1,AP2}}$ = 20 (b) r = 50, $d_{\mathrm{AP1,AP2}}$ = 40

Fig. 7. Expected beacon frame error rate (BFER) according to the STA's location (x, y). The peak BFER (7.6%) is observed at line $x = d_{\mathrm{AP1,AP2}}/2$.

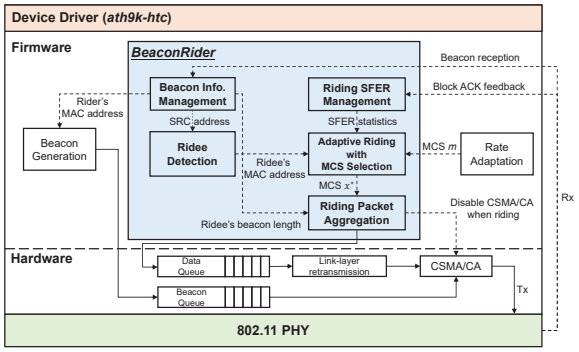


Fig. 8. Block diagram of *BeaconRider*.

C. Numerical Results

We numerically obtain $\overline{\mathrm{BFER}}(x, y)$ for different values r and $d_{\mathrm{AP1,AP2}}$, which is shown in Fig. 7. As expected, $\overline{\mathrm{BFER}}(x, y)$ decreases as STA1 moves toward AP1 (or away from AP2). We can observe the peak BFER (7.6%) at line $x = d_{\mathrm{AP1,AP2}}/2$, which is the same regardless of $(r, d_{\mathrm{AP1,AP2}})$. By integrating $\overline{\mathrm{BFER}}(x, y)$ over the coverage of AP1, i.e., the shaded area (A) in Fig. 5, we obtain the average BFER of a randomly deployed STA as

$$\overline{\mathrm{BFER}}_{\mathrm{AP1}} = \int_A \overline{\mathrm{BFER}}(x, y) dx dy. \quad (6)$$

In our results, we observe that $\overline{\mathrm{BFER}}_{\mathrm{AP1}}$ decreases from 3.0% to 1.7% as the ratio $\frac{r}{d_{\mathrm{AP1,AP2}}}$ decreases. This is due to reduced interference from riding data transmission.

In summary, our measurements show that beacon frames are rarely lost under the interference from riding data transmission due to the sender-first capture, and well protected by selecting the rider with the least RSS.

D. Beacon Loss Detection

Although beacon frames are well protected under our rider selection and riding data frame length selection algorithms, it is still important to detect a beacon loss to eliminate corner cases. Unfortunately, there is no explicit way at the AP side to detect a beacon loss at STAs since the beacon does not require any response (i.e., ACK frame). Instead, at the STA side, if an STA misses k beacons in a row ($k = 10$ in *ath9k* driver and ns-3 simulator by default), it considers the connection is lost and initializes the reassociation procedure by sending a probe request frame. Hence, the AP can use the probe request frame from an STA to detect a beacon loss.

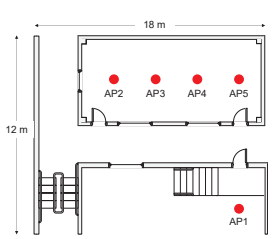


Fig. 9. Floor plan.

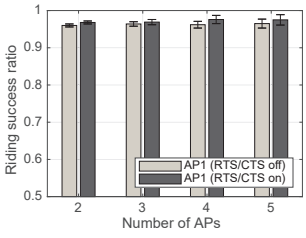
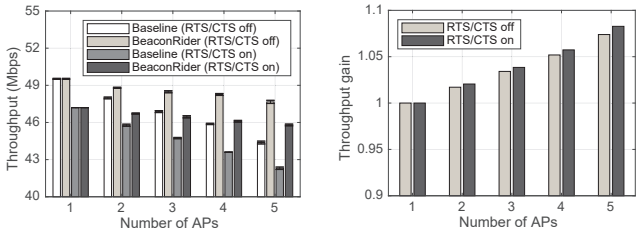


Fig. 10. Riding success ratio (RSR) for a single-source scenario.



(a) Throughput (b) Throughput gain

Fig. 11. Throughput and throughput gain for a single-source scenario.

VI. Performance Evaluation

In this section, we evaluate the performance of *BeaconRider* through prototype implementations and experiments. We present the results of extensive simulations with ns-3 for its performance in a wide range of network environments.

A. Prototype Implementation

We have implemented the prototype of *BeaconRider* over an off-the-shelf IEEE 802.11n dongle (TP-Link WN722N v1) with Qualcomm Atheros AR9271 and the open-source firmware of *ath9k-htc*. The source code is available at [34]. The block diagram of *BeaconRider* is illustrated in Fig. 8.

On receiving a beacon frame, the *beacon information management* block collects necessary information from the beacon frame's MAC header. To perform real-time monitoring of the receive buffer, we use direct memory access (DMA) function as in [35]. Using the information, the *ridee detection* block triggers riding transmission, and then the *adaptive riding with MCS selection* block makes a riding decision on an appropriate MCS level, according to SFER statistics. Once the decision is made, a riding data frame is prepared in the *riding packet aggregation* block, put in the data queue, and immediately transmitted by disabling CSMA/CA.

Experimental settings: We first conduct basic experiments in a controlled office environment as shown in Fig. 9. Each point represents an AP, and one STA per AP is placed within 1 m distance from its associated AP[7]. For each AP, we use *hostapd* 2.6 [36] on ubuntu 14.04 over a laptop (Samsung NT531U4E) with an IEEE 802.11n dongle. As a STA, we use a desktop PC with Qualcomm Atheros AR9380 Network Interface Card (NIC). Otherwise specified, we use

[7]We consider BSSs where only one STA is associated with an AP for experimental simplicity. Nevertheless, our scheme can simply support multiple STAs by managing SFER statistics for each ridee and receiver STA pair independently.

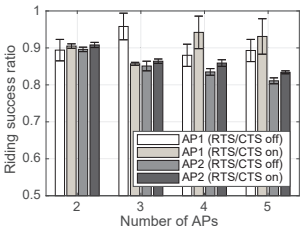


Fig. 12. Riding success ratio (RSR) for a multi-source scenario.

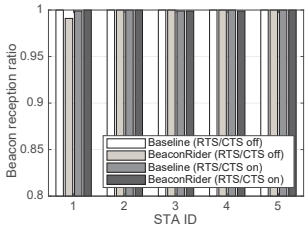
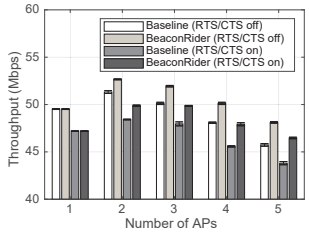
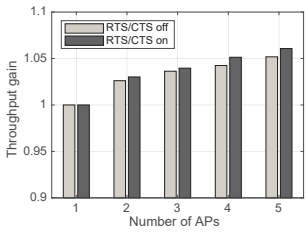


Fig. 13. Beacon reception ratio (BRR) with 5 APs (with data traffic).



(a) Throughput



(b) Throughput gain

Fig. 14. Throughput and throughput gain for a multi-source scenario.

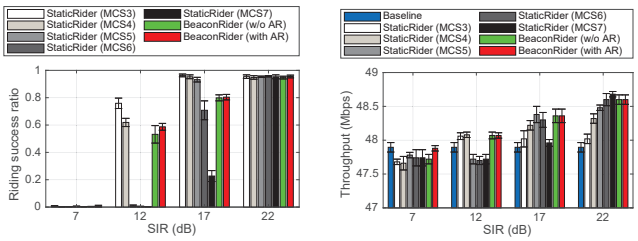
channel 10 centered at 2.457 GHz (after checking no external interference), and set the beacon frame length to 280 bytes.

When an AP has data traffic for its associated STA, we generate UDP traffic using *Iperf* 2.0.5 and saturate the link. For each experiment, the results are averaged over 5 runs, where each run lasts for 120 s.

Results: We first consider the scenario when only one of APs (i.e., AP1) has data traffic to send, and the other APs (AP2 to AP5) do not have data traffic and can be either active (on) or inactive (off). Thus, AP1 is the only rider among all active APs.

We evaluate the riding performance of AP1, through the ratio of the number of successful riding transmissions to the number of total riding trials, denoted as riding success ratio (RSR). Fig. 10 shows the results, where AP1 achieves high RSR for all the cases. This implies that riding data-ACK transmissions are well protected under the beacon interference. Also, we can observe that the impact of RTS/CTS on the RSR is negligible.

Fig. 11(a) shows throughput performance in the same scenario. For performance comparison, we use IEEE 802.11n as the *baseline*. It shows that, as expected, the total throughput linearly decreases as the number of active APs increases for both the baseline and *BeaconRider*. However, *BeaconRider* experiences much less performance degradation since it compensates a significant amount of air-time loss through air-time sharing. Fig. 11(b) illustrates the throughput gains of *BeaconRider* against the baseline according to the number of active APs, which clearly shows a linear relationship between the performance gain and the number of active APs. An interesting result is that when RTS/CTS is on, throughput performance decreases further for both the baseline and *BeaconRider*. This is due to additional overhead (or air-time consumption) incurred by RTS/CTS exchange for each data transmission. On the other hand, throughput gains of *BeaconRider* are enhanced



(a) Riding success ratio (RSR)

(b) Throughput

Fig. 15. Impact of MCS selection and adaptive riding (AR) on RSR and throughput.

under RTS/CTS as shown in Fig. 11(b) since *BeaconRider* initializes riding transmissions without incurring any overhead.

Next, we consider a multi-source scenario where all active APs have data traffic to send. In this case, AP1 and AP2 become the rider of AP2-5 and AP1, respectively. Fig. 12 shows the RSR values, which are lower than those in the single-source scenario. We observed that when the ridee AP has data traffic, it occasionally tries a data transmission during the residual time t^x_{residual} of riding transmission when it fails to detect the ongoing riding data-ACK frame transmissions due to the high threshold value for energy detection (i.e., -62 dBm). In that case, the rider often fails to receive BlockACK correctly even under high SIR conditions. We conjecture this is due to the limited capability of our dongle (TP-LINK WN722N v1), which results in slightly degraded performance of *BeaconRider*.

Fig. 13 shows the BRR of each individual STA when there are 5 APs. We can observe all the BRR values are close to 1, and thus, the beacon reception is well protected from the interference of riding data transmissions. For BRR, similar results are obtained for the case of 2-4 APs, which are omitted. Fig. 14(a) and Fig. 14(b) show the total throughput under the baseline and *BeaconRider*, and the gain of *BeaconRider* against the baseline. We observe a multiplexing gain when two APs generate traffic. Afterward, the performance linearly degrades due to the beacon congestion under the baseline and *BeaconRider*, but the degradation under *BeaconRider* is much milder. As before, the performance gain tends to linearly increase with the number of APs, and gets better under RTS/CTS.

We now investigate the impact of MCS selection and adaptive riding (AR) on RSR and throughput in *BeaconRider*. Choosing the optimal MCS level for riding transmission is crucial to achieve high riding performance. To this end, we build a network of two AP-STA pairs as shown in Fig. 5: the distance between AP1 and AP2 is set to 5 m, and STA2 is located between AP1 and AP2. AP1 and AP2 are the ridee and rider, respectively, and AP2 has saturated data traffic destined to STA2. We configure the SIR at STA2 by controlling the AP's transmission power: the transmission power of AP1 is set to a value in $[0, 15]$ dBm, and that of AP2 is fixed to 20 dBm, which corresponds to the SIR of $[7, 22]$ dB at STA2. We evaluate the performance of the baseline 802.11n, *BeaconRider* with a fixed MCS level for riding and without

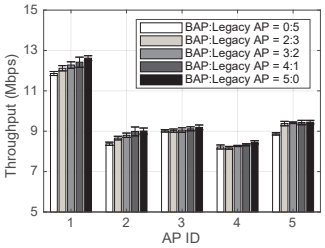
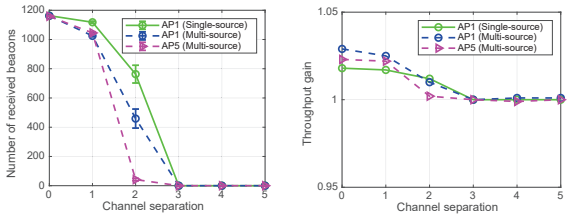


Fig. 16. Throughput of each AP under the coexistence of BAPs and legacy APs.



(a) Number of received beacon frames

(b) Throughput gain

Fig. 17. Beacon riding under partially overlapped channels.

AR (denoted as *StaticRider*), and *BeaconRider* with our MCS selection and without AR, and *BeaconRider*.

Fig. 15(a) and Fig. 15(b) show their performance in terms of RSR and throughput according to different SIR configurations. From Fig. 15(a), we observe that riding transmissions with high MCS levels often fail when the SIR is low (e.g., 12 dB). On the other hand, riding transmissions with low MCS levels result in low throughput gain when the SIR is high (e.g., 22 dB) as shown in Fig. 15(b). In contrast, *BeaconRider* with or without AR selects a right MCS level and thus achieves good performance for most SIR configurations.

An interesting point is that, when the SIR is 12 dB, *StaticRider*s with MCS 5-7 achieve even worse performance compared to the baseline. This is because their riding transmissions require higher overhead (i.e. longer residual time) due to lower transmission rate, and thus consume more air-time without contributing to throughput improvement since most of them fail as shown in Fig. 15(a). It shows that the MCS selection indeed plays a critical role to improve riding performance. In the case when the SIR is very low (e.g., 7 dB), riding transmissions often fail since their signal strength is lower than the capture threshold. In this case, *BeaconRider* with AR outperforms that without AR by refraining from riding, and achieves the performance comparable to the baseline.

In practice, there may be external interference in the ISM band, which comes from electronic devices such as microwaves, and other communication technologies like Bluetooth and Zigbee. A concern is that they may interfere with the operation of *BeaconRider*, leading to a worse performance compared to the baseline. However, *BeaconRider* performs well at least as the baseline since it operates in a conservative manner using the MCS selection and AR algorithms.

We also consider the coexistence scenario of BAPs and legacy APs. We deploy 5 APs and 5 STAs, and change the number of BAPs out of 5 APs. We measure the throughput of each individual AP, which depends on the locations of APs and STAs. The results are shown in Fig. 16. We set APs with smaller IDs as BAPs. For example, when we have 2 BAPs, AP1 and AP2 act as BAPs, and the rest as legacy APs. Interestingly, as the number of BAPs increases, the throughput performance of each AP improves. This implies that *BeaconRider* coexists well with legacy APs, and our solution not only improves the performance of BAPs but also is beneficial to legacy APs. We conjecture that riding transmissions using air-time sharing help reduce the contention overhead for typical data transmissions.

So far, we assume that all APs operate in the same channel. However, in practice, APs often operate in partially overlapped channels. Therefore, we investigate the riding performance of a BAP in partially overlapped channels. In our testbed shown in Fig. 9, we configure AP1 and AP5 as active, and each of which is associated with a STA. The other APs remain inactive. We let AP1 and AP5 operate in channels centered at channel $j\,(=10)$ and channel k, respectively. Changing *channel separation*, i.e., the difference in channel number $|j-k|$, from 0 to 5 by setting k from 10 to 5, we count the number of received beacon frames at each AP for 2 minutes and measure the throughput gain of *BeaconRider* against the baseline. Fig. 17 illustrates the results.

In a single-source scenario, where only AP1 has data traffic to send, the number of beacon frames received by AP1 decreases as the channel separation increases. When the channel separation is greater than or equal to 3, AP1 cannot receive a beacon frame from AP5. As the number of received beacon frames decreases, riding opportunities get reduced, which results in lower throughput gains as shown in Fig. 17(b).

In a multi-source scenario, where AP1 and AP5 have data traffic to send, we observe similar results. If their channel separation is greater than or equal to 3, they do not interfere with each other and operate independently. When their channels are partially overlapped, i.e., $|j-k|=1$ or 2, beacon frames from the other AP may work as interference. However, our results show that a significant number of beacon frames are successfully received and they can be used to improve the performance through air-time sharing.

B. Simulation Results

To investigate the excellence of *BeaconRider* in a more dense environment, we further evaluate its performance via ns-3 simulation.

Simulation settings: We employed *Minstrel* rate adaptation algorithm as the baseline and disabled RTS/CTS for all scenarios. We assume that APs and STAs use 802.11n transmission rates, i.e., MCS 0-7, with 20 MHz bandwidth, and conform to the standard protocols. We set the transmission power of APs and STAs to 20 dBm. We set the capture threshold for the sender-last capture as 10 dB. To generate Rayleigh fading effect, the Jakes' fading model with Doppler velocity of 0.1 m/s is applied. We also apply the pathloss model in IEEE 802.11ax task group (TGax) simulation scenarios which focus

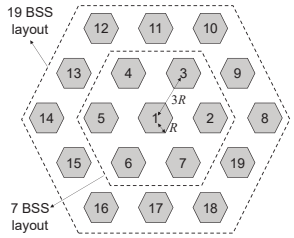


Fig. 18. 7/19 BSS layout for a static topology scenario.

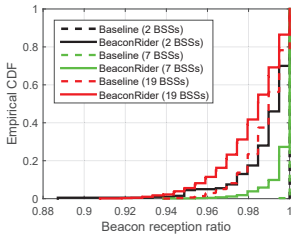


Fig. 19. Beacon reception ratio (BRR) in a static topology scenario.

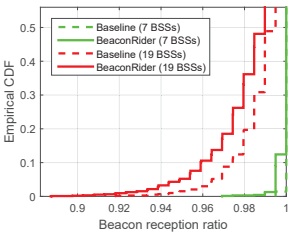


Fig. 22. Beacon reception ratio (BRR) in a random topology scenario.

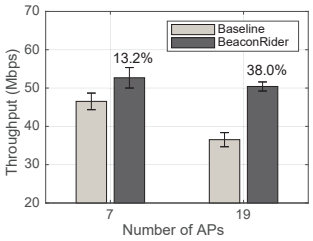


Fig. 23. Throughput in a random topology scenario.

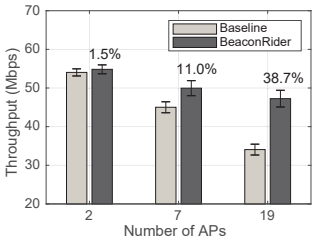


Fig. 20. Throughput in a static topology scenario.

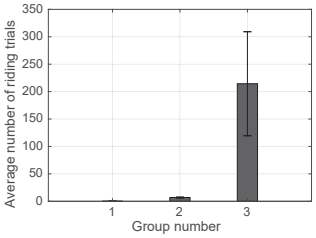


Fig. 21. Number of riding trials in a static topology scenario.

on densely deployed WLANs [37]. Lastly, we set the beacon frame length and its interval to 280 bytes and default, i.e., 102.4 ms, respectively. We compare *BeaconRider* with the baseline 802.11n, and averaged the results of 100 runs for each scenario. Each run lasts for 20 s.

Topology settings: We first compare the performance of static topology scenarios for different number of BSSs, i.e., 2, 7, and 19 BSSs. Each BSS is composed of an AP and a STA associated with it. For the case of 2 BSSs, we deploy two APs as shown in Fig. 5, where the distance between the two is given as 15 m. Each STA is randomly located within the radius of 5 m from its associated AP. For the case of 7 and 19 BSSs, we consider the hexagonal BSS layout of an indoor small BSS scenario in [37], which reflects high-density WLANs deployed in real environments. Assuming the frequency reuse factor of 3 is applied, we deploy 7 (19) APs from ID 1 to 7 (19) for the scenario of 7 (19) BSSs as shown in Fig. 18. The distance between neighboring BSSs and the coverage of each AP are given as $3R$ and R, respectively. We set $R = 5$ in all scenarios, and all APs have saturated downlink traffic destined to its associated STA.

We additionally consider the random topology of 7 and 19 BSSs. For the case of 7 (19) BSSs, we deploy BSSs randomly in a square of size 50 m $\times$ 50 m (80 m $\times$ 80 m). Each STA is randomly placed and associated with the AP with the strongest signal strength.

Results: The distribution of BRR of each STA in a static topology scenario is shown in Fig. 19. We observe that the BRR performance of *BeaconRider* slightly decreases compared with that of the baseline due to the interference from riding data transmission. Nonetheless, the degradation is very negligible and most beacon frames are well protected (i.e., the average BRRs for 2, 7, and 19 BSSs are 0.989, 0.997, and 0.980, respectively). Fig. 20 shows the total network throughput according to the number of APs. As expected, due to the beacon congestion problem, the throughput of the baseline and *BeaconRider* decrease with the number of APs. Nevertheless, *BeaconRider* significantly outperforms the baseline (up to 38.7% when the number of APs is 19) thanks to extra data transmissions by beacon riding.

To further show the effectiveness of the rider selection algorithm of *BeaconRider*, we compare the average number of riding trials according to the location of each AP for the case of 19 BSSs in Fig. 21. We divide all APs into three different groups depending on the distance from the center, i.e., group 1 (AP1), group 2 (AP2-AP7), and group 3 (AP8-AP19). Since a BAP selects the AP with the lowest RSS as its rider to alleviate interference between beacon and riding data transmissions, most of riding opportunities are assigned to the APs in group 3. Accordingly, the interference between beacon and riding data transmissions can be efficiently mitigated.

Fig. 22 shows the distribution of BRR of each STA in a random topology scenario. We observe that *BeaconRider* reliably protects beacon reception at STAs with negligible reception performance degradation (i.e., the average BRRs for 7 and 19 BSSs are 0.999 and 0.983, respectively). Fig. 23 shows the total network throughput performance according to the number of APs. Again, we can see that *BeaconRider* improves performance (up to 38% when the number of deployed APs is 19) with the number of APs using additional data transmissions riding on beacon frames.

VII. Concluding Remarks

In this paper, we addressed the beacon congestion problem and developed an opportunistic air-time sharing scheme, named *BeaconRider*, that *rides* on beacon air-time by exploiting the capture effect. *BeaconRider* is backward compatible and coordinates with each other in a distributed manner without incurring additional control message exchange. Our extensive experiments and simulations showed that *BeaconRider* achieves significant performance gains that increase linearly with the number of participating APs.

Our approach may be extended to transmission of other control, management, and low-rate data frames. For example, probe response frames used to discover 802.11 networks are transmitted at the lowest transmission rate. They can reduce available air-time further in high-density WLANs [38], [39]. Our riding technique can be applied on the probe response frames, which remains as an interesting future work.

REFERENCES

[1] Cisco visual networking index, "Global mobile data traffic forecast update, 2016–2021," Feb. 2017.

[2] IEEE 802.11, *Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE Std., Mar. 2012.

[3] IEEE 802.11ac, *Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Enhancements for very high throughput for operation in bands below 6 GHz*, IEEE Std., Dec. 2013.

[4] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala, "Sniffing out the correct physical layer capture model in 802.11b," in *Proc. IEEE ICNP*, Nov. 2004.

[5] J. Manweiler, N. Santhapuri, S. Sen, R. R. Choudhury, S. Nelakuditi, and K. Munagala, "Order matters: Transmission reordering in wireless networks," *IEEE/ACM Trans. on Networking*, vol. 20, no. 2, pp. 353–366, Apr. 2012.

[6] J.-W. Dai, "The system performance of wireless CSMA/CA protocol with capture effect," *KICS Journal of Communications and Networks*, vol. 6, no. 3, pp. 226–234, Sep. 2004.

[7] Ath9k-htc: Atheros wireless device firmware. https://github.com/qca/open-ath9k-htc-firmware.

[8] The network simulator 3 – *ns*-3. http://www.nsnam.org/.

[9] J. Lee *et al.*, "An experimental study on the capture effect in 802.11a networks," in *Proc. ACM WiNTECH*, Sep. 2007.

[10] S. Sati and K. Graffi, "Adapting the beacon interval for opportunistic network communications," in *Proc. IEEE ICACCI*, Aug. 2015.

[11] S. Nath, Z. Anderson, and S. Seshan, "Choosing beacon periods to improve response times for wireless HTTP clients," in *Proc. ACM MobiWac*, Oct. 2004.

[12] Y. Xie, X. Luo, and R. K. Chang, "Centralized PSM: an AP-centric power saving mode for 802.11 infrastructure networks," in *Proc. IEEE SARNOFF*, Mar. 2009.

[13] L. Le, R. Baldessari, P. Salvador, A. Festag, and W. Zhang, "Performance evaluation of beacon congestion control algorithms for VANETs," in *Proc. IEEE GlobeCom*, Dec. 2011.

[14] S. Bolufé *et al.*, "Dynamic control of beacon transmission rate and power with position error constraint in cooperative vehicular networks," in *Proc. ACM SAC*, Apr. 2018.

[15] Y. Mo, D. Yu, J. Song, K. Zheng, and Y. Guo, "A beacon transmission power control algorithm based on wireless channel load forecasting in VANETs," *PloS one*, vol. 10, no. 11, pp. 1–17, Nov. 2015.

[16] M. Torrent-Moreno, P. Santi, and H. Hartenstein, "Distributed fair transmit power adjustment for vehicular ad hoc networks," in *Proc. IEEE SECON*, Sep. 2006.

[17] G. Caizzone, P. Giacomazzi, L. Musumeci, and G. Verticale, "A power control algorithm with high channel availability for vehicular ad hoc networks," in *Proc. IEEE ICC*, May 2005.

[18] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, "CENTAUR: realizing the full potential of centralized WLANs through a hybrid data path," in *Proc. ACM MobiCom*, Sep. 2009.

[19] M. Vutukuru, K. Jamieson, and H. Balakrishnan, "Harnessing exposed terminals in wireless networks," in *Proc. USENIX NSDI*, Apr. 2008.

[20] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: Analog network coding," in *Proc. ACM SIGCOMM*, Aug. 2007.

[21] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *Proc. ACM SIGCOMM*, Aug. 2008.

[22] S. Gollakota, S. D. Perli, and D. Katabi, "Interference alignment and cancellation," in *Proc. ACM SIGCOMM*, Aug. 2009.

[23] J. Han, C. Joo, and S. Bahk, "Resource sharing in dual-stack devices: Opportunistic bluetooth transmissions in WLAN busy periods," *IEEE Trans. on Mobile Computing*, vol. 17, no. 10, pp. 2396–2407, Oct. 2018.

[24] W. Park, J. Han, S. Jang, and S. Bahk, "OAU: Opportunistic antenna utilization for Wi-Fi and bluetooth coexistence," in *Proc. IEEE GlobeCom*, Dec. 2016.

[25] A. Farshad, M. K. Marina, and F. Garcia, "Urban WiFi characterization via mobile crowdsensing," in *Proc. IEEE NOMS*, May 2014.

[26] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-e, A. Bhartia, and D. Aguayo, "Large-scale measurements of wireless network behavior," in *ACM SIGCOMM*, Aug. 2015.

[27] K. Sui, M. Zhou, D. Liu, M. Ma, D. Pei, Y. Zhao, Z. Li, and T. Moscibroda, "Characterizing and improving WiFi latency in large-scale operational networks," in *Proc. ACM MobiSys*, Jun. 2016.

[28] Wireshark 2.2.3. https://www.wireshark.org/.

[29] J. Yi, W. Sun, J. Koo, S. Byeon, J. Choi, and S. Choi, "BlueScan: Boosting Wi-Fi scanning efficiency using bluetooth radio," in *Proc. IEEE SECON*, Jun. 2018.

[30] Android open source project. https://source.android.com/.

[31] S. Kim, J. Yi, Y. Son, S. Yoo, and S. Choi, "Quiet ACK: ACK transmit power control in IEEE 802.11 WLANs," in *Proc. IEEE INFOCOM*, Apr. 2017.

[32] N. Kanirkar and J. Sarvaiya, "BER Vs SNR performance comparison of DSSS-CDMA FPGA based hardware with AWGN, spreading codes & code modulation techniques," *International Journal of Electronic Engineering Research*, vol. 1, no. 2, pp. 155–168, 2009.

[33] G. Pei and T. Henderson, "Validation of ns-3 802.11b PHY model," *http://www.nsnam.org/̃pei/80211b.pdf*, May 2009.

[34] *BeaconRider* source code. https://github.com/makesens86/BeaconRider.

[35] M. Vanhoef and F. Piessens, "Advanced Wi-Fi attacks using commodity hardware," in *Proc. ACM ACSAC*, Dec. 2014.

[36] Hostapd: IEEE 802.11 AP, IEEE 802.1X, WPA/WPA2/EAP/RADIUS Authenticator. https://w1.fi/hostapd/.

[37] IEEE P802.11, *TGax simulation scenarios*, Jul. 2015.

[38] R. Raghavendra, E. M. Belding, K. Papagiannaki, and K. C. Almeroth, "Unwanted link layer traffic in large IEEE 802.11 wireless networks," *IEEE Trans. on Mobile Computing*, vol. 9, no. 9, pp. 1212–1225, Sep. 2010.

[39] X. Hu, L. Song, D. Van Bruggen, and A. Striegel, "Is there WiFi yet?: How aggressive probe requests deteriorate energy and throughput," in *Proc. ACM IMC*, Oct. 2015.