2014

Implementing ZigBee assisted power management for delay bounded communication on mobile devices

Jinu Susan Varghese
Iowa State University

Implementing ZigBee assisted power management for delay bounded communication on mobile devices

by

Jinu Susan Varghese

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

Johnny S. Wong

Shashi K. Gadia

Iowa State University

Ames, Iowa

2014

DEDICATION

I would like to dedicate this thesis to my cat, Tigger who have been very loving and gave me company during the writing of my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Wensheng Zhang for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Johnny Wong and Dr. Sashi K. Gadia. I would like to thank Qin Hua for his research work which laid the foundation for this work. I would also like to thank all the members in the LAWNS laboratory for their advice and feedback.

ABSTRACT

Over the years WiFi has gained immense popularity in networking devices to transfer data over short distances. WiFi communication can consume a lot of energy on battery powered devices like mobile phones. The Standard Power Saving Management(SPSM) which is part of the standard specification for wireless LAN technology has been applied widely. However, it may not deliver satisfactory energy efficiency in many cases as the wakeup strategy adopted by it cannot adapt dynamically to traffic pattern changes. Motivated by the fact that it has been more and more popular for a mobile device to have both WiFi and other low-power wireless interfaces such as Bluetooth and ZigBee, we propose an implementation of a ZigBee-assisted Power Saving Management (ZPSM) scheme, leveraging the ZigBee interface to wake up WiFi interface based on the delay bound improve energy efficiency [Qin and Zhang (2013)]. The results obtained by applying this scheme on a Linux environment shows that ZPSM can save energy significantly without violating delay requirements in various scenarios.

CHAPTER 1. INTRODUCTION

1.1 Motivation

Mobile devices nowadays are extremely powerful devices used by people to stay in touch with others and manage everyday tasks. Mobile devices like smartphones are battery driven and hence they are energy constrained. And so their operational time is limited. It often happens that users forget to recharge their phones for days and a power source is not always available when the battery dies. Hence, it is important to address the energy consumption problem. A longer operational time is a desirable feature for users as well as the mobile manufacturers who are interested in offering longer operational times to their customers.

Smartphones dominate the mobile devices market and they are equipped with multiple network interfaces like Wi-Fi and bluetooth along with the cellular interface. WiFi technology is specified in the IEEE 802.11 standard [IEEE (1999)]. The presence of WiFi on mobile devices enables users to connect to networks without use of cables to the Internet which makes wireless networks very popular. Most of the web applications demand high bandwidth and low latency and high throughput. WiFi supports very good data rates, high latency and throughput. Also, the range of WiFi spans several meters. Hence it makes WiFi a very reliable and indispensible technology in present day mobile devices.

Unfortunately, WiFi consumes a lot of power when it is turned on always. This is undesirable as this would drain the battery of the mobile device very quickly and the user would have to keep recharching through an external power source. Users can conserve power by turning off the WiFi when they do not use it, but this can burden the user.

Improving energy efficiency of Wi-Fi networks is a long standing problem in wireless networks. The WiFi device supports two power management modes when operating in the infrastructure mode. A WiFi device operating in the Infrastructure mode would typically connect to an access point. One of the WiFi modes in which the radio is always ON is called constant awake mode(CAM). This mode ensures that the data transmission is prompt but it incurs a lot of energy consumption. The other one is the standard power save management(SPSM) which has been proposed in the IEEE 802.11 standard [IEEE (1999)]. The standard PSM (SPSM) has been widely adopted and is implemented by most wireless protocol stack across mobile devices. The key idea is that a WiFi radio interface can have several low energy consuming modes like the sleep mode, the idle listening mode apart from the awake mode. A station or mobile device which is in standard PSM (SPSM) would wake up periodically by reducing the time it spends listening for data packets and thereby reducing the energy consumption. Studies have shown that the battery lifetime lasts only several hours when a mobile device is in constant awake mode(CAM) and a slightly better lifetime when in standard PSM (SPSM) mode when compared to the battery lifetime when the WiFi radio interface is completely switched off [Shih et al. (2002)].

In order to improve the energy consumption against the standard PSM (SPSM), adaptive powersaving management(PSM) schemes were proposed [Krashinsky and Balakrishnan (2005)] in which the WiFi radio switches between the PSM and CAM modes according to some heuristics.

Another approach to improve the energy efficiency is to rely on another network interface apart from WiFi. One such radio interface is the Bluetooth. Some techniques have been proposed which saves energy by switching between low power bluetooth and back to WiFi only when required [Ananthanarayanan and Stoica (2009)]. But Bluetooth has limited communication range and network size. Another such low power radio interface that is gaining popularity is ZigBee. Mobile phone manufacturers have started embracing ZigBee technology as seen in the unveiling of the first Android phone with ZigBee capability [TazTagInc (2014)]. So it would be more commonly embedded in wireless devices with WiFi devices in near future. Unlike Bluetooth, it has lower energy consumption rate and longer communication range. But it cannot replace WiFi due to its low data rate [ZigBeeAlliance (2014)]. But this is a perfect channel for control messages and this can be exploited to design a mechanism to save power. This is the basis for the work done in this thesis which has been drawn from [Qin and Zhang (2013)] that proposes the ZigBee-assisted power saving management(ZPSM) for WiFi devices aiming to deliver energy efficiency with bounded packet delivery delay. The key idea is to use the low power ZigBee radio to dynamically wake up asleep high power WiFi radio for packet transmission between the AP and clients. ZPSM is build on on top the existing standard PSM(SPSM) scheme, hence no changes have been done to the WiFi protocol stack.

1.2 Contribution

In this work we demonstrate how we have achieved power efficiency by incorporating ZigBee-assisted power saving management (ZPSM) in real world environment. In order to do this, the implementation is carried out on Pandaboard [pandaboard.org (2014b)] which is an open mobile platform. The Pandaboard is equipped with TiWi [LSR (2014)] which has Integrated Dual-Mode Bluetooth and 802.11 b/g/n WiFi Module. The hardware setup involves implementing the ZigBee-assisted power saving management(ZPSM) system on the Linux 3.2 kernel which is deployed on Pandaboard ES Rev B2 platform. The system is evaluated and experimental results show that there is a significant energy saving when compared to CAM mode while achieving the desired delay bound requirement demanded by the ZPSM clients which is not promised by standard PSM (SPSM). Experiments have been done on real hardware rather than a simulation to make sure that the following objectives of the ZigBee-assisted power saving management(ZPSM) system has been achieved. The objectives has been achieved as adapted from [Qin and Zhang (2013)]:

- Energy Efficiency: ZigBee which consumes much lesser power than WiFi is used to exchange control messages between a client and the AP that tries to wake up the client WiFi radio only when it has a packet destined to it and the packet cannot be delivered within a certain required delay bound if the WiFi radio is not woken up. Thus it minimizes unnecessary wake up and idle listening, saving a lot of power when compared to CAM mode. In the infrastructure mode, the AP sends beacon packets every beacon interval(BI) which is 100ms. So the maximum delay that a packet could have is 200ms.

Once the client receives the beacon with the TIM (Traffic Indication Map) containing its association id, the client sends an acknowledgement and in the next beacon interval the data is exchanged. So it must be ensured that the delay requirement is met when the client is within and outside ZigBee range. When the client is in the ZigBee range the WiFi radio wakeup is controlled by the ZigBee control messages whereas when the clients move away from the ZigBee range but is within WiFi range, the delay bound and delay-meet requirement is satisfied by changing the listen interval (LI), the time at which the client polls for data from AP.

- Bounded Delay: In our approach we try to make sure that the client WiFi radio wakes up around the desired delay bound enforced by the client. In case of SPSM, the delay requirements for packets cannot be met always as the wake up period is usually 100ms equivalent to one BI (Beacon Interval). So in the worst case, the delay would be 200ms. This is contributed by delay cause due to two BIs(Beacon Intervals), one BI delay for the AP to wait for client to wake up and one BI delay for the client to be served by the AP. Our results show that there is a huge improvement in meeting the delay requirements and also the average delay of a packet is reduced compared to SPSM which matches the design objective in [Qin and Zhang (2013)].

- Compatibility: The system must be compatible with the existing 802.11 standard. The IEEE 802.11 standard [IEEE (1999)] proposes the standard PSM (SPSM) which is part of the WiFi protocol stack supported by Linux operating systems. In this work, it has been made sure that the standard implementation has not been altered.

1.3 Organization of the thesis

In the rest of the paper, Chapter 2 presents work related to the one presented here. This is followed by Chapter 3 that elaborates on the design of this technique after which Chapter 4 is presented. It deals with the implementation details. This is followed by Chapter 5 that talks about the experiments that have been conducted and our observations on the test environment. Section 6 summarizes and concludes the paper and describes our plans to extend this work in the future. Appendix A describes the terms and abbreviations used in this thesis. Appendix B contains the hardware and environment setup for the evaluation of this work. Main code snippets have been captured in Appendix C.

CHAPTER 2. RELATED WORK

WiFi technology has been around for a while and numerous techniques have been proposed to improve the WiFi energy efficiency on mobile devices. In this chapter, some of these techniques have been outlined-some that use a separate radio interface that works in conjunction with the Wi-Fi radio to make it more energy efficient and some that adapt the existing power management techniques running on the Wi-Fi radio stack.

Some work has been done to ensure that performance is not affected when using standard PSM (SPSM). In [Krashinsky and Balakrishnan (2005)], the authors have studied the correlation between energy saving protocols and TCP-performance for web applications. The problem of how the standard PSM affects the round trip delay is studied and a protocol called the Bounded Delay protocol has been proposed. This strategy ensures that the energy consumption is minimized without affecting the Round Trip Time (RTT). They have achieved considerable improvement in energy consumption against the Constant Awake (CAM) mode. A Smart PSM scheme has been proposed in [Qiao and Shin (2005)]. In this the desired user delay is translated into a penalty function which is thereby translated into an optimal WiFi radio sleep/wakeup sequence. Hence the energy consumed is minimized under the user delay constraints.

In [Agrawal et al. (2010)], the authors propose an algorithm called opportunistic PSM (OPSM) which is takes advantage of the activity over the WiFi radio which means that the radio needn't wakeup as frequently as in the case of static PSM (SPSM). So based on the traffic pattern the wakeup sequence changes which saves power.

WLAN radios typically conserve energy by staying in sleep mode. With real-time applications like Voice Over Internet Protocol (VoIP), the continuous nature of traffic makes it difficult for the radio to stay in the lower power sleep mode enough to reduce energy consumption significantly. In [Namboodiri and Gao (2010)], the authors propose the GreenCall algorithm to derive sleep/wake-up schedules for the WLAN radio to save energy during VoIP calls while ensuring that application quality is preserved within acceptable levels of users.

There have been work done in which a secondary radio is used in conjunction with the WiFi to save power. One of the former work done was proposed in the Wake-on-Wireless scheme [Shih et al. (2002) in which the cellular radio is used. This strategy was used in a PDA-based in which they tried to save energy by reducing the idle power. To do this, the phone was in a shutdown state most of the time and the radio was turned on when an incoming call was received. In this way the WiFi radio is not used all the time and only when taking calls.This way they were able to increase the battery lifetime considerably by 115%.

Most of the smartphones have a low-power/low-bandwidth bluetooth interface apart from high-power/high-bandwidth WiFi as a secondary radio for personal area network. In [Yoo and Park (2011)], the authors present an approach based on a distributed clustering called Cooperative Networking protocol (CONET). This technique improves the energy efficiency of wireless networks by dynamically configuring clusters according to each nodes bandwidth, energy, and application type. In this one of the nodes act as a cluster head in a PAN which would act as a gateway between the Personal Area Network (PAN) and WLAN.

[Ananthanarayanan and Stoica (2009)] presents a simple scheme called Blue-Fi in which energy is conserved by using WiFi only when WiFi connectivity is available. This reduces idle power consumed when trying to discover connectivity and thus saves power. The WiFi connectivity is predicted using bluetooth contact patterns and cell tower information.

[Agarwal et al. (2007)] presents Cell2Notify, a practical and deployable energy management architecture that leverages the cellular radio on a smartphone to implement wakeup for the high-energy consumption WiFi radio. Cell2Notify attempts to minimize energy consumption by powering off the Wi-Fi interface when there is no VoIP call in progress, and powering it on only on the reception of an incoming VoIP call.

CoolSpots are a concept introduced in [Pering et al. (2006)] that proposes a mechanism to enable a wireless mobile device to automatically switch between multiple radio interfaces, such as WiFi and Bluetooth, in order to increase battery lifetime. The mechanism chooses bluetooth if the application requires low bandwith and less performance and it would switch over to WiFi it requires higher bandwidth and better performance.

[Jin et al. (2011)] proposes the WiZiCloud in which both mobile phones and access points have dual WiFi-ZigBee radio. This scheme saves energy by using ZigBee to maintain connectivity. Since WiFi radio isn't used for this purpose it saves energy.

[Qin et al. (2011)] presents a system for ZigBee-assisted WiFi transmission in which mobile devices form clusters. The devices in the cluster save power by taking turns to transmit packets. This reduces the contention and collision as well as energy consumption.

CHAPTER 3. DESIGN

3.1 System Model

The system model in the implementation has been adapted from [Qin and Zhang (2013)]. A wireless network can be in either be an adhoc network or an infrastructure network . Our work has been focused only on the infrastructure network as it is more widely used. An infrastructure network includes one or more access points (APs) and mobile stations connected to the access point (AP) that form a Basic Service Set (BSS). This network is identified by a special identifier called the service set identifier (SSID). The system consists of an access point(AP) and several clients. This comprises the ZigBee Power Save Management(ZPSM) network. Each access point is known by a unique identifier called the basic service set identifier (BSSID). The packets in the network carry this basic service identifier (BSSID) to trace back to the originating network. The AP makes sure that all the clients are time synchronized so that all the stations within the network are synchronized to a common clock. [IEEE (1999)]

The AP and clients have both a WiFi interface and ZigBee radio interface on it. Additionally, the AP is considered to be static and the WiFi mode is always awake (CAM) mode which means that the AP is assumed to be always available with no energy constraints and stationary which is the case with most WiFi access points. The AP is also equipped with a short range radio interface called the ZigBee. Due to the high throughput, better range and bandwidth, WiFi is used to exchange data packets whereas due to the short range and possible packet loss while using the ZigBee radio interface, it is mainly used to send control messages between the clients and AP. This system operates purely in infrastructure mode.

On the other hand, the clients that are equipped with the WiFi radio interface and the ZigBee radio interface are energy constrained. The WiFi radio of the client is in infrastructure mode with the standard power save management (SPSM) enabled. The control messages send by the ZigBee radio interfaced with the AP are used to wake up the client WiFi radio which is in idle listening mode. Since ZigBee is a short range radio, it is susceptible to packet loss so some control messages could be lost because of which the wake up frame interval (WI) which determines how often a control message called Wakeup Frame (WF) is broadcasted by the AP. Since the clients are mobile and the ZigBee range is much lesser than the WiFi range, the clients could move out of the ZigBee network and be connected using only WiFi interface. So it could move outside the ZigBee range and then after sometime come back to ZigBee range which means a client alternates between the standard PSM (SPSM) and ZigBee PSM (ZPSM). It is practical scenario that the access point can be in a confernece room, cafe, etc and mobile nodes which are the clients can move away from the range of these access points. These changes in client lifetime has been captured in the design. We assume in the evaluation that the client mobility is very low. Here the clients are assumed to be time synchronized with the AP.

Some assumptions have been made in measuring the effectiveness of the implementation of the design in [Qin and Zhang (2013)] which is elaborated in the following paragraphs.

For most web applications, it is crucial that the delay bound be within a certain limit to ensure satisfiable quality of service (QoS). In infrastructure mode, all data goes through the AP and each of the clients connected to this AP has a desired delay bound. The delay can be defined as the time elapsed from the arrival of a packet at AP to the receipt of the packet at destination client. Since the client radio is awake, the uplink transmission time is negligible. Let client i have a desired delay bound for downlink packet transmission say d_i [Qin and Zhang (2013)].

Now, we define a delay meet ratio, δ_i. The percentage of packets received with a delay lower than the desired delay bound d_i among all incoming packets should be at least δ_i (called delay-meet ratio), where $0 < \delta_i < 1$. This is called delay requirement [Qin and Zhang (2013)].

Since any radio interface is susceptible to some packet loss, some of the packets may not reach within the desired delay bound the web application running on the client demands. This is particulary true for the ZigBee channel due to its unreliable link quality due to which the ZigBee packet transmission may fail. We use the link quality p_i to represent the probability that a packet sent by the AP through its ZigBee interface arrives at client i successfully. The value of p_i may vary over time depending on obstructions between the ZigBee radio interfaces between the ZigBee PSM (ZPSM) client and ZigBee PSM (ZPSM) AP and also interferences from other radio could affect this value. It has been observed that there is high packet loss if the channel used by both WiFi radio interface and ZigBee radio interface is the same. So it has been ensured that both radio interfaces are configured with different channels [Qin and Zhang (2013)].

3.2 Wakeup Strategy

3.2.1 Overview

In the standard PSM (SPSM) scheme, the WiFi radio wakeup periodically every listen interval (LI) during which it sends a PS-POLL message. This is when the radio wakeups and is ready to exchange data with the access point to which it is connected. However in the ZPSM scheme, the AP keeps broadcasting control messages which the client receives when the client is in ZigBee range. These control messages has information that notifies the prospective client when to turn on the radio. This control message is called the wakeup frame (WF) and this wakeup frame is broadcasted by the AP periodically within a certain interval called the wakeup interval(WI). Each tiem a client joins the ZPSM network, it is assigned a unique identifier. Also, each client packet would have a delay bound that the system should meet. The wakeup frame broadcasted by the AP contains information regarding the packets buffered by it for various client. This includes the unique authentication identifier assigned to the client and the delay bound for the client packet. When a client receives the wakeup frame through its ZigBee radio interface, it checks if its authentication identifier is included in the received wakeup frame. If that is the case, it extracts the delay bound for that client. Based on the delay bound it receives, the client makes sure that the WiFi radio is turned on within this delay bound. In this way the delay bound requirement is met. But if it moves out of the ZigBee range, based on the link quality and delay bound desired, a listen interval is calculated and the WiFi radio is woken up periodically. This scheme is similar to standard PSM (SPSM) but unlike SPSM, it ensures a better delay bound and hence the delay-meet ratio at the expense of a higher energy consumption when compared to SPSM.

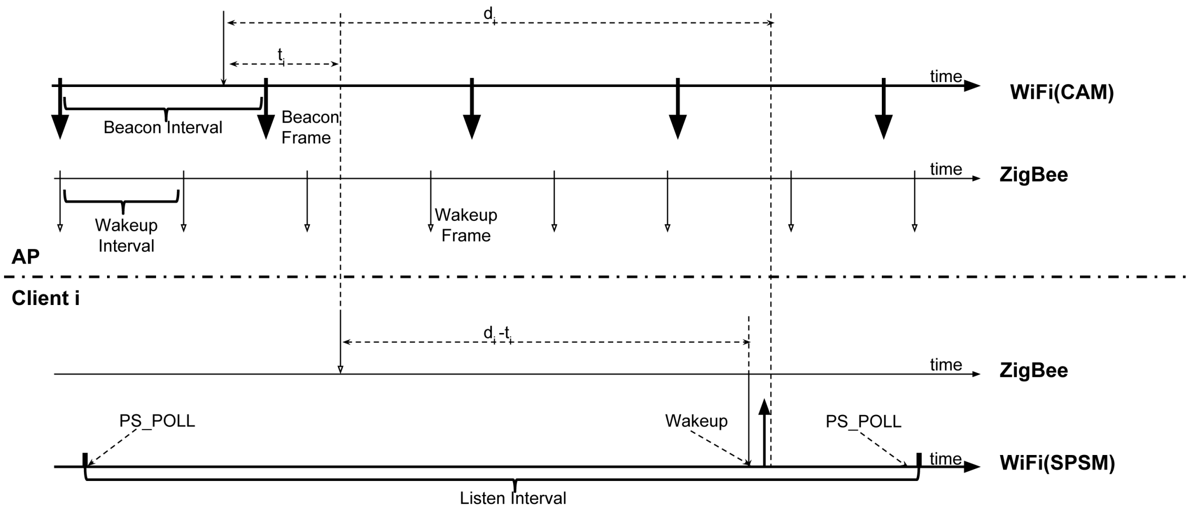


Figure 3.1: Wakeup behavior [Adapted from Qin and Zhang (2013)]

3.2.2 Wakeup Algorithm

The wakeup frame mechanism has been shown in Fig 3.1. The AP tries to synchronize the entire basic service set (BSS) by transmitting beacons according to an attribute set in the AP called the beacon period or beacon interval (BI). An AP schedules to send a beacon frame every Target Beacon Transmission Time (TBTT). The stations that have the power save (PS) mode enabled shall communicate this by setting the power management bit within the frame control field in the MAC header of the transmitted frames. The MSDUs of the stations operating in power save (PS) mode are buffered in the AP. The AP shall make known to the stations about buffered MSDUs by including a Traffic Indication Map (TIM) in the beacon frame. The stations listen to these beacons send by the AP only at designated times specified by the listen interval (LI) parameter. This is made known to AP when the station joins the AP. Once a station finds that it has buffered MSDUs in the AP on receiving the traffic indication map (TIM), it would send a PS-POLL to the AP and the AP would send the buffered MSDUs at a later time. In this way, the STA transitions between the awake mode during which it consumes full power and the sleep mode which is the low power mode in which it is unable to transmit or receive any packets [IEEE (1999)].

3.2.3 ZigBee packet loss

Since ZigBee is a low range radio, there is a chance of packet loss which could affect the delay meet ratio. In order to make sure that the delay requirement is met even when the zigbee channel quality is poor, an additional parameter called the Dynamic Listen Interval is computed. This is different from the Listen Interval parameter in the Standard PSM (PSM) mode when the radio wakes up every 200ms based on the power management protocol. The idea behind this is similar to that Listen Interval but ensures the delay meet ratio is met by considering the delay bound and the ZigBee channel quality. Based on these parameters a Dynamic Listen Interval is computed, the details of which are given in Section 3.4.2.2

3.3 Membership management

The access point traditionally keeps track of the clients that are connected to it using their WiFi radio interfaces in infrastructure or managed mode. But a client that is equipped with ZigBee radio interface and that is connected to the ZigBee radio interface of the AP must be differentiated. Hence on top of the traditional access point there is a need to keep track of the clients that have ZigBee radio interface on them. Also, clients move in and out of ZigBee range to WiFi range and the membership of such clients must be tracked [Qin and Zhang (2013)].

3.3.1 Beacon Exchanging

Each client periodically broadcasts a control message requesting for a authentication identifier if it has not been assigned to it. This is done over the WiFi radio interface as the number of such messages send is not high compared to data packets. The AP sends beacons periodically in which it sends the Traffic Indicator Map (TIM) information that contains information about data packets for clients in the WiFi network. Apart from this, the AP also sends control messages called wakeup frames periodically every wakeup interval [Qin and Zhang (2013)].

3.3.2 Client Joining

A client needs to first join a WiFi network following the standard 802.11 procedure before it is allowed to join the ZigBee network managed by the same AP. Only once the mobile station (STA) is associated with an access point (AP) can it send data via the AP. In order to establish connection with the AP, the client has to send an ASSOCIATE request. Upon receipt of the request, the AP sends a confirm and the AP assigns an association id [IEEE (1999)]. Now when the client is in ZigBee range and the client would listen to the wakeup frames that are being broadcast by the AP. At this point, the client is a SPSM client. Since a valid authentication identifier has not been assigned to the client, it sends an association request to the AP. Upon receiving the association request packet, the AP first checks if the client is already associated with the WiFi network (i.e., wireless LAN) managed by itself. If so, the AP accepts the association request and sends an association response packet containing the unique ID of the network (i.e., BSSID) as well as a unique index (typically the smallest unused index in the network) assigned to the client. Once the index packet is successfully received by the client, it switches to ZPSM mode [Qin and Zhang (2013)].

3.3.3 Client Leaving

When a ZPSM client with index say i finds it has moved out of the ZigBee range of its AP, i.e., failing to receive wakeup frames for a certain period of time, it can default to either SPSM or CAM, depending on the delay requirements. On the other hand, if the AP cannot receive the beacon message from client i for a certain period of time, it automatically dissociates with client i and assigns the index i to the client with the largest index in the network by sending a re-association packet. The goal is to keep continuity of indices in the network, which can ensure the efficiency of utilizing the bits of wakeup frame. Meanwhile, since it is likely that the AP cannot receive beacon messages from the client but the client can receive wakeup frames from the AP, the AP sends a ZPSM dissociation packet to the client through its WiFi interface so as to make sure that the client is aware of this and switches to SPSM or CAM accordingly. In addition, to avoid interference with wakeup frames, all ZigBee control packets (i.e., association request, association response and re-association packets) and beacon messages are always sent during the gap between any two consecutive wakeup frames[Qin and Zhang (2013)].

3.4 System Architecture

The system is mainly constituted by the access point and several clients both of which are equipped with the ZigBee radio interface and the WiFi radio interface. Each of the systems are equipped with several components which are outlined in Fig 3.2. The details of each component is given in the following sections.

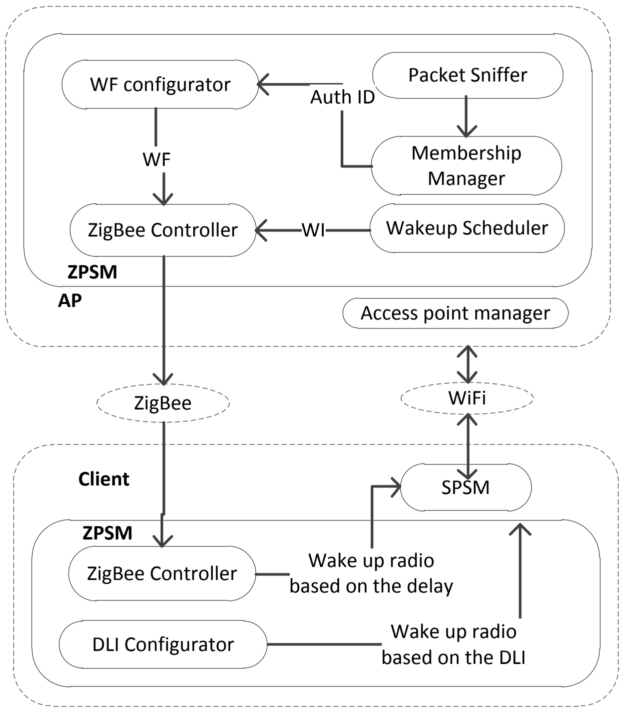


Figure 3.2: ZPSM Architecture [Adapted from Qin and Zhang (2013)]

3.4.1 AP

3.4.1.1 ZigBee Controller

The ZigBee controller is the one that interfaces the ZigBee radio of the AP with that of the client when the client is in ZigBee range. The controller interfaces with the framework configurator which configures the wakeup interval(WI). Based on the WI, the ZigBee controller broadcasts the wakeup frames which contain the information pertaining to the list of clients that have packets buffered in the AP. The ZigBee radio is in CAM (Constant Awake Mode) mode.

3.4.1.2 Wakeup Scheduler

In infrastructure mode, when the clients connected to the AP are in standard PSM (SPSM), the AP has to buffer packets until the client sends an acknowledgement when it wakes up its radio and then waits to receive packets from the AP. The AP also has information about the desired delay bound d_i for each client i. When a client joins the ZPSM network a unique identifier is assigned to it. The AP keeps track of all the incoming packets and creates a wake up frame for the buffered packets. This wakeup frame contains information about the delay bound and the unique identifier of all the clients that has buffered packets in the AP. This wakeup frame is forwarded to the ZigBee controller which would broadcast this frame every wakeup interval time. If the AP receives a packet that is from a client not in the membership table, the client information is ignored from the wakeup frame as the client does not belong to the ZPSM network. In this way, the control messages are send over ZigBee periodically based on which the client decides to wake up its WiFi radio hence saving lot of energy while meeting the delay requirements as well. Each time the wakeup frame is send to the client the AP updates the delay by subtracting the time elapsed since the packet arrived at the AP.

3.4.1.3 Framework Configurator

When the client is in ZigBee range, there is a chance that a lot of packets can be lost as the channel is not very reliable. Hence, the wakeup interval must be chosen in a way that ensures that the client would receive the wakeup frames. This is configured by this component. The client switches to standard PSM (SPSM) when it moves out of ZigBee range. When the client is in standard PSM(SPSM), the delay bound cannot be guaranteed when the delay desired by the client is less than two beacon intervals. Hence, the wakeup schedule of the WiFi radio at the client end must be modified. This is done by changing the listen interval of the client.

When the listen interval is less than the default listen interval value of 100ms, then the desired delay bound can be achieved even when the client is outside the ZPSM network. This value of listen interval (LI) is also configured by this component.

3.4.2 Client

3.4.2.1 ZigBee Controller

On the client side, the ZigBee controller receives the wakeup frames send by the access point. Each client receiving the wakeup frames, tries to match its unique authentication identifier with the one in the list in the wakeup frame. In case of a match, it extracts the delay in the frame. And then the client would wakeup the WiFi radio once this delay is reached. The standard PSM (SPSM) wakeup happens independently of the wakeup triggered by the proposed system. Hence, the delay bound is always better than the desired delay bound of the client. The duty cycle of the Zigbee radio on the client side is chosen such that it is not always on so that energy is not wasted when no wakeup frames are intended for the client. Hence energy is saved at the Zigbee radio as well as the WiFi radio.

3.4.2.2 LI Configurator

This component on the client side calculates the dynamic listen interval (DLI) and sets it so that the WiFi radio at the client side is turned on periodically based on the Dynamic Listen Interval (DLI). The DLI is calculated based on a formula we have devised. We have defined the channel quality, p to be the percentage of packets send by the AP that successfully reached the client. When the channel quality is ideal which is when p = 1, the Dynamic Listen Interval can be ignored. But another parameter called a heartbeat would be send every two minutes to let the AP know if the client is within zigbee range or not within zigbee range. This data is stored in the membership table of the AP based on which clients are marked within zigbee range or outside the range. But if the channel quality is between 0 and 1, then the Dynamic listen interval has to be calculated. The formulae used in our implementation is $DLI = \frac{D}{1-\frac{P\times D}{WI}}$. Here, D is the delay bound, P is the channel quality and WI is the Wakeup Interval. In this formulae, $\frac{D}{WI}$ denotes the number of packets that can be delivered u8nder current ZigBee channel quality. Multiplying this by P computes the expected number of packets delivered within the delay bound, D. If this value is more than 1, then the channel quality is ideal and the Dynamic Listen interval parameter is not computed. This formula follows from this $\frac{D}{DLI} + \frac{D}{WI} \times P \geq 1$ which means that the probability that a packet arrives at the receiver within the delay bound D is 1. So if the channel quality is not ideal and not all wakeup frames are received by the ZigBee interface then it should be accounted for by the proactive wakeups on the WiFi radio based on the DLI.

CHAPTER 4. IMPLEMENTATION

4.1 Hardware Environment

4.1.1 Overview

In order to implement the design presented in Chapter 3 which has been adapted from [Qin and Zhang (2013)], hardware and software components were used. A Pandaboard is used instead of a smartphone. The Pandaboard runs the Ubuntu Precise operating system which runs the linux kernel 3.2. The Pandaboard has an integrated WiFi chip built into it and for the ZigBee radio, a TelosB mote is connected to the Pandaboard through the USB port.The TelosB mote runs the Tiny OS operating system. These are the hardware components used for the implementation. As far as the software components are concerned, the TinyOS application called BaseStation which is part of the TinyOS SDK has been used. In order to send messages from the Pandaboard to the mote using serial port, the serial forwarder application which is also part of the TinyOS SDK has been adapted to work along with our implementation. The software components involve the use of linux kernel modules as well as linux user space application. Also, the incoming packets entering the wireless LAN interface are intercepted as part of the implementation, hence our implementation makes use of the netfilter framework as well. Each of these components shall be explained to some detail in the sections in this chapter.

4.1.2 Pandaboard

The implementation has been tested on Pandaboard ES which is a low cost, low power single-board development platform based on the Texas Instruments OMAP4460 system on a chip (SoC). It has a Dual-Core ARM Cortex A-9 MOCore with Symmetric Multiprocessing with a speed of upto 1.2GHz, an SGX540 graphics core that delivers better performance compared to SGC530 core, low power audio, DSI Support, LCD Expansion header. It has an HDMI v1.3(Type A) and DVI-D display ports, an audio in/out and two USB 2.0 ports. It comes with a 1GB low power DDR2 RAM. It also supports persistent storage through SD/MMC card upto 32GB. Apart from this, it has ethernet as well as wireless and bluetooth connectivity [pandaboard.org (2014b)]. Both Bluetooth and WiFi is based on based on WiLink 6.0 through a TiWi chipset on the Pandaboard [LSR (2014)].

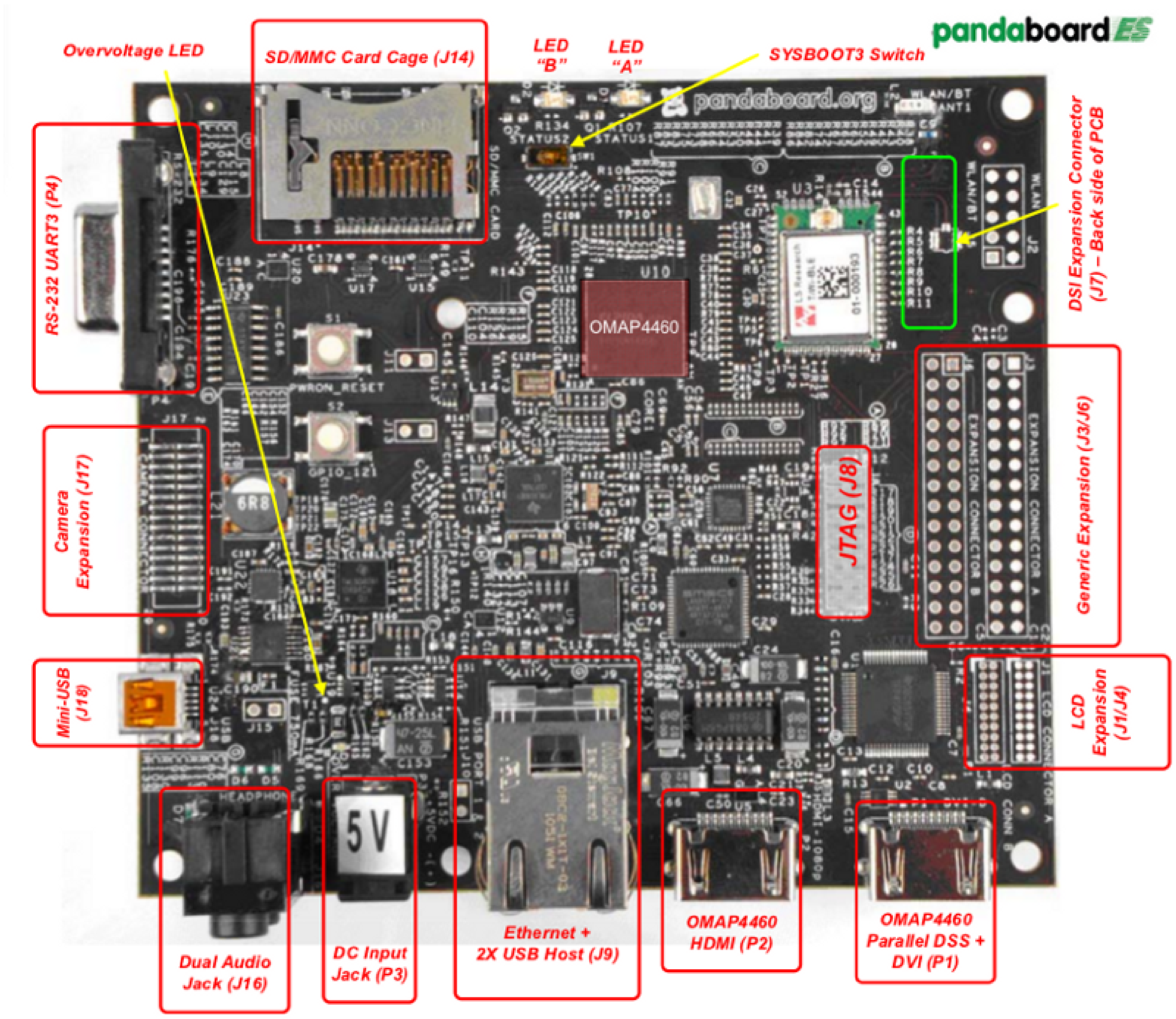


Figure 4.1: Pandaboard

[pandaboard.org (2014a)]

4.1.3 TelosB mote and ZigBee

Wireless sensor networks are very popular in applications in which the sensor nodes are used to monitor the physical environment in which it is deployed. A mote is the basic unit of a wireless sensor network. A mote is used by our implementation as it has an integrated ZigBee radio module residing on it. The parts of a TelosB mote is shown in Fig 4.2. The Crossbow's TelosB mote (TPR2400) is an open source platform developed by UC Berkeley for the research community. It features an IEEE 802.15.4/ZigBee radio with an integrated antenna and also USB ports for programming and data collection. It consumes low power and runs at a 250kbps data rate. The mote runs TinyOS operating system [CrossbowTechnologyInc (2014)].

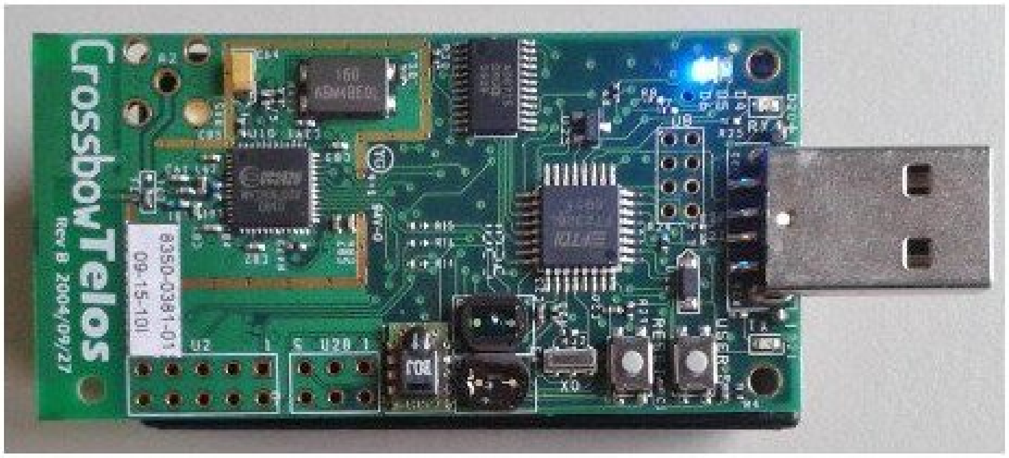


Figure 4.2: Telosb

[UCBerkeley (2014)]

4.1.4 TinyOS

TinyOS is an embedded operating system designed for low power wireless devices like wireless sensor networks. It is an open source operating under the BSD license. It was collaboration between the University of California, Berkeley in co-operation with Intel Research and Crossbow Technology and has grown into the TinyOS Alliance. TinyOS applications are written in the NesC programming language which is a dialect of C but optimized for low memory devices like sensors [Wikipedia (2006)]. TinyOs applications are non-blocking and they are compiled into a single binary image. So it runs as a single stack and the single application has full control of the hardware. These applications are made of components. A component uses and provides interfaces. Interfaces represent the functionality that the component provides. They specify a set of commands that the component implements as part of its functionality and also a set of events that are functions to be implemented by the user that uses the interface. Components are of two types: modules that provide implementations of one or more interfaces and configuration that wire components and interfaces together. Each nesC application is described by a top-level configuration. [stanford.edu (2013)]

4.2 Software Environment

4.2.1 Netlink

Netlink is a specialized socket for interprocess communication (IPC) between the user space and kernel processes. This technique comes in very useful when developers have to expose the features that are implemented by a kernel process in the user space.

It is implemented as a protocol with its own address family called AF_NETLINK. It is usually created using

```
netlink_socket = socket(AF_NETLINK, socket_type, netlink_family);
```

Netlink sockets are accessed like any other sockets-send and recv calls are used like other sockets.

```
ssize_t sendmsg(int sockfd, const struct msghdr *msg, int flags);
ssize_t recvmsg(int sockfd, struct msghdr *msg, int flags);
```

The messages send through these sockets have a particular format. The message has a message header and message data.

4.2.2 NetFilter

The netfilter architectures offers packet filtering functionality by loading a netfilter kernel module. This is made possible by inserting hooks in the dynamic routing function in the netfilter kernel module [Wehrle et al. (2002)]. The packet filtering architecture is illustrated in Fig 4.3. The following hooks are defined in the netlink architecture that routing code can hook onto.

- NF_IP_PRE_ROUTING: The incoming packets pass this hook before it goes to the routing code.
- NF_IP_LOCAL_IN: All incoming packets addressed to the local computer pass this hook.
- NF_IP_FORWARD: All incoming packets not addressed to the local computer pass this hook which are packets to be forwarded and leaving the computer over a different network interface.
- NF_IP_LOCAL_OUT: All outgoing packets generated in the local computer pass this hook.
- NF_IP_POST_ROUTING: This hooks provides last access to all outgoing packets before leaving the computer's network interface.

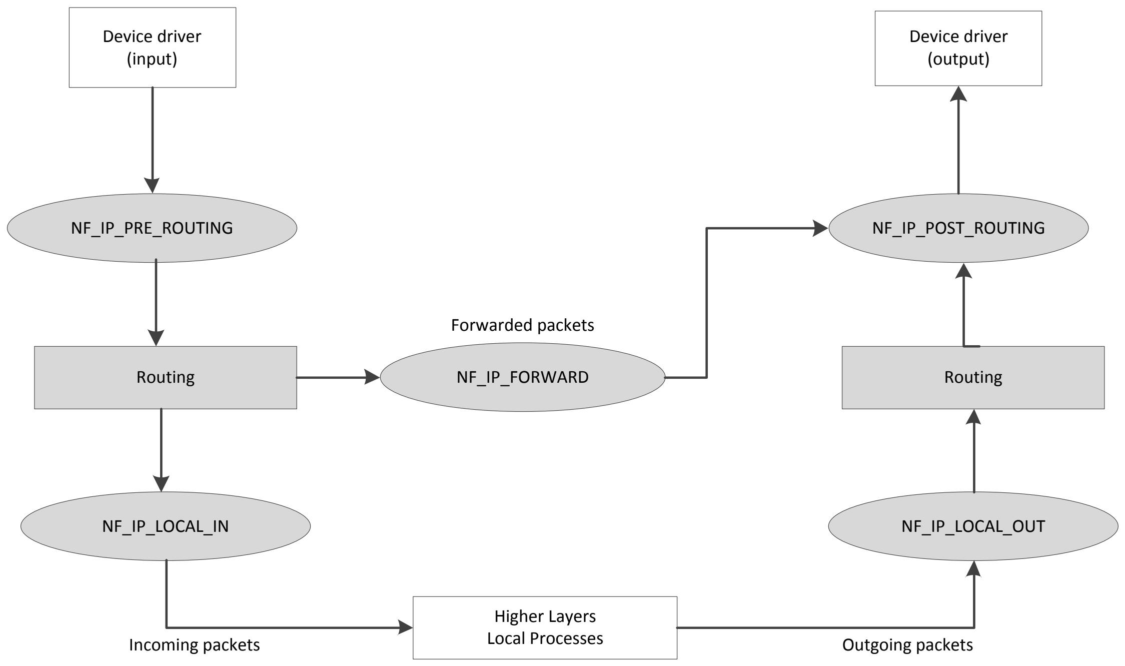


Figure 4.3: Netfilter Architecture [Wehrle et al. (2002)]

4.2.2.1 Registering and Unregistering Packet-Filter Functions

Calling `NF_HOOK()` macro defined in \include\linux\netfilter.h causes the routing code to process the filter functions hooked into a netfilter hook. The packet filter functions that are actually hooked into the netfilter hooks are so called `hook functions` of the type `nf_hookfn` defined in \include\linux\netfilter.h. The signature of this function is as follows:

```
typedef unsigned int nf_hookfn( unsigned int hooknum,
struct sk_buff **skb,
const struct net_device *in,
const struct net_device *out,
int (*okfn) (struct sk_buff *));
```

The return value of this packet-filter function specifies what should happen to the packet filter and can take the following values:

- NF_DROP: The packet is dropped after the active list processing is dropped.
- NF_ACCEPT: The packet goes onto the next packet filter function in the rules lists and then released for further processing.
- NF_STOLEN: The packet filter function withhold the packet and the list processing is stopped.
- NF_QUEUE: The packet is put to the queue for a user-space application to process it.
- NF_REPEAT: As opposed to NF_ACCEPT, the current filter function is invoked again.

In order to register and unregister the hook, the following functions are used:

```
nf_register_hook()
nf_register_unhook()
```

In addition, a structure is required called `struct nf_hook_ops` that holds the information required and can be found in `netfilter.h` [Wehrle et al. (2002)].

4.2.3 Hostapd

hostapd is a user space daemon for access point and authentication servers. It implements the IEEE 802.11 access point management that is used to connect the clients to the access point (AP). The Linux drivers for the TiWi wireless LAN chipset are based on the mac80211 driver architecture and hostapd supports this class of drivers. The Pandaboard that acts as the access point runs hostapd so that clients can connect to it [Malinen (2013)]

4.3 Software Architecture

The system implementation has a client side as well as an access point side.

4.3.1 Zigbee Power Saving Mechanism - Access Point (ZPSM-AP)

The software architecture for access point side of ZPSM is shown in Fig 4.4.

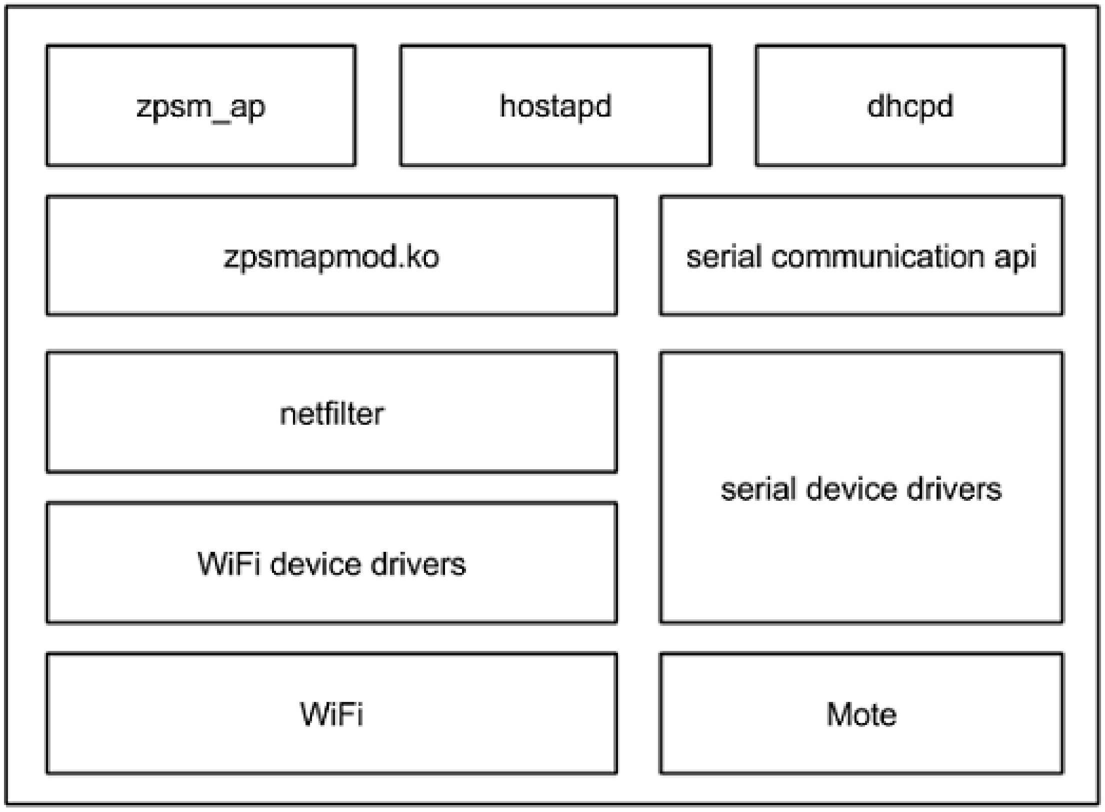


Figure 4.4: Software Architecture of ZPSM-AP

In order to serve as an access point (AP), the pandaboard has to host one which is done by hostapd daemon. Hostapd is capable of assigning a Service Set Identifier (SSID) to our network. This is done by using a configuration file hostapd.conf. When a client requests access to this AP, the hostapd assigns an association id but the client should also be assigned an Internet Protocol (IP) address. Inorder to do this, we need to have a dhcp server running on the pandaboard as well. This dhcp server also has a configuration file in which we can provide the class type for the IP address we wish to use depending on the class of our network.

The ZPSM access point implementation comprise of the kernel module called the zpsmapmod.ko and the user space application called zpsm_ap.

The kernel module zpsmapmod.ko takes care of membership management of the ZPSM network clients by assigning an index to a client that requests access to the network. It also makes sure that indexes are reused for clients that were disconnected temporarily from the network. It also periodically clears the membership table to ensure that clients that are not active for a very long period of time is removed. The linux kernel module timer is more accurate so the wakeup frame timer is implemented here. The kernel module is able to receive the packets from clients by hooking to the netfilter framework. This is how the membership management is done. The user side of ZPSM AP zpsm_ap interfaces with the serial port through which the mote is connected. This application sends the wakeup frame messages to the mote from the pandaboard so that it can be broadcasted through the ZigBee radio. This must be done every wakeup interval which has been set to 40ms. Once the timer expires, the kernel module sends a request to the user application. This request is send using the netlink socket interface that acts as a bridge between the user space application and kernel module of the ZPSM AP implementation. The expiration of the timer on the kernel side every wakeup interval triggers a request to the user application. Along with this request, the client index list from the membership table is also send to the user soace application. The application then creates a wakeup frame by inserting this list of indexes as well as the remaining delay bound for the client. Since the user side application can interact with the mote using serial communication, the wakeup frame is send to the mote which broadcasts the wakeup frame.
The standard power saving mode (SPSM) is disabled in the WiFi drivers. And also the ZigBee radio is in always ON mode on the Pandaboard and mote that comprise the physical ZPSM access point (ZPSM-AP) setup.

4.3.2 Zigbee Power Saving Mechanism - Client (ZPSM-Client)

The software architecture for access point side of ZPSM is shown in Fig 4.5.

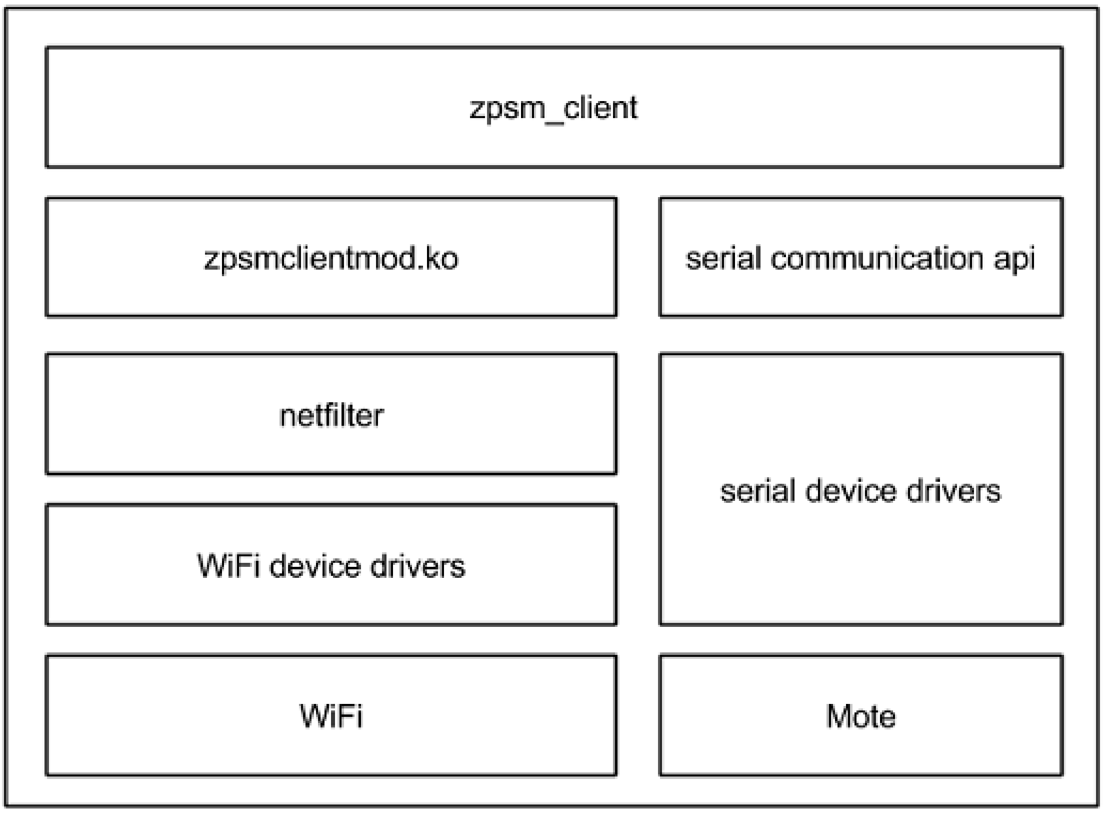


Figure 4.5: Software Architecture of ZPSM-Client

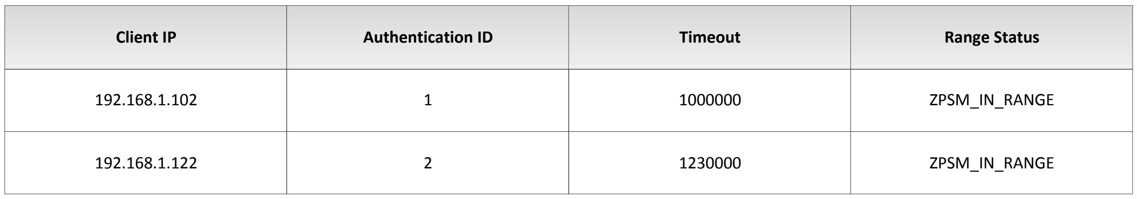
The client that is part of the ZPSM network must have a kernel module and a user space application running as well. The linux kernel module zpsmclientmod.ko captures packets from the access point (AP). This is used when establishing the session with the ZPSM AP. The ZPSM AP sends a unique index to the client and the client records this in the kernel module and shares it with the userspace application. In order to do this, the kernel module hooks to the netfilter framework. Apart from this, since the kernel module captures the system clock more accurately, the timer is placed here. In a ZPSM client, in order to make sure an entry exists in the membership table of the ZPSM AP, the client has to periodically send a heartbeat to the ZPSM AP. This is done by sending a dummy packet to the ZPSM AP over WiFi so that the ZPSM AP doesn't remove its entry. Also, when there is a poor ZigBee channel quality resulting in high packet loss, the WiFi radio has to be woken up based on a Listen Interval (LI) that depends on the application delay bound and the packet loss ratio. The user side of the ZPSM client, zpsm_client handles the incoming wakeup frame from the mote. This is using serial communication between the user application on the Pandaboard and the mote. On receiving the wakeup frame the user application checks if its index is in the wakeup frame and then starts a timer corresponding to the delay in the wakeup frame. The user application calculates the listen interval and sends it to the kernel module. This is done using the netlink socket mechanism which acts as a bridge between the user application and the kernel module. The listen interval computation can be performed in the application side as it involves floating point operations which shouldn't be done in the kernel space.

The standard power saving mode (SPSM) is enabled in the WiFi drivers. And also the ZigBee radio is not in always ON mode on the pandaboard and mote that comprise the physical ZPSM client (ZPSM-Client) setup.

4.4 ZPSM Data Structures

4.4.1 Membership table in ZPSM-AP

The information of the clients authenticated by the ZPSM-AP resides in the membership table that has a structure shown in Fig 4.6



Client IP	Authentication ID	Timeout	Range Status
192.168.1.102	1	1000000	ZPSM_IN_RANGE
192.168.1.122	2	1230000	ZPSM_IN_RANGE

Figure 4.6: Membership Table in ZPSM AP

Each client that is assigned an IP address by the access point has an entry in the membership table. When the client becomes part of the ZPSM network it is assigned an index or authentication identifier. The timeout field captures the time at which a packet was received from a client which basically tracks how recently active the client has been. The range status denotes the status of a client if it is within the ZigBee range or if it is only in WiFi range.

The code listing for the structure corresponding to the membership table can be found in the Appendix C.1.

4.4.2 Wakeup Frame

The packets that are send from the pandaboard to the library that handles the serial communication with the mote follows the format in Fig 4.7.

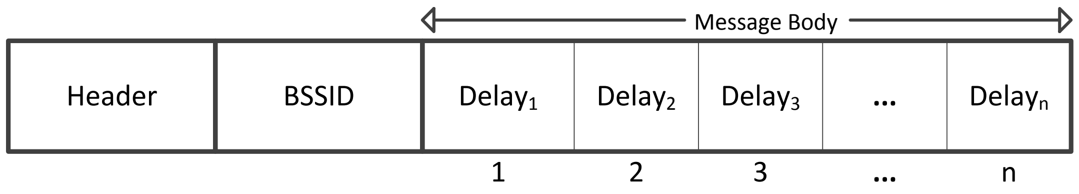


Figure 4.7: Wakeup Frame Message Format

Here the header flag which is 1 byte would denote the message header. In the implementation we have only had one scenario for the packet header which is the header that denotes that the message is a wakeup frame. It is followed by the Base Service Set Identifier (BSSID) which identifies the ZPSM access point to which the client is connected to. The remaining bytes that follow is fixed and this depends on the number of bytes that the mote can contain in its message body. The bytes in the message body indicate the remaining delay counter for the client which has the authentication identifier mapped to its index. On the client side, the serial communication library returns this information in this format.

The code listing of the structure used for wakeup frame can be found in Appendix C.2.

4.5 ZPSM Detailed Design

4.5.1 ZPSM AP Detailed Design

The components that make up the ZPSM AP is shown in Fig 4.8.

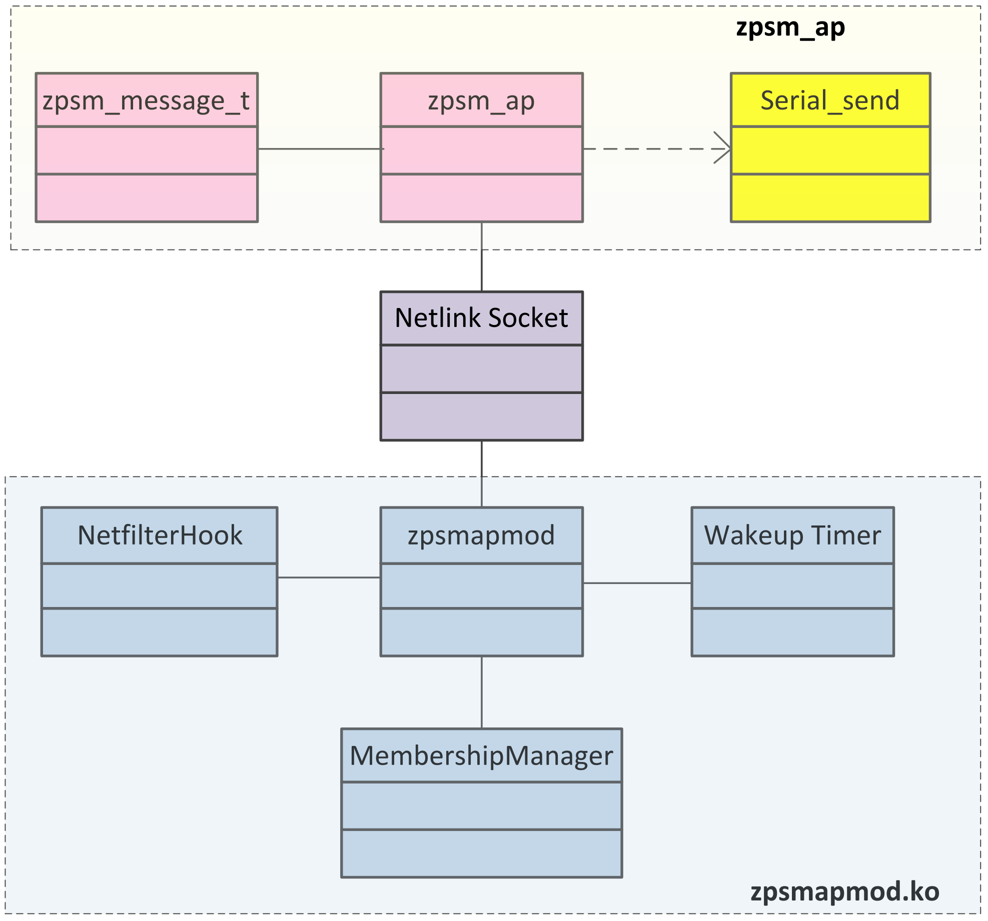


Figure 4.8: ZPSM AP Detailed Design

The ZPSM AP has a user application side zpsm_ap and a kernel module zpsmapmod.ko. They communicate by means of a netlink socket.

The zpsm_ap which is the user space application forms wakeup frames which follows the format discussed in Section 4.4.2. It also depends on SerialSend module [tinyos.net (2007b)]. The interface to this module can be found in code listing in Appendix C.3. This interface is called by the user space application zpsm_ap to send a message which in this case is the wakeup frame to the mote.

The kernel side modules zpsmapmod.ko hooks onto the netfilter frameworks to intercept all packets that arrives at the ZPSM AP. It also interacts with the membership manager module which maintains the membership table described in Section 4.4.1. This is filled based on the packet information that is intercepted by the netfilter hook. The functionality of the membership manager is listed as an interface in Appendix C.5. The Wakeup Timer is a kernel timer that triggers an event every wakeup interval which is 40ms and passes the client index list to the user space application through the netlink socket.

4.5.2 ZPSM Client Detailed Design

The components that make up the ZPSM Client is shown in Fig 4.9. The ZPSM Client has a user application side zpsm_client and a kernel module zpsmclientmod.ko. They communicate by means of a netlink socket.

The zpsm_client which is the user space application listens to incoming wakeup frames which follows the format discussed in Section 4.4.2. It also depends on SerialListen module [tinyos.net (2007b)]. The interface to this module can be found in code listing in Appendix C.4. The client application has a timer to wakeup the WiFi radio only when the delay bound of the packet is reached. This is done by using Linux Sockets by sending a dummy packet to ZPSM AP.

The kernel side modules zpsmclientmod.ko hooks onto the netfilter frameworks to intercept all packets that arrives at the ZPSM Client at a particular port as it should receive only control messages from the ZPSM AP. It also has a heartbeat timer the timeout value is calculated by the user application zpsm_client and send to kernel through the netlink socket. This timeout value is the dynamic listen interval computed by zpsm_client.

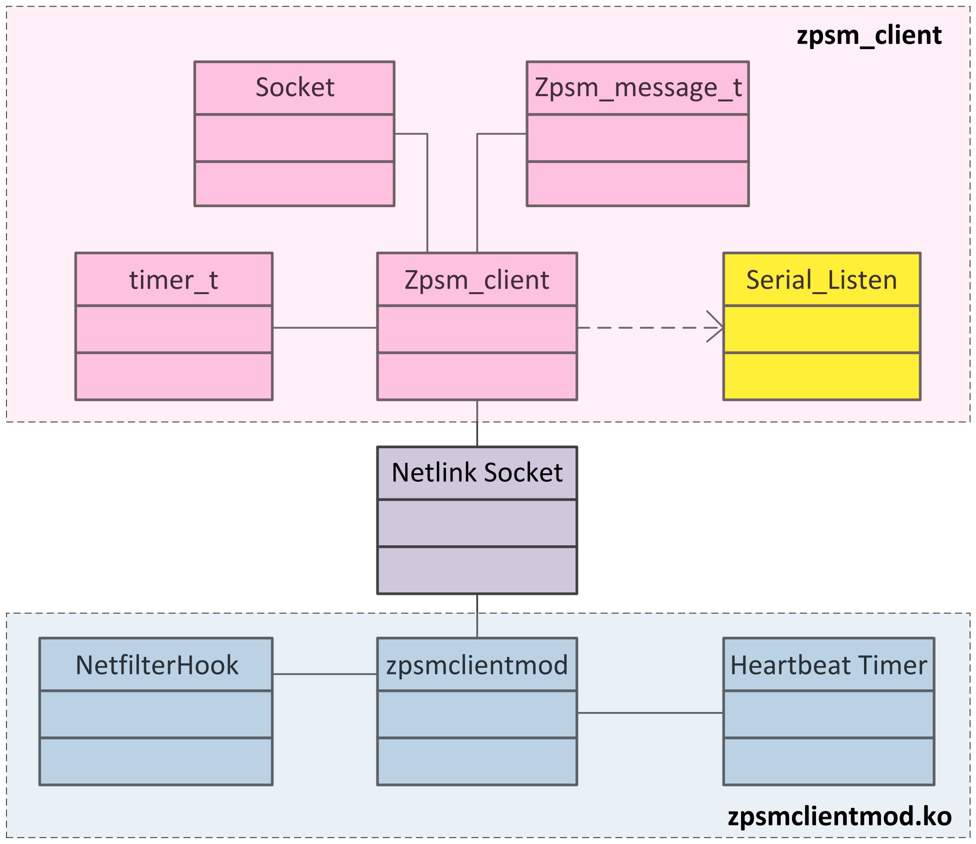


Figure 4.9: ZPSM Client Detailed Design

4.5.3 BaseStation Detailed Design

The BaseStationApp from the TinyOS SDK has been used [tinyos.net (2007a)]. On the ZPSM client however, there is no need for the radio to be ON always. So some code changes in the original BaseStation code was made to implement the duty cycle in mote. The code sets the ZigBee radio to ON for 20ms and OFF after that for 30ms. This happens periodically. The time interval for ON/OFF was chosen as the wakeup interval is 40ms.

The Makefile was modified in the original BaseStation so that both ZPSM AP and ZPSM Client listen in a specific radio channel. This was done by adding the following line in the

BaseStation makefile.

```
PFLAGS += -DCC2420_DEF_CHANNEL=5
```

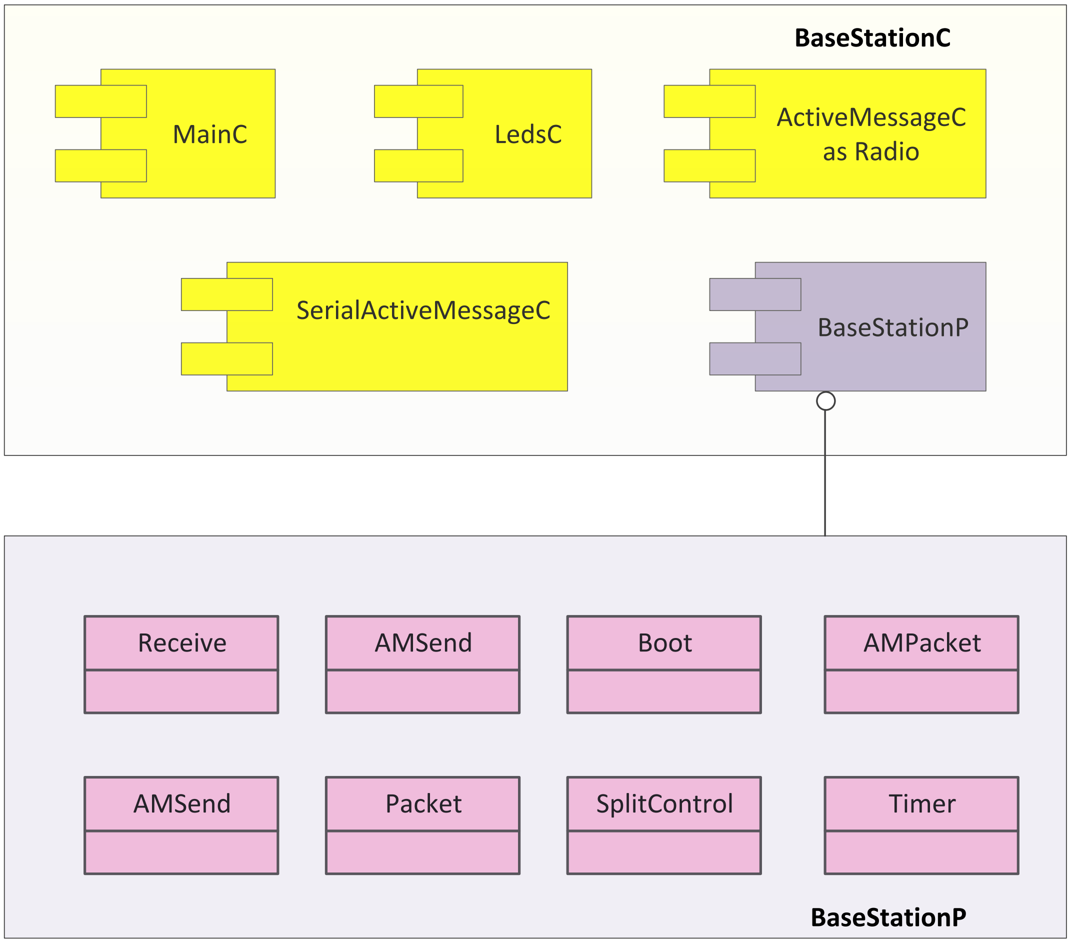


Figure 4.10: BaseStation with duty cycle

tinyos.net (2007a)

The components of the BaseStation on the client side are shown in Fig 4.10. The BaseStationC which is the configuration file has several components as shown. The ActiveMessageC components is the one that is the Radio interface and the SerialActiveMessageC is the components that provides interfaces to transfer information between the pandaboard and mote over the serial port. The BaseStationP uses interfaces provided by the components it wires itself to. The Boot interface provides the event that triggers the boot sequence of the mote. The SplitControl provides interfaces that triggers the tasks that performs the serial communication and radio transfer. The radio component can therefore be controlled by setting it on and off through the time interface periodically. The AMSend interface is used for the serial send and radio send tasks and the Receive interface for receiving data from serial port and intercepting radio data received by the ZigBee radio.

4.6 ZPSM Scenario Control Flow

4.6.1 Handling Control Messages at ZPSM AP

Fig 4.11 illustrates the sequence of instructions to handle the membership of clients by ZPSM AP. When the ZPSM AP kernel module zpsmapmod.ko is loaded, it establishes a netfilter hook that intercepts all packets that is sniffed by the WiFi radio.

If the packet intercepted is an outgoing packet from the ZPSM access point, then it must be meant for a client in the ZPSM network. So the IP address of the packet is matched with the entries in the membership table. If there is an entry in the membership table against the IP address then an index is assigned to it and also the delay counter is updated. The delay bound is assumed to be a multiple of the wakeup interval so that the information passed through radio carries only the delay counter which consumes less memory. If no entry exists in the membership table, an entry is created for the IP address, an index is assigned, the timeout is set to the current time and the out of range status to true which means that the client is out of ZigBee range.

If the netfilter hook receives an incoming packet from the radio interface at port 6789 then the packet contains control message. The control header is extracted from the packet. The control header could be one of the following based on which appropriate action is taken by the kernel module.

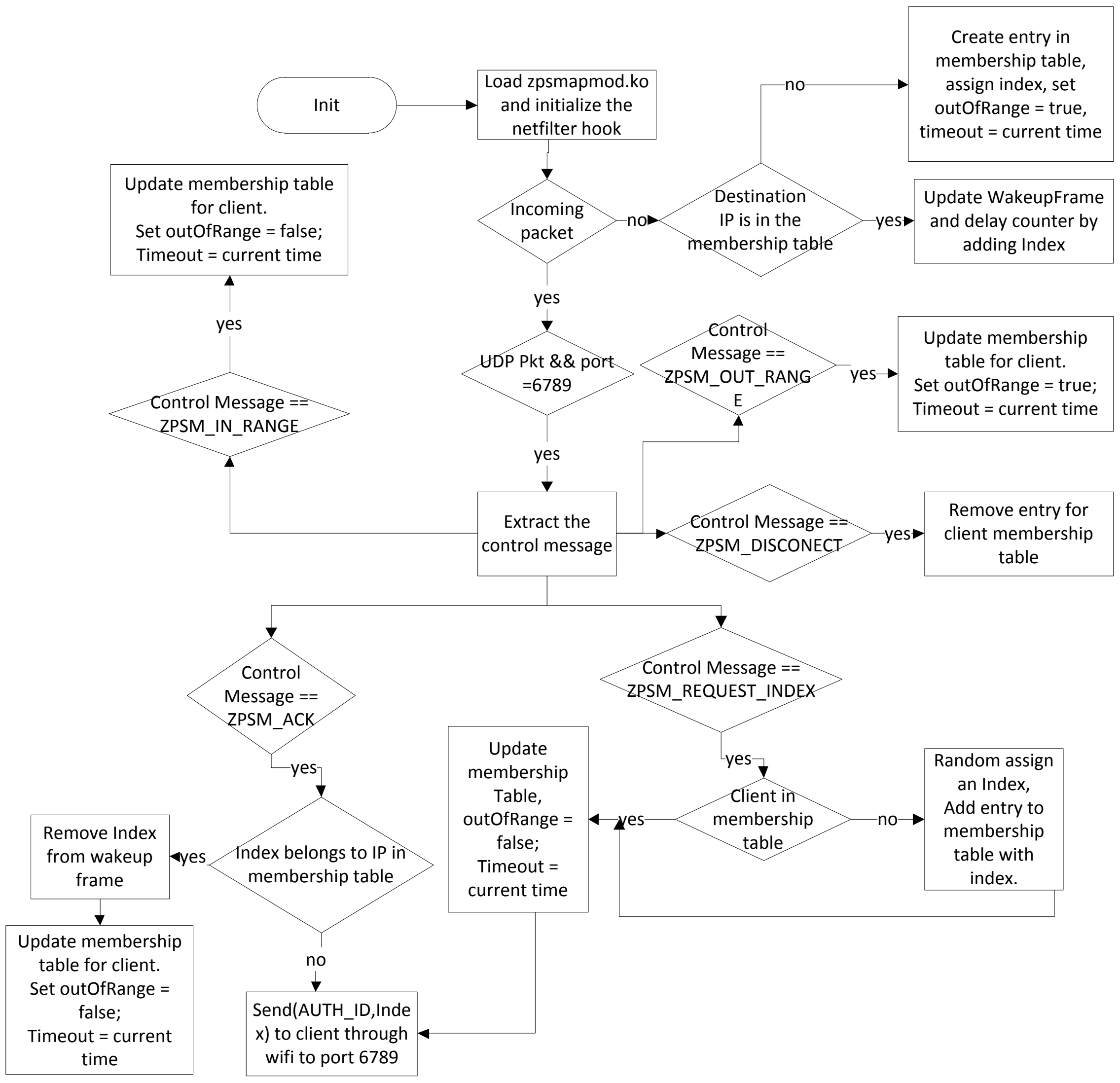

Init
Load zpsmapmod.ko and initialize the netfilter hook
Incoming packet
no
Destination IP is in the membership table
no
Create entry in membership table, assign index, set outOfRange = true, timeout = current time
yes
Update WakeupFrame and delay counter by adding Index
yes
UDP Pkt && port =6789
yes
Extract the control message
Control Message == ZPSM_IN_RANGE
yes
Update membership table for client. Set outOfRange = false; Timeout = current time
Control Message == ZPSM_OUT_RANGE
yes
Update membership table for client. Set outOfRange = true; Timeout = current time
Control Message == ZPSM_DISCONECT
yes
Remove entry for client membership table
Control Message == ZPSM_ACK
yes
Index belongs to IP in membership table
yes
Remove Index from wakeup frame
Update membership table for client. Set outOfRange = false; Timeout = current time
no
Send(AUTH_ID,Index) to client through wifi to port 6789
Control Message == ZPSM_REQUEST_INDEX
yes
Client in membership table
yes
Update membership Table, outOfRange = false; Timeout = current time
no
Random assign an Index, Add entry to membership table with index.

Figure 4.11: Membership Management and Control Message Handling Sequence

- ZPSM_ACK: If the IP address of the sender exists in the membership table then remove the index from the wakeup frame. Update the entry against this IP address with timeout as current time and the out of range status to false. If the entry does not exist for the client then an entry is created and the authentication identifier or index is send to this IP address of the client through port 6789.

- ZPSM_REQUEST_INDEX: If the ZPSM AP receives this control message from a packet with an IP address which is in its membership table, then it updates the information against it by setting the timeout to current time and out of range flag to false and sends the index to the requesting IP address. If it does not have an entry for the requesting IP address then it adds an entry to the membership table and updates the out of range status and timeout and sends the authentication identifier or index to the requesting client.
- ZPSM_DISCONNECT: When this control message is received by ZPSM AP, it removes entry against the IP address than made the request.
- ZPSM_OUT_RANGE: When this control message is received by ZPSM AP, it updates the entry against the IP address than made the request by setting the timeout to current time and out of range flag to true.
- ZPSM_IN_RANGE: When this control message is received by ZPSM AP, it updates the entry against the IP address than made the request by setting the timeout to current time and out of range flag to false.

4.7 Wakeup frame generation at ZPSM AP

Fig 4.12 illustrates how the wakeup frames are generated by ZPSM AP. Once the ZPSM AP kernel module is loaded, the wakeup frame timer is initialized with a timeout of 40ms which is the wakeup interval. The user side application is also loaded and the netlink socket connection is established between the user side and kernel side components. Each time the wakeup frame timer expires, it sends the wakeup frame to the application space through netlink. Once the application side receives it, it broadcasts it through the mote. Also at the kernel side, each time the timer expires, once five minutes have expired it attempts to clean the entries in the membership table by checking if the timeout value for the entry is more than ten minutes.

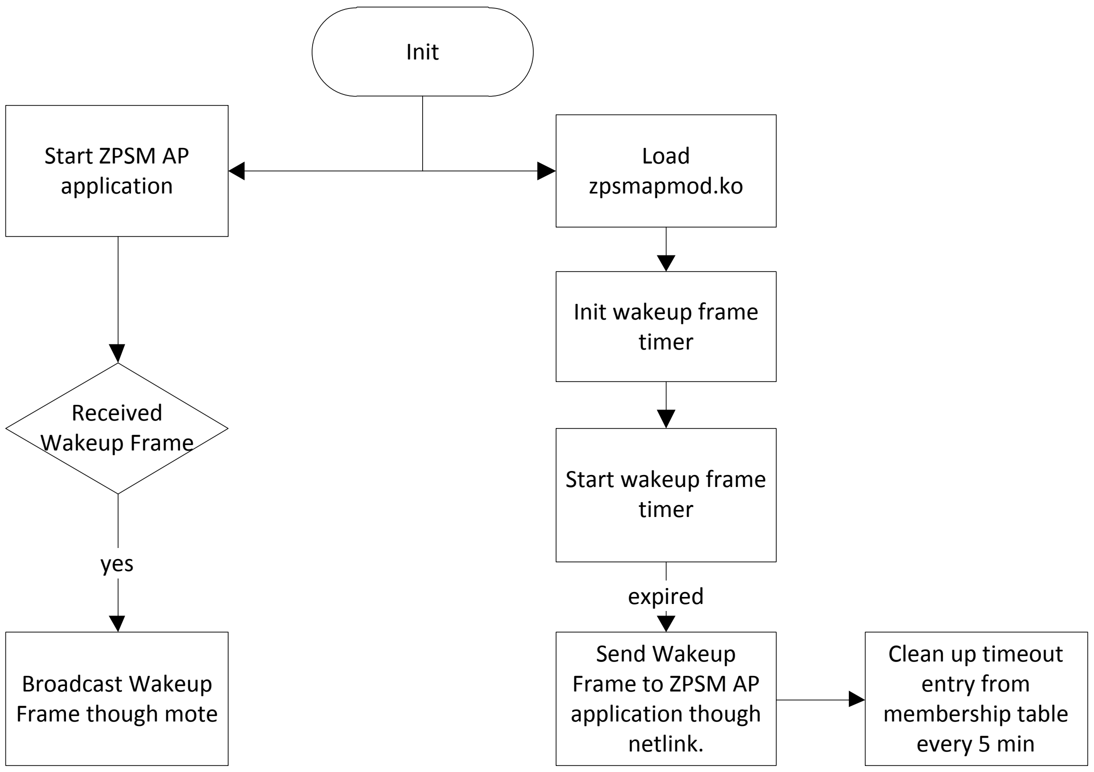

Init
Start ZPSM AP application
Load zpsmapmod.ko
Init wakeup frame timer
Received Wakeup Frame
Start wakeup frame timer
yes
expired
Broadcast Wakeup Frame though mote
Send Wakeup Frame to ZPSM AP application though netlink.
Clean up timeout entry from membership table every 5 min

Figure 4.12: Wakeup Frame Generation Sequence

4.8 ZPSM Client Authentication and Waking up WiFi Radio

Fig 4.13 shows what action the ZPSM client takes once it receives wakeup frame and how the WiFi radio is woken up.

The ZPSM client application is loaded and also the kernel module and a netlink socket connection is established between them. The application initialized the client index and its status as ZPSM_OUT_RANGE. The ZPSM client application starts to listen for incoming packets from the mote. It check if the wakeup frame received has the correct Basic Service Set Identifier (BSSID) for the ZPSM network. If not it is rejected. Else the state of the client is set to ZPSM_IN_RANGE and the wakeup frame counter is incremented.

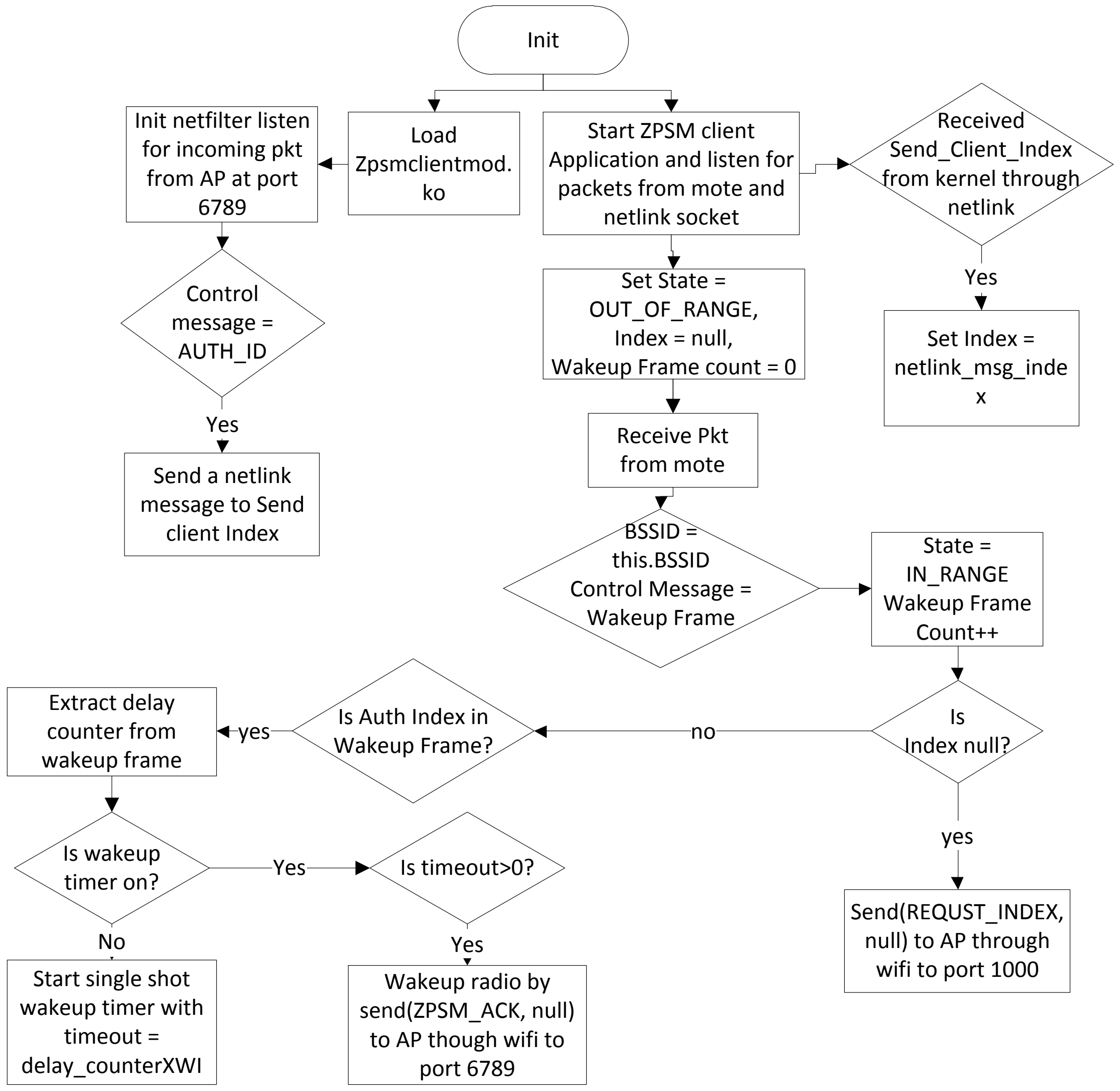

Init
Load Zpsmclientmod.ko
Init netfilter listen for incoming pkt from AP at port 6789
Start ZPSM client Application and listen for packets from mote and netlink socket
Received Send_Client_Index from kernel through netlink
Yes
Set Index = netlink_msg_index
Control message = AUTH_ID
Yes
Send a netlink message to Send client Index
Set State = OUT_OF_RANGE, Index = null, Wakeup Frame count = 0
Receive Pkt from mote
BSSID = this.BSSID Control Message = Wakeup Frame
State = IN_RANGE Wakeup Frame Count++
Is Index null?
no
yes
Send(REQUST_INDEX, null) to AP through wifi to port 1000
Is Auth Index in Wakeup Frame?
yes
Extract delay counter from wakeup frame
Is wakeup timer on?
Yes
No
Is timeout>0?
Yes
Start single shot wakeup timer with timeout = delay_counterXWI
Wakeup radio by send(ZPSM_ACK, null) to AP though wifi to port 6789

Figure 4.13: Authentication and Radio Wakeup Sequence

At this point, if it doesn't have a valid non-zero index, then it requests the ZPSM AP for one. If it is able to find its index in the wakeup frame received, then it extracts the delay counter from the wakeup frame which is the value at the index of the array of data bytes in the body of the wakeup frame. And then it checks if the wakeup frame timer has been started or not. If it has not been started, then it starts the timer with a timeout value delay_counter×WI where WI is the Wakeup Interval which is 40ms. This timer is a single shot timer and once it expires, it wakes up the WiFi radio by sending an acknowledgement back to ZPSM AP. As the Standard Power Saving Mechanism (SPSM) is still active, the radio turned on every 200ms. So in order to wakeup the radio for a timer period less than this, a dummy packet should be send through the network stack.

Once the ZPSM Client kernel module is loaded, it registers a netfilter hook that intercepts control messages at port 6789. When it receives a packet from ZPSM AP with control message as AUTH_ID, it updates its authentication identifier or index by sending it to the user space application which in turn updates the index.

4.9 ZPSM Client Channel Quality Adaptation

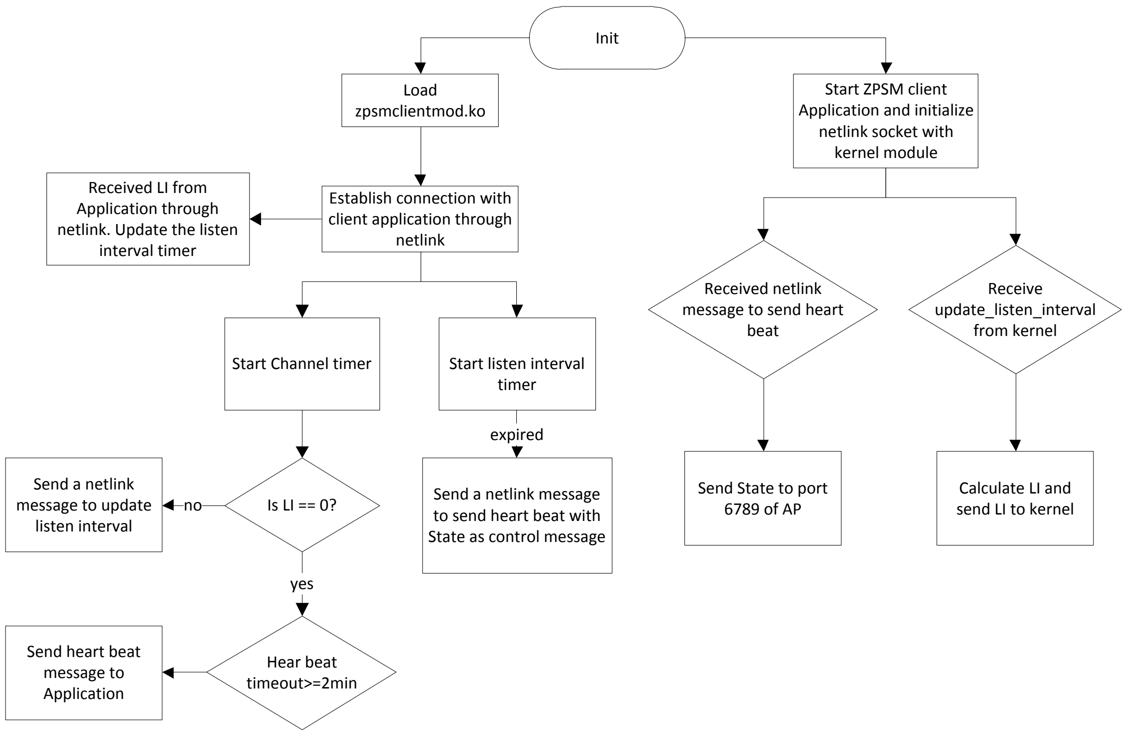

Init
Load zpsmclientmod.ko
Start ZPSM client Application and initialize netlink socket with kernel module
Received LI from Application through netlink. Update the listen interval timer
Establish connection with client application through netlink
Received netlink message to send heart beat
Receive update_listen_interval from kernel
Start Channel timer
Start listen interval timer
expired
Send a netlink message to update listen interval
no
Is LI == 0?
Send a netlink message to send heart beat with State as control message
Send State to port 6789 of AP
Calculate LI and send LI to kernel
yes
Send heart beat message to Application
Hear beat timeout>=2min

Figure 4.14: Adapting to changes in ZigBee Channel Quality Sequence

Fig 4.14 represents how the ZPSM client adapts to changes in ZigBee channel quality without hurting the performance.

The ZPSM kernel module and user application are loaded and netlink socket connection is established. Once the ZPSM client kernel module is loaded, the channel timer and listen interval timer is started. The timeout value of the channel timer is 200ms and is always active. The listen interval timer is only active when the channel quality is not optimum. Once the channel timer expires, it checks if the value of the ZPSM listen interval is not zero, in that case it sends a netlink message to the user space application to compute listen interval based on the formulae explained in Section 3.4.2.2. Then application computes it and sends it back to the kernel component through netlink socket. But if the ZPSM listen interval is zero, that means the ZigBee channel quality is optimum and the ZPSM client needs to send a heartbeat to ZPSM Ap by sending a request to the user space application. On receiving this request, the user space application sends a packet with the range status to ZPSM AP through port 6789.

CHAPTER 5. EVALUATION

5.1 Experiment Testbed

5.1.1 Overview

The experiment setup is shown in the Fig 5.1. The Pandaboard with Ubuntu 12.04 operating system installed in an 8GB SD card. The details of how the SD card was partitioned and the operating system installed is described in Appendix B. The TeloSB mote is inserted into the USB port of the Pandaboard so that the board can catch the ZigBee packets intercepted by the mote and also send ZigBee packets. The Pandboard activity is monitored by connecting it to a display using an HDMI-DVI adapter. A keyboard and mouse were connected to USB ports which was extended using a USB hub. The Pandaboard thus turned into a low power low cost computer. We used four such boards for our experiments. One of the pandaboards acted as the ZPSM Access Point and the other three as ZPSM Clients.

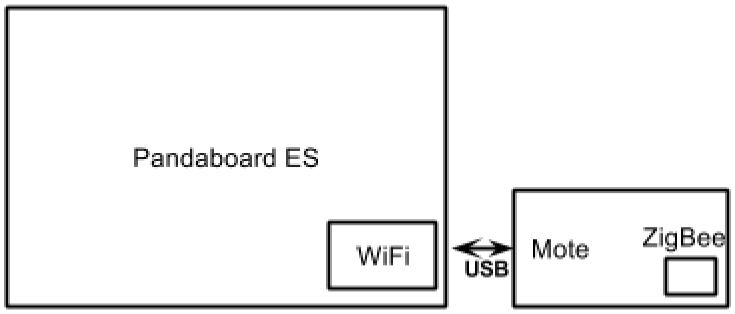


Figure 5.1: Typical setup for a AP or a client

5.1.2 Evaluation Assumptions

In order to test the implementation, we created an test application that send User Datagram Protocol (UDP) packets every 200ms. This is denoted by the symbol λ which in the test scenario is 5pkt/sec. This value has been chosen because if the packets are send too frequently the WiFi radio would switch from standard PSM (SPSM) mode to constant awake (CAM) mode. Also, the delay bound used for testing is less than 200ms as standard PSM (SPSM) wakeups every 200ms and our protocol is redundant. The WiFi channel is 8 and the ZigBee channel is 5 so that there is no interference from other WiFi and ZigBee devices. The ZPSM AP is in constant awake (CAM) mode. The energy consumption of the ZPSM client is what we are interested in. The WiFi radio is in standard PSM (SPSM) mode and the ZigBee radio is ON for 30ms and OFF for 20ms. The channel quality, P has been simulated by the test application by discarding some wakeup frames based on the target channel quality we wanted to achieve. The WiFi channel quality is assumed to be ideal in this experiment.

5.2 Evaluation metrics

In the evaluation the following metrics are measured. These are the same metrics mentioned in the simulation of the work in [Qin and Zhang (2013)].

- *Per-packet energy consumption*(mJ/pkt): This is defined as the total energy consumption including the energy consumed by WiFi radio and ZigBee radio divided by the total number of data packets transmitted.

- *Actual Delay Meet Ratio*: This is defined as the total number of packets received at the client side that meets the delay requirements divided by the total number of packets transmitted.

5.3 Energy Calculation

The energy is calculated based on the parameters mentioned in Table 5.1. The energy consumed by the WiFi radio and the ZigBee radio must be calculated based on these parameters.

Table 5.1: Parameters used to evaluate Energy Consumption [Qin and Zhang (2013)]

Network size	20 clients	Packets collected	200,000
WiFi range	120 m	ZigBee range	200 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 μs
PS-POLL	20 bytes	DIFS)	34 μs
ACK	14 bytes	Beacon interval (BI)	100 ms
Data Packet	2312 bytes	ZigBee channel 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

5.3.1 ZigBee Energy Consumption

The energy consumed by the ZigBee radio is the sum of the following two components:

- Receiving ZigBee Packet: The wakeup frame is received every wakeup interval of 40 ms. Since λ=5 pkts/s, there are 5 wakeup frames in 200 ms and this is used to compute ZigBee reception energy consumption from Table 5.1.
- ZigBee Idle Listening: The ZigBee radio needn't be turned ON always so it is in idle listening mode for 20 ms, and this is used to compute the idle listening energy consumption from Table 5.1.

5.3.2 ZigBee Energy Consumption

The energy consumed by the WiFi radio is the sum of the following components:

- Dynamic Wakeup: When the ZigBee channel quality is not ideal, the WiFi radio is woken up every ZPSM Listen Interval which is calculated by the number of times this was signaled multiplied by the energy consumed when transmitting a data packet.

- Data Packet Transmission: Apart from energy consumed on data sending, it includes the energy consumed by sending and receiving control messages, acknowledgments and PS-POLL messages. Additionally, the number of ZPSM ACKs is multiplied by the energy consumed by a data packet and added to the energy consumed when sending a data packet.

- Data Packet Reception: Apart from energy consumed on data reception, it includes the energy consumed by sending and receiving control messages, acknowledgments and PS-POLL messages. In order to calculate the transmission and reception time, the packet size is divided by the WiFi channel bit rate and then the time is multiplied by the energy consumption value in Table 5.1 to get the energy consumed.

- Idle Listening Time: The standard PSM (SPSM) idle listening time is taken into consideration when computing the energy

5.4 Evaluation Results

5.4.1 Comparison of Average Delay

Fig 5.2 depicts the average delay for different delay requirements at the ZPSM client. It can be seen that as the delay bound requirement increases the actual average delay also increases. The result shown is for the ideal channel quality of 1.0. The average delay is much lesser than the required delay because the standard PSM (SPSM) kicks in intermittently every 200ms. This is an improvement over standard PSM (SPSM).

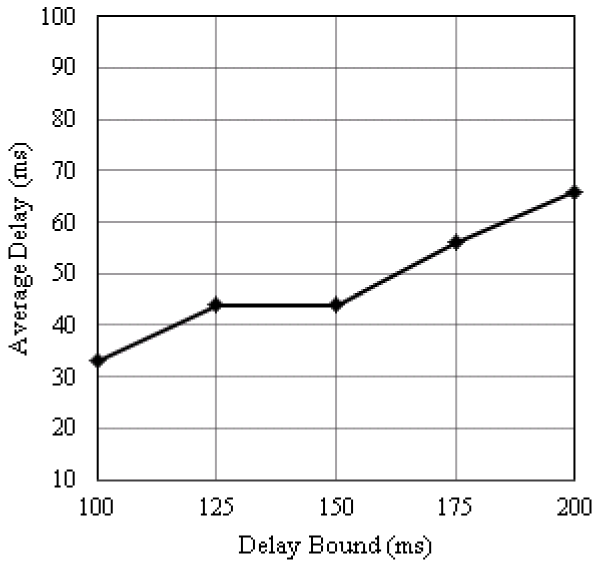


Figure 5.2: Average Delay with varying delay bound under ideal Zigbee Channel Quality

5.4.2 Comparison of Average Delay Meet Ratio

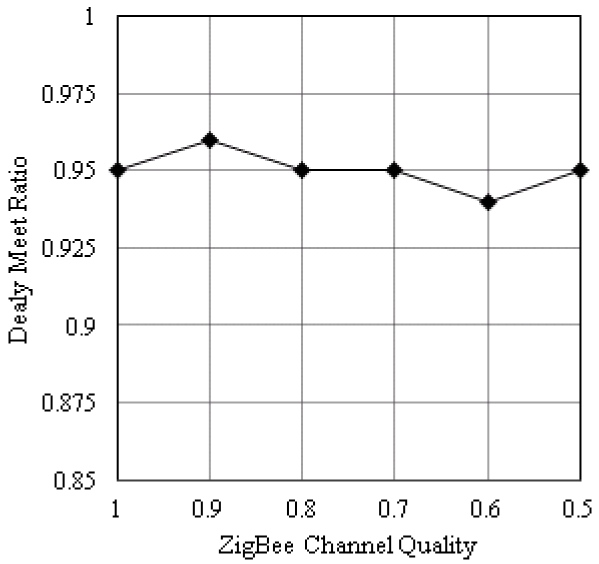


Figure 5.3: Average Delay meet ratio under varying Zigbee Channel Quality

Fig 5.3 illustrates how the average delay meet ratio changes when the ZigBee channel quality changes. The experiment was conducted with varying channel quality from 1.0 to 0.5 for delay bound requirement of 150 ms. It has been observed that even at very poor channel quality, we could achieve a delay meet radio of 0.95. This is much better than standard PSM (SPSM) that has an average delay meet ratio of 0.51.

5.4.3 Comparison of Energy Consumption

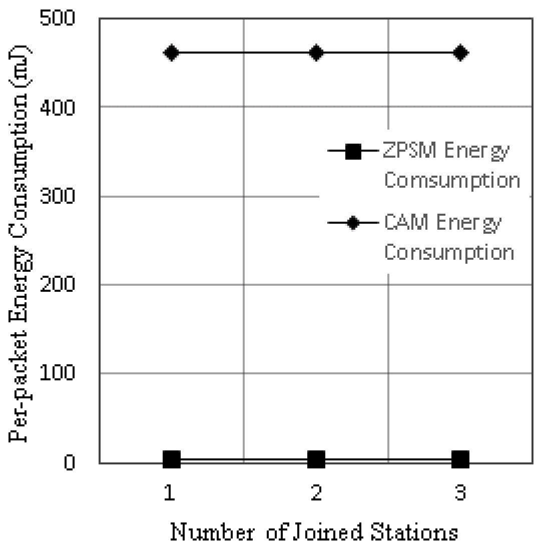


Figure 5.4: Energy consumption under ideal Zigbee Channel Quality

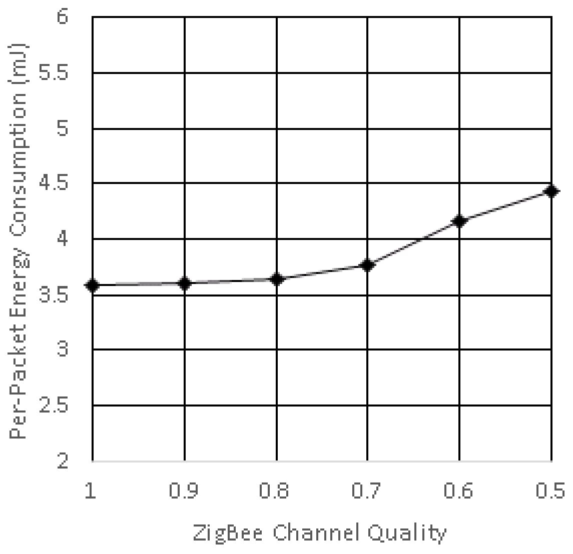


Figure 5.5: Energy consumption under varying Zigbee Channel Quality

Fig 5.4 shows the per packet energy consumption as the number of clients in the ZPSM networks changes. The ZigBee channel quality is ideal and the delay bound is 150 ms. The energy consumption is very less compared to constant awake mode and the consumption is arounf 3.5 mJ/packet irrespective of the delay bound requirement. The SPSM energy consumption is 8.25 mJ/pkt and CAM has energy consumption of 462.57 mJ/pkt.)

Fig 5.5 shows the per packet energy consumption as the channel quality of ZigBee link changes. The delay bound is 150 ms. The energy consumption increases as the link quality changes. This is because more packets are send due to dynamic listen interval.

CHAPTER 6. CONCLUSION

In this work we have proposed an implementation for a ZigBee assisted power saving mechanism. This mechanism is built on top of the existing standard PSM (SPSM). Since the SPSM has been implemented for WiFi operating in infrastructure mode our mechanism supports only this mode. The main highlights of this implementation is that the the ZigBee Power Saving Mechanism (ZPSM) Access Point (AP) sends a wakeup frame periodically to wakeup a ZPSM Client if there is a packet intended for it. The ZPSM client wakes up once the delay bound requirement is met. If the ZigBee channel quality is not ideal, then the WiFi is woken uo based on a dynamic listen interval computed at the client side.

From the experiment it can be observed that the delay requirements have been met and the energy consumption is very less. Hence, this scheme is a very effective to save power in battery powered mobile devices.

In the future, we would like to extend this implementation so that it works on devices in the Adhoc mode as well.

BIBLIOGRAPHY

Agarwal, Y., Chandra, R., Wolman, A., Bahl, P., Chin, K., and Gupta, R. (2007). Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, MobiSys '07, pages 179–191, New York, NY, USA. ACM.

Agrawal, P., Kumar, A., Kuri, J., Panda, M., 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.

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*, MobiSys '09, pages 249–262, New York, NY, USA. ACM.

CrossbowTechnologyInc (2014). TelosB Crossbow Mote Datasheet. `http://www.willow.co.uk/TelosB_Datasheet.pdf`. [Online; accessed 10-January-2014].

IEEE (1999). Wireless LAN medium access control (MAC) and physical layer (PHY) specification. In *IEEE 802.11 Specification*, New York, NY, USA. IEEE.

Jin, T., Noubir, G., and Sheng, B. (2011). Wizi-cloud: Application-transparent dual zigbee-wifi radios for low power internet access. In *INFOCOM, 2011 Proceedings IEEE*, pages 1593–1601.

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

LSR (2014). TiWi-R2. `http://www.lsr.com/wireless-products/tiwi-r2`. [Online; accessed 13-February-2014].

Malinen, J. (2013). hostapd. http://hostap.epitest.fi/hostapd/. [Online; accessed 13-February-2014].

Namboodiri, V. and Gao, L. (2010). Energy-efficient voip over wireless lans. *Mobile Computing, IEEE Transactions on*, 9(4):566–581.

pandaboard.org (2014a). Pandaboard ES 4460 Image. http://pandaboard.org/sites/default/files/4460_panda.png. [Online; accessed 13-February-2014].

pandaboard.org (2014b). Pandaboard ES Specification. http://pandaboard.org/content/pandaboard-es. [Online; accessed 22-December-2014].

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*, MobiSys '06, pages 220–232, New York, NY, USA. ACM.

Qiao, D. and Shin, K. (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 vol. 3.

Qin, H., Wang, Y., and Zhang, W. (2011). Zigbee-assisted wifi transmission for multi-interface mobile devices. In Puiatti, A. and Gu, T., editors, *MobiQuitous*, volume 104 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 248–259. Springer.

Qin, H. and Zhang, W. (2013). Zigbee-assisted power saving management for mobile devices.

Shih, E., Bahl, P., and Sinclair, M. J. (2002). Wake on wireless: an event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, MobiCom '02, pages 160–171, New York, NY, USA. ACM.

stanford.edu (2013). TinyOS Documentation Wiki. http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page. [Online; accessed 10-January-2014].

TazTagInc (2014). The TazTag Android Phone. http://www.taztag.com/. [Online; accessed 13-January-2014].

tinyos.net (2007a). Basestation App. http://www.tinyos.net/tinyos-2.x/apps/BaseStation/. [Online; accessed 13-January-2014].

tinyos.net (2007b). TinyOS 2.02 Documentation. http://www.tinyos.net/tinyos-2.x/doc/. [Online; accessed 10-January-2014].

UCBerkeley (2014). TelosB Crossbow Mote Image. http://www.eecs.berkeley.edu/~culler/eecs194/labs/lab1/telosb.JPG. [Online; accessed 10-January-2014].

Wehrle, K., Pahlke, F., Ritter, H., Muller, D., and Bechler, M. (2002). *The Linux Networking Architecture*. Pearson, NJ, USA.

Wikipedia (2006). TinyOS Wiki. http://en.wikipedia.org/wiki/TinyOS. [Online; accessed 10-January-2014].

Yoo, J.-W. and Park, K.-H. (2011). A cooperative clustering protocol for energy saving of mobile devices with wlan and bluetooth interfaces. *Mobile Computing, IEEE Transactions on*, 10(4):491–504.

ZigBeeAlliance (2014). ZigBee Alliance. http://www.zigbee.org. [Online; accessed 12-February-2014].

APPENDIX A. TERMS AND ABBREVATIONS

AP	Access point
BI	Beacon Interval
BSS	Basic Service Set
BSSID	Basic Service Set Identifier
CAM	Constant Awake Mode
DLI	Dynamic Listen Interval
LI	Listen Interval
SPSM	Static(Standard) Power Saving Management
STA	Station
TBTT	Target Beacon Transmission Time
TIM	Traffic Indication Map
WI	Wakeup Interval
WF	Wakeup Frame
WLAN	Wireless Local Area Network
ZPSM	ZigBee Power Save Management

APPENDIX B. ENVIRONMENT SETUP

B.1 Hardware

The platform used for the experiement is Pandaboard ES Rev B2. More details can be found in http://pandaboard.org. The mote used is Crossbow's Telosb platform. More information can be found at http://www.willow.co.uk/TelosB_Datasheet.pdf

B.2 Operating System

The operating system installed on pandaboard is Ubuntu Precise for ARM running Linux kernel version 3.2. Instructions to install Ubuntu on the pandaboard and given here https://wiki.ubuntu.com/ARM/OmapDesktopInstall

B.3 Setting up motes

B.3.1 Development environment

The motes run TinyOS and one system must be setup according to instruction found here http://tinyos.stanford.edu/tinyos-wiki/index.php/Installing_TinyOS_2.1.1

B.3.2 Connecting the motes

Connect the motes and observe where it is mounted and change the permissions. After that deploy the BaseStation application onto the mote.

- sudo chmod 666 /dev/ttyUSB0
- make telosb,reinstall 0

B.4 Setting up the access point

B.4.1 Hostapd and Dhcp server

This section explains how to install hostapd and dhcp server and configure them to setup an access point in infrastructure mode.

- sudo apt-get install hostapd
- sudo apt-get install dhcp3-server
- sudo service network-manager stop
- sudo hostapd hostapd.conf
- sudo ifconfig wlan0 up 192.168.1.1 netmask 255.255.255.0
- sudo ln -s /var/run/dhcp-server/dhcpd.pid /var/run/dhcpd.pid
- sudo dhcpd wlan0

B.4.2 Setting up ZPSM AP

Connect the mote onto the Pandaboard and change its permissions as shown in B.3.2. Our work has two parts-the kernel module and the application.

B.4.2.1 Loading the kernel module

- sudo insmod zpsmapmod.ko

B.4.2.2 Running the application

- ./zpsmap <device_path> <baud_rate>

B.5 Setting up the client station

Connect the mote onto the Pandaboard and change its permissions as shown in B.3.2. Our work has two parts-the kernel module and the application.

B.5.0.3 Loading the kernel module

- sudo insmod zpsmclientmod.ko

B.5.0.4 Running the application

- ./zpsmclientapp <device_path> <baud_rate> <delay_bound>

APPENDIX C. CODE SNIPPETS

C.1 Data Structures

Listing C.1: Membership Table

```
struct ZPSM_queue_entry {
        struct list_head list;
        unsigned long IP;
        int index;
        unsigned long timeout;
        int outOfRange;
};
```

Listing C.2: Wakeup Frame

```
typedef struct zpsm_message_t
{
        unsigned int delay_list[MAX_CLIENTS];
        unsigned int num_clients;
}zpsm_message_t;
```

C.2 Interfaces

Listing C.3: ZPSM AP Serial Send Interface

```
/**
* Send a message from PC to mote through serial port
* @param device_filename The filename of mote device eg. /dev/ttyUSB0
* @param msg The message to send to mote
* @param baud_rate baude rate of mote
* @return 0 for success else error
**/
int sendMsgToMote(const char* device_filename, zpsm_message_t msg,
int baud_rate);
```

Listing C.4: ZPSM AP Serial Listen Interface

```
/**
* Opens the mote and returns a structure to read or write from it
* @device_filename Device filename of the mote like "/dev/ttyUSB0"
* @baud_rate Baud rate of the serial device by default, 115200
* @return The device source object
**/
serial_source open_mote(const char* device_filename, int baud_rate);


/**
* Reads packets from serial device
* @src The device source object
* @len OUT Length of packets to read
* @return packets read
**/
void *read_serial_packet(serial_source src, int *len);
```

Listing C.5: ZPSM AP Membership Manager Interface

```
/**
 * Find index by IP address
 *
 * @param __u32 daddr the IP address
 * @return index if it is found; ohterwise -1
*/
int ZPSM_queue_find(__u32 daddr);

/**
 * Enqueue the entry based on IP address
 *
 * @param __u32 daddr the IP address
 * @return 1 if success; ohterwise -1
*/
int ZPSM_queue_enqueue(__u32 IP);

/**
 * Handle different control message base on verdict
 * If the verdict equals to ZPSM_QUEUE_UPDATE_TIMEOUT, find
 * the entry based on IP address and update active time
 * If the verdict equals to ZPSM_QUEUE_UPDATE_STATE, find
 * the entry based on IP address and update the state
 *
 * @param verdict quals to ZPSM_QUEUE_UPDATE_TIMEOUT
 *   or ZPSM_QUEUE_UPDATE_STATE
 * @param __u32 daddr IP address
 * @param int state station's state
*/
void ZPSM_queue_set_verdict(int verdict, __u32 daddr, int state);

/**
```

```
* Print out the total number of ZPSM members
*/
void ZPSM_print_total(void);

/**
* Dequeue based on IP address
*
* @param __u32 IP IP address
*/
void ZPSM_dequeue(__u32 IP);

/**
* Add station to request index queue
*
* @param int index the station's ZPSM index (pre-assigned by AP)
* @param __u32 IP IP address of the station
*/
void ZPSM_add_request_index(int index, __u32 IP);

/**
* send index to clients if the request index queue is not empty
*/
void ZPSM_request_index_flush(void);

/**
* Delete all the timeout station from membership table
*/
void ZPSM_timeout_flush(void);
```

```
/**
* Get active time by IP address
*
* @param __u32 IP IP address
* @return timeout the active time
*/
unsigned long ZPSM_queue_find_timeout(__u32 IP);
```