11-28-2012

Design Experiences on Single and Multi Radio Systems in Wireless Embedded Platforms

Somnath Mitra

DESIGN EXPERIENCES ON SINGLE AND MULTI RADIO SYSTEMS IN WIRELESS EMBEDDED PLATFORMS

Somnath Mitra

A Thesis

Submitted in Partial Fulfillment of the degree

Requirements for the Degree of

Master of Science

Major: Computer Science

Acknowledgements

First of all, I would like to thank my advisor Dr. Santosh Kumar - who always wanted nothing short of excellence from all his students, and pushed me to reach that level. Without him this thesis would not be possible. He has guided me and motivated me through all the times of weakness and when I wanted to give up. Secondly, I would like to thank all my committee members Dr. King Ip (David) Lin and Dr. Timothy Hnat for taking time out of their busy schedules to review and advise me on my theses. Next, I would like to thank all my lab members and all the related project associates, members for their time, feedback and their individual contributions to the projects that have made this thesis possible.

Thank you Dr. Prabal Dutta for the countless iterations and emails of advise and support, and teaching me everything you know about hardware and being such a gracious host. The time I spent at your lab at the University of Michigan trying to build the interface board was one of the most intellectually stimulating times of my life. The final hardware designs for Autowitness (Burglar Tracking) and Asset Tracking were developed by Dr. Prabal Dutta. The Asset Tracking alpha coverage algorithm was developed by Dr. Zizhan Zheng and Dr. Prasun Sinha at the Ohio State University, along with Dr. Santosh Kumar. The theft detection algorithm for Asset Tracking and was developed by Bhagwathy Krishna and Dr. Kurt Plarre at the University of Memphis. Animikh Ghosh and Santanu Guha helped in the simulations and testing for Asset Tracking. My friend, philosopher and guide Santanu Guha not only helped me with motivation and tremendous support, but also contributed the HMM tracking algorithm of the Autowitness paper. Daniel Lissner implemented the map matching and several other pieces in AutoWitness. Dr. Kurt Plarre an expert in data analysis, never turned me down for any help I asked him. He made significant contributions to the data analysis and simulations.

The Autosense and FieldStream projects have so many members that I feel I may have

left somebody out. In case I do, kindly forgive me. I would firstly thank Dr. Emre Ertin for being such a gracious host at OSU when I was working to come up with the initial design of the bridge. Also to Nathan Stohs, Taewoo Kon and Siddharta Shah at OSU who entertained every question I had regarding the AutoSense hardware suite they had built. To Dr. Mani Srivastava and his lab members who helped us in the early days of Android Bluetooth to make sense of it all - especially Haksoo Choi. Special thanks to Dr. Asim Smailagic and Patrick Blitz at CMU for their contributions to the software framework. Dr. Andrew Raij for his constant debugging support and other contributions to the framework. Dr. Mustafa Al Absi and his team at the Minnesota Medical School for helping us run the user studies.

I would also like to mention two special mentors - Dr. Suvendu Nath Bose and Dr. Satyajit Saha at the Saha Institute in Kolkata, India who inspired the love of research and academia in me. Last, but not the least, to my friends and family who helped me get to this point.

— Somnath Mitra

Abstract

Mitra, Somnath. MS.
The University of Memphis. MS in Computer Science. December, 2012.
Design Experiences from single and multi radio wireless embedded platforms.
Major Professor: Dr. Santosh Kumar.

The progress of radio technology has made several flavors of radio available on the market. Wireless sensor network platform designers have used these radios to build a variety of platforms. With new applications and different types of radios on wireless sensing nodes, it is often hard to interconnect different types of networks. Hence, often additional radios have to be integrated onto existing platforms or new platforms have to be built. Additionally, the energy consumption of these nodes have to be optimized to meet lifetime requirements of years without recharging.

In this thesis, we address two issues of single and multi radio platform design for wireless sensor network applications - engineering issues and energy optimization. We present a set of guiding principles from our design experiences while building 3 real life applications, namely asset tracking, burglar tracking and finally in-situ psychophysiological stress monitoring of human subjects in behavioral studies. In the asset tracking application, we present our design of a tag node that can be hidden inside valuable personal assets such as printers or sofas in a home. If these items are stolen, a city wide anchor node infrastructure network would track them throughout the city. We also present our design for the anchor node. In the burglar tracking application, we present the design of tag nodes and the issues we faced while integrating it with a GSM radio. Finally, we discuss our experiences in designing a bridge node, that connects body worn physiological sensors to a Bluetooth enabled mobile smartphone. We present the software framework that acts as middleware to connect to the bridge, parse the sensor data, and send it to higher layers of the software framework.

We describe 2 energy optimization schemes that are used in Asset Tracking and Burglar Tracking applications, that enhance the lifetime of the individual applications manifold. In

the asset tracking application, we design a grouping scheme that helps increase reliability of detection of the tag nodes at the anchor nodes while reducing the energy consumption of the group of tag nodes travelling together. We achieve an increase of 5 times improvement in lifetime of the entire group. In the Burglar Tracking application, we use sensing to determine when to turn the GSM radio on and transmit data by differentiating turns and lane changes. This helps us reduce the number of times the GSM radio is woken up, thereby increasing the lifetime of the tag node while it is being tracked. This adds 8 minutes of trackable lifetime to the burglar tracking tag node. We conclude this thesis by observing the future trends of platform design and radio evolution.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Rapid advancement in the fields of VLSI (Very Large Scale Integrated) design, low power MEMS (Micro Electro Mechanical Sensors), radio technology have helped make the term ubiquitous computing a reality. It has enabled several new types of applications, research areas and very interesting real life theoretical computer science problems. Sensor and ad hoc networks are the result of these advances and the effort of system area researchers who build, deploy and study the theory of such networks. In step with these advances in sensor networks and adhoc networks research, the progress in mobile technology has created several new fields such as mHealth and UbiComp, which have made several novel applications such as Zebranet (an application that tracks zebras in Africa), mPuff (an application that automatically detects smoking for body worn respiration sensors) possible, that were almost science fiction a few years back.

Wireless sensor networks can be described as networks built using small, low cost wireless sensors that are preferably asleep most of the time to conserve energy and wake up occasionally to sense, transmit the sensed data and then go back to sleep. Deployment scenarios

of such sensor networks comprise of situations where access to wall power is often not easily available and recharging batteries is hard, mainly due to size of deployment, discomfort to the user, or plain inaccessibility. Wireless sensors often are used in areas or applications where running wires is just not possible - such as sensors on humans to measure their stress in natural environments, or zebras in Africa. The end goal of such applications may be described as one of the two purposes - offline data storage and processing or online actuation. For example in the case of Zebranet, the application needed to track the rendezvous patterns of zebras and download the data to fixed relay nodes. This required ad hoc wireless communication between the nodes mounted on the zebras. In the case of human actuation a compelling case is that of PsychoSocial Stress monitoring application where a real time application process the data from the body worn sensors (ECG, respiration, GSR, Body Temperature), and sends an alarm to a doctor or displays a soothing picture on the mobile phone to alleviate the stress of the user. The common thread in all the above scenarios is the need to communicate with an external entity such as an external network of relay nodes, the Internet or a handheld device. The challenges in building such end to end systems come from a combination of factors which are of hardware, software and algorithmic in nature. There are currently many platforms that can be used by researchers as sensing nodes, but somehow the choices become very limited when it comes to gateways or bridges that have the capability to talk to both the sensor network as well as a back end network. Moreover the cost for many of these systems is still high to even consider buying them in bulk. This is mainly because sensor networks is currently viewed by hardware manufacturers as an experimental market.

The problem of designing, building and deploying such gateway platforms brings forth many challenges such as selection of the platform, or the components required to build the platform - such as radios to select, interfacing challenges for the radios, protocol limitations, constraints such as cost and energy budget. Every project that has a requirement of detecting

events or collecting data and shipping the results to a back end network, require a bridge or a gateway between the sensor network and an external network such as the Internet or a back end server where the data is processed, stored and visualized. We observe that a single radio maybe enough for many applications, but due to the segmentation of radios to market needs and application specifics, 2 radios may be required on the same platform.

In this thesis, we describe the engineering challenges faced while building such dual radio platforms. Once the systems have been built, the system has to be used in a real life application - which has constraints, minimum guarantees, etc. So these systems must be optimized for some parameters such that the constraints are observed and the guarantees are met. We suggest techniques to optimize on the radio's energy usage since radio is the major power hog in such applications, which can also be applied to single radio platforms. The techniques that we suggest can be used for more generic applications, since we use the idea of events rather than periodic duty cycling to wakeup the radio. Now, we describe dual radio platforms and how the challenges that come up in dual radio platforms serve as motivation to our work. Similar problems may also arise in single radio platforms.

1.2 Dual radio platforms

Dual radio platforms are platforms that have more than two wireless radios on board, which can be used to communicate with different kind of networks. Such solutions have been around in popular platforms such as desktop computers and cellphones for some time now. But most platforms don't operate on the stricter constraints, that we see in our applications which may be typical to sensor networks. A survey of dual radio platforms in industry and research will help us understand the kind of problems we aim to solve through this thesis and their motivation.

1.2.1 The rise of dual radios in Consumer Products

Desktop computers until recently came with only one type of connection to any type of network, the standard Ethernet port. The past decade witnessed the rise of wireless standards such as Bluetooth and WiFi, PC manufacturers started selling Bluetooth and WiFi cards, USB dongles so that users could connect their phones and transfer files, contacts, and music from their PCs. WiFi provided users the ease of roaming with their laptops to small distances of the orders of 20 meters and not lose connectivity with the Internet.

Bluetooth enabled several applications such as file and contacts transfer between a cellphone and a PC or a laptop. But as Bluetooth gained popularity users could use their phones as Internet access points if they had data plans on their phones. This enabled users to access the Internet from points that lack WiFi hotspots or wired Internet connections. The essential pieces in this application is the 2G or 3G GSM network, the PC on which the users surfs the web or checks email and the phone that acts as a tether point or a bridge between the PC and the GSM network that provides access to the Internet.

The basis of this functionality lies in the Bluetooth PAN (Personal Area Network) Profile [13] supported by any standard Bluetooth devices. The cellphone in our scenario would be using the Network Access Point (NAP) service provided by the Bluetooth stack, to bridge the GSM network and the PC. The process starts with the PC searching for the NAP device, alerting the user to pick a NAP device if several of them are present. The PAN User device establishes a secure connection with the NAP device. Finally, the NAP device establishes a secure connection with the external network such as GPRS or 3G, and the PAN User device using a protocol called BNEP (Bluetooth Network Encapsulation Protocol) [12] which lets the two devices to exchange packets as if a LAN cable existed between the two devices. This process is described in detail in section 2.5.1 of the Bluetooth PAN profile [13].

But, the advances in Bluetooth bridging cannot be directly applied to the problems

that are faced in bridging sensor network applications to other networks primarily due to differences in the radio. The sensor network research community has traditionally preferred radios that do not require explicit connection setup such as pairing and tethering, which is mandatory if one wants to use the Bluetooth stack and Bluetooth devices. The most commonly used radio in sensor networks is packet based radios such as the 802.15.4 based Chipcon CC2420. Besides limitations of explicit connection setup, there are also restrictions of the number of devices that can be on a Bluetooth network. The Bluetooth network uses a network called the piconet which restricts the number of active devices up to only a seven, other devices can be in the same network but they cannot be active, i.e., they have to be either in parked or standby mode. Moreover, there is a requirement of pre-establishing the roles of each node, such as there will be only one master node and the other devices in the piconet have to be slave devices. This helps Bluetooth to operate at very low interference levels by assigning predetermined time slots of transmission to each node. Bluetooth piconets can be combined together to form larger networks called scatternets. Scatternets suffer from a major problem of lacking good multihop routing, stemming from the fact that each device can act as a master or a slave in a given piconet. This is one of the main reasons Bluetooth were often not the first choice for the sensor network community. Although, Buetel et al. in [10] show that multihop is possible among Bluetooth devices but it requires a particular type of sensor node named the BTNode rev3. This platform lacks the most widely used radio in sensor networks which is the CC2420 radio. This creates a vacuum in the space of bridging sensor network that operate using other protocols such as 802.15.4 or proprietary protocols and Bluetooth based networks or base stations such as PCs, laptops, PDAs.

1.2.2 Dual radios in research platforms

Research in sensor networks for a long time was focussed on single radio platforms, until the landmark paper by Bahl et al. [4] - most systems typically were single radio platforms tailored for particular applications. The work by Bahl et al. provided a general guideline for designing multi radio wireless platforms that can operate in an integrated manner, whereby both the radios can operate simultaneously or in synchrony. The contribution of this work was to suggest 3 basic guidelines for platform design:

- Design for choice - multiple radios should be able to operate independently
- Design for flexibility - multiple radios should be unified by logical abstractions in software
- Design for separation - if possible use different radios to separate control and data traffic

The authors further comment on how these guidelines can be used to design platforms that can enhance robustness, qos, mobility and energy management of wireless platforms. This work has fueled several lines of research by itself. Several research works have looked into different areas of the problem, where dual radios are used for energy management, bandwidth optimization, capacity enhancement, etc. Agarwal et al. [1] proposed an energy management architecture to enable low power VoIP for WiFi enabled smartphones. Pering et al. [37] in Coolspots explored policy management strategies for platforms having Bluetooth and WiFi together.

In terms of hardware used, the research community also uses the Intel Stargate [18] as the gateway node. This node is a comprehensive laptop class device that has several features such as support for multiple radios, various interfacing options to many common nodes such as crossbow and mica motes. Due to limited availability, cost and power hungry nature of

the Intel Stargate we did not consider it as one of the options. There are other dual radio platforms also such as Shimmer, BTNode. The cost of Shimmer mote was prohibitively high for us to use, while BTNode lacked a version that had CC2420 radio on it.

1.3 Motivation for the thesis

Most of these platforms have a second radio that can be used at free will of the implementor since it is an optimization radio. From our experiences we observe that dual radios are often required whenever a bridging scenario comes up. For example, in 2 of our applications - Asset tracking and Psychophysiological monitoring, 2 radios are a requirement due to difference in protocol of the 2 networks to be bridged. In terms of hardware, we had to create a platform with 2 radios that communicated with its respective type of network and the data from one radio served as the input to another radio. This situation takes away most of the optimizations that have been addressed in the literature. Optimizations can be effectively done when the primary radio is of one type that is responsible for data communications. Hence, this has to be broken into two separate pieces. The first is coming up with the requirements, constraints and optimizations for the platform. This can be an abstract document that contains a list of features. The system integrator then follows a series of steps to actually build a working real life system. The system integrators job is two fold - implementing a version of the desired system and optimizing the energy efficiency of the entire system. Laying this out helps us to take another look at the general design guidelines suggested by Bahl et al. in [4].

- *Design for Choice*: What this means is that the radios should be able to operate simultaneously. Although this is desirable but from our experience, this is often not possible - mainly due to lack of serial ports available on the host micro controller, or timing dependancies on the data received from one radio to the other.

- *Design for flexibility*: This requires radio abstraction in software so that networking protocols can access each of them seamlessly. Most radios come with some kind of stack, and as all radios have vastly different properties that a common radio abstraction would either be an overkill to implement or too limited to exploit the characteristics of that radio class. The radio hierarchy in the next chapter, will help us understand the major differences among the major categories of radios. The idea in general seems very logical and we are not against all radios exposing some basic commands in the broad abstraction such as Send, Receive, Power On, Power Off, Sleep - which most radios and radio stacks do already provide. But in the optimization step most of these basic interfaces might not be that useful. Often a lot of effort has to be spent digging deep in the layers of the radio stack, documentation manuals to find the right commands and configurations to implement low power states.

- *Design for separation*: This requieres separation of control and data traffic if possible. Again, the general idea seems very useful if the second radio is free and is not dependent on the other radio. One common scenario is that of the MSP 430 based sensing nodes such as the TelosB where the UART 0 that is exposed is also interfaced with the CC2420 Radio Chip via the SPI bus. The same bus can be used in Async mode as the UART 0. So, if another radio is interfaced using the UART0, clearly they cannot be separated due to the underlying platform dependencies. It is these nuances that make dual radio platform integration a challenge.

In this thesis, we analyze the requirements of 3 real world applications. We design, build and deploy embedded platforms with these requirements in mind. The 3 applications that we discuss here are Asset Tracking, Burglar Tracking and Psychophysiological monitoring of human subjects in behavioral studies. This enables us to look at the requirements of platforms and their radio needs. Several choices are available to the embedded systems

designer designing bridge, sensor and gateway nodes - but how does one go about picking one or more of these radios? What are the base computing cores that are compatible with these radios and also match our requirements? Every applications data, latency, bandwidth, energy and cost needs are unique which makes applying a general set of principles a bit challenging. If a platform has more than two radios, there may be interdependencies of data and timing of one radio on the other. Based on these design experiences we find a common guiding principles evolving. At the core of the principles is a simple framework of understanding the design problems with the help of which, every embedded radio platform design problem cab be framed. The general framework for platform design is outline in the next section.

1.4 A General Framework for Platform Design

Now, we outline a general framework for single or dual radio embedded wireless platforms. They consist of two stages. The first stage is that of defining the constraints, requirements, optimizations which may be considered the task of a Platform Designer. The Platform designer then gives these to a system integrator or a hardware engineer, who actually engineers the system and optimizes them. We outline the steps that the System Integrator should follow in sequence.

1.4.1 The Task of a Platform Designer

- *Constraints*: All applications will have some constraints such as size or form factor, energy available, cost, effort or time required to build. Not all constraints can be satisfied at the same time - so some constraints may have to be relaxed.

- *Requirements*: These are the high level guarantees that the application provides. The

underlying hardware and software together must act cohesively to meet these overall requirements. Requirements can be latency, bandwidth utilization, packet drop rates or signal to noise ratio.

- *Optimizations*: Optimizations are different from meeting constraints or requirements. For example, if the main requirement was alerting when a tag passes and anchor node, and the constraint was the system should last for a day using a 700 mAh lithium polymer battery after it has been stolen, the optimization could be further lowering latency of detection or some other parameters such as the system should last for at least two days after it has been stolen. The optimizations can be non functional requirements, improving the given functional requirements and making further improvements, trying to tighten relaxed constraints.

1.4.2 The Task of a System Integrator

Given that the above 3 key points have been identified for a given embedded radio platform by a platform designer, the System Integrator has to follow these steps in sequence:

1. **Design for Correctness:** The system integrator's primary objective is to ensure that the applications correctness requirements are met. The integrator does not and in many cases should not try to optimize or satisfy constraints before designing for correctness. At this stage, constraints can be relaxed if needed one by one.

2. **Design for Constraints:** Once the applications correctness requirements have been met, the system integrator should try to tighten the constraints that were relaxed in the design for correctness phase. If all of the constraints cannot be met while trying to satisfy correctness requirements - some constraints can be left relaxed to be optimized for later.

3. **Design for Optimizations:** Once the applications have been designed for correctness and all possible constraints have been met (that were possible to meet while still keeping the correctness requirement satisfied), optimize as much as possible. Optimization can be done for functional or non functional requirements, tightening constraints, ease of use, etc.

Now, that we have understood our design methodology, it would be easily comprehensible as how we approach each of our applications and how we use this framework in general.

1.5 Our Thesis

Using these design principles we develop platforms for three applications. We describe the applications with their requirements and our contributions in each of these applications.

1.5.1 Asset Tracking

The first application is that of Asset Tracking. The requirements of this application, was to come up with a system that would help track stolen assets as they travel through the city in a burglar's vehicle. The assumption here was that the city government would provide an infrastructure of detection points or anchored nodes, which would detect the passing tagged asset and finally help pin point one of the possible suspect locations. It has been observed by police departments that the loot from burglaries end up in a few suspect pawn shops - but due to strict legislation obtaining search warrants for such pawn shops is often a problem. So, any type of indication of proximity near such target locations by the anchored nodes would save time of investigation among several suspect locations spread around the city. Also, if tagged items do end up in a pawn shop, they could be localized by wardriving of police patrol cars that are enabled with detection anchor nodes - which provides indication and possible

evidence which further could help make the police department's case stronger when pleading for a search warrant. The main challenges of this system was to design anchor nodes that could be easily deployed and would be able to form a mesh network. The second challenge was to ensure longevity of the tag node after theft and finally ensuring high detection rates at the anchor nodes. The problem of designing an anchor node that would be robust for city wide deployment was an engineering challenge, which we discuss in our design methodology and experiences. The next part was increasing the longevity after the detection of theft using inertial accelerometers on board the tag nodes hidden inside the assets. This required designing sleep schedules that worked well in real life situations. We propose a method that achieved high detection rates in tests on a real life busy city street intersection. Finally, we design a leader election scheme to alleviate congestion among many tag nodes trying to reach an anchor node that have similar sleep schedules. These techniques help us meet our application guarantees of higher than 90% detection ratios at the anchor nodes.

1.5.2 Burglar Tracking

The second application is that of Burglar Tracking. The requirements of this application, was to track a burglar en route post hoc after the event of burglary. This application was more real time in nature in comparison to Asset Tracking. The initial purpose of this application was to try and catch the burglar in the act with the evidence. Over the evolution of the system, the purpose became route discovery to uncover burglar hideouts and finally to philosophical questions such as the psychology of a burglar, and a possible study of common acts of burglars or trends in burglars. Are modern burglars really just a subject matter of Hollywood heist movies or do they make human errors? In a society that is plagued with perils of nuclear weapons and terrorists, burglary is not a minor crime. So, we decided to build a tag node that would be armed with inertial accelerometers and rate gyroscopes and

a GSM radio to communicate with a back end server. The server would process the sensor measurements to probabilistically estimate where the tagged asset is on the street based on distance and turn estimates sent by the tag sensors. This probabilistic estimation was done using an HMM on the map of the city and previous estimates and current estimates. In this thesis, we discuss the hardware challenges faced while building the tag node. Finally, we present optimization techniques that we used to reduce the number of times the GSM radio is switched on to transmit data. We chose against a fixed periodic sending of sensor measurements, if no turn was detected - as it would not add to information gain. We present the data processing techniques used to differentiate between lane changes and turns that helped us reduce the number of times the GSM radio was woken up.

1.5.3 Stress monitoring of human subjects in natural environments for behavioral studies

The third application is that of stress monitoring of human subjects for behavioral studies. Stress can be described as the plague of the modern society. The human genomics project has identified several genes and its relation to human diseases but they account to less than 20% of modern diseases. So, scientists in all disciplines are trying to understand the nature of the modern killer - the stress caused by the rat race of life and complex societal expectations of the 21st century from a human being. As it turns out, there are no good metrics or devices to measure stress. The approach taken is to estimate stress based on physiological parameters and cognitive factors. In order to co-relate physiology with real time feedback on various cognitive factors of the subject, a system had to be built that would connect a non intrusive device such as a smartphone to a body worn suite of physiological suite of sensors. The logical interconnect is a bridge device that we built to connect a 802.15.4 based body area sensor network to a Bluetooth based smartphone device. We describe the design

methodology, the hardware and the middleware framework that we wrote for Android based phone to connect to body worn sensors.

1.6 Organization of the Thesis

Throughout our thesis, we see that there are two main aspects of single or dual radio wireless embedded platform design namely, the engineering efforts, integration challenges and the optimization techniques used to improve system performance or energy efficiency. In Chapter 2, we discuss the hardware, software and deployment experiences of the three platforms mentioned above. We also present a radio hierarchy analogous to a memory hierarchy which helps understand the design space of embedded radio platforms. In the Chapter 3, we present a case for defining good events such as rendezvous with anchored nodes, or turns which can be found by analyzing the application. Such event definitions could be used for other purposes and are generic enough to be reused or modified for other applications. Finally, we conclude with a summary of our integration experiences and optimization techniques. We also comment on the future trends of radio technology and the Internet of small things.

Chapter 2

Engineering and Integration Issues of Wireless Single and Dual radio platforms

2.1 Introduction

Over the years of the evolution of sensor networks research, the community has been introduced to a variety of platforms. These platforms have seen their fair share of successes and challenges. The first set of platforms were called motes which actually means "Speck of Dust". Motes are a byproduct of the Smart Dust vision, laid out by Kristofer Pister at the University of California. The big picture is that with the help of MEMS and VLSI technology, we would be able to build tiny computers the size of dust enabled sensors to sense physical parameters of the environment or a subject of the environment, like the human body. The environments could be as diverse as the battlefield, homes, offices, malls, etc where the motes collaborate to initiate the actuation of some action based on the sensed data. Actuation can be as simple as turning a machine on or off, dimming the lights, or as complex as sending

a mobile robot to survey a region of chemical or biological pollution. Such sensing nodes resembling dust have to be very power efficient running on ambient power or very small batteries, without recharging for long periods of time. The motes would be in a mostly ultra low power sleeping state, waking up occasionaly to sense or transmit data. Although we have made large strides towards the smart dust vision we are not there yet. The fact that the vision is still not a reality leaves enough space for researchers to keep building platforms and testing the boundaries of platform design. Keeping this in mind, we lay out this chapter starting with a radio hierarchy, based on the different categories of radios that have evolved over the years. This provides a high level view of the radio design space. Then, we go into the details of each of the platforms that we have built. We comment about problems along with their requirements, the iteration of designs and the detailed explanation of the design choices we made.

2.2 Radio Hierarchy

Our radio hierarchy is analogous to a memory hierarchy, and has been divided into categories. Each category has a set of features that make it more viable than others for a particular set of requirements. We can make the following general observations about the lower categories:

- Cost is lower
- Radio consumes lesser energy in either one or all of the following Reception, Trasmission or in the Idle State.
- Lower range
- Lower bandwidth

The difference between a memory hierarchy and the radio hierarchy is that there are two variables in a memory hierarchy - speed and cost per bit. As one goes up the memory hierarchy towards the tip of the pyramid - the speed is higher and the cost per bit is also high. It gradually decreases from there onwards. Unfortunately, there are more than 2 variables while considering radios, such as **energy, cost, bandwidth and range**. The radio hierarchy that we present is not a linear gradation of all the variables as one goes up or down the hierarchy, but it may be a linear combination of the variables that change.

2.2.1 Category 1

This is the most popular category of radios used in sensor networks, due to the basic features of the category which are: low power, low voltage, low cost, short range and small form factor. Low voltage requirement enables this class to be powered by smaller size batteries such as coin cell, AA / AAA or Li polymer batteries. Low cost and small form factor also contribute to their adoption in sensing nodes, where cost and form factor are very high priority. Low power helps promote the idea of a mostly sleeping sensor network, that can run for days without recharging. Short range can be viewed as the only Achilees's heel of this class, although several researchers are working on antenna design issues which will help boost the range, but which will come at the cost of increased size due to a larger antenna. This class was the leader in terms of adoption by the research, industrial and hobby community due to all the afore mentioned reasons. We provide a brief tabularised summary of all the major features. The applications that this class has enabled are of different nature starting from Habitat monitoring of birds [34], trees [43], burglar tracking [26] and Psycho Physiological Stress monitoring of human subjects in behavioral studies [22] to name a few.

We discuss three major radios in Table 1 the CC1000 [16], CC2420 [17] from Chipcon, and the nRFAP2 [28] from ANT.

Table 1: Category 1 - Short range, low power

Feature	**CC1000**	**CC2420**	**ANT nRF24AP2**
Current Consumption	RX = 5mA, TX = 6mA	RX = 17.4mA, TX = 18.8mA	RX = 16mA, TX = 22mA
Supply Voltage	2.1 - 3.6 V	2.1 - 3.6 V	1.9 - 3.6 V
Output Power	-20 TO 10 dBm	-20 TO 10 dBm	0 dBm
Band	315 / 433 / 868 and 915 Mhz	2.4 GHz Public ISM	2.4 GHz Public ISM
Form Factor	TSSOP-28 6.4 X 9.7 mm	QLP-48 7 X 7 mm	QFN 5 X 5 mm
Cost	USD 3.00	USD 3.30	USD 1.00
Protocol	ANY PROTOCOL	IEEE 802.15.4	ANT
Interface	SPI	SPI	UART and SPI
Data Rate	76 kbps	250 kbps	1 Mbps RF, 20 Kbps burst throughput
Range *subject to antenna design*	100m outdoors	100m outdoors	50 ft indoors, 125 ft outdoors

2.2.2 Category 2

This category of radios comprises of various Bluetooth standards. Bluetooth deserves a special category by itself because of its unique nature. It has some qualities of the previous category such as low cost, short range, smaller form factor while differing in many ways. Bluetooth devices are all standards compliant devices, that have to be certified by the Bluetooth SIG. They require explicit connection setup with initial pairing at least once. Pairing is a means of authentication by physically exchanging (usually by the human user) and checking a shared key. Bluetooth usually requires higher energy than the previous category with the exception of Bluetooth Low Energy. Bluetooth devices come with a stack of other protocols, such as File transfer, Dial Up Networking, Object Exchange, etc. Devices using Bluetooth are ubiquitous such as watches, cellphones, cameras, etc. Bluetooth SIG has come out with several revisions of the Bluetooth protocol, the major ones being 1.1,2.0,2.1+EDR (Enhanced Data Rate), 3.0, 4.0 + ULP(Ultra Low Power). We do not discuss 3.0 since it is geared for applications that want to use the Bluetooth link as a control channel, such as a paired WiFi connection that uses Bluetooth to reduce high energy beaconing and control plane communication. Apart from these classifications, Bluetooth devices also have power level classifications such as Class I,II or III based on their transmit power rating. We list some popular Bluetooth modules in Table 2.

2.2.3 Category 3

The radio that has revolutionized the way people use the Internet is definitely WiFi. It was meant to be a replacement for the LAN cable, and has pretty much succeeded to do so. WiFi radios have a range of around 20 meters, and very high data rates starting from 1 Mbps to 100 Mbps in some latest chipsets. WiFi is now available in almost all smartphones, laptops and tablets. WiFi's advantage is the availability of fixed infrastructure. Since year 2000,

Table 2: Category 2 - Bluetooth

Feature	**Bluetooth 1.0b/1.1 (PBA 313)**	**Bluetooth 2.0 (BRC46AR)**	**Bluetooth 2.1 + EDR (RN41)**	**Bluetooth 4.0 + ULP (Ultra Low Power)**
Current Consumption	RX - 52mA TX - 44mA SLEEP - 200 uA	RX - 40 mA TX - 50 mA IDLE - 1.4 mA DEEP SLEEP - 30 uA	RX - 35mA TX - 65mA IDLE - 25mA	RX - 15mA TX - 5mA SLEEP - 0.4 uA
Supply Voltage	3.0 - 3.6 V	2.8 - 3.4 V	3.0 - 3.3 V	1.9 - 3.6 V
Range	Short range	+10 m +100m	100m	+150m
Output Power	-4 to 4.5 dBm	4 dBm	15 dBm	-25 dBm to +3 dBm
Band	2.4 GHz Public ISM	2.4 GHz Public ISM	2.4 GHz Public ISM	2.4 GHz Public ISM
Form Factor	10.2 x 14.0 x 1.6 mm	11.8(W) X 17.6(L) X 1.9(H)mm	13.4 x 25.8 x 2 mm	18.10 x12.05 x 2.3mm
Cost	USD 3.00	USD 3.30	USD 1	USD 1
Protocol	BR 1.0	BT 2.0	BT 2.1 + EDR	BT 4.0
Interface	UART	UART	UART	UART
Data Rate	1 Mbps	721 kbps	1.5 Mbps 3.0 Mbps burst mode	1.3 Kbps

Table 3: Category 3 - WiFi

Feature	Roving Networks WiFly RN-171
Current Consumption	4uA sleep, 38mA Rx, 120 mA Tx at 0dBm
Supply Voltage	3.7 - 3.0 V
Range	20 meters +
Output Power	0 to 12 dBm
Band	2.4 GHz public ISM
Form Factor	1050 x 700 x 130 mil
Cost	USD 49.95 in low volumes
Protocol	802.11 b and 802.11 g
Interface	SPI and UART
Data Rate	1 11Mbps for 802.11b / 6 54Mbps for 802.11g

the number of WiFi access points has grown manifold, and services have appeared on the market that collect free and open WiFi access points along with their locations. But, our main focus in WiFi is its use in low power embedded sensor network applications. There are many companies coming out with lower power WiFi on chip modules that are very easy to interface with using 2 power line VCC, GND and two IO/ Communication lines RX / TX. We summarize one such module below - the Roving Networks WiFly RN 171 [41].

2.2.4 Category 4

This category comprises of long range and low data rate modems. These networks are mostly used for long distance communication. This class is a favorite among robotics and UAV (unmanned aerial vehicles) community. Lately, this class of devices has received a lot of interest from the sensor networks community mainly due to its long range, comparative low power requirements (not as low as bluetooth or CC2420 radios), but comparable to WiFi yet providing long ranges. A prime example of this class of radios is the MaxStream Xtend modem from Digi. It has been used to study interaction patterns of Zebras in Zebranet [30]

Table 4: Category 4 - Long Range modems

Feature	**Digi Maxstream 9Xtend**
Current Consumption	TX - 90mA to 730mA RX - 80 mA PIN SLEEP - 147 uA CYCLIC SLEEP - 0.3 - 0.8mA
Supply Voltage	2.8 - 5.5 V
Range	RF LINE OF SIGHT= 0.5 mile to 40 miles with appropriate antennae
Output Power	0 to 30 dBm
Band	900 MHz public ISM
Form Factor	3.65 cm x 6.05 cm x 0.51 cm
Cost	USD 278.00 in low volumes
Protocol	API based operation Supports Mesh networks
Interface	UART
Data Rate	10kbps to 125 kbps

and for creating better opportunistic WiFi contacts in Dieselnet and henceforth reducing latency and improving bandwidth in a Citywide deployment of WiFi nodes that interact with buses [7].

2.2.5 Category 5

This class of devices comprise of all the cellular telephony based radio technologies such as GSM, CDMA, 3G, 4G, LTE, etc. Cellular telephones can be truly called ubiquitous technology mainly due to penetration of telecom providers and wide adoption by consumers everywhere in the world. This has led to a wide array of choices and rapid progress in speeds of data transfer. Another fact compounding the growth of the network bandwidth, is the penetration of smartphones where users are downloading apps, interacting on social networks, playing online games, watching videos and downloading other multimedia content such as music and movies. Again, with this technology, our focus was to find something

Table 5: Category 5 - Fixed infrastructure GSM,3G,4G

Feature	**Telit GE-865 Quad**
Current Consumption	Power off - 62 uA, Idle registered - 1.5 mA, Dedicated mode - 240 mA, GPRS - 420 mA
Supply Voltage	3.22 - 4.5 V
Output Power	2W @ 850 / 900 MHz 1W @ 1800/1900 MHz
Band	850/900/1800/1900 MHz
Form Factor	22 X 22 X 3 mm
Cost	USD 149.00 with interface board
Protocol	API based operation GPRS, TCP/IP stack available
Interface	UART
Data Rate	10kbps to 125 kbps

that would fit the bill of requirements of an embedded platform designer, more specifically a sensor network designer whose constraints are mostly power consumption and form factor. We summarize the Telit GE 865 quad module in Table 5 which is a very small form factor, power efficient module that we have used in our Burglar Tracking application.

2.2.6 Summary of the Radio hierarchy

The radio hierarchy helps us to understand the broad design space that is the available to platform designers and helps. It also helps to sift through various parameters to consider while picking the radios. In our experience, we had to move up and down the hierarchy, as and when required to satisfy the requirements of application. *Our experience has been that if the cost metric and the energy constraint is relaxed, we can pick the radio that has the best range possible as often data rates are not a big concern in many data collection or event detection based sensor networks.*

Given the light of the radio hierarchy and the discussion above, we present the platforms

built for each of the three individual applications.

2.3 Tag nodes in Asset Tracking

The tag nodes in asset tracking had to be small, lightweight, and preferably have lifetimes of the order of years without recharging. They were required to have enough wireless radio range to communicate with the anchor nodes, that would be deployed at street intersections or on utility poles. This made the most obvious choice of picking RFID tags a non option. Moreover, passive RFID tags have very limited to no computing capability. Although there is a whole new class of active battery powered RFID tags, most fall short for two reasons. The first being, the cost of RFID readers is very high. Secondly, the range of RFID tags is not high enough to reach an average distance of 100 feet, which is the diagonal length of a busy intersection in a medium sized city like Memphis, TN. We looked at the available research platforms and even used some of them for prototyping and experimentation, but decided to build our own version of the tag node. We now discuss each of the individual requirements of the tag node and our solutions.

2.3.1 Automatic Theft Detection

The tag nodes needed to have the ability of automatic theft detection, without the owner having to report theft. This meant we needed to have some mechanism to detect movement, and our first choice for this type of sensing was inertial accelerometers. The platforms available at the time that came close to our purpose was the TmoteInvent from MoteIV. This provided a plethora of sensors such as Microphone, 2-Axis Accelerometer, Light and temperature. It had several user input and output features such as speakers,leds and buttons. This would have been sufficient for theft detection if the position of the 2-axis accelerometer can always be aligned with the direction of motion. But, this is not a guarantee, given the

unknown position that the sensor may have to be placed inside the asset. Moreover, the design of this node did not meet the ulta low power requirement for a long lifetime. The was due to the underlying hardware design. The Tmote is based on the popular TelosB platform which sports a 16 bit TI MSP 430 microcontroller, a Chipcon CC2420 radio with expansion ports opening up some I/O lines and ADC channels. The Tmote Invent is a board mounted on the Tmote Sky module, which has the extra circuitry for the sensors and extra peripherals such as buttons, leds and speakers. The problem is that there are no more additional IO or ADC lines left to add any more sensors, such as the third axis of Accelerometer or a low power motion sensor such as Vibratab. So we started using a Tmote Sky which exposes 5 ADC channels ADC0,ADC1,ADC2,ADC6 and ADC7. This enabled us to build a prototype of the tag nodes using an ADXL 335 (A triple axis MEMS accelerometer) breakout board. The ADXL 335 accelerometer has a very low power consumption of 320uA, and a full sensing range of +/- 3g (g = Acceleration due to earth's gravity). The issue with this design was that another interface board had to be designed to house the circuitry, with the required components such as wiring to the expansion port and power supply for the accelerometer interface board. On the software side, TinyOS (one of the most commonly used mote operating systems) provides several abstractions for sampling the ADC. Most of these abstractions are not truly decoupled from the underlying hardware. The interpretation of ADC values to engineering metric units requires dividing the ADC values by a reference voltage [27]. Mostly, such conversion was done offline for us at the prototype phase. This helped us to build a breadboarded version of the 3 axis accelerometer based mote and show that the concept actually works.

2.3.2 Low Power Operation

We quickly realized that even though the power requirements of the ADXL 335 is low, this model of constantly sampling the ADC is very expensive. The CPU is awake most of the time, responding to interrupts from the ADC conversion process. The ADC on the MSP430 onboard the TelosB or Tmote Sky has the capability to do DMA (Direct Memory Access) which stores the sampled value directly in memory. But, raw ADC counts do not mean much in the real world, hence processing has to be done to detect anything significant. This real world requirement of processing raw data into something meaningful, makes all the DMA only technique useless. In XSM Dutta et. al. [20] demonstrated the concept of using hierarchical sensing where analog passive sensors would help detect some base activity level, which would interrupt the CPU and wake it up from deep sleep. Once the CPU is woken up it can start the ADC and begin the sampling of the ADC and run higher power processing algorithms. Our design is inspired from Cargonet [35] where Malinowski et. al., demonstrate similar application of a Vibratab as an analog front end low power passive sensor for an RFID tag. In [35] Malinowski et. al., effectively show that a Vibratab is able to detect jerks and shocks, acting as a low power wakeup circuitry for the microcontroller. *A Vibratab is a cantilevered strip of exible piezoelectric lm encased in polyurethane, with a small proof mass at its tip; shocks set the mass in motion, and the piezoelectric lm transduces the vibrations to voltage*. This seemed sufficient enough for our use case since our target was fixed assets. In our final design we ended up using a Vibration Dosimeter with an integration circuit. The Vibration Dosimeter is a switch that chatters open and close on vibration, and a capacitor gets charged based on the potential generated by the switch which is pulled up to the supply voltage.

2.3.3 Final Tag node design

The final tag node design which assimilated all related work and the prototyping experiences, was done by Prabal Dutta at UC Berkeley. The final sensor board called Irene shown in Figure 1, used the Epic Mote core [21] which is a small integrated core consisting of MSP430 and CC2420. It also housed an onboard 34 mAh Lithium coin cell battery, CR1216.

Figure 1: Final Asset Tracking Tag node Irene

2.4 Tag nodes in Burglar Tracking

We discuss the tag node design in Burglar Tracking next as it is related to, and heavily derives from the tag node design in Asset Tracking. The difference between asset tracking and burglar tracking is that the burglar's route has to be traced in real time without any

existing infrastructure. This called for an on board long range communication medium such as GPRS, GSM or CDMA. A GPS seems to be an ideal fit for the tracking part of the device. Upon deeper analysis, it can be seen that GPS may not be a right fit for this particular application. A GPS device must have clear line of sight with the sky to get a good fix from at least 3 satellites to correctly estimate its own position. The circumstances of a burglary may not grant such luxuries, where the loot may be piled up at the back of a truck. Our experiences show us that there is hardly any connection time or window of opportunity when a GPS module is taken from indoors to outdoors, to a vehicle. The second problem is the metric called Time to First Fix, which is the time required for the GPS receiver to get a first fix after loosing its fix or after a hard reset. This can vary for a commercial grade device such as for an Android based G1 phone, this time can be as high as 10 minutes. This can be a considerable delay in the process of tracking. The final problem is that, an extra antennae required by the GPS module, which cannot be accommodated due to limitations in space. This is the reason we resorted to a map matching technique, where the tag node would estimate distances and turns, send the estimates via SMS or GPRS data to a central server. The server would then take these estimates, and combine then with a map and use a Hidden Markov Model to estimate the location of the tag on the map. Since the initial location of deployment is known, the next series of distances and turns can be used to probabilistically estimate where the tag node is currently. This would take into account the stopping and turning pattern of the tag node and use traffic lights as markers which are unique for each road segment. The HMM would used Bayesian estimation to estimate the transition of states, which are all possible road segments the vehicle can be, given the previous states. Transitions of the HMM are computed when the vehicle makes a turn. Further details can be found in [26].

2.4.1 Low Power Theft Detection

We use the same technique as in the Asset Tracking node of using Vibration Dosimeters as passive front ends to the accelerometer. When there is sufficient movement, such as jerks or walking with the tagged item, the vibration dosimeters generates interrupts to wake up the micro controller. The micro controller wakes up, samples the accelerometer, runs a decision tree based machine learning algorithm to detect theft [26]. The low power theft detection brings down the current consumption of the tag node to around 5 uA on average. This gives us an estimated lifetime of around 3.8 years with no theft detection running.

2.4.2 Distance Estimation

In order to compute distances, we use accelerometer readings and use various noise removal techniques such as Butterworth filtering, and finally use double integration to estimate distances. More detail can found in [26].

2.4.3 Orientation Detection and Angle Estimation

In order to compute turns, we needed some kind of directional sensor. The options for directional sensors are compasses or gyroscopes. Compasses often rely on magnetometers, as their base sensor to calculate heading based on the earth's magnetic field. The problem with such compasses are that in the presence of metals, compasses can produce erroneous output due to hysteresis. So instead we used a gyroscope, specifically the LPR530AL which is a 2 axis gyroscope that can be sampled up to a 100Hz. It can measure angular velocity in the pitch and roll axis and another LY530ALH yaw sensor. This design was inspired from the hobby RC plane community and the robotics community who use this design often. These units can measure upto 300 degrees/sec of angular velocity of each axis. But the problem with rate grade MEMS gyroscopes is that, they suffer from static bias due to calibration or

manufacturing errors. The technique used by manufacturers is to put the gyroscope on a turn table and apply known angular velocities to it, and measure the output of the gyroscopes. This produces an error map for different angular velocities and an equation can be found to estimate the actual angular velocity from the output of the sensor. We obtain these from the datasheets but we also use a simple reset after every estimated turn to get the sensor back to its original state. We discuss the actual turn detection algorithm in the next chapter. Finally we use a simple linear averaging filter to remove noise, and estimate the angle for each second by a single integration.

The final tag node was designed and laid out on a PCB by Prabal Dutta at University of Michigan. The base computing core was again the EPIC mote core [21] using a Li Polymer battery as the power source. A picture of the tag node named Hermes, is shown in the Figure 2. We did not use the CC2420 radio in the actual system, a possible use could be detecting jamming and cloaking of the tag node. Cloaking is the act of encapsulating a radio device inside a Faraday cage, such as metal foil to block all radio communications. This could potentially waste a lot of energy, if the GSM radio was constantly searching for the cellular network, and was unable to connect to one. We now talk about the GSM radio and its integration issues.

2.4.4 Long range communication

The main difference between asset tracking and burglar tracking is that of real time trackability. The only way we could achieve this is by, adding a long range radio such as GSM or Xtend. Xtend requires an infrastructure of such radios to detect the tag node which was not available in the burglar tracking application. We picked the GSM radio, because of its ubiquitous operation. There were some hardware challenges with respect to integrating a low end microcontroller to a GSM radio. Among several GSM radios that we examined we found

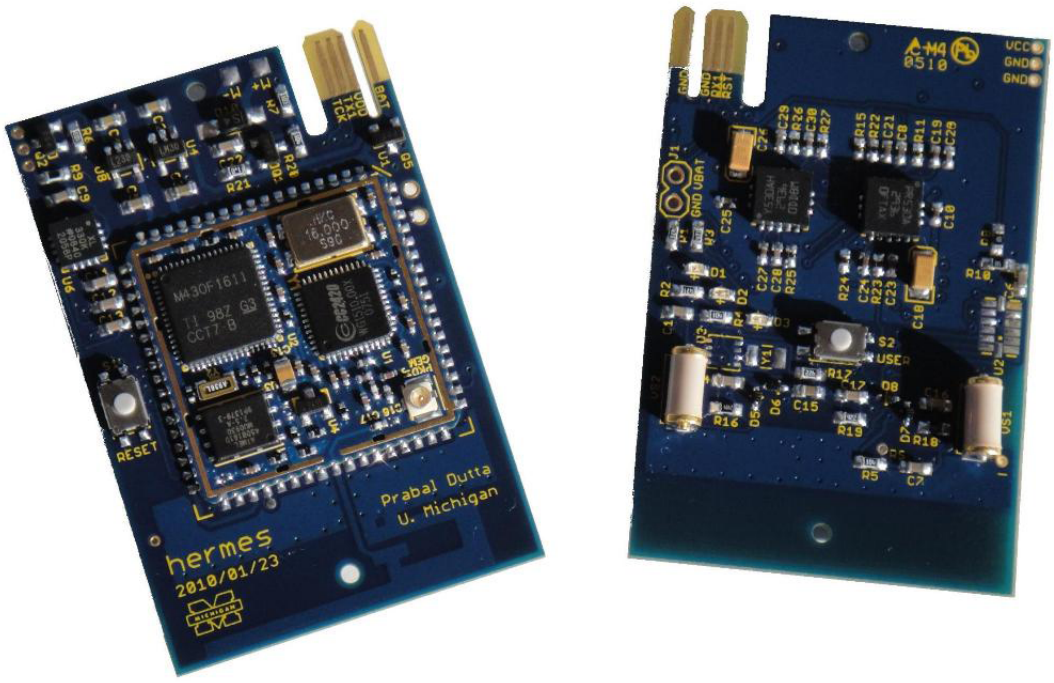

Figure 2: Final Burglar Tracking Tag node Hermes

the Telit GE865 Quad [42] to suit our needs. This was because of low power consumption and small form factor. In order to achieve a working prototype, we used a breakout board from Sparkfun electronics and designed our own interface board. The questions to be answered were, which pins to interface and export. We needed bidirectional communication from the micro controller and the GSM module so, we needed both RX and TX. This would enable sending AT commands, and receiving AT command responses from the module. The EPIC core uses a MSP430 which has 3.3V TTL logic, and the GSM module uses a CMOS 2.8 V logic. So we had to use a voltage divider circuit, to step down the voltage from 3.3V. The TX side from the GSM module to MSP430 didn't need any translation. The other pins that were used on the GE865 were the RESET, ONOFF, PWRMON. We also used a MOSFET to control the power input to the GSM module. The Power Off current consumption is around 62 uA due to leakage current, but with the use of MOSFET we can cut the power

consumption of the GSM module to around 2uA. This helps us to increase the overall lifetime of the node.

2.5 Anchor nodes in Asset Tracking

Asset tracking had the assumption that the hideouts of burglars are a known set of locations, and when stolen assets end up at these or at pawn shops for selling they need to pointed out. Also implicit to this was the assumption, that many cities in the US are now developing city wide public WiFi networks which could be leveraged as a backbone for an asset tracking network. A lot of research has been done in the deployment and system design of public WiFi networks, where there are mobile agents such as people, moving vehicles, or buses with WiFi bricks to provide connectivity to passengers. In [5], the authors seek to enable system support for interactive web applications, to tolerate disruption in WiFi networks for mobile nodes and explore mobile to mobile node routing to improve delays. In [9], Banerjee et. al., explore the various options of delay reduction in an infrastructure WiFi network. The do so by efficiently choosing between adding basestations, relays and wireless meshes as each comes at a price. These techniques along with deployment algorithms such as Trap Coverage [6] can be used to deploy a city wide network of anchor nodes, that would detect and alarm police personnel about the presence of tagged assets. The benefit of using Trap Coverage is that it allows nodes to be deployed incrementally, where at each step of the increment a certain minimum guarantee of coverage is provided. By coverage, we mean any tag passing through that area is bound to be detected. Now we discuss the desirable features of the anchor node along with our design considerations.

2.5.1 Detection of Tag nodes

The anchor nodes should be able to detect tag nodes with high precision. Since our Tag nodes have a CC2420 radio the anchor nodes should also have a CC2420 radio to detect the tag nodes. The ideal solution would be to use a TelosB mote since it is low power, readily available and considerably inexpensive.

2.5.2 Long range Communication

The anchor node needs to communicate the detections of tag nodes to a base station, for reporting to and alarming police personnel. We went over several versions of the design, and finally decided to go with an approach that is similar to [9] where there would be a combination of different types of nodes. Many cities such as Amherst and San Francisco are coming up with city wide deployments of WiFi networks. These networks need to be enabled with some ubiquitous low range radio such as bluetooth or 802.15.4 to communicate with personal devices or tags. In [8], the authors use a PDA class device, the Intel Stargate (which has WiFi support), along with a low energy front end. This low energy front end consisted of a mote attached to a Long Range MaxStream Xtend radio which has a non line of sight outdoor range of around a mile. Buses carrying a WiFi brick attached to an Xtend radios, constantly keep looking for contact opportunities with relay nodes or mesh nodes, to forward their data. The authors use a two tier architecture and mobility prediction to decide which connections should be accepted, so that the number of packets forwarded by the relay node is maximized and energy consumption of the relay nodes is minimized. The first tier of the architecture is a low power front end which consists of a mote and an Xtend radio. The mobile nodes placed in the buses send beacons via the Xtend channel and the mote runs a mobility prediction algorithm that decides whether to take the contact opportunity or not. The parameters to decide is whether the mobile node will come inside the radio

range within the time required to wake up the PDA class device and the WiFi radio. In this architecture the central aid to the mobility prediction algorithm is the fact that beacons can be received from the mobile nodes that have GPS coordinates over a long range Xtend radio link. But since our tag nodes are not enabled with high power long range radios, this was not an option. But we did consider Xtend Long Range Radios for creating a version of the anchor node that would be like a relay or mesh. These relays or meshes may not have internet connectivity and hence will require a long range radio to connect to its neighbors.

We decided on using the Gumstix board [29] which is a small mini computer board that runs on a PXA-270 processor, supports add on modules such as WiFi, display, ethernet and other expansion ports. The Gumstix board is of the size of a small Wrigley's chewing gum pack and is able to run Linux on a micro SD card. This would help us hook up WiFi or any other long range radio such as the Xtend to communicate with relays or other meshes if required or with GSM cards to provide long range links to comunicate with back end networks via the expansion ports or add on modules available from the store. We initially tried to experiment with an even simpler design where a long range radio such as the Xtend would be connected to a TelosB mote. Upon deeper investigation, we realized this design would not work. In our case, the long range radio would have to be attached via the UART 0, that is exposed on expansion port of the TelosB. The problem with that was the TelosB has an MSP430F1611 microcontroller on board which has two USART(A generalized chip for communications which can act both in synchronous or asynchronous mode). One of the USART, the USART1 is used in UART (asynchronous mode) for USB chip communications for programming the mote with new firmware. Rewiring this is a lot of problem and often leads to broken motes, due to the placement of the USB hardware plug near the battery connectors. Often the battery connectors can get disconnected while desoldering the USB connector. Secondly, desoldering the USB connector makes reprogramming hard. We also tried using the UART0 exposed through the expansion header - but the USART0 is shared

between the UART lines and the radio which communicates with the processor over SPI mode of the USART. Multiplexing can be used to arbitrate the lines between the radio and the UART which is connected to the long range radio, but there may be cases where data is coming in from the long range radio for routing and at the same time there may be tag nodes that are being detected. But due to multiplexing, either one can be blocked out. An attack can also be used by the burglar where a jamming CC2420 radio keeps transmitting to the anchor node which in turn always hogs the bus. In such a situation, the microcontroller is unable to communicate with the external network, to even alert the presence of a jammer.

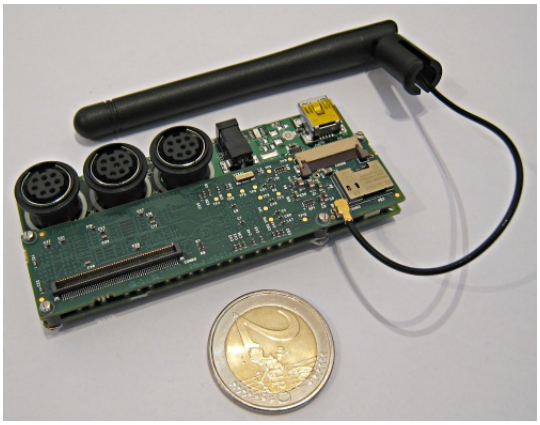

Figure 3: Anchor node: Gumstix board with WiFi antennae

2.6 Bridge nodes in Psychophysiological Sensing in the wild

Psychophysiological sensing is sensing the physiological parameters such as Blood Pressure, ECG, Heart Rate, Respiration rate etc along with psychological parameters such as affect, mood, cognitive load, perceived stress measures. Till recently, it was hard to measure both at the same time due to lack of feasible technology solutions. But, the evolution of body

area sensors that have been reduced to the size and form factor of clothing and chest straps - it has becoming increasingly easier to measure such parameters. A problem that is of great interest to psychologists and behavioral scientists is recording psychological states by making subjects respond to EMA(Ecological Momentary Assessment), upon detecting physiological states such as increased Heart Rate, etc. This requires a robust communication mechanism between the body area sensors and the feedback device such as handheld computer, PDA or smartphone. Autosense [22], brought the idea of human wearable biological sensors coupled with real time feedback collection from human subjects using Android smartphones. This helped behavioral scientists, psychologists, sociologists, drug abuse researchers to conduct a variety of studies ranging from studies of drug abuse, stress monitoring, alcohol intake and cigarette intake monitoring, emotional monitoring.

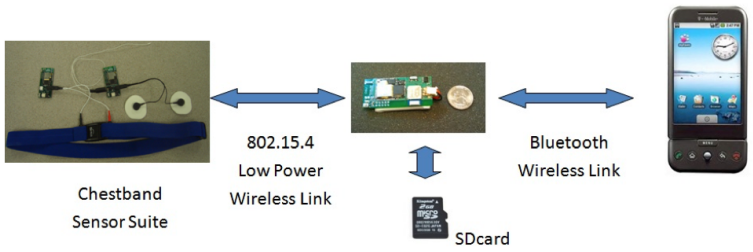

Figure 4: AutoSense Architecture

AutoSense consists of 2 mote class devices and a smartphone and accompanying software framework called FieldStream. The first one consists of a body sensor equipped with Electrocardiogram, Respiration, Galvanic Skin Response, 3 Axis accelerometer, Skin Temperature and Ambient Temperature. The second mote class device is a bridge node that connect the body sensor node to a smartphone via bluetooth. It implements buffering, null packet insertion in case the sensing node doesn't send a particular packet which can be identified by sequence numbers. In this thesis we focus on the bridge node.

2.6.1 Mobility and Connectivity

Sensor measurements (or features derived from them on the mote) can be transmitted wirelessly to a laptop or PC by plugging a mote programmed with TOSBase. Measurements are displayed on the screen using a modified version of the Oscilloscope program and the measurements are logged. A visual interface also allows the study coordinator to provide labels (such as starting and ending times of stress sessions) that can be used during the analysis.

In the natural environment, measurements are wirelessly transmitted on to a smart phone. In body area sensor networks (e.g. Alarmnet [45], Mercury [33], Dexternet [32]), data collected by wearable sensors are usually streamed to a gateway or a data repository. In Alarmnet [45] the data from sensors are streamed to a Stargate gateway from where it is transmitted to a central server over the internet. In Mercury [33], data from sensors is shipped to a laptop. In Dexternet [32], a Nokia n800 tablet is used. Since Nokia n800 allows a USB host mode, a mote plugs into USB port of Nokia n800 and acts as a gateway to the mote based wearable sensors. We stress that connecting to a mobile device carried by the user allows the sensory data to be shipped off to a server on the internet for offline analysis, and processing of the sensory data on the mobile device to infer complex physical or psychological states of the subject (e.g., activity level, stress, etc.) that can be used to initiate real-time actions (fix sensor detachments, solicit self-report, etc.).

To facilitate wireless connection between mote class sensors and a mobile device, three approaches have been been used, First, the mote class devices can be made Bluetooth enabled, as in Mercury [33]) and in Intel mote [36, 11]. This is arguably the most elegant approach, but, Bluetooth radio can drain the battery that is shared with the sensors. Second, a mote with TOSBase can be plugged into the mobile device using USB or another digital I/O interface, as in Dexternet [32], where a mote is plugged into the USB port of

Nokian n800. Similarly, in Alarmnet [45], data from sensors are streamed to a Stargate gateway from where it is transmitted to a central server over the Internet. In [38], a custom board was built to connect a mote called the Intel PSI mote to the SD card interface of the Motorola ROKR E2 (E680) phones. This approach suffers from strong hardware dependency and/or bulkiness for natural environment. Third, a Bluetooth or WiFi bridge can be used that can host a mote class radio as well.

We adopt the third approach, namely adding a bridge node in between the wearable sensors and the mobile device [25]. Adding a Bluetooth gateway node frees up the constraints of the sensor network designer and the mobile device application developer, yet enables the two communities to combine forces in developing human centric mobile sensing applications. A bridge node can help pair up any mobile phone and wireless sensor mote combination without introducing strong coupling requirements. In our design, we connect the BlueTooth serial module BlueRadio CR46-AR to the UART1 lines of the MSP430. We use the UART1 lines on the MSP430 as it is not shared with any other peripherals and does not need any resource arbitration. We use a queueing mechanism on the bridge node to buffer the data from the sensors before sending it out on the Bluetooth link, to minimize packet loss. By using a moderate buffer size of 12, we are able to keep the packet loss rate to 2% on the phone. We also include a digital switch on the bridge to enable duty cycling of the Bluetooth radio.

2.6.2 Middleware for Sensor hardware in Android based inferencing engine: FieldStream

Once the hardware was prototyped the next challenge that we faced was providing appropriate middleware that would parse the incoming data from multiple sensors, handle missing data and send data to higher layers in an intelligible fashion. This requires interpreting the

data coming from the sensor suite, via the bridge. In order to solve this problem we designed a sensor layer for higher layer applications, to receive a buffer of samples to operate on.

AutoSense sends sensor measurements to an Android mobile phone via a Bluetooth-to-ANT bridge. On the phone, the FieldStream software framework robustly collects sensor measurements, processes these measurements to produce inferences about the user (e.g., speaking from respiration, physical activity from accelerometers, and stress from ECG and respiration), and then shares these measurements and inferences with subscribing external applications on the phone and logs them to a local database. Inferences are produced in real-time by the device (approximately 1-2 minutes after sensor measurements are collected, depending on the type of inference), allowing smart-phone applications to be responsive to changes in the body. For instance, self-report can be solicited from participants in a scientific study, if they are detected to be stressed.

Architecture

Figure 5 depicts how data flows through the FieldStream framework to produce inferences. First, the Android OS transports the raw byte stream from the bridge to the framework's **Network Layer** via the Android Bluetooth API. The Network Layer packetizes the raw byte stream and then demultiplexes the packets, identifying which sensor each packet corresponds to. Next, packets are passed to the **Windowing Layer** where they are added to an abstract sensor, a software abstraction of a sensor that buffers sensor data into windows. A window is a buffer of sensor data corresponding to a contiguous block of time, e.g., one minute. When a minute's worth of data is buffered in the window, the window is passed to the **Features Layer**.

The Features Layer calculates descriptors or features of the window using the **Feature Statistics Module** and **Virtual Sensors**. The feature statistics module computes basic statistics of a window, such as mean, variance, heart rate, and respiration rate. Virtual

Sensors compute windows of intermediate features from other windows. For example, a virtual sensor could produce a window of R-peak locations from a window of ECG data, which are then further processed by the feature statistics module into heart rate. Once feature statistics are computed, they are passed on to the **Inferencing Layer**, where inferences are computed from the features.

Communication between the various layers is provided by a set of buses that follow the Observer design pattern [23]: a Mote Bus that passes packets from the Network Layer to the Windowing Layer; a Window Bus that passes windows from the Windowing Layer to the Features Layer; a Feature Bus that passes feature statistics from the Features Layer to the Inferencing Layer; and a Context Bus that passes context inferences to external applications. A logger listens on all the buses, and logs all sensor, feature, and context data for off-line post-processing and validation.

Several mobile phone applications have been built on top of FieldStream. These include an experience sampling program that collects self-reported ratings of stress from the user to validate stress inferences and an oscilloscope application for visualizing bodily indices (e.g., heart rate and respiration rate). Real-time health intervention applications, such as stress management, are under development.

In this section, we discuss the Network Layer only. For discussion of other layers, please refer to [22].

Packet Parsing and Demultiplexing

The network layer receives a multiplexed stream of byte data from the Bluetooth layers. The Bluetooth layer is managed by a BluetoothStateManager class which manages bluetooth connections, and keeps checking the status of bluetooth connections. Bluetooth connections have start, stop commands that have been exposed as interfaces. The Bluetooth connection passes data to a Packetizer class that is modeled using a Producer Consumer pattern using

a Blocking Queue which is a thread safe Queue provided by Java. Multiple sensors are identified using a Channel number on the second byte of the data stream. Whenever data is received for a channel number it is forwarded to its specific sensor via the mote bus. Each of the mote sensors are implemented as subscribers of the mote bus which acts as a publisher.

Handling Data Losses

Packets sent from the motes to the bridge or from the bridge to the phone could be lost in transmission. To ensure proper calculation of features and inferences, the system must be aware of these losses. AutoSense and FieldStream handle lost data as follows. The bridge detects lost packets and inserts null packets in the data stream to replace them. For losses incurred on the Bluetooth wireless channel, they are detected at the phone, using sequence numbers. The FieldStream network layer detects lost packets between the bridge and the phone and inserts null packets in the data stream. This effectively creates windows where some portion of the window corresponds to null packets. Features cannot be computed on these windows directly due to the null packets. Instead, virtual sensors serve as a filter. They decide if there is enough valid data in a window to compute accurate features of interest from that window. If there is not enough valid data in the window (e.g., $< 66\%$), the window is discarded and no further processing occurs over it. If there is enough valid data, the null packets are removed from the window. If necessary, an additional processing stage may also occur to transform the valid data into a derived measurement (e.g., from ECG to RR intervals). The virtual sensor then sends the clean data onto the feature statistics module or onto another virtual sensor for further processing. The use of a virtual sensor as a filter on data losses means that feature computation and inferencing algorithms later in the pipeline do not need to individually implement custom strategies to deal with missing data.

2.7 Key Lessons Learnt

Over the course of the 3 applications we learned several design tricks and things to avoid. We now summarize these.

2.7.1 Dual radio integration on 8 bit micro controllers

Usually 8 bit micro controllers have 1 or 2 serial ports on them such as the MSP430, and often other peripherals are using 1 or both of these ports such as on TelosB both the UART s are used for USB communication and CC2420 radio. Any application using the CC2420 must explicitly request the resource, acquire it from the radio, use it and release it back. One of the solution could be to use a micro controller that has more serial ports. We faced this problem in two applications, Asset tracking and Bridge design for psychophysiological monitoring. The lesson that we learnt was that if two radios are kept on two separate buses, it becomes significantly easier to write code where data is just passed from one radio to another without the need for extensive buffering. This does introduce delays, and maybe some new design strategies can be used such as making direct connections between radios without the need for microcontroller intervention.

2.7.2 Multiple peripheral support

Time and again we realize that in order for a platform to be usable, it should be able to support adding and integrating peripherals. Peripherals can be integrated onto existing platforms via buses such as I2C, UART or SPI. Currently on MSP430 F1611, the most commonly used micro controller on sensor network platforms, use a single chip called USART, which can be operated in Asynchronous mode as a UART or in synchronous mode as SPI. SPI is a technology where multiple peripherals can be attached to a master device as slaves. The master selects the peripheral to communicate via a chip select signal (CS). The actual

communication happens using 2 other lines, while timing is provided from the master using a 3rd line. But essentially they share the same bus, and arbitration is required by the processor. This may introduce delays, and may not be suitable for all scenarios such as in the case of Anchor nodes in Asset Tracking. If more than 2 peripherals have to be attached, that may be operated asynchronously or independently of each other without being dependent on each other for resources such as the bus - we suggest using a core that has better support for more peripherals. Peripheral support not only means hardware support, but availability of drivers and drop in modules. Using a board like Gumstix, has advantages like the ability to run Linux, write C or JAVA programs. Availability of USB host mode on Gumstix enabled us to connect a Tmote Sky/ TelosB mote directly to the platform and run the standard TOSBase that comes out of the box with TinyOS.

2.7.3 Prototyping stage

Every application should be prototyped with breadboarded hardware. This presents challenges in testing and prototyping itself - but helps in reducing manufacturing errors and helps to iterate over the design process. There are several prototyping platforms such as [21] that are available, but many are proprietary and still require a breadboarding process since problems of supplying power and interfacing hardware will have its own challenges. Voltage translations and software issues can be figured out at this stage.

2.7.4 Software support

Software support in the form of libraries is important for any platform. The main problem that we faced was finding the right serial library that can send AT command strings over to the peripheral devices. We used the **printf** library for TinyOS over other libraries, mainly due to support for UART1 of MSP430. Similarly, software stacks are available for Gumstix

such as [40] can be used that come with support for several mote platforms.

2.8 Conclusion

In this chapter, we presented a hierarchy of various radios and how it was used for designing the various radio platforms in our applications. We also went over each of the platforms, and their requirements and the design challenges. In the next chapter we describe how we can address energy efficiency issues for each of these platforms based on events, and deciding when to turn on the radios based on these events.

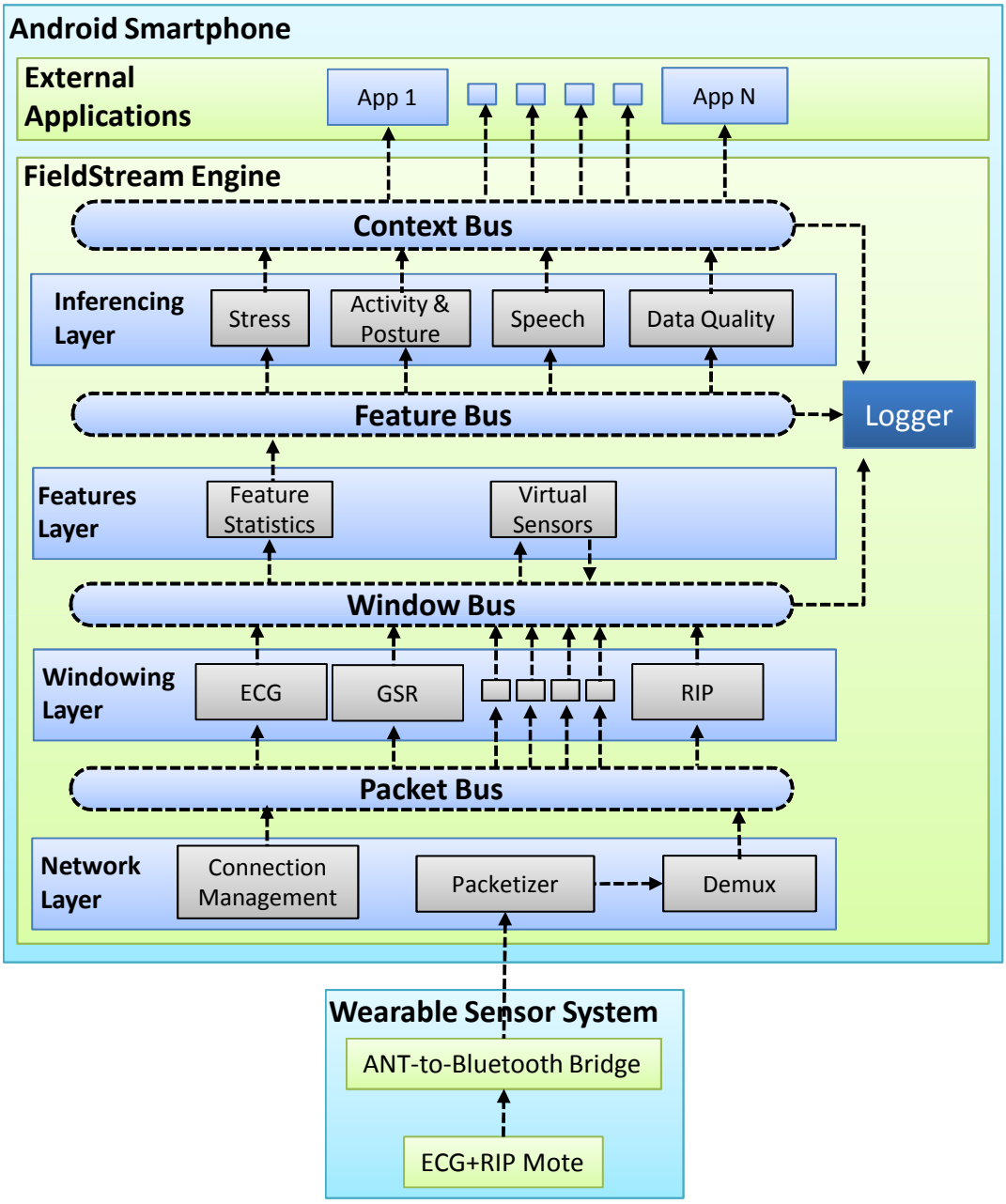

Figure 5: The FieldStream system, including the physical sensors, engine (Network, Windowing, Features, and Inferencing Layers), as well as the Mote, Window, Feature, and Context communication buses. Dotted arrows represent the flow of data through the system.

Chapter 3

Energy Optimization

Energy optimization is one of the most important problems in sensor networks. In all the applications that we mentioned previously energy optimization serves a critical role, more than just an optimization tool. For example in two of our cases asset tracking and burglar tracking energy optimization actually determines the lifetime of tracking and essentially whether the applications purpose is met or not. If the asset tracking node ran out of energy before reaching the pawn shop the application's purpose is lost. Secondly if in the case of Burglar tracking if the tag node ran out of energy and died, we would not be able to track the burglar en route. This makes energy optimization for these two platforms even more important. In most embedded wireless sensor platforms, the radio is the most energy hogging device, mostly due to idle listening costs. Henceforth, most MAC platforms try to optimize on energy wasted in idle listening. In our experiences, transmission and startup of the radio also cost a lot of energy - so it is often important to decide when to start the radio and when to transmit. Now we review of energy optimization techniques in the literature and explain how our event definition technique helps us optimize the radio's energy consumption.

3.1 Energy optimization techniques in the literature

There are several energy optimization techniques that either try to minimize overall energy consumption of the entire system or of a local node. Various researchers have looked at different aspects of energy optimization. In BMAC [39] the authors look at creating a tunable CSMA style MAC suited for a broad range of sensor network applications based on principles such as using software gain control mechanisms to estimate when the channel is free or not, Low power listening interfaces where the length of the preamble of the radio packet for transmitters and the channel sampling time for receivers can be modified to lower duty cycles. BMAC also uses a backoff time, for scheduling transmission when the channel is found to be busy due to network congestion. There are several other MAC protocols but BMAC has found to stand the test of time, and has been a part of the TinyOS MAC since a long time. Most MAC layers in sensor networks try to minimize the time that radio wastes in idle listening since often the idle listening energy is very high as explained in the Category I of the radio hierarchy. The long preamble scheme suffers from inherent problems such as keeping the nodes that are not the intended receiver to be awake for at least half the preamble period on average, before they can go back to sleep. This wastes energy and the long preambles waste a lot of channel capacity and bandwidth. B-MAC falls under the category of asynchronous MAC protocols, which can operate without any global control or synchronization. The other end of the spectrum is synchronized wakeup, where nodes wake up only when data is meant to be sent. These require explicit synchronization between the nodes, which brings the overhead of sending time synchronization messages periodically. For example in SMAC [46], a MAC protocol for sensor networks, uses synchronization messages every 15 seconds or so. Several works have been done after this, that improve upon this work. One example of such a MAC protocol is X-MAC [15]. This scheme uses the idea of strobed wakeup using WakeUp Frames, which are a series of wakeup messages in short sizes,

which cary a sequence counter and the receiver address. This helps non receivers to totally sleep for the most of the period of wakeup sequence transmissions and the entire data frame. The sequence number is a count of wakeup frames till the actual data frame, which enables the actual intended receiver to sleep some more. In between synchronous and asynchronous schemes, there is another class of schemes that estimate the clock drift of receiver nodes based on a mathematical model. UBMAC [24] is an example of this.

Routing algorithms also attempt to minimize energy consumption on a global network level by reducing the number of messages, by using methods like aggregation, suppression, etc. [2] provides a comprehensive survey of routing protocols for wireless sensor networks. The problem of energy optimization has been addressed in the wireless sensor networks in various ways such as hardware level energy optimization, MAC level, routing, energy harvesting. The applicability of these techniques to our problems may not be so obvious. MAC level optimization using synchronous techniques cannot be used as achieving tight synchronization in mobile applications is hard due to lack of globalized control and unpredictable nature of mobile rendezvous. Mobile rendezvous means, one or two mobile nodes can meet each other at any random time. Such applications may use models of mobility or use apriori knowledge of time schedules. In many applications, time schedules may not be so clear and rendezvous can occur any time. The literature has a class of algorithms that deal with such rendezvous scenarios. For example in [19] Dutta at al. propose *an asynchronous discovery scheme, where two nodes pick a prime number each such that the sum of the reciprocals of the primes equals to the desired duty cycle. Each node increments a local counter with a globally fixed period. If a nodes local counter value is divisible by either of its primes, then the node turns on its radio for one period.* There are other solutions to the rendezvous problem such as Quorum [44], which divides time into a grid of mXm slots and uses multipler theorem to find active slots. Kandhalu at al. in [31], demonstrate a 1.5 approximation algorithm for the same problem. There are other techniques that deal with reduction of active sampling. [3]

provides a very detailed survey of all techniques used in Wireless Sensor networks for energy optimization.We believe all the rendezvous based discovery algorithms are very useful. All of the above techniques use the notion of duty cycling. Duty cycling is very useful, but in some applications events can be used to make the radio completely sleeping, until some event occurs. These techniques fall under the broad category of data driven techniques, which use the data to make decisions about when to turn the radio on and when to transmit.

3.2 Event based transmission

Events can be defined as a point in space and time when certain conditions are met. Duty cycling can also be viewed as an event where, a timer expires to indicate the end or start of a time period. Duty cycling helps makes analysis of lifetimes much easier and help derive closed form expressions on lifetime. If the power consumption for each of the activities of a hardware platform given an application is known, the lifetime can be estimated very easily with duty cycling. But our question is can we do more to conserve energy and increase the lifetime of the nodes, by turning on the radio only when data is meant to be sent - and sending only what is required. This leaves out the receiver since the applications that we will discuss have little or no constraints of energy on the receiver, since these are part of the infrastructure. We are going to look into two of the applications - Asset Tracking and Burglar Tracking, and how we optimized energy for them. In the Asset Tracking application we suggest a sleeping till rendezvous scheme, that is coupled with a grouping scheme. Grouping is used to exploit the fact that some nodes may be travelling together as a group, and all of them do not need to waste energy trying to send beacons to alert their presence to an anchor node. This helps us increase the lifetime of a co-travelling group to more than 5 times than an individual per node duty cycle of 5In Burglar tracking we use sensors to estimate when the vehicle has taken a turn, versus a lane change to send the sensor readings to a central server. This helps

us increase the trackable lifetime of the node to twice than a periodic sending scheme. Now we discuss the two individual applications and the schemes used separately.

3.3 Asset Tracking

Once a tag node determines it is being stolen and transported in a vehicle (using accelerometer sensors and machine learning) it starts to send beacon messages to the search for the first nearest anchor node. Once an anchor node has been found, the anchor node responds with a sleep message. This sleep message is an acknowledgement of receipt of the beacon message, as well as a time to sleep. Optimizing sleep messages by itself do not help us confront another problem that we faced while solving, the energy issue. We saw that reliability is another issue when many tags are travelling together and the tags have a small amount of time to beacon the anchor, receive a sleep message and go back to sleep. Making the tags last longer without meeting applications goals do not hold much meaning. Henceforth, we came up with a leader election and grouping scheme to organize the tags into groups. This helped alleviate congestion, increase reliability and also ensure greater sleep times henceforth increasing application lifetimes.

3.3.1 Sleeping between anchors and beaconing

Since the anchor nodes are deployed sparsely, there may be long gaps for a moving tag node in between meeting successive anchor nodes. Significant energy can be saved if tag nodes can sleep in this duration (i.e., enter the timed sleep state). The main issue is how to reliably determine the duration for sleep so that rendezvous with the next anchor node is not missed, while maximizing the sleep time. Each anchor node maintains the travel distance to all of its neighboring anchor nodes in the anchor node network, where two anchor nodes a and b are called neighbors if a tag node can travel between a and b without being detected by

any other anchor node in the network. The minimum time to reach the nearest neighboring anchor node is provided to the moving tag node in the acknowledgement. The minimum time is the time taken to travel the straight line path between the two anchors (although a straight line path may not exist), at the maximum speed limit on any of the road segments on all possible paths on a street map from one anchor to the other. This helps keeps our sleep times bounded and small. Although we want our nodes to sleep, we do not want them to sleep so much that they miss the next anchor node. Another assumption helps us in this regard is that the driver is rational and is not driving at an abnormally high speed through the city. Although one could predict a route, instead of using the distance to the closest neighbor, we opt for the latter since the former provides only probabilistic guarantees, while the latter provides a deterministic guarantee that the anchor node will not be missed. When the sleep timer expires, the tag node again goes back to beaconing to search for the next anchor. In beaconing mode, the tag node transmits a small message and waits for its sleep message back from the anchor. The nodes use BMAC LPL with long preambles to minimize radio ON time in the beacon mode. Now we can discuss the data structure for tag beacon messages and anchor sleep messages.

```
struct TagBeaconMsg1 {
     uint16_t owner_id;
     uint16_t node_id;
}

struct AnchorSleepMsg {
     uint16_t owner_id;
     uint16_t node_id;
}
```

3.3.2 Ensuring detection at the anchor nodes

The application's main goal is to ensure successful rendezvous with anchor nodes that a moving tag node encounters en-route. Although standard networking techniques can be used to ensure reliable communication between one tag node and one anchor node, if several tag nodes traveling together attempt to get an individual response from the anchor node, some of them may not be detected by the anchor node. The reasons for these are:

- *Hidden Terminal effect* - Other tag nodes may be trying to send beacons to the anchors which the tag node may not be able overhear.
- *Congestion* - Many nodes trying to send beacons to the anchor node at the same time will cause congestion at the anchor node, making two things difficult - detecting tag nodes and sending tag nodes the sleep messages.

The above reasons may block the sleeping of the tag nodes, henceforth wasting their energy. The tag nodes may keep trying to sending the beacon messages, failing all the time due to MAC layer back offs. This wastes a lot of energy and reduces chances of detection at the anchor nodes. We cannot do much about hidden terminals, but we believe that we can use some techniques to mitigate congestion. We now discuss our study of congestion when several tag nodes are moving together.

3.3.3 Effect of Congestion on Detection and Sleeping

To measure the extent of congestion, we conducted tens of hours of real-world driving experiments. Fifty tag nodes were carried together in the trunk of a car (at 40 miles/hour), each of which was in the anchor search state, i.e., searching for an anchor node. The number of tag nodes was varied between one and fifty. One anchor node was deployed on the roadside to respond to the beacons it receives with sleep acknowledgements. Figure 6 shows that

when each node transmits a beacon of its own and sleeps only when it receives a response from the anchor node to its own beacon, more than 50 % nodes do not receive a sleep acknowledgement from the anchor node. Even for 10 nodes traveling together some nodes miss the sleep acknowledgement. Detection rates of tag nodes at the anchor node side, do a little better than the sleep message receipt by the tag nodes. We believe that this is due to the static nature of the anchor node. But still the detection rates can be as bad as 60 % when there are 50 tag nodes.

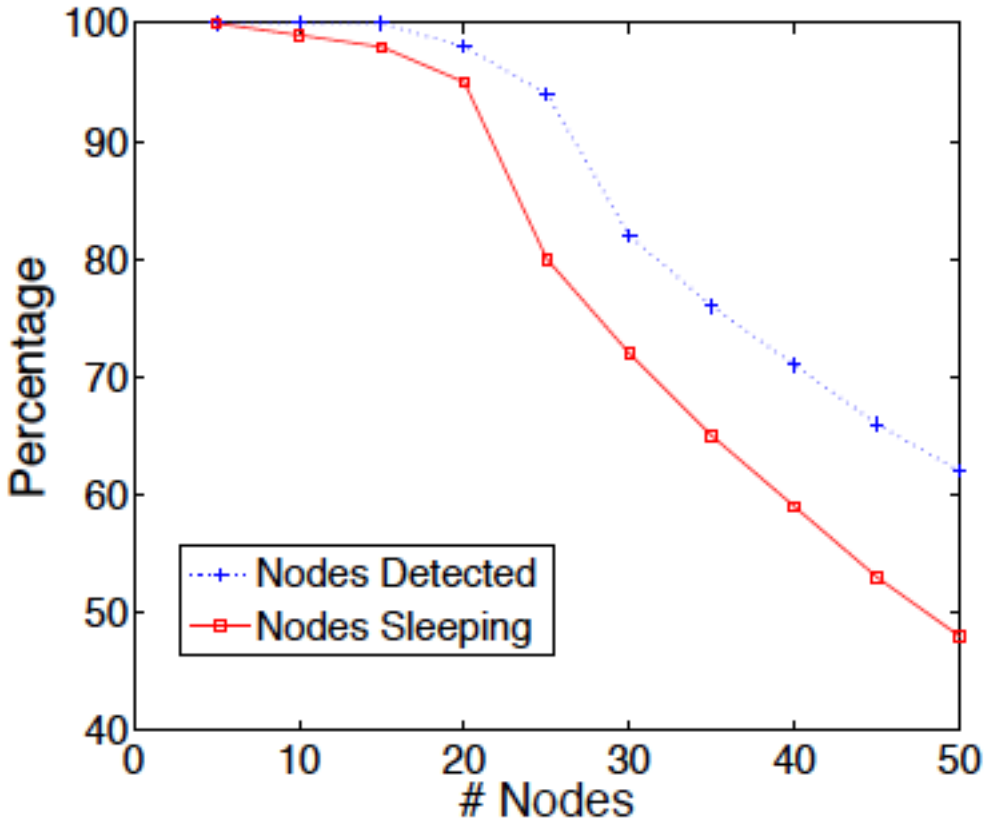

Figure 6: Percentage of nodes (moving together at 40 miles per hour) that are detected by an anchor node deployed on the roadside if they are to be acknowledged individually. The figure also shows percentage of nodes that successfully receive a sleep acknowledgement from the anchor node. The number of nodes traveling together is varied between 1 and 50.

3.3.4 Mitigating congestion by grouping

To mitigate congestion, we organize tag nodes into groups. Groups are tag nodes that are travelling together and may belong to the same owner or establishment. If they start to diverge in their paths, new groups would have to be formed. For example imagine an accomplice of the burglar taking half of the items in a separate vehicle after a certain point - the tag nodes should be able to start beaconing again by themselves.

Each tag nodes id comprises of an owner id and their own node id. In anchor search state, each tag node independently sends a beacon with its owner id and node id. Other tag nodes overhearing this message, chose to suppress their beacon messages while still keeping their radios on duty cycle, for listening to sleep messages. An implicit group leader is elected based on the highest node id. This is similar to the Bully Algorithm in distributed systems. The group leader now starts sending out a new beacon message with the ids of the node it has heard in the beacon msg. The upper limit on the size of a group depends on the number of ids that can be included together in a beacon message. In TinyOS, the size of the payload for a message is 29 bytes which means we can accomodate around 12 members in a group.This is shown in the structure of the new beacon message that is sent out by the group leader.

```
struct TagBeaconMsg2 {
     uint16_t owner_id;
     uint16_t leader_node_id;
     uint16_t group_member_node_ids[size_of_group];
}
```

Once the other nodes overhear and confirm that they are a part of the group, they do not send their own beacon messages. If they do not find their node id included, they start beaconing using the TagBeaconMsg1 structure. This ensures that each node is ultimately

responsible for its own beaconing, in case it cannot be a part of a group.

The anchor node responds to each such message with sleep acknowledgement that contains the group id (a 32 bit version of owner and leader node id combination) and (k1) other group ids from whom it has heard in the recent past. Any tag node receiving this acknowledgement treats this as an acknowledgement for itself (i.e., its group has been detected by the anchor node) if its group id is included in the message, and goes back to timed sleep state. The maximum number of groups that can be allowed due to the TinyOS payload size is 6. As we show next, with $k = 6$ is just good enough to solve our congestion and reliability issues.

```
struct AnchorSleepMsg2 {
    uint16_t sleep_time;
    uint32_t group_ids[max_no_of_groups];
}
```

If a particular tag node is not able to hear the Anchor Sleep Msg within a timeout period, it sends its own BeaconMsg to get a Sleep Msg directly from the anchor node. We set this timeout period to 500 ms.

3.3.5 Estimating the Maximum number of groups to be acknowledged

The maximum number of groups that can acknowledged with a sleep message by the anchor node is 6. But is 6 the right number to be used, can we do lesser than that. It turns out that 6 is the right number as we show with a real life experiment To evaluate the effect of grouping in mitigating congestion and to find the value of k, we conducted an experiment with upto 50

tag nodes (in the anchor search state) carried together in a car, with an anchor node deployed on the roadside. We organized the tag nodes in groups. All groups except for one had 5 or 6 nodes; one group had only one node (in order to observe the effect of a small group). For k, we used 1, 2, and 6. By comparing the blue curve in Figure 3 with the blue curve in Figure 7, we observe that 17% more nodes are able to receive a sleep acknowledgement, if $k = 1$. However, if 20 or more nodes are traveling together some nodes still miss out sleep acknowledgement. The number of nodes missing the sleep acknowledgement is still more than 30% for 50 nodes traveling together (blue curve in Figure 7). Increasing the value of k to 6 addresses the congestion for the most part, even if the total number of groups traveling together is 11.

3.3.6 Real Life Tracking

We evaluate the AssetTrack system on a reallife deployment of five anchor nodes that make for a loop in an urban road network. The anchor nodes were located approximately 1.9 miles apart making for a total loop distance of 9.5 miles as shown in Figure 8. The distance between successive anchor nodes was 2 miles. Eleven tag nodes (organized in groups of 5, 5, and 1) were carried in a car, while the anchor nodes were kept static at the designated anchor node locations (upto 30 meters away from the road). Driving in the designated loop was repeated ten times making for a total of 95 miles of driving over more than 5 hours continuously. The five hours of driving spanned heavy, moderate, and light traffic.

Travel estimate between successive anchor nodes (also called loop segments) was obtained from Google Maps, and provided by respective anchor nodes in response to the beacons they received from tag nodes. The results of our experiment show that out of a total of 550 anchor node encounters (11 tag nodes, 5 anchor nodes, 10 rounds of the loop), no group was ever missed detection by any anchor node. Out of an average travel time of 32.1 minutes to make

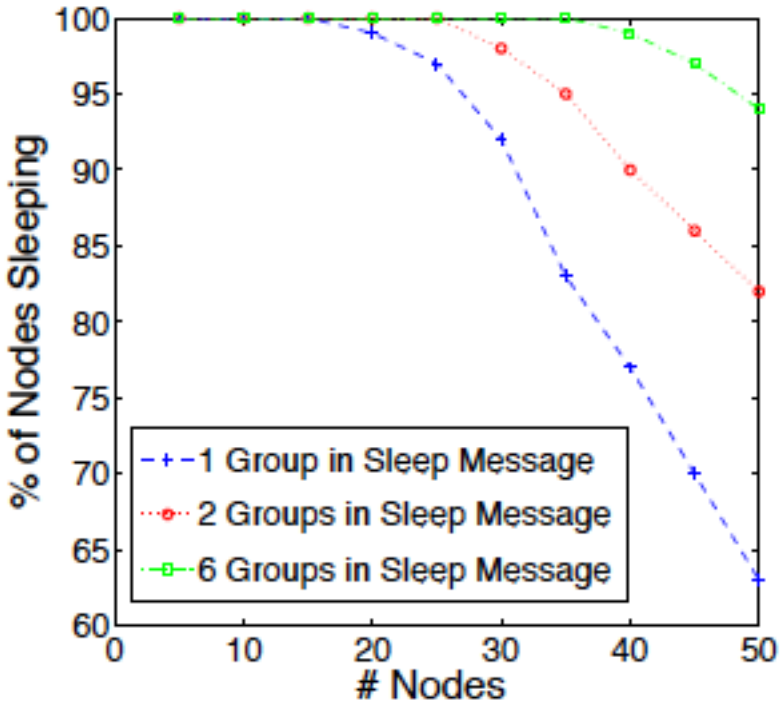

Figure 7: Effect of including multiple groups in a sleep acknowledgement on mitigating congestion as the number of nodes traveling together is varied between 5 and 50. All groups consist of 5 or 6 nodes, except for one with a single node. The figure shows the percentage of nodes (all moving at 40 miles per hour) that successfully receive a sleep acknowledgement during their encounter with an anchor node deployed on the roadside.

one round of the loop, tag nodes spent 26.16 minutes in deep sleep.

3.3.7 Effect of sleeping on Energy Optimization

We now study the effect of sleeping on Energy Optimization. We use the data from the previous experiment and compare it to a node running BMAC LPL with a and a 50 ms check interval. This means that the nodes will check every 50 ms, to decide whether they are receiving a message or not. Each time the radio is awoken to sample the channel it

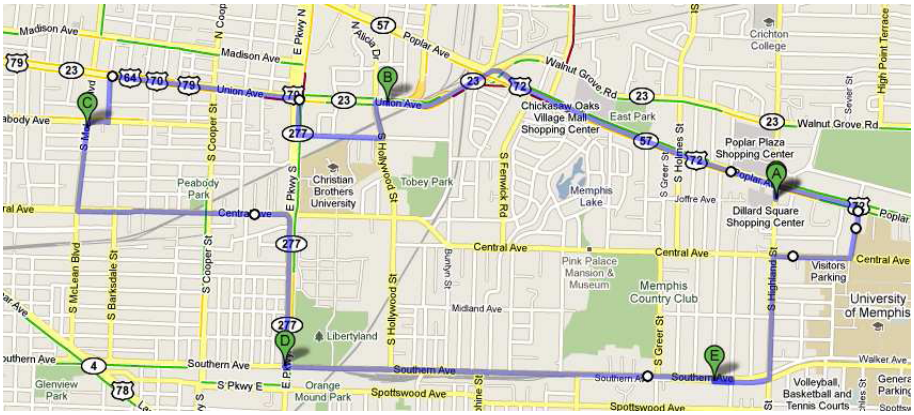

Figure 8: A street map with green markers showing the positions of anchor nodes used for tracking experiments.

consumes 17.4 mA of energy. On top of this we use simple beaconing scheme that sends a beacon message every 1 second, which consumes 18.8 mA which is spent over the time required to send the beacon message. The reason we use 1 second is that the average times spent by us at each intersection was around 3 to 4 seconds. This would give tag nodes at least 2 shots to send a beacon message and expect a response back from the anchor node and yet be fair in terms of analysis.

Given our loop travel time of 32 minutes, which is around 1920 seconds - we come up with a calculation of how much energy the Tag node running our 1 second beaconing and 50 ms check interval would spend. Even if we ignore the time spent due to idle listening, beaconing every 1 second would mean that the radio would have woken up 1920 times, and spent 18.8 mA X 1ms(Approximate Time to send the packet) = 1920 X 18.8 X 1ms = 36.096 mA, which means we would run out of battery in 1 trip with the 34 mAh battery on the final tag node. This is exactly what happened in our real life experiment with the final tag node.

Using our simple travel time estimate based sleep, our nodes in our driving experiment spent 26 minutes out of the 32 minutes in deep sleep. In the remaining time, if we assume

that the nodes spent all the time beaconing every 1 second, the nodes would have spent 7 X 60 X 18.8 X 1 ms = 7.89 mA.

Now if we just divide the two times = $\frac{Energy-spent-in-1s-periodic-beaconing}{Energy-spent-in-sleeping-and-1s-beaconing}$

$= \frac{36.0696mA}{7.89} = 4.57$

which is almost a 5 time improvement in the trackable lifetime of the tag node. Now that we have established the importance of sleeping on tracking lifetime, it is quite understandable the need to get the sleep message across. If the sleep messages are not be able to get across due to congestion - we can use grouping to reduce the number of messages in the network. Once congestion has been taken care of, we need to look at reliability, which we ensure by using group acknowledgement techniques.

3.3.8 System Limitations

The system suffers from problems of travel time estimation, mainly because we use a fixed estimate of speed for each anchor node. This means if the burglar is caught in a traffic jam, or drives really fast - much faster than the speed limits to beat each of the anchor node travel time estimates our schemes would fail and the nodes would start resorting to periodic beaconing scheme. This can be improved by using accelerometers, with directional sensors to properly estimate velocity. The need for directionality is to estimate speed in the direction of motion. This is a hard problem to solve with 3 axis accelerometers alone. The second problem is if the burglar is always taking very long winding roads between the anchors to waste time, and henceforth energy. Other schemes that leverage our grouping scheme and co-ordinated duty cycling to remove idle listening completely and imposing, tight co-ordination between the group leader and the other nodes may be able to solve this problem. But, if there is only one node, we can't really do much.

3.4 Burglar Tracking

In the Burglar Tracking application, we had a GSM radio that requires a 2A current surge to start at least for a second, which makes the startup cost of the GSM radio considerably high. The second problem is that of the transmission cost which is around 240 mA for GSM mode. So if we try to answer the two basic questions of radio energy optimization, we realize that in order to answer the question *When to turn the radio on ?*, it is important to come up with the good definition of an event. For Burglar tracking our event was detecting turns, that is when the theft vehicle has taken a physical turn to another road segment on the road. This was detected using gyroscopes present on the tag node. But the next question that we faced is to define what is a turn and what can confound the turn detector. The answer to that was zigzag driving and frequent lane changing. The next question to answer was *what to transmit*, and for that we already had the answer - distance and turn estimates. In this work, we will only talk about the turn estimation process and how we used the estimates to differentiate between real turns and lane changes. This helped us do slightly better in terms of reducing the number of times we transmit our readings to the central server when compared to a simple threshold based turn detector, that would transmit data upon detecting angles higher than the threshold value.

3.4.1 Obtaining angle estimates

We use the gyroscopes and accelerometers to estimate the angle of change in the direction of movement of the vehicle. In strapdown inertial navigation applications, gyroscopes are used to estimate instantaneous attitude of the object along a fixed reference frame, by a single integration of the angular velocity. The instantaneous attitude along each axis (X,Y,Z) can be represented using Direction Cosine matrix, Euler Angles or Quaternions. We use Euler Angle system to maintain the instantaneous attitude information. Rate-grade low-

Figure 9: Lane Change followed by a right turn

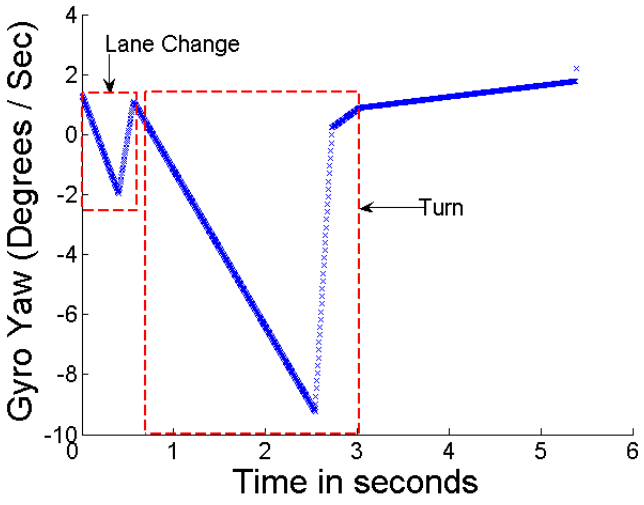

Figure 10: Corresponding yaw measurements

cost gyroscopes generally suffer from two kinds of errorsscale factor error and static bias [14]. The scale factor error that may be introduced due to mounting and placement errors when putting the gyroscopes on the board, can be estimated once for each board using a turn table. Static bias is the output produced by gyroscope when there is no angular velocity

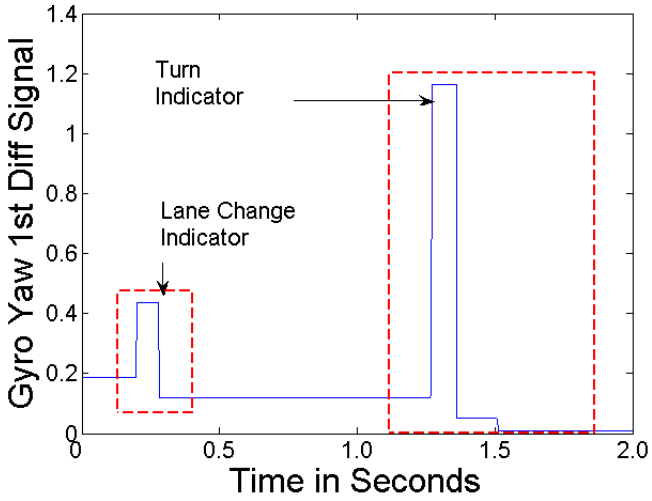

Figure 11: Corresponding yaw first difference measurements

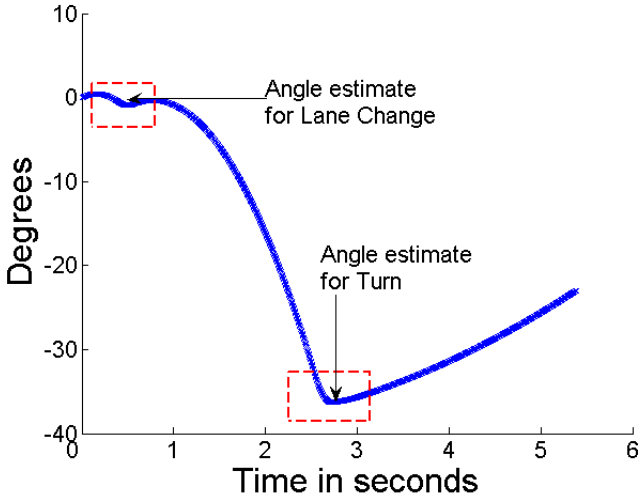

Figure 12: Corresponding angle measurements

applied to it. This value is often not a constant, and when integrated to find the angle of rotation may lead to large drifts over time. We account for this potential error by estimating the drift at every stop. Although we are interested in measuring the change in the angle of heading of the vehicle (for use in map matching), the tag may experience a change in

orientation not only due to legitimate turns and curves in the road, but also due to change in its orientation from the vehicles frame of reference (e.g. due to jerks, or skidding when taking a turn). We use the absolute value of the first difference in yaw readings to detect these changes in orientation. We maintain two thresholds ($D_h \geq D_l$) for amplitude so as to detect a spike in the first difference feature (when it rises above D_h) and we record the time it takes for the first difference to return to normal (when it drops below D_l). This duration is called the activation time. Figures 9, 10, 11 and 12 shows the effect of lane shifts and turns on these two features. We can clearly observe the difference in the two signals. We observed that a lane change is very sharp change that lasts for less than a second, and the overall angle changes are less than 30 degrees.

3.4.2 Effect on energy optimization

We tested our lane change versus turn detector on a 2 mile radius loop, driving 10 times at various traffic conditions during different times of the day. Our ground truth data was collected on an Android G1 phone which has compass and GPS. The number of real lane changes that happened over a single 2 mile loop were around 10 on average, but the number of real turns were 4 left and 2 right turns. The total number of real lane changes were around 102. We observed a pattern, that a turn was often preceded by a lane change. Intuitively, this makes sense, since many roads have a right lane made just for making a right turn into the adjacent lane. Left turns have buffer lanes for big intersections. Moreover, many drivers tend to drive on the middle lane, although we cannot generalize this. On this 2 mile loop we were able to correctly detect lane changes using our simple detector, around 9 times on an average. We compare against a single threshold based turn detector versus our detector that has thresholds as well as activation times.

We observe in Figure 13 that when a threshold based detector is used, with varying

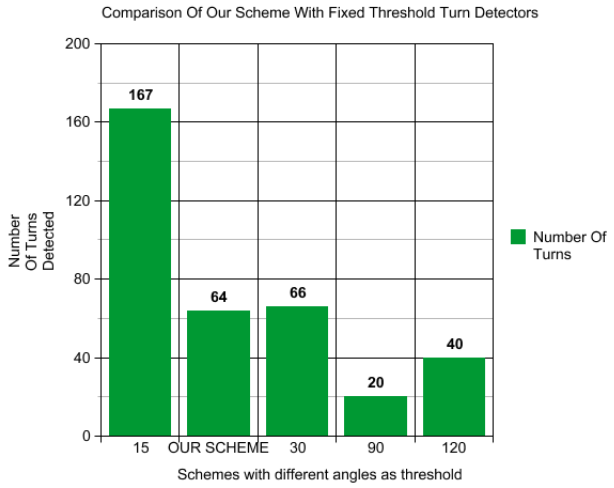

Figure 13: Comparison of our turn detector vs other schemes

thresholds we end up with different types of detectors - the 90 and 120 degree correspond to right and left turn detectors respectively. The 30 degree detector comes close to our scheme, since our scheme terms anything less than 30 degrees as a turn also. But the difference comes in using activation times. The simple 30 degree threshold based detector gets confounded by the act of hopping multiple lanes, whereas our scheme looks at the time domain also to determine, whether this is a lane change or an actual turn. The 15 degrees threshold detector doesn't perform well as it considers everything above 15 degrees to be a turn. All the threshold based schemes as well as ours may be affected by jitters in integration, curvature of the road, instability of the sensor casing and placement and possible vibration of the car. Even though we were able to do just better in 2 cases compared to the 30 degrees case - we believe this approach saved us 4 Ampere-second (Asecond) of energy required to start the GSM radio. This may be just a 3 % improvement, but this 4 Asecond energy could be used to improve sensing. If out of the 4 Asecond, 2 Asecond can be used towards sampling of the accelerometer and the gyroscope which consume less than 4 mA (including ADC, CPU, sensor sampling) - we can add around 500 seconds of tracking lifetime which is

around 8 minutes. The next 2 Asecond, can be used to turn the GSM on and transmit the readings.

3.5 Conclusion

It is often tempting to pick arbitrary duty cycles and using duty cycles as the only metric for energy optimization. But, the basis of duty cycles is only a heuristic, rather than deterministic. If we can examine our application and find the answer to the two fundamental questions that we pointed out in this chapter and then use the general design principles we have suggested earlier, it can be simple to figure out what events may be used to trigger the wakeup of the radio. The events may be deterministic as in the case of the Asset Tracking, where the travel time to the next nearest anchor node was known. Events may also be estimated as in the case of Burglar tracking, where we used gyroscopes to estimate turns.

Chapter 4

Conclusion

4.1 Introduction

In the last 2 chapters we saw the two major challenges that might help us realize the vision for smart ubiquitous sensing and computing applications. But, the real question is are we there yet ? Most of the applications that are currently available are a partial realization of the vision, but the future is bright - and we can safely claim that we are just standing at the helm of a new era. Innovations have to be made in all fields, which makes it even more interesting. In this chapter, we prophesize and look at current trends in the industry and academia to try and come up with a list of possible problems and directions of future research.

4.2 Applications

Just imagine our asset tracking scenario, where every item comes pre loaded with a very tiny tag that will serve various purposes. It will help us interact with the object, and aid us in using the object better or help us find it, or remind us take some action with it. Objects

could be anything such as medicine pill containers that would know how many pills there are lift, and if a patient is off their drugs or not and would remind the patient of taking their medicine. Every package will have some kind of tag attached to them, so that when we open them - usage instructions will show up on our smart phone. An example here would be tools or electronics that we buy and spend time to figure out - when such tools can be better used to create a painting, fix a car, or just set up a WiFi modem out of the box. Every object manufactured would have a kind of tag to help track it across the globe from the time it is created at a factory somewhere in China to the time it is being checked out at the store at a city in the US. Such applications open up possibilities such as better usage of tools, easier navigation through an already stressful life, life saving applications. But they also raise concerns over the level of invasion of privacy such a connected world would cause.

4.3 Evolution of radios

Over the years radios have seen a lot of exciting improvements in range, bandwidth, energy efficiency, protocol. GSM has come a long way and currently seeing rise of the 4G LTE, which basically provides an always on everywhere present high speed data connectivity by means of a higher frequency, higher bit rate radio. With the further improvement of the energy efficiency of 4G, we may be able to keep an always on radio, through which we are sending instantaneous sensing measurements. We will see a variety of radios coming out tailored for different applications - which may cause even more fragmentation with regard to protocols used. Currently we can see two major small range radio technologies evolving and gaining a lot of critical mass, namely NFC (Near Field Communication) and UHF (Ultra High Frequency) RFID radio tags. NFC is mostly a passive technology, where a reader has to be in close proximity to the tag almost of order of 5 or less centimeters. UHF has a bit of higher range that that - of the order of a few feet. WiFi radios will slowly start to reduce in

the power consumption, and maybe perhaps cut connections establishment times whereby WiFi may also be a ubiquitous radio with auto switching roaming capabilities.

Radio protocols will keep evolving to accomodate different type of applications and spaces in which they operate and radio manufacturers will try to find winning products that suit their market positions. But, eventually there are going to be problems also such as cluttering of bandwidth due to very high number of devices co existing in the same geo spatial region. There would be problems of interconnection if devices do not adhere to some kind of rules for interactivity. Henceforth device manufacturers and committees such as IEEE, etc are coming up with standards such as 802.15.4, whereby policies about the protocol structure, etc are going to be made in a democratic scientific manner. Radios will eventually become the size of dust itself, that are highly optimized in energy usage and range to the extent of lasting years and providing high fidelity data and tracking. In all of this, major roles will be played by nano technology, VLSI design and cognitive radio. As more devices get added to the market, there will tough competition between devices for wireless spectrum and media access. Media access policies, frequency hopping policies, will all get tested to their limits very soon. This has fueled off various lines of research such as arbitration of the medium, relocation of bandwidth to unused channels, etc. Policy should also be set for the type of data that can be shared, and what are the privacy risks behind it. Security is also important, and policy makers have to decide levels of security. Public infrastructure has to be developed that can interact with various kinds of radios, which can then be further used by developers to create seamless applications. Already many cities are coming up with WiFi backbones to help provide internet connectivity to their citizens. The need for a good ubiquitous network was felt by us in our Asset Tracking project where we came up with our own version of the infrastructure. If this infrastructure is developed by the government and opened up to developers without the intervention of corporations - it may be the next step required for

the successful realization of the vision.

4.4 Future of Platform Design

If the job of a system integrator is to build better systems that can fit the application's requirements better - we can focus on the more important problems such as the application itself and understanding the data from the application. We envision a system where, the platform designer would understand the application's requirements and would type out the platform in a platform definition language. This tool would help us cut our prototyping time from days to hours, as it would be easily able to tell us whether creating the platform was possible or not. If it is feasible how much would it cost, and what would be the integration effort required. It would also interface to hardware prototyping tools such as VLSI boards etc that would burn the hardware right from the description. But what is truly required is a radio development kit, that would help burn a software radio onto hardware that is integrated with VLSI based hardware prototyping kits - along with our machine description language tool and analog front end sensors. If it is not feasible it would be automatically come up with alternatives. This form of prototyping would help many of today's platform building efforts.

4.5 Conclusion

In this work, we have laid out the state of the art of the radio space with the help of a radio hierarchy. We have shown the use of general design principles to understand the requirements of 3 real world applications namely - Asset Tracking, Burglar Tracking and Psychophysiological monitoring. We discussed various engineering issues that could be reused by other designers. Then, we showed the use the notion of events and how events can be

used to reduce the energy consumption of radios in wireless embedded platforms based on the application scenario. Finally, we hope that these experiences have raised interesting questions that can be solved by the community, to help realize the smart dust and ubicomp vision.

Bibliography

[1] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 179–191. ACM, 2007.

[2] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.

[3] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537–568, 2009.

[4] P. Bahl, A. Adya, J. Padhye, and A. Walman. Reconsidering wireless systems with multiple radios. *ACM SIGCOMM Computer Communication Review*, 34(5):39–46, 2004.

[5] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. Enabling Interactive Web Applications in Hybrid Networks. In *Proc. ACM Mobicom*, pages 70–80, September 2008.

[6] P. Balister, Z. Zheng, S. Kumar, and P. Sinha. Trap coverage: Allowing coverage holes of bounded diameter in wireless sensor networks. In *INFOCOM 2009, IEEE*, pages 136–144. IEEE, 2009.

[7] N. Banerjee, M.D. Corner, and B.N. Levine. Design and field experimentation of an energy-efficient architecture for dtn throwboxes. *IEEE/ACM Transactions on Networking (TON)*, 18(2):554–567, 2010.

[8] Nilanjan Banerjee, Mark D. Corner, and Brian Neil Levine. An Energy-Efficient Architecture for DTN Throwboxes. In *Proceedings of IEEE Infocom*, pages 776–784, Anchorage, Alaska, May 2007.

[9] Nilanjan Banerjee, Mark D. Corner, Don Towsley, and Brian Neil Levine. Relays, Base Stations, and Meshes: Enhancing Mobile Networks with Infrastructure. In *Proc. ACM Mobicom*, pages 81–91, San Francisco, CA, USA, September 2008.

[10] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald. Next-generation prototyping of sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 291–292. ACM, 2004.

[11] J. Beutel, O. Kasten, and M. Ringwald. Poster abstract: BTnodes–a distributed platform for sensor nodes. In *ACM SenSys*, pages 292–293, 2003.

[12] SIG Bluetooth. Bluetooth network encapsulation protocol (bnep) specification. *Revision 0.95 a, June*, 2001.

[13] SIG Bluetooth. Personal area networking profile. *Version 1.0, February*, 2003.

[14] J. Borenstein. Experimental evaluation of a fiber optics gyroscope for improving dead-reckoning accuracy in mobile robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3456–3461. IEEE, 1998.

[15] M. Buettner, G.V. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006.

[16] AS Chipcon. Cc1000: Single chip very low power rf transceiver, 2004.

[17] S.R.F.C.C. Chipcon. 2.4 ghz ieee 802.15. 4. *ZigBee-ready RF Transceiver*, 1, 2003.

[18] INC Crossbow Technology. Stargate. *XSCALE PROCESSOR PLATFORM*, 6020-0049-01 Rev A, 2003.

[19] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 71–84. ACM, 2008.

[20] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 70. IEEE Press, 2005.

[21] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler. A building block approach to sensornet systems. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 267–280. ACM, 2008.

[22] E. Ertin, N. Stohs, S. Kumar, A. Raij, M. al'Absi, and S. Shah. Autosense: unobtrusively wearable sensor suite for inferring the onset, causality, and consequences of stress in the field. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 274–287. ACM, 2011.

[23] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[24] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, and M.B. Srivastava. Estimating clock uncertainty for efficient duty-cycling in sensor networks. In *Proceedings of the 3rd*

international conference on Embedded networked sensor systems, pages 130–141. ACM, 2005.

[25] G. Giorgetti, G. Manes, JH Lewis, ST Mastroianni, and SKS Gupta. The personal sensor network: A user-centric monitoring solution. In *BodyNets*, 2007.

[26] S. Guha, K. Plarre, D. Lissner, S. Mitra, B. Krishna, P. Dutta, and S. Kumar. Autowitness: locating and tracking stolen property while tolerating gps and radio outages. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 29–42. ACM, 2010.

[27] J. Hauer, P. Levis, V. Handziski, and D. Gay. Tinyos extension proposal 101: Analog-to-digital convertors (adcs). *URL–http://www. tinyos. net/tinyos-2. x/doc/html/tep101. html*.

[28] DynaStream Innovation Inc. nrfap2: Single chip very low power rf transceiver, 2010.

[29] Gumstix Inc. Gumstix starter pro pack, 2012.

[30] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ACM Sigplan Notices*, volume 37, pages 96–107. ACM, 2002.

[31] A. Kandhalu, K. Lakshmanan, and R.R. Rajkumar. U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 350–361. ACM, 2010.

[32] P. Kuryloski, A. Giani, R. Giannantonio, K. Gilani, R. Gravina, V.P. Seppa, E. Seto, V. Shia, C. Wang, P. Yan, et al. DexterNet: An open platform for heterogeneous body sensor networks and its applications. In *BSN*, pages 92–97, 2009.

[33] K. Lorincz, B. Chen, G.W. Challen, A.R. Chowdhury, S. Patel, P. Bonato, and M. Welsh. Mercury: A Wearable Sensor Network Platform for High-Fidelity Motion Analysis. In *ACM SenSys*, 2009.

[34] A. Mainwaring, R. Szewczyk, J. Anderson, and J. Polastre. Habitat monitoring on great duck island. In *Proceedings of ACM SenSys*, volume 4, 2007.

[35] M. Malinowski, M. Moskwa, M. Feldmeier, M. Laibowitz, and J.A. Paradiso. Cargonet: a low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asychronous monitoring of exceptional events. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 145–159. ACM, 2007.

[36] L. Nachman, R. Kling, R. Adler, J. Huang, and V. Hummel. The Intel® Mote platform: a Bluetooth-based sensor network for industrial monitoring. In *ACM IPSN*, 2005.

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

[38] T. Pering, P. Zhang, R. Chaudhri, Y. Anokwa, and R. Want. The PSI Board: Realizing a Phone-Centric Body Sensor Network. In *IFMBE PROCEEDINGS*, volume 13, page 53, 2007.

[39] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM, 2004.

[40] A. Reinhardt, J. Hennecke, S. Gottwald, M. Kropff, J. Schmitt, M. Hollick, and R. Steinmetz. Tubicles: Heterogeneous wireless sensor nodes.

[41] INC Roving Networks. Rn-171 datasheet. *ROVING NETWORKS*, RN-171-DS v3.1r 1/27/2012, 2012.

[42] Telit. Ge865 hardware guide, 2012.

[43] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, et al. A macroscope in the redwoods. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 51–63. ACM, 2005.

[44] Y.C. Tseng, C.S. Hsu, and T.Y. Hsieh. Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 200–209. IEEE, 2002.

[45] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, and J. Stankovic. ALARM-NET: Wireless sensor networks for assisted-living and residential monitoring. *University of Virginia Computer Science Department Technical Report*, 2006.

[46] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1567–1576. IEEE, 2002.

Appendix A

GSM Interface Board

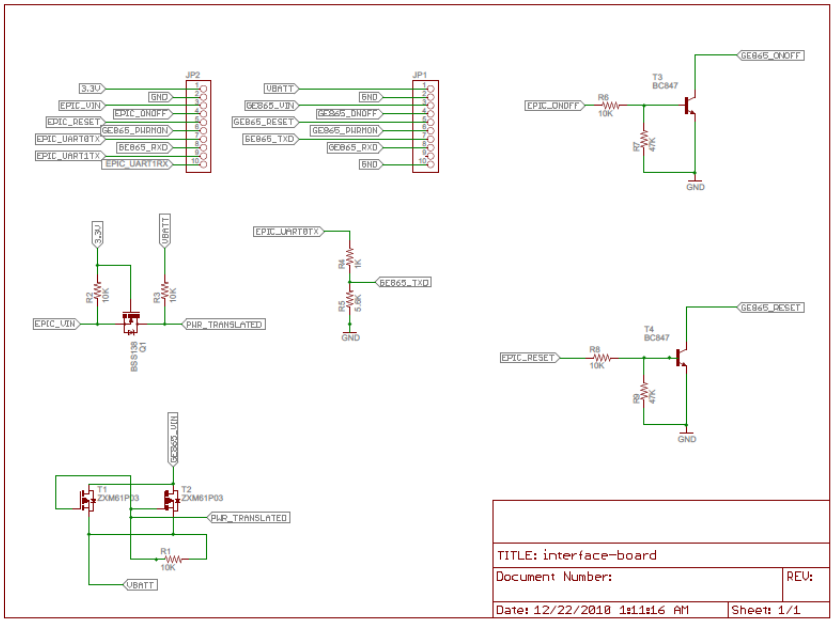

Figure 14: The schematic of the interface board used to bridge the GSM radio and the Tag node in Burlgar Tracking