WiZi-Cloud: Application-transparent Dual ZigBee-WiFi Radios for Low Power Internet Access

Tao Jin[1*], Guevara Noubir[1*], and Bo Sheng[2†]

[1]Northeastern University, Email:{ taojin, noubir }@ccs.neu.edu
[2]University of Massachusetts Boston, Email: shengbo@cs.umb.edu

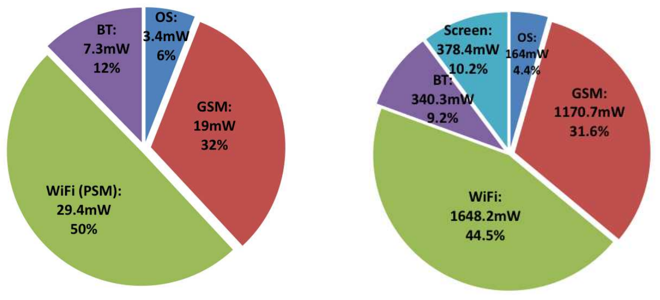
***Abstract*—The high density of WiFi Access Points and large unlicensed RF bandwidth over which they operate makes them good candidates to alleviate cellular network's limitations. However, maintaining connectivity through WiFi results in depleting the mobile phone's battery in a very short time. We propose WiZi-Cloud, a system that utilizes a dual WiFi-ZigBee radio on mobile phones and Access Points, supported by WiZi-Cloud protocols, to achieve ubiquitous connectivity, high energy efficiency, real time intra-device/inter-AP handover, that is transparent to the applications. WiZi-Cloud runs mostly on commodity hardware such as Android phones and OpenWrt capable access points. Our extensive set of experiments demonstrate that for maintaining connectivity, WiZi-Cloud achieves more than a factor of 11 improvement in energy consumption in comparison with energy-optimized WiFi, and a factor of 7 in comparison with GSM. WiZi-Cloud has a better coverage than WiFi, and a low delay resulting in a good Mean Opinion Score (MOS) of 4.26 for a VoIP US cross-country communication.**

I. Introduction

Smartphones are becoming powerful as hardware evolves and their ability has gone far beyond providing telephony services. Nowadays, smartphones are enabling and increasingly large set of applications. More importantly, a lot of Internet based applications, such as web browsers, VoIP, email clients, and instant messengers, have become more and more popular for daily use. Such applications necessitate a reliable and ubiquitous Internet access.

Smartphones typically access the Internet either through cellular networks or WiFi networks. However both these networks have limitations in providing the last mile access. Cellular networks have issues when serving a large volume of clients. In some urban areas, dropped calls can reach 30% [1]–[4]. The service quality and scalability of cellular systems is limited by fundamental constraints. Even if 3G and 4G systems, such as Long Term Evolution (LTE) and WiMax, can provide data rate of tens of megabits per second , this is *shared* among all the users of a base station. Therefore, scaling cellular networks requires a high density of base stations [5] which incurs a substantial cost in terms of sites construction and maintenance.

WiFi networks can significantly help scale wireless access, in cooperation with cellular technologies, especially within urban areas. WiFi networks have the advantage of operating over large license-free bands, and have been densely deployed in urban areas [6]. In addition, WiFi hardware and standards have been well developed for years. However, it is well known that the WiFi interface on mobile devices suffers from high energy consumption even in Power Saving Mode [7]. Although the new phones have shown great improvements, WiFi is still a big energy consumer compared to other components. Fig. 1 shows the breakdown power consumption measured on Android G1 phone, for both idle and active modes. Particularly, our experiments show that WiFi is very inefficient when no traffic is occurring or when the traffic load is low (See Section VI). This is especially limiting for applications requiring continuous reachability such as VoIP but cannot afford the energy cost of periodic wakeups of WiFi.



(a) Radios Idle, Screen Off (b) Radios Active, Screen On

Fig. 1. Android Power Consumption Breakdown.

With the above constraints in mind, we design and develop WiZi-Cloud which utilizes ZigBee to establish an efficient connection between cell phones and access points. We envision that future mobile phones will be equipped with multiple radios that can connect to the Internet, e.g., current mobile phones already have WiFi, Bluetooth [8], and GSM. The ZigBee link we propose will co-exist with other network interfaces. Each of these network interfaces has different characteristics in terms of energy consumption, capacity, and coverage. The mobile phone should be able to determine which interface to carry the packets according to its traffic demands and other system conditions. The ZigBee link we prototyped in WiZi-Cloud is an ultra low power link, but has a limited bandwidth compared to WiFi. It is particularly designed for mobile phone applications with low traffic demand.

In this paper, we propose the architecture, protocols, and hardware/software implementation of WiZi-Cloud with an

[*]Research partially supported by NSF awards 0959584 and 0915985.

[†]This work was done while the author was a postdoctoral fellow at Northeastern University.

emphasis on the following key features.

- *Energy-efficiency*: WiZi-Cloud system is extremely efficient for maintaining connectivity and low rate applications such as VoIP in terms of energy consumption.
- *Leverage of existing HW/SW*: WiZi-Cloud runs on off-the-shelf mobile phones and wireless routers without hardware modifications.
- *Flexibility*: In WiZi-Cloud design, a mobile phone is able to determine the network interface to use according to user-specified policy. WiZi-Cloud provides the mechanism to switch between WiFi and ZigBee interfaces.
- *Seamless*: WiZi-Cloud system and its protocols (e.g., inter-AP handover) is completely transparent to the applications running on the mobile phones and peer entities in the Internet.

To the best of our knowledge, this work is the first prototype that integrates ZigBee into commercial cell phones for Internet access. Also, we have conducted comprehensive experiments and measured realistic performance. Our design details, experience, and the evaluation results will certainly benefit and inspire other similar research work in the community.

In the following sections, we provide an overview of WiZi-Cloud and a summary of results, followed by the related work. In Section IV, we present the WiZi-Cloud system and prototype details. In Section V, we outline the protocols underlying WiZi-Cloud. Section VI summarizes the experimental data collected with our prototype.

II. WiZi-Cloud Overview and Summary of results

The WiZi-Cloud system extends the hardware and network-stack of existing WiFi access points and mobile devices with a set of protocols and mechanisms to support an additional low-power air interface. We chose ZigBee because of its zero-time connection establishment, and good radio range (a significant advantage over Bluetooth). ZigBee is also available as a low cost System on Chip (SoC) with an integrated low power microcontroller such as in the TI CC2530 [9]. These important features allow the mobile phone to be in sleep mode while the microcontroller handles the wakeup and some of the network functionality.

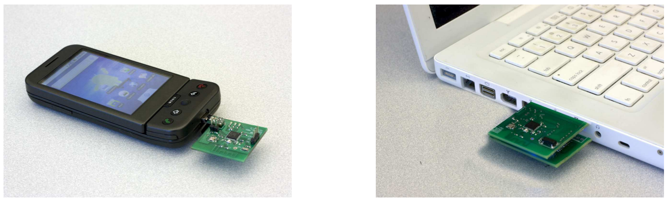

(a) Phone Dongle (b) Laptop Dongle

Fig. 2. WiZi-kit: fully custom made ZigBee modules.

Hardware: On the mobile device the ZigBee is integrated as a low cost accessory, in our case interfacing with an Android phone using the serial link. This could be made more compact by using a ZigBee microSD card [10]. We have prototyped a hardware module, *WiZi-kit*, which integrates TI CC2530, on-board PCB antenna, and connectivity interfaces including UART and FTDI-USB. WiZi-kit can be attached to mobile phones and laptops as a small dongle (See Fig. 2). On the AP, we use OpenWrt compatible access points which gives us hundreds of choices from many manufactures [11]. Our current prototype runs on two particular models, Linksys WRT54GL, and Planex Wireless USB router MZK-W04NU (See Fig. 3). On WRT54GL, the ZigBee is integrated by soldering four wires on the router board. On the Planex router, the ZigBee dongle can be attached to the USB host.

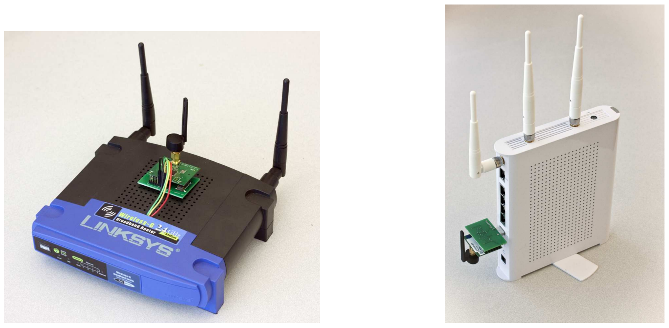

(a) With UART connection (b) With USB connection

Fig. 3. Extended routers of the WiZi-Cloud system.

Software: The network stack of the access point is extended to maintain connectivity with the mobile devices through the ZigBee interface (e.g., beaconing and paging for ZigBee), as well as to coordinate with peer APs to locate mobile devices. The network stack of the mobile device is extended using a virtual network interface through which all traffic is directed. The network stack maintains connectivity at low energy cost (periodic ZigBee wakeup), and seamlessly switches between the WiFi and ZigBee links using an intra-device handover mechanism depending on the traffic load. It also supports handover across a network of WiZi-Cloud access points as the mobile phone roams around. The network stack extensions are designed to be transparent to the application.

While several previous work considered multi-radio interfaces for energy efficiency in wireless networks, and as we discuss in the related work section, no previous solution achieves our target design objectives in terms of seamless communication, low delay, energy efficiency, and minimal hardware/software modifications.

To demonstrate the feasibility and advantages of the proposed approach, we implemented our solution, built a hardware/software prototype, and carried an extensive set of experiments. Below is a summary of our findings:

- *Energy-efficiency*: We show that WiZi-Cloud solution leads to more than a factor of 3 in energy improvement in comparison with an energy-optimized WiFi-based system in standby mode. In active mode, the WiZi-Cloud solution achieves twice more energy efficiency for some mobile applications such as VoIP, and Email.
- *Coverage*: We compare the ZigBee coverage at 4dBm transmit power on channel 26, which is free of WiFi interference, to the 24dBm WiFi coverage when using the most robust WiFi rate (i.e., 1Mbps). This is because the lower ZigBee rate (i.e., 250Kbps) compensates for

the lower transmit power. We also show that ZigBee coverage can be significantly improved by using a RF signal booster, which results in a single WiZi-Cloud AP covering a three floors of a 70 ft. by 250 ft. building.

- *Latency:* When WiZi-Cloud mobile device works in ZigBee mode, the radio can wakeup in 0.75ms. The end-to-end latency includes the transmission time on UART and ZigBee link, the latency along the end-to-end route, and the latency occurred in UART kernel driver. In our prototype, the average one-way client-AP delay is 27ms, and 33ms when packets are tunnelled through two APs.
- *VoIP MOS:* WiZi-Cloud achieves a *good* Mean Opinion Score of 4.26 for a US cross-country communication.

III. Related Work

WiFi energy consumption on mobile phones has attracted a lot of attentions in the literature [12]–[15]. Prior work has considered using alternative low-power wireless links, such as Bluetooth [8], [16], [17] and GSM [7], to help improve the energy efficiency.

One research direction is to keep the WiFi interface off for most of time and turn it back on when needed through other wireless interfaces. In [18], Shih et al. developed an efficient wake-up mechanism particularly for the VoIP service on PDA-based mobile devices using a special low power control channel between the mobile client and a proxy server. Cell2Notify [7] is another work with the same design goal, but targeting regular cell phones with WiFi capability. In Cell2Notify, WiFi is turned on through the cellular network for the incoming VoIP calls. Both [18] and [7] focus on the paging mechanism that wakes up WiFi for VoIP traffic. And their implementations involves additional hardware such as laptops. Our prototype considers not only the paging but also the voice delivery. In addition, our design includes complete protocols, such as handover mechanism for both paging and data delivery. Finally, our system is implemented solely on regular mobile phones without assistance from other devices.

Some other work [16], [17] uses Bluetooth to wake up the WiFi interface. In [16], Agarwal et al. developed a paging scheme assuming each mobile device and the associated AP are connected with a Bluetooth link. Then, WiFi can be turned on via the Bluetooth link. In Blue-Fi [17], the mobile devices predict the availability WiFi connectivity according to the Bluetooth contacts with other nearby Bluetooth devices, and then determine whether to turn on the WiFi. Compared to a Bluetooth link, the Zigbee connection in this paper is significantly superior in terms of handover performance and coverage range. In addition, our system is designed not to wake up the WiFi, but to establish an alternative ZigBee link to carry low rate traffic in a transparent way to the applications.

CoolSpots [8] is a closely related work to this paper. The authors set a Bluetooth link between a mobile device and the associated access point and the traffic can go through either the WiFi or Bluetooth link. CoolSpots focuses on the switching algorithm assuming the bluetooth link has been created by standard. This paper introduces another low-power link using Zigbee which is complimentary with the network interface switching in CoolSpots. In fact, our system can also dispatch packets through different wireless based on specified policy. Furthermore, CoolSpots implements the interface switching by periodically changing the routing rules. Our implementation in this paper supports finer grained control of per packet switch, i.e., the mobile device can determine which network interface to use for each packet.

In addition, VoIP performance in WiFi networks has been well studied in the literature [19]–[21]. They have discussed problematic issues in the current 802.11 for VoIP services and proposed approaches to improve the performance. In our system, the radio link over ZigBee is overwritten from scratch, and their solutions can be easily implemented. But in this paper, we still follow the common 802.11 mechanisms.

Handover of mobile clients in 802.11 and wireless mesh networks has been well studied in the literature [22]–[25]. Their major goal is to reduce the handover delay caused by the sub-processes such as DHCP and AP scanning. In this paper, regular WiFi handover is a part of the handover scheme. Thus, all previous work can be adopted as a component. In constrast, our handover scheme includes additional ZigBee specific functionality. Mobile IP [26]–[28] is close to the tunneling protocol between APs after ZigBee handover in our system. However, our system is more complex as it has to deal with two radio interfaces. Additionally, our design incorporates a paging protocol and achieves much better performance in terms of energy efficiency.

IV. WiZi-Cloud System Design

WiZi-Cloud system consists of a server end and a client end software/hardware support. We built a ZigBee link between each mobile phone client and the associated access point as an ultra low power alternative to the WiFi link. In this section, we present the details of our system design.

A. System Overview

The WiZi-Cloud system is designed as to run below the Internet Protocol layer in the TCP/IP model, and above the link layer. Fig. 4 shows the WiZi-Cloud system framework which consists of three components, Service Module, WiZi-Cloud Bridge & UART I/O, and ZigBee Modem.

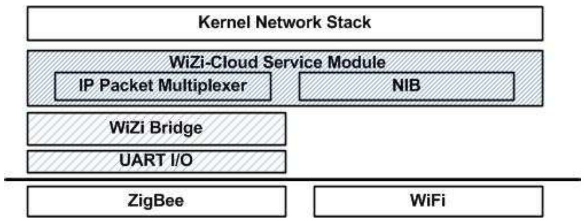


Fig. 4. WiZi-Cloud System Framework.

1) Service Module: The main task of this service module is to distinguish the WiZi-Cloud management traffic from generic IP packets and respectively handle them. For regular IP packets, the service module plays the role of a multiplexer passing packets between the kernel network stack and the active radio interface (either WiFi or ZigBee).

For WiZi-Cloud management messages, such as registration and paging, the Service Module always forwards them to the ZigBee interface. In addition, WiZi-Cloud Service Module maintains a NIC Information Base (NIB) to track the status of the currently active interface for transmission. WiZi-Cloud Service Module has different designs at client and AP. We will discuss the service module in detail in the next subsection.

For management packets and generic IP packets that will be sent through ZigBee, the service module passes the following packet to the lower layer. The first row lists all the fields and the second row indicates the size of each field in Byte.

Type	ZigBee Dst. MAC	LEN	Payload
1	2	2	-

Essentially, the Service Module encapsulates the packets with an extra header containing three new fields. The value of 'Type' distinguishes management packets from data packets. 'ZigBee Dst. MAC' specifies the ZigBee destination and 'LEN' is the length of this message. The field 'Payload' contains the original packet and has varying length depending on the message type. For IP packets, the payload's size is up to the MTU (e.g., 1500 bytes).

2) WiZi-Cloud Bridge & UART I/O: WiZi-Cloud Bridge Module mainly handles fragmentation for the IP packets. In WiZi-Cloud system, the maximum ZigBee frame payload size used in CC2530 network stack is 116 byte, which is much smaller than the IP MTU (1500 byte in Ethernet). Thus, WiZi-Cloud Bridge chops the IP packets from the Service Layer and get each fragment ready to be transmitted with the ZigBee RF. When receiving an IP packet from the ZigBee interface, WiZi-Cloud Bridge buffers all the fragments, reassembles them and forwards the IP packet to the Service Module.

Affiliated with the WiZi-Cloud Bridge, the UART I/O module is responsible for reliable communication on the UART link between WiZi Bridge and the ZigBee device. The message sent through UART has the following format.

SFD	Type	SEQ	ZigBee Dst. MAC	LEN	Payload	CRC	EFD
1	1	1	2	1	103	2	1

Since the data carried on UART is a bit stream, we use a 1-byte start frame delimiter (SFD) and end frame delimiter (EFD) to determine the beginning and the end of a message. In addition, each message indicates its 'Type', either data packet or management packet, such as ACK and UART flow control messages. The maximum payload each message can carry is 103 bytes. Each message in this layer contains a CRC checksum and the receiver side will check the CRC and send an ACK back on a successful delivery. Otherwise, a timer at the host will trigger retransmission.

3) ZigBee Modem: ZigBee Modem provides the host with read and write operations on the ZigBee link. As UART bit streams arrive at ZigBee, ZigBee translates the bits into frame. Upon successful CRC verification, ZigBee sends ACK back to host. The new frame is buffered in egress buffer to be sent through radio to the destination with the following format.

Type	Unique ID	Frag Num	Frag Idx	LEN	Payload
1	2	1	1	1	97

Similarly, as ZigBee receives a packet from the air, it buffers the packet in ingress buffer, and sends to host through UART.

Considering the limited storage space on ZigBee, we have also implemented flow control for UART RX to avoid egress buffer overrun. As egress buffer length crosses threshold, ZigBee sends RNR(Receive Not Ready) or RR(Receive Ready) to the host to request host to pause/resume sending. Since host, e.g. mobile phone, has more UART buffer and faster CPU, we suppose the flow control on the other direction is not required. As we implement the WiZi-Cloud prototype, we learned that it is critical to fully explore the link capacity of both UART and ZigBee radio in order to get good system throughput. Therefore, we also implement windowing logic on UART to pipeline the data flow and use DMA for data transmission.

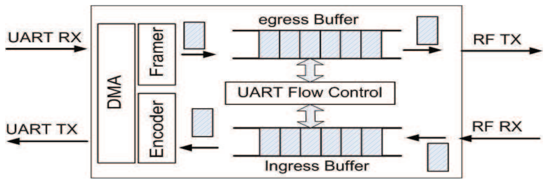


Fig. 5. ZigBee Modem Logic.

B. Service Module Variants

Recall that the WiZi-Cloud Service Module is responsible for managing the dual RF interfaces, and propagating the IP packets to the proper network interface, which makes the underlying interface switching transparent to the kernel network stack and the applications running in the OS. Although the service module on the mobile phone and the AP share the same functionality, the design varies.

1) Virtual Interface at Client: In oder to make the physical interface switching transparent to the rest of the system, the WiZi-Cloud Service Module at client end creates a virtual interface, which is assigned with the same IP address as the one the mobile client obtained from the registration-AP. When the WiFi interface is active, the WiZi-Cloud Service Module sends the IP packets received on the virtual NIC as raw IP packets to the WiFi NIC without any modification, as the virtual NIC has the same IP address with the WiFi interface. When the mobile client switches to the low power ZigBee interface, or moves to another primary-AP, the virtual interface keeps the same IP address so that the active connections can be maintained. All the IP traffic will be passed to the WiZi-Cloud Bridge, and converted to WiZi-Cloud packets. Similarly, the incoming packets that arrive on either WiFi or ZigBee interface will be reassembled to IP packets, propagated to the service module and reinjected to the kernel network stack as a raw IP packet. Having all traffic propagate through virtual NIC makes underlying interface changes transparent to the applications. Besides, we can have finer granularity of traffic monitoring and can determine which interface to use at any moment.

2) Netfilter Extension at AP: Compared with the client, the AP has a different role in the wireless LAN. The AP works as a gateway to route packets between different clients, or route the packets between the internal LAN and the external backbone network, carrying functions such as address translation. The

AP is primarily about a set of policies as to how to route packets for each client. Considering the differences between the AP and the client, we chose a different solution when we designed the WiZi-Cloud Service Module for the AP, which is based upon the Linux netfilter framework. Instead of working as a virtual network interface between the kernel network stack and the WiZi-Cloud framework, the WiZi-Cloud Service Module dynamically changes the iptables rules to determine the IP packet propagation path for certain clients. As shown in Fig. 6, normal IP packets follow path 1. When an IP packet arrives at the AP either on the WAN or the WLAN interface, the netfilter framework, kernel network stack and routing module work together to carry the address translation and route this IP packet to the proper interface. For the client that is registered as ZigBee active, the AP will insert an iptables rule such that all the packets for this client will be queued to our WiZi-Cloud Service Module process.

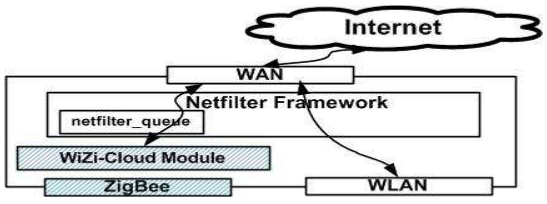


Fig. 6. WiZi-Cloud Service Module at AP

V. WiZi-Cloud Protocols Design

THe WiZi-Cloud system relies on several mechanisms, (1) registration of the mobile device, (2) maintaining reachability, (3) paging, and (4) handover.

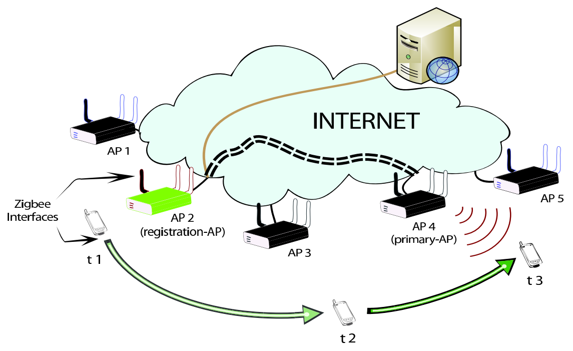


Fig. 7. Dual radio mobile device moving across the WiZi-Cloud system.

A. Registration

A mobile device first associates with one AP in the WiZi-Cloud system, which is denoted by *registration-AP*, and obtains an IP address through DHCP. As the mobile device travels across the WiZi-Cloud network, it may obtain a new IP address from new APs, but the original IP is always bonded to the virtual interface with no change. This has the advantage of making the network connectivity changes transparent to the applications. The mobile device has to update the registration-AP with its current location to allow the tunnelling of packets to the current AP, which is denoted by *primary-AP*. The application packets from the mobile device can be transmitted over either the ZigBee or WiFi interface to the primary-AP and then tunnelled to the registration-AP which forwards them to their destination. If the mobile device only runs applications that periodically check changes in the IP address (such as some VoIP clients), the mobile device can reduce the cost of tunnelling by re-registering at a primary-AP.

B. Ubiquitous reachability

In order to guarantee ubiquitous reachability, the mobile devices need to be covered by a WiZi-Cloud access point, and they need to inform the system on how they can be reached. We propose a beaconing mechanism that aims at reducing the energy consumption of the mobile devices while still maintaining the complexity of the overall system low.

Access Points: Similar to WLANs, APs periodically broadcast beacons using ZigBee every T_{BC} units of time. The APs do not have to be synchronized with each other. The beacon interval depends on the APs density and target energy consumption. A typical value used in our system is 100ms.

Mobile Devices: The mobile devices periodically wake up to listen for the beacons. A mobile device is synchronized to the *primary-AP*. If it does not hear the beacon, the mobile device remains awake for several periods and collects all the beacons it hears from nearby APs. The mobile device also maintains information about the APs that cover his current location, called *Coverage Set*. If the link to the primary-AP is lost or significantly degraded, the mobile device can select another AP as the primary-AP, preferably from the old Coverage Set. If the mobile device notices a significant change in the *Coverage Set*, or in the link quality to the primary-AP, it informs the registration-AP of this change. The registration-AP updates its database with the new primary-AP information and the Coverage Set for this mobile device. The use of a Coverage Set has the advantage of limiting the number of updates sent by the mobile device, specially if the mobile device remains within an area covered by a small number of APs (e.g., building, or campus).

Fig. 8 illustrates the wakeup pattern of a mobile device following the trajectory. Before registration, the mobile device scans the medium and identifies AP_2 and AP_1 as the best covering APs. The mobile device registers with AP_2 and provides $\{AP_2, AP_1\}$ as the Coverage Set. The mobile device now wakes-up only to listen to the beacon of AP_2. After moving away it stops hearing the beacon of AP_2. It scans the medium again, identifies AP_3 as the primary-AP and $\{AP_3, AP_4\}$ as the Coverage Set. It then updates the registration-AP (i.e., AP_1) with the new primary-AP and Coverage Set. When the mobile device moves out of the range of AP_3, it locks on AP_4. It does not have to update the registration-AP because AP_4 is already in the Coverage Set.

C. Paging mechanism

Upon incoming traffic for a mobile device, the registration-AP needs to inform the mobile device to wakeup and start receiving data packets. This is done by extending the beacon message with a paging message. The paging includes a list of mobile devices that need to wakeup. First, the registration-AP

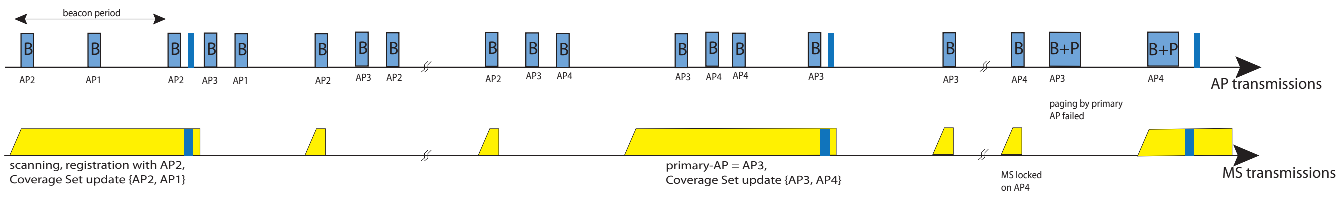

beacon period
B
AP2
AP1
AP3
AP4
B+P
AP transmissions
paging by primary AP failed
scanning, registration with AP2, Coverage Set update {AP2, AP1}
primary-AP = AP3, Coverage Set update {AP3, AP4}
MS locked on AP4
MS transmissions

Fig. 8. Wakeup pattern and messages during mobility of MS1 according to Fig. 7.

informs the primary-AP to page the mobile devices, and the paged devices acknowledge the receipt of the paging message. Second, if the primary-AP fails, all the APs in the Coverage Set are requested to page the mobile device. Such a two-phase mechanism has the advantage of keeping the traffic low, without decreasing the chances to reach the mobile device. This comes at the expense of a potentially higher delay when the mobile device is no more covered by the primary-AP.

Fig. 8 illustrates the paging mechanism. Some traffic is sent towards the mobile device when it is locked on AP_4 but the current primary-AP is AP_3, and the current Coverage Set is $\{AP_3, AP_4\}$. The registration-AP pages the mobile device on the primary-AP AP_3, however the attempt fails. Then the registration-AP pages all the APs in the Coverage Set. AP_4 succeeds in reaching the mobile device. The registration-AP can now tunnel the traffic to the mobile device through AP_4.

D. Handover

The WiZi-Cloud system supports multiple forms of handover with the goal to minimize energy consumption, and connectivity disruption.

1) Intra-device handover and traffic scheduling: While the ZigBee link is significantly more energy efficient than the WiFi link, it can only sustain a limited load. The WiZi-Cloud AP has a traffic scheduler that monitors the network traffic on the ZigBee link and instructs mobile devices to switch-on their WiFi interface and communicate over it. Only, the mobile devices with the lowest rate remain on the ZigBee interface. The ZigBee interface remains active until when the WiFi association is complete.

2) Seamless inter-AP handover: When moving, the mobile device only updates the Coverage Set and the primary-AP information. The mobile device is always reachable at the best covering AP through paging. For delay-insensitive sessions, the mobile device can switch to a new WiZi-Cloud AP, and update the primary-AP information at the registration-AP. For delay-sensitive sessions (e.g., VoIP), the mobile device initiates a WiFi association with a new AP, and then sends a primary-AP update. The mobile device achieves a seamless handover by maintaining both the ZigBee link to the old AP, and the WiFi link to the new AP.

E. Stateless vs. stateful sessions

In characterizing the performance of the WiZi-Cloud system, one can note that stateless sessions, such as web browsing, is not negatively impacted by the proposed mechanisms, since such traffic can still go through the physical WiFi or ZigBee interface without tunnelling. The dual-radio allows for a reduction in energy consumption when the data rate is low. Stateful traffic such as VoIP and mobility unaware applications can operate in a transparent and energy-efficient way. Even, network aware applications (e.g., SIP clients that periodically check IP address changes and update the SIP server) benefit through a reduction in the number of registrations and update messages and through the handover capability of the WiZi-Cloud system.

F. Security and Privacy Considerations

Our goal so far is to demonstrate the performance advantages of dual WiFi-ZigBee radios in providing energy-efficient ubiquitous reachability that is seamless to the mobile phone applications. For a real world deployment of a WiZi-Cloud system many security issues have to be addresses, including privacy (both in terms of communicated data, anonymity to prevent users tracking), robust reachability (e.g., poisoning registration-AP), and DoS protection (both on energy and load). We believe that these considerations can be appropriately addressed with adequate mechanisms. We plan to investigate them in our future research.

VI. Performance Evaluation

In this section, we evaluate the performance of our WiZi-Cloud prototype with an extensive set of experiments . We will evaluate the overall system performance on the Android G1 integrated with the WiZi-Cloud system, from the perspectives of energy consumption, throughput and user experience.

A. Energy Consumption

The energy consumption is one of the most important metrics in our experiments. First, we show the breakdown of energy consumption measured with Android G1 in Table I. To measure the phone energy consumption, we power the phone with an external power generator (4.1V), and connects the Agilent U1252A multimeter in series. The multimeter logs the instantaneous current value every 5ms.[1] The result shows that ZigBee in idle mode achieves more than factor of 11 improvement in energy consumption in comparison with WiFi in Power Saving Mode, and a factor of 7 in comparison with GSM. However, energy usage of the radio interface cannot tell the whole story. Due to the low data rate and limited computation capability of ZigBee chipset, it may not be suitable for all applications and it is important to study how ZigBee would impact the overall system energy usage.

Next, we present the experimental data collected from real mobile applications running on the Android G1 phone with our WiZi-Cloud prototype. We will discuss the application

[1]The ZigBee entry in Table I is the energy used by a standalone ZigBee hardware with 3.5V power, excluding the energy cost by WiZi stack.

	GSM	WiFi	Bluetooth	ZigBee	OS	Screen
RF Idle	19.04	29.42	7.32	2.57	3.54	378.144
RF Active	1170.71	1648.2	340.3	94.5		

TABLE I
BREAKDOWN ENERGY CONSUMPTION ON ANDROID PHONES IN MW.

performance from two perspectives: *feasibility* and *energy consumption*. We categorize the mobile applications into three classes by two criteria, *latency sensitivity* and *network traffic load* (See Table II). Applications such as VoIP, requires limited bandwidth. For example, the GSM codec for VoIP consumes 20Kbps bandwidth each direction. However, the VoIP application is highly sensitive to latency and jitter because late packets are discarded which leads to a significant degradation of the voice quality. In contrast, Email has a reasonable tolerance to latency, and consumes limited bandwidth. Applications such as Web browsing, may consume much higher bandwidth, due to the rich media content on the web page. Although it is not a real time application, a long delay may hurt the user experience, as well as the phone energy consumption.

sample app	latency sensitivity	traffic load
VoIP, stream media	moderate	moderate
Email	moderate	moderate
Web	low	high

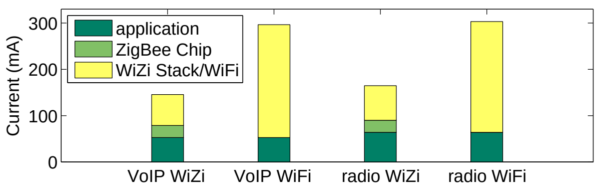
TABLE II
MOBILE APPLICATION CATEGORIES

1) High Delay Sensitivity, Moderate Traffic Load: We tested a VoIP application called sipdroid with two popular codecs, GSM 13Kbps and Speex 11Kbps. The voice is clear, however sipdroid does not report any statistical data indicating the call quality. We capture the sipdroid traffic, and use iperf to emulate the VoIP traffic by generating two-way UDP flow, with proper packet size and packet rate. The traffic pattern, plus the bandwidth and jitter reported by iperf are listed in Table III. We will show that the obtained values correspond to a very good VoIP Mean Opinion Score in Section VI-C.

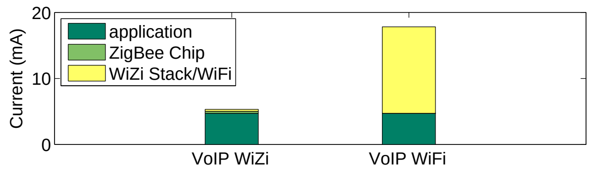
codec	pkts/sec (two way)	UDP pkt size (B)	BW (Kbps)	jitter (ms)
GSM	95	53	39.3	4.38
Speex	97	49	37.1	3.86

TABLE III
VOIP CLIENT TRAFFIC PATTERN.

To further verify the suitability of the WiZi system for delay sensitive applications, we tested an Internet Radio application called *iheartradio*, which runs over the TCP. One local Boston music channel kiss108 consumes about 49Kbps bandwidth, with an average TCP packet size of 214 Byte. iheartradio also delivers a very good quality on the WiZi system. Fig. 9(a) shows the total energy consumption by sipdroid and iheartradio in active mode, in which sipdroid is making a voice call and iheartradio is streaming music. Each bar consists of three components: 1) the base energy usage, including the energy consumed by the OS, speaker, and application; 2) the energy consumed by the WiFi or the whole WiZi software stack; 3) the energy consumed by the external ZigBee hardware (none in WiFi case). In this type of applications, packets come at a fast pace, which prevents both WiFi and WiZi from entering the power save mode. This results in a high WiFi energy consumption, of around 250mA in both applications. In contrast, ZigBee consumes only around 27mA even in active mode. Since our WiZi stack runs as a user space program, the energy usage of the WiZi software stack takes a large portion. However, the WiZi system still reduces the overall system energy consumption by 50%. As shown in Fig. 9(b), when sipdroid is in standby mode, WiZi system shows even higher energy efficiency because the energy usage by the ZigBee hardware and WiZi stack is very little. The phone standby time with VoIP software is extended by three times.



(a) sipdroid & iheartradio, active mode



(b) sipdroid, standby mode

Fig. 9. Energy consumption of sipdroid and iheartradio on G1, with WiZi or WiFi, screen off.

2) Moderate Delay Sensitivity, Moderate Traffic Load: In this section, we experiment with an email application on the G1. We captured the email traffic for three tasks, checking email, sending one email, and checking and downloading one email. We set up the G1 email client with one graduate student's school email account, and profiled the email traffic for 10 days. We generated traffic with the same average packet size and average packets per second, and measured the overall system energy consumption. Table IV lists the average duration of each operation, and the average current drained. In our experiment, the average email traffic is limited, which allows both WiFi to function in power save mode during each operation. However, the ZigBee frames carrying IP fragments happens three times more frequently than WiFi, which forces the ZigBee device to remain in active mode. In this, case WiZi is comparable with WiFi in terms of total energy usage.

	Duration (s)		Current (mA)		Energy (Joule)	
	WiFi	WiZi	WiFi	WiZi	WiFi	WiZi
Send	8.082	7.044	7.60	35.75	2.014	1.03
Check	7.587	8.244	26.01	42.18	0.89	1.43
Download	14.399	10.734	28.42	36.17	1.678	1.59

TABLE IV
EMAIL APPLICATION PROFILE, SCREEN OFF.

3) Moderate Delay Sensitivity, Moderate Traffic Load: We experiment with Web browsing on the G1. We visited the Google Reader web site, and loaded the top 14 news feeds in the Engadget channel. We counted the time to load all the text and image content for these 14 news, and the total traffic generated. In this experiment, there are in total 1216 IP packets, the average IP packet size is 710 Byte. Web browsing is an interactive application, so we kept the screen ON during the whole experiment. As shown in Table V, even though ZigBee is occasionally more energy efficient, it usually takes much longer to finish loading the content, which result in almost twice more energy consumption. In this case, the screen, another major energy draining source, becomes the bottleneck. Besides, the long loading time degrades the user experience. Due to the slow link speed of ZigBee, WiZi system does not provide any benefit to such applications which generate bursty traffic, and require user interaction.

	avg current (mA)	loading time (sec)	energy (Joule)
WiZi	199.606	239.8	196.248
WiFi	294.73	93.411	112.88

TABLE V
OVERALL SYSTEM ENERGY CONSUMPTION OF WEB BROWSER.

B. *Throughput*

This experiment was carried out in the campus LAN, the phone accesses network through a WiZi-enabled AP. The end host is a Linux PC. All experiments were carried out with a good link quality. The throughput is measured by iperf with a duration of 30 seconds. For each particular parameter setting we conduct 10 iperf trials and report the average value.

1) UDP Throughput: We first measure the UDP throughput for different UDP payload size. Fig. 10 shows the UDP throughput and variance. When payload size is smaller than maximum ZigBee payload size, the WiZi-Cloud packet header incurs a large overhead yielding a low throughput. As the payload increases, the throughput quickly increases due to the better utilization of the ZigBee channel. When the payload exceeds 500 bytes, the curve becomes flat, because the whole data flow along the WiZi, UART, and radio link is efficiently pipelined. In our experiment, the peak UDP throughput is 70.4Kbps with 1400 Byte payload and the UART link throughput (including headers overhead) is 83Kbps, which is close to our prototype UART link limit (115Kbps).

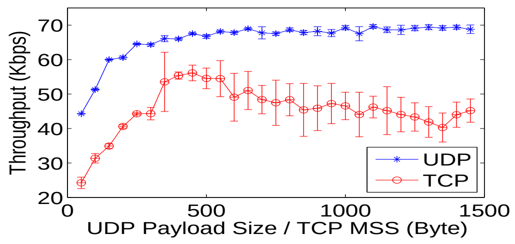


Fig. 10. WiZi TCP / UDP Troughput vs. TCP MSS . UDP Payload Size.

2) TCP Throughput: In the TCP scenario, traffic occurs in two directions. The ZigBee device is carrying out four tasks, Tx/Rx on UART and Tx/Rx on radio. As ZigBee radio receives messages from the air, it also receives messages from UART, which needs to be sent out through RF. Thus, the ZigBee cannot send the messages in the ingress buffer to the host in a timely manner. When messages arrive at the radio too frequently, due to the slow UART link, the ingress buffer will be full and start discarding the incoming RF message. If one IP packet fragment is lost, all the rest of the fragments will be of no use. Thus, the maximum TCP packet size (MSS) becomes a trade off between better channel utilization and the risk of wasting bandwidth. As shown in Fig. 10, the optimal TCP MSS is 450 Byte, achieving 60.2Kbps throughput.

C. *Mean Opinion Score for VoIP*

In this subsection, we evaluate the performance of VoIP applications running over WiZi. The commonly accepted metric for QoS of VoIP is the Mean Opinion Score (MOS) which ranges from 1 to 5[2]. According to [29], MOS can be approximately derived from the R-factor as follows,

$$1 + 0.035R + 7 \cdot 10^{-6}R(R-60)(100-R), \text{if } 0 < R < 100.$$

The R-factor is defined as $R = 100 - I_s - I_d - I_{ef} + A$, where I_s is the signal-to-noise impairment, I_d is the mouth-to-ear delay of the path, I_{ef} is the equipment impairment, and A is the expectation factor. According to [29], the R-factor can be simplified as $R = 94.2 + 0.024d + 0.11(d - 177.3)H(d - 177.3) - 11 - 40\ln(1 + 10e)$, where d is the one-way delay and H is the Heavyside function,

$$H(x) = \begin{cases} 1 & \text{if } x > 0; \\ 0 & \text{otherwise.} \end{cases}$$

Following the general setting in [29], $d = d_{network} + 85ms$, where $d_{network}$ is the network delay.

We generate a 20-byte packet every 20ms and measure the one way network delay (<30ms). We also add on 40ms for cross-country delay. The final MOS is 4.26 which matches the very good experience we had with the Sipdroid application.

D. *Coverage Performance (ZigBee vs. WiFi)*

For the paging mechanism, a better coverage means more reliable link between the primary AP and the mobile device, and fewer updates needs to be sent to the registration-AP. In this section, we compare the coverage of ZigBee and WiFi, and use *packet loss rate* to represent the coverage performance.

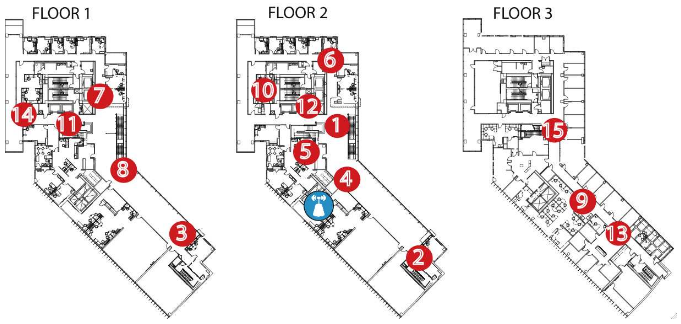


Fig. 11. College's building floor plan with location of measurements points.

We carried out the experiments in our College facility, a three-floor building (shown in Fig. 11). A broadcasting node is placed in the blue spot and a mobile receiver measure the packet loss rate at 15 different locations. In the ZigBee tests, the sender uses channel 26, one of the WiFi interference free channels, and 4dBm Tx power, the maximum manufacturer recommended Tx power. In the WiFi tests, we use a regular wireless AP (24 dBm Tx power) as the broadcasting node. As shown in Fig. 12, ZigBee has a better coverage than WiFi within a range of around 50ft. Even though WiFi transmits with higher energy, ZigBee has a higher E_b/N_0 than WiFi, which results in lower packet loss rate. Beyond that

[2]MOS: (5) Excellent/Imperceptible, (4) Good/Perceptible but not annoying, (3) Fair/Slightly annoying, (2) Poor/Annoying, (1) Bad/Very annoying

range, however, the ZigBee performance degrades significantly because the RSSI level drops below the RF sensitivity threshold of the CC2530. In contrast, WiFi performance gradually degrades. Furthermore, we have measured the coverage of an enhanced ZigBee sender equipped with a 27dBm signal booster in Fig. 12. The "good" ZigBee coverage is extended to around 100ft, which can cover almost the entire building.

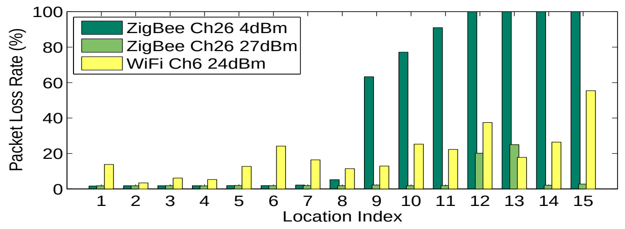


Fig. 12. Packet loss rate of ZigBee on channel 26 at 4dBm and 27dBm, vs. WiFi on channel 6 at 24dBm.

VII. Discussion

Our prototype WiZi-Cloud system can provide enough throughput to some mobile applications, such as VoIP and stream radio, and achieves significantly better energy efficiency than WiFi. We believe the system performance can be further optimized by alleviating the following bottlenecks:

- Android G1 UART module supports up to 115Kbps, which is less than 50% of ZigBee data rate, 250Kbps. The UART link is the key bottleneck in our prototype. We are currently working on integrating ZigBee with Ethernet and Bluetooth interfaces, so that the ZigBee device can connected with AP and mobile phones through high speed link. We expect to boost the throughput performance by two times, which also benefits the energy efficiency.
- The WiZi stack is currently running as a user space program, which generates extra computation while interacting with the kernel. This results in extra energy consumption, as shown in Fig. 9. By integrating the stack to the kernel module, we expect to further increase the energy efficiency.

VIII. Conclusion

We propose WiZi-Cloud, a network architecture, set of mechanisms, and HW/SW system solution to achieve an energy efficient, ubiquitous and real time reachability that is transparent to applications. We have prototyped WiZi-Cloud on commodity mobile phones and WiFi APs. Our extensive set of experiments demonstrate that ZigBee achieves a factor of 11 better energy efficiency than WiFi in Power Saving Mode. With all system energy usage counted, WiZi still can be 2 times more energy efficient than an optimized WiFi while active transmitting, and standby lifetime can be extended up to 3 times. Similar results apply to GSM, as well. Besides, WiZi-Cloud has better coverage than WiFi within 50ft indoor environment. Finally, in the case of VoIP delivery over WiZi-Cloud, a good Mean Opinion Score of 4.26 is achieved.

References

[1] "iPhone vs. Pre: Satisfaction Bakeoff." [Online]. Available: http://brainstormtech.blogs.fortune.cnn.com/2009/08/14/iphone-vs-pre-satisfaction-bakeoff/

[2] "Customers Angered as iPhones Overload AT&T." [Online]. Available: http://www.nytimes.com/2009/09/03/technology/ companies/03att.html

[3] "Mobile Broadband Still Crawling at Below 1Mb, Despite 'up to' 7.2Mb Claims." [Online]. Available: http://mobile.broadbandgenie.co.uk/broadband-news/mobile-broadband-still-crawling-at-below-1mb-despite-up-to-7mb-claims

[4] "Apple Genius Bar: iPhones' 30% Call Drop Is "Normal" in New York." [Online]. Available: http://gizmodo.com/5370493/apple-genius-bar-iphones-30-call-drop-is-normal-in-new-york

[5] M. Rumney, "Identifying Technology to Deliver the Next 100x Capacity Growth in Wireless," *The 3rd LTE World Summit*, 2008.

[6] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden, "A Measurement Study of Vehicular Internet Access Using In Situ Wi-Fi Networks," in *MobiCom '06*.

[7] Y. Agarwal, R. Ch, A. Wolman, P. Bahl, K. Chin, and R. Gupta, "Wireless Wakeups Revisited: Energy Management for VoIP over Wi-Fi Smartphones," in *MobiSys 2007*, Puerto Rico, 2007, pp. 179–191.

[8] T. Pering, Y. Agarwal, R. Gupta, and C. Power, "CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces," in *MobiSys 2006*, Uppsala, Sweden, 2006.

[9] Texas Instruments, "CC2530 - A True System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications," April 2009. [Online]. Available: http://focus.ti.com/lit/ds/symlink/cc2530.pdf

[10] Spectec Computer Co., "microSD ZigBee card." [Online]. Available: http://www.spectec.com.tw/sdz537.htm

[11] "OpenWrt Hardware List." [Online]. Available: http://wiki.openwrt.org/oldwiki/openwrtdocs/hardware

[12] D. Bertozzi, A. Raghunathan, L. Benini, and S. Ravi, "Transport Protocol Optimization for Energy Efficient Wireless Embedded Systems," in *Design, Automation and Test in Europe Conference and Exhibition*, 2003.

[13] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access Using Bounded Slowdown," in *MOBICOM 2002*.

[14] J. Liu and L. Zhong, "Micro Power Management of Active 802.11 Interfaces," in *MobiSys*, 2008.

[15] D. Qiao and K. Shin, "Smart Power-Saving Mode for IEEE 802.11 Wireless LANs," in *INFOCOM 2005*, March 2005.

[16] Y. Agarwal, R. Gupta, and C. Schurgers, "Dynamic Power Management Using On Demand Paging for Networked Embedded Systems," in *Proceedings of the 2005 Conference on Asia and South Pacific Design Automation*, vol. 2, Jan. 2005, pp. 755–759 Vol. 2.

[17] G. Ananthanarayanan and I. Stoica, "Blue-Fi: Enhancing Wi-Fi Performance Using Bluetooth Signals," in *MobiSys 2009*.

[18] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on Wireless: : An Event Driven Energy Saving Strategy for Battery Operated Devices," in *MobiCom 2002*, Atlanta, Georgia, USA, 2002.

[19] P. Verkaik, Y. Agarwal, R. Gupta, and A. C. Snoeren, "Softspeak: Making VoIP Play Well in Existing 802.11 Deployments," in *NSDI'09*, 2009, pp. 409–422.

[20] S. Shin and H. Schulzrinne, "Experimental Measurement of the Capacity for VoIP Traffic in IEEE 802.11 WLANs," *IEEE Transaction on Mobile Computing*, 2009.

[21] F. Guo and T. cker Chiueh, "Software TDMA for VoIP Applications Over IEEE802.11 Wireless LAN," in *INFOCOM*, 2007, pp. 2366–2370.

[22] Amir, Yair and Danilov, Claudiu and Hilsdale, Michael and Musăloiu-Elefteri, Raluca and Rivera, Nilo, "Fast Handoff for Seamless Wireless Mesh Networks," in *MobiSys '06*.

[23] R. Hsieh, Z. G. Zhou, and A. Seneviratne, "S-MIP: A Seamless Handoff Architecture for Mobile IP," in *Proceedings of INFOCOM*, 2003.

[24] R. Hsieh and A. Seneviratne, "A Comparison of Mechanisms for Improving Mobile IP Handoff Latency for End-to-End TCP," in *MobiCom '03*.

[25] I. Ramani and S. Savage, "SyncScan: Practical Fast Handoff for 802.11," in *Proceedings of IEEE Infocom*, 2005.

[26] "RFC 3344 - IP Mobility Support for IPv4."

[27] "RFC 3024 - Reverse Tunneling for Mobile IP."

[28] U. Jönsson, F. Alriksson, T. Larsson, P. Johansson, and G. Q. Maguire, Jr., "MIPMANET: Mobile IP for Mobile Ad Hoc Networks," in *MobiHoc '00*.

[29] R. G. Cole and J. H. Rosenbluth, "Voice over IP Performance Monitoring," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 2, pp. 9–24, 2001.