Enabling Out-of-Band Coordination of Wi-Fi Communications on Smartphones

Xianjin Xia, *Student Member, IEEE*, Shining Li, *Member, IEEE*, Yu Zhang, *Member, IEEE, ACM*, Bingqi Li, Yuanqing Zheng, *Member, IEEE, ACM*, and Tao Gu, *Senior Member, IEEE*, *Member, ACM*

***Abstract*— This paper identifies two energy saving opportunities of Wi-Fi interface emerged during smartphone's screen-off periods. Exploiting the opportunities, we propose a new power saving strategy, BackPSM, for screen-off Wi-Fi communications. BackPSM regulates client to send and receive packets in batches and coordinates multiple clients to communicate at different slots (i.e., beacon interval). The core problem in BackPSM is how to coordinate client without incurring extra traffic overheads. To handle the problem, we propose a novel paradigm, Out-of-Band Communication (OBC), for client-to-client direct communications. OBC exploits the Traffic Indication Map (TIM) field of Wi-Fi Beacon to create a free side-channel between clients. It is based upon the observation that a client may control $1 \rightarrow 0$ appearing on TIM bit by locally regulating packet receiving operations. We adopt this $1 \rightarrow 0$ as the basic signal, and leverage the time length in between two signals to encode information. We demonstrate that OBC can be used to convey coordination information with close to 100% accuracy. We have implemented and evaluated BackPSM on a testbed. The results show that BackPSM can decode the traffic pattern of peers reliably using OBC, and establish collision-free schedules fast to achieve out-of-band coordination of client communications. BackPSM reduces screen-off energy by up to 60% and outperforms the state-of-the-art strategies by 16%–42%.**

***Index Terms*— Smartphone, screen-off traffic, Wi-Fi coordination, power saving, side-channel.**

I. Introduction

SMARTPHONES stay in stand-by mode (e.g., when the screen is off) for a large fraction of time during a day [6]. Although users are not actively interacting with phones, many apps and services still run in the background [7], [27], [35]. They stay connected to the Internet, updating app status, syncing with cloud servers or waiting for various incoming events such as emails, instant messages, notifications of social networking apps, etc. Such screen-off traffic is important for smartphones to provide good user experience. The energy consumption of screen-off traffic, however, has become a matter of concern. Recent studies [7], [11] reveal that screen-off traffic accounts for the total system energy by 29%–58%. To preserve battery power, mobile platforms like iOS and Android often prohibit screen-off background traffic under 3G or LTE, which are energy-costly, and only enable it when Wi-Fi is available (e.g., at homes or offices) [4], [6], [7].

The current Power Save Mode (PSM) of Wi-Fi, however, performs poorly when applying to screen-off traffic. We identify two energy sources in PSM that can be specially optimized for screen-off traffic. *First*, when a PSM client wants to send an upstream packet, it has to wake up the radio for transmission and switch back to sleep after sending. Although incurring expensive radio switch costs, it will be necessary for ordinary user traffic since it adds no delays to packet sending (i.e., good user experience). Whereas for the screen-off traffic which are mainly short, frequent packets [11], this would severely impair the energy saving benefits of PSM. Since screen-off traffic can often tolerate long delays due to absence of user interactions [11], a better way is to amortize such radio switch energy by sending multiple upstream packets in batches. *Second*, since PSM clients wake up simultaneously, at the beginning of beacon interval, to receive downstream packets, a client may spend long time contending for channel access, which can cause up to 4 times more energy consumption [22]. NAPman [28] and SleepWell [22] optimize this contention energy by staggering the beacon timing of APs (or virtual APs), which separates clients to communicate at different time of a beacon interval to avoid contentions. However, such schemes do not scale. The delay-tolerance nature of screen-off traffic enables us to break the limit of one beacon interval on traffic isolation. One can adopt coarse-grained time slice, e.g., use the whole beacon interval as a slice, to achieve better scalability.

In this paper, we explore the two opportunities by designing a new power saving strategy, BackPSM, for screen-off Wi-Fi communications. BackPSM regulates a client to send and receive packets in batches, and coordinates multiple clients to communicate at different beacon intervals (termed *slots*). The conventional idea to enable client coordination is to employ a central controller, such as a Wi-Fi AP [10] or the WLAN back-haul [36], to compute coordination information and disseminate them to all clients. However, this would incur tremendous traffic overheads, defeating the goal of

Manuscript received June 10, 2018; revised November 14, 2018; accepted December 17, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. F. Chiasserini. Date of publication March 6, 2019; date of current version April 16, 2019. This work was supported in part by the China NSFC under Grant 61872434 and Grant 61702437, in part by the China State Grid Jilin Electric Power Co., Ltd., under Grant SGJLXT00JFJS1800150, in part by the Australia Research Council Discovery under Grant DP180103932, and in part by the Hong Kong ECS under Grant PolyU 252053/15E. *(Corresponding author: Yu Zhang.)*

X. Xia, S. Li, and B. Li are with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China (e-mail: jinchenxia@mail.nwpu.edu.cn; libingqi@mail.nwpu.edu.cn; lishining@nwpu.edu.cn).

Y. Zhang is with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China, and also with the School of Computer Science and IT, RMIT University, Melbourne, VIC 3000, Australia (e-mail: zhangyu@nwpu.edu.cn).

Y. Zheng is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csyqzheng@comp.polyu.edu.hk).

T. Gu is with the School of Computer Science and IT, RMIT University, Melbourne, VIC 3000, Australia (e-mail: tao.gu@rmit.edu.au).

Digital Object Identifier 10.1109/TNET.2019.2891263

energy saving. Different from existing approaches, in this work, we ask: *is it possible to directly exchange coordination information between clients yet without extra traffic?*

We answer the question by presenting a novel paradigm to support client-to-client direct communications without occupying the regular Wi-Fi communication band (named *Out-of-Band Communication, OBC*). OBC exploits the TIM (Traffic Indication Map) field of Beacons, broadcasted periodically by a Wi-Fi AP, to create a free side-channel among clients, termed *TIM channel*. TIM bits are originally designed to notify presence of buffered downstream packet for PSM clients. We observe that a TIM bit will change from '1' to '0' when the corresponding client receives all buffered packets from the AP. This implies that a client may control the appearing of $1 \rightarrow 0$ on TIM bit by locally regulating packet receiving operations. Exploiting the observation, we adopt $1 \rightarrow 0$ of TIM bit as the basic signal and employ the distances between such signals as alphabets to encode information. An OBC-enabled client detects raw signals ($1 \rightarrow 0$) from the TIM bit sequence of a source client, and decodes out embedded information by interpreting the patterns created by signal distances. We demonstrate that, by subtly selecting the encoding symbols (i.e., signal distance), client can decode information with accuracy close to 100%. As OBC encodes information based on regulations of client's normal traffic activities, it incurs no extra packet exchanges on the Wi-Fi communication band.

We apply OBC in BackPSM to exchange coordination information (i.e., client's period and slot schedules) among clients. A client acquires traffic patterns of peer clients from the TIM side channel, and properly selects its own slots to avoid collision. With the wide adoption of BackPSM, it allows Wi-Fi clients to coordinate communications in a way akin to distributed TDMA, yet requiring no changes to the protocol.

We implement a prototype system on a testbed composed by two Nexus 4 phones and 10 wireless NICs connecting to 5 PCs. We perform extensive experiments on the testbed to evaluate the OBC paradigm, the out-of-band coordination scheme and the BackPSM strategy. Results show that clients can exchange coordination information reliably using OBC, and establish collision-free schedules fast to coordinate communications with peers. As compared with the default SPSM strategy of Nexus 4, BackPSM reduces screen-off system energy by up to 60%. BackPSM outperforms state-of-the-art strategies by 16$\sim$42%, and scales well to a dense network.

In summary, this paper makes the following contributions.

- We identify two energy hot spots in PSM. We show that they are hard to be reduced in ordinary cases, but can be further optimized for screen-off traffic.
- We propose a novel Out-of-Band Communication paradigm, which enables client-to-client communications without packet exchange. We demonstrate that OBC can enable phone-to-phone data sharing, Wi-Fi coordination and interference mitigation.
- We design and implement BackPSM, a new Wi-Fi power saving strategy for screen-off traffic. We evaluate BackPSM using extensive testbed experiments.

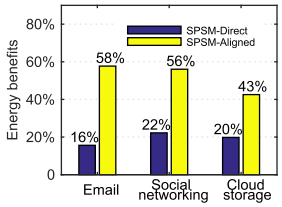


Fig. 1. The energy benefits of SPSM-Direct and SPSM-Aligned over APSM.

II. Background and Motivation

To preserve battery power, smartphones generally put Wi-Fi radio into Power Save Mode (PSM) when the network interface is idle. Two widely adopted power management strategies are: *Static PSM (SPSM)* and *Adaptive PSM (APSM)*. Both SPSM and APSM put Wi-Fi radio into sleep (i.e., a low-power state) when there is no traffic, and wake up periodically to receive Beacons, through which the AP informs clients the presence of downstream traffic. The main difference between SPSM and APSM is: SPSM switches radio back to sleep immediately after communication; while APSM keeps radio on for a certain time and turns off the radio if no traffic arrives during the time. Therefore, APSM has lower traffic delays, but SPSM may save more energy [28].

Since screen-off traffic can often tolerate long delays due to absence of user interactions [11], one may prefer smartphones adopting SPSM during screen-off periods. However, we observe that there is still room for improvement while applying SPSM to the screen-off traffic. We identify the energy saving opportunities for SPSM in screen-off scenarios as below.

Opportunity 1: (Radio Switch Energy on Upstream Sending): When an SPSM client sends upstream packet, it has to switch radio into active state for communication and switch back to sleep right after the packet sending. Frequent radio switches may degrade the energy benefits of SPSM, especially for screen-off traffic which are mainly frequent, short packets [11].

We replay the traffic traces of three typical screen-off apps on a Nexus-4 phone. We compare the energy benefits of two SPSM schemes over APSM. As shown in Fig. 1, the energy gain of the standard SPSM scheme, denoted as SPSM-Direct, over APSM is merely around 20%. The energy benefit of SPSM is impaired by the expensive radio switch costs of upstream sending, because when we defer upstream sending to the time when radio regularly wakes up to listen Beacon (SPSM-Aligned in Fig. 1), the energy benefit increases accordingly to nearly 50%. Ideally, a client should send upstream packets in batches to amortize the radio switch costs. Such a strategy does not work for the ordinary user traffic because it will add delays to packet sending, impairing user experiences. Whereas in the case of delay-tolerant screen-off traffic, we can exploit the idea to achieve significant energy savings.

Opportunity 2: (Contention Energy on Downstream Receiving): When multiple SPSM clients reside in the same Wi-Fi cell (or nearby cells working at the same channel), they wake up simultaneously, at beginning of a beacon interval, to receive buffered downstream packets from the AP [22]. Contentions will force radio to stay awake in idle/overhear mode when

other clients are transmitting. It may cause up to 4 times more energy consumption according to the measurements in [22]. Motivated by these results, we aim to avoid contentions by isolating screen-off devices into different beacon intervals to receive downstream packets.

The idea of isolating client communication at different time slices to optimize contention energy has been used by NAPman [28] and SleepWell [22]. However, due to small-delay restrictions of ordinary traffic, they can only divide limited number of time slices within the time of one beacon interval, and thus face scalability issues when the number of clients becomes large (see Fig. 17). The delay-tolerance nature of screen-off traffic enables us to break the limitation of one beacon interval. We exploit coarse-grained time slice, in unit of beacon interval, to isolate client traffic for better scalability.

Coordination Problem: By leveraging the two opportunities, we design a new power saving strategy for screen-off traffic based on SPSM. The new strategy schedules clients to send and receive packets in batches, and coordinate multiple clients to communicate at different beacon intervals in a way like TDMA. We implement the strategy at the client side since upstream sending can only be controlled at client. Here, the key problem is *how to coordinate clients without incurring extra traffic overheads*. We handle the problem by proposing a novel out-of-band client-to-client channel, which enables distributed coordination among clients without incurring extra communication cost.

III. Out-of-Band Communication Paradigm

A. Screen-Off Transmission Model

We divide the screen-off traffic of smartphone into multiple transmission tasks, where a task w typically corresponds to a packet. w can be upstream or downstream. Screen-off traffic is scheduled in time unit of beacon interval, termed *slot*. We denote the slot that task w arrives as $t_a(w)$, and the slot that w is scheduled as $t_s(w)$. We assume that w is delay-tolerant.

We denote the set of clients (i.e., screen-off smartphones) by $\mathcal{C}$. W represents the set of transmission tasks of all clients. Let $v(w)$ denote the traffic volume (in Bytes) of task w, and $c(t)$ the capacity of slot t, i.e., the maximum number of data bytes that can be transferred through AP. The condition

$$\sum_{w \in W \cap \{w | t_s(w) = t\}} v(w) \leq c(t) \tag{1}$$

should be met for all slots.

B. TIM Channel

A Wi-Fi AP uses the TIM (Traffic Indication Map) field of Beacons to inform PSM clients the presence of buffered downstream packets. Assume that c_i is a PSM client. A downstream packet of c_i arrives at the AP at t_a. The AP will set the TIM bit of c_i to '1' in the next slots. After c_i retrieving the packet at t_s, the AP will clear c_i's TIM bit (i.e., set c_i's TIM bit to 0) at slot t_{s+1}. Therefore, when the downstream packets of c_i are pending at the AP, c_i can regulate local packet receiving operations to control the appearing of $1 \rightarrow 0$ on TIM bit. Thus, it can further control the time length in between two $1 \rightarrow 0$ signals to encode information. A peer client detects $1 \rightarrow 0$ signals from the TIM bit sequence of c_i, and decodes the embedded information by interpreting the appearing patterns of $1 \rightarrow 0$ signal. Since Beacon frames (and the TIM field included) are broadcasted by the AP at every slot, all clients can obtain the TIM bit sequences of other peers. It provides a free side channel for client-to-client direct communications. Specifically, we call the TIM bit sequence of c_i as c_i's *TIM channel*, as illustrated in Fig. 2.

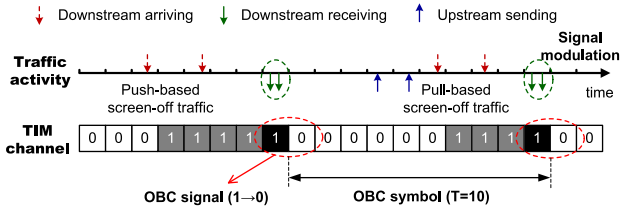


Fig. 2. Illustration of TIM channel.

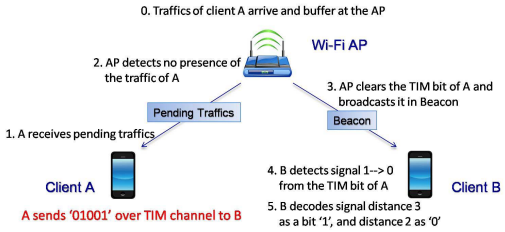


Fig. 3. General workflow of the OBC paradigm.

We build an Out-of-Band Communication (OBC) paradigm on the basis of TIM channel. Figure 3 shows the general workflow of OBC. Suppose that the traffics addressed to A have arrived at the AP. The TIM bit of A would be set in the Beacon broadcasted by the AP. Let t_0 denote the last time slot of client A receiving traffic from the AP. We employ communication intervals 2 and 3 to encode bit '0' and '1', respectively. At the sender side, client A needs to control the interval of two communication slots to be exact 3 slots (or 2 slots) to encode a bit '1' (or bit '0') on the TIM channel. Only at slot t_3 (or t_2), will A receive the pending traffics. In the next slot, AP detects no presence of buffered traffics of client A. It will clear the TIM bit of A and broadcast out in the Beacon frame. At the receiver side, upon receiving such a Beacon, client B will detect a $1 \rightarrow 0$ signal on the TIM bit of client A. B computes the distance between this new signal and the last signal, which is 3 slots, and decodes it as bit '1'. As OBC does not transfer through the Wi-Fi communication band, it adds no extra traffic and energy overheads to the radio.

We present the detailed encoding and decoding schemes of OBC in the following subsections.

C. Information Encoding

Basically, we adopt the change of $1 \rightarrow 0$ on TIM bit as signal and employ the time length in between two signals (i.e., *signal distance*) as symbol to encode information, as shown in Fig. 2. We term the set of signal distances used for information encoding as *alphabet set*, denoted by $\mathcal{A}$. For example, if $\mathcal{A} = \{2, 5\}$, we may use length 2 to encode an information bit '0' and 5 for bit '1'.

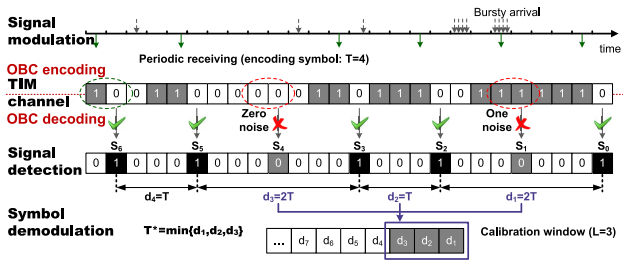
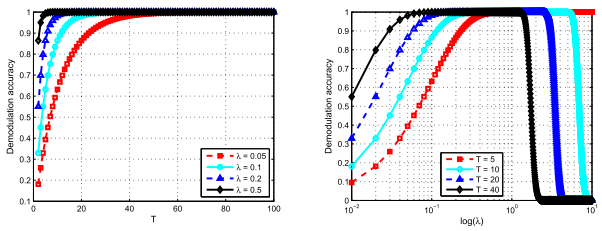


Fig. 4. Illustration of OBC encoding and decoding.



(a) (b)

Fig. 5. Relationships between demodulation accuracy and λ, T when $L = 2$. (a): demodulation accuracy vs. T, with T ranges from 2 to 100 slots; (b) demodulation accuracy vs. log(λ), λ ranges from 0.01 to 10 frames/slot.

We next address the problem of how to modulate signal $1 \rightarrow 0$ on TIM channel. Assume that client c_i has continuous downstream traffic. We schedule c_i at every T slots to receive packets from the AP. Signal $1 \rightarrow 0$ appears at c_i's communication slots (termed *signal slot*) if two conditions are satisfied:

- at least one packet arrived during the last T slots,
- all packets must be retrieved from the AP in one slot.

If the first condition fails, $0 \rightarrow 0$ will occur at the signal slot (see Fig. 4). We call it a *zero noise*. While the second condition fails, $1 \rightarrow 1$ occurs and we call it *one noise*. We want to reduce noise rate as low as possible. We achieve this by delicately selecting T for client c_i as below.

Theorem 1: Given that the arriving of downstream traffic is a Poisson process, λ is the average number of packets arrived per slot. The packet size of downstream traffic follows normal distribution $N(\mu, \sigma^2)$. We demand $\frac{1}{\lambda} \leq T \leq \frac{c(t)}{\mu\lambda}$, where $c(t)$ corresponds to the capacity of c_i's communication slot t.

Proof: Please refer to our conference paper [34] for the detailed proof. ■

D. Information Decoding

When a client receives information from TIM channel, it involves two procedures: signal detection and symbol demodulation. The signal detection procedure detects $1 \rightarrow 0$ signals from the TIM channel. Then the symbol demodulation procedure extracts symbols from the detected signal distances. We describe in details as below.

Signal Detection: Noises will impact the accuracy of signal detection. Assume that AP does not drop client's downstream packets. If $1 \rightarrow 0$ appears on client c_i's TIM channel, it would only be triggered by c_i retrieving all buffered packets from the AP. It is an authentic signal modulated by c_i. Therefore, the *false positive* rate (i.e., detecting $1 \rightarrow 0$ as a signal while it is not) of signal detecting is 0. However, when noise (either zero noise or one noise) occurs at the signal slot, a peer client can not observe $1 \rightarrow 0$, resulting in a *false negative* error. We conclude the analysis above with Theorem 2.

Theorem 2: Let P_I, P_{II} denote the false positive and false negative rate of signal detecting, respectively. We have $P_I = 0$, $P_{II} = 1 - \sum_{k=1}^{N_c} \frac{(\lambda T)^k}{k!} e^{-\lambda T}$.

Proof: Please refer to our conference paper [34] for the detailed proof. ■

Symbol Demodulation: Let d stand for the detected signal distance. As false negative error is the only error that may happen in signal detecting, the detected signal distance (d) and the source client's encoding symbol (T) must meet $d = nT, (n = 1, 2, \cdots)$. If no error occurs, $d = T$, otherwise $d > T$. Hence, we employ multiple signal distances for calibration and demodulate symbol as the minimum distance in the calibration window, as shown in Fig. 4. Formally, let d_i represent the i'th latest signal distance, T^* denote the demodulated symbol. We have $T^* = \min_{i=1}^{L}\{d_i\}$, where L is the calibration window size.

Theorem 3 gives the accuracy rate of this calibration based demodulation scheme when $L = 2$.

Theorem 3: Let p denote the error rate of signal detecting, $p = P_I + P_{II}$. $P(T^* = T)$ stands for the probability that demodulated symbol T^* equals to the source client's encoding symbol (T). When $L = 2$, $P(T^* = T) = 1 - p^2$.

Proof: Please refer to our conference paper [34] for the detailed proof. ■

On the basis of Theorem 3, Theorem 4 further provides the general demodulation accuracy when $L > 2$.

Theorem 4: p denotes the error rate of signal detecting. Generally, when L ($L \geq 2$) signal distances are employed for calibration, the accuracy of symbol demodulation is $P(T^* = T) = 1 - p^L$.

Proof: The detailed proof is presented in Appendix A. ■

We conclude by presenting the analytic results of demodulation accuracy in Fig. 5. We see that, for any traffic rate λ, one can always select a proper encoding symbol T to achieve 100% demodulation accuracy. Results in Fig. 5(b) indicate that a selected T produces high accuracy only in certain ranges of λ, which is consistent with Theorem 1. It suggests us to select T dynamically based on the real-time estimation of client traffic rate λ, which will be presented in Section IV-B.

E. Handling Beacon Loss

If an AP or client fails to send or receive Beacons, due to software/hardware failure or wireless contentions, it will impair both the sending and receiving of signal $1 \rightarrow 0$ over TIM channel. To remedy the problem, we present strategies to restore the missing TIM bits.

Assume that the Beacon frame of slot t is lost. Let b_t denote the TIM bit of slot t. An OBC sender can restore the missing TIM bit (b_t) with the bit of the former slot, i.e., $b_t = b_{t-1}$. On the other hand, an OBC receiver can restore the missing TIM bits of peer clients with two rules: (i) set $b_t = b_{t-1}$; (ii) if slot $(t-1)$ is a signal slot of peer client, reset b_t with b_{t+1} after receiving the Beacon frame of slot $(t+1)$.

As the strategy restores missing bit with either the old- or newly-received TIM bit, it adds no bit changes in non-signal slots. The false positive error rate is 0. This will be confirmed by experiment evaluations in Section VI-B.2.

IV. BackPSM Design

In this section, we design a new power saving strategy, BackPSM, for screen-off traffic. We apply OBC in BackPSM to exchange traffic pattern information of client beyond the regular Wi-Fi communication band. The usage of traffic pattern exchanging simplifies the selection of alphabet set for information encoding. But BackPSM also raises new issues on OBC decoding, calling for subtle strategy revisions.

A. Overview

The high-level idea of BackPSM is to locally regulate screen-off traffic into periodic pattern and coordinate with peer clients to transfer in different slots, i.e., like the idea of TDMA. The motivation for periodic upstream sending is to let client process outgoing traffic in batches so as to amortize the radio switch costs. While the goal of periodic downstream receiving is to encode traffic pattern information onto the TIM channel for Wi-Fi coordination (i.e., to realize traffic isolation).

BackPSM divides time into slots. Since Beacons are broadcasted every beacon interval, it provides synchronized timing for slot managing on clients. To deal with Beacon loss, we also use a local slot timer for backup. We present the pseudo-code of BackPSM in Algorithm 1. Procedure *Core* implements the main scheduling logic of BackPSM. It is invoked every slot, upon receiving a Beacon frame. *Core* extracts the TIM field of Beacon and pass it to *PatternDecoding*, which maintains TIM channels for peer clients and decodes traffic pattern information from the side-channel. *Core* periodically schedules client to send and receive. At the end of client's communication slot, *Core* invokes *PeriodManaging* and *SlotMaintaining*.

Out-of-Band Coordination: BackPSM empowers clients to spread and receive traffic schedules over TIM channel. After acquiring the traffic patterns of coexisting peers, BackPSM client progressively adjusts local schedules to isolate from peers, realizing out-of-band Wi-Fi coordination. In this context, client exchanges the information of communication period and slots through OBC. We adopt T as the only symbol for OBC encoding, i.e., $\mathcal{A} = \{T\}$. T corresponds to client period. The slot where signal $1 \rightarrow 0$ appears indicates client's communication slot. However, BackPSM raises specific issues on the strategy design of coordination information encoding and decoding:

- T should be properly selected to ensure high success rate for both symbol encoding and decoding. Unlike the basic OBC strategy using fixed encoding symbols, coordination information exchange must adapt the encoding symbol (T) to client's traffic dynamics.
- Dynamic changes of client traffic pattern may incur new noises to OBC decoding (see Section IV-C). The decoding strategy must not only be robust to noise, but also respond fast to pattern changes.

We address these issues in the following subsections.

B. Managing Period

To configure a client with the proper period (T), we estimate the traffic rate of client (λ) on-line and determine T dynamically based on Theorem 2.

To estimate λ, i.e., the number of packets arrived per slot, we count the number of packets received by client at every communication slot (N_T) and compute λ_0 with $\lambda_0 = \frac{N_T}{T_0}$, here T_0 represents the current period of client. We smooth λ_0 with the old estimation (λ') and obtain

$$\lambda = (1-\alpha)\lambda' + \alpha\lambda_0 \tag{2}$$

where, α ($0 \leq \alpha \leq 1$) corresponds to the weight of λ_0.

Note that N_T represents the number of arrived packets only if TIM bit signal $1 \rightarrow 0$ appears at the communication slot. If too many traffics arrive and cannot be retrieved by the client within one slot, it will result in $1 \rightarrow 1$ (i.e., one noise) on the TIM bits, and N_T would be smaller than the number of packets arrived. To precisely count the number of arrived packets, we allow the client to communicate more frequently by cutting period T by half. On the other hand, if no traffic arrives, it will

Algorithm 1 The BackPSM Scheduling Algorithm

```
1: procedure CORE(Beacon)
2:     Extract TIM field from Beacon;
3:     Call PatternDecoding(TIM field);
4:     if (current slot is a communication slot)
5:         Send outgoing packets (i.e., upstream);
6:         if (the TIM bit of client is set)
7:             Retrieve downstream traffic;
8:             N_T ← # of downstream packets;
9:         Call PeriodManaging(N_T);
10:        Call SlotMaintaining(SOM);
11: end procedure
12: procedure PATTERNDECODING(TIM field)
13:    foreach peer client c_k ∈ C
14:        Update the TIM bit sequence of c_k;
15:        if (1 → 0 occurs on c_k's TIM channel)
16:            Compute signal distance d_1;
17:            Filter short-distance noise using the
18:              last-decoded period of c_k;
19:            T* ← min{d_1, d_2};
20:            if (T* is detected twice)
21:                Decode T* as the period of c_k;
22:                Add T* in SOM;
23: end procedure
24: procedure PERIODMANAGING(N_T)
25:    Estimate client traffic rate λ with N_T;
26:    if (1 → 1 appears in two successive periods)
27:        T' ← T/2;
28:    if (0 → 0 appears in two successive periods
29:       or appears in every other period
30:       or P_II(λ, T) > δ for two periods)
31:        Select T' with P_II(λ, T') ≤ δ;
32: end procedure
33: procedure SLOTMAINTAINING(SOM)
34:    K = {t_k | t_k ∈ SOM ∧ |U_k| = 0};
35:    if (client c_i first joins)
36:        Randomly select a slot from K;
37:    else if (|U_i| > 1) //collision detected
38:        q ← the order of c_i's client ID in U_i;
39:        if (q > 1)    //slot change required
40:            Select the (q − 1)'th slot in K;
41: end procedure
```

result in $0 \rightarrow 0$ (i.e., zero noise) on the TIM bits and a client has to wait for more periods to receive. To cope with the problem, we modify the computing of λ_0 with $\lambda_0 = \frac{N_T}{T_s}$, where T_s denotes the time duration since the last slot when signal $1 \rightarrow 0$ appears.

Using the estimated traffic rate as an input, we apply Theorem 2 to choose period T for the client. Since T is also used as the encoding symbol of OBC, we attempt to restrict the error rate of OBC encoding within an upper-bound δ. We provide m levels of period, $T_1, T_2, \cdots, T_m$, and select T from the predefined set with a filter condition $P_{II}(\lambda, T) \leq \delta$. If multiple levels of period satisfy the condition, we select the one closest to the current period.

To ensure period stability, instead of adjusting T every time upon estimating a new λ, we reconfigure T only when one of the following conditions is met.

- One noise $1 \rightarrow 1$ appears for two successive periods;
- Zero noise $0 \rightarrow 0$ appears for two successive periods or in every other period;
- $P_{II}(\lambda, T) > \delta$ for two successive periods.

C. Decoding Traffic Pattern

BackPSM client maintains the TIM channel of all peers by collecting their TIM bit sequence from periodic Wi-Fi Beacons. We adopt the method in Section III-D to decode symbols from the TIM channel. If a symbol T is decoded twice in succession, we interpret T as the communication period of peer client and the slot at which $1 \rightarrow 0$ signal is detected as the communication slot.

Short-Distance Noise: The calibration-based demodulation strategy of OBC is built on that errors in signal detecting only produce noises with long signal-distance. However, BackPSM allows client to dynamically adjust traffic pattern, which may incur short-distance noise (i.e., signal distance being shorter than client period) on the decoding side. A short-distance noise will be mistakenly demodulated as OBC symbol. It causes demodulation errors in not only the slot at which the noise occurs, but also the following slots as long as noise resides in the calibration window. More specifically, the basic strategy is inefficient in decoding traffic pattern in two aspects:

- When client changes communication slot, the offset of new slot from the old slot is shorter than client period and can be mistakenly decoded as an OBC symbol, incorrectly interpreted as a 'new' period.
- When client enlarges period, the short signal distances of old period reside in the calibration window, preventing the decoding of new traffic pattern.

We extend the basic strategy to handle short-distance noise. The main challenge lies in how to filter noise from the short signal-distance caused by regular adjusting of client period, e.g., period decrease.

Pattern-Assisted Decoding: We exploit the last-learned traffic pattern of peer client to assist noise filtering in the decoding process. Let T_0 denote the last-decoded period, and d_1, d_2 be the last and second last signal distance of the peer client. If $d_2 < T_0, d_1 = T_0$, d_2 is a short-distance noise. We must remove d_2 from the calibration window while performing symbol demodulation. Whereas, if d_1, d_2 are both smaller than T_0, then it is caused by peer client shortening the communication period. We can use the basic strategy for symbol demodulating. Similarly, if both d_1 and d_2 are larger than T_0, it indicates that peer client is enlarging the period. We remove the residing short signal-distances corresponding to old period (T_0) from the calibration window to achieve fast and correct symbol demodulating.

D. Selecting Slot

The task of slot selecting is to realize distributed coordination by isolating client traffic in different slots. After acquiring the traffic patterns of coexisting peers through TIM channel, a client can select the slot that is not occupied by any client to communicate. However, if multiple clients join at the same slot, they would obtain the same knowledge on peer traffic patterns and choose the same slot, resulting in collisions. We employ a hash-table like idea to resolve slot collision. The implementation of our slot selecting strategy relies on a data structure named Slot Occupation Map (SOM).

Slot Occupation Map: SOM is an array with T_m elements. Each element stands for a slot. We formally represent an SOM as $SOM = \{t_i | 0 \leq i \leq T_m - 1\}$. For each slot t_i, U_i denotes the set of clients communicating at t_i. If client $c_j \in U_i$, t_i is occupied by c_j. We set the size of SOM as the maximum client period (i.e., T_m). To record the traffic patterns of any peers, we uniformly map the system time (t) to a slot t_k in SOM as: $t_k = t \bmod T_m$.

A client adopts SOM to track traffic patterns of peer clients in procedure *PatternDecoding* (line 22). SOM is passed to procedure *SlotMaintaining* where slot selecting is performed: When a new client first joins, it randomly selects one slot from set $\mathcal{K} = \{t_k | t_k \in SOM \wedge |U_k| = 0\}$. The client performs collision detection (line 37) at the end of every period. If slot collision is detected, BackPSM rearranges slot for colliding clients in a distributed way. The client with the smallest ID stays in the current slot. The other $|R| - 1$ clients re-select slot in the order of client ID (lines 37-40).

E. Overhead Analysis

The computation overhead of BackPSM comes mainly from the decoding of out-of-band coordination information, i.e., procedure *PatternDecoding*, which is invoked every slot. Let T_i denote the period of client c_i ($c_i \in \mathcal{C}$). In *PatternDecoding*, we parse the TIM bit of every peer client (say c_i) and record the decoded traffic pattern in SOM when a signal $1 \rightarrow 0$ is being detected, which occurs every T_i slots. Therefore, the computation complexity is $\sum_{i=1}^{|\mathcal{C}|}(1 + \frac{1}{T_i} \cdot \frac{T_m}{T_i})$. Since $T_1 \leq T_i \leq T_m$ (T_1 and T_m are constant), we have $(1 + \frac{1}{T_m})|\mathcal{C}| \leq \sum_{i=1}^{|\mathcal{C}|}(1 + \frac{T_m}{T_i^2}) \leq (1 + \frac{T_m}{T_1^2})|\mathcal{C}|$. The computation complexity of *PatternDecoding* is $O(|\mathcal{C}|)$. We also derived the computation complexity of *PeriodManaging* and *SlotMaintaining* as $O(\frac{1}{T_i})$ and $O(\frac{|\mathcal{C}|}{T_i})$. Therefore, the overall computation complexity of BackPSM is $O(|\mathcal{C}|)$, where $\mathcal{C}$ is the set of BackPSM clients.

The memory overhead mainly comes from data structure SOM. The required storage space of SOM is: $T_m + \sum_{i=0}^{T_m - 1}|U_i| = T_m + |\mathcal{C}|$. Although $PatternDecoding$ parses the TIM bit of all peers, it does not need to store these TIM

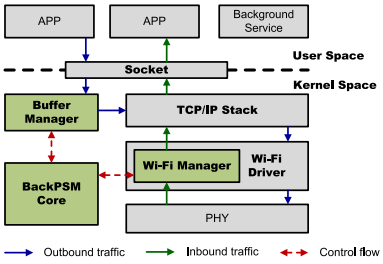


Fig. 6. The system architecture of BackPSM.

bits. Instead, we only record the last detected signal distance (d_1) and the slot of last $1 \rightarrow 0$ signal, which consumes $O(1)$ storage space. *PeriodManaging* and *SlotMaintaining* also require just $O(1)$ storage. Therefore, the overall space complexity of BackPSM is $O(T_m + |\mathcal{C}|)$.

V. System Implementation

We have implemented BackPSM on two platforms: a smartphone platform (LG/Google Nexus 4) and a Linux-based PC platform (Ubuntu 14.04). We employ the Ralink rt2800usb wireless NIC as Wi-Fi interface for PC. For the smartphone platform, Nexus 4 is equipped with Atheros WCN3660 Wi-Fi chipset. The OS version is Android 4.2 (kernel version: 3.1). BackPSM is implemented in kernel space. By default, smartphone starts BackPSM when screen is off for five minutes. Fig. 6 shows the architecture of BackPSM. It includes the driver modification *Wi-Fi Manager* and two kernel components, *Buffer Manager* and *BackPSM Core*.

BackPSM Core implements the main scheduling algorithm of BackPSM. It receives out-of-band coordination information and controls two other components to send and receive screen-off traffic. *Buffer Manager* receives outgoing traffic from upper layer apps and transfers them in batches. *Wi-Fi Manager* is a component resident in the Wi-Fi driver. It delivers received Beacons to *BackPSM Core* and exposes downstream receiving to the control of *BackPSM Core*.

To ensure client sending upstream packets in its own slots, *Buffer Manager* intercepts the packets sent by upper layer apps before they are passed to the TCP/IP stack. *Buffer Manager* stores outgoing packets in an FIFO queue. When client's slot starts, *BackPSM Core* sends a message to *Buffer Manager*. The later delivers pending packets to the TCP/IP stack to send them out in batches.

In the current implementation of Wi-Fi driver, received Beacons are processed by a *beacon_handler* that simply drops Beacon after processing. We modify it to pass Beacon to *BackPSM Core* after conventional processing. We also modify *beacon_handler* for downstream control. By default, *beacon_handler* issues PS-Poll to retrieve downstream packets from AP once detecting a TIM bit set. We modify the condition as '*TIM bit is set* $\wedge$ *the current slot is a communication slot*'.

BackPSM requires no modification at the AP end since it only demands an AP to periodically send Beacons, which is a standard operation supported by any commodity AP device. At the client end, we can include the changed driver into the firmware and install on smartphones through software upgrade.

VI. Evaluation

A. Methodology

Devices: Our testbed consists of one server, one AP, and 12 Wi-Fi clients (two Nexus-4 phones and 10 wireless NICs connecting to 5 PCs). We employ the TP-LINK TL-WDR7500 wireless router as AP. We use an Agilent N6750A power meter to measure the realtime current draw of Nexus-4 phone. The purpose of employing PC-controlled NICs is to flexibly generate traffic and replay the traces collected in real screen-off smartphones.

Traffic Traces: We set up a Wi-Fi hotspot in an office room and use Wireshark to collect traffic traces from 20 users when their phones are leaving in standby mode (i.e., screen is off). We label each traffic record with its origin app id, and observe that $> 90\%$ of the collected traffic traces are from Email, Social Networking Apps, Cloud Storage and News. The rate of screen-off traffic varies across users and apps. Generally, a client sends/receives 1.89 packets per second.

B. Out-of-Band Communication

This section evaluates the encoding/decoding performance of OBC. We setup one server, one AP and two smartphones (one OBC sender and one receiver). The server sends downstream traffic to the OBC sender client. The OBC receiver collects and decodes the TIM bit sequence of OBC sender.

1) Reliability and Accuracy: We evaluate OBC encoding with two metrics:

$$\text{signal success rate} = \frac{\text{\# of signals}}{\text{\# of signal modulating}} \tag{3}$$

$$\text{symbol success rate} = \frac{\text{\# of symbols}}{\text{\# of symbol encoding}} \tag{4}$$

where, # of signal modulating (# of symbol encoding) corresponds to the number of modulating (encoding) attempts of OBC sender, which equals to the number of receiving slot (period) of the client.

We first study encoding performance under different (λ, T) combinations. We vary traffic rate λ from 0.01 to 5 packets/slot and change the period of OBC sender (i.e., T) from 5 to 80 slots in the experiment. The results are reported in Fig. 7, where the theoretic signal success rate computed by Theorem 2 is also displayed. We observe high consistency between theory and experiment results. The signal success rate is lower than 50% when $\lambda = 0.01$, due to the impact of zero noises. As λ increases, fewer zero noises occur and the signal success rate improves to nearly 100% under proper T settings (Fig. 7 (b-e)). While λ increases to ≥ 1, one noise starts to appear under long client period; the signal success rates drop to 0, as shown in Fig. 7 (f,g,h). Such experiment results confirm that the encoding symbol (T) should be selected adaptively based on client's traffic. Figure 7 also evaluates the period adjusting strategy of BackPSM. It produces high success rates in almost all experiment settings, indicating that BackPSM is able to select the right symbol (i.e., client period) for OBC encoding.

Next, we examine the performance of OBC decoding. We compare different schemes of symbol demodulation: with no calibration ($L = 1$) and with calibration window $L = 2, 3, 4$.

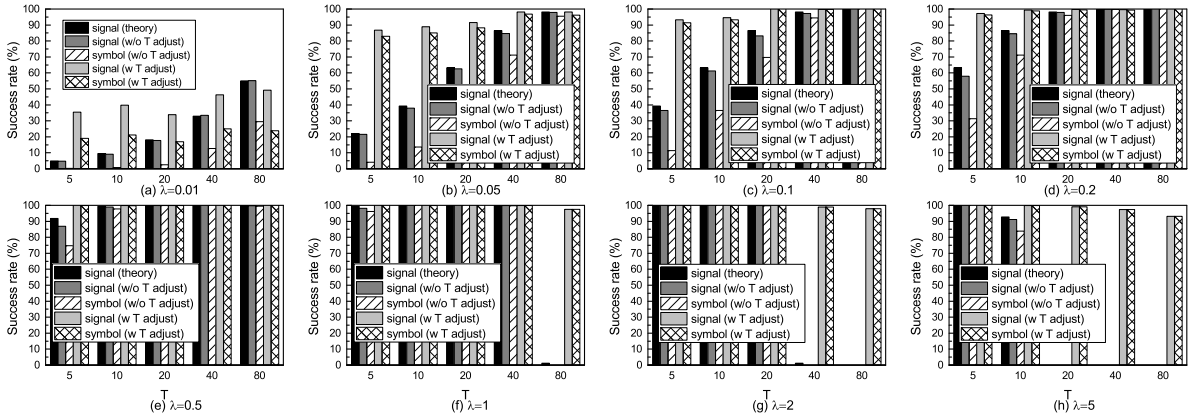


Fig. 7. OBC encoding performance under different combinations of traffic rate (λ) and client period (T).

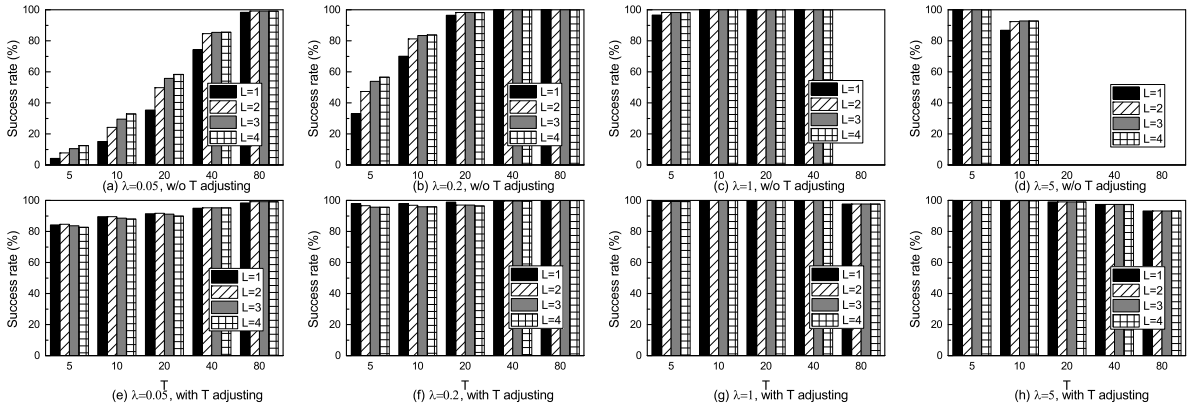


Fig. 8. OBC decoding performance under different λ-T configurations, (a-d): OBC sender adopts no period adjustment; (e-h): with period adjustment.

We record the symbols encoded by OBC sender as ground truth to compute the decoding accuracy of OBC receiver.

We report the results of $\lambda = 0.05, 0.2, 1, 5$ in Fig. 8(a-d). As we can see, the calibration-based scheme performs better in low traffic-rate scenarios compared to other schemes without calibration. When $\lambda = 0.05, T = 20$, it improves the decoding accuracy of no calibration method from 35.4% to 49.7%, by simply calibrating detected signal distances with $L = 2$. The larger L it adopts, the higher decoding accuracy it achieves. However, in high traffic-rate scenario where one noises occur (e.g., $\lambda \geq 1, T = 80$), calibration-based schemes produce no performance improvements, because few signals can be detected in noisy situations, providing no signal distances for symbol calibration. Figures 8(e-h) present the decoding results when OBC sender adaptively adjusts period T. The decoding accuracy improves significantly to above 80% and 90% in low and high traffic-rate scenarios.

2) Impact of Beacon Loss: In this experiment, we test the robustness of OBC in the presence of Beacon loss. We emulate Beacon loss by randomly dropping Beacons at the Wi-Fi driver of OBC sender and receiver.

We first examine the impact of Beacon loss on OBC encoding. As shown in the top of Fig. 9, the signal success rate decreases linearly as Beacon loss rate increases, when the OBC sender adopts no bit repair strategy. By restoring missing TIM bit with the old bit of the previous slot, it produces close to 100% success rate.

In the middle of Fig. 9, we evaluate the error rate of signal detecting when OBC receiver adopts different bit repair strategies. We observe higher error rate under high loss rate of Beacons. Compared to the Naive strategy that restores missing TIM bit with the previous bit, our repair strategy (Section III-E) produces no false positive errors. This is preferable since it introduces no short-distance noises for OBC decoding.

We present the decoding accuracy in the bottom of Fig. 9. We see that decoding accuracy decreases linearly as Beacon loss increases. It also exhibits the benefits of calibration-based decoding strategy on handling Beacon loss. As compared to the no calibration strategy ($L = 1$), employing a calibration window of $L = 2$ can improve decoding accuracy by 17.1% and 37.5% when Beacon loss rate is 0.2 and 0.4. Larger L produces even better results.

3) Impact of Locations: In this experiment, we place the OBC sender and receiver at different locations of the lab floor, as shown in Fig. 10, and examine the impact of sender-receiver locations on OBC performance. The sender and receiver can communicate through OBC as long as they both are in the range of AP. Since the Wi-Fi Signal-to-Interference-plus-Noise-Ratio (SINR) varies with locations, it will affect Beacon receptions of smartphone. We show the measured signal, noise power and Beacon loss rate at different locations in Fig. 11(a). We observe the maximum coverage range of the AP is about 30m in the lab building. Signal power at F, G and H are too

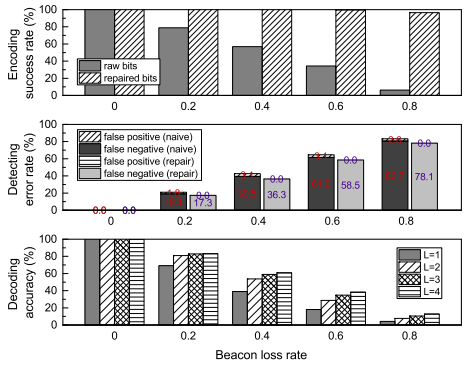


Fig. 9. Impacts of Beacon loss on OBC encoding (top) and decoding (middle and bottom) when $\lambda = 2, T = 10$.

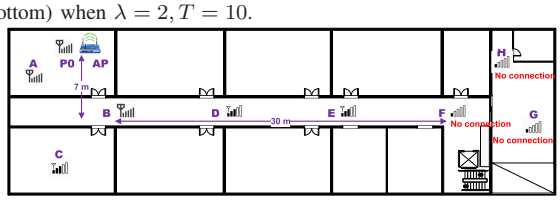


Fig. 10. Illustration of OBC sender/receiver locations.

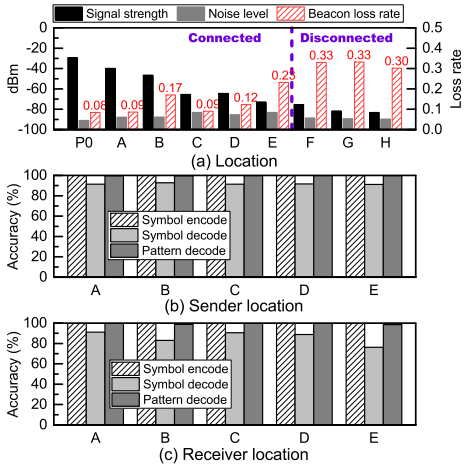


Fig. 11. Impact of locations on OBC performance.

weak to connect the smartphone to AP. The Beacon loss rate ranges from 8% to 23% at locations P0 and A-E.

Figure 11(b) presents the results when the receiver resides at P0 and the sender locates at A-E. We adopt restoring strategies to handle Beacon loss on both smartphones and employ $L = 2$ for OBC decoding. The results show that, regardless of the varying Beacon loss rate, the sender encodes symbol with 100% reliability at all locations. The receiver is more sensitive to Beacon loss. Although the loss rate at P0 is low (8%), it demodulates symbol with only 92.3% accuracy. Similar results are obtained when the receiver locates at A-E; the decoding accuracy ranges from 79.4% to 91.6%, as shown in Fig. 11(c). Nevertheless, the receiver client can decode the traffic pattern of OBC sender with over 98.5% accuracy at all locations. This validates the robustness of our pattern decoding strategy.

C. Wi-Fi Coordination

This section evaluates the performance of out-of-band Wi-Fi coordination. We are interested in (1) how fast and accurate a client decodes the traffic pattern of peers, (2) the stability and self-adaptability of out-of-band coordination, and (3) how it reacts to collisions. Unless stated otherwise, we set $L = 2$ when decoding traffic pattern.

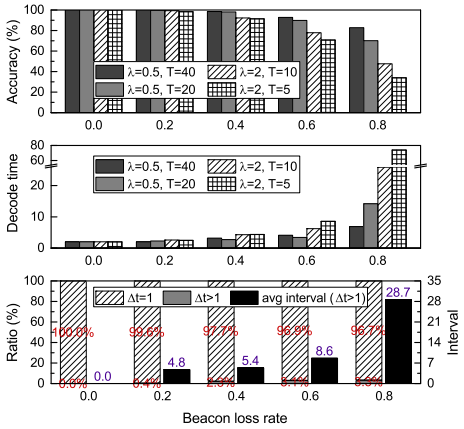


Fig. 12. Top: decoding accuracy; Middle: decoding latency; Bottom: decoding time interval ($\lambda = 2, T = 10$).

1) Traffic Pattern Decoding: In this experiment, we evaluate the traffic pattern decoding strategy of BackPSM in terms of decoding accuracy and latency. We set up one OBC sender and one receiver. We record the traffic patterns decoded by OBC receiver and compare with ground truth, i.e., the real patterns of OBC sender, to compute the decoding accuracy. Results are presented in the top of Fig. 12. We see that, ideally with no Beacon loss, our strategy achieves 100% decoding accuracy. Although the performance drops down a little as Beacon loss becomes frequent, it still produces >80% accuracy under 40% Beacon loss. We also observe that shorter period suffers more from Beacon loss, e.g., $T = 5$. This is because shorter period corresponds to more frequent OBC signaling, which is more vulnerable to TIM channel errors (i.e., TIM bit missing).

The middle figure of Fig. 12 reports the time when OBC receiver first decodes the traffic pattern of sender, i.e., decoding latency. We normalize decoding latency in unit of the period of OBC sender. As we can see, it generally takes two periods to decode the traffic pattern of OBC sender. The time increases when Beacon loss happens.

To study the stability of decoding strategy, we further examine the time interval (Δt) in between two correct pattern decoding. As shown in the bottom of Fig. 12, with no Beacon loss, the traffic pattern is correctly decoded at every period, i.e., $\Delta t = 1$. But when Beacon loss happens, the traffic pattern may not be correctly decoded at some periods, resulting in $\Delta t > 1$. The ratio of $\Delta t > 1$ only exhibits a tiny increase (<4%) when the Beacon loss rate increases from 0 to 0.8. This demonstrates the stability and robustness of traffic pattern decoding in BackPSM.

2) Responding to Pattern Changes: In this experiment, we control the OBC sender to change traffic pattern and evaluate the response time of receiver. We compare the basic traffic pattern decoding strategy with pattern-assisted decoding.

We first evaluate the impact of slot change. The OBC sender is configured with $\lambda = 1, T = 10$. It communicates at the first slot (i.e., slot 0) of every period. At the 11th

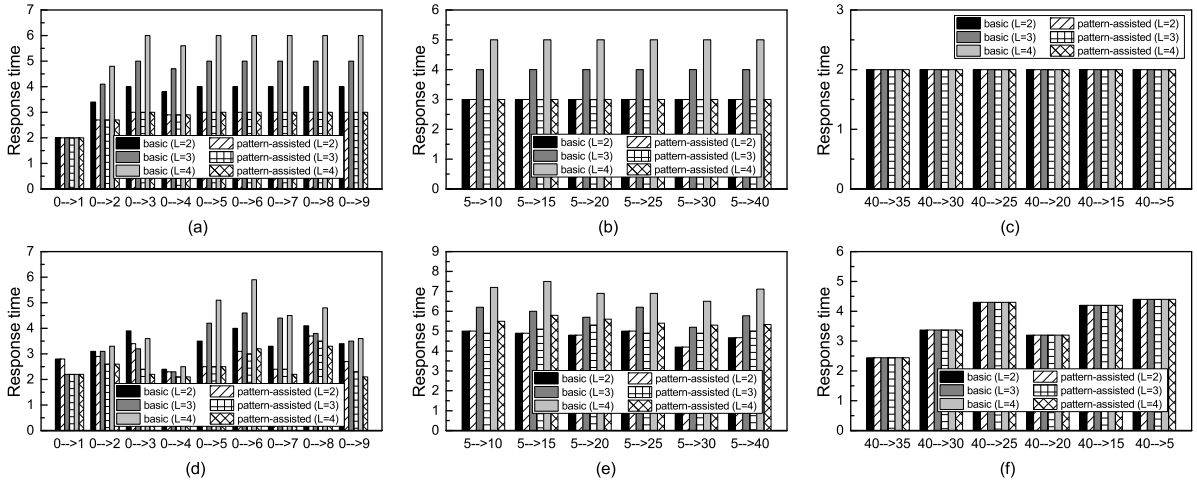


Fig. 13. The response time of OBC receiver when the OBC sender changes traffic pattern.

period, we change the OBC sender to a different slot (i.e., slot 1 to 9). Figure 13(a) presents the response time of OBC receiver. As we can see, the basic strategy takes 3.9 periods, on average, to decode the new traffic pattern when adopting calibration window $L = 2$. And the larger L it adopts, the longer time it takes, due to impact of short-distance noise incurred by the slot change of OBC sender. By filtering out such noises, the pattern-assisted decoding strategy produces fast response time of ≤ 3 periods. More performance gains are achieved in Beacon loss scenarios, as shown in Fig. 13(d).

Next, we evaluate the impact of period change. Figure 13(b) presents the response time of OBC receiver when the sender increases T from 5 to 10, 15, 25, 30 and 40. The average response time of the basic decoding strategy is 3 periods when $L = 2$, and increases to 5 periods when $L = 4$. This is because short symbols of old period residing in calibration window prevents the decoding of new period. The problem is handled by the pattern-assisted decoding strategy, which produces fast response time of 3 periods in all cases. When the OBC sender decreases T from 40 to 35, 30, 25, 20, 15 and 5, both decoding strategies are sensitive to short symbols of new period and respond fast in 2 periods, see Fig. 13(c). When Beacon loss occurs, both strategies take longer time to decode the new traffic pattern of OBC sender, as shown in Fig. 13(e, f).

3) Adapting to Traffic Dynamics: In practice, the screen-off traffic rate varies dynamically. We set up experiment to evaluate the self-adaptability of BackPSM. We are interested in (1) how fast the OBC sender can adapt period T to traffic rate λ (i.e., *adjusting time*), and (2) how fast an OBC receiver can decode the adjusted traffic pattern of sender (i.e., *decoding time*). In the experiments, we empirically configure the parameters of BackPSM period managing (see Section IV-B) as $\alpha = 0.8$, $\delta = 0.01$.

Figure 14(a) presents the results when OBC sender increases traffic rate. We initialize the OBC sender with $\lambda = 0.05, T = 80$, and increase λ to a higher rate λ' at the 11th period. We see that no period adjusting occurs when $\lambda' < 0.6$, indicating a good stability of BackPSM. When $\lambda' \geq 0.6$, it takes the OBC sender 7.6 periods to estimate the new traffic rate and adjust period. The adjusting time varies little in different

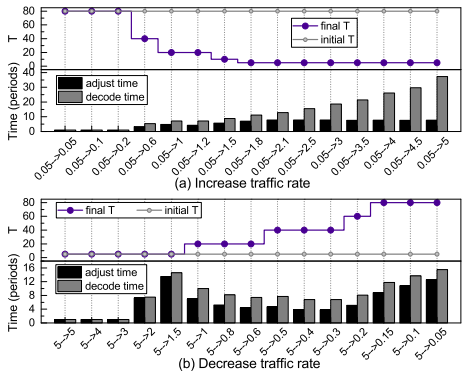


Fig. 14. Adaptability of BackPSM to traffic dynamics.

λ increasing cases (i.e., stability and fast convergence). But when λ' becomes higher, the decoding time of OBC receiver increases drastically. This is because, lots of packets arrive when λ increases; they cannot be received within one slot before the OBC sender adjusts to a shorter period. It incurs one noises on the TIM channel and costs long time to decode the adjusted traffic pattern of OBC sender at the OBC receiver.

Figure 14(b) reports the results when OBC sender decreases traffic rate from an initial setting of $\lambda = 5, T = 5$. We see that, only when λ decreases to lower than 3, will period adjusting be required. The adjusting time of OBC sender ranges from 4 to 12 periods. It takes OBC receiver 3 more periods to decode the adjusted traffic pattern.

4) Collision Resolving: Collision happens when multiple clients select the same slot. This experiment tests the performance of BackPSM on resolving collision. We evaluate the time it takes to reach a stable collision-free state, termed *convergence time*.

Figure 15(a) shows the CDF of convergence time when 8 clients join simultaneously. We configure $\lambda = 1, T = 10$ for each client and repeat 20 times the experiment. We observe that, it takes 4.9 periods, on average, to build a collision-free schedule. And 95% of the convergence time are less than 7 periods. We further evaluate the impact of client number on convergence time. As shown in Fig. 15(b), BackPSM converges fast, $\leq$ 10 periods, in all experiment settings.

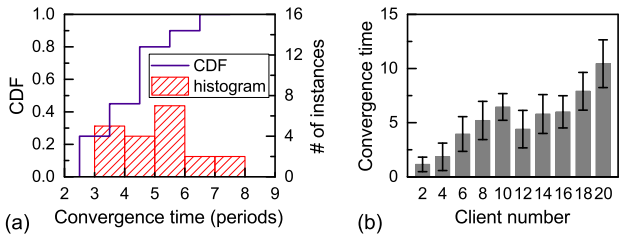


Fig. 15. Convergence time of collision resolving.

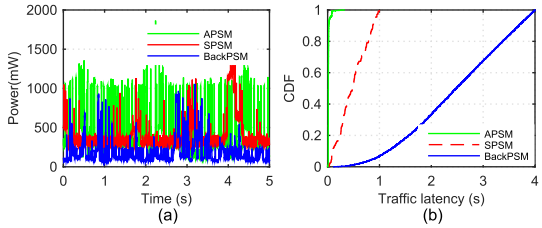


Fig. 16. Comparison of BackPSM with APSM and SPSM in network of 10 clients: (a) System power of Nexus 4 phone; (b) Traffic latency.

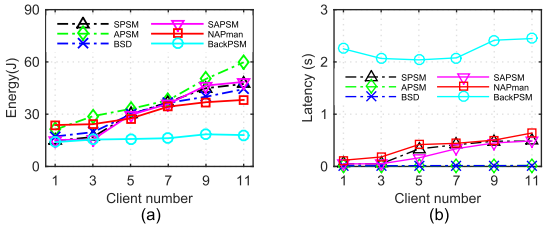


Fig. 17. Comparison of BackPSM with state-of-the-art strategies under different network scales: (a) Energy consumption; (b) Traffic latency.

As expected, the convergence time becomes longer if the client number is large.

D. BackPSM Performance

This section evaluates BackPSM via controlled experiments, aiming to test BackPSM in an extreme situation. We deploy all clients within one meter of the AP to ensure that any client is within the interference range of one another. We set up client to send data requests (500 Bytes) every 80ms. Upon receiving a request, the server responses immediately with a 1024-Byte packet. All clients have the same traffic. We focus on the power consumption of Nexus-4 phone.

1) Power Consumption vs. Traffic Latency: We first compare BackPSM with two benchmark strategies, i.e., APSM and SPSM. Figure 16(a) plots a snapshot of power consumption for the three strategies in a network with 10 clients. As expected, APSM consumes the highest power, with an average of 633.5 mW. The next is SPSM, 532.4 mW on average. BackPSM has the lowest power consumption of 358.6 mW. As compared with APSM and SPSM, BackPSM reduces the screen-off system power by 43.4% and 32.6%, respectively.

Figure 16(b) compares the traffic latency performance of three strategies. Traffic latency is computed as the time from when client sending a request to that of receiving a response. The average latency of BackPSM is 2447.2 ms. It is one order of magnitude larger than that of SPSM (484.3 ms), and two orders larger than APSM (19.1 ms).

2) Scaling to Dense Wi-Fi: We next compare BackPSM with state-of-the-art strategies. In the experiments, we increase client number from 1 to 11, with a step of 2. Figures 17(a) and (b) exhibit the average energy consumption and traffic latency of these strategies under different network scales. We see that as client number increases from 1 to 11, the energy of state-of-the-art strategies increase dramatically (increased by 1 to 2 times), while BackPSM experiences only a slight increase. In a network of 6 clients, the energy benefits of BackPSM over other strategies range from 52.8% to 61.1%. The benefit grows as network becomes dense. But the energy gain of BackPSM comes at the cost of long traffic latency, as shown in Fig. 17(b).

3) Incremental Deployment: In practice, BackPSM clients may coexist with clients not employing BackPSM. In this experiment, we explore how BackPSM affects the performance of both kinds of clients. We set up 10 clients and vary the ratio of client adopting BackPSM from 0% to 100%, here the settings of 0% and 100% are used for benchmark comparison. The client not employing BackPSM will use SPSM instead. Figure 18(a) shows the results. As we can see, the energy of both BackPSM client and non-BackPSM client decrease dramatically as more clients adopt BackPSM. It indicates that deploying BackPSM is beneficial for both kinds of client.

E. Trace-Driven Experiments

The goal of trace-driven experiments is to study the performance of BackPSM in real scenarios. We randomly deploy clients in the lab room and replay the collected traces.

1) Energy Savings in Real-World: We compare BackPSM with state-of-the-art strategies using traces of synthetic screen-off traffic. In the experiments, we replay the same traces on all clients and measure the energy drain of a Nexus 4 phone. Figure 18(b) shows the results. We see that, as compared to SPSM and BSD, the energy benefit of BackPSM is not appealing when client number is small. In this case, the energy consumption of BackPSM is comparable to that of SPSM and BSD. BackPSM exhibits its advantage on energy saving when network becomes crowded. When client number reaches 6 and 10, BackPSM saves 16.1∼42.5% more energy, as compared to other strategies. This validates the effectiveness of BackPSM in real scenarios.

2) Coexisting with Foreground Traffic: In practice, screen-off traffic often coexist with foreground traffic, i.e., the traffic from active devices (called *foreground client*). In this experiment, we study the impact of foreground traffic on BackPSM performance. We setup 10 devices and vary the ratio of foreground client from 0% to 80%. We employ APSM for foreground clients, and BackPSM for screen-off clients. We measure the energy drain of BackPSM client under different network configurations. Figure 18(c) shows the results. We observe that BackPSM energy increases only a little while the ratio of foreground client increases. This implies that foreground traffic does not impact much on BackPSM.

3) Multi-AP Scenario: Finally, we setup trace-driven experiments to evaluate BackPSM in the multi-AP scenarios. We increase the number of APs from 1 to 3, with each AP serving four clients. We use SleepWell, the best energy saving

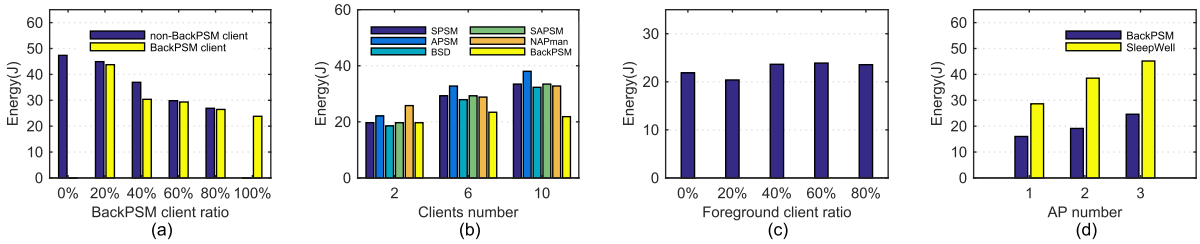


Fig. 18. (a) Energy consumption of BackPSM client vs. non-BackPSM client under different ratio of clients employing BackPSM; (b) Energy comparison of BackPSM with state-of-the-art strategies under real-world screen-off traffic; (c) Energy consumption of BackPSM client under different ratio of foreground clients; (d) BackPSM vs. SleepWell under different number of Wi-Fi APs.

strategy for multi-AP network in the literature, as comparison. Figure 18(d) compares the energy performance of two strategies. The client energy of SleepWell increases from 28.6J to 45.2J, while that of BackPSM increases from 16.0J to 24.6J. BackPSM outperforms SleepWell by up to 45.6%.

VII. Related Work

Energy Optimizing for Screen-Off Traffic: Recent studies [6], [7], [11] have highlighted the energy issues of smartphone during screen-off periods. Huang *et al.* [11] perform the first measurement study on screen-off traffic. They reveal that screen-off traffic accounts for 58.5% of the total network energy consumption. Aucinas *et al.* [3] show that apps maintaining connectivity in the background can drain battery 9 times faster. Reference [6] performs large-scale measurement on the energy drain of 1520 smartphones in the wild. Reference [27] examines the energy characteristic of background traffic through a two-year user study. More recently, [9] and [32] present measurement study on how batteries are consumed when smartphone is in stand-by mode. These studies help us understand the basic energy features of screen-off traffic.

Prior efforts have been made to address the screen-off energy issues. HUSH [7] introduces a metric to measure usefulness of background activities and kills non-useful activities to save screen-off battery. Lee *et al.* [16] develop a context-aware scheduling framework, CAS, to adaptively unload and preload background apps for energy saving. Chakraborty *et al.* [4] propose a method to sense cellular load and schedule background transfers when the cell is idle. Xu *et al.* [35] identify the energy sources of email sync operations and develop techniques to improve the energy performance accordingly. However, these studies focus mainly on the energy consumption of screen-off traffic under 3G/LTE. Peng *et al.* [23] study the power consumption of Wi-Fi interface when smartphone is in suspend mode. They propose to smartly filter out unnecessary Wi-Fi traffic for energy saving. In contrast, our work aims to reduce power consumption incurred by useful screen-off traffic.

Energy Saving in Wi-Fi: Power consumption of Wi-Fi accounts for a significant portion of smartphone system energy. Extensive efforts [13], [29] [30], [31] have been devoted to the measurement study and analytic models of Wi-Fi energy. Reference [13] experimentally studies the relationship between link rate and energy depletion of 802.11n wireless cards. Reference [30] discovers a cross-factor that reflects the processing energy of packet traversing protocol stack and uses this cross-factor to amend traditional energy models. Sun *et al.* [31] propose a throughput-based energy model to capture the complex charateristics of wireless channel. Reference [29] investigates the power consumption of 802.11n/ac new features. These studies provide us insightful guidelines for Wi-Fi energy optimization.

To reduce Wi-Fi energy consumption, a large body of researches focus on sleep scheduling of Wi-Fi radio. The 802.11 standard [1] defines a basic power-save mode (PSM). To bound the traffic delay of standard PSM, BSD [15] dynamically adapts radio's sleep duration according to the traffic activities of client. STPM [2] takes hints from upper-layer apps for sleep scheduling. SAPSM [25] assigns priority to apps and prohibits low-priority app to switch on radio for energy saving. Snooze [12] explores micro-time sleep opportunities (e.g., inter-packet intervals and the period when other peers transfer) and puts radio to sleep during the time. Similarly, μPM [18] and Catnap [8] exploit the bandwidth disparity between wired links, Wi-Fi and user apps to combine small inter-packet gaps into meaningful sleep intervals. SiFi [26] predicts the silence periods during a VoIP call and places radio to sleep when it is silent. All these researches target at energy reduction under regular traffic type that demands short delays. In contrast, our study focus on the energy saving of delay-tolerant screen-off traffic.

Wireless contention emerges as a major energy source when multiple devices coexist in the vicinity. He *et al.* [10] propose Scheduled PSM that divides a beacon interval into multiple time slices and schedules AP to serve each client at appointed slices. However, Scheduled PSM requires modifications to the standard protocol. NAPman [28] leverages virtual APs to isolate PSM clients from each other. SleepWell [22] further extends NAPman to the multi-AP scenario. NAPman and SleepWell require no protocol changes, but need hardware support for AP virtualization and Beacon re-adjusting. HPSM [17] schedule clients with less traffic transferring first to minimize the overall network energy. However, due to stringent delay constraint, the previous strategies merely separate client traffic into different slots within one beacon interval. And they all locate at AP side to centrally coordinate client's traffic schedules. In contrast, our strategy locates at the client side and adopt coarse-grained slots for screen-off traffic isolation, which comes with the advantage of better scalability.

Some researches, HoWiES [39] and Bluesaver [24], use multiple radios of smartphones. They achieve energy saving by

delegating certain power-hungry Wi-Fi operations or transmissions to a low-power radio like ZigBee and Bluetooth. However, HoWiES incurs high complexity on supporting Wi-Fi to ZigBee communication, Bluesaver requires extra Bluetooth radio on the AP. Other studies explore to save Wi-Fi energy by reducing radio's clock rate, i.e., downclocking. E-MiLi [38] reduces the clock rate of radio during idle listening periods, and switches radio to full clock rate for receiving, upon detecting a specially designed PHY preamble. By employing compressive sensing, Slomo [19] allows radio to operate at lower clock rates when transmitting and receiving at low bit rates. Sampless Wi-Fi [33] adopts a rateless-code like idea to recover under-sampled packet (received in downclock mode) from multiple transmissions. These strategies, however, require changes to either the radio hardware or PHY protocol. They cannot be applied to commodity Wi-Fi devices.

Coordination in Wireless Communications: Scheduled PSM [10] and OpenTDMF [36] are two works that enable TDMA in today's Wi-Fi and WLAN networks. They adopt a central controller (e.g., Wi-Fi AP or WLAN back-haul) to coordinate client communications, and require explicit packet exchanges to disseminate coordination information. DOMINO [40] employs a Relative Scheduling technique that adopts the transmissions in a previous slot to trigger transmissions in the next slot, akin to the domino effect. Magistretti *et al.* [21] propose 802.11ec (Encoded Control) as a control-message-free MAC paradigm. 802.11ec employs correlatable symbol sequences (CSSs) and their transmission time to encode control information. Both DOMINO and 802.11ec require MAC/PHY changes. In contrast, our BackPSM works upon standard protocols to realize distributed TDMA-like coordination with no control message exchanges.

Our idea of OBC is partly inspired by studies in Cross-Technology Communication (CTC) [5], [14], [37]. To disseminate control information from Wi-Fi AP to ZigBee radio, HoWiES [39] builds a WiFi-ZigBee side channel by using different Wi-Fi transmission patterns. Lu and Gao [20] exploit the in-band SNR margin to encode data as patterned interference, and create a side channel that operates concurrently with the OFDM main channels. In contrast, our OBC establishes a side channel for only homogeneous Wi-Fi clients. OBC brings a new Client-to-Client communication paradigm, akin to Wi-Fi Direct, to a Wi-Fi network, in addition to the traditional AP-Client communication. Different from Wi-Fi Direct, OBC requires no extra traffic on Wi-Fi communication band.

VIII. Discussions

In this section, we discuss the limitations of our strategy and possible solutions.

Firstly, the batch sending and periodic scheduling schemes of BackPSM would introduce long latency to client traffics. This may not introduce big issues for the screen-off traffics which are typically delay-tolerant since users generally do not interact with their smartphones during the screen off period. In case that some screen-off traffics are delay-sensitive, we can differentiate such traffics and only apply BackPSM to delay-tolerant traffics.

Secondly, the maximum number of clients supported by a collision-free schedule is limited by the number of slots in a period, i.e., capacity. When the number of clients exceeds the limit, BackPSM can increase the period for larger capacity. However, it will lead to longer traffic delays. To avoid traffics being delayed for too long, we can introduce an upper-bound on the scheduling period and allow multiple clients to share a slot. In this case, the energy benefits may degrade slightly. We need to trade-off between the energy and delay performance.

Finally, the out-of-band coordination scheme adopts a distributed algorithm for Wi-Fi coordination. It exchanges traffic pattern information over the TIM side channel and locally resolves collisions, which takes long time to converge (e.g., about 5 periods when 8 clients collide). This may not be a big issue for screen-off traffics, because it is more important to reduce control overhead and save power when smartphone stays in screen-off/standby mode.

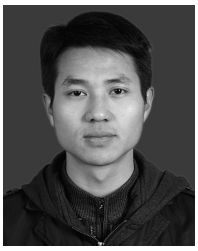
IX. Conclusion

In this work, we address the energy issues of Wi-Fi PSM on handling screen-off traffic. We propose a new power saving strategy—BackPSM. The strategy regulates traffic in batches and coordinates client communication with collision-free schedules like TDMA. The key innovation is the Out-of-Band Communication paradigm (OBC) that enables client-to-client direct communications through a TIM side-channel. We employ OBC in BackPSM to exchange information of client schedules beyond the Wi-Fi communication band, realizing out-of-band coordination among clients. Extensive testbed evaluations show that BackPSM client can decode the traffic pattern of peers using OBC with close to 100% reliability and establish collision-free schedules in 10 periods. BackPSM reduces screen-off system energy by up to 60% and outperforms state-of-the-art strategies.

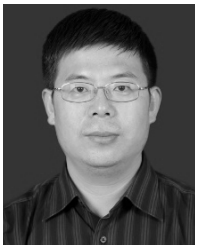
References

[1] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11, 2012.
[2] M. Anand, E. B. Nightingale, and J. Flinn, "Self-tuning wireless network power management," in *Proc. MobiCom*, Sep. 2003, pp. 176–189.
[3] A. Aucinas *et al.*, "Staying online while mobile: the hidden costs," in *Proc. CoNEXT*, Dec. 2013, pp. 315–320.
[4] A. Chakraborty, V. Navda, V. N. Padmanabhan, and R. Ramjee, "Coordinating cellular background transfers using loadsense," in *Proc. MobiCom*, Sep. 2013, pp. 63–74.
[5] K. Chebrolu and A. Dhekne "Esense: Communication through energy sensing," in *Proc. ACM MobiCom*, Sep. 2009, pp. 85–96.
[6] X. Chen, N. Ding, A. Jindal, Y. C. Hu, M. Gupta, and R. Vannithamby, "Smartphone energy drain in the wild: Analysis and implications," in *Proc. ACM SIGMETRICS*, Jun. 2015, pp. 151–164.
[7] X. Chen, A. Jindal, N. Ding, Y. C. Hu, M. Gupta, and R. Vannithamby, "Smartphone background activities in the wild: Origin, energy drain, and optimization," in *Proc. ACM MobiCom*, Sep. 2015, pp. 40–52.
[8] F. R. Dogar, P. Steenkiste, and K. Papagiannaki, "Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices," in *Proc. MobiSys*, Jun. 2010, pp. 107–122.
[9] Y. Guo, C. Wang, and X. Chen, "Understanding application-battery interactions on smartphones: A large-scale empirical study," *IEEE Access*, vol. 5, pp. 13387–13400, Jul. 2017.
[10] Y. He, R. Yuan, and W. Gong, "Modeling power saving protocols for multicast services in 802.11 wireless LANs," *IEEE Trans. Mobile Comput.*, vol. 9, no. 5, pp. 657–671, May 2010.
[11] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck. "Screen-off traffic characterization and optimization in 3G/4G networks," in *Proc. ACM IMC*, Nov. 2012, pp. 357–363.
[12] K.-J. Jang, S. Hao, A. Sheth, and R. Govindan "Snooze: Energy management in 802.11n WLANs," in *Proc. CoNEXT*, Dec. 2011, Art. no. 12.

[13] M. O. Khan *et al.*, "Model-driven energy-aware rate adaptation," in *Proc. ACM MobiHoc*, Jul. 2013, pp. 217–226.
[14] S. M. Kim and T. He, "FreeBee: Cross-technology communication via free side-channel," in *Proc. ACM MobiCom*, Sep. 2015, pp. 317–330.
[15] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless Web access with bounded slowdown," in *Proc. MobiCom*, Sep. 2002, pp. 119–130.
[16] J. Lee, K. Lee, E. Jeong, J. Jo, and N. B. Shroff, "CAS: Context-aware background application scheduling in interactive mobile systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1013–1029, May 2017.
[17] D. Liu, H. Wang, G. Zhou, W. Mao, and B. Li, "Arbitrating traffic contention for power saving with multiple PSM clients," *IEEE Trans. Wireless Commun.*, vol. 15, no. 10, pp. 7030–7043, Oct. 2016.
[18] J. Liu and L. Zhong, "Micro power management of active 802.11 interfaces," in *Proc. MobiSys*, Jun. 2008, pp. 146–159.
[19] F. Lu, G. M. Voelker, and A. C. Snoeren, "Slomo: Downclocking WiFi communication," in *Proc. USENIX NSDI*, Apr. 2013, pp. 255–268.
[20] H. Lu and W. Gao, "Supporting real-time wireless traffic through a high-throughput side channel," in *Proc. ACM MobiHoc*, Jul. 2016, pp. 311–320.
[21] E. Magistretti, O. Gurewitz, and E. W. Knightly, "802.11ec: Collision avoidance without control messages," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 1845–1858, Dec. 2014.
[22] J. Manweiler and R. R. Choudhury, "Avoiding the rush hours: WiFi energy management via traffic isolation," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 739–752, May 2012.
[23] G. Peng, G. Zhou, D. T. Nguyen, and X. Qi "All or none? The dilemma of handling WiFi broadcast traffic in smartphone suspend mode," in *Proc. IEEE INFOCOM*, Apr./May 2015, pp. 1212–1220.
[24] A. Pyles, D. T. Nguyen, X. Qi, and G. Zhou, "Bluesaver: A multi-PHY approach to smartphone energy savings," *IEEE Trans. Wireless Commun.*, vol. 14, no. 6, pp. 3367–3377, Jun. 2015.
[25] A. J. Pyles, X. Qi, G. Zhou, M. Keally, and X. Liu, "SAPSM: Smart adaptive 802.11 PSM for smartphones," in *Proc. ACM UbiComp*, Sep. 2012, pp. 11–20.
[26] A. J. Pyles, Z. Ren, G. Zhou, and X. Liu, "SiFi: exploiting VoIP silence for WiFi energy savings insmart phones," in *Proc. UbiComp*, Sep. 2011, pp. 325–334.
[27] S. Rosen *et al.*, "Revisiting network energy efficiency of mobile apps: Performance in the wild," in *Proc. ACM SIGCOMM IMC*, Oct. 2015, pp. 51–60.
[28] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, "NAPman: Network-assisted power management for WiFi devices," in *Proc. MobiSys*, Jun. 2010, pp. 91–106.
[29] S. K. Saha, P. Deshpande, P. P. Inamdar, R. K. Sheshadri, and D. Koutsonikolas, "Power-throughput tradeoffs of 802.11n/ac in smartphones," in *Proc. IEEE INFOCOM*, Apr./May 2015, pp. 100–108.
[30] P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs, and A. Azcorra, "Per-frame energy consumption in 802.11 devices and its implication on modeling and design," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1243–1256, May 2014.
[31] L. Sun, H. Deng, R. K. Sheshadri, W. Zheng, and D. Koutsonikolas, "Experimental evaluation of WiFi active power/energy consumption models for smartphones," *IEEE/ACM Trans. Mobile Comput.*, vol. 16, no. 1, pp. 115–129, Jan. 2017.
[32] C. Wang, Y. Guo, Y. Xu, P. Shen, and X. Chen, "Standby energy analysis and optimization for smartphones," in *Proc. MobiCloud*, Mar./Apr. 2016, pp. 11–20.
[33] W. Wang, Y. Chen, L. Wang, and Q. Zhang, "Sampleless Wi-Fi: Bringing low power to Wi-Fi communications," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1663–1672, Jan. 2017.
[34] X. Xia *et al.*, "Surviving screen-off battery through out-of-band Wi-Fi coordination," in *Proc. INFOCOM*, May 2017, pp. 1–9.
[35] F. Xu *et al.*, "Optimizing background email sync on smartphones," in *Proc. ACM MobiSys*, Jun. 2013, pp. 55–68.
[36] Z. Yang, J. Zhang, K. Tan, Q. Zhang, and Y. Zhang, "Enabling TDMA for today's wireless LANs," in *Proc. IEEE INFOCOM*, Apr./May 2015, pp. 1436–1444.
[37] X. Zhang and K. G. Shin, "Gap sense: Lightweight coordination of heterogeneous wireless devices," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 3093–3101.
[38] X. Zhang and K. G. Shin, "E-MiLi: Energy-minimizing idle listening in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 9, pp. 1441–1454, Sep. 2012.
[39] Y. Zhang and Q. Li, "Exploiting ZigBee in reducing WiFi power consumption for mobile devices," *IEEE Trans. Mobile Comput.*, vol. 13, no. 12, pp. 2806–2819, Dec. 2014.
[40] W. Zhou, D. Li, K. Srinivasan, and P. Sinha, "DOMINO: Relative scheduling in enterprise wireless LANs," in *Proc. ACM CoNEXT*, Dec. 2013, pp. 381–392.



Xianjin Xia (S'15) received the B.S. degree in software engineering and the M.Sc. and Ph.D. degrees in computer science from Northwestern Polytechnical University, Xi'an, China, in 2010, 2013, and 2018, respectively. His research interests include wireless communication and networking, Internet of Things, and mobile computing. He is a Student Member of IEEE.



Shining Li (M'15) received the B.S. and M.S. degrees in computer science from Northwestern Polytechnical University, Xi'an, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, in 2005. He is currently a Professor with the School of Computer Science, Northwestern Polytechnical University. His research interests include mobile computing and wireless sensor networks. He is a member of the IEEE.

Yu Zhang (M'15) received the Ph.D. degrees in computer science from Northwestern Polytechnical University, Xi'an, China, and RMIT University, Melbourne, Australia. He was a Scholarship Researcher with The University of Melbourne. He is currently an Associate Professor with the School of Computer Science, Northwestern Polytechnical University. His research areas include protocol design, wireless sensor networks, mobile computing, and Internet of Things. He is a member of the IEEE and the ACM.

Bingqi Li received the B.S. degree in computer science from Northwestern Polytechnical University, Xi'an, China, where he is currently pursuing the master's degree. His research interests include Internet of Things and mobile computing.

Yuanqing Zheng (M'14) received the B.S. degree in electrical engineering and the M.E. degree in communication and information system from Beijing Normal University, Beijing, China, in 2007 and 2010, respectively, and the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, in 2014. He is currently an Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University. His research interest includes mobile and network computing, RFID systems, and Internet of Things. He is a member of IEEE and ACM.

Tao Gu (SM'14) received the bachelor's degree from the Huazhong University of Science and Technology, the M.Sc. degree from Nanyang Technological University, Singapore, and the Ph.D. degree in computer science from the National University of Singapore. He is currently an Associate Professor in computer science and IT with RMIT University, Australia. His research interests include mobile computing, ubiquitous computing, wireless sensor networks, sensor data analytics, and Internet of Things. He is a Senior Member of IEEE and a member of ACM.