2013

Implementing ZigBee-assisted power saving management for short-delay traffics

Lisen Peng
Iowa State University

Implementing ZigBee-assisted power saving management for short-delay traffics

by

Lisen Peng

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:

Wensheng Zhang, Major Professor

Carl K. Chang

Ying Cai

Iowa State University

Ames, Iowa

2013

DEDICATION

I would like to dedicate this thesis to my wife Yue without whose support I would not have finished this work. I would also like to thank my friends and family for their assistance during the writing of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my thanks to those who helped me with conducting research and completing this thesis. Firstly and most importantly, Professor Wensheng Zhang, has guided me with patience and support throughout this research and writing of this thesis. I would also like to thank my committee members, Professor Ying Cai and Professor Carl K. Chang, for their suggestions and contributions to this work.

ABSTRACT

Wi-Fi transmission can consume much energy even when it has no data packet to receive or transmit. Standard power saving mode (PSM) has been used to save energy but under PSM a station cannot achieve satisfactory performance when traffic pattern changes frequently. Now that more and more mobile devices have been equipped with multiple wireless interfaces, such as Wi-Fi, Bluetooth and ZigBee, we proposed and implemented a ZigBee-assisted power saving management (ZPSM) scheme, which wakes up Wi-Fi interface on-demand to increase energy efficiency and reduce delay time. Experiments have shown that ZPSM can achieve both energy efficiency and low packet delivery delay for stations, and the scheme is feasible to implement in resource constrained mobile devices.

CHAPTER 1. INTRODUCTION

Mobile devices, such as mobile phones, tablets and laptop, are widely used nowadays. Mobile phones alone have a recorded quantity of over 6 billion in 2013 and will reach 7.3 billion by 2014 according to Wikipedia (2013). With the prevalence of mobile devices, new mobile technologies continue to emerge and bring exciting changes to the computing world. Despite continuous functionality upgrading, power management has also driven the innovation because energy efficiency is important for a mobile device and has substantial influence on user experience. Research has been conducted to save energy consumption on different aspects such as computation, graphic processing, and wireless connection. This thesis work aims to study power management in wireless connection of mobile devices.

Wireless network connection is a fundamental feature of a mobile device. Infrastructure mode is most commonly used for networking mobile devices. With this mode, the network is composed of numerous mobile devices, which are also called stations or nodes hereafter, distributed in a certain area, and an Access Point (AP). All communications are centered at the AP. Specifically, a node can only transmit or receive data through the AP. An instance of such networking technology is Wi-Fi, which refers to a WLAN compliant with the IEEE 802.11 standards. Much effort has gone into the study of Wi-Fi power saving methods, because according to Kravets and Krishnan (1998) Wi-Fi interface consumes much energy and effective Wi-Fi power management can significantly reduce the energy consumption of mobile devices.

A Wi-Fi interface consumes a high level of energy in data transmission and reception, and a considerable amount of energy even in the idle listening state when Wi-Fi radio does not transmit or receive, but stays awake. While the energy consumed for data transmission and reception is inevitable, the major purpose of Wi-Fi power management is to reduce idle listening. The intuitive idea is to turn off Wi-Fi radio as long as possible when it is idle listening,

so that the waste of energy can be minimized. However, since Wi-Fi cannot send or receive data when radio is turned off, the radio needs to be turned on appropriately in order not to lose incoming data, which poses as a challenge.

A standard approach is the IEEE 802.11 Power Saving Management (PSM). With PSM, stations have two different power states: awake and sleep. When a station is in the awake mode, it is fully powered and able to communicate with the AP. When a station is in the sleep mode, it almost shuts down all the Wi-Fi components to save energy. When PSM is enabled, stations can go to sleep when they are idle, and wake up periodically to communicate with the AP. If the AP has data packets for a station during its sleeping time, instead of sending the packets to the station immediately, the AP shall buffer the data packets and transmit them only when the station becomes awake again. The AP periodically announces which stations have buffered packets. All the stations shall wake up periodically to receive the announcements, then either communicate with the AP to retrieve data or go to sleep again.

With PSM, energy consumption of Wi-Fi can be reduced by turning off radio. However, it results in higher latency since stations can only receive data packets once every a certain time interval. Therefore, the PSM has high performance when long delay is tolerable, but may not be applicable to applications that require low packet latency. Web browsers typically do not tolerate long packet transmission delay. If a data packet is not delivered within a designated time interval, the packet has to be dropped. To address such a problem, when a web browser is running, a station has to temporarily switch to the Constantly Awake Mode (CAM) in which the station should keep awake all the time. Under the CAM, a considerable amount of energy is incurred due to long idle listening time.

A number of existing protocols by Agrawal et al. (2010); Pyles et al. (2011); Qiao and Shin (2005); Krashinsky and Balakrishnan (2005); Agrawal et al. (2010); Ananthanarayanan and Stoica (2009); Pering et al. (2006) have been proposed to enhance the performance of the standard PSM. The authors in Krashinsky and Balakrishnan (2005),InfoCom:Smart proposed adaptive PSM schemes which allow stations switch between PSM and CAM based on designated network thresholds or heuristics. The authors in Agrawal et al. (2010) proposed an opportunistic PSM. In order to reduce the congestion at the AP side, the opportunistic

PSM only allows one station to download at one time and other stations shall go to sleep and save energy. However, these schemes highly rely on accurate prediction of client network usage patterns. Hence, it is hard for stations to achieve high energy efficiency when the network traffic pattern changes frequently.

Besides designing PSM schemes, another approach studied widely is to use an alternative network interface to assist Wi-Fi communication. Bluetooth, a low-power technique which has been embedded in many mobile devices, has already been successful in this kind of designs. Particularly, research has been carried out to exploit Bluetooth interfaces to assist Wi-Fi transmission Namboodiri and Gao (2008); Pering et al. (2006). A Blue-Fi system proposed by Ananthanarayanan and Stoica (2009) combines Bluetooth contact-patterns and cell-tower information so that a Wi-Fi interface is switched on only when it is possible to join a Wi-Fi network, avoiding long idle listening time and reducing the number of scans for discovering Wi-Fi networks. CoolSpots Pering et al. (2006) was proposed to improve energy efficiency by switching between radio interfaces with different features and ranges. Particularly, the authors proposed a set of policies which significantly reduce energy consumption at the cost of minimal changes to an existing infrastructure.

ZigBee is another low-power technology, with a transmission distance ranging from 10 meters to 100 meters. Because of the low-power feature, ZigBee is most suitable the networks that require low data rate and long battery life. ZigBee interfaces have been embedded in more and more mobile devices. In 2012, the first embedded ZigBee interface has been equipped in an Android phone according to TazTag TPH-One Rostli (2012).

In this thesis work, we propose a new and simple Wi-Fi power saving management, ZigBee-assisted Power Saving Management (ZPSM), to achieve high energy efficiency and low latency. In our proposed system, we use the low-power ZigBee radio to dynamically wake up the high-power Wi-Fi radio for transmitting data packets between the AP and clients Qin and Zhang (2012). In ZPSM, both stations and the AP have ZigBee interfaces, which are constantly awake. The AP broadcasts announcements through its ZigBee interface to inform which stations have incoming data packets. Once the ZigBee interface at a station knows the existence of buffered packets from an announcement, it wakes up the Wi-Fi radio immediately to retrieve the packets.

Compared to standard PSM, ZigBee devices send message much more frequently so that latency is significantly reduced. In addition, our proposed system is easy to implement and compatible with the standard PSM.

A prototype system of the design has been implemented and deployed on real mobile devices. Experiments based on the prototype have also been conducted. The results show that our proposed system can significantly reduce latency compared to the standard PSM, and achieve higher energy efficiency. The high performance of ZPSM is limited by ZigBees range, which is smaller than Wi-Fi range. When stations are in the APs ZigBee range, ZPSM has high performance. Once a station moves out of the AP's ZigBee range, the performance is the same as standard PSM. In addition, our experiments also indicate that the performance of ZPSM largely depends on the quality of ZigBee channel. When the quality of ZigBee channel decreases, the energy consumption increases.

The rest of the thesis is organized as follows: Chapter 2 presents related works. Chapter 3 will cover preliminaries. Chapter 4 describes design of ZPSM. In Chapter 5, we will talk about implementation of ZPSM. We will show the experiments and results in Chapter 6. Chapter 7 concludes this thesis.

CHAPTER 2. RELATED WORK

This section will first introduce how standard PSM works in an infrastructure network. Then designs related to our research are introduced in detail.

2.1 Standard Power Saving Management for Wireless LANs

With PSM, a station has two different power states. The first one is awake mode, under which a station is fully powered and able to transmit or receive packets immediately. The other one is doze mode, under which a station almost shuts down all the components so that it is not able to transmit or receive packets, and only consumes little energy.

The AP periodically sends beacon frames to all the stations in the network every beacon interval (BI) to announce which stations have buffered packets at the AP. A station under PSM periodically wakes up every listen interval (LI) to receive beacon frames. A power saving station can only retrieve packets from AP after receiving a beacon frame. However, if it has a data packet to send to the AP, it is able to wake up at any time and transmit the packet.

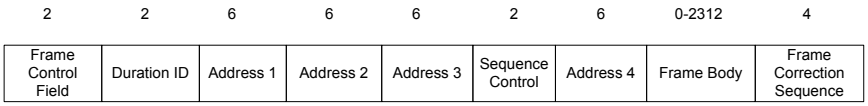

Figure 2.1 Frame Format

General frame format is shown in Fig 2.1, which consists of frame control field, duration ID, address filed, frame body and frame correction sequence. Power management field, a field within the frame control field, is 1 bit length and maintains information that whether PSM is turned on. A value of 1 in this field indicates that the station is under the PSM and a value of

0 indicates that the station is under the CAM. When PSM is enabled, a station shall initiate a frame exchange with the AP and use the power management field to inform that it will switch mode. After successful completion of frame exchange sequence the value in power management field remains constant. Meanwhile, when the AP knows that a station changes to PSM, it will buffer incoming data packets for the station, and only transmit them at designated time points. The stations which have buffered frames at the AP are identified in the traffic indication map (TIM), which is included in each beacon frame.

After a station receives a beacon frame and checks the TIM, if the station finds out there are incoming data packets buffered at the AP, it shall send power saving polling (PS-POLL) frame to the AP and stays awake in order to receive data, otherwise it sleeps again. Once the AP receives a PS-POLL frame, it either immediately transmits buffered data packets to the corresponding station, or acknowledges the PS-POLL and inform the station that it shall transmit the data packets later so that the station will stay awake and wait for the packet. A station will receive all the buffered data before it goes to sleep.

In addition, for a station to receive multicast frames, it has to wake up at every *delivery traffic indication map* (DTIM), which is consisted of multiple TIM intervals. At every DTIM, no polling is needed and a station shall stay awake to receive broadcast and multicast data packets after it receives a beacon frame with delivery traffic indication message.

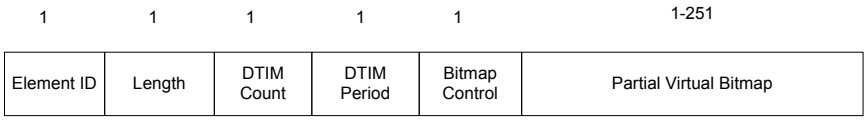

Figure 2.2 TIM Format

Fig 2.2 shows the format of TIM, which consists of DTIM count, DTIM period, bitmap control and partial virtual bitmap. The DTIM count and DTIM period fields indicate the frequency of DTIM and when the next DTIM occurs. During the association, each station is assigned with an association ID (AID) by the AP. AID 0 is reserved for broadcast or multicast. The Bit 0 of the bitmap control field contains the traffic indicator bit with AID 0. This bit is

set to 1 in TIM right before the next DTIM when one or more broadcast or multicast frames are buffered at the AP.Partial virtual bitmap field contains the information that which traffic is pending and buffered in the AP. When there is an incoming packet buffered at AP, the station's AID will be added to partial virtual bitmap by setting the corresponding AID bit to 1.

2.2 Bounded Slowdown (BSD)

Minimizing energy for wireless web access with bounded slowdown (BSD) has been proposed by Krashinsky and Balakrishnan (2005). While web-related applications typically use TCP layer to transport data, standard PSM can cause performance issues for TCP protocol due to increased round trip times (RTTs), which dominates the overall transfer time and accounts for most of web browsing performance. Under standard PSM, a station can wake up at any time to send a data packet containing a web request, then go to sleep immediately after the packet has been transmitted to the AP. However, the response data, or acknowledge packet, will be buffered at the AP because the station has already been in sleep mode. The RTT in this situation depends on the interval between the time when the station sends the request and the next beacon frame. As a result, RTT is more significant when the request is sent at a earlier time point during a beacon period. For example, with an actual RTT of 20ms and a beacon period of 100ms, if the mobile device sends the request immediately after a beacon, the response will be buffered at the AP and received after the next beacon; thus the observed RTT will be 100ms; however if the same station transmits packet 79ms after the a beacon, the RTT will be 20ms.

When TCP is run under PSM, due to the RTT slowdown caused by PSM, the performance of TCP protocol degrades. RTT dominates the performance of TCP because the fact that TCP connections are typically very small. Meanwhile, the TCP data packets are always delayed and sent to a mobile device right at the beginning of a beacon period and the mobile device transmits acknowledge packet after the beacon. As a result, the TCP connection is synchronized with the beacon period and the observed RTT is rounded to the nearest 100ms. The greatest slowdown happens when actual RTT is significantly less than 100ms.

Although standard PSM protocol can guarantee that RTTs delays are less than one beacon

period, it is not energy efficient enough. To address such a problem, the authors in the paper present an adaptive PSM algorithm BSD which aims to minimize the energy consumption by adding a bounded-slowdown to RTTs so that RTTs will not be increased by more than a given percentage. In BSD, to minimize the energy consumption, each station has a specified parameter p (p is bigger than 0), all the RTTs are bounded in (1 + p) multiplied by RTT. Specifically, at the beginning, a client stays awake for a designated time T(awake) after it sends out data to the AP. Then the client switches to the doze mode for a time period T(sleep) equals to p multiplied by T(awake) before it wakes up to receive buffered packets from the AP. Then the client repeats this process and every time sleeps for the awake period multiplied by p. If the client sends data, the process is reset.

In adaptive PSM, stations do not go to sleep immediately after completing transmission a data packet to the AP but keep awake for a steady period. As a result, the AP can transmit response data to the stations without delay. Meanwhile, the AP could also announce the station if there are incoming data packets in response frame instead of waiting for the next beacon. Another difference is that the station skips some beacons. Although longer LI might result to more packets buffered at the AP, the awake duration after a station transmit data release the buffering load at the AP. In summary, with the BSD protocol, quick response time will not be delayed, while longer ones will be increased by up to a parameterized maximum factor, 1 + p.

A constraint with the BSD is the assumption that all data packets sent from the stations are requests, because the protocol is operating at the data link layer and cannot distinguish request from response. As a result, there might be unnecessary awake time for a station after sending response frame. Instead of using designed parameter p and currently network actively to adapt PSM, ZPSM in our paper is able to wake up Wi-Fi at will since it uses ZigBee device to transmit control massages. Hence, ZPSM has less limitation than BSD, and the performance is also better.

2.3 Opportunistic Power Saving Mode (OPSM)

Opportunistic power saving mode (OPSM) for infrastructure IEEE 802.11 WLAN has been proposed in Agrawal et al. (2010) to solve the long thinking time issue in web browsing. In

OPSM, all the stations are engaged in web browsing. These stations download files and wait for a short of *listening time*, then download files again. The standard PSM performs well under this traffic mode because stations can switch to sleep mode to save power during listening time. However, when the number of stations increases, the performance of standard PSM degrades because of congestion.

In standard PSM, stations send PS-POLL to retrieve data after receiving a beacons frame. In existence of congestion at AP side, the AP may not be able to transmit data to awake stations immediately. The AP will acknowledge PS-POLL and send data to these stations later. During the waiting time, the stations have to keep awake. Since the throughput is shared by all stations, the whole download time could last long so that some stations keep active for a long time, which consumes much energy. To solve this problem, the authors propose an opportunistic PSM to address this problem. The OPSM enables only one station to download at one time. If a station wakes up and finds that any other station is receiving data from the AP, this station will go to sleep again. All stations follow this strategy to avoid idle listening and ensure minimum energy consumption. When multiple stations finish their listening time, the stations should generate a random duration to decide how long they should wait before starting to download. Station that samples the shortest duration will download first and other stations will sleep again.

A problem with OPSM is that the performance highly depends on station network usage pattern. In order to achieve high energy efficiency, the system needs to accurately predict the usage pattern of the stations. Thus, its performance is not as good as ZPSM, in which ZigBee interface can wake up the Wi-Fi radio exactly when there is incoming data packets.

2.4 Blue-Fi

To enhance Wi-Fi performance, Ananthanarayanan and Stoica (2009) proposed *Blue-Fi*, a system combining Bluetooth contact-patterns and cell-tower information to predict the availability of Wi-Fi connection, therefore the Wi-Fi interface is only switched on when there is possible Wi-Fi connectivity.

When there is no Wi-Fi connectivity available, the Wi-Fi interface could stay in a long pe-

riod of idle listening time, scanning for discovery unnecessarily. The idle listening and scanning can waste a lot of energy. The Blue-Fi system is able to achieve energy efficiency because a station can dynamically wake up Wi-Fi only when there is available Wi-Fi connectivity. The key idea of Blue-Fi is using low power Bluetooth and cellular interfaces to predict the Wi-Fi connectivity for high power consuming Wi-Fi interface. In terms of Bluetooth prediction, the Blue-Fi predicates Wi-Fi availability when any of the Bluetooth devices nearby indicates available Wi-Fi connectivity. In addition, the Blue-Fi interface also checks with visible cell-towers to predict Wi-Fi availability. Each mobile device periodically logs all the network signals locally. In the log, Bluetooth devices are identified by MAC address, cell-towers are identified by the tower identifier, and Wi-Fi APs are identified by their SSID/BSSID. Blue-Fi predicts Wi-Fi connectivity based on the log.

The main challenging of the prediction is that the ranges of Bluetooth and cellular are different from that of the Wi-Fi interface. For example, the range of a Bluetooth device is typically much shorter than that of a Wi-Fi device. There is a likelihood that a mobile device within Wi-Fi coverage may fail to predict Wi-Fi availability only because that the closest Bluetooth device is still too far to be detected. On the other hand, the range of cellular interface is much larger than Wi-Fi interface, which might lead to false positive when the Blue-Fi system uses cellular interface to predict Wi-Fi availability. In the paper, the authors proposed a hybrid algorithm based on both Bluetooth and cellular interfaces. In addition, the authors measure a prediction reliability threshold to balance accuracy and coverage. The result shows that Blue-Fi is up to 62% more energy efficient according to Ananthanarayanan and Stoica (2009). Although using Blue-Fi to enhance Wi-Fi performance is different from using ZPSM, the energy efficiency present in the Blue-Fi system is a good proof that low power interface can assist Wi-Fi interface and significantly reduce Wi-Fi energy consumption. In this paper we will present the performance of ZPSM using another low power technique - ZigBee.

2.5 Other Researches

Namboodiri and Gao (2008); Pering et al. (2006) have conducted research to save energy on real-time applications like voice over IP (VoIP), which is widely used in mobile device such

as Apple iPhone. As a real-time application, VoIP call has low tolerance in terms of latency. There is a deadline before which all the packets must be delivered to the stations; packets delayed above this threshold will be dropped. Therefore, energy consumption can be difficult to measure. Meanwhile, high packet rate exacerbate the situation because it is hard for a station find a significant time to switch to the doze mode.

Namboodiri and Gao (2008) have proposed an algorithm GreenCall, providing a trade-off of the quality and energy conserved in the phone call. GreenCall allows users to set the maximum delay time they can tolerate and minimize energy consumption based on users choice by adopting an optimized sleep time. The authors also present how to use CreenCall to solve two different cases: i) Client is trying to save energy through PSM and the peer radio stays in CAM ii) Both client and peer desire to save energy and stay in PSM. In the first case, GreenCall keeps track of historical latencies to calculate future sleep time based on deadline and estimated time to receive packets without PSM. In the second case, the calculated sleep period is equally shared by the two stations.

While above two papers target real-time applications with low tolerance in latency, Qin and Zhang (2012) proposed ZigBee-Assisted Power Saving Management that is also applicable to delay tolerable applications, such as short file transfers, web browsing, video streaming, etc. Specifically, Hua has designed two wakeup strategies with the assistance of ZigBee: one for short delay (delay is equal or smaller than 200ms) applications and long delay (delay is equal or bigger than 200ms) applications. For short delay applications, in order to deliver the data packets within delay bound, the Wi-Fi radio at the station will be immediately turned on once the station receives the wakeup beacon. For long delay applications, in order to achieve higher energy efficiency, when a station receives wakeup beacon, it will wait for the AP to buffer more data packets and wakes up the Wi-Fi radio at a later BI.

This thesis is different from Hua Qins paper. We simplify his on-demand strategy and only focus on how to achieve shortest delay. Besides, Hua Qin has conducted simulation with Mad Wi-Fi and assumes full control of the hardware, while we have implemented the system on real mobile devices rather than simulation.

CHAPTER 3. PRELIMINARIES

3.1 Power Management for Wi-Fi networks

Wi-Fi devices usually support two power management modes: power saving mode (PSM) and constantly awake mode (CAM). With the PSM, a station can switch to doze (sleep) mode to save energy and wake up periodically to receive beacon frames. In the sleep mode, the Wi-Fi radio is turned off and the station cannot receive any data packets. Since data packets for a certain station will be buffered at the AP and sent to the station when it wakes up every listen interval (LI), packet transmission delay could be longer than expected. The increase in delay may not be tolerated by interactive applications such as web browsers Krashinsky and Balakrishnan (2005). Under low delay tolerance situation, a common and simple implementation is to switch the Wi-Fi interface to the CAM in order to minimize delay. The low latency comes at a price - energy consumption is 20 times higher than in the PSM because energy waste for idle listening time is high. Thus, the main challenge in power management for Wi-Fi devices is the trade-off between performance and energy efficiency.

One solution is to design adaptive PSM schemes, which usually require relatively stable communication pattern and accurate prediction of the pattern. An alternative solution is using other wireless interfaces to assist Wi-Fi interface as it has been more and more common to have multiple wireless interfaces embedded in mobile devices. Prior researches which combine Bluetooth and Wi-Fi interface have already shown that a system with different wireless interfaces can work out well. In this thesis, we propose the ZPSM system which combines Wi-Fi interface with ZigBee interface to build a ZigBee-assisted system atop the standard PSM, without any change to the IEEE 802.11 standards.

3.2 Co-existence of Wi-Fi and ZigBee Interfaces

In order to deploy the ZPSM system, a mobile device must be equiped with both Wi-Fi and ZigBee interfaces. Since both of the two wireless interfaces work on the 2.4 GHz frequency band, a major concern is that they may interfere with each other when working simultaneously. According to Qin and Zhang (2012), since ZigBee is a low powered technique, when Wi-Fi and ZigBees channels overlap, Wi-Fi communication can cause severe performance degradation for ZigBee. However, if their working channels do not overlap, both Wi-Fi and ZigBee interfaces work well, where the packet delivery ratio of ZigBee communication is higher than 95% and the Wi-Fi communication is nearly not affected.

3.3 System Model

In ZPSM system, AP and mobile stations are all equipped with a Wi-Fi (IEEE 802.11) and a ZigBee (IEEE 802.15.4) interface. The ZigBee interface at the AP is used to send power saving control massages and the ZigBee interfaces at stations are used to receive these control messages. Wi-Fi interfaces operate according to the standard PSM or the ZigBee-assisted PSM (ZPSM) for data transmission. On the AP side, both Wi-Fi and ZigBee interfaces are always awake, while on the station side, Wi-Fi and ZigBee interfaces wake up periodically to minimize energy consumption.

The standard PSM (SPSM) and the ZigBee-assisted PSM (ZPSM) co-exist in our system. In fact, the communication range of a ZigBee radio is smaller than that of a Wi-Fi radio. If a station is out of the ZigBee range but still in the Wi-Fi range, the station will automatically switch to standard PSM. The standard PSM is also used when the quality of ZigBee channel is below the desirable level so that ZigBee communication fails frequently. ZigBee communication failure might result from two reasons: 1) link quality of ZigBee channel is unreliable or; 2) ZigBee interface is currently used for other purposes. We use p_i to represent the ZigBee channel quality for station i. It is the probability that a ZigBee packet sent from the AP and can successfully arrive at station i. p_i may vary over time based on different network conditions.

Each station i has a delay bound of downlink packet transmission. Here the *delay bound* is

the maximum time a data packet will be buffered at the AP. If a packet cannot be delivered within delay bound, this packet might be dropped. We define $delta_i$ as *delay-meet ratio* for particular station i; that is, it represents that the percentage of packets received with a delay lower than the delay bound. Delay meet ratio $delta_i$ is not fixed but changes over time with different ZigBee channel quality p_i.

In practice, the ZigBee link quality of a mobile device may change when this device is moved around. In order to simulate the movement, in our experiment we assume that the AP is static and the stations have limited mobility. Specifically, each mobile device first stays static for a fixed period of time, and then moves towards a random direction at a random speed ranging from zero to a specified maximum speed. After the station reaches the destination, it stays for a fixed period of time and then moves again. This movement assumption is consistent with the well-known random waypoint model.

3.4 Design Objectives

- *Energy Efficiency*: Our system should be able to minimize both Wi-Fi and ZigBee interfaces energy consumption.
- *Minimum Delay*: Our system should reduce delay as much as possible.
- *Compatibility*: Our system does not need to change IEEE 8021.11; instead, it should be compatible with standard PSM.

3.5 On-demand Wakeup Strategy

In order to successfully transmit and receive data packet, the Wi-Fi radio at a station should be turned on. In an adaptive PSM such as the scheme proposed by Krashinsky and Balakrishnan (2005), if the delay bound is short, the station has to set up a shorter LI or even switches to CAM. In order to achieve higher energy efficiency while achieving the short delay bound, we use a wakeup strategy implemented with the assistance of ZigBee interface installed on the AP and stations. The ZigBee interface on the AP is constantly awake and broadcasts a wakeup beacon frame every *wakeup interval* (WI), which is much shorter than a BI so that

packet delay could be reduced. A wakeup beacon frame also contains indexes to indicate which stations should wake up to retrieve data packets. All stations shall listen to the wakeup beacon every WI through their ZigBee interface. Once a station receives a wakeup beacon with wakeup notice, it should wake up its Wi-Fi interface immediately and sends out PS-POLL to retrieve the packet; otherwise, the Wi-Fi radio stays inactive.

An example is shown in Figure 3.1 from Qin and Zhang (2012) with our on-demand wakeup strategy. When the delay bound is shorter than LI, it is possible that a data packet that arrives between two listening points expires its delay bound before the next beacon frame. However, under our proposed system, the ZigBee interface at the station will receive a wakeup beacon frame and then knows about the buffered data packet. Therefore, the station could wake up the Wi-Fi interface to retrieve data. Under ZPSM, a station has higher possibility to successfully receive a data packet than in PSM, and spends less energy than it does in CAM because energy waste in idle listening time is reduced. Meanwhile, the time in our proposed system is less than WI if the ZigBee channel quality is good. As a result, the ZPSM can achieve high energy efficiency at a low latency.

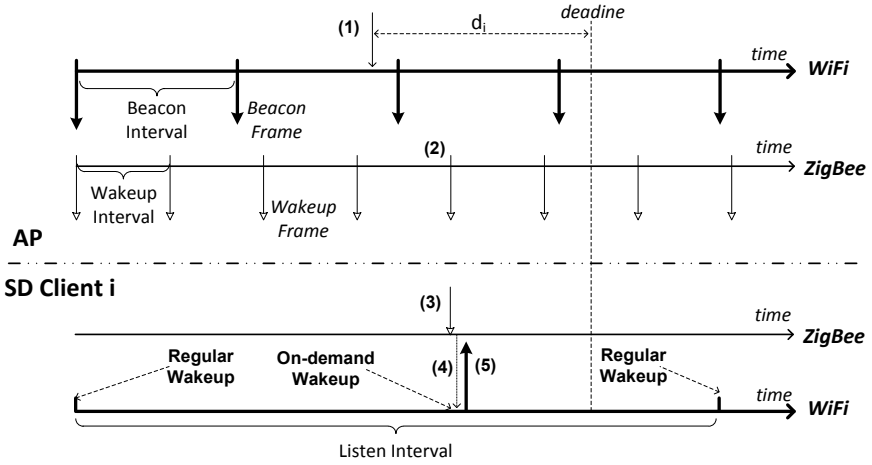

Figure 3.1 Wakeup behaviors

CHAPTER 4. SYSTEM DESIGN

Since the communication range of ZigBee could be smaller than that of Wi-Fi, it is possible that a station is in the Wi-Fi range of the AP but outside of its ZigBee. When the mobile devices move, the ZigBee network is more dynamic than the Wi-Fi network. Therefore we introduce a membership mechanism for ZPSM.

4.1 Membership Management

An AP broadcasts wakeup beacon frames periodically through its ZigBee interface. Once a station is within the ZigBee range of the AP, it can receive these ZigBee frames. If the station finds out that it is not a member of the ZigBee network, the station could send a request through its Wi-Fi interface to the AP to apply for membership. After the AP receives the request, it shall randomly generate a unique ID for the station. The AP shall store the ID in the membership table where all member stations and corresponding IDs are maintained, and send the ID back to the station so that the station can store the ID as well.

The membership table also records the last activity time of each station. When the ZigBee interface at a station wakes up the Wi-Fi interface, the station shall send a control message to the AP through the Wi-Fi interface. The AP knows from the control message that the station is ready for data reception and updates the last activity time of this station. If a station is inactive for a certain period of time, it shall be deprived of membership. Therefore, a station shall send control messages periodically through its Wi-Fi interface as heartbeat if it wants to maintain membership.

If a station is out of the APs ZigBee range and not able to receive wakeup beacon, in order to save energy, it might leave the ZigBee network and shut down the ZigBee interface. In this

case it shall send a message to inform the AP of its leaving. After the AP receives the message, it shall delete the ID of this station from the membership table.

4.2 Communication under ZPSM

When a member station has incoming data packet, the AP shall send a wakeup beacon frame through its ZigBee interface. The format of a wakeup beacon is shown in the Fig 4.1. The first 2 bytes are reserved for the Frame Control field, and the following 4 bytes are reserved for the BSS ID field, containing the identification information for each AP to distinguish from other APs. The rest bits are used for the Index field to indicate which mobile device needs to wake up its Wi-Fi radio to retrieve data packets. The value of 1 in a field indicates that the corresponding mobile device has incoming packet and shall wake up.

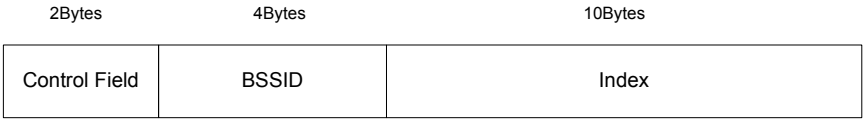

Figure 4.1 Wakeup Beacon Format

If a station can receive wakeup beacon frames through its ZigBee interface within the delay bound, the data packet could be delivered successfully. However, when the delay bound is shorter than a LI and the ZigBee channel quality is low, if a station fails to receive several consecutive wakeup beacon frames, the data packet will be dropped. To address this issue, we introduce extra wakeups in the case of low ZigBee channel quality.

4.2.1 Channel Quality Scheme

We use p to denote ZigBee channel quality, representing the probability that a station can receive a wakeup beacon from the AP. Each node measures its ZigBee channel quality by computing the percentage of wakeup beacons that it can receive during a period of time. The ZigBee link quality is not fixed and therefore the value of p is not constant, but is calculated periodically.

When the ZigBee channel quality is low, the ZigBee-based on-demand wakeup solely may not ensure delay bound be achieved. In such a case, proactive wakeup of Wi-Fi interface become necessary. Hence, we introduce a new concept of dynamic listening interval (DLI) to regulate how frequently the Wi-Fi interfaces should proactively wake up in order not to miss delay bounds in the case of bad ZigBee channel. Specifically, the DLI is computed as follows:

$$DLI = D * \frac{1}{1 - \frac{D}{WI} * p} \tag{4.1}$$

Here D denotes the required delay bound, which is the maximum time that a data packet can be buffered at the AP. We first calculate how many WIs a station has before a data packet is expired. This number multiplied by channel quality p ($=\frac{D}{WI} * p$) is the expected number of ZigBee wakeup beacon frames the station is able to receive during the period of delay bound, while $\frac{D}{DLI}$ is the number of proactive Wi-Fi wake-ups when a station can retrieve data packets. We want to find a DLI that satisfies the following inequation:

$$\frac{D}{DLI} + \frac{D}{DLI} * p \geq 1 \tag{4.2}$$

The left side of the inequation presents the expected number of wakeups when a station can retrieve a data packet within delay bound. When it is equal or larger than 1, we expect that the station can successfully receive a data packet before it is dropped. To avoid unnecessary wake-ups, we let the left side equal to 1. With transformation the equation, we obtain the formula 4.1. It is obvious that $\frac{D}{DLI} * p$ can be larger than 1 so that the DLI is a negative number. In our scheme, negative DLI indicates that no proactive Wi-Fi wake-up is needed. Otherwise when DLI is positive, we shall calculate the change in DLI; if change in DLI is larger than a threshold, the Wi-Fi radio will wake up more frequently to retrieve data.

Fig 4.2 shows how proactive Wi-Fi wakeups take effect in our proposed system. LI is 400ms; WI is 40ms. In (a), three wakeup beacons out of ten are delivered successfully so that measured p is 0.3. According to formula 4.1 DLI is 400ms. We can see that even if the station has missed all the wakeup beacons within the delay bound, if it has one proactive Wi-Fi wakeup when the

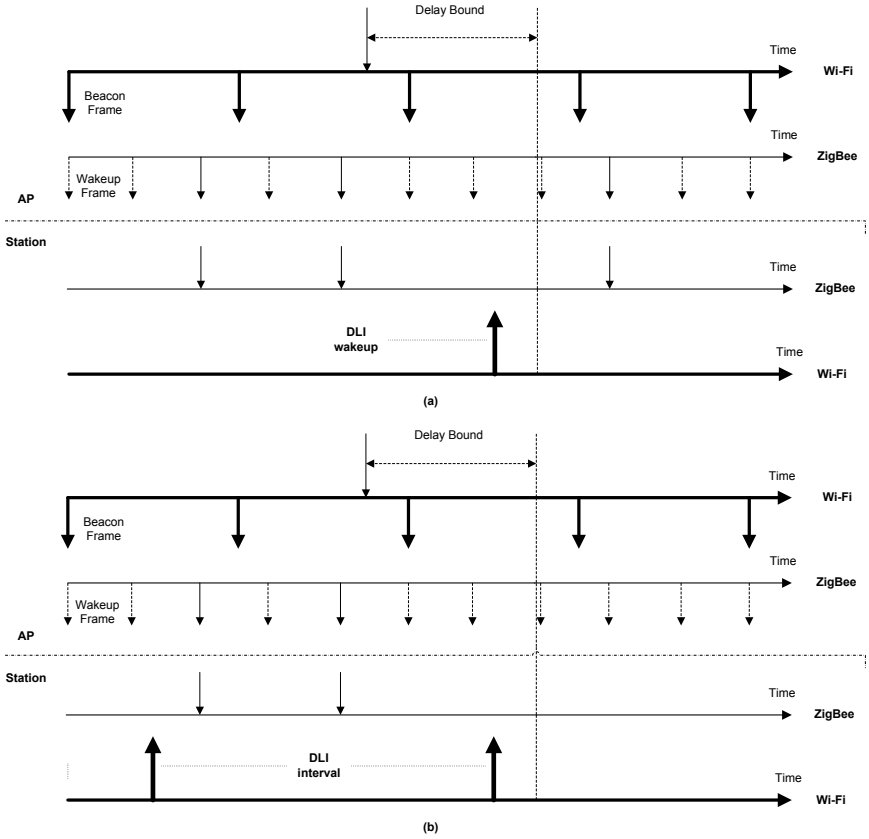

Figure 4.2 The Wakeup Behavior of the AP and station i

data packet is still buffered at the AP, it can successfully retrieve the packet. In (b), p is 0.2 and DLI is 200ms, so that the expected number of proactive wakeups is 0.5, versus 0.25 times in (a).

4.3 Design Summary

As shown in Fig 4.3, the ZPSM system consists of six components. The membership manager (MMgr) component is responsible for membership registration and maintenance. After the MMgr obtains header information for the each data packet, it will check the ID contained in the header with IDs in the membership table. If the ID exists in the table, the MMgr update the last activity time the corresponding station. The MMgr shall periodically de-associate inactive stations by deleting the corresponding ID in the membership table. Since the ZPSM

is built atop the PSM, after the AP handles a data packet and obtain the header, the packet is proceeded and handled as the same as in the PSM.

The wakeup scheduler (WS) component records which stations need to wake up to retrieve data packets. Whenever the AP has incoming data packet for a station, the WS shall be update the wakeup schedule in wakeup beacon frame and pass the new frame to the ZigBee controller (ZC) component at the AP side.

The ZPSM AP controller (ZAC) is the core controller of the AP system. The ZAC can register and cancel other components to turn on or turn off the ZPSM. More importantly, it will periodically configure WI and order the ZC to send the beacon frame every WI.

The ZC on station side receives the wakeup beacon every BI and sends it to the wakeup coordinator (WC) component. WC is responsible for membership application and Wi-Fi radio control. Since ZigBee link quality is not stable, an important component in our system is the ZPSM station controller (ZSC). The ZSC could order the ZC at the station to record the channel quality p. Based on different channel quality levels, ZSC dynamically controls Wi-Fi interface through the wakeup coordinator, so that the Wi-Fi device will either wake up more frequent or switch to the standard PSM when channel quality is low.

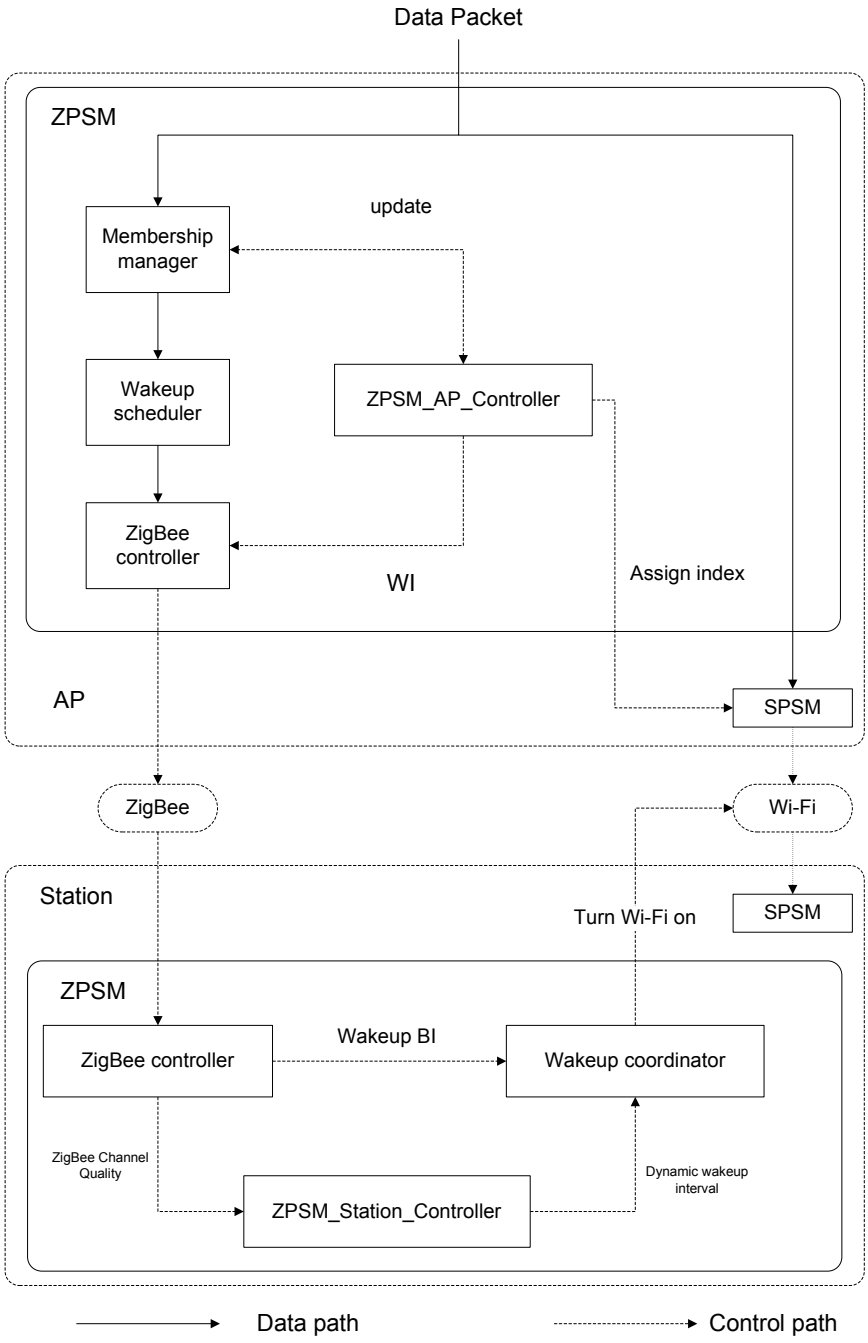

Figure 4.3 The Architecture of ZPSM

CHAPTER 5. IMPLEMENTATION

5.1 Hardware and Operating System

In this section, we will introduce the hardware, operating system and two Linux kernel modules we used to implement our proposed system.

5.1.1 Telos-B Mote

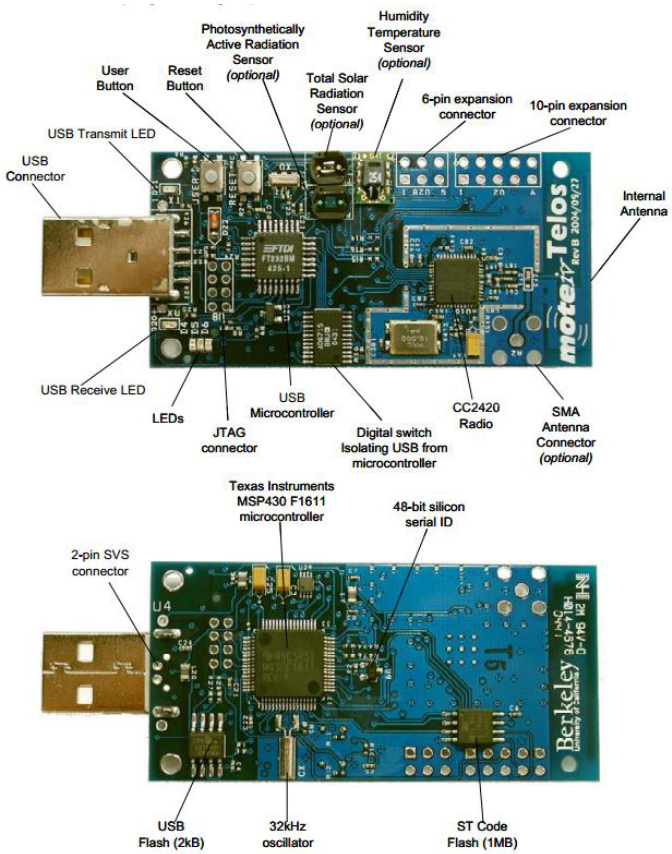

Figure 5.1 TELOS-B

Figure 5.1 is the Crossbows Telos-B mote (TPR2400), an open source platform designed for conducting experiment. The mote has USB programming capability, an IEEE 802.15.4 radio, a low-power MCU, and an optional sensor suite. The IEEE 802.15.4 radio, integrated with antenna, is a ZigBee compliant RF transceiver. The transceiver has the same basic transmission features as defined in IEEE 802.15.4: 2.4 to 2.4835 GHz, 250 kbps data rate. The Texas Instruments MSP430 F1611 microcontroller with 10kB of RAM, 48kB of flash, and 128B of information accounts most for the low power feature of the mote.

5.1.2 TinyOS

We run TinyOS on the Crossbows Telos-B mote. TinyOS (2013) is an open source operating system integrated with toolchain and drivers for microcontrollers. In TinyOS, typically the system space is not protected, which means all components of system space and user space run in a single address space. When we compile an application, there is a single binary image and a mote only runs one TinyOS image at one time.

An AP or a station is developed on a PandaBoard with a TELOS-B mote plugged in. We enable a data packet transmission between TinyOS and Linux through the serial communication by using an application called BaseStation in TinyOS (2012). BaseStation is a basic application included in the TinyOS software development kit. The application connects the ZigBee compliant radio with the serial port so that data packets received from the radio can be passed to the Linux system through a serial port, and vice versa. Since TinyOS has a toolchain, with the help of BaseStation we enable a Linux application to communicate directly with a ZigBee network. To enable Linux to interact with mote, on the Linux side we use another application called SerialForwarder. The SerialForwarder in TinyOS (2012) can open a packet source so that other programs are able to connect to the SerialForwarder and use the source.

5.1.3 PandaBoard

PandaBoard is low-cost open mobile software development platform integrated with WLAN and Bluetooth. Figure 5.2 is the structure of PandaBoard. We develop programs on a SD card and insert the card into the SD/MMC Card Slot so that the Panda Board can load the program

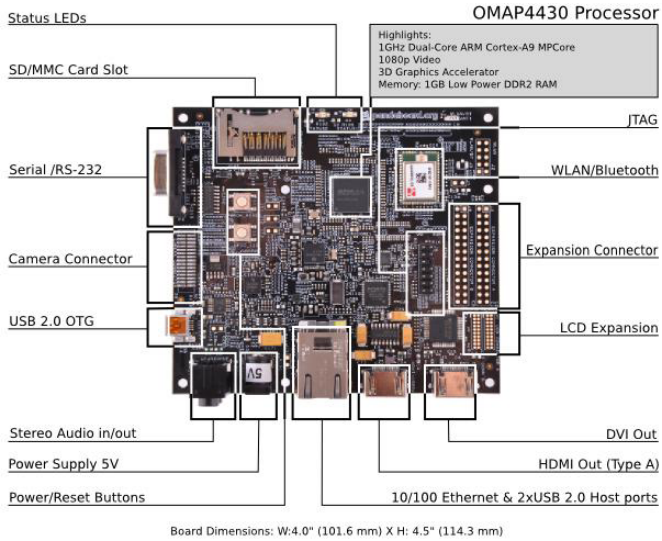

Figure 5.2 Panda Board

from the SD card. The Serial/RS-233 is a serial port compliant with the RS-233 standard and enables data transmission. The Camera Connector supports SCI-2 cameras, while USB cameras and other portable devices can be plugged into the USB 2.0 OTG. The Stereo Audio in/out supports multichannel sound input and output. The Power Supply 5V is the power socket, and the Power/Reset Buttons can switch the board on and off. The High-Definition Multimedia Interface (HDMI) Out is an interface used to connect display and sound device at the same time. And the Digital Visual Interface (DVI) Out is an interface used only to connect with display devices. The LCD Expansion and Expansion Connector are used for LCD touchscreen. The PandaBoard is based on Texas Instruments OMAP4430 processor.

5.1.4 Linux

The basic components of the Linux system are kernel, C library, toolchain, basic system utilities and desktop environment. The kernel is the core of the Linux system and provides basic services. In Linux, the kernel is a monolithic process in a single address space, consisting of some separate modules providing different services, such as network, graphic and sound. A module is an object which integrates related subroutines, data and entry and exit points of a

certain service. In kernel, modules can be dynamically loaded at runtime.

Since the kernel accounts for the basics and essentials of the system, protected memory management units, which is called kernel space, is given to the kernel with full access to hardware. On the other hand, the applications are run on user space. Kernel code is executed in kernel mode, while regular process is executed in user mode. The Figure 5.3 from Love (2010) shows that how applications use the kernel services. Typically, the applications can call functions trough system call interfaces or library to access to kernel services.

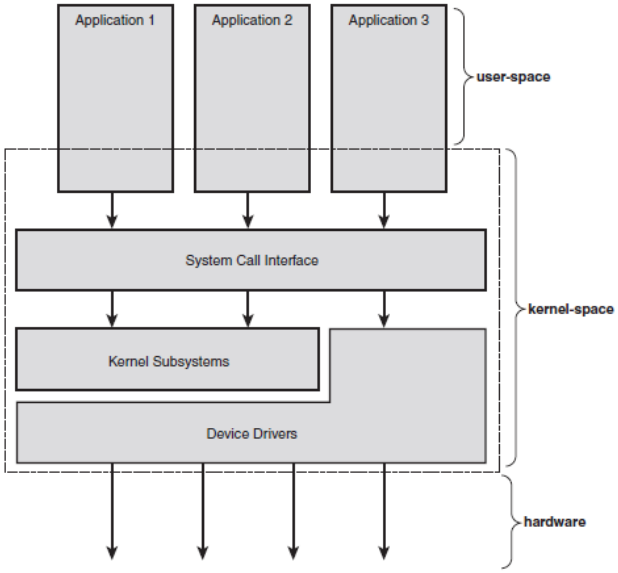

Figure 5.3 Linux System

When hardware needs to communicate with the system, it interrupts the processor to interrupt the kernel. The kernel then runs an interrupt handler to address the request. The interrupt handler runs in a special interrupt context in kernel space separate from any other process. Some device drivers are based on hardware components, other device drivers are virtual.

Devices are divided into three categories: block device, character device and network device. Block devices, such as hard drive and flash, are addressable, while character devices, such as keyboard and mice, are not addressable. Network devices are a special kind of device that can

provide network access via a physical adapter and special protocols. Unlike block devices and character devices, network devices cannot be accessed via a device node but via socket API.

In our Linux part architecture, there are two parts: the kernel part and the user part. Although kernel extension is loadable in runtime with low latency so that a loadable kernel module has high efficiency, we implement the ZPSM using a two-part architecture because of the following three reasons:

1) *Reliability*

The kernel modules have higher privilege than the user space and free access. As a result, a bug can destroy the system. Furthermore, since the kernel is in a single address space, there is no protection for memory overflow and it is possible that a module corrupts with data in other modules. On the other hand, the user space allows multiple programs to run in their own virtual space, while a bug can be terminated without affecting the kernel. To ensure the reliability of our system, we only put the core functions in the kernel extension in kernel space.

2) *Compatibility*

Typically, a kernel modules code needs to be compiled for each version of kernel. The module needs to include a specific header file for a particular version of kernel. Compatibility issue occurs if some interfaces change. Since the user interface does not change as frequent as the kernel interface, it has lower version dependency. For compatibility concern, we keep minimum size of the kernel module so that less update needs to be made when API changes.

3) *Extensibility*

Kernel space development more complicated than user space development. First Linux kernel code has large size and high complexity. Beside, delicate concern should be given in development process to avoid kernel panic. While in our two-part implementation, for further development we can add the extension into the user space, which is more efficient and easier.

5.1.5 Netlink

Netlink is a Linux kernel module that enables inter-process communication by socket. We use Netlink to build a data transmission channel between the kernel space and the user space. A message will be handled by a callback function after fair queuing.

In Linux, system calls provide some basic and standardized services for user space processes. If we want to use system calls implement our proposed ZPSM, we need to add customized system calls, which might pollute the kernel and needs compiling every time the kernel module is loaded. To address these concerns, we use Netlink because it is simpler, needs no modification of the kernel, and enables a socket that can be used immediately.

Another essential feature of Netlink is that the socket is asynchronous. We use Netlink to implement a duplex channel between the kernel space and the user space. When user or kernel needs to send a message, Netlink shall put the message into the receiver's message queue. Then the callback function of the receiver will decide to process the message immediately or postpone it. The asynchronous mechanism provides more efficient scheduling of system resources. While system call is a synchronous mechanism, it might have a negative influence on kernel performance in the case that the time to process a message is long.

Figure 5.4 shows how the communication channel is built. First the kernel module creates a socket using netlink_kernel_create function and designates a callback function that can handle message reception. Then user process links a local address to the existed socket and sends the process id of itself to the kernel module. After the callback function of the kernel module receives the process id and also knows that the user process is ready, the channel is completed.

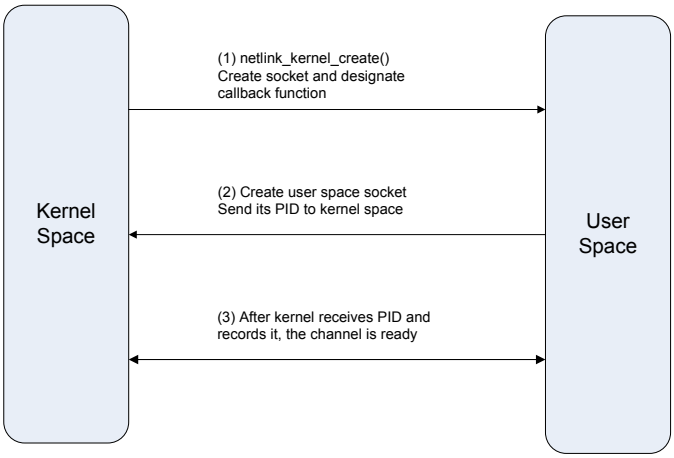

Figure 5.4 Netlink Creation

Below is the code for the kernel module to initiate a socket. The code realizes the procedure

(1) in the Figure 5.4. Here &init_net is an address structure; NETLINK_ZPSM is a certain protocol number which we reserve for the socket; nl_sock is the socket we create; nl_data_ready is the callback function at the kernel side, which is designated at the initiation of the socket and will handle received messages. Note that if the socket is not successfully initiated, an error shall be reported.

```
static int __init netlink_init(void)
{
    printk(KERN_INFO "enter ZPSM module\n");
    //create a netlink socket
    nl_sock = netlink_kernel_create(&init_net, NETLINK_ZPSM, 0,
                    nl_data_ready, NULL, THIS_MODULE);
    printk(KERN_INFO "Netlink init finished\n");
    if(!nl_sock)
        return -ENOMEM;
    return 0;
}
```

After the kernel module initiates a socket, a user process creates a socket with standard socket API: int socket(int domain, int type, int protocol). Below is the code for a user process to create a socket. The code realizes the procedure (2) in the Figure 5.4.

```
static int __init netlink_init(void)
{
    nl_fd = socket(AF_NETLINK, SOCK_RAW, NETLINK_PROTNUM);
    memset(&nl_src_addr,0,sizeof(nl_src_addr));
    nl_src_addr.nl_family = AF_NETLINK;
    nl_src_addr.nl_pid = getpid();
    nl_src_addr.nl_groups = 0;
    bind(nl_fd,(struct sockaddr*)&nl_src_addr, sizeof(nl_src_addr));
}
```

After the user process creates a socket, the getpid function shall obtain the process id of this user process. When all settings are configured, the bind function links the local address to the socket, and the user process sends a message to the kernel with USER_APP_LOADED_MSG, which is pre-defined variable with a constant value, indicating that a user process is ready. The process id should also be contained in the header of this message. The kernel module shall store the process id and then send back an acknowledgement message to the user process. At this point, the duplex channel is ready.

When the kernel module is removed, the socket shall also be canceled. Our program will first check whether the nl_sock is null. If the socket is not empty, it shall be canceled by the sock_release function in the system.

5.1.6 Netfilter

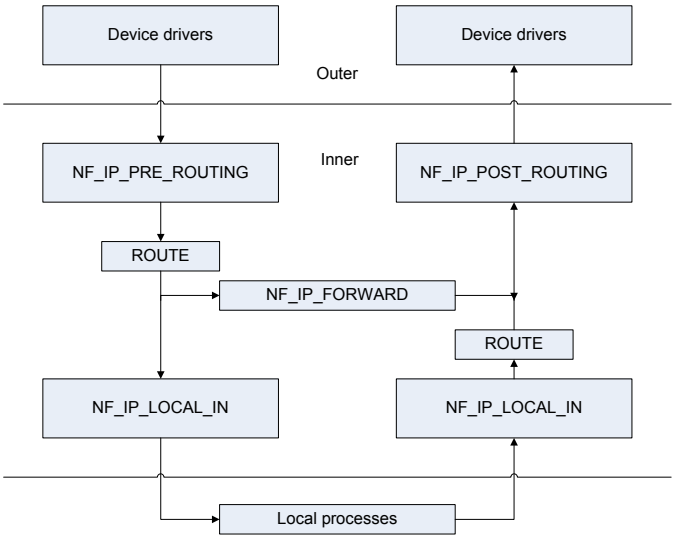

Figure 5.5 Netfilter Hook Points

Netfilter is a set of hooks that enables a kernel module to handle incoming and outcoming data packets. Netfilter provides five hooks at five different points in data transmission process in the IP layer as shown in the Figure 5.5. The five parameters shown below are called hook number, each of which represents a hook point.

Incoming packets:

1) NF_IP_PRE_ROUTING: Handle all incoming data packets before the routing decision is made. The hook is after the reception function ip_rcv. Usually a hook is place here to obtain information such as message type, length, port number and IP address from the packet header.

2) NF_IP_LOCAL_IN: Handle incoming data packets that are sent to the local process after the routing decision is made. The hook is after the local deliver function ip_local_deliver, and will send the data packet to the ip_local_deliver_finish function after processing the packet. A firewall could be place here with a hook and a iptable module to filter data packets.

3) NF_IP_FORWARD: Handle incoming data packets that are forwarded to other notes after the routing decision is made. The hook is after the forward function ip_forward, and will send the data packet to the ip_forward_finish function.

Outcoming packets:

4) NF_IP_LOCAL_OUT: Handle all outcoming data packets before routing decision is made. The hook is after the ip_build_and_send_pkt function. The iptable module can use the hook and related rule chains at this point to filter output messages.

5) NF_IP_POST_ROUTING: Handle outcome data packets after routing decision is made. The hook is after the ip_finish_output function.

Now we introduce how to use Netfilter in the AP. In order to use the hooks provided by Netfilter, first we define a structure array as followed:

```
static struct nf_hook_ops zpsm_ops[] = {
{
    .hook = zpsm_hook,
    .pf = PF_INET,
    .hooknum = NF_INET_PRE_ROUTING,
    .priority = NF_IP_PRI_FIRST,
},
{
    .hook = zpsm_hook,
```

```
    .pf = PF_INET,
    .hooknum = NF_INET_LOCAL_OUT,
    .priority = NF_IP_PRI_FILTER,
},
};
```

Note that we handle incoming and outcoming packets using two hooks that are closest to the MAC layer: NF_INET_PRE_ROUTING and NF_INET_LOCAL_OUT. In Netfilter, different proprieties need to be assigned to each hook. Priority level determines the order of the hook function in a function queue. Each hook has an assigned priority variable; lower value of this variable means higher priority. We want to allocate the highest priority to the hook placed before an incoming packet is handled by routing function so we assign the variable NF_IP_PRI_FIRST with a minimum value INT_MIN. The hook that filters the outcoming has a lower priority with variable NF_IP_PRI_FILTER. The hook function designates zpsm_hook as the callback function to handle hooked packets. Now that the two hook points have been initiated, next step is to register the hook functions.

After the socket is ready between the kernel space and the user space, the user process will prepare a serial port for data transmission and then inform the kernel module after the serial port is ready. When the kernel module knows that the serial port is ready, it will register the hooks using the nf_register_hook function. After the registration, Netfilter will hook data packets at the two points and then the packets go to the callback function, which process the packet and decide destination of it. There are six possible destinations as followed:

NF_DROP: discards the packet releases any resources allocated for the packet

NF_ACCEPT: the packet passes and goes to next stage

NF_STOLEN: the hook will handle the packet and Netfilter stops processing the packet

NF_QUEUE: inserts the packet into a different queue

NF_REPEAT: iterates the same cycle and calls the hook function again

NF_STOP: accepts but holds the packet

We only hook unicast packets. If the data packet is a broadcast or multicast frame, the

callback function will put them back to the network stack. The main purpose that we hook data packets is to maintain the membership. The callback function will check whether the incoming or outcoming packet is associated with a ZigBee member. If it does come from or go to a member device, the callback function will update membership status, setting the last activity time to the current time. After the processing, the callback function will return NF_ACCEPT, which means the packet is put back and can go to the next stage. Note that a special case here is that a packet coming through port 6789 will be sent to the control message handler, and the callback function returns NF_DROP, requiring Netfilter to discard the packet.

In a station, Netfilter has a much simpler application. The hook function will only obtain incoming packet. If the incoming packet is a control message, the callback function will pass it to the control message handler, otherwise, the callback function will obtain the assigned index from the packet header. The callback function then records the index in the kernel and also sends the index to the user process.

5.2 Implementation Details

5.2.1 Overview

We implement the ZPSM on Telos-B Mote and PandaBoard. Both the AP and the station are implemented on a PandaBoard with Telos-B Mote plugged in. TinyOS is run on Telos-B Mote and this part is responsible for ZigBee data packet transmission only. Other parts are built in Linux system on PandaBoard. In the Linux system, first the kernel module is loaded and linked to the socket, then the user process is loaded and linked to the socket so that they can communicate through the socket. After the serial port is ready, a data transmission channel between Linux and TinyOS is built. Now Netfilter will also be registered and corresponding callback function will handle incoming and outcoming data packets.

5.2.2 The AP Architecture

Figure 5.6 shows implementation details of the AP side. The architecture includes both kernel space and user space. The kernel part is responsible for some core functions of our

system: Timer Generator, Packet Handler, and ZigBee Membership.

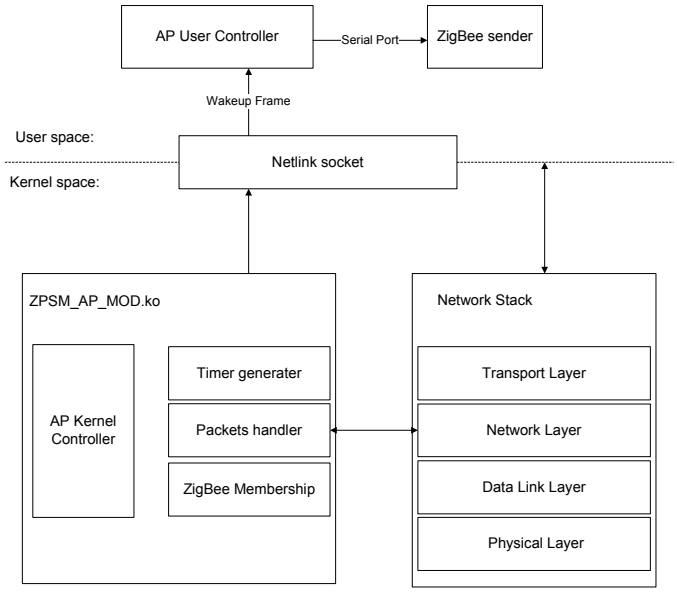

Figure 5.6 Architecture of the AP

The timer generator is built in the kernel in order to avoid the latency resulting from the calling of kernel services. It is more accurate and ensures better performance of the ZPSM. The generator should be able to set up a timer, invoke the timer's call-back function when the timer expires, and cancel the timer if needed. Besides, it should be able to generate multiple timers simultaneously to be used in configuration for several types of time intervals such as wakeup beacon time interval, and the time interval used by the membership manager (MMgr) to periodically update and cancel membership.

The packet handler is built in the kernel because we need to use Netfilter to obtain packet information. The packet handler can hold packets, process packets, poll packet from the network stack and put them back to the stack. It is able to handle packet information such as header, delivery time, destination address, and port number. The ID information contained in the header is used by the MMgr to manage ZigBee membership.

The ZigBee membership component maintains the membership table. It is built in the kernel space because this component needs the time interval generated by a timer, as well as

the ID information obtained from data packets.

The AP kernel controller (AKC) should be able to coordinate the three components in the kernel part. The AKC and AP user controller (AUC) carries out the functions of the ZAC. The AKC logs warnings, and errors generated by ZPSM_AP_MOD.ko, and also generates wakeup beacon frames. It communicates with the AUC via Netlink socket so that the kernel part and the user part can transmit data packets to each other. When wakeup beacon is updated, the AKC shall transmit the wakeup beacon to the AUC; every WI the AKC shall require the AUC to send out the wakeup beacon to the ZigBee sender.

The AP user controller(AUC) should be able to communicate with the AKC. The AUC shall be able to receive control messages from the AKC. It stores the wakeup beacon frame and transmits it to the ZigBee Sender upon request. The AUC should ensure that every wakeup beacon is sent via serial port. After the ZigBee sender received the wakeup beacon, it shall put the frame into the sending queue.

5.2.3 The Station Architecture

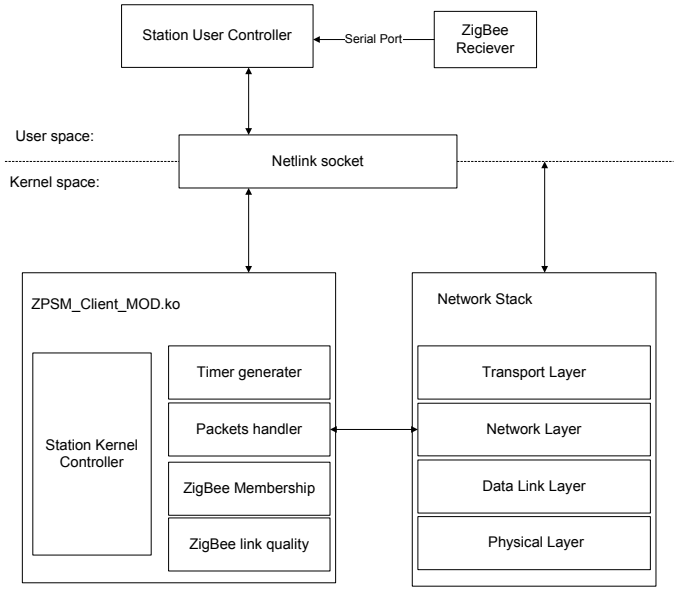

Figure 5.7 Architecture of the Station

Figure 5.7 shows the architecture of the software running at the mobile station side. The architectures of a mobile station and the AP have slight differences. The kernel part in a mobile device carries out similar functions as that in an AP. An extra component, ZigBee Link Quality, is added to control how often the ZigBee channel quality is calculated. We can set different time interval length via this component. Based on this information the Timer Generator will build the timer.

The ZigBee membership component is responsible for membership application, maintenance and cancellation. The station kernel controller (SKC) conducts similar functions as the AKC on the AP. It coordinates the four components on client kernel space and responsible for communication with the station user controller (SUC) in the user space. Furthermore, the SKC should also dynamically require the Wi-Fi radio to wake up proactive every DLI in the case of low channel quality.

The ZigBee receiver in TinyOS wakes up periodically to receive wakeup beacon. Each time the receiver wakes up 10ms before the wakeup beacon in order to adapter to time difference between the station and the AP. After it receives wakeup beacons, it shall transfer the frame to the station user controller (SUC). The SUC obtains the index for the mobile device from the ZigBee membership in the kernel space. Every time when the SUC receives a wakeup beacon, it checks the frames and wakes up the Wi-Fi interface if there is data packets buffered at the AP. The SUC should also be responsible to calculate the ZigBee channel quality p and DLI. If DLI is positive and the change in DLI is larger than a designated threshold, the SUC shall send the value of DLI to the SKC so that Wi-Fi radio wakes up proactive.

5.2.4 Packet Loss and Channel Quality

To address the channel quality issue, we calculate channel quality periodically and add proactive Wi-Fi wake-ups. Each station is responsible for calculating its channel quality and deciding whether to change DLI.

Detailed flowchart of this part is shown in Figure 5.8. DEFAULT_HEARTBEAT is set as 2 minutes. Initially ZigBee channel quality is good. DLI is equal to two minutes, which means except normal wake-up every LI, the station only wakes up every two minutes when it sends

heartbeat message. When channel quality is not stable and DLI needs to be reset, DLI is calculated using formula 4.1. When channel quality is low and DLI becomes smaller, DLI is reset immediately. If channel quality gets better and DLI increases, in order to avoid frequent change, DLI is only reset if it the change in calculation result is larger than the threshold, set as 20mm. When DLI is reset, the mobile device sends message to the AP through Wi-Fi interface so that the Wi- Fi radio is turned on. After the AP receives this message, it will send back an acknowledge frame so that the mobile device can check from the frame whether there are buffered packet at the AP or not. This mechanic can reduce packet loss ratio when ZigBee channel quality is low.

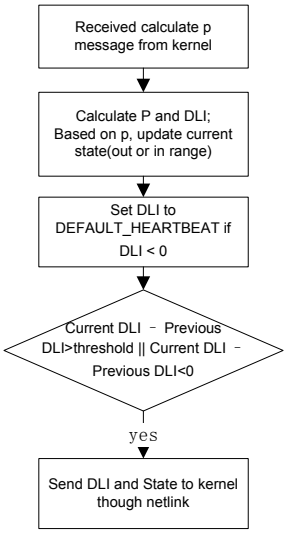

Figure 5.8 DLI calculation

CHAPTER 6. EXPERIMENT

In order to measure the performance of our proposed system, we have chosen the following measurement metrics:

Energy per packet (mJ/pkt): Total amount of energy consumed divided by the number of packets transmitted. Total energy here refers to the energy consumption for the ZPSM system, including energy consumed by Wi-Fi interface and ZigBee interface. Power consumption rate of wireless module and ZigBee component from Qin and Zhang (2012) is shown in Table 6.1.

Delay-meet ratio: The percentage of data packets that are delivered within delay bound. A data packet is sent every one second. To simulate unstable ZigBee channel quality, we randomly drop some wakeup beacon frame.

Table 6.1 Power Consumption Rate

Network size	20	clients Packets collected	200,000
WiFi range	120 m	ZigBee range	100 m
Wakeup slot (W)	40 ms	Update interval (UI)	10 s
MAC header	34 bytes	WiFi channel bit rate	54 Mbps
PHY header	17 bytes	WiFi basic bit rate	1 Mbps
Beacon packet	28 bytes	SIFS	16 s
PS-POLL	20 bytes	DIFS	34 s
ACK	14 bytes	Beacon interval (B)	100 ms
Data packet	2312 bytes	ZigBee channel bit rate	250 Kbps
WiFi transmission	1.152 Watt	ZigBee transmission	0.087 Watt
WiFi reception	0.561 Watt	ZigBee reception	0.072 Watt
WiFi idle listening	0.462 Watt	ZigBee idle listening	0.019 Watt

In our experiment, we have one AP and three stations. Both the AP and the station are built on PandaBoards with Crossbow ZigBee-enabled Telos-B motes plugged in. We set Wi-Fi interface to channel 8 and ZigBee interface to channel 26 so that they will not interfere with

each other. The ZigBee interface on stations wakes up every WI to receive message from the AP, and the underlying MAC protocol is CSMA/CA compliant with IEEE 802.15.4 standards. First we measure the number of data packets received, acknowledgment packets transmitted, and control messages transmitted. Then according to the Table 6.1 we calculate the total energy consumed by the system. Another energy consumption item is Wi-Fi radios waking up, which is 1.5mj per waking up according to Qin and Zhang (2012). Detailed calculation approach is described in next section.

6.1 Energy Consumption Calculation

In our experiment the AP sends one data packet per second to each of the station. In order to calculate the energy consumption of each data packet for a station, we need to consider four respects:

(1) ZigBee Usage

The AP broadcasts ZigBee wakeup beacon frame every WI, which is 40ms in our experiment. There are 25 WIs in one second, so that the energy consumption per wakeup beacon reception needs to be multiplied by 25. Besides, since the station's ZigBee interface is awake 10ms before the beacon time, we also need to calculate the idle listening energy consumption: the consumption rate multiplied by 10ms which is the approximately the idle time for each beacon.

(2) Dynamic Wakeup

Due to unstable ZigBee channel quality, a station also wakes up every DLI which is calculated using the ZigBee channel quality and the delay bound. Energy consumption includes Wi-Fi radios wake-up and control message transmissions. We first calculated average value of this interval from the experiment data, then determine how many DLIs the station has during each experiment, and calculate the energy consumed by dynamic wake-up.

(3) Reception of Data Packet

If a data packet is transmitted after the station receives wakeup beacon, energy consumption includes that consumed by Wi-Fi radios wake-up, control message transmission, acknowledgement message reception, data packet reception, and acknowledgement message transmission after successfully receiving the data packet. If a data packet is transmitted after the station

receives beacon frame, energy consumption includes that consumed by PS-Poll frame transmission, data packet reception, and acknowledgement message transmission after successfully receiving the data packet. Time needed for transmission and reception is calculated as the corresponding packet size divided by Wi-Fi channel bit rate. Then energy consumed is calculated using the time multiplied by relevant energy consumption rate.

6.2 Experiment Result: Ideal Channel Quality

To simulate good and stable channel quality, in this set of experiments the AP does not drop any wakeup beacon frames. Each experiment is repeated 10 times. In order to measure system performance when multiple nodes join the network, under delay bound of 100ms the stations have joined one by one and the data are recorded when there are one, two and three stations. Since the channel quality is good, there is no dynamic wake-ups in this set of experiments.

The results show that under the good ZigBee channel quality, average observed delay is around 39ms and average delay meet ratio is 100%. We have also measured the scalability of the network by adding stations one by one. Figure 6.1 shows that per-packet energy consumption does not change when number of stations increases.

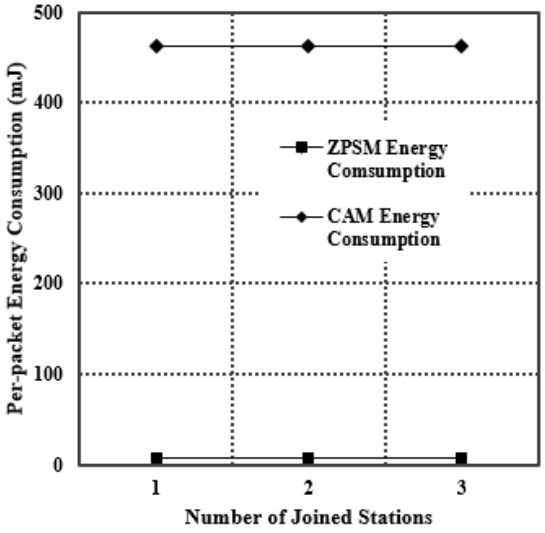

Figure 6.1 Performance of ZPSM in constant ZigBee Channel Quality

6.3 Experiment Result: Varying Channel Quality

To simulate the unstable low-quality ZigBee channel, the AP random drops some wakeup beacon frames. For example, if the channel quality is 0.9, the probability that a wakeup beacon is lost is 10 percent. The AP uses a random function and determines whether a wakeup beacon frame is dropped or not before sending the frame. We have six scenarios from channel quality of 1 to channel quality of 0.5, decreasing 0.1 per time. Delay bound is 100ms. In this set of the experiments, stations shall add energy consumed by dynamic wake-ups.

From Figure 6.2 we can see that when the ZigBee channel quality decreases, the energy consumption per packet increases and the delay meet ratio decreases. Energy consumption per packet increases because under the lower ZigBee channel quality the Wi-Fi radio will be dynamically waken up more frequently, which consumes more energy. Delay meet ratio decreases because the opportunity that ZigBee wakes up Wi-Fi radio on demand decreases - a packet is timed out if the station miss several consecutive wakeup beacon frames and the station does not have a listening or dynamic listening wake-up during this period.

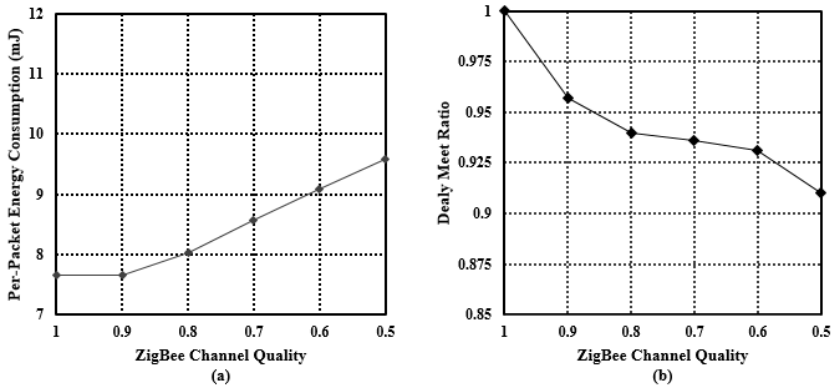

Figure 6.2 Performance of ZPSM in different ZigBee Channel Quality

Our proposed ZPSM has a high performance when the delay bound is small because of the relatively high delay meet ratio and the low energy consumption. Under the delay bound of 100ms, the PSM has energy consumption of 8.25 mJ/pkt, but the delay meet ratio is 51%, which is much lower than ZPSM even when the ZigBee channel quality is low. In most of cases

when there is a required delay meet ratio, a station shall be switched to CAM in order to ensure the requirement. In CAM, energy consumption is 462.57 mJ/pkt, which is much higher than in ZPSM.

CHAPTER 7. CONCLUSION

The ZigBee devices, with the low power and low cost feature, have a wide range of applications and are expected to be plugged into more and more mobile devices. This thesis proposes a ZigBee-assisted Power Saving Mode (ZPSM) in a Basic Service Set (BSS) network compliant with IEEE 802.11 standard. The ZPSM, built atop the standard PSM, enables on-demand wake-up of stations to enhance Wi-Fi performance and reduce energy consumption of the mobile devices.

The Wi-Fi radio can consume large energy during transmission, even when staying idle. A typical approach to save energy is the standard PSM, which enables a station to wake up periodically every LI to retrieve data packets. However, the standard PSM needs to make tradeoff between the network performance and the energy efficiency, because long LI may result in delay of data packets and short LI may result in higher energy consumption.

In this thesis we proposes a new approach, ZPSM: (a) the ZigBee interface wakes up the Wi-Fi radio of a station when there are incoming data packets for this station, and (b) the Wi-Fi radio also proactively wakes up in a certain interval according to different ZigBee channel quality. The period time interval in (b) is a dynamic interval that can ensure a certain portion of data packets can be received within delay bound when the ZigBee channel quality is low.

Experiments have been conducted to test the feasibility of implementing ZPSM in resource constrained mobile devices and the performance of ZPSM. The results show that the ZPSM can achieve energy efficiency while still meet delay requirements. Compared to the standard PSM, it can achieve shorter packet delay that is required by the applications. Meanwhile our proposed ZPSM consumes much lower energy than CAM does.

BIBLIOGRAPHY

Agrawal, P., Kumar, A., Kuri, J., Panda, M. K., Navda, V., and Ramjee, R. (2010). Opsm-opportunistic power save mode for infrastructure ieee 802.11 wlan. In *Communications Workshops (ICC), 2010 IEEE International Conference on*, pages 1–6. IEEE.

Ananthanarayanan, G. and Stoica, I. (2009). Blue-fi: enhancing wi-fi performance using bluetooth signals. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 249–262. ACM.

Krashinsky, R. and Balakrishnan, H. (2005). Minimizing energy for wireless web access with bounded slowdown. *Wireless Networks*, 11(1-2):135–148.

Kravets, R. and Krishnan, P. (1998). Power management techniques for mobile communication. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 157–168. ACM.

Love, R. (2010). *Linux Kernel Develpmnet*. Addison Wesley.

Namboodiri, V. and Gao, L. (2008). Towards energy efficient voip over wireless lans. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 169–178. ACM.

Pering, T., Agarwal, Y., Gupta, R., and Want, R. (2006). Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232. ACM.

Pyles, A. J., Ren, Z., Zhou, G., and Liu, X. (2011). Sifi: exploiting voip silence for wifi energy savings insmart phones. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 325–334. ACM.

Qiao, D. and Shin, K. G. (2005). Smart power-saving mode for ieee 802.11 wireless lans. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1573–1583. IEEE.

Qin, H. and Zhang, W. (2012). Zigbee-assisted power saving management for mobile devices. In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pages 93–101. IEEE.

Rostli, J. (2012). Taztag. http://www.nfc-phones.org/taztag-tph-one/.

TinyOS (2012). Serialforwarder. http://sing.stanford.edu/tinyos-wiki/index.php/Mote-PC_serial_communication_and_SerialForwarder_(TOS_2.1.1_and_later).

TinyOS (2013). Documentation. http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page.

Wikipedia (2013). Number of mobile phones in use. http://en.wikipedia.org/wiki/List_of_countries_by_number_of_mobile_phones_in_use/.