Bluesaver: A Multi PHY Approach to Smartphone Energy Savings

Andrew Pyles, David T. Nguyen, Xin Qi, and Gang Zhou, *Senior Member, IEEE*

***Abstract*—WiFi effectively has two extremes: low power consumption and high latency, or low latency and high power consumption. WiFi Power Save Mode saves energy by trading added latency for less power consumption. Minimal latency but maximum power on the other hand, is consumed with WiFi Active Mode. While research has advanced in mitigating these extremes, certain types of network traffic such as constant bitrate streaming make the contrast unavoidable. We introduce Bluesaver which provides low latency and low energy by maintaining a Bluetooth and WiFi connection simultaneously. Bluesaver is designed at the MAC layer and is able to opportunistically select the most efficient connection for packets while still assuring acceptable latency. We implement Bluesaver on an Android phone and Access Point, and show that we can save more than 25% energy over existing solutions and attain the capability of quickly adapting to changes in network traffic.**

***Index Terms*—Bluetooth, WiFi, Smartphone, Energy Savings.**

I. Introduction

ENERGY efficiency on smartphones is a driving factor because of limited battery life. Due to the always-connected nature of smartphones, the efficiency of Internet access is particularly important. Wireless networking choices for smartphones typically consist of either WiFi or 3G/4G networking. When the mobile device is in a fixed location such as a home or business, WiFi is faster and more energy efficient than 3G networking [1]. Additionally, mobile plans typically place data usage limits.

Although WiFi networking is more energy efficient than 3G, considerable research has been done to make it more efficient. The WiFi standard includes Power Save Mode (PSM) which saves energy by sleeping during idle periods. Then periodically the radio wakes up to detect if packets are waiting at the Access Point (AP). While this is generally energy efficient, the buffering of packets at the AP adds additional delay. Previous work includes enhancing sleep periods during periods of inactivity [2] [3]. While this body of work has made significant progress, the high power requirements for the WiFi radio still allows room for improvements, especially for low bitrate traffic.

WiFi is more efficient on a per-bit basis [4] than other radios such as Bluetooth. An obvious question arises: should we not always use WiFi? When the WiFi Radio is predominately idle, ironically, this is also when it can be most inefficient. WiFi drivers on smartphone's come equipped with Adaptive PSM [3], the ability to switch the current power mode between sleep and active based upon the current data rate. When the WiFi radio data rate is high enough where it triggers the Adaptive PSM threshold, it will switch from sleep to Active Mode. Active Mode in-turn can consume up to 20 times more energy than Sleep mode when idle [5].

Others investigate the use of multiple radios [4] [6] to be more energy efficient. For low bitrate traffic the low power radio can be used, then when network traffic conditions change, the schemes can adapt to the other radio. A major challenge to this approach is that the act of switching between radios can be an expensive operation. In [4], only a single radio is powered on at the same time. However, when network conditions change, the other radio has to be powered on and configured which can cause a delay of several seconds, consumes extra energy each time a switch occurs and also terminates all active sockets. A key challenge is to allow the use of multiple radios without disrupting existing socket connections and allow rapid adaptation to changing conditions all while saving energy.

In this paper we also investigate using Bluetooth and WiFi with the goal of saving energy. In order to address existing challenges of previous work, we focus on the ability to switch between multiple radios without disrupting existing socket connections and have the ability to switch between radios immediately. We do this by implementing our solution at the MAC layer. Recent developments in low power WiFi radios and Bluetooth allow us to keep both radios active at the same time. The inactive radio is kept in the low power mode.

Applications for which high network throughput is desirable should use WiFi due to its superior speed (802.11n handles speeds of 300Mb/sec). But for *typical* smartphone Internet traffic is high speed always necessary? According to a recent study [7], data rates for mobile video are considerably less than 1Mbit/sec. In fact fast data speeds may not be as common as might be expected. Certainly for LAN applications high throughput is expected and should be used. Internet connections, however, are magnitudes of order slower than the WiFi router speed, throttled by [8] slower Internet routers and the broadband connection speed.

Bluetooth, explained further in the background section, has a max data speed of about 2-3Mbit/sec and a range of between 10-50m. Constant bitrate traffic is a special case where the WiFi connection is over utilized. The WiFi radio has to stay on for the duration in order to minimize latency. However if the bitrate is also below the maximum Bluetooth speed,

Manuscript submitted November 06, 2014; revised February 13, 2015. This work was supported in part by U.S. National Science Foundation under grant CNS-1250180 and CNS-1253506 (CAREER).

Andrew Pyles is with MITRE Corporation, USA. E-mail: andy.pyles@gmail.com

David T. Nguyen, Xin Qi, and Gang Zhou are with the College of William and Mary, USA. E-mail: {dnguyen, xqi, gzhou}@cs.wm.edu

then Bluetooth is more energy efficient while maintaining an acceptable latency. For certain types of traffic Bluetooth is a viable alternative.

Significant WiFi network traffic exists that underutilizes the WLAN connection. Are there other alternatives that will not impede network performance and still save energy? While WiFi PSM is energy efficient by sleeping during idle periods, the added latency is unacceptable for many applications. Bluetooth is an acceptable alternative and Bluetooth devices are present in virtually all smartphones. Although Bluetooth can handle a much smaller data rate than WiFi, Bluetooth power consumption even in its highest power state is significantly less than WiFi in Active Mode.

To address these concerns we introduce Bluesaver: A Multi PHY Approach to Smartphone Energy Savings. Bluesaver combines Bluetooth and WiFi together both at the phone and at the Access Point. When the phone is in range of the Bluetooth radio on the AP it can efficiently send and receive packets over Bluetooth. When out of range or the requirement for a higher data rate is requested, the phone uses WiFi. Bluesaver is implemented on a Motorola RAZR Android phone and can save 25% energy over existing solutions.

The individual traffic patterns of smartphones are difficult to predict. While some applications such as Skype and web-browsing may be ideal for Bluetooth, other applications such as Youtube clearly benefit from WiFi. If the goal is to save energy, how can you switch between the radios with minimal impact to the user?

To address this problem, Bluesaver provides a mechanism to seamlessly switch between WiFi and Bluetooth without impacting current applications in such a way that will save energy. Therefore, if current network traffic can be more efficiently transmitted over Bluetooth, then the smartphone can seamlessly switch between WiFi and Bluetooth for best efficiency.

In summary, the contributions of our paper are as follows:

- First, we propose a novel approach that allows seamless switching between WiFi and Bluetooth without impacting current applications while saving energy. The method combines the low latency, low power consumption characteristics of Bluetooth with high speed, higher power consumption characteristics of WiFi implemented at the MAC level.
- Second, we design and implement Bluesaver, a system that combines Bluetooth and WiFi at the phone and at the Access Point to send and receive packets efficiently while still assuring acceptable latency.
- Third, we evaluate our system by comparing energy consumption of Bluesaver and Wifi adaptive PSM. The results show that Bluesaver can save up to 25% energy over existing solutions for certain types of network traffic. Additionally, we evaluate the system's ability to quickly switch between Bluetooth and WiFi by demonstrating its adaptation to fluctuations in data rates and its connection quality adaptation.

II. Background and Motivation

In this section we cover the background section specifically related to the Bluesaver architecture. Since Bluesaver covers both WiFi and Bluetooth, we give a brief overview of both WiFi Power Save Mode (PSM) and Bluetooth. While WiFi PSM does save energy, it has the downside of adding considerable delay. Bluetooth has the advantage of a low power solution when low data rates can be used. We show that combining elements from both Bluetooth and WiFi PSM, we can potentially save more energy while minimizing delay.

A. *WiFi PSM*

WiFi PSM is part of the original 802.11 spec first standardized in 1999 [9]. WiFi clients connecting to an Access Point (AP) can negotiate a low power state. In this way, the client's radio will remain off, while incoming packets are buffered at the AP. Specifically, each beacon interval (typically 100ms), the AP broadcasts a beacon frame, and the client wakes up to receive it. This frame includes a Traffic Indication Map (TIM) indicating clients having at least one frame buffered at the AP. If the client is indicated in the TIM, it downloads the frames from the AP by sending a PS-POLL frame. Upon receipt of a PS-POLL, the AP sends frames to the client. When the client has downloaded all the buffered frames, it switches to the sleep mode [10]. While this approach works well for power saving applications, it adds an approximate 100-300ms of delay caused by the buffering of packets during the beacon intervals.

Recently, there have been several alternatives to PSM. Most deal with switching between Active Mode and PSM. Active Mode requires the WiFi radio to remain active, requiring significantly more power. Adaptive PSM as described in [3], [11] use an approach to adaptively switch to Active Mode based upon the observed data rate. When the data rate drops, the WiFi radio switches back to PSM. The "switching" occurs by sending a NULL management frame from the client to the AP. The client sets the power management bit according to whether active of PSM mode is desired. If switching from PSM to active, the buffer on the AP is first cleared using a PS-POLL management frame, also initiated by the client.

As shown in Figure 1, still the underlying trade-off with latency and power remain as WiFi research challenges. The data in the figure published in [5] shows that WiFi is suited for high speed operations. Network traffic with moderate data transfer speeds will either suffer high latency or consume extra power. Therefore, it is logical to investigate the use of other means of transmitting data that falls into this category.

B. *Bluetooth*

Bluetooth, in contrast to WiFi, is designed with low energy and small distance in mind. Data rates have an upper bound between 1Mb/sec to 3Mb/sec with version 2.0 [12] enhanced data rate. Additionally the range is limited to around 50m with the BT 4.0 specification, compared to 100m WiFi range, while older versions are limited to approximately 30m. Bluetooth is effectively used for a variety of close range applications such

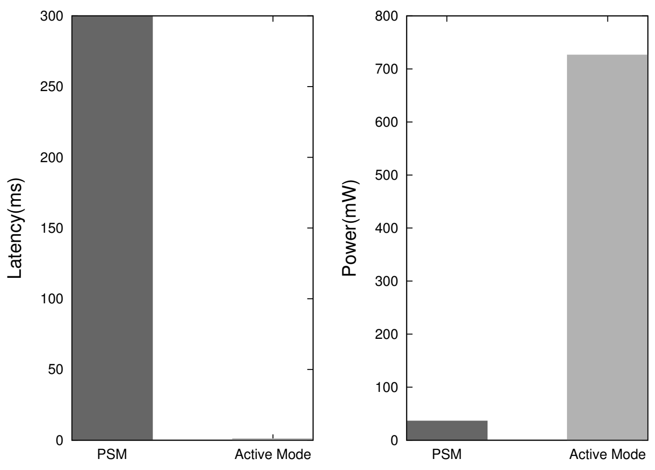

Fig. 1: WiFi energy latency tradeoff. Measurements reflect recently published measurements on smartphone WiFi PSM.

as streaming audio to headsets to peer to peer data sharing applications.

One disadvantage of sharing peer data via Bluetooth is that the slower speed can take a significant amount of time when large files are transmitted. In order to address this concern, the 3.0 specification has included the High Speed (*HS*) [13] specification. This specification allows files to be transmitted at high speed by utilizing the high speed capabilities of a co-existing WiFi card. The connection is established with Bluetooth and then the file can be transmitted to the peer (which also must support HS) via WiFi.

The 3.0 *HS* specification provides the capability for Bluetooth connections to ultimately be more energy efficient for the transmission of large files. Additionally, when having a constant stream based connection such as a Bluetooth headset for streaming audio, it is more energy efficient to utilize Bluetooth than the higher energy cost WiFi. WiFi is energy inefficient when it comes to streaming audio or other applications that require small data rates, since in such cases, WiFi is forced to use the high speed, high power radio, which results in higher energy consumption.

C. Motivation

In this subsection we investigate cases where WiFi alone can use some improvement from an energy savings perspective and look to see how prevalent such cases are. Specifically, we examine cases where either *bandwidth is limited*, or *streaming applications such as video or audio are in use*.

In Figure 2, we see the bandwidth required over time as a Youtube video is watched on a recent Motorola Android smartphone. As can be seen, the bandwidth tends to spike and then quickly drop off. Static video content, typically delivered in chunks can take advantage of the speed and efficiency of a WiFi connection. In this case, WiFi handles the bursty nature of static video content providers such as Youtube efficiently. The WiFi driver is suited to quickly switch to Active Mode and download the next available chunk of video. As can be seen the broadband connection allows connections of up to approximately 3Mb/sec, and so WiFi is a good fit.

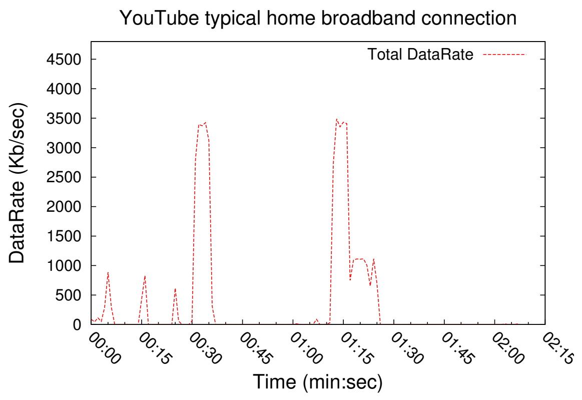

Fig. 2: Youtube typical broadband connection

In Figure 3 we see a different story. In this case, instead of bursty traffic, we see the bandwidth of a constant bitrate live streaming Skype call. Figure 3 shows a trace of a Skype audio call between two users. In this case, the trace shows a constant bandwidth of slightly less than 100Kb/sec which ends approximately 2.5 minutes later. The bandwidth is high enough that the WiFi driver will switch to Active Mode, thus minimizing latency. However, due to the higher power requirements of the WiFi radio, the lower bandwidth requirements of this particular applications could just as easily be fulfilled by another radio such as Bluetooth. This wastes unnecessary energy and can use some improvement.

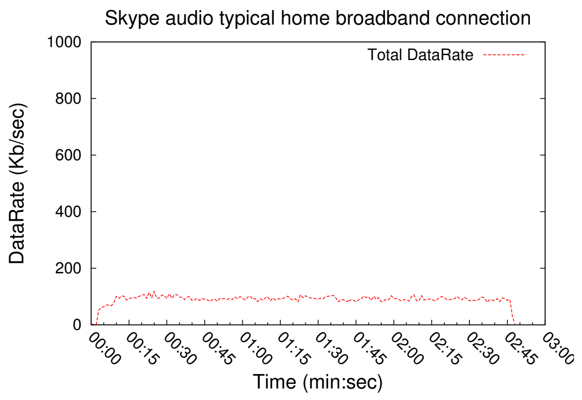

Fig. 3: Skype audio typical broadband connection

Figure 4 shows a Skype video call that was set to high quality and again is at a constant bitrate. In this case, the bandwidth requirements are approximately ten times higher than that of the previous audio call, only slightly below 800Kib/sec. Note that since the constant bitrate never exceeds 1Mbit/sec, the WiFi radio again is under-utilized. Even at the highest available video streaming rate, Bluetooth is a viable alternative for video streaming. Bluetooth, with its lower power draw, could be used to save energy. At the same time since the connection is below one megabit per second, the QoS requirements are not impacted.

As we have shown, for constant bitrate network traffic, WiFi is under utilized. In order to minimize delay, the WiFi radio must be kept active most if not all of the time. Therefore, the

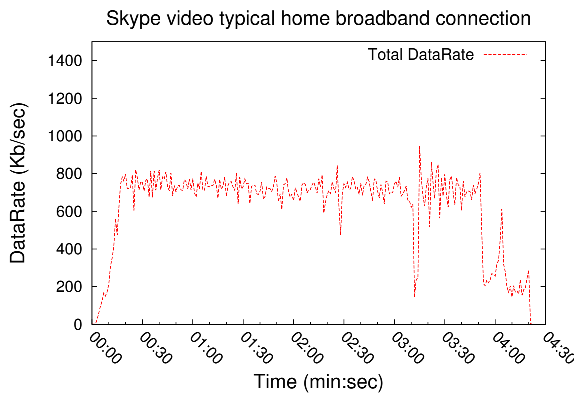

Fig. 4: Skype video typical broadband connection

use of an alternative radio such as Bluetooth could easily be utilized to save energy.

For video streaming at HD, the video streaming is peaked at 1.5 Mb/sec [14], which is available for premium members only; not available as an option for the Android client. The more likely high quality video streaming is at 500Kb/sec. Facetime has been measured in [15], with results less than 400Kb/sec. Clearly in these cases the WiFi radio is mostly idle.

Limited Bandwidth. How realistic is the case of limited bandwidth for WiFi connections? Clearly WiFi Wireless LAN connections are getting anything but slower. WiFi 802.11n for instance supports speeds up to 300Mbit/sec. However, research conducted recently by Dogar [8] shows that the bottleneck is not the highspeed WiFi connection between the AP and the client, but rather between the AP and the Internet. Therefore it is entirely plausible to have a smartphone with a 300Mb connection to the AP and a 1Mb connection to the Internet.

By finding opportunities where traffic patterns meet the criteria above where Bluetooth consumes less energy than WiFi, we can exploit these periods to save energy on smartphones.

III. Bluesaver Design

We have described the challenges facing WiFi clients. To address these challenges, we introduce Bluesaver: Multi PHY approach to smartphone energy savings. In this section we describe the system architecture and discuss the design of the Bluesaver system. Bluesaver has been designed to function at the MAC layer. Both radios are kept on simultaneously. This way, packets can be sent either via Bluetooth or via WiFi. To save energy, both WiFi and Bluetooth connections are kept in a low power state when idle to save energy.

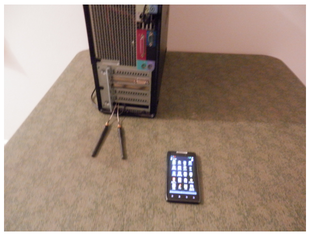

Fig. 5: Lab Setup.

The Bluesaver design consists of a modified WiFi AP which also includes a Bluetooth adaptor. The client is an Android smartphone which also includes WiFi and Bluetooth. The lab setup is shown in Figure 5. While our lab setup is used with a smartphone, Bluesaver can be used with any system that has multiple radios. Bluesaver could be easily extended to laptops and tablets.

Architecture. The Bluesaver architecture is spread out between the client portion running on the smartphone and the WiFi AP shown in Figure 6. It is comprised of three main components: The Health Monitor (HM), the Bluesaver Connection Manager (BCM), and the Sending Decision Manager (SDM). All of these separate components used together are responsible for switching packets over the best available PHY interface.

The HM component is responsible for tracking the health of each Bluetooth connection. When a Bluetooth connection with a peer is established, the HM monitors traffic going through the device. Specifically, for each connection the HM component is responsible for monitoring the current data rate, connection status, packet loss and delay. Once this information is gathered, information can be passed onto the SDM in order to determine through which interface the packet should be sent.

A crucial part of the HM component is the Bluetooth Availability Manager (BAM). This component is responsible for checking the connection status of the peer. The BAM determines the health of the peer, using *l2ping* which is similar to ICMP ping but instead uses Bluetooth l2cap packets. To save energy, the BAM operates periodically, currently once per second and only when network traffic is observed.

The bulk of the HM operation occurs within an asynchronous timing thread that re-occurs every second. If a new packet has been transmitted in the past second, that is if any network traffic has been observed, the current connection quality and the current data rate are updated. If the data rate exceeds the threshold of what Bluetooth can handle (1.5Mb/sec) then *UseBluetooth* is set to *false*. Additionally, if the results of the RTT observations retrieved by the BAM using l2ping either fails (unable to connect) or shows an unacceptable latency (greater than 100ms), then *UseBluetooth* is set to *false*.

The HM notifies BAM to refresh the latest health statistics through a netlink socket. The BAM (which is running in userspace) sends an l2cap ping to the peer only when current traffic is detected. We use an l2cap ping operation because Bluetooth devices have l2ping operation enabled by default as part of the firmware. We send 4 packets and record the average RTT (the firmware of most devices closes the socket after 4 packets). If the delay is determined to be more than 100ms, then we assume that the connection is unsuitable, and BAM then sends a notification through the netlink socket. The BAM currently only runs on the AP.

When a packet is ready to be transmitted, the SDM determines which interface will be used. Figure 7 describes the interaction. When the host OS sends a packet, it will pass the packet onto the driver. Bluesaver will intercept that packet and determine which interface to use. When the packet is placed in the transmit queue, the PHY interface to use is determined

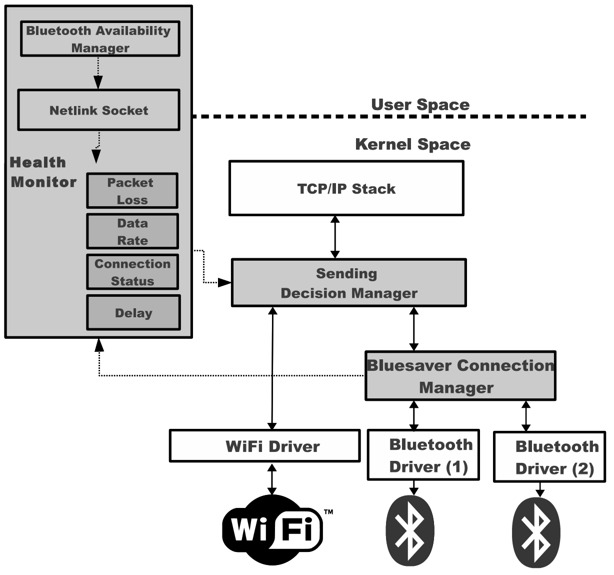

Fig. 6: Bluesaver Architecture.

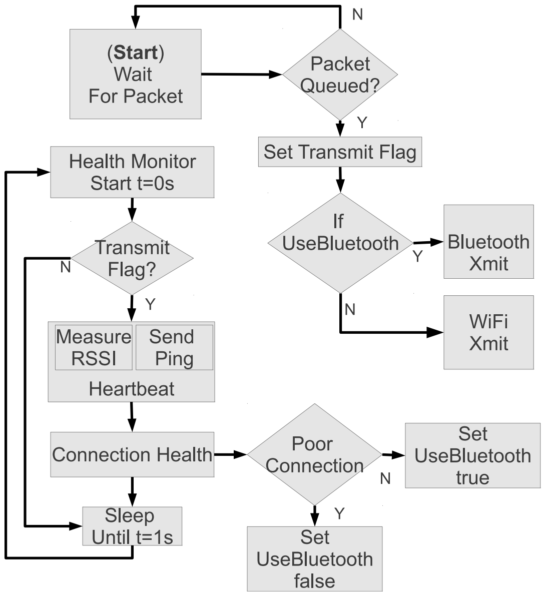

Fig. 7: Bluesaver Design.

via the *UseBluetooth* variable. Then the packet will be either transmitted through the WiFi driver or through Bluetooth.

Once these parameters have been obtained, a decision is made whether the connection is suitable for Bluetooth operation. If a threshold is crossed for either latency, packet loss or data rate, the variable *UseBluetooth* is set accordingly. At this point when an outbound packet destined for the WiFi interface is queued in the driver transmit queue, the WiFi driver will either transmit the packet directly if set to *true*, else the BCM will transmit the packet over the bluetooth interface. The AP is also elaborated in Algorithm III.1. Note that we only highlight key parts that require changes to the state-of-the-art.

The AP is responsible for checking the health of each client through the HM component as described earlier. When a decision is ultimately made to send via a specific PHY interface, an important consideration is to keep this decision in sync with the client. That is, when the AP sends packets over WiFi, the phone should also send over the same medium. In order to accomplish this, we have a much simpler design on the client. The client's initial setting is to *UseBluetooth*. The AP will determine the overall health of the system using the method previously described and the appropriate PHY will be selected. On the client, when a packet is received on a different interface than what is expected, the local *UseBluetooth* will be set accordingly. For instance if the client is sending packets over Bluetooth, but then receives a packet over WiFi, *UseBluetooth* will be set to false. For the sake of clarity, the client is also elaborated in Algorithm III.2. We only highlight key parts that require changes to the state-of-the-art.

Algorithm III.1: AP()

```
while no packet queued
do { wait for a packet
choose interface to transmit
set transmit flag
if UseBluetooth == true
then {//transmit via Bluetooth
else {//transmit via WiFi
```

Algorithm III.2: CLIENT()

```
//set Bluetooth as default
UseBluetooth = true
while exists a buffered packet
do { download a packet

     if packet received over WiFi
     then { //switch to WiFi
            UseBluetooth = false
     else { //keep using Bluetooth
            UseBluetooth = true
```

The Bluesaver Connection Manager (BCM) is responsible for transmitting Bluetooth packets through the system. It does this by opening a Bluetooth l2cap socket to the peer. When l2cap packets are received over this socket, the packet will be passed to the host so that the host OS cannot differentiate it from a WiFi packet. In order to accomplish this, the Bluetooth MAC addresses are replaced with the WiFi source and destination MAC addresses in an 802.3 header that the host OS is expecting. When packets are transmitted over Bluetooth, they are taken from the WiFi driver transmit queue and sent over the l2cap socket.

On the AP, the BCM requires one Bluetooth adaptor for a single client. A single Bluetooth adaptor can also scale to multiple clients, since our implementation requires a separate l2cap port for each client. Depending on the traffic loads from each client, the combined bandwidth of multiple clients could overwhelm a single adaptor. In this case, additional Bluetooth hardware can be used to support multiple users. Our initial design is to dedicate a single Bluetooth adaptor to each client. In future, we plan to extend this to make this more extensible by extending the BCM to have multiple socket connections to the Bluetooth adaptors on the AP. Then utilize an appropriate load balancing scheme to determine which destination interface to which to transmit.

By operating at the MAC layer, Bluesaver is able to quickly adapt to adverse network connections and switch quickly between WiFi and Bluetooth.

IV. Evaluation

We implement Bluesaver on the Motorola RAZR [16] phone. The RAZR comes with Android version 2.3.5 and Linux kernel version 2.6.35.7. The WiFi AP is implemented on a PC equipped with Ubuntu 12.04, an ath9k WiFi driver and an ath3k Bluetooth driver. The implementation consists of a Linux kernel module on both the phone and the AP. The workstation is equipped with a Qualcomm-Atheros reference design PCI card [17] that includes both Bluetooth 4.0 and WiFi capabilities. The Bluetooth Availability Manager (BAM) is implemented as a user level daemon process. Further implementation details are elaborated in Appendix.

To evaluate the system correctly, we must show that it first saves energy over existing solutions. Second, we must show that Bluesaver can adaptively switch between Bluetooth and WiFi due to changing network conditions.

This section consists of our evaluation method followed by the energy comparison section. We conclude with an evaluation of how Bluesaver responds to dynamic network conditions.

A. Evaluation Method

To measure the power consumption, we use the Monsoon [18] power monitor as previously in [19], [20]. The Monsoon bypasses the existing battery and provides power to the phone. It measures the instantaneous voltage and current with a sample rate of 5kHz. We can then determine the overall system power draw over a given time interval. In order to isolate the power consumption specifically to the test in process, we enable "Airplane" mode which disables all PHY interfaces. Then the interface that is about to be tested is explicitly enabled. Additionally, we make a best effort attempt to disable all services and background processes running on the phone during the test.

B. Energy Comparison

To assess the energy comparison between Bluesaver and Wifi adaptive PSM, we compare power consumption levels between WiFi and Bluetooth first for data rate throughput testing. Second, we compare the power consumption of video streaming at incrementally increasing data rates between the AP and the phone. In this section we are ultimately comparing the power and energy consumption of Bluetooth vs. WiFi. Although Bluesaver can handle both cases, it can really save the most power and energy when using Bluetooth.

We measure the throughput by sending ICMP ping. By varying the packet size and packet sending rate, we were able to accurately measure throughput and data rate. The reason ICMP ping was used for throughput testing, is that it is an accurate bi-directional throughput testing tool. Since the payload size for ping packets is identical for sending and receiving, the sending and receiving operations are equally tested. This is especially important for WiFi. Recall from the Background section that when WiFi is in PSM mode, receiving packets that are queued at the AP has an added delay of several hundred milliseconds. ICMP ping therefore places equal weight on sending and receiving packets.

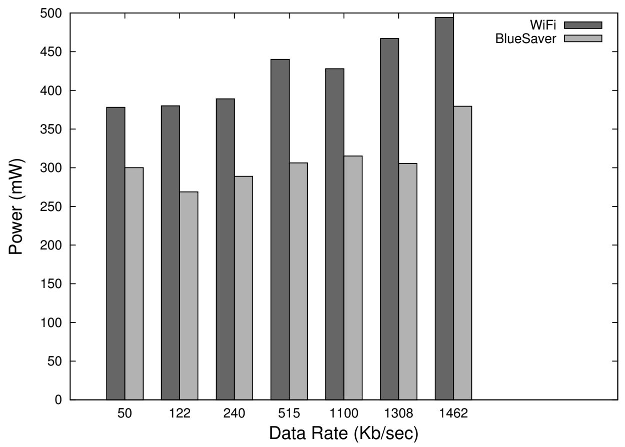

Fig. 8: Bluetooth vs Adaptive PSM power consumption. Adaptive PSM consumes between 20% more and 35% power than Bluetooth.

Using ICMP ping for throughput testing, we test bitrates from 50Kb/sec up to 1400Kb/sec, as can be seen in Figure 8. We initiate the test from the phone to the AP. For the entire bitrate range that we test, the Adaptive PSM implementation within the WiFi driver switches to CAM for a majority of the time resulting in extra power consumption. Clearly the WiFi driver is performing the best it can under the circumstances. Recall from the Background section that staying in PSM will result in unacceptable delay, while staying in CAM results in the higher power consumption. Within the bounds of the bitrate range that we test, Bluetooth is more efficient than WiFi. Once the rate exceeds 1500Kb/sec, we start to see packet loss and added RTT delay over the Bluetooth radio. Therefore, as part of the Bluesaver design, when the rate exceeds 1500Kb/sec, we switch to WiFi to minimize delay.

As can be seen, Bluetooth consistently saves between 20% and 35% power consumption compared to WiFi adaptive PSM. This shows that when data rates are within this range, Bluetooth should be used to extend the battery of smartphones.

This result also shows an interesting trend. In [5], the power consumption of CAM mode on an Android smartphone from 2010 was measured to approximately 20 times higher than that of PSM around 720mW when idle. Note that with the Motorola RAZR, which came on the market less than two years later, we find that the WiFi driver is much more efficient, approximately 375mW with a light load of 50Kb/sec. The reasons for this improvement are not that clear, however, it could be that the Adaptive PSM implementation has been improved as well. Even with these improvements, Bluetooth is still a better alternative with lower bitrate network traffic.

Video Streaming. According to [7], 80% of videos for smart phone traffic use a bitrate of less than 256kbps. In order to evaluate the energy efficiency of Bluetooth, we setup a streaming server located on the same LAN as the AP. We set the video streaming server VLC to stream via RTSP and streaming at video bitrates from 64kbps to 512kbps. We measure the total energy consumed and compare to Adaptive PSM.

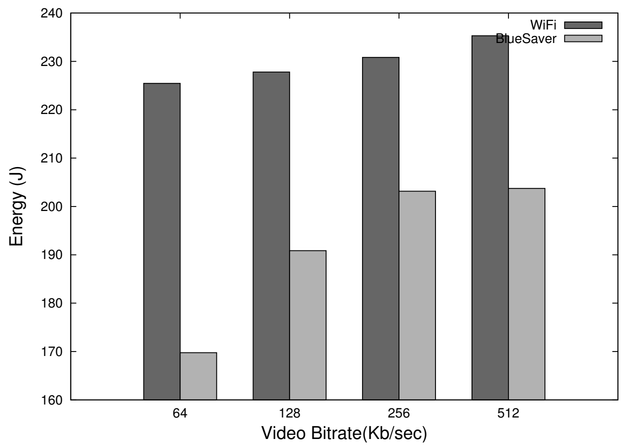

Fig. 9: Bluesaver vs WiFi streaming video energy comparison. Bluesaver saves up to 25% energy for bitrates ranging from 64 to 512 kbps.

We install a popular RTSP streaming application called MoboPlayer on the phone. This application is capable of playing back RTSP stream over the Internet. We measure the average power consumption during the test and measure the total time taken to play back the entire video at the various bitrates. This particular application requires time to buffer the video stream before it is played back. Figures 9 and 10 show the results of this test. In Figure 10, it is quite clear that Bluetooth consumes much less power than WiFi in all tests.

Since we are using the RTSP protocol for streaming, there are several seconds of buffering that occurs before the stream actually begins playing. In this case, WiFi clearly has the advantage due to its superior speed. Therefore, the Bluetooth energy results, shown in Figure 9 reflect a more modest energy savings ranging from 25% for video codec streaming at 64kbps to 13.5% at 512kbps. This shows that even if a streaming protocol requires extra time for buffering, Bluetooth is still a better solution for streaming video.

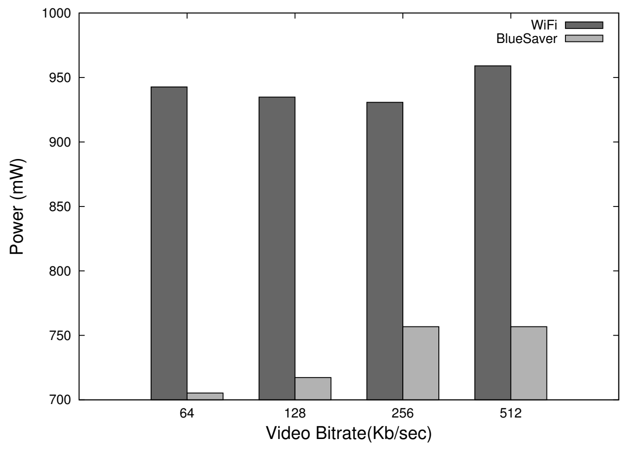

Fig. 10: Bluesaver vs WiFi streaming video power comparison. Bluetooth consistently uses less power than WiFi.

C. Network Adaptation

In this subsections we demonstrate that Bluesaver has the ability to quickly switch between Bluetooth and WiFi. A key characteristic of the Bluesaver architecture is to nimbly switch between radio types with minimal delay. In order to thoroughly test this aspect of Bluesaver design, we test two key components. How quickly does Bluesaver adapt to fluctuations in data rates. Second, we address connection quality adaptation. That is, how quickly and how does Bluesaver adapt when the phone is outside of the useful range of Bluetooth and still within the useful range of WiFi.

Data Rate Adaptation. We measure the responsiveness, or how quickly the system can detect changes in data rate and respond accordingly. When the AP first starts to send packets to the phone, the data rate spikes above the Bluetooth data rate threshold of 1.5Mbit. When the data rate exceeds the threshold, the AP will switch from Bluetooth to WiFi. When the phone detects that packets are received on the WiFi interface, it will disable Bluetooth and transmit packets over WiFi. When the data rate again drops below the threshold, it again switches back to Bluetooth.

We place a 10MB and 100MB file on a web server running on the same subnet as the AP. We then proceed to download the file using an http client on the phone. This is the worst case energy-wise for Bluetooth because the high WLAN speeds available over WiFi make Bluetooth inefficient. Figure 11 shows the results of this test. We first download a 10MB file, then wait a few seconds and download the same 10MB file again. A few seconds later, we download the 100MB file. As soon as the first download starts, Bluesaver rate adaptation detects that the download speed exceeds what Bluetooth can handle. At that point, it switches to WiFi. The packets are received over WiFi to quickly download the file. When the file

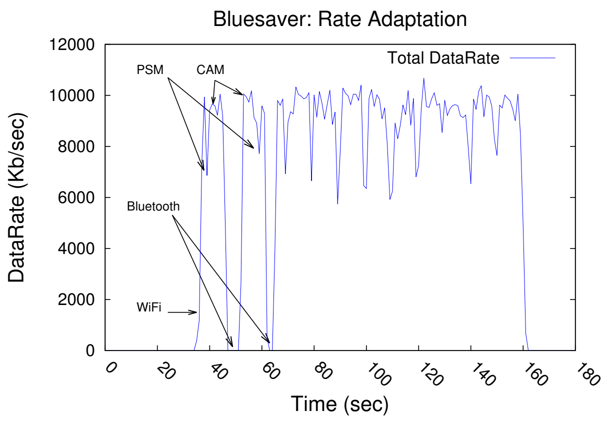

Fig. 11: Rate Adaptation: Bluesaver switches from Bluetooth to WiFi while sequentially downloading a 10MB file followed by a 100MB file without interrupting the download.

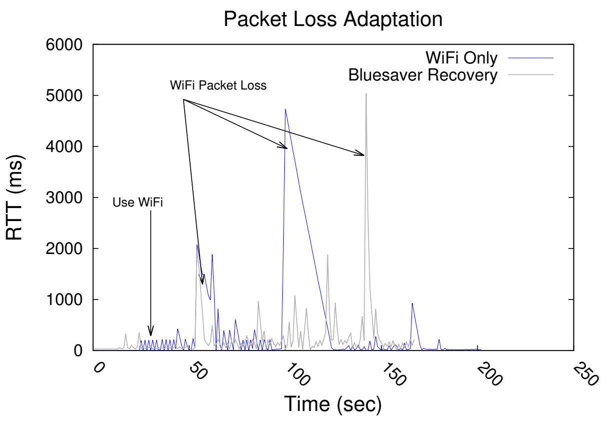

Fig. 13: Connection Adaptation: Bluesaver switches from Bluetooth to WiFi when Bluetooth connection issues are detected. A constant ping from the server to the phone was done for the duration of the test. When the switch to WiFi occurs at around 30 seconds, the ping test is unaffected.

is done downloading, the connection falls back to Bluetooth to save energy.

We note that during the test, the WiFi driver did switch between CAM and PSM during the download as noted in the figure. It is not clear why the WiFi driver switched to PSM during the middle of the transfer. However, this explains the dips shown in the download of the 10MB files and shown especially clearly in the 100MB file. Additionally, due to limitations (or a configuration error) of the AP software, we were unable to exceed speeds of 10Mb/sec.

Connection Adaptation. We measure how quickly Bluesaver can adapt to changes in the network environment. To perform this test, we setup a Bluesaver enabled AP in a building described in Figure 12. We initially setup WiFi only and walked from within one meter of the AP to points A, B,C,D, then A,B,C,D again. After which we proceeded back to the AP. The total distance from the AP to point D is about 10 meters. During the duration of the test we setup an ICMP ping running on the AP that will record the RTT time.

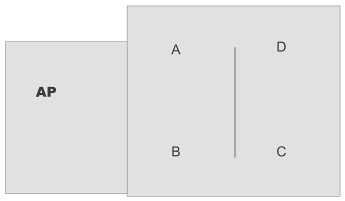

Fig. 12: Room Layout showing lab environment for connection adaptation evaluation.

The results of this test are shown in Figure 13. The WiFi results show that around time 50 and 100s the phone experienced extreme packet loss and was unable to respond for several seconds. Tracking down the root cause of packet loss is challenging, but most likely this is the result of a combination of the fact that the phone was rapidly moving and the multiple obstructions between the phone and the AP.

The same test was then performed with Bluesaver enabled. In this case, the Bluetooth Availability Manager running on the AP sent a steady l2ping to the phone during the duration of the test. At 30 seconds into the test, the BAM was unable to connect to the phone over Bluetooth. Immediately, the HCM was notified over the netlink socket and it started sending packets over WiFi for the duration of the test until the phone got back into Bluetooth range of the AP again. The phone then detected that packets were retrieved over WiFi and in turn started sending packets over WiFi as well.

The result is that at 30 seconds into the test, there is no perceptible difference in the RTT measurement. This shows that Bluesaver can switch between radios without adding any additional delay. After 30 seconds into the test, we see severe WiFi related packet loss similar to the WiFi only test which in this case is unavoidable.

We have shown that Bluesaver can adapt to fluctuating data rates and adverse network connection issues. Furthermore, Bluesaver is able to save energy when using low data rate applications by as much as 25% over existing solutions.

V. Discussion

In this section, we discuss potential limitations of the Bluesaver system and other related matters that may concern readers.

When a packet is ready to be transmitted, Bluesaver determines the network interface to send through. The decision is made based on the recent history of the network traffic. While this is true that Bluesaver may not always provide optimal energy savings due to varying network quality, the system utilizes a simple yet energy efficient moving average based on previously measured network quality parameters. Certainly, there is no way to predict the future, but by sampling over the past several seconds we can determine with high confidence whether it is appropriate to switch to another interface.

Bluesaver utilizes l2ping to determine the health of the connection. This is a standard approach to test the communication quality with a Bluetooth device. It provides timing information on how long it takes to send and receive packets of a certain size. The only required parameter is the address of the Bluetooth device to ping. While it is true that this introduces additional energy overhead, the l2ping packets are very small (44 bytes), and the amount of the energy required is minuscule. Moreover, the system sends an l2ping to the peer only when current traffic is detected, which means the l2ping energy overhead occurs only when data is being transmitted.

Bluesaver requires an AP to have a Bluetooth radio, which adds to the hardware complexity and the cost of the AP. However, thanks to the recent advancements in the radio technology, it is now possible to add the Bluetooth capability to an AP through adapters in a simple and affordable way (from $10.95 [21]).

The proposed scheme does not add any modification at the physical layer, therefore it cannot be a potential source of any Bluetooth/WiFi glitch or cross-interference. Bluetooth and WiFi are enabled simultaneously on many devices, and the two interfaces are naturally designed so that there is a minimal probability of radio collisions.

As discussed in Section III (10th paragraph), the client utilizes a simple mechanism to identify the AP's radio in use via the *UseBluetooth* variable. The client assumes that the AP uses Bluetooth by default, and the *UseBluetooth* variable is set to *true*. When a packet sent from the AP is received on a different interface than what is expected, the client's local *UseBluetooth* will be set accordingly. For instance, if the client is sending packets over Bluetooth, but then receives a packet over WiFi, *UseBluetooth* will be set to *false*.

Section III (4th and 7th paragraph) elaborates switching criteria, where in addition to the data rate, Bluesaver also considers the connection status, packet loss, and delay. Specifically, the Health Monitor gathers the mentioned information, and passes onto the Sending Decision Manager in order to determine through which interface a packet should be sent. For instance, if the delay is determined to be more than 100ms, then the system assumes that the connection is unsuitable.

This work proposes to switch between the WiFi and Bluetooth interface to save power consumption. It may be also interesting to investigate opportunities of increasing data rate through the simultaneous use of both interfaces. We believe that it may be faster, but probably not as fast as some other standards such as 802.11ac approved in January 2014 [22]. This is out of the scope of our work that primarily focuses on power saving opportunities. We reserve this therefore for future work.

The fact that the proposed solution requires that smartphones have Bluetooth presents no serious limitation. According to [23], the market for smartphones is forecast to reach almost 700 million units by 2015, with Bluetooth technology incorporated in all such phones.

VI. Related Work

Although there is a significant body of work focused on mobile energy efficiency, we focus primarily on those closely related to Bluesaver.

Multiple PHY. We found a few papers that focus on combining multiple radios for best efficiency. ZiFi [24] and Blue-Fi [25] use low power radios to detect the presence of WiFi access points. Coolspots [4] uses a combination of Bluetooth and WiFi to save energy. However, only one radio is active at the same time causing serious delay and dropped connections when switching between radios. BlueStreaming [26] also uses WiFi and Bluetooth. Both radios are on at the same time with 2 separate IP addresses. This is a serious limitation, since applications have to be specifically designed to bind to one of the IP addresses. Bluesaver uses both WiFi and Bluetooth but it applies a completely different approach. Since it is implemented at the MAC level, it is able to seamlessly switch between radios without impacting applications. Ylitalo et al. [27] present an interface selection mechanism for multihomed mobile hosts. However, the decision-making process depends on user-defined preferences, which makes it impractical to ordinary users and inefficient for energy saving purposes. Friedman et al. [28][29] provide a study on power and throughput tradeoffs of WiFi, Bluetooth, and other interfaces in smartphones, but do not go beyond offering theoretical analysis and insights for phone developers. SwitchR [30] devises a multi-client switching policy enabling communication among clients within a multi-radio environment. However, the work does not go far beyond providing the framework and policies. Moreover, its architecture requires a Bluetooth gateway, a device that functions as a Bluetooth AP, which presents additional complexity to deployment. Corvaja [31] conducts a theoretical analysis and simulation of the QoS performance under different traffic and network density conditions for handoffs from Bluetooth to WiFi. Our work proposes a practical approach to smartphone energy saving through conscious switching between the WiFi and Bluetooth interface. The system is implemented on an Android-based smartphone and a Linux-based access point.

Client modifications. Numerous client side research exists addressing the problem of making the client more energy efficient. Micro power management [2] predicts and utilizes microsecond sleep periods to save energy. While others [32][3] focus on application specific traffic shaping based upon priority or sensitivity to delay; traffic not sensitive to delay or given a low priority is forcibly sent through PSM. Others [5] examine application specific idle periods to save energy by switching to PSM during idle periods. PSM-throttling [33] uses traffic shaping to streamline TCP traffic for PSM efficiency. E-Milli [34] uses a downclocking scheme which reduces the voltage to the WiFi driver during idle listening times. Bluesaver is complementary to these approaches; client side enhancements can be extended with the use of multiple radios.

Infrastructure modifications. Others have modified the wireless infrastructure to be more efficient. Napman [11] uses a fair energy-aware scheduling algorithm to increase WiFi efficiency. Sleepwell [35] reduces client collisions with the use of multiple APs staggered at different time intervals while Catnap [8] exploits the difference between WLAN and WAN speeds. Others [36] explore the use of proxies to reduce client-side polling. All of these approaches save energy by streamlining the wireless traffic to and from the client. These approaches are complimentary to Bluesaver; any enhancements in the infrastructure to make WiFi more efficient will also enhance Bluesaver.

Bluetooth and WiFi standards. Bluetooth low energy (BLE), also known as Bluetooth Smart, is a Bluetooth technology designed by the Bluetooth Special Interest Group, and merged into the main Bluetooth standard in 2010 as part of the Core Version 4.0. [37]. BLE 4.0 provides reduced power consumption while maintaining a similar communication range. Additionally, the version is already supported in a number of devices, including the Motorola RAZR phone used in our experiments (see full list of supported devices at [38]). BLE 4.0 and our proposed solution are therefore mutually inclusive. Finally, the RAZR device is also shipped with WiFi 802.11 b/g/n that supports scheduled APSD (S-APSD) and unscheduled APSD (U-APSD), also known as WiFi Multimedia (WMM) [39]. This technology increases the efficiency and flexibility of data transmission. Specifically, the client device can doze between packets to save power, while the access point buffers down-link frames. The client chooses the time to wake up and receive data packets to maximize power conservation without sacrificing Quality of Service. Clearly, this is complementary to Bluesaver.

VII. Conclusions & Future Work

Smartphones suffer from the dilemma that network traffic to and from the device either has excessive latency and low power consumption or low latency with high power consumption. To address this problem, we have presented Bluesaver, a novel approach that combines the low latency, low power consumption characteristics of Bluetooth with high speed, higher power consumption characteristics of WiFi implemented at the MAC level. We have demonstrated that Bluesaver is able to adapt to changing network conditions by routing network traffic between different PHY interfaces. Finally, we have demonstrated that we can save up to 25% energy over existing solutions for certain types of network traffic.

In future work, we plan to extend the Bluesaver platform to better support multiple clients. As we described in the Design section, our current solution is to pair a single client with a dedicated Bluetooth adaptor. In future, we plan to extend this approach to use a load balancing algorithm to more efficiently support multiple clients.

Appendix A
Implementation Details

In this section, we discuss implementation details of the Bluesaver system and its key challenges.

One of the challenges faced with the implementation is that the smartphone comes with a locked bootloader. This makes it nearly impossible to modify the kernel. Therefore, all of our modifications had to be done within the confines of a kernel module. Due to this limitation, we were unable to access health related statistics from the Linux kernel from the Bluetooth device because the symbols were not exported. Also due to differences in the Linux kernel between the AP and the phone, the Bluetooth kernel API's were slightly different.

The BCM component requires modification to the WiFi drivers on both the phone and AP. When a packet is about to be transmitted over WiFi in the driver, we modify the transmit portion of the WiFi driver to check if the packet should be sent over Bluetooth. If so, the packet is placed in a transmit queue and transmitted over the l2cap socket. The receive functionality on the driver is not modified.

Phone Challenges:

Another subtle difference between the Android implementation and the AP is the issue of wakelocks. The Android kernel supports the concept of entering a deeper sleep when a wakelock is not held. When packets are transmitted or received, we hold a wakelock. Each time a packet is received we set a flag. We have a timer that runs every second. If a packet is received during that time window, we hold a wakelock for one second. In this way, the maximum amount of time we hold a wakelock is one second when idle.

The client component is implemented with a queue data structure. When new outbound packets need to be transmitted, they are inserted into the transmit queue. A separate thread is run periodically whenever the queue length has at least one packet in it.

Bluetooth Availability Manager. The BAM is a userlevel process that is responsible for determining the status of the Bluetooth connection. Every 500ms the BAM sends a BT heartbeat packet to its peer. If a heartbeat packet is not received within one second then the connection is bad. This could be caused from the phone being outside of the range of bluetooth. The connection status is then transmitted to the Health Monitor via a netlink socket.

BAM uses a *select()* loop to respond to incoming packets. In order to precisely send packets at a given interval, we use the Linux *timerfd()* system call. However, Android's bionic libc does not support this particular system call. Therefore, we had to explicitly add support for this system call with *system()*.

Bluetooth Notes. We made the best effort to obtain the highest Bluetooth throughput possible. Recall from the Background section, that Bluetooth supports up-to 3Mbit/sec. In our tests, we were able to achieve 1.7Mbit/sec, which is close to the theoretical maximum of 2.1Mbit/sec. We used l2cap based sockets with default options for the implementation.

References

[1] A. Rahmati and L. Zhong, "Context for Wireless: Context-Sensitive Energy-Efficient Wireless Data Transfer," in *ACM MobiSys*, 2007.
[2] J. Liu and L. Zhong, "Micro Power Management of Active 802.11 Interfaces," in *ACM MobiSys*, 2008.
[3] A. Pyles, X. Qi, G. Zhou, M. Keally, and X. Liu, "SAPSM: Smart Adaptive 802.11 PSM for Smartphones," in *ACM UbiComp*, 2012.
[4] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "Coolspots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces," in *ACM MobiSys*, 2006.
[5] A. Pyles, Z. Ren, G. Zhou, and X. Liu, "SiFi: exploiting VoIP silence for WiFi energy savings in smart phones," in *ACM UbiComp*, 2011.
[6] Y. Liu, F. Li, L. Guo, Y. Guo, and S. Chen, "Bluestreaming: towards power-efficient internet P2P streaming on mobile devices," in *ACM MM*, 2011.
[7] J. Erman, A. Gerber, K. Ramakrishnan, S. Sen, and O. Spatscheck, "Over The Top Video: The Gorilla in Cellular Networks," in *ACM IMC*, 2011.
[8] F. R. Dogar, P. Steenkiste, and K. Papagiannaki, "Catnap: Exploiting High Bandwidth Wireless Interfaces to Save Energy for Mobile Devices," in *ACM MobiSys*, 2010.
[9] "IEEE 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," 1999, aNSI/IEEE Std. 802.11dcf.
[10] 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 *IEEE INFOCOM*, 2015.
[11] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, "NAPman: Network-Assisted Power Management for WiFi Devices," in *ACM MobiSys*, 2010.

[12] "Bluetooth Core Version 2.0 + EDR," 2009, bluetooth SIG Std. Core Version 2.0 + EDR specification.
[13] "Bluetooth Core Version 3.0 + HS," 2009, bluetooth SIG Std. Core Version 3.0 + HS specification.
[14] "How much bandwidth does Skype need?" https://support.skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need, 2012.
[15] "iPhone FaceTime bandwidth gets measured," http://www.digitalsociety.org/2010/08/iphone-facetime-bandwidth-gets-measured/, 2010.
[16] "Motorola RAZR," 2012, www.motorola.com.
[17] "Qualcomm-Atheros WB225 Reference Design," 2012, http://www.qca.qualcomm.com/media/product/product _106_file1.pdf.
[18] "Monsoon Solutions," 2011, http://www.msoon.com/LabEquipment/PowerMonitor.
[19] D. T. Nguyen, G. Zhou, X. Qi, G. Peng, J. Zhao, T. Nguyen, and D. Le, "Storage-aware smartphone energy savings," in *ACM UbiComp*, 2013.
[20] D. T. Nguyen, "Evaluating impact of storage on smartphone energy efficiency," in *ACM UbiComp*, 2013.
[21] A. Industries, "Bluetooth 4.0 USB Module (v2.1 Back-Compatible)," *http://goo.gl/RF4NkW*, 2014.
[22] IEEE, "IEEE 802.11ac," *http://goo.gl/H9xU4Y*, 2014.
[23] A. West, "Smartphone, the key for Bluetooth low energy technology," *http://goo.gl/BeiaKP*, 2014.
[24] R. Zhou, Y. Xiong, G. Xing, M. L. Sun, and J. Ma, "ZiFi: wireless Lan discovery via ZigBee interference signatures," in *ACM MobiCom*, 2010.
[25] G. Ananthanarayanan and I. Stoica, "Blue-Fi: Enhancing Wi-Fi Performance using Bluetooth Signals," in *ACM MobiSys*, 2009.
[26] R. C. Shah, L. Nachman, and C.-Y. Wan, "On the performance of Bluetooth and IEEE 802.15.4 radios in a body area network," third International Conference on Body Area Networks (BodyNets'08).
[27] J. Ylitalo, T. Jokikyyny, T. Kauppinen, A. J. Tuominen, and J. Laine, "Dynamic Network Interface Selection in Multihomed Mobile Hosts," in *IEEE HICSS*, 2003.
[28] R. Friedman, A. Kogan, and Y. Krivolapov, "On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones," in *IEEE INFOCOM*, 2011.
[29] R. Friedman and A. Kogan, "Efficient Power Utilization in Multi-radio Wireless Ad Hoc Networks," in *Springer-Verlag Berlin Heidelberg*, 2009.
[30] Y. Agarwal, T. Pering, R. Want, and R. Gupta, "SwitchR: Reducing System Power Consumption in a Multi-Client, Multi-Radio Environment," in *ISWC*, 2008.
[31] R. Corvaja, "Qos analysis in overlay bluetooth-wifi networks with profile-based vertical handover," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1679–1690, 2006.
[32] M. Anand, E. B. Nightingale, and J. Flinn, "Self-Tuning Wireless Network Power Management," in *ACM MobiCom*, 2003.
[33] E. Tan, L. Guo, S. Chen, and X. Zhang, "PSM-throttling: Minimizing Energy Consumption for Bulk Data Communications in WLANs," in *IEEE ICNP*, 2007.
[34] X. Zhang and K. G. Shin, "E-MiLi: Energy Minimizing Idle Listening in Wireless Networks," in *ACM MobiCom*, 2011.
[35] J. Manweiler and R. Choudhury, "Avoiding the Rush Hours: WiFi Energy Management via Traffic Isolation," in *ACM MobiSys*, 2011.
[36] T. Armstrong, O. Trescases, C. Amza, and E. de Lara, "Efficient and Transparent Dynamic Content Updates for Mobile Clients," in *ACM MobiSys*, 2006.
[37] B. S. I. Group, "Adopted Bluetooth Core Specifications," *http://goo.gl/MpJJb6*, 2014.
[38] ——, "Bluetooth Smart Ready products," *http://goo.gl/lK2VFH*, 2014.
[39] IEEE, "IEEE 802.11e-2005," *http://goo.gl/vLGpkI*, 2014.

Andrew Pyles Andrew is a Cyber Security Researcher at MITRE Corporation. He received his Ph.D. in Computer Science from William and Mary in 2013. Wireless networking, smartphones, and cyber security are among his research interests.

David T. Nguyen David has been working on his Ph.D. in Computer Science at the College of William and Mary, USA, since Fall 2011. He is working with Dr. Gang Zhou, and his research interests include mobile computing, ubiquitous computing, and wireless networking. Before joining W&M, he was a lecturer in Boston for 2 years. He received his M.S. from Suffolk University (Boston, 2010), and his B.S. from Charles University (Prague, 2007).

Xin Qi Xin Qi received his BSc degrees in computer science from Nanjing University, China, in 2007 and ME Degree from LIAMA, a joint lab between Chinese Academy of Science and INRIA, in 2010, respectively. He is currently pursuing Ph.D. degree in the Department of Computer Science, The College of William and Mary. His research interests are mainly in Ubiquitous computing and mobile systems.

Gang Zhou Dr. Gang Zhou is an Associate Professor in the Computer Science Department at the College of William and Mary. He received his Ph.D. degree from the University of Virginia in 2007 under Professor John A. Stankovic. He has published over 50 academic papers in the areas of smartphones and ubiquitous computing, sensor networks, and wireless communication and networking. The total citations of his papers are more than 3700 according to Google Scholar, among which three of them have been transferred into patents and the MobiSys'04 paper has been cited more than 700 times. Ten of his papers have each attracted more than 100 citations since 2004. He is currently serving in the Journal Editorial Board of IEEE Internet of Things and also Elsevier Computer Networks. He received an award for his outstanding service to the IEEE Instrumentation and Measurement Society in 2008. He also won the Best Paper Award of IEEE ICNP 2010, which was given to only one paper from among the 170 papers submitted (acceptance rate: 18%). He received NSF CAREER Award in 2013. He is a Senior Member of ACM and IEEE.