WiZi-Cloud: Application-transparent Dual ZigBee-WiFi for Mobile Internet

Tao Jin, *Member, IEEE,* Guevara Noubir, *Senior Member, IEEE*

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. To demonstrate the feasibility of WiZi-Cloud system, we carry out an extensive set of experiments with real Android mobile applications on Android G1 phone, and evaluate the performance in terms of throughput, energy efficiency and coverage. With VoIP and audio streaming Android applications running over ZigBee and WiFi respectively, we show that WiZi-Cloud achieves satisfactory throughput performance, and leads to 2 times better energy efficiency in active mode, 3 times in standby mode. While the basic WiZi-Cloud system already has a better coverage than WiFi, we describe and experimentally demonstrate how further improvements can be achieved using time and spatial diversity mechanisms.

Index Terms—ZigBee, WiFi, mobile devices, energy saving, multi-radio

1 Introduction

SMARTPHONES are becoming powerful as hardware evolves and their ability has gone far beyond providing telephony services. Nowadays, smartphones are enabling an 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 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.

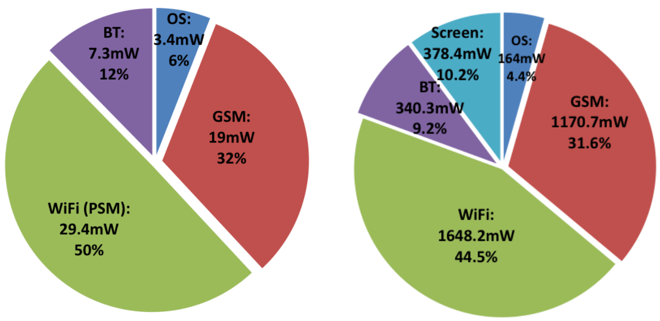
- *T. Jin and G. Noubir are with the College of Computer and Information Science, Northeastern University, Boston, MA, 02115.*
  E-mail: noubir@ccs.neu.edu

- *More details about WiZi-Cloud project can be found at web page http://www.ccs.neu.edu/home/noubir/projects/wizi/*

- *Research partially supported by NSF awards 0959584 and 0915985.*
- *A shorter version appeared in IEEE Infocom 2011.*

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 (i.e., 80 MHz in the 2.4GHz ISM band, and 240 MHz in the 5.15- 5.824 GHz U-NII 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 power consumption breakdown 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 6). This is especially limiting for applications which require continuous reachability such as VoIP , but cannot afford the energy cost of periodic wakeups of WiFi.

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 and Bluetooth [8] radios in addition to a cellular interfaces. Some phones also have a ZigBee interface [9] or are capable of integrating it as a microSD card [10]. The ZigBee link we propose will co-exist with other network interfaces. Each of these network interfaces has



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

Fig. 1: Android Power Consumption Breakdown.

different characteristics in terms of energy consumption, capacity, and coverage. Therefore, the mobile phone should be able to choose the optimal interface to carry the packets according to its traffic demands and other system conditions. However, WiZi-Cloud is not a one-size-fits-all solution. The ZigBee link prototyped in WiZi-Cloud is an ultra low power link, but has a limited bandwidth (with 250Kbps data rate) compared to WiFi. We do carefully consider the low data rate limitation of ZigBee while designing WiZi-Cloud. Fig. 2 shows a list of popular mobile applications and their required bandwidth consumption. These applications such as Skype, Pandora, IM, etc., do not make use of screen and therefore the radio is the main energy sink. Also, to maintain connectivity, such applications generally result in much faster energy draining. Whereas, with 250Kbps data rate, ZigBee is capable of carrying traffic for many well demanded mobile applications. With an extensive set of experiments, we evaluate the potential gain of WiZi-Cloud for a variety of mobile applications and show that WiZi-Cloud is best suitable to the mobile phone applications with moderate traffic demand.

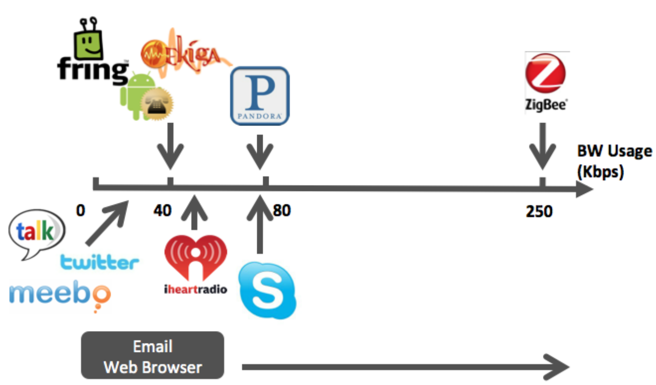


Fig. 2: Popular mobile applications and their required bandwidth.

In this paper, we propose the architecture, protocols, and hardware/software implementation of WiZi-Cloud with an emphasis on the following key features.

- *Energy-efficiency*: the WiZi-Cloud system is extremely efficient for maintaining connectivity and supporting low rate applications such as VoIP in terms of energy consumption.
- *Leveraging of existing HW/SW*: WiZi-Cloud runs on off-the-shelf mobile phones and wireless routers without hardware modifications.
- *Flexibility*: in WiZi-Cloud a mobile phone is able to determine the adequate network interface to use according to the user-specified policy. WiZi-Cloud provides intra-device handover mechanism to switch between WiFi and ZigBee interfaces.
- *Seamless*: WiZi-Cloud system and its protocols (e.g., intra-device and inter-AP handover) are completely transparent to the applications running on the mobile phones and peer entities in the Internet.
- *Coverage*: WiZi-Cloud achieves a better coverage than WiFi thanks to the higher E_b/N_0 of ZigBee and the coverage can be further improved with commonly used diversity mechanisms such as spatial and time diversity.

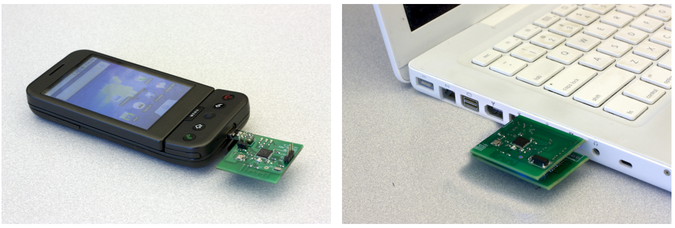
To the best of our knowledge, this work is the first set of protocols that are implemented into a prototype that integrates ZigBee into commercial mobile phones for Internet access. Also, we have conducted a comprehensive set of experiments and measured realistic performance to evaluate the proposed architecture, and mechanisms. For a real world deployment of WiZi-Cloud, many security issues need to be addressed. For example, proper encryption and authentication mechanism over the ZigBee link is required to enforce the privacy and access control. At this moment, we are not considering the security issues, and plan to investigate then in our future research.

In the following sections, we provide an overview of WiZi-Cloud and a summary of results, followed by the related work. In Section 4, we present the WiZi-Cloud system and prototype details. In Section 5, we outline the protocols underlying WiZi-Cloud. Section 6 summarizes the experimental data collected with our prototype. Two potential mechanisms to extend the coverage are empirically evaluated in Section 7.

2 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 choose 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) solution with an integrated low power microcontroller such as in the TI CC2530 [11]. These important features allow the mobile phone to be in sleep mode while the ZigBee module handles the wakeup and some of the network functionality.

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

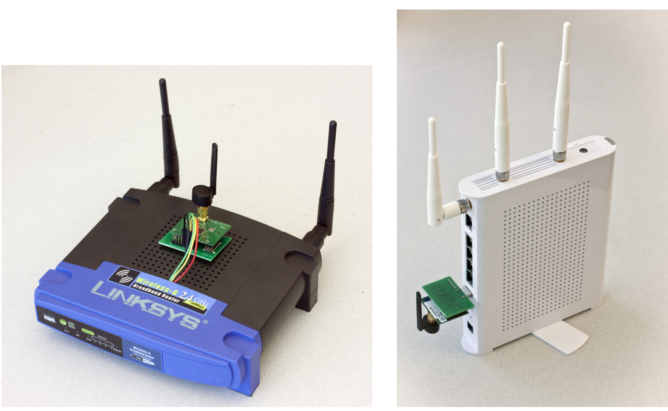


(a) Phone Dongle (b) Laptop Dongle

Fig. 3: WiZi-kit: fully custom made ZigBee modules.

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. 3).

On the AP, we use OpenWrt compatible access points which gives us hundreds of choices from many manufactures [12]. Our current prototype runs on two particular models, Linksys WRT54GL and Planex Wireless USB router MZK-W04NU (See Fig. 4). On WRT54GL, the ZigBee is integrated by soldering four wires on the router board. On the Planex router, the ZigBee dongle can be directly attached to the USB host.



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

Fig. 4: 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, as we discuss in the related work section, no previous solution achieves our target design objectives in terms of seamless communication, low delay, high energy efficiency, and minimal hardware/software modifications.

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

- *Energy-efficiency*:
  ZigBee energy usage is negligible in idle mode. Thus, to maintain connectivity, ZigBee can be used as a paging interface and wakes up WiFi or GSM when needed. In this case, ZigBee achieves a factor of 11 in energy improvement in comparison with WiFi, and a factor of 7 in comparison with GSM.
  Asides from paging, WiZi-Cloud is also designed for the mobile applications with moderate traffic demand. We experiment with VoIP and audio streaming Android applications running over ZigBee and WiFi respectively. In active mode, ZigBee solution leads to a 2 times better energy efficiency in comparison with an energy-optimized WiFi. In standby mode, the battery lifetime can be extended by 3 times with ZigBee solution.
- *Coverage*:
  We compare the ZigBee coverage at 4dBm transmit power on channel 26, which is free of WiFi interference, to the WiFi coverage with 24dBm transmit power and the most robust WiFi rate (i.e., 1Mbps). This is because the lower ZigBee rate (i.e., 250Kbps) compensates for the lower transmit power, and achieves higher E_b/N_0 than WiFi. 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. Besides, we explore two mechanisms, time and spatial diversity, to extend the coverage of ZigBee. Our extensive experiment shows that retransmission and dual-antenna can achieve up to 50% gain to reduce Packet Loss Rate. With further investigation on the bit error pattern, we conclude that the potential gain that is achievable with any Forward Error Correction is negligible.
- *Latency:*
  When the 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.
- *Throughput:*
  In our WiZi-Cloud prototype system, UART link

data rate is the major bottleneck of the complete data path, because Android G1 UART link supports up to 115Kbps data rate. With such constraint, our throughput test achieves up to 70.4Kbps UDP goodput and 60.2Kbps TCP goodput, which translates to around 72% of G1 UART link speed limit. Our demo shows that such data rate is sufficient to achieve satisfactory user experience in VoIP and audio streaming applications [1].

3 Related Work

WiFi energy consumption on mobile phones has attracted a lot of attentions in the literature [13]–[16]. Prior work has considered using alternative low-power wireless links, such as Bluetooth [8], [17], [18] 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 [19], 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 [19] and [7] focus on the paging mechanism that wakes up WiFi for VoIP traffic. Our prototype considers not only the paging but also the data delivery. Also, our system is implemented solely on regular mobile phones without assistance from other devices.

Some other work [17], [18] uses Bluetooth to wake up the WiFi interface. In [17], 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 [18], 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 closely related to our work. The authors establish 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 established. [20] studies the preferred usage patterns for WiFi and Bluetooth to improve the transmission to power ratio. However, the Bluetooth is subject to high connection establishment overhead and much lower coverage compared with ZigBee, which we believe sets a big limitation for the feasibility of CoolSpots solution. Our paper considers an alternative low-power link using ZigBee which complements the network interface switching algorithm in CoolSpots, and features zero connection establishment overhead and comparable coverage with WiFi.

Recently, there has been an increasing interest in using ZigBee as an assisting interface for conserving energy. ZiFi [21] uses ZigBee to detect the existing WiFi networks by identifying the beacon patterns using energy sampling. Esense [22] enables information delivery from WiFi to ZigBee by sending some carefully tailored, encoded energy patterns which rarely happens in real WiFi traffic. [23], [24] proposes a solution to use ZigBee to coordinate the communication activities of WiFi to reduce contention and collision under the DCF. Unlike this work, WiZi-Cloud introduces a complete suite of SW/HW system to enable an alternative ultra-low power ZigBee link between AP and mobile devices for not only paging but also for regular network traffic. A recent work [25] considers using ZigBee as the alternative interface to 3GPP LTE Advanced for an extended power-saving operation. To the best of our knowledge, our work is the first architecture, set of protocols, and prototype that integrates ZigBee into commercial cell phones and access points for seamless and efficient Internet access.

In addition, VoIP performance in WiFi networks has been well studied in the literature [26]–[28]. They have discussed problematic issues in the current 802.11 for VoIP services and proposed approaches to improve the performance. In our system, we redesigned/implemented the ZigBee link/network layer from scratch to support VoIP traffic constraints, while we kept the IEEE802.11 mechanisms unchanged.

Handover of mobile clients in 802.11 and wireless mesh networks have been well studied in the literature [29]–[32]. A major goal of previous work has focussed on reducing the handover delay caused by the discovery and configuration sub-processes such as DHCP and AP scanning. In this paper, the standard WiFi handover is part of our handover scheme. Thus, all previous work can be adopted as a component. In contrast with previous work, our handover scheme includes additional ZigBee specific functionality. Mobile IP [33]–[35] is similar to the tunneling protocol between APs to seamlessly support the 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 supports the energy efficiency goals.

1. See http://www.ccs.neu.edu/home/noubir/projects/wizi for the demo video of Android G1 plays stream radio and makes VoIP calls with ZigBee interface.

4 WiZi-Cloud Architecture and Software Stack

The WiZi-Cloud system incorporates a set of protocols to achieve energy efficient, ubiquitous and real time connectivity. One key enabling technology is to allow both mobile device and AP to seamlessly switch between WiFi and ZigBee interfaces without disturbing ongoing traffic. In this section, we first present the system architecture and the implementation of a complete software stack, called *WiZi stack*. In Section 5, we will present a suite of network protocols that are implemented on top of such architecture.

4.1 System Overview

The WiZi-Cloud system consists of a server-end and a client-end with a 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. The WiZi-Cloud system is designed to run below the Internet Protocol layer in the TCP/IP model, and above the link layer. Fig. 5 shows the WiZi-Cloud software architecture which consists of three components, Service Module, WiZi-Cloud Bridge & UART I/O, and ZigBee Modem.

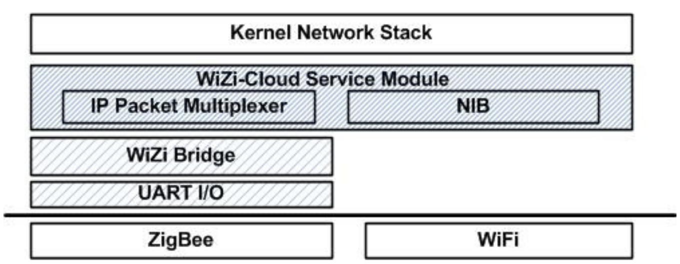


Fig. 5: WiZi-Cloud System Framework.

4.1.1 *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. The 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 size is up to the MTU (e.g., 1500 bytes).

4.1.2 *WiZi-Cloud Bridge & UART I/O*

WiZi-Cloud Bridge Module mainly handles the 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, the WiZi-Cloud Bridge fragments the IP packets from the Service Module and prepares the fragments for transmission by 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.

Complementing 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. In CC2530 SoC, the maximum payload each message can carry is 103 bytes. Given the limited storage and computation capability on ZigBee, UART byte corruption is likely to occur due to buffer overflow or UART interrupt not being handled promptly. We add a CRC field in all the UART messages sent from the host device to ZigBee. The ZigBee receiver always checks the CRC and sends an ACK back on a successful delivery. Otherwise, a timer at the host will trigger a retransmission. Since the host generally has much bigger UART buffer, and faster CPU, we do not apply the CRC field in the reverse direction in order to alleviate the extra computation overhead on ZigBee.

4.1.3 *ZigBee Modem*

The ZigBee Modem provides the host with read and write operations on the ZigBee link. As the UART bitstream arrives at ZigBee, ZigBee translates the bits into frame. Upon successful CRC verification, ZigBee sends ACK frame back to the host. The new frame is buffered in the egress buffer, and will be wirelessly transferred 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 the ZigBee chipset, we have also implemented a flow control mechanisms for the UART RX to avoid the egress buffer overrun. As the egress buffer size crosses a threshold, ZigBee sends a RNR (Receive Not Ready) or RR (Receive Ready) message to the host to request the host to pause or resume sending. Since the host, for example a mobile phone, has a larger UART buffer and a faster CPU, we assume that the flow control in the other direction is not necessary. As we implemented the WiZi-Cloud prototype, we learned that it is critical to fully explore the link capacity of both the UART and ZigBee radio in order to get a good overall system throughput. Therefore, we also designed and implemented a windowing logic on the UART to pipeline the data flow and make use of a DMA transfer.

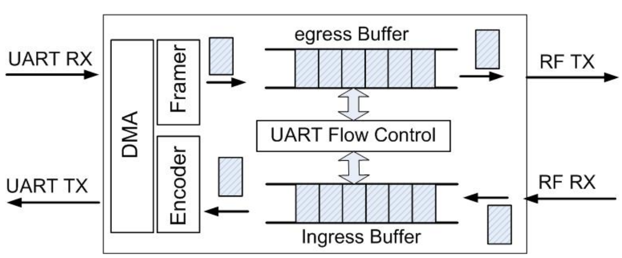


Fig. 6: ZigBee Modem Logic.

4.2 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.

4.2.1 Virtual Interface at the Client

In order to make the physical interface switching transparent to the rest of the system, the WiZi-Cloud Service Module at the client end creates a virtual interface, which is assigned 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 the WiFi or ZigBee interface will be reassembled to IP packets, propagated to the service module and re-injected to the kernel network stack as a raw IP packet. Having all the traffic propagated through the virtual NIC makes the underlying interface changes transparent to the applications. Besides, we can have finer granularity of traffic monitoring and can determine which interface to use at certain moment.

4.2.2 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 the 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 choose a different solution when we design 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. 7, 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.

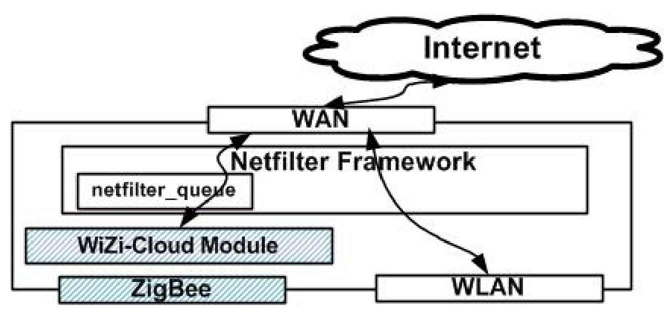


Fig. 7: WiZi-Cloud Service Module at AP

5 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.

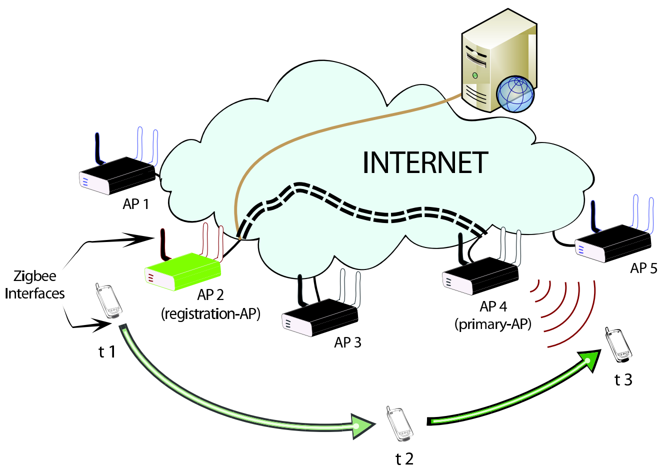


Fig. 8: Dual radio mobile device moving across the WiZi-Cloud system.

5.1 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.

5.2 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 the nearby APs. The mobile device also keeps track of the set of the APs that cover his current location. This set is called the *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. 9 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.

5.3 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 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. 9 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.

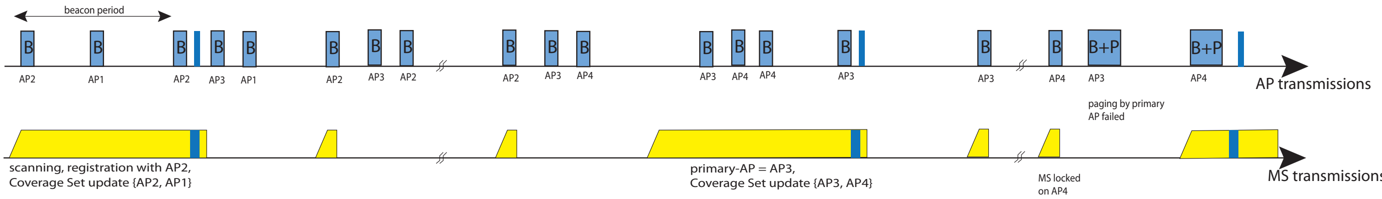


Fig. 9: Wakeup pattern and messages during mobility of mobile device according to Fig. 8.

5.4 Handover

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

5.4.1 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.

5.4.2 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.

5.5 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.

6 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 coverage.

6.1 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 1. 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.[2] The result shows that ZigBee in idle mode achieves more than a 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. Also, it is worth mentioning that although screen is another major energy drainer, it is rarely used in applications such as VoIP, audio streaming, etc. Thus, improving the energy efficiency on network interfaces becomes most critical.

	GSM	WiFi	Bluetooth	ZigBee	OS	Screen
RF Idle	19.1	29.4	7.3	2.6	3.5	378.1
RF Active	1170.7	1648.2	340.3	94.5		

TABLE 1: Breakdown Energy Consumption on Android Phones in mW.

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 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 2). Applications

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

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 2: Mobile Application Categories

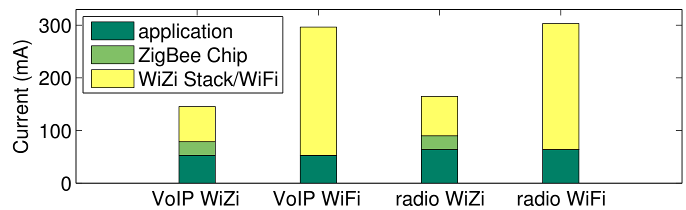
6.1.1 *High Delay Sensitivity, Moderate Traffic Load*

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 flows with the same packet size and interval as we observe from sipdroid traffic. The traffic pattern, plus the goodput and jitter reported by iperf are listed in Table 3.

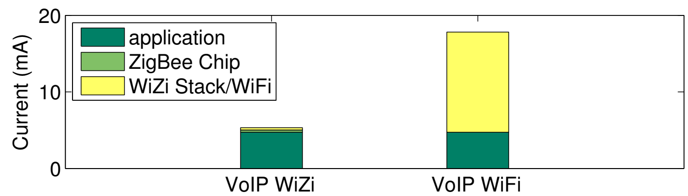
codec	pkts/sec (two way)	UDP pkt size (B)	BW (Kbps)	jitter (ms)
GSM	95	53	39.3	4.38
Speex	97	49	37.1	3.86

TABLE 3: VoIP Client Traffic Pattern.

To further verify the suitability of the WiZi system for delay sensitive applications, we tested a popular Internet Radio application called *iheartradio*, which runs over 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. 10(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-Cloud system still reduces the overall system energy consumption by 50%. As shown in Fig. 10(b), when sipdroid is in standby mode, the WiZi-Cloud system shows an 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 3 times. We believe that further energy efficiency can be achieved by integrating the WiZi-Cloud system as a kernel module, which minimizes the kernel-user space context switching and computation overhead.



(a) sipdroid & iheartradio, active mode



(b) sipdroid, standby mode

Fig. 10: Energy consumption of sipdroid and iheartradio on G1, with WiZi or WiFi, screen off.

6.1.2 *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 4 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	8.08	7.04	7.60	35.75	2.01	1.03
Check	7.59	8.24	26.01	42.18	0.89	1.43
Download	14.40	10.73	28.42	36.17	1.68	1.59

TABLE 4: Email Application Profile, Screen Off.

6.1.3 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 5, 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	199.61	239.8	196.25
WiFi	294.73	93.41	112.88

TABLE 5: Overall System Energy Consumption of Web Browser.

6.2 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.

6.2.1 UDP Throughput

We first measure the UDP throughput for different UDP payload size. Fig. 11 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).

6.2.2 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

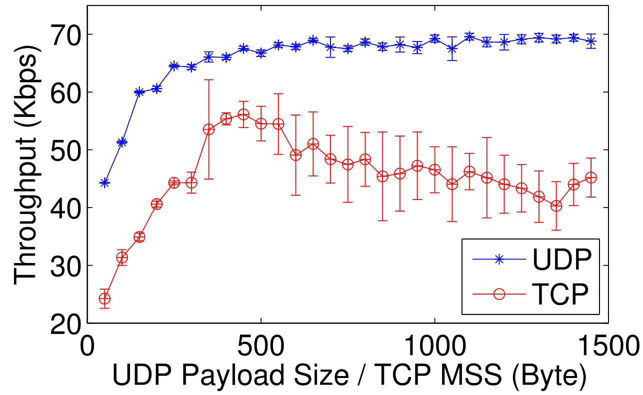


Fig. 11: WiZi TCP / UDP Troughput vs. TCP MSS . UDP Payload Size.

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. 11, the optimal TCP MSS is 450 Byte, achieving 60.2Kbps throughput.

6.3 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.

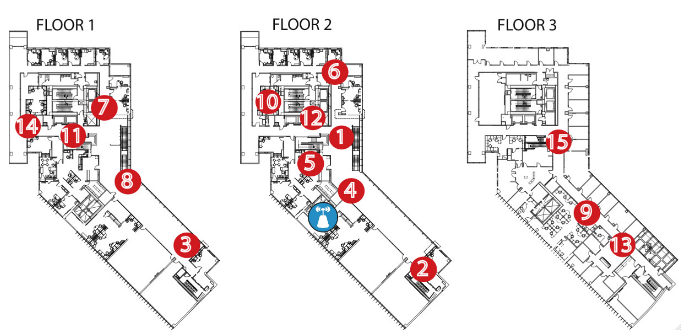


Fig. 12: College's building floor plan with location of measurements points.

We carried out the experiments in our College facility, a three-floor building (shown in Fig. 12). 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. 13, 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 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. 13. The "good" ZigBee coverage is extended to around 100ft, which can cover almost the entire building.

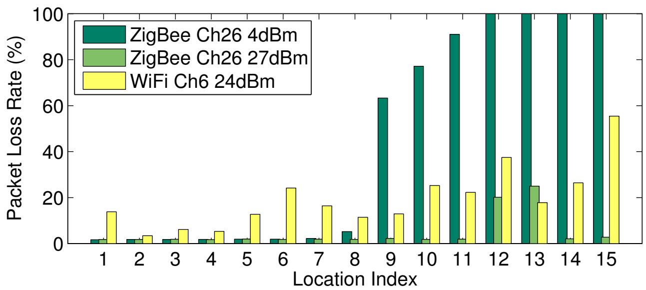


Fig. 13: Packet loss rate of ZigBee on channel 26 at 4dBm and 27dBm, vs. WiFi on channel 6 at 24dBm.

7 More In-Depth Discussion on ZigBee Coverage Performance

In this subsection, we conduct a more comprehensive study of the coverage of the ZigBee radio. It is motivated by two concerns. First, the deployment of WiZi-Cloud could be incremental, i.e., there are only a small portion of WiZi-Cloud APs in the WLAN. Thus, the effective density of WiZi-Cloud APs is lower than WiFi APs which consequently requires larger coverage of ZigBee radios to maintain the connectivity. Second, when a mobile user is on an active communication session over ZigBee radios, e.g., making a VoIP call, a larger coverage can avoid inter-AP handoffs whose overheads may interrupt the ongoing sessions.

As discussed in Section 6, ZigBee shows lower packet loss rate than WiFi within a certain range due to the higher E_b/N_0 of ZigBee transmission. However, beyond the "grace" distance, ZigBee packet loss rate starts increasing drastically. It has been well known that TCP performs poorly while experiencing serious non-congestive packet loss [36], thus high packet loss rate on a lossy wireless link may result in very poor throughput performance in the application layer. Even though with a signal booster on the WiZi-Cloud AP, we can extend the ZigBee coverage to enhance the paging functionality, still this makes the link asymmetric and to some extent limits the applicability of WiZi-Cloud .

In this section, we explore the following two commonly used strategies, *retransmission* and *spatial diversity*, to extend the coverage by mitigating the packet loss, and show their potential gains. Besides, for each of the two strategies, by studying the packet error pattern, we empirically show that the potential gain of applying forward error correction mechanism is negligible.

1) Retransmission: In this scheme, a sender simply sends each packet twice consecutively, back to back. A packet is lost only when both of two trials fail. Intuitively, this scheme sacrifices the possible best throughput, i.e., when the link quality is good, the throughput is half of that in regular protocol. However, the redundancy in this scheme may improve the package loss rate over lossy links.
2) Spatial Diversity: In this scheme, we deploy dual ZigBee antennas at the receiver side. A transmission succeeds if one of the antenna correctly receives the packet. This scheme tries to harness the spatial diversity caused by the physical distance between the two antennas and further reduce the packet loss rate.

7.1 Experiment Setup

We adopted a similar setup as in Section 6. One ZigBee node, as marked in Fig. 14, broadcasts one packet every 20 ms, on channel 26, with 4 dBm transmit power. We disable the CCA functionality on the sender, so that the sender always broadcasts packets at the specified rate. Each packet has 100-Byte data payload, which is a fixed byte sequence, 8-bit integers from 1 to 100. Thus, we know exactly the content the receivers are expected to receive, as well as how many packets that the sender is supposed to broadcast within a certain time window. The mobile ZigBee receivers are connected to a netbook's USB interfaces. We carried out the measurement at 20 different locations in the 3-floor college building. Each measurement last 10 minutes. The receiver node reports all the received packets, including both integral and corrupted ones, to the netbook through UART link. The UART reader running on the netbook timestamps each report with the Linux system time. The report message is of the following format,

Timestamp	CRC_OK	RSSI	Data Payload

- *Timestamp*: timestamp of receiving this message
- *CRC_OK*: it indicates if the Data Payload is integral or corrupted
- *RSSI*: 8-bit signed integer read from the ZigBee receiver RF module
- *Data Payload*: 100-Byte raw data payload received by the receiver. This may be a corrupted data payload.

With such log, we can easily measure the packet loss rate. For all the packet loss, we differentiate corrupted packets and unheard packets. For the corrupted ones, we can identify all the corrupted bits and analyze the error pattern. Then we can analytically estimate the potential gain of applying some forward error correction mechanism (FEC). For the unheard packets, we cannot

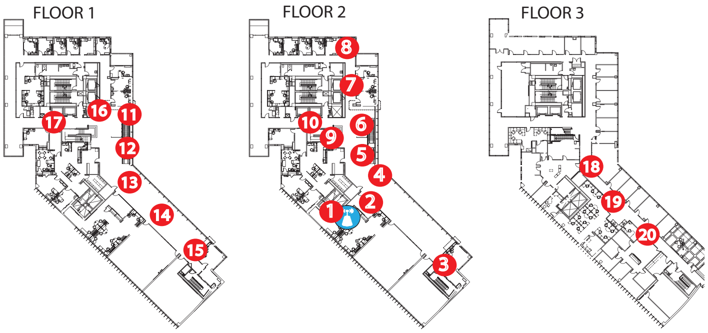


Fig. 14: College's building floor plan with 20 measurement locations.

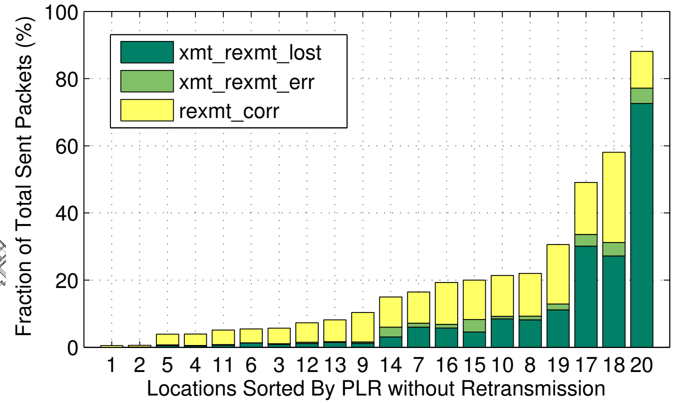


Fig. 15: Packet Loss Breakdown. Single RF with Retransmission.

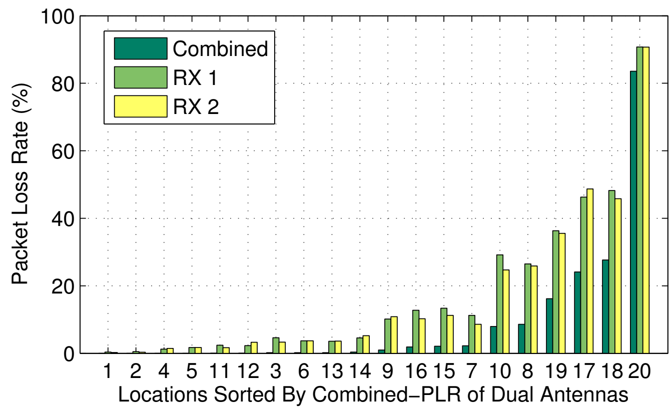


Fig. 16: Packet Loss Rate Measurement with Dual-Radio ZigBee Receiver. PLR on each antenna, and the combined PLR.

do anything with them. We will explain in more details in the rest of the subsection.

Lesson: When we carried out the experiments in the college building environment, we observed extremely inconsistent packet reception performance even at the same location. We believe this is related to the multi-path reflection and interference from WiFi signals. To make our result consistent and reproducible, at each location we moved around within a 3x3 ft area, which artificially introduced more spatial diversity, whereas the overall packet reception performance should be representative for the specific location. We carried out the experiments multiple times during different time of day and week, and we got very consistent result.

7.2 Retransmission & Forward Error Correction

First, we explored the potential gain of reducing PLR by simply retransmitting the packets. In this experiment, the sender broadcasts the same packet twice consecutively, back to back. We call these two copies, copy 1 and copy 2. The receiver logs all the received packets as described in the above section. With the receiver's log, we can classify all the possible receptions into the following five cases:

1) xmt_corr: Copy 1 is integral. The first transmission already delivers the packet successfully, so the receiver can ignore copy 2.
2) rexmt_corr: Copy 1 is corrupted, copy 2 is integral. In this case, the retransmission can deliver the packet successfully.
3) xmt_rexmt_err: Copy 1 and copy 2 both are received, but corrupted. This case serves as the upper bound of the gains of FEC.
4) xmt_rexmt_lost: At least one of the two copies are unheard. We suppose this is not error correctable, because we cannot obtain enough information to carry out forward error correction.

The cases 2 to 4 represent the total packet loss without retransmission. Fig. 15 shows the breakdown of total packet loss by cases. Case 2, rexmt_corr, accounts for 12% to 100% of all packet loss, depending on the locations. Case 2 is inversely proportional to the PLR, and happens more and more rarely as PLR increases. To the contrast, case 4, xmt_rexmt_lost, becomes the predominate type of packet loss as PLR increases. Case 3, xmt_rexmt_err, implies the upper bound of the gain we can achieve with any FEC mechanism. However, our analysis shows that case 3 accounts for only up to 4% of the total packet loss. Thus, we conclude that the achievable gain with any FEC is negligible. Simple retransmission mechanism is sufficient to handle most recoverable packet loss.

7.3 Spatial Diversity & Forward Error Correction

Next, we evaluation the spatial diversity scheme. In this experiment, the sender broadcasts one packet every 20 ms, and the receiver logs the packets received from both antennas. Fig. 16 shows the PLR of each of the two antennas, respectively, as well as the combined PLR. By combined PLR we mean that only when the packet is lost on both antennas, the receiver take it as a packet loss. Our measurement shows that the packet loss on the two antennas shows independence. With dual radios, by simply combining the packets received on each radio, we can reduce the PLR by at least 50% at 19 out of 20 experiment locations. Only one location with very bad reception shows limited improvement.

To study the potential gain if any forward error correction mechanism were applied, we study the packet

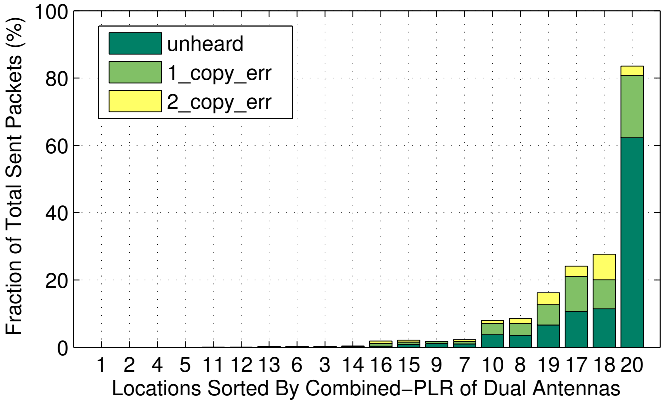


Fig. 17: Packet Loss Breakdown. Receiver with Dual Antennas.

error pattern in this setup. We classify all the possible receptions into the following 5 cases:

1) 2_copy_corr: both antennas receive correct packets
2) 1_copy_corr: only one antenna receives correct packet
3) 2_copy_err: both antennas receive corrupted packets. This case serves as the upper bound of the gain that can be achieved with FEC.
4) 1_copy_err: one antenna receives corrupted packet, the other does not hear this packet
5) unheard: neither of the antennas hear the packet.

The case 3 to 5 represents the total packet loss in the dual-antenna scenario. Fig. 17 shows the breakdown of cases 3 to 5. Our results show that at all locations, case 3 accounts for very small fraction of the total sent packets. The other two cases are the predominant types of the packet loss in all cases. This means that applying FEC mechanism achieves negligible performance improvement. In conclusion, with dual ZigBee receivers, we can reduce PLR by at least 50% at most experiment locations.

A natural question to ask is whether dual-receivers result in higher energy cost. Please note that the dual-receiver is equipped at AP side only, so that the uplink PLR from client to AP can be reduced. As for downlink, we can use a signal booster at AP to extend the coverage, and AP is not energy constrained.

8 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. 10. By integrating the stack to the kernel module, we expect to further increase the energy efficiency.

9 Conclusion

We propose WiZi-Cloud, a dual ZigBee-WiFi network architecture, a set of mechanisms, and a complete suite of HW/SW solution to achieve an energy efficient, ubiquitous and real time network connectivity that is transparent to applications. To the best of our knowledge, our work is the first system design and prototype that integrates ZigBee into commodity mobile phones and WiFi APs for seamless and efficient Internet access. With an extensive set of experiments, we thoroughly evaluate our system in energy efficiency, system throughput and coverage to show the potential of WiZi-Cloud. With detailed empirical results, we demonstrate the advantage and limitations of WiZi-Cloud system in a variety of scenarios. Our experimental results demonstrate that ZigBee significantly improves the energy efficiency in maintaining connectivity compared with WiFi and GSM. By testing with real audio streaming and VoIP mobile applications on Android G1, the total system energy efficiency can be improved by 2 times and 3 times, in active transmission mode and standby mode respectively. Besides, with standard ZigBee network stack and transceiver module, WiZi-Cloud has better coverage than WiFi within 50ft indoor environment. To have a thorough understanding of the actual potential of ZigBee coverage, we explore two mechanisms, retranmission and spatial diversity. With a comprehensive set of experiments, we show that retransmission and dual-antenna can achieve up to 50% gain to reduce PLR. With an in-depth study of bit error pattern, we conclude that the potential gain achievable with any Forward Error Correction mechanism is negligible.

Acknowledgments

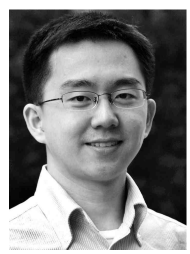
We would like to thank Don Straney, Arash Kakhki, and Hooman Javaheri for their help with the prototyping of our WiZi-kit PCB board. We would like to thank Dr. Bo Sheng for the initial investigation of Mean Opinion Score to evaluate the VoIP performance using WiZi-Cloud system [37].

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] TazTag, "First secure Android mobile devices with NFC & Zigbee technologies." [Online]. Available: http://www.taztag.com/
[10] Spectec Computer Co., "microSD ZigBee card." [Online]. Available: http://www.spectec.com.tw/sdz537.htm
[11] 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
[12] "OpenWrt Hardware List." [Online]. Available: http://wiki.openwrt.org/oldwiki/openwrtdocs/hardware
[13] 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.
[14] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access Using Bounded Slowdown," in *MOBICOM 2002*.
[15] J. Liu and L. Zhong, "Micro Power Management of Active 802.11 Interfaces," in *MobiSys*, 2008.
[16] D. Qiao and K. Shin, "Smart Power-Saving Mode for IEEE 802.11 Wireless LANs," in *INFOCOM 2005*, March 2005.
[17] 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.
[18] G. Ananthanarayanan and I. Stoica, "Blue-Fi: Enhancing Wi-Fi Performance Using Bluetooth Signals," in *MobiSys 2009*.
[19] 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.
[20] R. Friedman, A. Kogan, and Y. Krivolapov, "On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones," *IEEE Transactions on Mobile Computing*, 2012.
[21] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma, "Zifi: Wireless lan discovery via zigbee interference signatures," *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, 2010.
[22] K. Chebrolu and A. Dhekne, "Esense: communication through energy sensing," in *MobiCom '09: Proceedings of the 15th annual international conference on Mobile computing and networking*, 2009.
[23] H. Qin, Y. Wang, and W. Zhang, "Zigbee-assisted wifi transmission for multi-interface mobile devices," *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, 2012.
[24] H. Qin and W. Zhang, "ZigBee-Assisted Power Saving Management for Mobile Devices," *Proc. of MASS*, 2012.
[25] S. Jin and D. Qiao, "Numerical Analysis of the Power Saving in 3GPP LTE Advanced Wireless Networks," *IEEE Transactions on Vehicular Technology*, 2012.
[26] 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.
[27] S. Shin and H. Schulzrinne, "Experimental Measurement of the Capacity for VoIP Traffic in IEEE 802.11 WLANs," *IEEE Transaction on Mobile Computing*, 2009.
[28] F. Guo and T. cker Chiueh, "Software TDMA for VoIP Applications Over IEEE802.11 Wireless LAN," in *INFOCOM*, 2007, pp. 2366–2370.
[29] 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*.
[30] R. Hsieh, Z. G. Zhou, and A. Seneviratne, "S-MIP: A Seamless Handoff Architecture for Mobile IP," in *Proceedings of INFOCOM*, 2003.
[31] R. Hsieh and A. Seneviratne, "A Comparison of Mechanisms for Improving Mobile IP Handoff Latency for End-to-End TCP," in *MobiCom '03*.
[32] I. Ramani and S. Savage, "SyncScan: Practical Fast Handoff for 802.11," in *Proceedings of IEEE Infocom*, 2005.
[33] "RFC 3344 - IP Mobility Support for IPv4."
[34] "RFC 3024 - Reverse Tunneling for Mobile IP."
[35] U. Jönsson, F. Alriksson, T. Larsson, P. Johansson, and G. Q. Maguire, Jr., "MIPMANET: Mobile IP for Mobile Ad Hoc Networks," in *MobiHoc '00*.
[36] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *Networking, IEEE/ACM Transactions on*, vol. 5, no. 6, pp. 756–769, 1997.
[37] T. Jin, G. Noubir, and B. Sheng, "WiZi-Cloud: Application-Transparent Dual ZigBee-WiFi Radios for Low Power Internet Access," *Proceedings IEEE INFOCOM*, 2011.



Tao Jin received his B.S. degree in Computer Science from Peking University in 2005. He is currently a Ph.D. student in the College of Computer and Information Science at Northeastern University. He worked at Nokia Research Lab, Palo Alto as Research Intern in 2007 and 2008. He worked at HP Labs, Palo Alto as Research Intern in 2012. His present research interests include mobile and ubiquitous computing, wireless networks, and distributed systems.

Prof. Guevara Noubir 's research covers both theoretical and practical aspects of secure and robust wireless communication systems. He holds a Ph.D. in Computer Science from the Swiss Federal Institute of Technology in Lausanne (1996). He is a Professor of Computer Science at Northeastern University since 2001. He was a senior research scientist at CSEM SA (Switzerland) between 1997 and 2000 where he led several research projects and contributed to the definition of the third generation Universal Mobile Telecommunication System (UMTS). He is a recipient of the NSF CAREER Award. Dr. Noubir held visiting positions at Eurecom, MIT, and UNL. He is a Senior Member of the IEEE, and a member of the ACM.