TOWARDS ACHIEVING HIGHER THROUGHPUT WITH MICROCHIP BASED SMALL FOOTPRINT WIRELESS VIBRATION SENSORS USING ZIGBEE AND IEEE802.15.4 PROTOCOL

By

XINWEI CAI

Master of Science in Computer Science

Oklahoma State University

Stillwater, Oklahoma

2008

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2008

TOWARDS ACHIEVING HIGHER THROUGHPUT

WITH MICROCHIP BASED SMALL FOOTPRINT

WIRELESS VIBRATION SENSORS USING ZIGBEE

AND IEEE802.15.4 PROTOCOL

Thesis Approved:

Dr. Venkatesh Sarangan

Thesis Adviser

Dr. Johnson Thomas

Dr.Nophill Park

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my great appreciation to Dr. Venkatesh Sarangan, my principal advisor, for his instruction, knowledge, assistance, encouragement, and friendship throughout my research work. With his supervision and directions, I could have successfully completed my thesis research. I sincerely thank the other members of my graduate committee, Dr. Johnson Thomas and Dr. Nophill Park for their advices for my research. I also appreciate Dr. Satish Bukkapatnam, who provides the experimental environment for the research work. In addition, I need to give thanks to the endeavor of previous and current team workers, Steven Welch, Vinay Abburi, Yun Wu, and Jakkrit Kunthong.

Additional thanks go to Computer Science Department and Oklahoma State University for the uses of their facilities.

Special thanks go to my wife, Xiaomei Bai, and my parents Guangdao Cai and Baozhen Wang, whose love, encouragement, sacrifice and incredible help carry me through my studies and make my education goal accomplish.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

A wireless sensor network is a disturbed wireless network system that consists of autonomous devices with sensors at distributed locations. 802.15. 4 and ZigBee are standard protocols that provide the network infrastructure required for a wireless sensor network system. ZigBee protocol defines the Application Layer and Network Layer, whereas 802.15.4 defines the Physical Layer and Media Access Control (MAC) Layer. To develop a sensor network application, low cost, small footprint, long battery life, and high throughput are always the key design requirements. Our research goal is to develop and design a sensor network system with high throughput, small footprint vibration sensors. Such a sensor network system can be used in applications, such as Pallet and Container monitoring, Pipeline Integrity Monitoring.

1.1 Applications

Applications of ZigBee Network include Energy Management and Efficiency, Home Automation, Building Automation and Industrial Plant Monitoring, and so forth. In this section, we give an introduction of some typical applications.

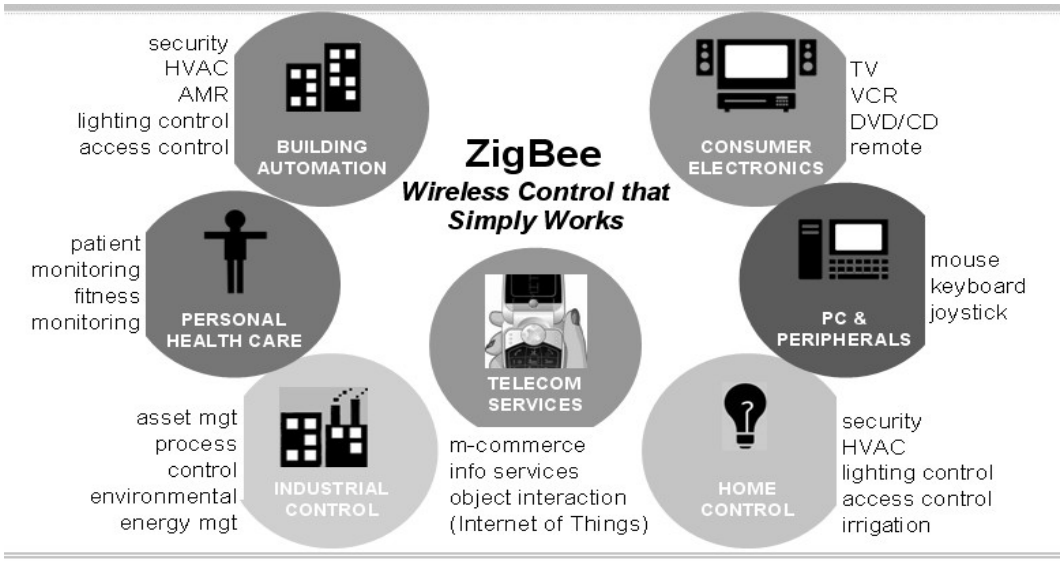

Figure 1-1 ZigBee applications (source: ZigBee Alliance)

1.1.1 Industrial plant monitoring

Wireless sensor and control network using ZigBee technology can provide a more accurate, more efficient and safer way of Industrial Plant Monitoring. The benefits are extending manufacturing and process control systems reliably, improving asset management by continuously monitoring critical equipment, automating acquisition of data from remote sensors to reduce user intervention, especially in the environments in which it is hazardous for human exposure.

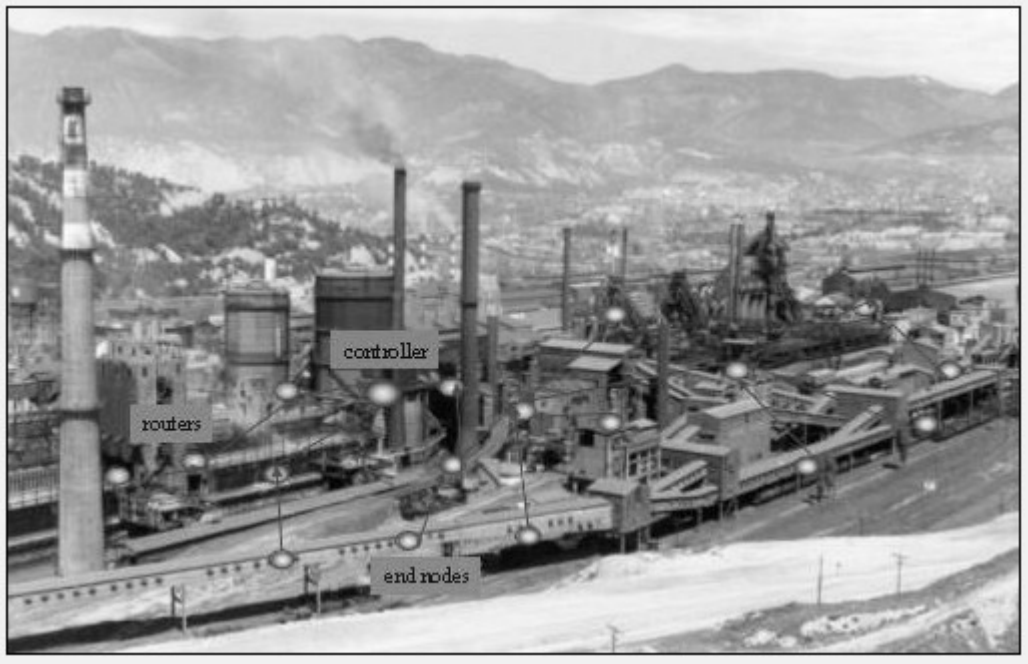

Figure 1-2 Industrial Plant Monitoring using ZigBee
(Source: Mitsubishi Electric Research Laboratories)

1.1.2 Home automation

Low power consumption and low data rate ZigBee network makes it possible to provide an affordable and flexible solution for home automation. People may enjoy flexible management of lighting, heating and cooling systems from anywhere in their home, run and configure multiple systems from remote control, automate multiple home control system to improve safety and conservation.

By using ZigBee technology, devices or equipments that can be included in the ZigBee Home Automation Network are:

- Light switches and lights
- Heating, ventilating, AC controls, and thermostats
- Smoke detectors/alarms
- Electricity, water and gas meters

- TV, computer and other electronic devices
- Security devices

Figure 1-3 ZigBee Home Automation (source: Popular Science Magazine)

1.1.3 Building automation

ZigBee network can combine lighting, heating, AC, security system and monitoring system into a single platform, thus to make building automation easier and more efficient. The benefits that ZigBee network bring us in Building Automation are integrating and centralizing management of lighting, heating, cooling and security, automating control of multiple systems to improve conservation, flexibility and security.

Figure 1-4 ZigBee Building Automation

1.2 Container monitoring using wireless sensor network

One of our research goals for which we are designing small footprint wireless vibration sensors is to serve for a container monitoring system. Hence, we give more explanation here for such an application.

1.2.1 Motivation

With over tens of thousands of containers arriving each day at the ports, information technologies are necessary to attain end-to-end supply chain integrity and operational efficiency. Here are some motivations for the development of container monitoring system.

- ~1 billion tons of container traffic emanate from US ports
- 6% growth in container traffic expected
- ~50-50% split between local and transcontinental shipping
- <1% of containers inspected and movements tracked

1.2.2 Method

A container monitoring system is a comprehensive system which combines computer technology, wireless sensing and RFID (Radio Frequency Identification) to provide effective means for monitoring of transport containers and their contents. Each node has a vibration senor on it. Samples acquired from these sensors are sent to the Coordinator node through the wireless technology of ZigBee. The Coordinator acts as a gateway to a computer system, which analysis the sample data from the End nodes and tells people the integrity and status of containers. In addition, the RFID tags help to generate container

status reports for customers receiving goods, suppliers and shippers of containers, as well as container carriers from the departure site to destination site and return.

Figure 1-5 Container monitoring using ZigBee wireless sensors

CHAPTER II

ZIGBEE & IEEE 802.15.4

The core of our research and project is ZigBee protocol. ZigBee is the only wireless standard which is developed for very low-cost, very low power consumption, two-way, wireless communication. In this section we will give an introduction of the ZigBee protocol, wireless sensors using ZigBee technology. We also give the reasons why we choose Microchip's ZigBee products as a platform to do our research. At last we give an introduction of IEEE 802.15.4 protocol.

2.1 ZigBee protocol

ZigBee is a standard wireless network protocol, which uses the IEEE 802.15.4 specification as its Physical Layer and Medium Access Layer and itself defines the Network Layer and Application Layer. ZigBee was created to address the market need for a cost-effective, standards-based wireless networking solution that supports low data-rates, low-power consumption, security, and reliability. ZigBee is the only standards-based technology that addresses the unique needs of most remote monitoring and control and sensory network applications. ZigBee protocol is developed and maintained by ZigBee Alliance, "The ZigBee Alliance is developing very low cost, very low power consumption, two-way, wireless communications standard. Solutions adopting the ZigBee standard will be embedded in consumer electronics, home and building automation, industrial controls, PC peripherals, medical sensor applications, toys and games."[1]

From the figure, we can see that ZigBee works at a low data rate of bandwidth, as ZigBee Alliance said, "ZigBee standard uniquely fills a gap for low data rate applications."

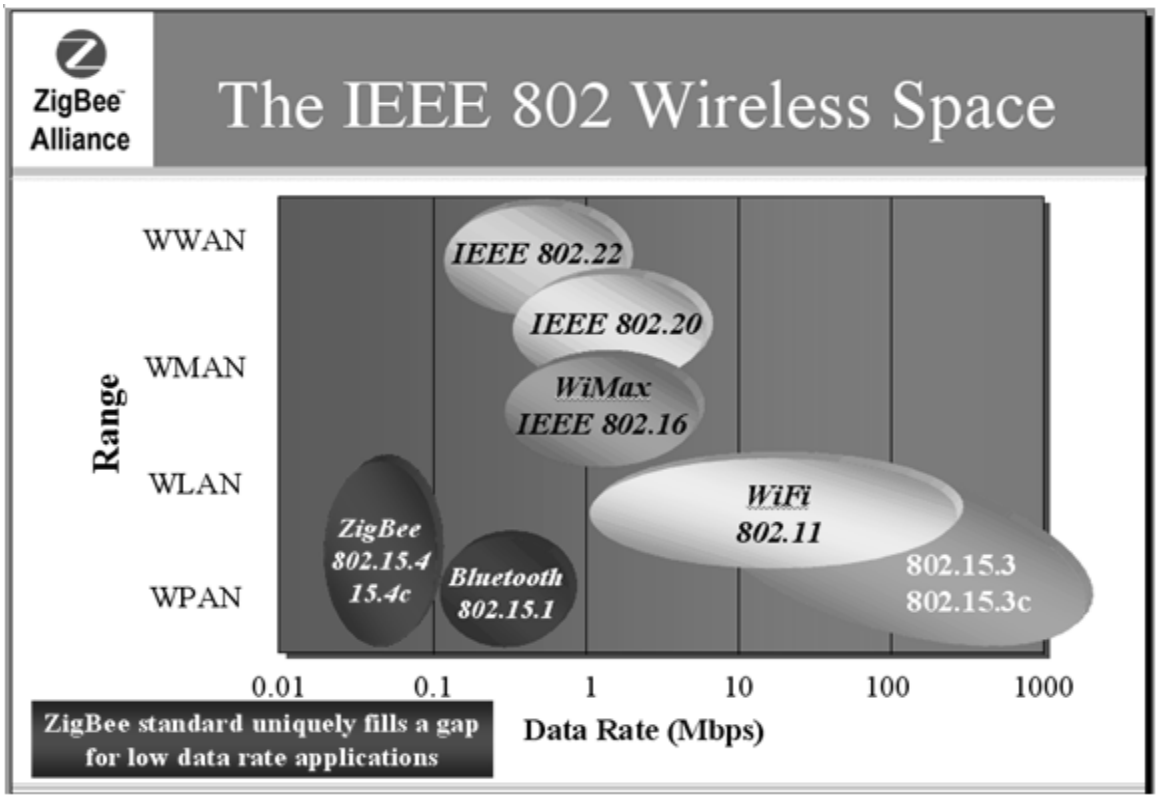

Figure 2-1 IEEE 802 wireless spaces (Source: ZigBee Alliance)

2.1.1 Device types

A ZigBee network may include three kinds of devices, the Coordinator, Router, and End Device. On IEEE 802.15.4's view, these devices are divided into two categories, FFD (Full Function Device) and RFD (Reduced Function Device).

2.1.1.1 Coordinator

Coordinator is a FFD. There is only one this kind of device for each network. Coordinator forms the network, allocates network addresses, holds binding table.

2.1.1.2 Router

Routers are FFD devices. They are Optional in the ZigBee network. Routers are used to extend the physical range of the network. They allow more nodes to join the network. Router may also perform monitoring and control functions.

2.1.1.3 End device

End Devices can be either FFD or RFD. End Devices perform monitoring or control functions.

2.1.2 A Typical ZigBee hardware device

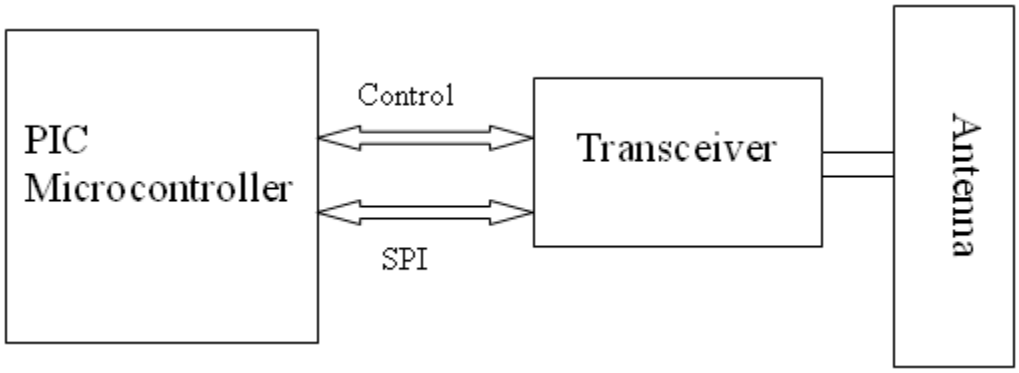

Figure 2-2 A typical ZigBee hardware device

A ZigBee device has three main parts, a PIC (Peripheral Interface Control) Microcontroller, which is the SPI (Serial Peripheral Interface) master, a Transceiver, which is the SPI slave, and an Antenna. Generally, if the device is an End Device, it may also include a sensor which can be temperature sensor, vibration senor, etc.

2.1.3 ZigBee network topology

Network Topology refers to the configuration of the hardware components and how the data is transmitted through that configuration. ZigBee support three basic different network topologies: star network, cluster tree network, and mesh network (For some applications, like monitoring a pipeline, we may use a special configuration, linear topology).

2.1.3.1 Star topology

Star topology is the simplest configuration. It is also called point to point topology. All End devices are within the effective radio frequency range of the Coordinator. There are not routers. A star network configuration consists of one ZigBee protocol coordinator node and one or more end devices. End devices can not communicate with each other. Message from a source End Device node first goes to the Coordinator. The Coordinator then forwards the message to the destination End Device.

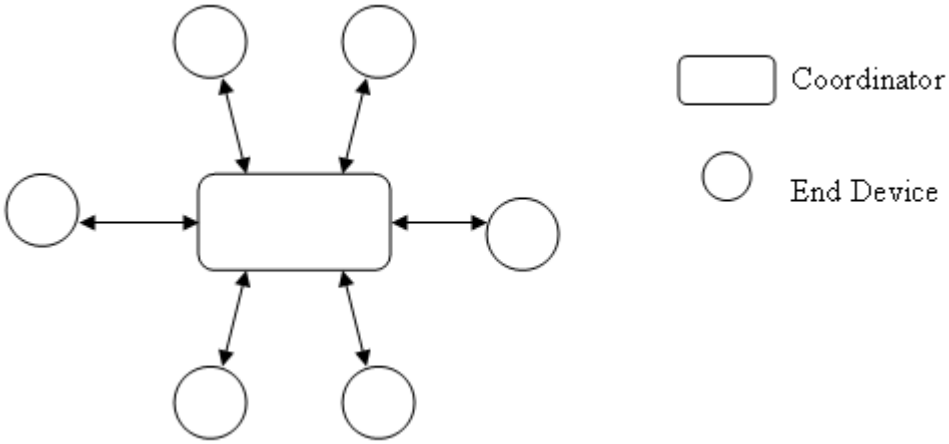

Figure 2-3 Star topology

2.1.3.2 Cluster tree topology

Star topology has constraints for its radio range. To extend the physical range of a network, sometimes we may use a cluster tree topology. In this configuration, End Devices may either join to the Coordinator or to the routers. With the addition of routers, an End Device does not need to be in the radio range of the Coordinator. All messages in a cluster tree topology are routed along the tree. It means that a message from a source End Device first goes to that End Device's parent router, then it goes to the next upper level parent router until it reach the Coordinator. And then, the message will be forwarded downstream until it gets to the destination node.

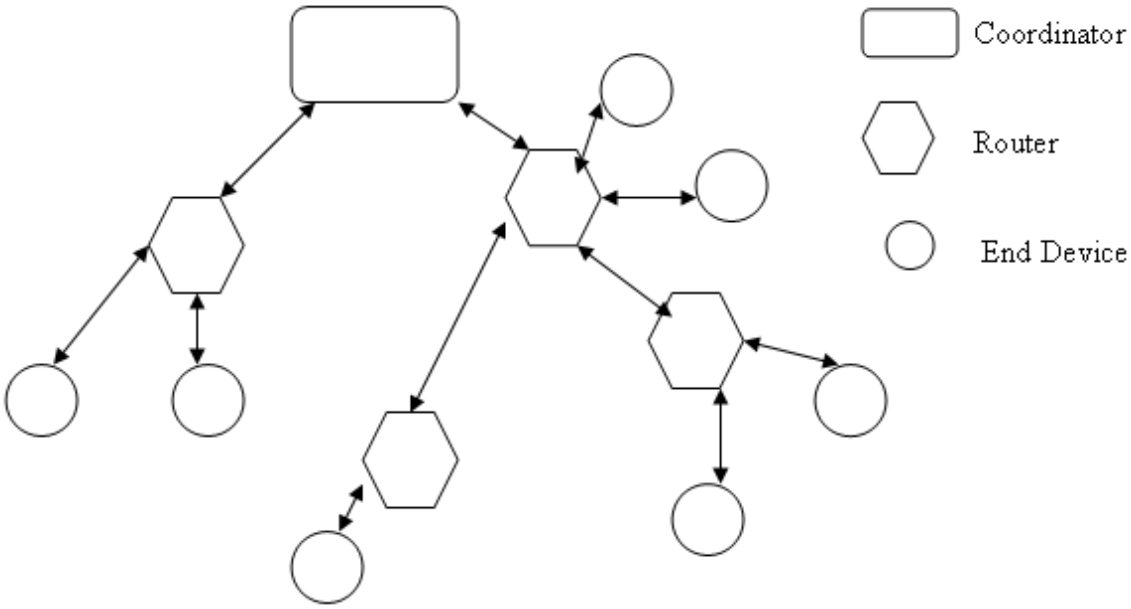

Figure 2-4 Cluster Tree topology

2.1.3.3 Mesh topology

A Mesh topology is similar to a cluster tree topology, except that routers can rout messages in multiple ways that do not need to belong to the tree structure. A mesh topology is also called peer to peer topology. It consists of a mesh of interconnected routers and end devices. Each router is typically connected through at least two pathways, and can relay messages for its neighbors.

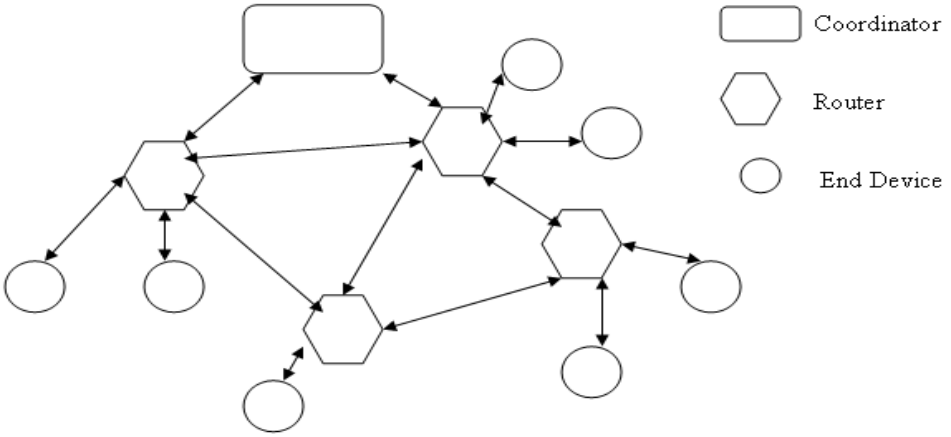

Figure 2-5 Mesh topology

2.1.4 ZigBee protocol stack

ZigBee uses the IEEE 802.15.4 specification as its Physical Layer and Medium Access Layer. ZigBee protocol itself defines the Network Layer and Application Layer.

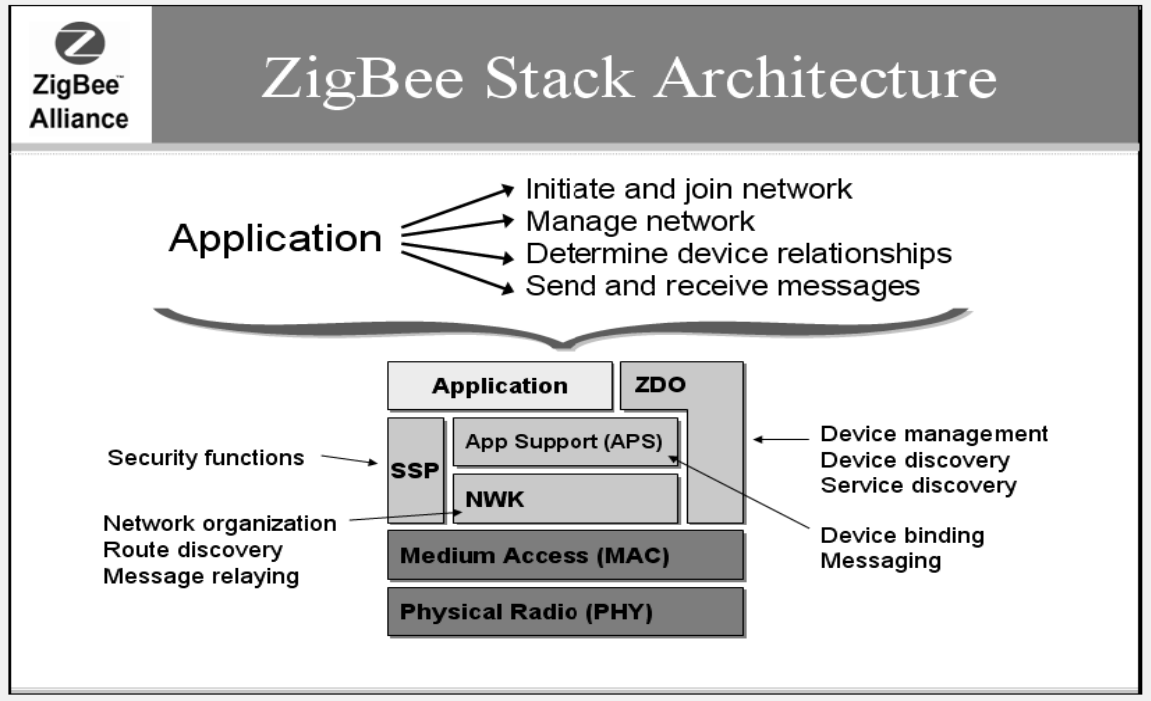

Figure 2-6 ZigBee Stack Architecture (Source: ZigBee Alliance)

2.1.4.1 PHY layer (Physical Layer)

The PHY of the stack provides two services: PHY data services and PHY management service. The PHY data service enables PPDU (PHY Protocol Data Unit) to be transmitted and received across the physical radio channel. The PHY management service is to interface to the PLME (Physical Layer Management Entity) service access point (SAP). There are three available frequency bands defined by IEEE 802.15.4, 868 MHz in Europe, 915 MHz in the USA and Australia, and 2.4 GHz in worldwide. The 2.4 GHz band provides a data rate of 250 kbps, 915 MHz provides 40 kbps and 868 MHz provides a 20 kbps data rate.

2.1.4.2 MAC layer (Media Access Control Layer)

The MAC of the stack provides two services: MAC data services and MAC management service. The MAC data service enables MPDU (MAC Protocol Data Unit) to be transmitted and received across the PHY data service. The MAC management service is to interface to the MLME (MAC Layer Management Entity) service access point (SAP). The features of the MAC layer are beacon management, channel access, GTS (Guaranteed Time Slot) management, frame validation, acknowledged frame delivery, association, and disassociation.

2.1.4.3 NWK (Network) layer

The NWK Layer provides two services: data service and management service. The NLDE (NWK Layer Data Entity) provides the data transmission service via its associated service access point. The NLME (NWK Layer Management Entity) provides the management service via its associated service access point.

2.1.4.3.1 Services provided by NLDE [2]:

- Generation of the Network level PDU (NPDU): The NLDE shall be capable of generating an NPDU from an application support sub-layer PDU through the addition of an appropriate protocol header.
- Topology specific routing: The NLDE shall be able to transmit an NPDU to an appropriate device that is either the final destination of the communication or the next step toward the final destination in the communication chain.
- Security: The ability to ensure both the authenticity and confidentiality of a transmission.

2.1.4.3.2 Services provided by NLME [2]:

- Configuring a new device: The ability to sufficiently configure the stack for operation as required. Configuration options include beginning an operation as a ZigBee coordinator or joining an existing network.
- Starting a network: The ability to establish a new network.
- Joining and leaving a network: The ability to join or leave a network as well as the ability for a ZigBee coordinator or ZigBee router to request that a device leave the network.
- Addressing: The ability of ZigBee coordinators and routers to assign addresses to devices joining the network.
- Neighbor discovery: The ability to discover, record, and report information pertaining to the one-hop neighbors of a device.
- Route discovery: The ability to discover and record paths through the network, whereby messages may be efficiently routed.
- Reception control: The ability for a device to control when the receiver is activated and for how long, enabling MAC sub-layer synchronization or direct reception.

2.1.4.4 Application layer

There are three components in application layer. They are Application Frame, ZDO (ZigBee Device Object), Application Support Sub-layer.

2.1.4.4.1 Endpoint

To describe Application Frame and ZDO, we need to first explain the concept of endpoint. Every ZigBee device can contain up to 241 endpoints. An endpoint is analogous to a TCP port – each endpoint can provide a kind of application service.

Endpoint0 itself is reserved for device management, and is called the ZigBee Device Object (ZDO). Endpoint 0 provides the means to control and manage devices and endpoints. The remaining 240 endpoints can be allocated for any required services.

2.1.4.4.2 Application frame

Application Frame consists of application profiles which may be public profiles or vendor defined profiles. Application profiles define what messages are sent over the air for a given application. Devices with the same application profiles can interoperate each other.

2.1.4.4.3 ZigBee device object

ZigBee Device Object, ZDO, is an application that implements the primitives of application support sub-layer and network layer to implement ZigBee devices.

2.1.4.4.4 Application support sub-layer

The application support sub-layer provides the interface between the application layer and network layer through a general set of services. Both ZDO (ZigBee Device Object) and application objects, which may be public or vendor defined, use these services. These services are offered via two entities: APSDE (Application Support Sub-layer Data Entity) and APSME (Application Support Sub-layer Management Entity).

2.1.4.4.4.1 Application support sub-layer data entity [2]

- Generation of the Application level PDU (APDU): The APSDE shall take an application PDU and generate an APS PDU by adding the appropriate protocol overhead.
- Binding: This is the ability to match two devices together based on their services and their needs. Once two devices are bound, the APSDE shall be able to transfer a message received from one bound device over to the second device.
- Group Address Filtering: This provides the ability to filter group addressed messages based on whether the device is a member of the group or not.
- Reliable transport: This increases the reliability of transactions above that available from the NWK layer alone by employing end-to-end retries.
- Duplicate rejection: Messages offered for transmission will not be received more than once.

2.1.4.4.4.2 Application support sub-layer management entity

The APSME provides a management service to allow an application to interact with the stack. The APSME also provides the ability to match two devices together based on their services and their needs. In addition, the APSME provides service for AIB (Information Base Maintenance) management security and group management.

2.2 IEEE 802.15.4 protocol

ZigBee defines the application layer and network layer of the ZigBee stack, whereas IEEE 802.15.4 protocol defines the MAC layer and physical layer, which provide the network infrastructure for ZigBee sensor network.

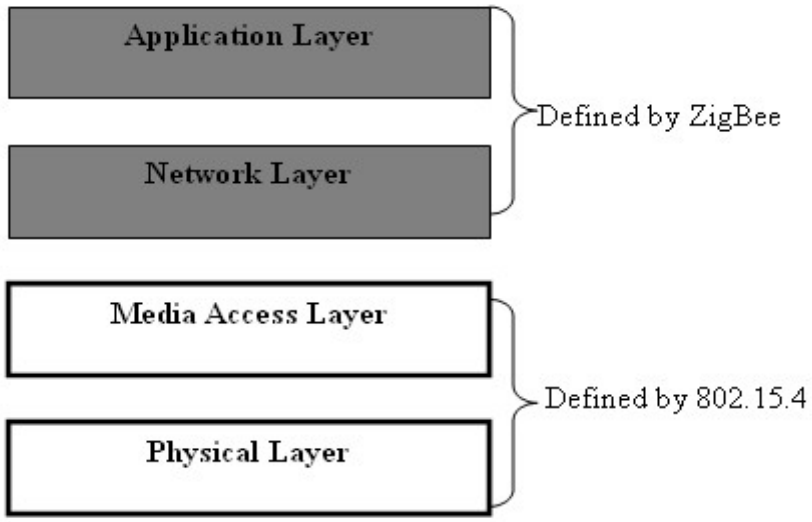

Figure 2-7 Stack architecture of ZigBee

There are three frequency bands, 2.4 GHz, 915MHz and 868 MHz, defined by IEEE 802.15.4. 868 MHz offers channel 0, 915 MHz offers channels from 1 to 10, and 2.4 GHz offers channels form 11 to 26.Whereas, 2.4 GHz provides the maximum bit rate, which is 250 kbps. Due to the overhead of protocol header and footer, as well as the processing delays, the actual throughput will be much less than theoretical value. Also, the maximum length of MAC packet of IEEE 802.15.4 is 127 bytes.

Device Type	Services Offered	Typical Power Source	Typical Receiver Configuration
Full Function Device (FFD)	Most or all	Mains	On when Idle
Reduced Function Device (RFD)	Limited	Battery	Off when Idle

Table 2-1 Device type of IEEE 802.15.4 (Source: Microchip)

“The PICDEM Z 2.4 GHz Daughter Card features Microchip’s MRF24J40 the 2.4 GHz IEEE 802.15.4 transceiver, Targeted for the ZigBee protocol and proprietary wireless protocols such as MiWi, the MRF24J40 transceiver is perfectly suited for RF applications requiring low power and excellent RF performance. Equipped with a modular connector,

this daughter card plugs into Microchip's popular PICDEM Z demonstration board and provides a complete, embedded wireless solution.” [7]

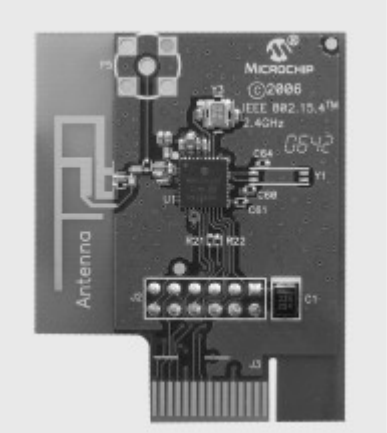

Figure 2-8 Microchip’s IEEE 802.15.4 transceiver

CHAPTER III

RELATED WORKS

ZigBee is a wireless network protocol specifically designed for low data rate sensor or control networks. ZigBee fills up the gap for low data rate wireless personal area network, LR_WPAN. Since ZigBee is designed for low data rate applications, thus there are no reported research works which address high throughput of ZigBee networks by far. For example, Microchip provides us an application for monitoring the temperature of environment, which takes 2 second to response a query (which is only two bytes). To satisfy the future applications, pipeline integrity monitoring and container monitoring, we need to make the throughput of the system not less than 36 kb/s (2000 samples with 2 bytes each in one second). Since ZigBee is the unique standard/protocol for wireless sensor network. Therefore, improving the throughput of ZigBee network to satisfy high data rate applications is a significant research work.

3.1 Why do we need ZigBee

ZigBee is the only wireless standards-based technology [1]:

- That addresses the unique needs of remote monitoring & control, and sensory network applications.
- Enables broad-based deployment of wireless networks with low cost, low power solutions.
- Provides the ability to run for years on inexpensive primary batteries for a typical monitoring application.

3.2 Wireless Devices using ZigBee

In this section, we introduce typical ZigBee products on the market and also give a simple comparison of these products.

3.2.1 PICDEM 2.4GHz by Microchip Inc.

PICDEM Z 2.4 GHz Demonstration Kit is an easy-to-use evaluation and development platform for ZigBee and IEEE 802.15.4 application designers. This kit includes Microchip's MRF24J40 transceiver and also features Microchip's PIC18 high-performance microcontroller family.

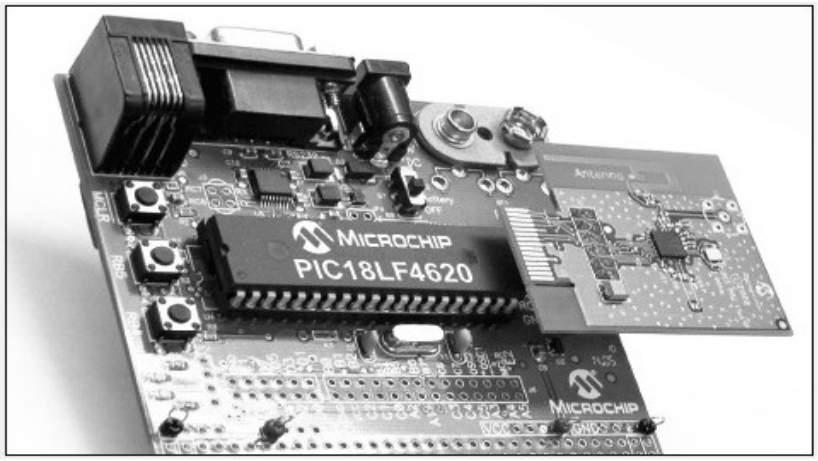

Figure 3-1 Microchip ZigBee product (source: Microchip)

3.2.2 Tmote by Moteiv Corporation

The Tmote Mini is a wireless sensor network node, or mote that supports the ZigBee(tm) specification, runs TinyOS, and is packaged in an industry-standard miniSDIO(tm) form factor. The ultra-compact form factor allows for immediate integration of the Tmote Mini "mote core" into devices such as cell phones, PDAs, and other mobile products.

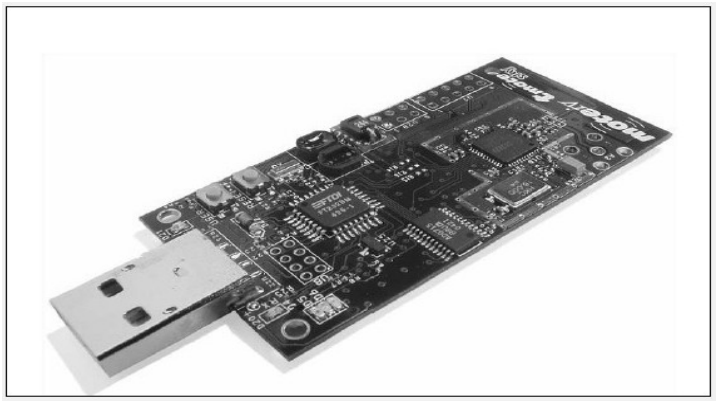

Figure 3-2 Tmote Sky (source: Moteive Corporation)

3.2.3 MICA 2.4GHz by Crossbow Technology Inc

The MICA is a 2.4 GHz, IEEE/ZigBee 802.15.4, board used for low-power, wireless sensor networks. The MICA Mote is a second generation mote module used for research and development of low power, wireless, sensor networks. The MICA mote was developed by UC Berkeley's research group on wireless sensors.

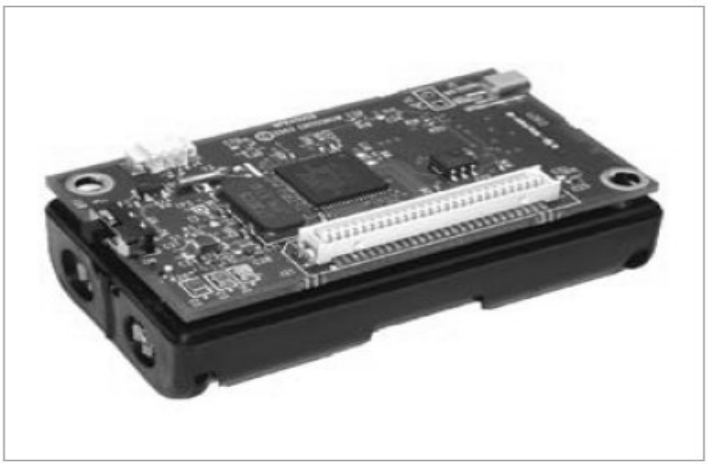

Figure 3-3 Mica2 (source: Crossbow Technology Inc.)

Company	Microchip	T-mote	MicaZ	Imote	Xbee
Power Consumption	Extremely Low	Low	Low	Very low	Very low
Operating System	No	Tiny OS	Tiny OS	Tiny OS	AT or API command
Programming Flexibility	High	Moderate	Moderate	Moderate	High
Programming Complexity	Moderate	High	High	High	Low
Expandability	Yes	Yes	Yes	Yes	Yes
Firmware	Open	Proprietary	Proprietary	Proprietary	Proprietary
Cost	Very Low	Moderate	High	Very High	Very Low

Table 3-1 Comparison of typical wireless devices using ZigBee in the Market

3.2.4 Why we choose Microchip

Microchip offers a complete ZigBee wireless solution for application system requirements. In concert with Microchip MRF24J40 Transceiver and PIC 8F4620 microcontrollers, Microchip offers a free ZigBee software stack, enabling lower development and system costs. With its ZigBee compliant platforms, the Microchip stack was written to meet the ZigBee industry standard and enable embedded system designers to utilize the stack to develop applications quickly.

CHAPTER IV

PERFORMANCE OF MICROCHIP'S PRODUCTS

This chapter we explain the issues with microchip's ZigBee products, which are data loss, conflict between ADC and radio frequency / transceiver, and low throughput. In the next chapter, we will give the solutions for there issues.

4.1 Experimental setup

There are two ZigBee devices in our current system. One device is the Coordinator device, the other is the End device. We also added a voltage generator to the End device, which can be replaced by a vibration sensor. We acquire data from the voltage generator through the ADC channel, and put these data in a packet. Then the End device sends the packet to the Coordinator device. At the Coordinator side, we receive and collect the data to a disk file. We also have the packets sniffer, programmer, power supply, and serial to USB cables, which satisfy our research work.

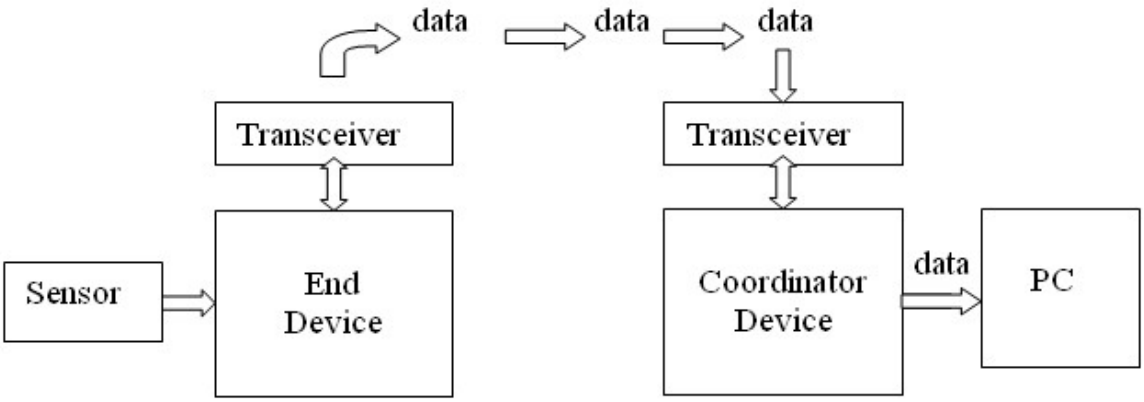

Figure 4-1 Configuration of Experimental Setup

This is an embedded environment. There are no library functions that support to write data to a disk file directly. What we did is to connect the Coordinator device to the PC's serial port (More accurate, we connected the cable to the PC's USB port and use virtual serial port techniques.), and then we use the Hyper Terminal of PC's operating system to save data into a plan text file. The purpose to collect data is that we can analysis and monitor the environment by the samples we collect. For example, we can create a model based on the samples we collect when a pipeline is normal, and once the model changed we may know there are something happened. But these works are out of our current research work.

We first tested the ZigBee Stack by sending packets with some dummy numbers as payload, such as 1 to 40. We found that there was a high rate of packet loss. Also, the default throughput is very low. A sample in-built application consumes two seconds for each packet of size 80 bytes. In addition, there was a conflict between the radio frequency and ADC. We will give the details in the following sections.

Figure 4-2 Experimental setup in the lab
(Device on the left side is the End device with sensor, in the middle is a Packet sniffer, device on the right side is the Coordinator, which we use to collect data.)

4.2 Packet loss

Coordinator device can not correctly receive all packets sent by the End device node. We have tested the receiving rate at the Coordinator device side. The sampling rates for our experiments are 5k, 10k, 15k and 20k. At each sampling rate, we did tests with different packet size, which are 20 bytes, 40 bytes, 60 bytes and 80 bytes. There are 1000 packets sent from End device to Coordinator device for all of our tests. In this stage we did not collect real data from the vibration sensor through the ADC channel. To simplify our tests, we put some dummy numbers, form 1 to 80, as our payload.

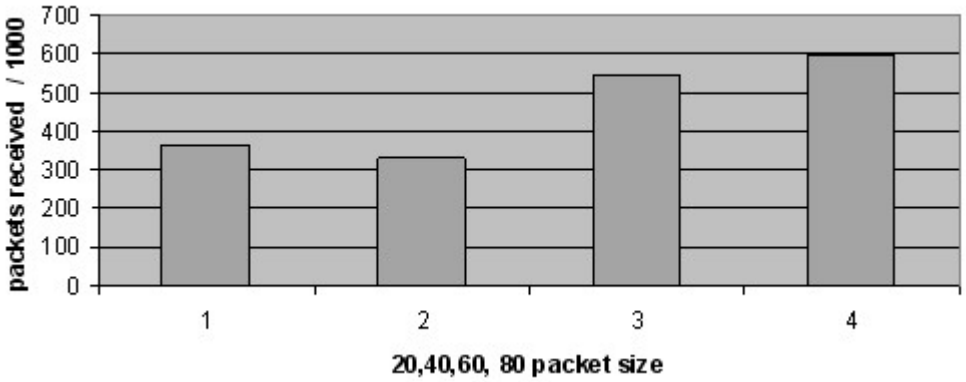

Figure 4-3 Receiving rate at 5K sampling rate

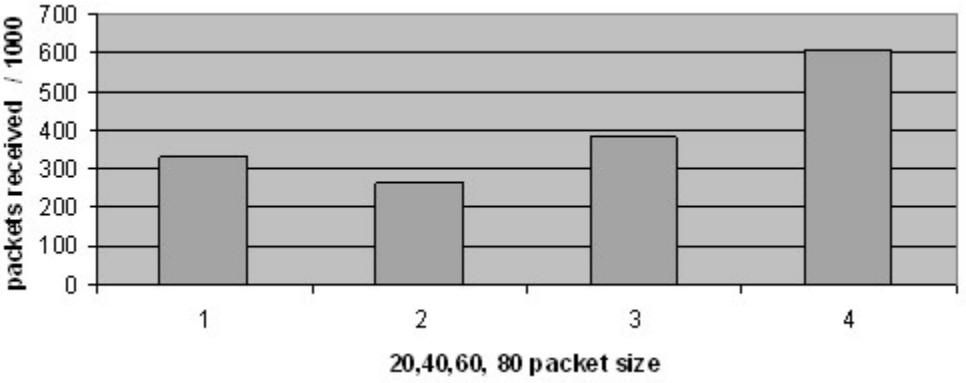

Figure 4-4 Receiving rate at 10K sampling rate

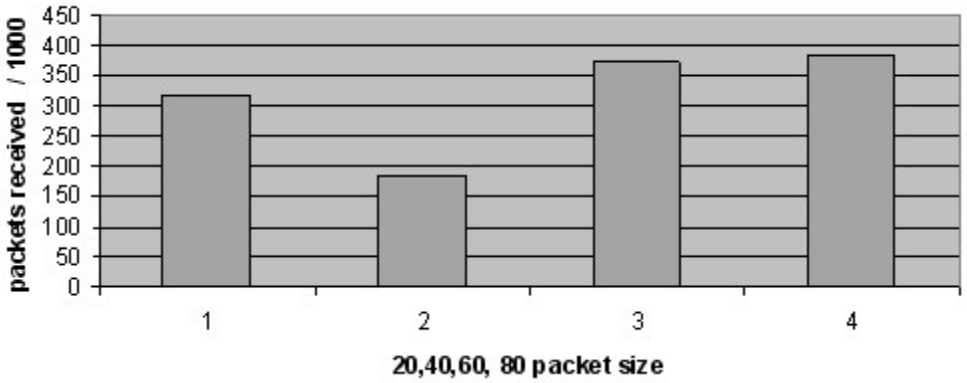

Figure 4-5 Receiving rate at 15K sampling rate

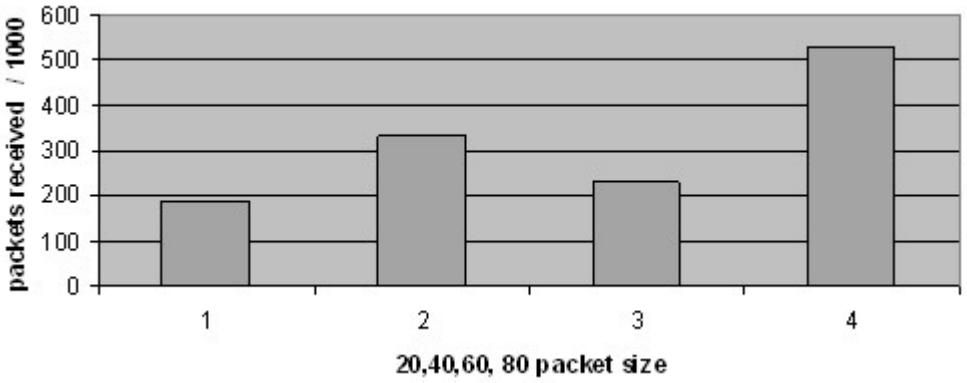

Figure 4-6 Receiving rate at 20K sampling rate

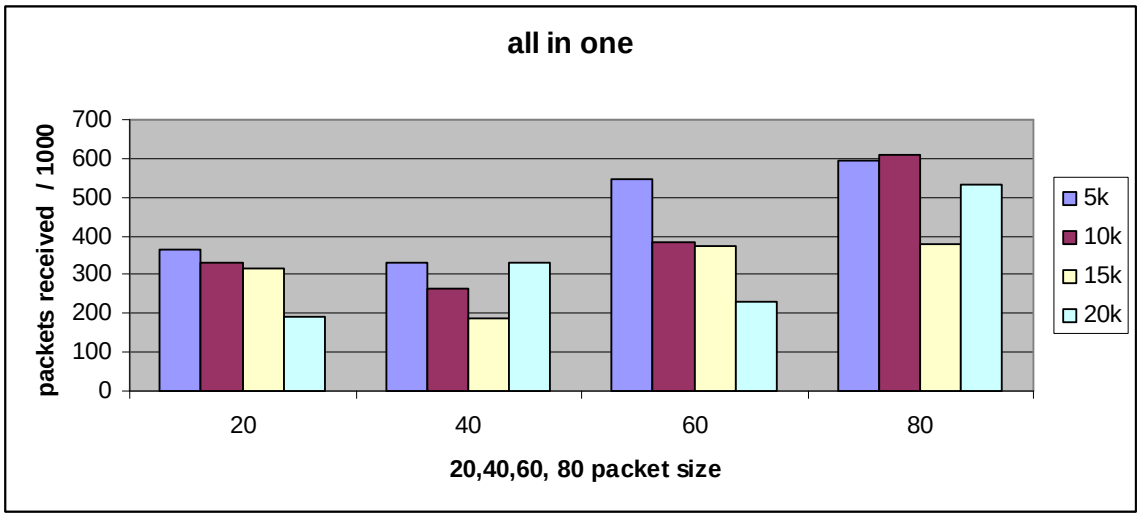

Figure 4-7 Receiving rate at different sampling rate with different packet size

From these figures we can see that there is a high rate of packet loss, to do our research and develop the applications, we first need to increase the receiving rate to an acceptable value, and otherwise all other developments may have no significances.

4.3 Conflict between ADC and RF

After we solved the packet loss issue, we added a voltage generator to the End device. In this stage, we acquired data from the voltage generator through the ADC channel, and put data in the packet as payload. A voltage generator makes it easier to measure or check if we get a correct value from the ADC. If the voltage generator works correctly, then we can simply replace it with sensors later, depending which application we aimed.

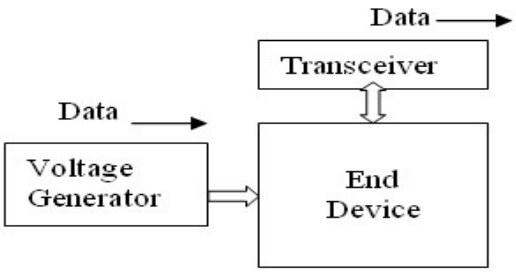

Figure 4-8 Configuration of adding a Voltage Generator to the End device

However, we got a problem once we added the voltage generator to the End device. The two devices, End device and Coordinator Device, can not communicate each other. As we observed, the Coordinator still correctly established the network, but the End device can not join the network. The Coordinator even did not know the existence of the End device. When we took away the voltage generator, the End device could join the network and communicate with the Coordinator successfully. Thus, there must be a conflict between ADC and radio frequency. We will give the reason and detailed solution in the section 4.2 of next chapter, avoiding conflicts between ADC and RF.

4.4 Low throughput

Microchip's ZigBee protocol stack is quite slow. A sample build-in application consumes two seconds to send a data packet. However, we need 2000 samples to be sent in each second. Otherwise, at low sampling rate, the application may not give an accurate analysis due to the lack of data collection.

We test the throughput at different payload size after we solved all conflicts / problems and created a working set. The table below summarizes the system performance which we have achieved.

Payload Size (byte)	Seconds(for receiving a packet)	Sampling Rate
96	0.082	585
80	0.079	506
60	0.077	389
40	0.074	270
20	0.070	142
2	0.069	14

Table 4-1 Sampling Rate at different payload size (One sample needs 2 bytes)

CHAPTER V

SYSTEM PERFORMANCE OPTIMIZATION

Our research goal is to improve the throughput of the system. Microchip has provided us an application which does not need a high throughput system. Users can push a button on the coordinator device to send out a query for a temperature. Once the end device receives this query, it may reply by sending a temperature value which is gotten form the temperature sensor on it. This process takes about 2 seconds, the payload data is 2 bytes. However, our applications need a higher speed. The system needs to be able to send out 2000 samples in one second. Hence, we need to increase the throughput of the system to satisfy our applications. In addition, we must first solve the problems of data loss, ADC and RF conflicts as we mentioned in last section, then we can focus on improving the system throughput.

5.1 Avoiding data loss

As we have tested with different payload size at different sampling rate, the packet loss rate is more that 40%. This is a serious problem which makes the application less significant. Thus, data loss becomes the first problem that we need to work on.

5.1.1 Why data loss occur

When we did tests, we noticed that the stack keep returning error codes at times which were showed on the console. We checked the user manual and found that this error codes indicate transaction overflow.

To avoid transaction overflow, we first slowed down the speed of sending data from the End device to the Coordinator device by adding more delays. But the results were not as what we had expected. Even at the speed of sending one packet at a second with the payload of one byte, packet loss still occurred and error messages were returned. This result has once made us puzzled.

IEEE 802.15.4 provides us a maximum bit rate of 250 kb/s at 2.4 GHz frequency band. There should not have problems when we just send out one byte in one second. The ZigBee stack could be slow. This may cause the transaction taking more time, but may not cause the transaction overflow. Thus we can draw the conclusion that slowing down the transaction speed / throughput does not help to avoid data loss. The figure below was captured by the Hyper Terminal, which clearly shows that there are some packets lost.

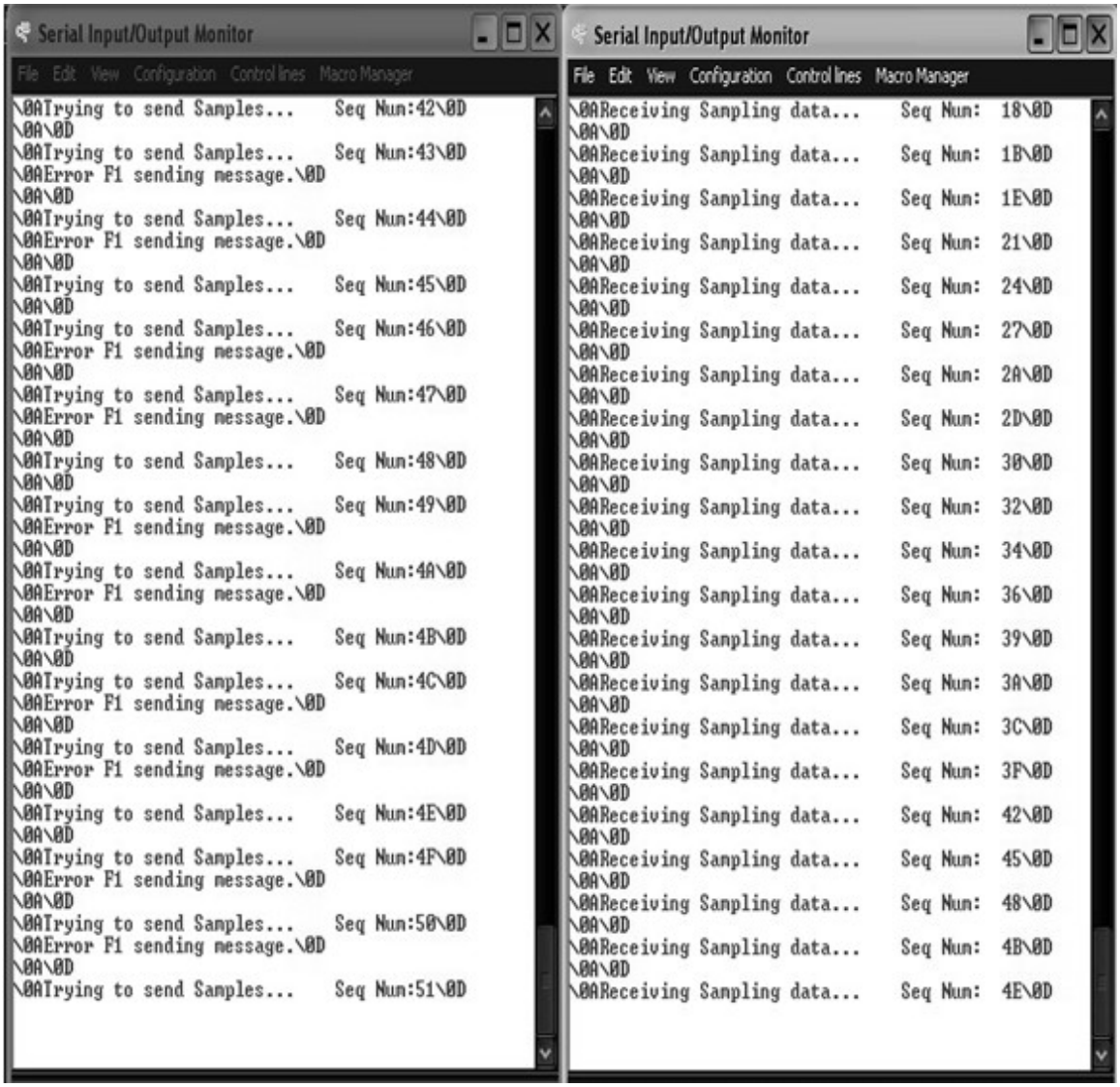

Figure 5-1 Traffic monitoring before modification.
(*Left side window is monitoring the End device; right side window is monitoring the Coordinator side)*

Since data loss is the direct result of transaction overflow and slowing down the speed of sending packets still causes transaction overflow, then we must find an effective way to control the behavior of sending packets. We added some code to implement this idea. As we tested, this is an effective method.

5.1.2 Solution to avoid data loss

Before adding controls to the transaction, we rewrote the code and defined two new endpoints, thus to isolated from other applications and also made the code more clear. By going through the code of the ZigBee stack, we found that before the End device is ready to communicate with the Coordinator device it must implement the synchronization primitive. Thus, we added code for transaction control here. Our idea is that after the Coordinator device establishes the network and the End device successfully joins the network, we make the End device call the synchronization primitive first, coordinator device will give a response, then after the stack gives a return which shows that the stack is ready to do other operations, such as sending or receiving operations , we make the End device send a packet to the Coordinator device, once the coordinator successfully receives the packet it will send an acknowledge, then the End Device call the synchronization primitive again and prepare to send out the next packet.

This idea works, based on our tests. The two devices can communicate each other correctly, and Coordinator device can receive all the packets sent by the End device. In other words, the receiving rate is 100%. Also, there are no delays injected, the devices/stack work at full speed. At this stage, system throughput is 9.37 kb/s, sampling rate is 585. Figure below was captured by the Hyper Terminal, which clearly shows that there are no packets lost.

Figure 5-2 Traffic monitoring after modification

(*Left side window is monitoring the End device; right side window is monitoring the Coordinator side)*

5.2 Avoid conflicts between ADC and RF

After we solved the problem of data loss, we planed to add the vibration sensor to the systems. Microchip's PIC4620 microcontroller provides us a 10 bit Analog to Digital converter module. We can get data from the vibration sensor through the ADC channel of the module. To make it easy, we first added a voltage generator to the main board of the End device. If the voltage generator works fine, then we can just simply replace it with a vibration sensor later. However, another problem occurred. When we added the voltage generator to the main board of the End device, the end device could not joined the

network. It means that the End device and the Coordinator device could not communicate each other when the voltage generator was added to the system. Thus, there must be a conflict between ADC and the radio frequency.

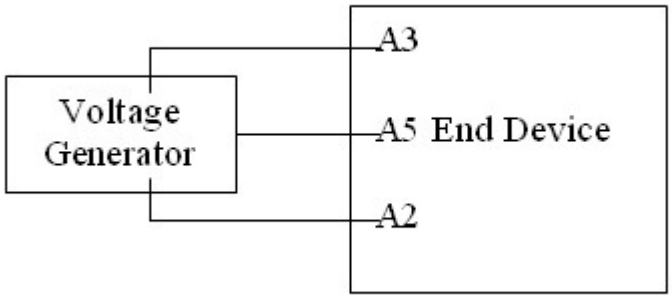

Figure 5-3 Configuration of adding Voltage Generator
(*Voltage generator was adding to the End device through A2, A3, A5 pins of the board*)

5.2.1 Where & why conflict occurs

We have done some research of the datasheets of Microchip's products and we found that there is a temperature sensor, TC77, on the main board. This temperature module shares the SPI bus with the transceiver, MRF24j40. When we added the voltage generator to the board, we connected it to the board through A2, A3, A5 pins. However, when A2 pin was connected, it turned a chip selection bit on. This chip selection makes the TC77 temperature module always occupy the SPI but. In other words, the transceiver was isolated from the system. Thus, the End device could not communicate with the Coordinator device.

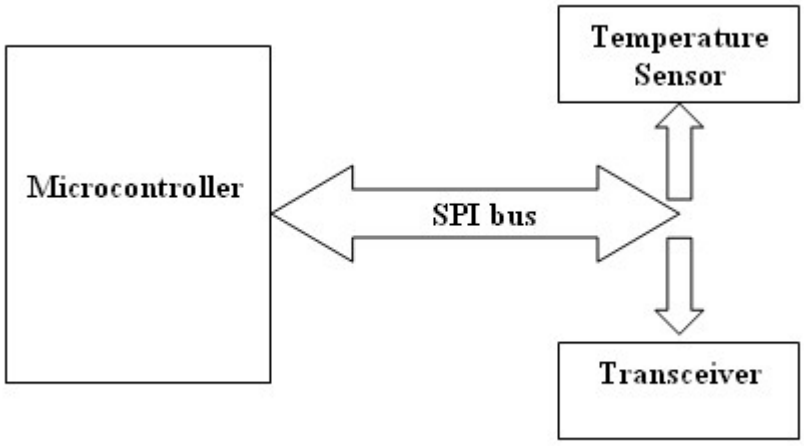

Figure 5-4 Temperature module and the Transceiver share the SPI bus

5.2.2 Solution for the conflict

For our research / application, the TC 77 temperature transceiver is totally useless for us. Thus, we can remove this module from the system board. Our goal is to make the transceiver always occupy the SPI bus and the ADC work correctly.

We first removed the TC 77 temperature sensor. We still physically connected the A2 pin, which may choose the TC77 temperature module as the SPI bus slave, to the main board. The reason we need to connect this A2 pin is that the voltage generator needs A2 pin as its voltage reference. However, from the code, we logically turned off this control bit. Thus, we not only made the ADC work, but also chose the Transceiver, MRF24j40, as the SPI bus slave.

We did tests, both ADC and RF worked correctly. There were no conflicts any more. But, we got another problem here. We can not put the calling of ADC function in the loop, in which we can continually acquire samples. When we put the Calling out side the loop, it means all the data we put in the packets as payload are all the same, the first one. We did some researches and test. Finally, we solved this problem by rewrite the ADC calling function. We put the ADC Open function outside the loop, but still called the ADC function in the loop.

At this stage of our research / project, we make both End device and Coordinator device work correctly. Also, we collect data from the ADC channel, instead of putting dummy numbers in the packets as payload.

5. 3 Improving system throughput

Microchip provides us a demonstration application, which is for queries of temperature values. Each time users put the button on the Coordinator device, a query for temperature is sent out, when the End device gets this message, it may collect a temperature value from the temperature module, and then the End device replies the Coordinator device with this temperature value. This process takes about 2 seconds. However, for our future applications, pipeline integrity monitoring and container monitoring, we need more than 2000 samples to be received in one second. Thus, we need to improve the system throughput, for which we need to optimize the ZigBee stack to satisfy our applications.

802.15.4 Protocol tells us that the maximum MAC frame size is 127 bytes. Transceiver bit rate is 250 kb/s at 2.4GHZ. However, the application receiving data rate is only 3.5 kb/s. We contacted with Microchip, and they told us," The data rate depends on the protocol standards defined by IEEE and the payloads etc." Their reply is consisted with our observations. The low data rate is due to the header and footer overhead and processing delays.

Our initial tests showed that the default Microchip's ZigBee protocol stack could at most support a sampling rate of 233, with each sample being 2 bytes. This sampling rate can

not satisfy the applications which required sampling rate of 2000 or more, our research efforts are focused on improving the sampling rate to the required levels. We developed and implemented 5 strategies to improve the overall system throughput/sampling rate.

- Increasing the payload size of the packet
- Buffering samples in memory and employing batch transmissions
- Optimizing the implementation of the ZigBee Stack
- Using data compression
- Increasing the clock rate of the microcontroller

5.3.1 Increasing the payload of the packet

Within the same processing delay, larger payload size could achieve higher throughput. As we have tested, before optimizing the implementation of the ZigBee Stack, the receiving rate is only 14 when payload size is 2 bytes. When payload size is 96 bytes, which is the maximum size based on our tests, the receiving rate could be as high as 585. Thus, using larger payload size, the throughput of the system can be increased correspondingly. However, the sampling rate of 585 is still too low to satisfy our future applications. Hence, we must find other methods to improve the system throughput even more.

5.3.2 Buffering Samples

When we did not use the technique of buffer, sampling rate and throughput were interchangeable concepts. The system throughput determined the sampling rate. However, it could be in different scenarios when we use buffering technique.

5.3.2.1 Idea of buffering samples

Sampling rate of 2000 or more is our ideal value. Whereas, what we achieved so far were only 585. To fill up this gap, we could buffer the samples from the sensor to the memory of the End device at higher rate. After that, we send data at a lower speed. For example, we could buffer 2000 samples in one second (As we had tested, this operation took far less than one second, since the ADC works extremely fast, which is the magnitude of micro-second), and send the data at a rate of 585. In this way, we can acquire higher sampling rate with lower throughput.

5.3.2.2 Implementation the idea of buffering

To implement this idea, we must first define a large memory block as the buffer. As we have tested, the maximum array size which can be defined here is 113 for integer type, or 227 for BYTE type in this system, otherwise the executable can not be successfully built. As we studied, we found that the array size is limited by the bank size of PIC18F4620 microcontroller. Data memory of PIC18F4620 microcontroller is 4 K bytes, which is divided into 16 banks (see the table below). By default, we can not define an array larger than 256bytes.

Using preprocessor and modifying the linker script, we can combine/merge several banks to one larger memory block. What we have done and tested is to merge bank11 to bank14. Thus, in theory, we can create a buffer with 1012 bytes. In practice, the maximum array size we can define is 848 bytes based on our tests.

Bank	Range	Size	Status
Bank 0	0x0080 -- 0x00FF	127 bytes	Free
Bank 1	0x0100 -- 0x01FF	256 bytes	Reserve for Heap
Bank 2	0x0200 -- 0x02FF	256 bytes	Reserve for Heap
Bank 3	0x0300 -- 0x03FF	256 bytes	Reserve for Heap
Bank 4	0x0400 -- 0x04FF	256 bytes	Reserve for Heap
Bank 5	0x0500 -- 0x05FF	256 bytes	Reserve for Heap
Bank 6	0x0600 -- 0x06FF	256 bytes	Reserve for Heap
Bank 7	0x0700 -- 0x07FF	256 bytes	Reserve for Heap
Bank 8	0x0800 -- 0x08FF	256 bytes	Reserve for Heap
Bank 9	0x0900 -- 0x09FF	256 bytes	Reserve for Stack
Bank 10	0x0A00 -- 0x0AFF	256 bytes	Reserve for RX_BUFFER
Bank 11	0x0B00 -- 0x0BFF	256 bytes	Free
Bank 12	0x0C00 -- 0x0CFF	256 bytes	Free
Bank 13	0x0D00 -- 0x0DFF	256 bytes	Free
Bank 14	0x0E00 -- 0x0EF3	244 bytes	Free
For” dbgspr”	0x0EF4 -- 0x0EFF	12 bytes	--
Bank 15	0x0F00 -- 0x0F7F	127 bytes	Free

Table 5-1 Bank allocation for the data memory

5.3.2.2 Conclusion of the solution using buffer

The buffer size is still not ideal for our application. To use this method (to write data to the memory), we must at least be able to buffer 2000 samples (4000 bytes), which is the minimum value to satisfy our future applications. In theory, we can not create an array larger than 1268 bytes in current conditions. Then we must find other methods to improve the system throughput, so as to increase the sampling rate.

5.3.3 Optimizing the implementation of the ZigBee stack

5.3.3.1 Idea of optimization

By study the ZigBee stack, we found that some operations demanded by the ZigBee protocol are unnecessary for our applications. Thus, we can eliminate these operations to

make the system more efficient, thus to increase the throughput. As the ZigBee protocol demands, each End node should require data from their parent node (in case other nodes sent it messages) at some time period, polling. Microchip makes the End device require data from the Coordinator/ router in every 2 seconds. It may be possible that we could modify the protocol stack to eliminate these operations that are unnecessary for our application, thus to make the stack work more efficiently, and satisfy our future applications.

5.3.3.2 Analysis & implementation

As we tested, we can not simply skip / eliminate these operations, otherwise problems occur. The devices may either not be able to communicate each other, or data loss occurred. Thus, we need to find a solution, which not only let us eliminate some unnecessary operations, but also make the devices communicate without problems.

Before the End device can send a data message to the Coordinator device, it needs to implement the synchronization primitive, which will call the poll request primitive. After that, a poll confirm primitive will be returned, which also implement the synchronization confirm primitive. Once all of these primitive are done, the End device then can send out a data message. Next time, when the End device wants to send data to the Coordinator device, it needs to do the above operations again. This could be very inefficient. Our idea is that we only implement the above operations once, and then make the End device continually send data to the Coordinator device for ever. Thus, we can make the stack work in an efficient way to improve the system throughput. On the other hand, since we only implement the synchronization primitive one time, then we need to add controls to determine when the next packet can be sent out. By study the ZigBee protocol and the stack, we found and tested the best place that we can put this control is under application support sub-layer data confirm primitive. We implement these ideas to our code, and we acquired the result as we expected.

We used the tool, packets sniffer, to monitor the traffic. Figure 4-2 and figure 4-3 clearly shows that the unnecessary operations have been eliminated.

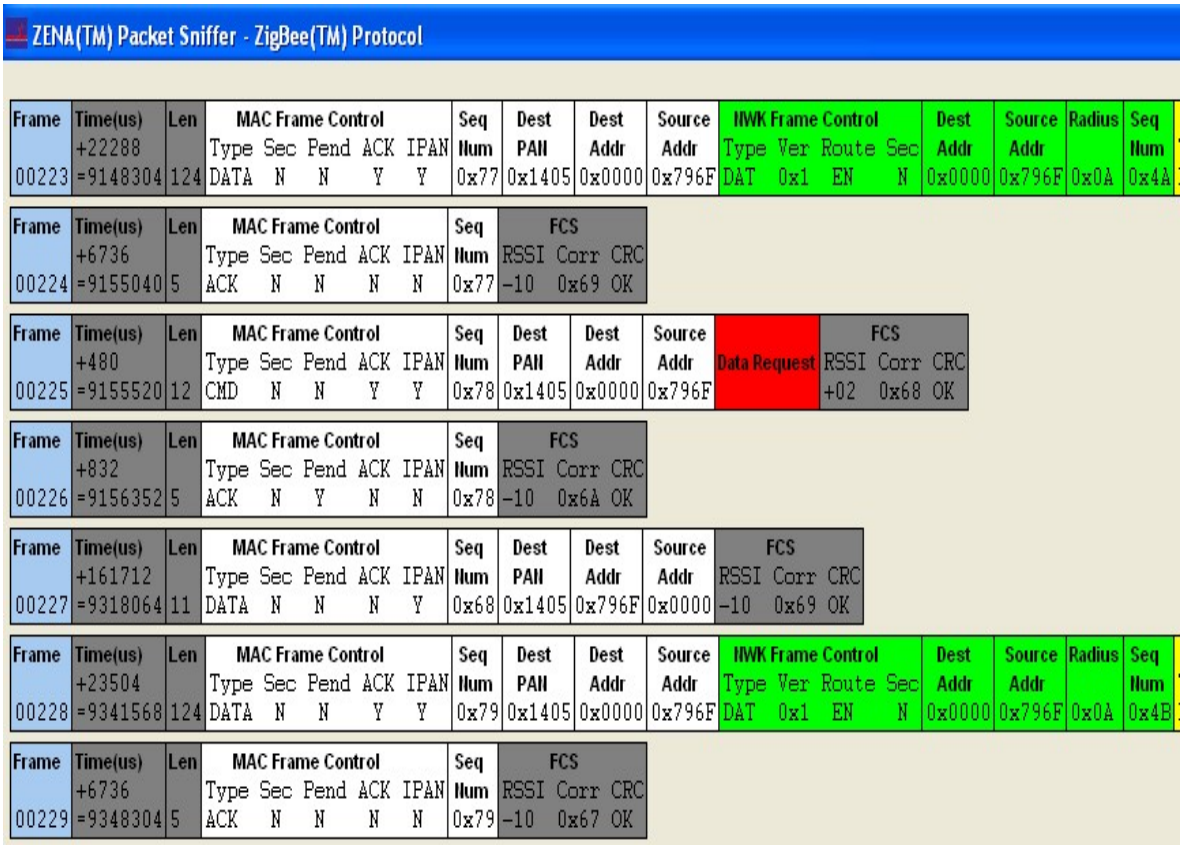

Figure 5-5 Traffic between End Device (Address 0x796F) and Coordinator (Address 0x 0000), before improvement

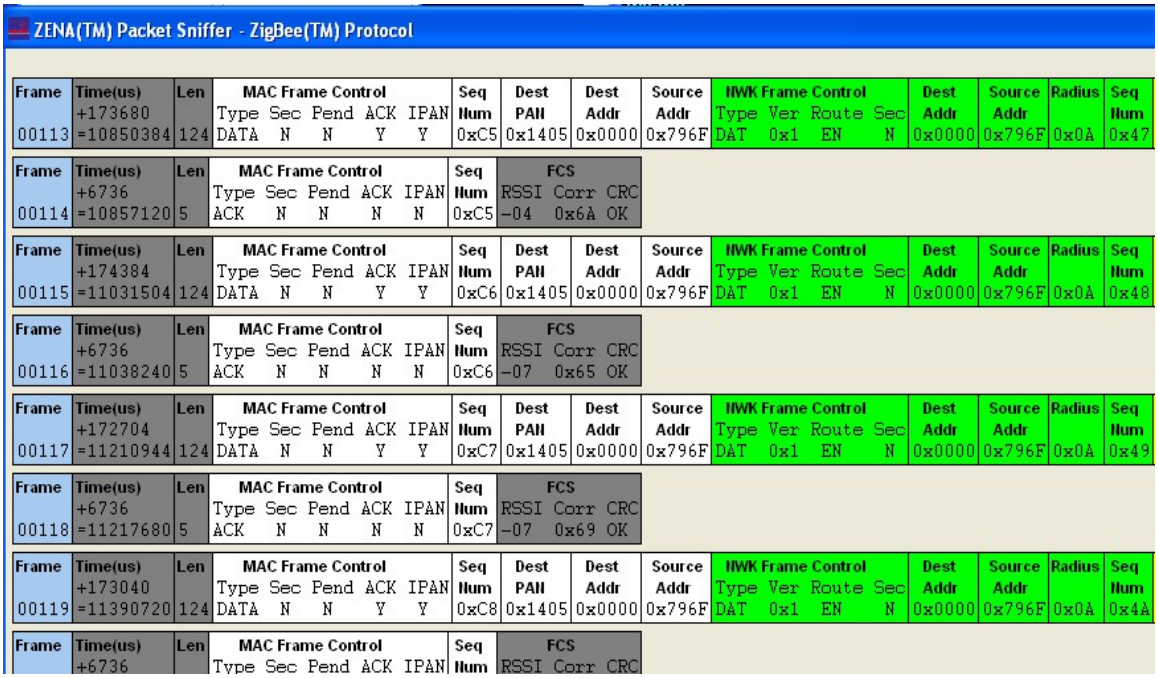

Figure 5-6 Traffic between End Device (Address 0x796F) and Coordinator (Address 0x 0000), after improvement

5.3.3.3 Conclusion of optimizing the operation of ZigBee Stack

We measured the throughput of our current system, which is 42.67kb/s. Comparing to the previous system throughput, 9.37 kb/s, performance of the system has been improved by 4.5 times. In addition, the sampling rate is 2666. This sampling rate can totally satisfy our future applications, pipeline integrity monitoring and container monitoring, which need sampling rate not less than 2000.

5.3.4 Using data compression

Before using compression, each sample takes 2 bytes. When payload size is 96 bytes, we can put 48 samples in one packet.

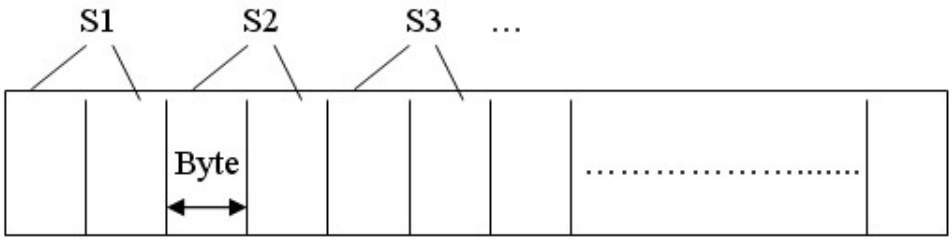

Figure 5-7 Samples in packet without data compression

When we use data compression, the first sample still takes 2 bytes. After that, we put the differences of continuing samples in the packet. Each difference is mapped to 7 bits, with one additional bit for sign, which indicates positive by “0” and negative by “1”.

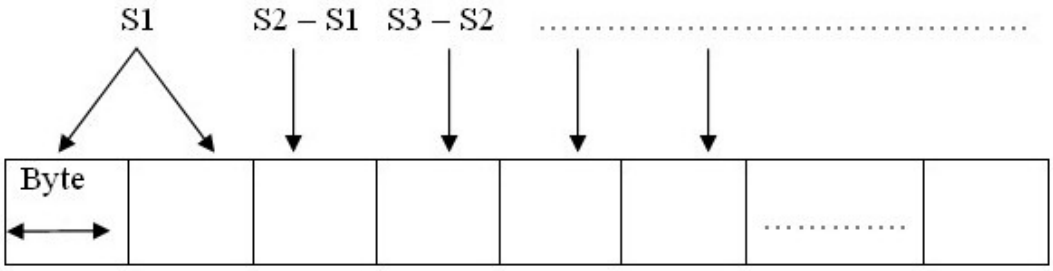

Figure 5-8 Samples in packet with data compression

5.3.5 Increasing the clock rate of the microcontroller

The default CPU clock rate for PIC18F4620 is 16 MHz, with a 4 MHz oscillator crystal. The data sheet for PIC18F4620 indicated that the microcontroller’s CPU can function at rates as high as 40 MHz. As we have tested, with a fast CPU, we get a higher sampling rate.

Sampling Rate	**CPU Frequency**	**Bit Rate of Serial Port**	**Crystal**	**Data Compression**
6520	40M Hz	275,000	10M Hz	Yes
3531	40M Hz	115,200	10M Hz	Yes
2597	40M Hz	275,000	10M Hz	No
1772	40M Hz	115,200	10M Hz	No
1413	16M Hz	110,000	4M Hz	No
233	16M Hz	19,200	4M Hz	No

Table 5-2 Summary of sampling rate at different settings
(*It is to be noted that all the results given in this table employ previous strategies)*

5.4 Saving data to a file

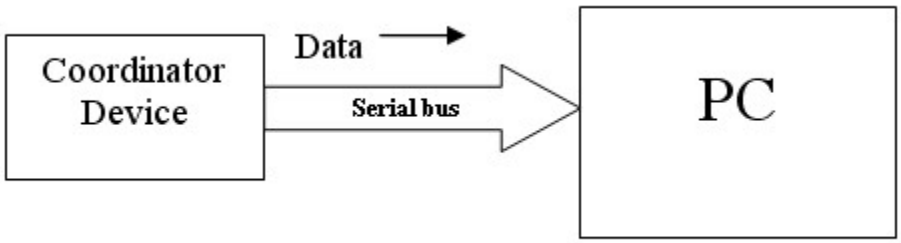

Figure 5-9 Configuration of saving data to a PC file

Microchip's ZigBee device provides us a RS 232 serial port, which can be connected to a Personal Computer to output data to the console. The default baud rate of the ZigBee device's serial port is 19,200, which can afford printing general messages on the console, but it is too slow to satisfy a data rate as high as 41.55 kb/s. Thus we need to increase the baud rate of the Coordinator device. However, PC's serial port's baud rate can not be set higher than 57,600. Thus we use a serial cable, which has a serial to USB converter to connect the PC's USB port. With this configuration, we can set the baud rate of the Coordinator device as high as 115,200, which can afford to save data at sampling rate 3531 with data compression.

. CHAPTER VI

CONCLUSION & FUTURE WORKS

Our research work has provided a wireless sensor network platform for future applications, such as pipeline integrity monitoring and container integrity monitoring, for which a relative high throughput is needed. We have showed and proved that all packets sent by the End device could be successfully received by the Coordinator device at a receiving data rate of 41.55kb/s, which can satisfy a sampling rate of 3531 Hz with data compression or 2597 Hz without data compression.

Besides the Coordinator device and End Devices, ZigBee network also supports a third type of devices, Routers. In chapter 2, we have introduced the topology of ZigBee network, which are star, cluster tree and mesh network topologies. With all kinds of ZigBee devices, Coordinator, Routers, and End devices, we can not only establish a more flexible wireless sensor network, but also a network which covers a larger physical area. In the future work, we could add more Routers and End devices, so as to build a more powerful and robust wireless sensor network.

FERENCES

[1] ZigBee Alliance, www.zigbee.org.
[2] ZigBee Alliance, ZigBee-2006 Specification, Dec. 2006
[3] The Institute of Electrical and Electronics Engineers, IEEE 802.15.4-2003 Specification, May 2003.
[4] Microchip Technology Inc, www.microchip.com
[5] Microchip Technology Inc, PICDEM Demonstration Kit User Guide
[6] Microchip Technology Inc, PIC18F2X1X/4X1X Data Sheet
[7] Microchip Technology Inc, MRF24j40 Data Sheet
[8] Microchip Technology Inc, MPLAB C18 Compiler User Guide
[9] Microchip Technology Inc, MPLAB C18 Libraries
[10] Motive Inc, http://www.moteiv.com/
[11] Crossbow Technology Inc, http://www.xbow.com/
[12] R. Barnett, L. Cull, S. Cox, Embedded C Programming and the Microchip PIC, CENGAGE Delmar Learning

VITA

Xinwei Cai

Candidate for the Degree of

Master of Science

Thesis: TOWARDS ACHIEVING HIGHER THROUGHPUT WITH MICROCHIP BASED SMALL FOOTPRINT WIRELESS VIBRATION SENSORS USING ZIGBEE AND IEEE802.15.4 PROTOCOL

Major Field: Computer Science

Biographical:

Personal Data:

Education:
Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December, 2008

Completed the requirements for the Bachelor's degree in Network Engineering at Zhengzhou University, Zhengzhou, Henan, China in December, 2004

Completed the requirements for the Associate degree in Electrical Engineering at TsinghuaUniversity, Beijing, China in July, 2000

Name: Xinwei Cai Date of Degree: December, 2008

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: TOWARDS ACHIEVING HIGHER THROUGHPUT WITH MICROCHIP BASED SMALL FOOTPRINT WIRELESS VIBRATION SENSORS USING ZIGBEE AND IEEE802.15.4 PROTOCOL

Pages in Study: 39 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study: The research work of this thesis is to create a platform based on Microchip's ZigBee products with high throughput to satisfy future applications, pipeline integrity monitoring and container integrity monitoring.

Findings and Conclusions: ZigBee is a wireless network protocol specifically designed for low data rate sensor or control networks. The default sampling rate of Microchip's ZigBee devices is 233, when payload size is set to 80 bytes. However, our future applications need a sampling rate as high as 2000. Thus, in this thesis, we present five strategies to increase the throughput of the System. The original throughput of ZigBee stack is 3.73 kb/s. After optimizations, we make the throughput of the system as high as 41.55 kb/s, which can support a sampling rate of 6520 with data compression or 3531 without data compression. The performance of the system has been improved at least 15 times than the original. This sampling rate can totally satisfy our future applications, pipeline integrity monitoring and container monitoring.

ADVISER'S APPROVAL: Venkatesh Sarangan