BLEND: BLE Beacon-Aided Fast WiFi Handoff for Smartphones

Junyoung Choi, Gyujin Lee, Yeonchul Shin, Jonghoe Koo, Minseok Jang, and Sunghyun Choi
Department of ECE and INMC, Seoul National University, Seoul, Korea
{jychoi, gjlee, ycshin, jhkoo, msjang}@mwnl.snu.ac.kr, schoi@snu.ac.kr

Abstract—Recently, WiFi has become irreplaceable wireless technology to support increasing mobile applications and data traffic. While today’s smartphone users frequently use WiFi on the move, smartphone does not perform WiFi handoff to nearby access points (APs) even if signal quality is severely degraded. Such undesired operation of smartphone is known as sticky client problem. In this work, we focus on resolving sticky client problem through fast WiFi handoff. First, we analyze the causes of sticky client problem based on experiments with commercial Android smartphones by focusing on WiFi scanning and handoff operations. Note that smartphones and state-of-the-art APs are equipped with Bluetooth Low Energy (BLE), we propose a practical solution, called BLEND, utilizing BLE to enable fast WiFi handoff. Smartphone acquires information about nearby APs through BLE advertising packet sent by APs, and then judiciously performs WiFi handoff. We implement BLEND on Android smartphones and demonstrate that BLEND achieves up to 61% and 111% higher throughput and video bitrate, respectively, compared with a commercial Android application.

I. INTRODUCTION

Over the past few years, the extreme proliferation of smartphones has made WiFi an indispensable wireless technology in our daily lives. WiFi is expected to serve as a hub for immense mobile applications and data traffic in the future [1]. Well-known mobile applications include twittering, video streaming, and Voice over IP, and smartphone users frequently use these applications on the move.

When smartphone users roam while using mobile applications, handoff between WiFi access points (APs) should be conducted in a timely manner in order to provide continuous service. However, we frequently experience that smartphone maintains AP connection with very low signal strength even though nearby APs can provide much higher signal strength, which means smartphone does not perform handoff properly. Such undesirable behavior of smartphone, known as sticky client problem [2], has been considered one of the most annoying problems that causes smartphone users to simply deactivate WiFi. To overcome the sticky client problem, smartphone needs to perform fast handoff on time to nearby APs which can provide higher throughput.

In relation to the aforementioned problem, there have been several studies to trigger fast handoff, by utilizing only a single or multiple WiFi interfaces. The first approach exploits only a single WiFi interface to minimize WiFi scanning and handoff delay [3–7]. However, to find nearby APs relying on a single WiFi interface, it is needed to scan all WiFi channels even if there is no available AP, since it has no prior knowledge of when, i.e., exact timing to scan, and where, i.e., operating channel of APs, to scan. The second approach leverages multiple WiFi interfaces. That is, the secondary WiFi interface is dedicated to scanning purpose to reduce scanning delay [8, 9] or pre-association with nearby APs [10]. Using multiple WiFi interfaces is inapplicable to smartphones, since commercial smartphones have only a single WiFi interface. As a result, with the above approaches, there is no way to acquire prior knowledge of nearby APs without WiFi scanning.

However, recent studies show that it is possible to detect nearby APs by utilizing a collocated wireless personal area network (WPAN) radio such as Bluetooth and ZigBee [11–14]. The motivation of the existing work is to minimize energy consumption caused by unnecessary WiFi scanning. Accordingly, as the above approach, WPAN radio can give clue to find nearby APs before handoff and provides the possibility to overcome the limitation of smartphone equipped with a single WiFi interface. In addition, state-of-the-art APs such as Samsung Connect Home [15] and Google WiFi AP [16] are embedded with Bluetooth 4.1 and Zigbee as well as WiFi. Therefore, we expect there is a chance to exploit WPAN.

Motivated by the fact that handoff dedicated to smartphone should be supported, we set our goal as designing a practical and simple handoff scheme, which is directly applicable to smartphones. In this paper, we propose fast handoff scheme called BLEND exploiting Bluetooth Low Energy (BLE) as secondary radio. Each AP collocated with BLE periodically transmits BLE advertising packet containing its information such as operating channel and channel utilization. Upon an advertising packet reception, smartphone can acquire prior knowledge about existence and information of nearby APs without WiFi scanning. After that, if handoff is needed, smartphone performs WiFi scanning and estimates throughput using received signal strength indicator (RSSI) and channel utilization. Based on the estimated throughput, smartphone performs handoff to the nearby AP if it can provide better throughput.

We envision the approach of BLEND to be increasingly practical and feasible for the following reasons. First, many of today’s APs mentioned above are equipped with WPAN modules. Furthermore, legacy APs without BLE modules can be easily connected with BLE module [17] through a USB port. Second, most commercial smartphones are equipped with Bluetooth module. BLE is supported beginning Android 4.3 Jelly Bean launched in 2013, and no hardware modification is
required on smartphone. Third, the application-layer solution is possible through user-friendly Android application programming interfaces (APs) of BLE.

Our major contributions are summarized as follows:

- We conduct detailed analysis of the sticky client problem on commercial smartphones with experiment and close examination of Android source code. Through the comprehensive study, we figure out that the sticky client problem is caused by 1) unacceptably long WiFi scanning interval and 2) lack of handoff mechanism.
- We propose BLEND, which exploits BLE modules to provide smartphones with prior knowledge of the presence and information of APs operating at both 2.4 and 5 GHz WiFi channels. BLEND operating with only application requires no hardware and Android source code modification of smartphone. To our best knowledge, BLEND is the first BLE-aided handoff scheme.
- We also propose an advanced version of BLEND that can be applied to smartphone enabling hidden Android API, which optimizes WiFi scanning through modification of Android source code.
- We prototype BLEND with commercial smartphones and evaluate the performance in real environments. Our measurement results demonstrate that BLEND significantly improves throughput and video bitrate by up to 61% and 111%, compared to a commercial Android application, respectively, with negligible energy overhead.

The rest of the paper is organized as follows. Related work is summarized in Section II. Background knowledge is presented in Section III, and the sticky client problem is analyzed in Section IV. We present detailed operation of BLEND in Section V, and evaluate the performance of BLEND in real environments in Section VI. Finally, we conclude our work in Section VII.

II. RELATED WORK

A. WiFi-based Handoff

SyncScan [7] periodically scans WiFi channels to obtain information about nearby APs before handoff. DeuceScan [6] uses spatiotemporal graph to predict the next AP. As a similar approach, D-Scan [4] and Proactive Scan [5] eliminate WiFi scanning delay by actively probing all WiFi channels. However, since above schemes have no prior knowledge of nearby APs, they cannot prevent unnecessary WiFi scanning.

SWIMMING [18] and WGTT [19] focus on fast vehicular handoff. SWIMMING supports seamless WiFi-based Internet access with “group unicast” manner. All APs have to be configured with the same MAC and IP addresses to provide group unicast. WGTT proposes fine-grained AP selection and queue management algorithm in picocell environment. It is difficult to directly apply in practice because additional infrastructure and hardware/software modification of AP are inevitable to implement WGTT.

802.11k [20] allows an associated AP to provide site report of potential candidate APs according to the movement of a mobile station (STA). On the other hand, 802.11r [21] optimizes authentication process to remove authentication delay. When STA enters mobility domain, it finishes authentication process before actual handoff occurs. However, 802.11r is not widely adopted because it focused on enterprise network, which needs infrastructure, and is unsuitable for home deployment [22].

Another approach to seamless handoff makes simultaneous connection with multiple APs in link or transport layer. MultiNet (also known as VirtualWiFi [23]) and Juggler [24] allow STA to associate with multiple APs on different WiFi channels in a round robin manner. FatVAP [25] focuses on balancing the traffic load of multiple APs. However, above approaches cannot be implemented in application layer. In Critoru et al. [26], STA associates with multiple APs through MPTCP subflow for each AP. Authors argue that MPTCP is available in the iOS 7. However, MPTCP in iOS uses cellular and a single AP rather than multiple APs.

MultiScan [10] leverages extra WiFi interface to make a connection to a candidate AP before disconnecting the current one. The commercial smartphone has only a single WiFi interface and adding an additional WiFi interface is impractical. In [8, 9], AP is equipped with multiple WiFi interfaces, where one interface is set to operate as an exclusively reserved channel for the WiFi scanning purpose. The STA has to know the reserved channel of candidate AP in advance.

All the above approaches cannot be directly applied to smartphone because they need modified hardware or source code to be implemented on smartphone. In BLEND, smartphone can find nearby APs and perform handoff only through the application. No hardware and Android source code modification is required, and BLEND can also coexists with other BLE or Classic Bluetooth (BT) connections with smart devices, e.g., smart band or headphone.

B. WPAN-aided AP Discovery

The motivation for exploiting low-power WPAN technologies for WiFi is to reduce large energy consumption caused by unnecessary WiFi scanning. Especially, recent several researches have addressed the problem of finding nearby APs through WPAN. In [11–13], WPAN is exploited to predicts the existence of nearby APs by detecting the signature of periodic WiFi beacon frame through RSSI sampling. These schemes require computational cost and have false positive to detect AP depending on WiFi channel utilization. Since they ceaselessly conduct RSSI sampling, detecting AP is impossible when WPAN connection exists. In addition, in the case of Zigbee, an external transceiver is needed on smartphone. Blue-Fi [14] predicts nearby APs based on history of heterogeneous network connections and location. However, Blue-Fi is difficult to operate in an unknown environment without history. In conclusion, none of the above approaches deals with fast handoff, and there is no handoff scheme exploiting BLE.

III. BACKGROUND

A. Handoff Procedure in IEEE 802.11

IEEE 802.11 standard defines handoff procedure as the following three phases: scanning, authentication, and re-
association. In scanning phase, STA scans WiFi channels to find APs in vicinity and to acquire information such as RSSI of APs [5]. If STA determines an AP to perform handoff, STA and the AP exchange authentication request and response during authentication phase. After authentication phase, re-association request and response are exchanged during re-association phase, then handoff procedure finishes. Handoff delay represents time between scanning phase and re-association phase. Typically, both authentication and re-association phases are less than 20 ms [7]. The scanning phase empirically accounts for 95% of handoff delay [27] and has been a goal to minimize for fast handoff in literature.

B. BSS Load Element in IEEE 802.11

IEEE 802.11e [28] defines Basic Service Set (BSS) Load element field, which can be optionally included in beacon and probe response. BSS Load element is composed of station count, channel utilization, and available admission capacity. Station count indicates a total number of STAs currently associated with an AP. Channel utilization is defined as the ratio of time that the AP sensed the medium was busy. Available admission capacity defines remaining amount of time via explicit admission control. BSS Load element information can be used by STAs when performing handoff.

C. Bluetooth Low Energy

Bluetooth 4.0 specification [29] defines Classic BT and BLE. BLE is a low-power short-range wireless communication technology operating on 2.4 GHz unlicensed band. BLE uses 40 physical channels separated by 2 MHz from 2402 MHz to 2480 MHz. Three of 40 channels are used as advertising channels with center frequencies of 2402, 2426, and 2480 MHz. BLE beacon, which represents device equipped with BLE module, periodically broadcasts advertising packet on advertising channels. The interval of advertising packet is in range of 100 ms to 10.24 s.

Advertising packet is composed of preamble, access address, cyclic redundancy check (CRC), and packet data unit (PDU). Preamble is used for detection of advertising packet, synchronization, and automatic gain control. Access address defines the type of BLE packet, e.g., advertising packet. CRC is used to check the validation of advertising packet. PDU includes header field which contains MAC address of BLE beacon, advertising data, and information about advertising packet, e.g., transmission power and packet length.

IV. STICKY CLIENT PROBLEM

In this section, we experimentally confirm that the sticky client problem occurs on commercial smartphones. Moreover, we analyze the causes of the sticky client problem by investigating experiment results and Android source code [30], especially focusing on WiFi scanning and handoff operations.

A. Sticky Client Problem of Commercial Smartphone

We conduct an experiment in a controlled office environment where floor plan is illustrated in Fig. 1. AP 1 and AP 2 operate on different channels of 5 GHz and transmission power of both APs is set to the maximum value of 15 dBm. We use Google Nexus 5 with Android 6.0.1 Marshmallow for the experiment. The smartphone is initially located at P1 for 20 s, moves from P1 to P2 with an average walking speed of 1 m/s, and then stays at P2 for the last 20 s. In scenario 1 (S1) and scenario 2 (S2), smartphone is initially associated with AP 1 and AP 2, respectively. Both APs generate saturated UDP traffic only when smartphone is associated. We measure throughput and RSSI at smartphone every second during 60 s and repeat measurement five times.

Figs. 2(a) and 2(b) represent average throughput and RSSI in time domain. In S1, throughput and RSSI decrease as smartphone moves away from AP 1. When smartphone reaches P2, we expect that smartphone performs handoff to AP 2. However, smartphone never performs handoff to AP 2 even though RSSI of AP 1 is −81 dBm and RSSI of AP 2 is 15 dBm higher. Similar to above phenomenon, smartphone does not perform handoff to AP 1 at first 20 s in S2. We observe the sticky client problem in both scenarios. The smartphone does not perform handoff even if RSSI of the associated AP is extremely low and there exists another AP which can support higher RSSI. If smartphone performs handoff as Ideal, throughput can be enhanced by 65.7% as shown in Fig. 2(c).

B. Cause of Sticky Client Problem

To analyze causes of the sticky client problem, we examine conditions to trigger WiFi scanning through measurement and Android source code. Next, we investigate whether smartphone performs handoff when WiFi scanning is triggered.

1) Scanning operation: To examine WiFi scanning operation in Android, we investigate the source code of Android 6.0.1 Marshmallow. The scanning can be triggered by two different roots: WiFiService and startScan. WiFiService is the system service of Android, and startScan is Android API. Both of them trigger WiFi scanning as an active scanning by default. During the active scanning, STA transmits a probe request and listen for probe responses from APs. WiFiService starts WiFi scanning when the screen of smartphone turns on. The WiFi scanning interval starts at 40 s, increases to 60, 80, 120, 160, 240, 360 s, and then maintains 360 s. We also double-check that smartphone performs WiFi scanning with the above intervals through packet capture using Airpcap [31]. However, WiFiService never performs WiFi scanning while the screen
turns off. In case of the startScan, it triggers WiFi scanning whenever called by Android application manually. Therefore, smartphone depends on WifiService for WiFi scanning with long time intervals if startScan is not called by application. As a result, WiFi scanning operation of Android is insufficient to support fast handoff on time without startScan.

2) Handoff operation: We conduct an experiment to observe handoff operation of Android smartphones when scanning is manually triggered. Google Nexus 5 and Samsung Galaxy S7 are used for the experiment. The smartphones are located at P1 in Fig. 1 and associated with AP 2 for 60 s. We trigger startScan by force with the minimum WiFi scanning interval of the API. The minimum WiFi scanning intervals of Nexus 5 and Galaxy S7 are 1 s and 3.5 s, respectively. During the experiment, smartphones do not perform handoff to AP 1 even though average RSSI of AP 2 is $-81$ dBm while that of AP 1 is $-44$ dBm. As a result, the sticky client problem cannot be resolved by triggering WiFi scanning, and hence, appropriate handoff operation is absolutely necessary.

V. BLEND: PROPOSED SCHEME

It is essential to perform WiFi scanning and handoff at appropriate timing to overcome the sticky client problem. In BLEND, smartphone can explicitly acquire the existence and information about candidate AP through reception of advertising packet. By exploiting acquired information such as the channel number and channel utilization, smartphone determines the best AP to perform handoff by estimating performance improvement. We also proposed an advanced version of BLEND, which selectively scans WiFi channel where a candidate AP exists.

A. Advantages and Necessities of BLE as Secondary Low-Power Radio

The main concept of BLEND is that BLE beacon collocated with AP periodically transmits advertising packet to notify the existence of AP. Considering that BLE is short-range wireless technology compared with WiFi, receiving advertising packet on smartphone can guarantee that there is AP in the vicinity without WiFi scanning. Therefore, BLE can overcome the limitation that smartphone has no prior knowledge of nearby APs before WiFi scanning caused by single WiFi interface. In addition, information such as channel utilization to help handoff decision is conveyed through the payload of advertising packet. Including the information of AP into the payload of advertising packet is the most important factor to allow BLEND to work in application layer. The payload of advertising packet can be delivered directly to application layer through BLE Android API. Therefore, BLEND can operate without any additional hardware and Android framework modification. Since BLEND makes no BLE connection, it can operate without interfering other BLE connections, e.g., wearable devices.

In fact, it is possible to design BLEND through only WiFi without BLE. BSS load element including channel utilization can be optionally contained in WiFi beacon. However, using WiFi alone has two drawbacks. First, smartphone should perform WiFi scanning in a periodic or event-driven manner to find nearby APs. Second, even if WiFi beacon contains channel utilization, there is no direct root to deliver information in WiFi beacon to application layer. In order to utilize information in WiFi beacon, firmware modification of smartphone is inevitable. We rule out such approach because it is impractical and not feasible to modify the firmware of all smartphones.

B. Overall Architecture

Fig. 3 shows the overall architecture of BLEND. In AP, available channel occupancy ratio (ACOR) is updated in ACOR updater. ACOR conceptually represents channel idle ratio (CIR). BLE beacon periodically transmits advertising packet containing information of AP including ACOR. When BLE receiver in smartphone receives advertising packet from BLE beacon, WiFi scanner triggers scanning event depending on scanning allowance indicator (SAI). SAI adaptively allows WiFi scanner to perform scanning event. Based on RSSI from scanning result, achievable PHY rate ($aR$) is selected. Throughput estimator calculates achievable throughput ($aTH$) and effective ACOR ($eACOR$) of AP using $ACOR$, $aR$, and ongoing airtime ratio (OAR). OAR conceptually represents the ratio of airtime consumed by smartphone itself. Finally, decision maker determines whether to perform scanning and handoff using $aTH$ and $eACOR$, and allows WiFi scanner to scan channel depending on SAI.
Eddystone header format, and hence, smartphone can receive the proposed advertising packet with AP address. If AP address is the same as MAC address of WiFi scanner operates in two ways based on RSSI update:

D. Smartphone Operation

1) ACOR updater: To allow smartphone to determine whether or not to perform handoff based on the expected performance improvement, ACOR updater calculates ACOR every second. We first define channel idle ratio (CIR) which represents the ratio of channel idle time during the last 1 s.

\[
CIR = 1 - T_{bus \gamma},
\]

where \( T_{bus \gamma} \) is channel busy ratio during the last 1 s which includes the time when AP sends and receives packets. To avoid fluctuation of ACOR due to burst traffic, ACOR is defined as the exponentially weighted moving average (EWMA) of CIR as follows:

\[
ACOR = \alpha \cdot CIR + (1 - \alpha) \cdot ACOR.
\]

2) BLE beacon: BLE beacon takes ACOR from ACOR updater whenever ACOR is updated, and periodically transmits proposed advertising packet. Fig. 4 shows proposed advertising packet format. The proposed format basically follows Eddystone beacon format, which is open BLE beacon format from Google [32]. We modify advertising data field to contain information of AP. The proposed advertising data field includes Eddystone header, identifier (ID), channel, ACOR, and AP address.1 Eddystone header is required to maintain Eddystone advertising packet format. ID is needed to verify proposed advertising packet. Channel and AP address indicate the channel number and MAC address of AP, respectively. The rest of data field (11 B) is reserved, which can contain more information about AP.

D. Smartphone Operation

1) BLE receiver: It receives advertising packet through monitoring three BLE advertising channels. Upon receiving advertising packet, BLE receiver checks ID field to confirm whether received advertising packet is proposed advertising packet of BLEND. If receiving proposed advertising packet, BLE receiver reads channel, ACOR, and AP address fields.

2) WiFi scanner: It takes AP information whenever BLE beacon receives proposed advertising packets and updates RSSI of AP to determine achievable PHY rate (\( aR \)).

RSSI update: WiFi scanner operates in two ways based on AP address. If AP address is the same as MAC address of associated AP (\( AP_a \)), WiFi scanner skips scanning because RSSI of \( AP_a \) is periodically updated without extra scanning.2 On the other hand, if AP address is different from \( AP_a \’s, it means that candidate AP (\( AP_c \)) exists in the proximity. Therefore, WiFi scanner carries WiFi scanning out to update RSSI of \( AP_c \). In the advanced version of BLEND, WiFi scanning is performed in only one channel that \( AP_c \) is operating.

Achievable PHY rate: is selected for a given RSSI after updating RSSI of AP. We use receive (Rx) sensitivity values of BCM4339 WiFi chipset [33] for IEEE 802.11n. We employ the Rx sensitivity values of BCM4339 because Nexus 5, a primary smartphone for performance evaluation, is equipped with BCM4339.3 The Rx sensitivity represents the minimum RSSI value that satisfies 90% packet delivery ratio (PDR) for each PHY rate as shown in Table I. \( aR \) is selected as the maximum PHY rate that satisfies PDR 90% for a given RSSI. For example, if RSSI is \(-76 \text{ dBm}, 58.5 \text{ Mb/s} \) is selected.4

3) OAR updater: ACOR of \( AP_a \) is affected by its own traffic on the smartphone. Throughput of \( AP_a \) cannot be determined directly by using ACOR of \( AP_a \). We define ongoing airtime ratio (OAR) which is the ratio of airtime consumed by transmission and reception of itself during the last 1 s. OTA updater calculates OAR every second using EWMA with the same weighting factor (\( \alpha \)) as that used for ACOR to maintain consistency between OAR and ACOR:

\[
OAR = \alpha \cdot \left( \frac{txBits + rxBits}{aR_a} \right) \cdot \frac{1}{t} + (1 - \alpha) \cdot OAR,
\]

where \( txBits \) and \( rxBits \) are the number of transmitted and received bits through WiFi during the last \( t \) s, respectively, and \( aR_a \) is the latest achievable PHY rate of \( AP_a \). We set \( t \) to 1 s. Note that OAR updater at smartphone does not need to be synchronized with ACOR updater at AP. In the case of \( AP_c \), OAR is zero because no data is exchanged between \( AP_c \) and the smartphone.

4) Throughput estimator: calculates achievable throughput (\( aTH \)) using \( aR \), ACOR, and OAR.

Achievable throughput: The operation of throughput estimator differs from \( AP_a \) to \( AP_c \). When we estimate \( aTH \) of \( AP_a \), OAR should be added to ACOR to reflect the airtime used by smartphone running throughput estimator. We here define

\[ \text{Rx sensitivity (dBm): } -94, -91.5, -89.2, -86.1, -82.8, -77.9, -74.3, -71.1 \]

2RSSI of current associated AP is updated every 3 s in Android by default.
3It is also possible to employ different Rx sensitivity values for different smartphone model depending on the embedded WiFi chipset. However, using a common set of values seems to be also acceptable as we observe that the performance is not much affected even if we use a common set for a different smartphone in Section VI.
4In fact, there is application-level Android API named getLinkSpeed, which returns \( aR \) in Mb/s of only associated AP. However, we observe that the API cannot reflect real-time \( aR \) transition, and hence, we use the Rx sensitivity table. Also, the API cannot provide \( aR \) of candidate APs.
effective ACOR \( (eACOR) \), which is the sum of ACOR and OAR, as follows:

\[
eACOR = ACOR + OAR.
\]

In case of \( AP_e \), \( eACOR \) is same as \( ACOR \) because \( OAR \) of \( AP_e \) is zero. After obtaining \( eACOR \), \( aTH \) is calculated by a product of \( aR \) and \( eACOR \):

\[
aTH = aR \cdot eACOR,
\]

where \( aTH \) conceptually represents the throughput when smartphone fully utilizes the channel during idle time as \( eACOR \) with \( aR \). After calculating \( aTH \), Throughput estimator updates both \( aTH \) and \( eACOR \) of AP.

5) Decision maker: makes handoff and scanning decisions based on \( aTH \).

Handoff decision: BLEND determines whether to perform handoff to \( AP_e \) based on \( aTH \). To avoid ping-pong effect, previous approaches [7, 34] utilize handoff hysteresis (\( \Delta \)). BLEND also utilizes hysteresis so that smartphone performs handoff if the following condition is satisfied:

\[
aTH_a + \Delta < aTH_e,
\]

where \( aTH_a \) and \( aTH_e \) are achievable throughput of \( AP_a \) and \( AP_e \), respectively. If \( aTH_e \) is higher than the sum of \( aTH_a \) and \( \Delta \), the smartphone performs handoff. In BLEND, \( \Delta \) is calculated as follows:

\[
\Delta = eACOR_a \cdot \delta,
\]

where \( eACOR_a \) is effective ACOR of \( AP_a \) and \( \delta \) is 6.5 Mb/s, which is the minimum throughput difference between adjacent PHY rates as shown in Table I. Therefore, handoff is triggered only when \( aTH_e \) is larger than \( aTH \) obtained with one step higher achievable PHY rate of \( AP_a \) for a given \( eACOR_a \).

Scanning decision: Thanks to advertising packet from \( AP_e \), smartphone can explicitly recognize the existence of the \( AP_e \). However, if smartphone triggers scanning event every time it receives advertising packet, unnecessary scanning event can be triggered. Therefore, we define scanning allowance indicator (SAI) to avoid unnecessary scanning event, which is set to true if the following scanning condition (8) is satisfied.

\[
aTH_a + \Delta < rTH,
\]

where \( \Delta \) is the same hysteresis used at handoff decision, and \( rTH \) is a reference throughput. Scanning condition implies that scanning event can be triggered only when \( aTH_a \) is significantly lower than \( rTH \). It is not desirable to set a constant \( rTH \) because the required throughput differs for service type in smartphone. Therefore, BLEND needs to adapt \( rTH \) to trigger scanning event depending on the situation, and hence, \( rTH \) is controlled by the following two ways.

First, when smartphone triggers scanning event but (6) is not satisfied, it means that there is no appropriate AP to perform handoff. In this case, to reduce overhead due to unnecessary scanning, \( rTH \) decreases to \( aTH_a \). Note that \( aTH_a \) is lower than \( rTH \) because scanning condition (8) is satisfied. Second, once \( rTH \) is set to an extremely low value by the above operation, it is difficult to satisfy (8), thus causing a problem of not allowing scanning at all. To avoid such problem, \( rTH \) is updated to \( aTH_a \) if \( aTH_a \) is larger than \( rTH \) to allow scanning aggressively for fast handoff when smartphone receives advertising packet of \( AP_e \).

E. Verification of \( aTH \) estimation

We validate \( aTH \) estimation process with a simple experiment for 60 s. The experiment is conducted using a smartphone (Nexus 5) without mobility. AP generates 20 Mb/s downlink traffic to the smartphone between 10 s and 40 s. The AP broadcasts advertising packet every 500 ms, and the smartphone receives advertising packet and updates ACOR. Average RSSI of smartphone is \(-79\) dBm.

Fig. 5 shows the estimation results. First, \( aR \) is selected to be 39 Mb/s since RSSI is \(-79\) dBm (cf. Table I). \( ACOR \) and \( OAR \) are converged to 0.52 and 0.47, respectively, after traffic generation starts. \( eACOR \), the sum of \( ACOR \) and \( OAR \), is 0.99, which means the smartphone can occupy up to 99% of channel. However, since \( ACOR \) is bound to 0.98 due to WiFi beacon even if there is no data traffic at all, \( eACOR \) is overestimated by 1%. Therefore, \( aTH \) is 35.1 Mb/s, which is the product of \( aR \) and \( eACOR \). We identify that \( aR \) and \( aTH \) estimation work well with 1% error of \( eACOR \) even if traffic is generated.

VI. PERFORMANCE EVALUATION

A. Implementation and Measurement Setup

Implementation: APs are configured by Hostapd-2.5 with Ubuntu 14.04 Linux laptops equipped with Qualcomm Atheros AR9380 chipsets. We modify the latest ath9k device driver, backports-4.2.6-1 [35], to implement ACOR update. Ubertooth [17], an open source Bluetooth platform equipped with CC2400 transceiver, is attached to each AP for BLE beacon through a USB port. Ubertooth transmits advertising packet
every second with the proposed advertising packet format described in Section V-C.5 The operations of BLEND are implemented as an Android application on smartphone. By default, smartphone performs full scanning, which scans all 2.4 GHz and 5 GHz WiFi channel. For the advanced version of BLEND, direct scanning (DS), which perform WiFi scanning only selected channel, is implemented by enabling Android hidden API, i.e., customizedScan. We set α, a weighting factor of EWMA, as 0.2 for ACOR and OAR calculation.

Measurement setup: We conduct our experiments in two scenarios: 1) saturated traffic scenario and 2) video streaming scenario. We conduct all experiments in the topology described in Fig. 1. The APs operate at channels 40 and 48 in 5 GHz, respectively, and transmission power is set to 15 dBm. In the saturated traffic scenario, the measurement environment is the same as that in Section IV-A. In video streaming scenario, we use ExoPlayer [36] that is an Android open-source video player supporting dynamic adaptive streaming over HTTP (DASH). We encode a 120 s video clip with 16 bitrates from 0.5 Mb/s (320x240) to 15 Mb/s (1920x1080) using FFmpeg [37]. All experiments are repeated five times.

We evaluate BLEND in comparison with the following schemes:

- Legacy: operates without any additional handoff scheme.
- WiFi Manager (WM): with this commercial application [38], smartphone performs scanning event periodically every T interval without condition. When RSSI from the associated AP is under −65 dBm, smartphone performs handoff to a candidate AP supporting 10 dB higher RSSI than the associated AP. To enable fast handoff, we set scanning interval T to 3 s, the minimum configurable value in the application.
- WiFi Only (WO): this home-made application triggers scanning event every T interval only when RSSI is under −65 dBm. We set T to 3 s. Handoff decision is the same as that of WM.
- BLEND w/ DS: the advanced version of BLEND, which performs DS. BLEND w/ DS also works with an application on smartphone by enabling Android hidden API.
- Ideal: the uppermost performance assuming that the smartphone performs handoff ideally.

To evaluate the detail operation of BLEND, we use Nexus 5 for saturated traffic scenario. In video streaming scenario, both Nexus 5 and Samsung Galaxy S7 are used to show the device and Android version independence. Due to the unavailability of Android hidden API for Galaxy S7, BLEND w/ DS cannot be implemented in Galaxy S7. We also measure the average energy of Nexus 5 using Monsoon power monitor [39].

We set the interval of advertising packet to 1 s, which is the minimum scanning interval of Nexus 5, to avoid redundant transmission of advertising packet.

Full high definition video (1920x1080) can be supported in both Galaxy S7 and Nexus 5.
(b) Average HCL.

Fig. 7(b) shows average scanning overhead and scanning count during measurements. Scanning overhead denotes the ratio of total scanning time to 60 s, and scanning count is the number of scanning events during 60 s. WM always triggers scanning events every 3 s, thus resulting in the largest scanning overhead and count. WO continues scanning events until performing handoff to AP 2, and hence, scanning overhead is 19.6%. However, BLEND and BLEND w/ DS reduce scanning overhead to 7.8% and 0.6%, respectively, because they trigger scanning only when handoff is required. In case of BLEND w/ DS, scanning overhead is drastically reduced thanks to DS.

Various channel utilization: For various channel utilization (CU) of AP 1’s channel, we generate background UDP traffic. CU 0, CU 30, and CU 60 denote that smartphone can utilize maximum 100%, 70%, and 40% of AP 1’s channel, respectively. Fig. 8 shows average throughput and handoff completion latency (HCL) under various CU. HCL represents difference between the time when smartphone starts to move (20 s) and the time when smartphone finishes handoff. When smartphone cannot fully utilize AP 1’s channel due to the background traffic, it is advantageous to perform handoff to AP 2 more quickly. Therefore, HCL of Ideal decreases as CU increases. Thanks to throughput estimator, HCL of BLEND also decreases as CU increases, and hence, BLEND can achieve average throughput close to Ideal in every CU. Note that WM and WO do not consider CU, HCL of WM and WO do not decrease as CU increases. BLEND can achieve higher throughput than WM and WO up to 61% and 24%, respectively. BLEND w/ DS shows similar results to BLEND, but BLEND w/ DS achieves slightly higher average throughput than BLEND due to DS. Legacy does not perform handoff, and hence, its throughput is less than that of all other schemes.

C. Video Streaming Scenario

We evaluate the performance of BLEND and measure energy consumption during video streaming. The smartphone is initially associated with AP 1 and stays at P3. Average RSSI of both AP 1 and AP 2 is equally −70 dBm at P3. There is no traffic on both channel of APs at first, but saturated background traffic is generated only on AP 1’s channel after 60 s, which means CU of AP 1’s channel is almost 100.

Snapshot of video bitrate: Fig. 9 shows a time snapshot of video bitrate from Nexus 5. Legacy can support the maximum video bitrate for the first 60 s, but bitrate drops sharply during the next 60 s due to background saturated traffic. In case of WM and WO, video bitrate is even lower than Legacy due to unnecessary WiFi scanning. WM and WO do not take into account CU for handoff, and hence, smartphone does not perform handoff to AP 2. On the other hand, with BLEND and BLEND w/ DS, smartphone performs handoff quickly when background saturated traffic is generated, and hence, the maximum video bitrate is supported for the most of time. Since the interruption of video traffic of BLEND w/ DS is shorter than BLEND during handoff due to DS, BLEND w/ DS recovers to the maximum bitrate within 1 s much faster than that of BLEND, i.e., 7 s.

Average bitrate and energy consumption measurement: Figs. 10(a) and 10(b) show average video bitrate and energy consumption of Nexus 5. Legacy shows the lowest energy consumption because there is no WiFi scanning at all. WM and WO consume 0.44 mWh and 0.67 mWh more energy, respectively, compared to Legacy due to unnecessary WiFi scanning. In addition, WM and WO show lower video bitrate compared to that of Legacy due to unnecessary WiFi scanning. However, thanks to fast handoff and intelligent WiFi scanning operation of BLEND, it achieves 104% and 111% higher video bitrate compared to WM and WO, respectively. BLEND w/ DS shows 0.5 Mb/s higher bitrate compared to that of BLEND due to DS. The reason that the energy consumption of BLEND and BLEND w/ DS is 0.67 mWh and 0.63 mWh higher than that of Legacy is playing higher video bitrate consumes more energy. However, it is a negligible cost to support almost maximum video quality.
Device independence: Fig. 10(c) presents average video bitrate from Galaxy S7. Because BLEND w/ DS requires enabling Android hidden API, experiment is conducted without BLEND w/ DS in Galaxy S7. Galaxy S7 shows 9.0 Mb/s, which is almost same as Nexus 5. BLEND shows the highest average bitrate of 12.6 Mb/s, which is 173% and 152% higher than WM and WO, respectively. The average bitrate on Galaxy S7 excluding Legacy is lower than that of Nexus 5 because Galaxy S7 has two times longer WiFi scanning delay compared to Nexus 5.

VII. CONCLUSION

We have proposed BLEND, a novel and practical fast handoff scheme for smartphone in IEEE 802.11, based on the in-depth investigation of smartphones’ WiFi scanning and handoff operation. BLEND enables fast handoff to candidate AP by solving the sticky client problem. Our solution is aided by BLE beacon that broadcasts collocated AP’s information with advertising packet. We have implemented a prototype as an Android application. With BLEND, smartphone finds target AP and performs fast handoff based on throughput estimation, handoff decision, and adaptive scanning. Our experiment results confirm the practicality, feasibility, and performance of BLEND in diverse environments. We observe that BLEND enhances throughput and video bitrate by up to 61% and 111%, respectively, compared to the commercial Android application. As future work, we plan to reflect coexistence with legacy AP, which has no BLE module and design more fine-grained throughput estimation algorithm reflecting multiple spatial streams and channel bandwidth.

VIII. ACKNOWLEDGEMENT

This work has been conducted as part of SNU-Samsung smart campus research program, which is supported by Samsung Electronics.

REFERENCES


Fig. 10. Experiment results in video streaming and energy measurement: BLEND shows always the highest video bitrate without significant energy overhead.