RELIABLE MIDDLEWARE FOR SENSOR NETWORKS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Mark D. Krasniewski

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2005

This thesis is dedicated to my parents with thanks for all the support and encouragement they have given me.

ACKNOWLEDGMENTS

Special thanks to Saurabh Bagchi for his patience and guidance over the last two years as I have completed my work. I also wish to give thanks to the members of my Advisory Committee William Chappell and Rudolf Eigenmann for taking the time to evaluate my work.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Krasniewski, Mark, D. M.S.E.C.E.. Purdue University, August 2005. Reliable Middleware for Sensor Networks. Major Professor: Saurabh Bagchi.

As sensor networks operate over long periods of time deployed in inaccessible places, each sensor's requirements, location, and reliability may change as they each are subjected to unpredictable influences, both external and internal to the network. As access to these sensors will usually be limited, it is important that when sensor requirements change that there is a reliable, efficient, and fast means of propagating code updates over the network. This work provides a protocol for quickly disseminating data over a sensor network with high reliability and intelligent power management. It achieves energy savings through forwarding of local information on the network scale, providing nodes an estimation of distance from code updates. This energy saving mechanism is further enhanced through directional antenna hardware, which accurately estimates node locations given a few nodes already aware of their locations. This hardware scheme shows much greater accuracy than triangulation with omni-directional antennas.

Coupled with reprogramming the network and locating sensor nodes is the challenge of locating and isolating nodes within the network that function improperly despite software-level updates. To solve this problem this work implements a trust-based protocol that aggregates individual sensor decisions and performance over time to mask out nodes of questionable reliability. Nodes that show continued poor performance are either ignored or removed from the system; the locations of these faulty nodes are easily obtained through directional hardware localization. This scheme proves robust even in cases where half of the network is functioning unreliably.

1. INTRODUCTION

Sensor networks are increasingly becoming a core area of research with potential implementation benefits in the future that have just begun to be realized. In particular, the Berkeley Mote architecture provides a physical real-world implementation of viable sensor networks using commercial off the shelf (COTS) components. These motes already provide interfaces that easily allow new task-specific sensors to be attached while still using the common mote radio module. It is important that once these motes are deployed in a system that they be able to function reliably over time and meet the varying challenges associated with their environment. This work explores various challenges associated with deploying these motes in different environments and proposes several solutions to these challenges, which are verified through both simulation and hardware implementation. In particular, this work explores *in situ* reprogramming of a sensor network, mote localization through directional antennas, and protocols to mask sensors that are faulty or malicious through a time-based trust index.

1.1 Organization of the Thesis

Chapters 2 and 3 discuss the background and related work on network reprogramming and localization in sensor networks. Chapter 4 provides a detailed description of Freshet, a protocol to efficiently and reliably disseminate code throughout a sensor network. Chapter 4 also includes implementation results from Freshet. Chapter 5 discusses localization in greater detail, and presents a solution to localization using directional antennas. Chapter 6 presents TIBFIT, a protocol for trust-based fault tolerance. Chapter 6 discusses the motivation for TIBFIT, its applications, and simulation results.

Chapter 7 discusses the results of all of these applications and describes possible future research.

2. INTRODUCTION TO NETWORK REPROGRAMMING

In this chapter, we provide more detail on the motivation for sensor network reprogramming. In particular, we are interested in the challenges presented in reprogramming a network and wish to classify these challenges to present a viable means of tackling this problem. We also provide a background in the current state of the art in sensor network reprogramming.

2.1 The Basics of Sensor Network Reprogramming

Large scale sensor networks may be deployed for long periods of time during which the requirements from the network or the environment in which the nodes are deployed may change. The change may necessitate uploading a new version of existing code or retasking the existing code with different sets of parameters. We will use the term *code upload* for referring to both these forms. A primary requirement is that the reprogramming be done while the nodes are *in situ*, embedded in their sensing environment. This has spurred interest in remote multihop reprogramming protocols over the wireless link. For such reprogramming, it is essential that the code update be 100% reliable and reach all the nodes that it is destined for. The code upload should be fast since the network's functionality is likely degraded, if not reduced to zero, during the period when the nodes are being reprogrammed. It is also important to minimize the resource cost of the reprogramming and querying for availability of new code. It is conceivable that the process of code upload will be infrequent for many deployments and therefore its resource consumption need not be optimized. However, as has been pointed out in [2], while the cost of transmitting code is high, the cost of periodically transmitting meta-data about the code can also be high. Applications such as Tiny Diffusion [3], Maté

[4], and TinyDB [5], use concise, high-level virtual code representations to give programs which are 20-400 bytes long, a handful of packets. The sensor network environment has inherent unreliability in the network links due to interference, fading, and mobility and unreliability in the nodes which may have transient failures. The code dissemination therefore must be a continuous rather than a one shot process and therefore resource consumption, mainly bandwidth and communication energy, becomes an important issue. This resource cost which is incurred during the quiescent state or steady state of the network must be optimized since that is the dominant phase in the network lifetime. This may be done at the cost of less responsiveness to newly joining nodes.

The underlying model for the class of network reprogramming protocols is that the binary image is to be transmitted to a set of nodes, called the *interested nodes*, in the network. The images have monotonically increasing version numbers. The image is segmented into pages (typical size 1104 Bytes) and each page is sent using multiple packets (typical size 36 Bytes). To start off, there are only a few sources of the binary image, e.g., base stations located at ends of the sensor field. The code progressively ripples through the network with the exchange happening between neighbors through a three way handshake of advertisement, request, and actual code transfer. The advertisement and the request will collectively be referred to as meta-data. The meta-data is typically much smaller in size than the data (the code) and is used to suppress redundant data transmission.

2.2 Related Work in Sensor Network Reprogramming

The field of network reprogramming in the large scale wired distributed systems has focused on the problem of reliability and efficient utilization of bandwidth. For example, [9] provides methods for efficiently computing increments to the update. They have not dealt with resource constraints on the nodes themselves. Due to the wired environment, the solutions do not have the ability to leverage overhearing neighbor communication.

In a large scale wireless network, data dissemination through unregulated flooding using broadcast by each node is known to cause a broadcast storm [10], thereby limiting the scalability of such a solution. Hence, researchers have proposed randomized tree based multicast protocols with the source at the root of the tree, receivers at the leaves, and intermediate nodes responsible for local recovery at the intervening levels of the tree. Scalable Reliable Multicast (SRM) [11] is an important protocol in this class. In SRM, when a member detects a message loss, it initiates a recovery procedure by multicasting a retransmission request in the local region. Any member having the desired message in its cache responds by multicasting the message with a back off mechanism being used to prevent redundant requests and replies. Further scalability in unreliable environments, such as ad-hoc networks, can be achieved by epidemic multicast protocols based on each node gossiping the message it received to a subset of neighbors [12]. This class of protocols gives probabilistic guarantee for the update to reach all the group members. The probability is monotonically increasing with the fanout of each node (the number of neighbors to gossip to) and the quiescence threshold (the time after which a node will stop gossiping to its neighbors). By increasing the quiescence threshold, the reliability can be made to approach 1, which is the basic premise behind all the epidemic based code update protocols in sensor networks.

The push-pull method for data dissemination through the three way handshake of advertisement-request-code has been used previously in sensor networks with sensed data taking the place of code. Protocols such as SPIN [13] and SPMS [14] rely on the advertisement and the request packets being much smaller than the data packets and the redundancy in the network deployments which make several nodes disinterested in any given advertisement. However, in the data dissemination protocols, there is only suppression of the requests and the data sizes are much smaller than the entire binary code images. Freshet borrows ideas from hop-by-hop NACK based error recovery protocols proposed for wireless sensor networks (WSNs), such as PSFQ [15], Garuda [16], and RMST [17]. This class of protocols performs local recovery typically within one hop using selective NACKs.

There are four major sensor network reprogramming approaches that have appeared in the literature. TinyOS [18] includes limited support for network programming via XNP [19]. However, XNP only operates over a single hop and does not provide incremental updates of the code image. The Multihop Over the Air Programming (MOAP) protocol extends this to operate over multiple hops [6]. MOAP introduced several concepts which are used by later protocols, namely, local recovery using unicast NACKs and broadcast of the code, and sliding window based protocol for receiving parts of the code image. However, MOAP does not leverage the pipelining effect with segments of the code image. The two protocols that are substantially more sophisticated than the rest are Deluge and MNP. Both use the three way handshake for locally propagating the code. Deluge is a protocol with a similar design approach to ours [1]. It segments the binary code image into pages and pipelines the different pages across the network. It builds on top of Trickle [2], a protocol for a node to determine when to propagate code in a one hop case. Deluge leverages overheard advertisements or requests to decide when to create a new advertisement or send a new code update. MNP is a more recent protocol [7] whose design goal is to choose a local source of the code which can satisfy the maximum number of nodes. The authors provide a detailed algorithm for sender selection using the number of requests seen by a sender as the key parameter for the selection. They provide energy savings by turning off the radio of all the nodes that are not selected as the sender.

3. BACKGROUND ON LOCALIZATION

In this chapter we discuss background information and detail on sensor network localization. For localization we focus on different techniques and means of localizing sensor nodes depending on both the number of nodes with already determined locations and the types of antennas that the nodes use.

3.1 Introduction to Localization

Sensor networks provide a promising infrastructure for gathering information about parameters of the physical world. Tiny wireless nodes equipped with different kinds of sensors can be distributed over a field and can collect and transmit the data to a data aggregation point, such as a cluster head or a base station. In order to interpret the sensed data, it is often necessary to know the location of the node which is the source of the data. In addition, position information is valuable for optimizing the routing process, as shown in many position aware routing protocols [23][24]. In general, these strategies seek to avoid wasting valuable bandwidth by minimizing the control traffic for route determination. Most position-based routing schemes also remove the need to maintain routing tables at the nodes. Also, the node's location may change. Mobile sensor networks are becoming an important class in which the nodes may move in a controlled manner or through passive mobility.

It is possible for a node to have up-to-date information of its location if it contains location determination hardware, such as a GPS receiver, mounted on it. However, from an economic standpoint, this would violate the requirement for the deployments to be cost-effective. The economic considerations have been driving the cost of the individual sensor nodes down to the point where sub-$1 nodes are beginning to look achievable

[26]. GPS hardware would increase the price of sensor networks substantially. Commercially available GPS receivers come in a wide price range of \$10-\$10,000. The receivers at the lowest end give poor accuracy, with inaccuracies of tens of meters possible [27]. Receivers that give sub-meter accuracy, which may be needed for many sensor applications, are more than \$5,000 in price. The hardware also adds to the weight of the unit with typical receivers ranging upwards of 5 oz. Finally, the battery lives' of the receivers are much shorter than that of the sensor nodes themselves, e.g., tens of hours for the typical GPS receivers compared to multiple months for a representative sensor node, the Berkeley mote. Thus, the combined unit of the sensor node and the GPS receiver will have to be replaced far too frequently for it to be practicable for a large class of deployments. More generally, the received signal strength for a GPS can be as low as -130 dBm, orders of magnitudes less than the strength of traditionally received signals in terrestrial applications and lower than the sensitivity of receivers on typical sensor nodes (-100 dBm for Berkeley motes). Therefore, expensive receivers would be needed. Also, since relatively unobstructed views are required for GPS localization, in many sensor network deployments, the GPS measurements would need to be supplemented with ranging data from the local network.

Though it may not be feasible for all the nodes to be equipped with special purpose location determination hardware, it may be possible to equip a small fraction of the nodes in the network with such hardware. Such nodes, called "anchor nodes", can act as reference points for location information and other sensor nodes, called "target nodes", can use information from anchor nodes to estimate their location. In the most commonly used technique called *lateration* the distance measurements are required from $(k+1)$ neighbors in a k dimensional plane. The example of lateration in a 2-dimensional plane is called *triangulation* in which the sensor node needs to know the distances from three neighboring nodes. Several approaches exist for estimating distance from a neighbor, e.g., signal attenuation and time of flight. In signal attenuation, the power of received signal is measured by the sensor node and knowing the signal strength emitted by the source node and the attenuation relationship with distance (such as, $1/r^2$ where r is the separation distance), the relative distance can be calculated. Typically for indoor environments or large distances, the attenuation relationship becomes complex and difficult to represent concisely due to multi-path effects and reflection of the radio waves. Other techniques for measuring relative distances, such as time of flight ([34],[28]), are less useful in our environment since the radio signal travels at the speed of light and the distances traveled for signals by the sensor nodes are relatively short.

Directional antennas provide important benefits in sensor networks. Directionality can be used as a form of diversity built into the sensor node, which helps in coping with the variability in the communication channel and reduce the link error rate. The directionality provides increased transmission ranges compared to omni-directional antennas by focusing the transmission energy in the desired direction. They can also increase the security of communication by restricting the set of neighbors that can overhear a communication [37]. Directionality in expensive communication systems is commonly achieved through the creation of a phased array. However, this is extremely expensive and is used predominantly only in high cost military applications. In addition, it is required that the elements of the phased array be an appreciable fraction of a wavelength apart. This would not be possible in electrically small form factor sensor nodes. This precludes the use of a traditional array to provide the desired beam scanning. However limited directionality can be cheaply integrated into a small form factor sensor node. In this investigation, reduced size patch antennas have been developed using standard patch arrangements with high dielectric constant antennas. Multiple directional antennas are utilized and a simple switching network enables us to switch between polarization states ([29],[30]) and the direction of radiation.

The solution to location determination with omni directional antennas is *not* applicable to directional antennas since the radiation patterns are different and the received power is dependant on angle as well as distance. In this paper, we use a model for sensor nodes equipped with four directional antennas. Directionality provides relative angle measurements between anchor nodes and target nodes with unknown positions and has been argued to improve localization estimates [52].

3.2 Related Work on Localization

Triangulation is a common method for locating objects using other objects which do know their position. This is an applicable model for our environment where the positions of some sensor nodes, possibly equipped with GPS receivers, are known. A nice overview of triangulation based location determination techniques is to be found in [32] and [33]. The triangulation techniques can be sub-divided into two categories – lateration, which uses distance measurements, and angulation, which uses angle measurements along with distance.

If individual distance measurements are completely accurate, lateration requires *(n+1)* neighbors with knowledge of location to pinpoint the target node in an *n* dimensional plane. An example of lateration in two dimensional space is shown in Fig. 3.1(a) and is called triangulation. Example use is in the Active Bat Location System [34].

Different approaches exist for estimating the distance from a neighbor, for example time of flight, attenuation of signal strength, and directionality ([33],[35]). Measuring signal strength relies on the property that radio waves attenuate in their signal strength with increasing distance between the transmitter and the receiver. The receiver can calculate the distance if it knows the transmission power and the attenuation model [36].

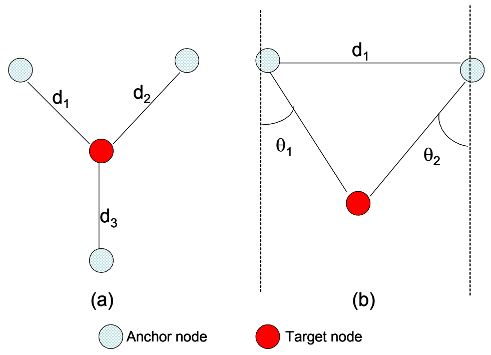


Fig. 3.1 Location determination with neighboring anchor nodes. Lateration is in (a) and angulation in (b)

The attenuation is often modeled as $1/r^2$, where r is a relatively short distance outdoors. Indoors, reflection, and multi-path fading make the model and hence, the location estimate, inaccurate. The third way of estimating location is to compute the angle of each reference point with respect to the sensing node in some reference frame. The position of the mobile node can then be computed using angulation.

In practice, the individual distance measurements are inaccurate because the exact relation between the measurement of physical properties, such as signal strength, and the inter-node distance is not known. Hence, information from greater than *(n+1)* nodes is needed for pinpointing a target node in an *n* dimensional plane. The work in [38] presents an approach for minimizing the aggregate error by considering measurements from a redundant number of anchor nodes. A redundant set of equations is linearized and solved to minimize the least square error.

In [39], Savarese *et al.* propose an iterative protocol that diffuses the location information gathered from nearby anchor nodes through the network. Using this technique, Bagchi *et al.* [40] show the relationship between the number of anchor nodes and the errors in the location determination, given a certain error in one-hop neighbor distance estimation.

Angulation is an alternate method to lateration for computing location based on neighbor information, where angles are used in addition to distance. A schematic of the use of angulation is shown in Fig. 3.1(b). Directional antennas are needed for the angle measurements. Previous work has used phased antenna arrays to use the angulation technique [32]. Sukhatme et al. show that using range and approximate sector information can improve localization accuracy at reasonable node densities [52]. Niculescu discusses using the angle of arrival (AOA) of the signal and node orientation adjustment to find node locations [49].

There is a class of location determination techniques that do not rely on any property of the received signal. Instead, they rely on the connectivity measure, i.e., if a node α is able to hear from another node β, then α is connected to β and its location is constrained to be within the transmission range of β ([35],[41],[42]). This class of techniques based on connectivity measure provides location estimates which are quite coarse-grained. The granularity becomes coarser with larger transmission ranges of the reference nodes. An overhead of beacon or hello messages is also incurred and the convergence times of the algorithms are often sensitive to the frequency of these messages [42]. Also, some of the protocols ([41],[42]) require centralized processing which limits their scalability.

Römer proposes a technique geared to dust-sized sensor nodes which only have passive optical communication capability and do not have active RF communication capability. It relies on a powerful base station that sends a photo beam and rotates. Each sensor node has a photo beam detector and a clock and marks how long it sees the beam and the period of rotation and determines its location based on this. The method is only applicable if single hop communication is possible between all nodes and the base station. Also, as has been demonstrated in [38] and appears well accepted, distance measurements over large distances are very inaccurate.

4. FRESHET – ENERGY-EFFICIENT NETWORK REPROGRAMMING

In this chapter we present a protocol called *Freshet*[1], which fits in the genre of network reprogramming protocols introduced in Chapter 2. Each node in Freshet operates in three phases. The first phase is the blitzkrieg phase, which occurs only within the first few moments once new code is injected into the network. The second phase is the code distribution phase where the code segments are pipelined and disseminated through a three way handshake over individual hops. The final phase is the quiescence phase when no new code is being injected into the network. To motivate this work, this protocol makes several important realizations within the sensor network to ensure efficient delivery of code updates throughout the network.

The first realization of this protocol is that a brute force flooding method for network reprogramming is not feasible due to the enormous bandwidth overheads. Also, a node may need to be reintegrated into the network after the code upload process is complete and therefore a pure push based mechanism will not work. It is crucial to suppress meta-data and data wherever possible. The suppression utilizes the shared nature of the wireless medium and the capacity of a node to overhear its neighbors' communication. For example, from the point of view of a node A in the network, if it has version v and a neighbor node B requests for a page of version v' $(< v)$ from a node C, then A can proactively send the more recent code to B. This will cause a suppression of the transmission from C to B if C and A are neighbors. Next, we use pipelining of the different pages in a binary image to speed up the process of code upload. Each interested node may initiate the process of forwarding the code in units of a page as it receives the

[1] OED: *Freshet* – (i) A small stream of fresh water (Obs. exc. poet.); (ii) A stream or rush of fresh water flowing into the sea; (iii) A flood or overflowing of a river caused by heavy rains or melted snow. Used by Bowen in *Virgil* as "A cave … sweet fountain freshets within it."

pages and aggregates them to create its own complete binary image. This is in contrast to the approach in Mote Over the Air Programming (MOAP) [6] where the forwarding happens only when the entire code has been assembled at a node. Since a binary image may consist of many pages and the wireless links are failure prone, the MOAP approach may lead to excessive retransmissions and therefore bandwidth overheads. The segmentation of the image into pages is also useful when multiple sources are present. Freshet uses interleaving of the pages in the image from different sources to speed up the code upload to the interested nodes. The key insight to enable this is to allow nodes to receive pages out of sequence. This leads to somewhat more state maintenance at the node but substantially speeds up the process.

A fundamental insight used in Freshet is that nodes can be put to sleep by making the advertisement-request-data handshake happen only at certain points in time. When new code is introduced into the network, Freshet has an initial phase, the *blitzkrieg phase*, when information about the code propagates through the network rapidly along with some topology information. The topology information is used by each node to estimate when the code will arrive in its vicinity and the three way handshake will be initiated – *the distribution* phase. Each node can go to sleep in between the blitzkrieg phase and the distribution phase thereby saving energy. The potential for energy savings grows with the size of the network. Freshet also optimizes the energy consumption by exponentially reducing the meta-data rate during conditions of stability in the network when no new code is being introduced, called the *quiescent phase*. The possibility of a node missing the code advertisement is made vanishingly small by redundant transmissions of the advertisement during the blitzkrieg and the distribution phases.

In order to demonstrate the behavior of Freshet, we build simulation models in TOSSIM, which is a discrete event network simulator that compiles directly from unmodified TinyOS application code. TOSSIM captures the behavior of the entire TinyOS network stack in a detailed manner and is used to get around the problem of scaling of our actual sensor network test bed.

It must be noted that in some of the high level goals and design approach, Freshet has similarities with Deluge [1] and MNP [7]. However, there are substantial differences in the protocol design which lead Freshet to make the following novel contributions.

1. Freshet shows that combining local and network topology information provides energy benefits while preserving scalability, the advantage of using local information.
2. Freshet addresses the problem of code upload from multiple original sources. It shows the benefit of using interleaved transmission of pages to speed up the code upload process in the multiple source situation.
3. Freshet shows a method for energy optimization in the quiescent phase while preserving the reliability guarantee of other protocols.

Freshet optimizes the energy consumption more aggressively through turning off the nodes between the blitzkrieg phase and the distribution phase using limited topology information. It also trades off the responsiveness of the protocol to newly joining nodes for saving further energy during the steady state. It also uses out of order paging to speed up the code update with multiple sources of the code.

4.1 Design of Freshet

4.1.1 Blitzkrieg phase

In the blitzkrieg phase, Freshet propagates information about the nature of the new code to all nodes in the network. This is accomplished through a fourth type of message, a *warning message*, different from the advertisement, request, and broadcast data messages. This message contains information about the new code in the form of the version number, the number of pages, and how far the sending node is from the data source through a hop count metric. The blitzkrieg phase enables energy optimization by each node that can use the hop count information to determine when it will enter the distribution phase.

The hop count is incremented by each intermediate node routing the warning message. Every time a node hears a unique warning message with code information more recent than its own, it starts a short, randomized timer. Once this timer fires, and the node has not heard more than *w* warning messages with the same code version as its own, then the node sends out the warning message. The node sends the exact same message as the one it first received, except that it increments the hop count from the original message. This information therefore gives the receiver an estimate of how many nodes have seen and propagated the warning message. Based on empirical results of time to propagate code over one hop, Freshet estimates when the hop count is sufficiently large that energy savings are possible by stopping advertising. The node then starts a timer for how long to cease advertisements. Given that the sleeping will happen for source to node distance beyond h hops, a node h_a hops away sleeps for time $t_{off}*(h_a-(h-1))$, where $h_a > h$. This choice is dictated by the result from Deluge that the time to propagate a page is linear in the number of hops for a fixed object size. Empirically a threshold hop count of 4 is found to be reasonable. However, if further accurate information about the topology were available, it may be possible for each node to estimate the timeout more accurately. But we feel this would violate the design paradigm of using local information that has proved so valuable in sensor network design.

The warning messages are two bytes smaller than the original advertisement messages. They use essentially the same information as the advertisement messages except that they do not need to know how many pages are complete, since it is assumed that the new code is sent from a source with the entire image. The blitzkrieg phase adds a fixed number of warning messages from each node when it starts receiving the first page of a code image and is transparent to the normal operation of the original Deluge. A redundant number of warning messages is used to guard against losses. The distribution phase of Deluge achieves efficient and robust dissemination of code pages. Thus, Freshet leaves this phase unchanged and chooses to optimize aspects of Deluge not associated with the active distribution of code, while still maintaining the same performance.

4.1.2 Distribution phase

The distribution phase of Freshet is identical to that of Deluge and is mentioned here for completeness. It functions through a three-way handshake protocol of advertisement, request, and broadcast code. Each node keeps a time window in which it listens for advertisements. Within this window it randomly selects a time at which to send an advertisement with meta-data containing the number of complete pages in its code image and the total number of pages in the image. When the time to transmit the advertisement comes, the node sees whether it has heard s_a advertisements with identical meta-data, and if so, it suppresses the advertisement. When a node hears code that is newer than its own, it sends a request for that code and the lowest number page it needs, to the node that advertised the new code. The node on hearing this sends the appropriate page to the requesting node. This process continues until the requesting node has updated its code. A node only fills its pages in monotonically increasing order thereby eliminating the need for maintaining large state for missing holes in the code.

In order to ensure fast code dissemination, Deluge uses several mechanisms for message suppression. The first is sender selection. When a node needs new code, it designates which node it wants to have as a sender for the new code. This sender is selected by the most recently heard advertisement. The second mechanism is through request messages. When a node overhears a request for the same code it needs, then it does not send its request out, unless it does not receive the new code within some time interval. The third mechanism is advertisement suppression as described earlier in this section.

4.1.3 Quiescence phase

The next phase of Freshet is the quiescence phase. This phase occurs once code has been disseminated completely within the transmission range of the node. Thus, a node no longer hears requests from any node needing code and the node itself has the complete image. Since there will be no further code transfers for the immediate future, the node does not need to advertise at all. In Trickle, a scheme is proposed for sending an advertisement every so often to ensure that if new code is added to the network the nodes are aware of the update, but at the same time limiting energy use. However, since the quiescent phase is typically the most long-lasting phase, Freshet optimizes the energy consumption further by switching to a complete pull-based mechanism to service new nodes. If any new node enters the network, it will advertise its old data and thus will alert the already present nodes that they need to start transmitting again.

Once a node has received all of its code, it sends out *r* advertisements to ensure (with high probability) all interested nodes have heard it and then it stops advertising altogether. If it hears any new code or advertisements with old code, then it will immediately start advertising again and either obtain the new code and then transmit it or transmit its current code. Typically, Deluge is a hybrid of push and pull, but in this scenario Freshet picks one or the other depending on whether there is new code (push) or code is requested (pull).

Freshet can function in either a dynamic or a static network. The dynamic nature may be a result of failures, which will cause new routes to be discovered that Freshet will use in the propagation of code. In the case of a mobile network, Freshet needs extra control messages – a node needs to notify its old neighbors before moving and the new neighbors after moving, using the warning message in order to update the topology information.

4.1.4 Turning the radio off

As [22] shows, the major energy expenditure for the radio is the idle receive time and not the transmission energy level or number of messages sent. Therefore, while sending fewer messages saves some energy it is more valuable to turn the radio off whenever possible. Freshet seeks to turn off the radio between the blitzkrieg and the distribution phases and in opportune moments of the quiescence phase. MNP in [7] turns off the radio of nodes which are not selected as senders of code, but does not address radio usage in the long time periods before and after code updates. After the blitzkrieg phase, each node estimates the number of hops distance from the source of data and turns off the radio while waiting for the data to arrive in its vicinity. In this way, a large network that needs to disseminate a large data object can save substantial amounts of energy by turning off the radios for nodes far from the originator of the code image.

In the quiescent phase, it is more difficult to decide deterministically when a node may safely shut off its radio. Since new nodes may enter the network at any location and new code may be injected at any time, only a portion of the network can sleep and the nodes that sleep must probabilistically ensure that the network will still respond to any new events. The means of accomplishing this task is through recording how many neighbors, b_n, are within each node's vicinity. Consider a time slot of length τ. Each node listens for a period $\tau/2$ and then decides with probability $1-1/b_n$ that it should sleep for the next $\tau/2$ period. This design is a tradeoff between energy saving and responsiveness of the network to new code or new nodes.

4.1.5 Interleaved pages

A significant component of the design of Freshet deals with situations where a network may have multiple identical code sources in different locations. For example, each of the multiple data sinks may act as a code source. In many cases with a deployed sensor network it is hard to access nodes inside the mesh of the network, but easy to access the outside edges of the network. A user may deploy additional sources with the goal of reducing the time to propagate code through the network. Since internal nodes also become sources of code through dissemination, we use the term *code originator* (or *originator* for short) to indicate the original sources that initiated the code propagation.

The use of multiple data originators would be in partitioning the network into smaller portions. Two data originators at opposite ends of a network will effectively halve the size of the network. We propose a scheme to distribute pages out of order to improve dissemination in the network as a whole. Through out of order dissemination of pages it is possible that when pages distributed from different originators meet, they may fill in the "gaps" in each node's code image. This allows us to create fresh originators from which code can be disseminated. With an appropriate negotiation scheme, nodes with different pages can help each other complete their code images well before the remaining pages would reach them from the original sender.

Thus, we propose the concept of node *parity*, where the parity of a node is determined by which set of pages it chooses to disseminate first when it already knows that there are other originators in the network sending pages with different parity. In particular, Freshet has *numSrc* originators sending code of size *p* pages into the network. For a given originator s_j, it will first send out pages numbered *i* such that *i* mod *numSrc* = *j*. Originator s_j will be said to have parity *j*. After distributing these *p/numSrc* pages, it will then distribute pages numbered *i* such that *i* mod *numSrc =j-1*, *j-2*, and so on until all pages have been disseminated, cycling through all *j* such that $0 \leq j \leq numsrc$. It is assumed that the deployment of the originators is done with some thought – they are relatively evenly spread and are assigned non overlapping parities.

The next problem is how to resolve conflicts between nodes with pages of different parity. For a node that has the complete image, its sending is dictated by parity as described above. The next rule governs a node with an incomplete image. For such a node there is the concept of *cycles*, one for each parity in the network, with the node likely switching through the different cycles. Consider Fig. 4.1 which depicts the behavior of a node in a network with two parities. It goes through an even cycle and an odd cycle. Each cycle has one *slot* for listening and one for advertising and requesting. The cycle is dedicated to the particular parity when activity pertaining to both parities is happening around the node. However, if the node hears *a* consecutive advertisements of one parity, where *a* is a user-defined parameter, then it will use all available cycles for that parity. This is to ensure that cycles are not idled for pages of a given parity that are still far off from a node.

Listen for even advertisement	**Advertise/Req Even Pages**	**Listen for odd advertisement**	**Advertise/Req Odd Pages**

Fig. 4.1 Advertisement scheme for parity *j*=2

As in Deluge, pages may only be downloaded sequentially within that parity. For example, with two parities, the motes must download page 5 before page 7.

An optimization in Freshet for interleaved pages is that if a node's radio is idle in a given cycle and data is available, the node will utilize the idle period to get the data. What *is* sacrosanct is that a node does not advertise or request for data outside the turn. This is important to prevent the protocol from thrashing in which only meta-data exchanges happen and the network's throughput tends to zero.

4.2 Analysis of Freshet

4.2.1 Analysis 1: number of redundant advertisements.

First we analyze the number of redundant advertisements that are needed to achieve a given reliability of reaching a node in the network which is *relatively* isolated. This is defined as the reliability of the code update protocol. Let the number of nodes in the network be n, the size of the sensor field be A, and the radius of transmission be r_0. We assume for the analysis that the nodes are uniformly distributed in the sensor field. The density of the sensor field is $\rho = \frac{n}{A}$ and the average number of nodes in the transmission range of a given node is $\lambda = \pi r^2 \rho$. The probability that a node has n_0 neighbors is given by a Poisson distribution. The approximation used is $n >> n_0$ and can be approximated by ∞. $P(b = n_0) = \frac{\lambda^{n_0}}{n_0!} e^{-\lambda}$, $n_0 = 1, \ldots, n$, where d is the random variable representing the number of neighbors that a node has. The mean and the variance of the random variable follow from the property of the Poisson distribution. Expected value $E(d) = \lambda$ and Standard deviation $S(d) = \sqrt{\lambda}$. Let us consider an arbitrarily isolated node, say α, which is a fraction τ of the SD away from the mean. Thus, the number of neighbors of the isolated node is $b_\alpha = E(d) - \tau S(d) = \sqrt{\lambda}(\sqrt{\lambda} - \tau)$, $\tau < 1$. If $\tau = 1$, the node is disconnected from the network and can never get the code update.

Now, consider the probability of successful transmission of an advertisement from one of the neighbors of α to node α. Note that we only need to consider a successful transmission of the advertisement and not the subsequent request and code packets since if the node α is made aware of the presence of a new code, it will continue to request arbitrarily long till successful transmission of the code is achieved. Of course, realistically collisions will cease on the channel to node α and the transmission will be successful within a few attempts. In order to estimate the probability of successful transmission of the advertisement, we use the analysis of the 802.11 CSMA/CA protocol given in [20]. For the protocol, binary exponential backoff is being used with minimum size of the contention window $CW_{min} = 2^{m}W$ and the maximum size $CW_{max} = 2^{m'}W$. We assume that any contention for the wireless channel comes from the neighbors of node α. The number of retries by a given node for transmitting the advertisement is then $M = m'-m+1$. The probability of successful transmission in one time slot is $P_s = P_{tr}P_{s|1}$, where P_{tr} is the probability that there is transmission and $P_{s|1}$ is the probability of successful transmission in a slot, given there is a transmission. We obtain using equations (10) and (11) in [20], $P_{tr} = 1-(1-P_t)^{b_\alpha}$ and $P_{s|1} = \frac{b_\alpha \mathrm{P_t}(1-\mathrm{P_t})^{b_\alpha -1}}{1-(1-\mathrm{P_t})^{b_\alpha -1}}$, where P_t is the probability that a station chooses to transmit at a randomly chosen slot time and is given by equation (7).

Therefore, the probability of successful transmission $P_S = 1-(1-P_s)^M$, assuming that the probability in each time slot is i.i.d. Therefore the probability of success of at least one advertisement from among the r sent by a node i which is a neighbor of node α is $P_{S,i} = 1-(1-P_S)^r$. Therefore the probability of success of at least one advertisement reaching the node α, i.e., by definition the reliability of the protocol, is $R = 1-(1-P_{S,i})^{b_\alpha}$. This can be made arbitrarily close to 1 by increasing the value of r and asymptotically goes to 1 as $r \rightarrow \infty$.

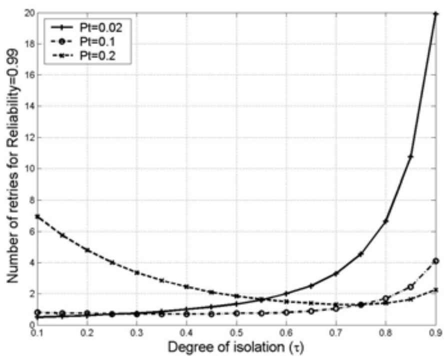


Fig. 4.2 Variation of number of retries to reach 99% reliability for an isolated node

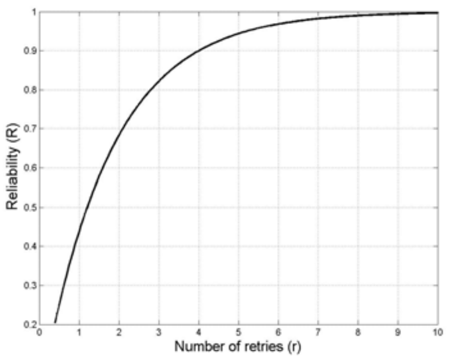


Fig. 4.3 Variation of reliability with number of retries for an isolated node (τ=0.9)

The analytical results are plotted in Fig. 4.2 and Fig. 4.3 for n = 15×15, A = 200×200, CW_{min} = 16, CW_{max} = 1024 from the 802.11 standard for FHSS Physical layer [20], Tx power = -20dBm, and minimum Rx power = -85dBm giving r_0= 39.0937 m (for the Mica motes). Fig. 4.2 shows the non intuitive result that the number of retries is not monotonically increasing with increasing τ. For higher values of P_t, the increased contention due to the number of neighbors of the isolated node causes the number of retries to decrease with τ to a minimum before increasing. Fig. 4.3 shows that as expected the reliability asymptotically approaches 1 which puts the reliability claim of Freshet on the same ground as that of other epidemic based protocols.

4.2.2 Analysis 2: time between blitzkrieg and distribution phases.

Next, we analyze the separation in time between the blitzkrieg and the distribution phases and show how this depends on the density of the network. Consider that the code spreads as a wave from the source with an illustration in Fig. 4.4 with the source at the top left of the field. A line connecting a set of nodes implies that a page reaches all the nodes the set in the same round of the three way handshake. For a given node *i*, this is called the **W**ave **C**ompanion **S**et (WCS_i).

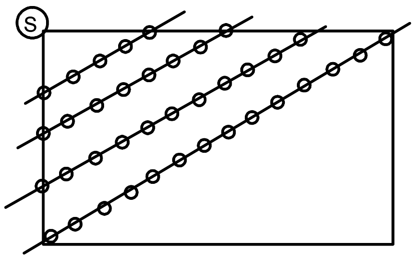


Fig. 4.4 Pattern for propagation of code

First, let us analyze the time for a single round of a three way handshake. The time has three components – the delay due to the CSMA/CA contention, the transmission time, and the processing time at the node. The MAC delay is difficult to compute analytically for 802.11 and no closed form solutions exist.

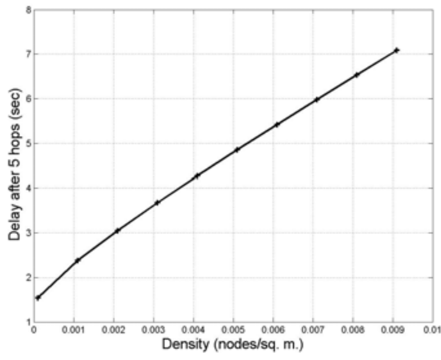


Fig. 4.5 Variation of delay of code dissemination with network density

The curve shown in [21] indicates that for the region of interest (low contention) the delay is approximately proportional to n^2, where n is the number of contending nodes. Let the nodes be placed on a square grid of area A and grid separation δ. The separation from a diagonal node is $\delta' = \sqrt{2}\,\delta$. The density of the network is $\frac{1}{\delta^2}$. Let the radius of transmission $r_0 = M\delta'$. Therefore, $M = \frac{r_0}{\delta'} = \frac{1}{\sqrt{2}} r_0 \sqrt{\rho}$. Observe that the contention for each phase of the handshake is caused by the members of the WCS which are within transmission distance away, which are $2M+1$ in number. Let the sizes of the advertisement, request, and code page be A, R, and C, respectively, the time to transmit one bit (the bandwidth) be T_{tx} and the processing time be T_{proc}. Therefore, the total delay introduced by a single round of the handshake is

$$T_{round} = T_{Adv}+T_{Req}+T_{Code} = (G.(2M+1)^2 + A.T_x + T_{proc})+(G.(2M+1)^2 + R.T_x + T_{proc})+(G.(2M+1)^2 + C.T_x + T_{proc})$$

$$= 3G.(2M+1)^2 + (A+R+C)T_{tx} + 3T_{proc}$$

Hence, assuming perfect pipelining of the single page of the code, the time to go through h hops is $T_{delay,h} = h.T_{round}$. The relation of this with the density of the network (replacing M by its expression containing ρ) we get the plot shown in Fig. 4.5.

4.2.3 Analysis 3: time for dissemination with multiple originators.

The third analysis is for striping across one or two code originators. Let us assume a rectangular sensor field as shown in Fig. 4.4, with the number of hops across the diagonal being D_0. The number of pages in the code image is P_0. For a given originator, the maximum number of hops along the diagonal it has to transfer code to is D and the number of pages it has to transfer is P (since the behavior is identical for all originators, we drop the subscript i.) The transfer along the diagonal is of interest to us even though the transger may happen over larger number of hops along a side of the field due to the results reported by Deluge [1]. Let us call the set of nodes to the left-upper half of the field as LU and the set to the right-lower half as RL. The nodes in LU (RL) which are closest to the diagonal (border nodes) are called BN_{LU} and BN_{RL}. We will consider four cases – case 1 has a single originator with $D=D_0$, $P=P_0$ called Single Originator (*SO*), case 2 has two originators with $D=D_0/2$, $P=P_0$ called **D**ual **O**riginator with **N**on-Interleaved Pages (DON), case 3 has two originators with $D=D_0$, $P=P_0/2$ and no handshake happens when the code waves from the two originators meet, called **D**ual **O**riginator with **I**nterleaved Pages and **U**nregulated **C**ollision (DOI-UC), and case 4 also has two originators with $D=D_0$, $P=P_0/2$ but handshake happens as described in Section

4.1.5, called **D**ual **O**riginator with **I**nterleaved Pages and **R**egulated **C**ollision (DOI-RC). The metric for comparing the different schemes is the time to disseminate the code to all the nodes in the field, T_c. As given in analysis 2, the time for the three-way handshake over one hop is T_{round}, shortened here as T_R, which is a function of the number of nodes contending for the channel due to the MAC layer delay component. This number is going to vary for different cases and also different time points within each case. Let us simplify that each node's transmission radius is such that it can interfere with all its one hop neighbors on the grid, i.e., M in the second analysis is 1.

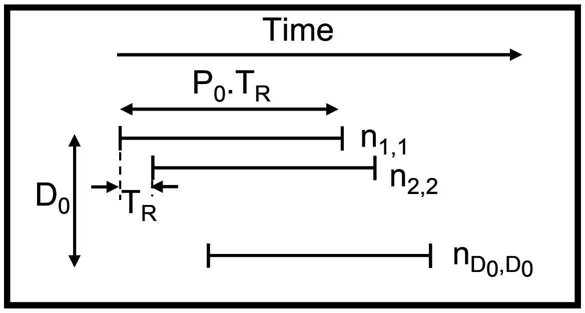


Fig. 4.6 Timeline for pipelined code dissemination

Case 1 (SO): The pipeline in Fig. 4.6 is drawn with each horizontal line showing the reception of the different pages by a given node with time. Here a node $n_{i,i}$ contends with nodes $n_{i+1,i+1}$ and $n_{i-1,i-1}$. Other nodes do not contend due to the suppression mechanisms in the protocol. Therefore the number of contending nodes $n = 3$.

$$T_c = (D_0-1)T_R+P_0T_R = T_R(D_0-1+P_0)$$

Case 2 (DON): $D=D_0/2, P=P_0$. All nodes in LU will get all the code pages from originator S_1 and all the nodes in RL from S_2. The maximum time is to reach the border nodes. Time to reach BN_{LU} = time to reach $BN_{RL} = T_R(D_0/2-1+P_0)$. Here also $n = 3$.

$$T_c = T_R(D_0/2-1+P_0)$$

Case 3 (DOI-UC): $D=D_0$, $P=P_0/2$. The nodes in BN_{LU} get all the odd numbered pages after time $T_1 = T_R(D_0/2-1+P_0/2)$, with $n = 3$. In this time, the nodes in BN_{RL} get all the even numbered pages. Now the contention increases as the nodes in BN_{LU} try to disseminate the odd numbered pages in RL and vice-versa. Notice now that for each round the number of contending nodes is 6 (the node itself along with its two diagonal neighbors in the same set LU or RL plus the three nodes "facing it" in the other set). The time to get P pages across the LU-RL boundary is $T_2 = T_{R'}P_0$, with $n = 6$. Next, the even numbered pages are disseminated through LU and vice-versa, as before with $n = 3$. Thus, $T_3 = T_1$.

$$T_c = T_1 + T_2 + T_3 = T_R(D_0/2\text{-}1+P_0/2) + P_0T_{R'} + T_R(D_0/2\text{-}1+P_0/2)$$

Case 4 (DOI-RC): $D=D_0$, $P=P_0/2$. T_1 and T_3 remain the same as in case 3. The contention is handled through the handshake mechanism and therefore $T_2 = T_RP_0/2 + T_RP_0/2 = T_RP_0 < T_{R'}P_0$.

Plotting T_c for the various cases, we see that dual originators give a clear advantage over single originator; DOI-RC is favored over the others when the number of pages in the image is large, while the three dual originator cases are comparable for large sized networks with small code images to disseminate.

4.3 Freshet Experiments and Results

We build a simulation model for Freshet using the three way handshake mechanism of Deluge in TOSSIM, the simulator for TinyOS. We also simulate Deluge from TinyOS release 1.1.11. While TOSSIM does not imitate hardware precisely, it is a bit level simulator and therefore provides accurate modeling of the physical layer characteristics not seen as accurately in other simulators, such as ns-2. The TOSSIM code runs directly on hardware and closely mimics the trend in the behavior though the measures may have to be scaled to give accurate absolute numbers for the Mica-2 hardware. It is important to stress that the code running on TOSSIM was downloaded to the actual motes and executed there. However, the gains of Freshet are evident for network sizes of the order of tens to hundreds of nodes and therefore TOSSIM rather than the actual motes were used for the results showing the comparative gains of Freshet. This approach is valid because of the accuracy of the simulation infrastructure and has been used by other researchers [1],[7]. We use the notion of code image being fragmented into pages and each page consisting of multiple packets. The default page consisting of 48 packets of 36 bytes each is used. The nodes are arranged in a rectangular grid with constant 15’ spacing between adjacent grid points. A square placement of nodes on the grid is used, giving rise to N×N nodes, where N is varied for the experiments. Henceforth, the term "N nodes square" will imply a total of N^2 nodes in the network. The amount of sleep time for a node h hops away from the warning message is $8(h-1)$ for $h \geq 4$. This equation was found empirically and generally yielded adequate responsiveness in the network while still guaranteeing some period of sleeping for nodes far from the source of the code.

TOSSIM does not have built in simulation for energy computation, nor does it have a radio model with power management features. To work around this problem, we used PowerTOSSIM [22] to track energy usage, particularly in the radio, and disabled motes' radios when they were to be put in the sleep mode according to the protocol. For energy consumption we used the Mica-2 hardware model with the parameters as in Table 4.1. As shown in [1], the completion time in Deluge scales linearly with object size. Through our experiments with Freshet we discovered that energy use followed a linear increase with object size as well, and hence we do not discuss the problem further.

Table 4.1
Energy model used for results

Radio idle or receive	7.03 mA	EEPROM Write current	18.4 mA
Radio transmission (max transmit only)	21.5 mA	EEPROM Write time	12.9 ms
CPU Active, Idle	8.0 mA, 3.2 mA	EEPROM Read current	6.2 mA
Radio sleep	1 μA	EEPROM Read time	565 µs

4.3.1 Single originator results

We run our first set of experiments with code image consisting of 5 pages in networks of sizes of 6, 8, 10, 12, 14, 16, and 20 motes square. For the purpose of further energy evaluation, we also analyzed networks ranging from 8 to 20 motes square and networks with dimensions of Y by (Y+1) with Y ranging from 8 to 19. The simulations are started with all the nodes being active and 10 s into the simulation, the originator starts transmitting the code pages. The simulations are run until all the nodes receive all the pages. We then analyze the time to finish receiving all data and recognize that this time can be highly variable from one run to another, though preserving the relative order of performance of Freshet and Deluge. This occurs because in both Freshet and Deluge measures are taken to decrease advertisement frequency, which makes it possible for a node with few pages downloaded to "disappear" if its link quality is poor or inconsistent. Essentially, some nodes may not be able to request the new code they need because of network contention and their packets may be dropped. In some cases a handful of nodes may take much longer periods of time to complete code download than others due to this phenomenon. Therefore, we evaluate the performance of Deluge and Freshet till the point of acquiring 92% of all pages needed in the network, i.e., 92% of the number of nodes times the number of pages have been downloaded in total. We chose 92% because it was the highest network completion percentage that showed very consistent code distribution times.

In all cases we are evaluating the radio energy usage of Deluge and Freshet. We also track the CPU energy usage and energy from EEPROM writes and reads, but we found that the differences in this energy use due to these heads between Deluge and Freshet were negligible.

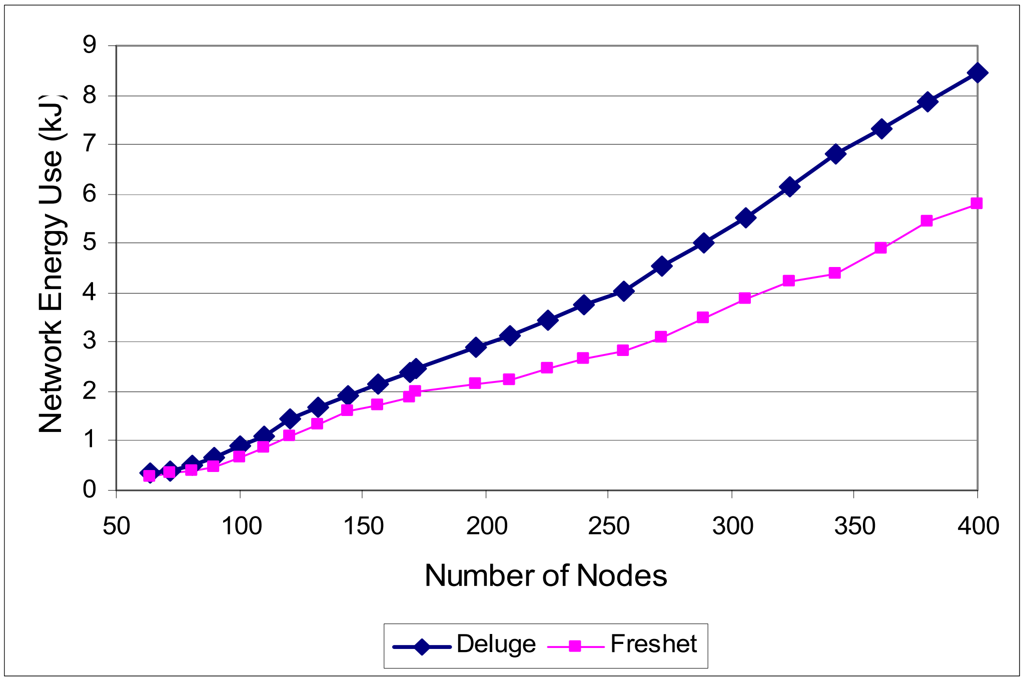


Fig. 4.7 Radio energy usage of the entire network for a given number of nodes

Fig. 4.7 shows that as the number of nodes increases in the network, Freshet saves more energy compared to Deluge. The energy gains of Freshet over Deluge increase with network size since the energy spent per node is lower in Freshet. These plots scale based on the energy used per node. Clearly, a larger network uses more energy due to more nodes, but since there is also more time for code to propagate, each node will need to spend more time waiting for code. Fig. 4.10 shows the average energy use per node based on network size. This figure shows two main characteristics. First, the smaller networks use much less energy than the middle-sized networks. This is primarily due to the increase in the average hop distance between the originator and the nodes; in the 8x8 network the diameter of the network is 2-3 hops. In the 11x11 network the diameter is 4-5 hops. Each hop increases download time and therefore increases energy use. However, as the network size continues to increase, the energy use begins to level off. This effect is due to Freshet's and Deluge's complexities in transferring code. [1] found that for networks with diameters of less than 8 nodes code transfers are proportional to the product of the code size and the network diameter. Our simulations tended to see this behavior present in networks with diameters up to 10 nodes, as shown in Fig. 4.10. For larger networks, this trend is linear in the size of the code and the network diameter, as discussed in [1]. The combination of these characteristics causes the plot to be linear as the network size increases. For the purpose of prediction, we made a 2nd order regression line for both Deluge and Freshet based on Fig. 4.7, and show those results in Fig. 4.8. These lines have R^2 values of 0.9926 and 0.9978 for Freshet and Deluge, respectively. What we see quite clearly from this figure is that while both Deluge and Freshet proceed with nearly linear increases in energy savings, Deluge's energy use increases faster than that of Freshet.

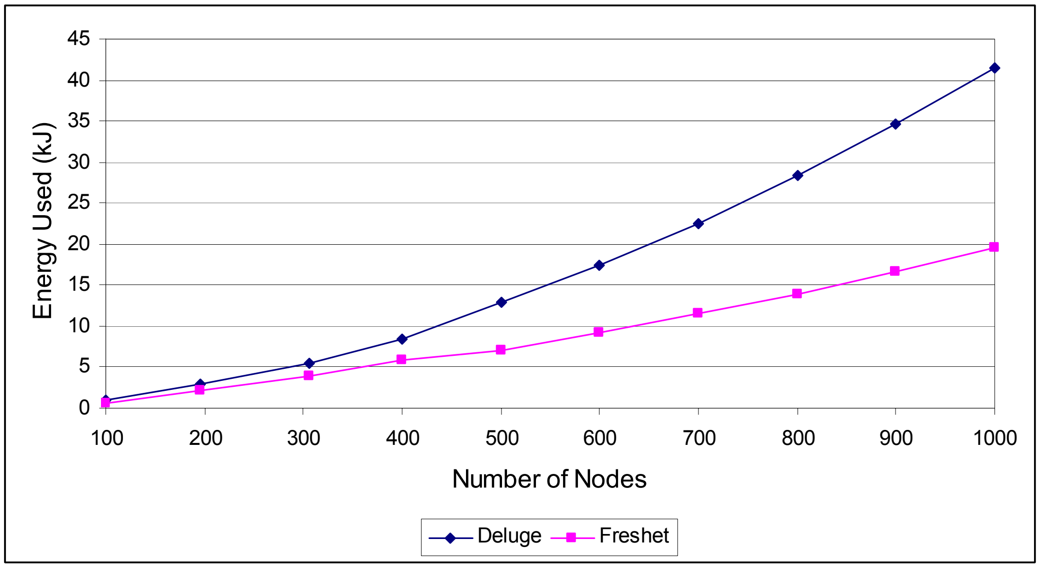


Fig. 4.8 Trend line for larger networks

As would be indicated by the design, the energy savings happen for two reasons. The nodes far from the originator node use the blitzkrieg phase to turn off their radios for the appropriate period of time before they must start transferring pages. The second reason is that nodes near the source that complete their code transfers first will have lower duty cycles for their radios as they enter the quiescent phase.

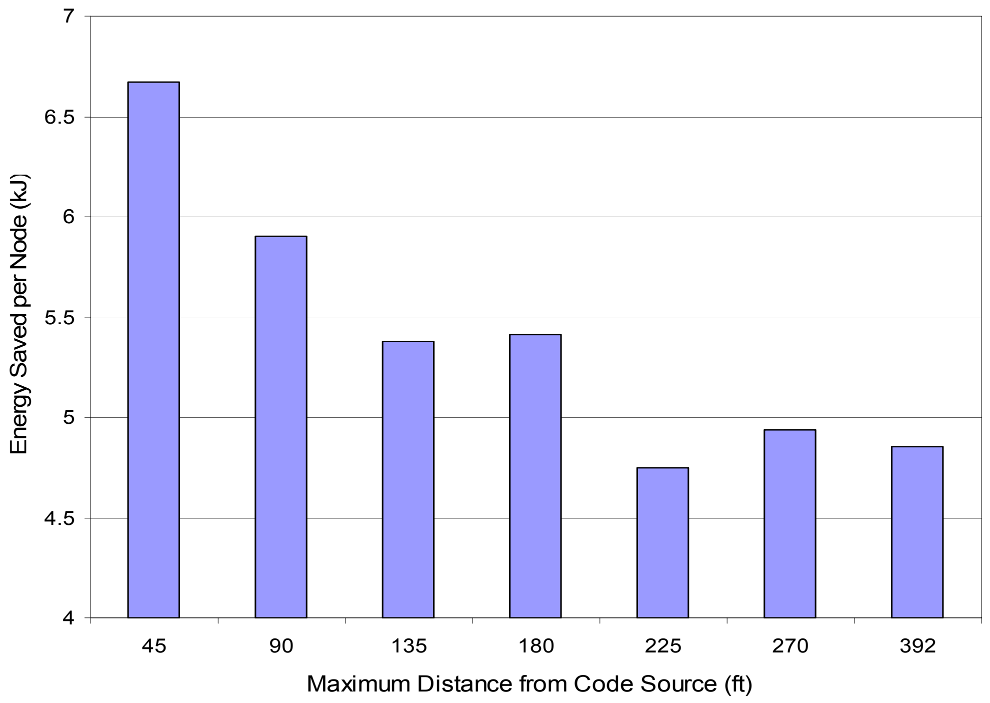


Fig. 4.9 Average energy saved per node grouped by distance from code source

Fig. 4.9 shows the average energy saved per node grouped by distance from the code source. This simulation is for a 20x20 network. The maximum distance of any node from the code source is 392 feet, the minimum is 0 feet, the code source itself. What this figure demonstrates is that nodes closer to the code source are able to save energy through the quiescent phase by turning off their radios once they have acquired all of the code. Similarly, nodes far from the code source can save energy through the blitzkrieg phase but must still spend more time with their radios on to acquire the code updates. This energy saving calculation is made based how long the network takes to download code. We know that the idle radio draws 7.03 mA, and therefore can calculate how much energy Deluge would normally use through its radio by multiplying the time to download by the idle radio current by the voltage of the motes. We find the energy saved by then subtracting the radio energy used in the Freshet simulation from the calculated energy for a Deluge simulation of the same duration.

In Fig. 4.11 we show relative completion times of Deluge and Freshet. In all cases Deluge finishes transferring 92% of its pages before Freshet. So while we found that Freshet in the single source case is typically 10-15% slower than Deluge, it also uses much less energy. The increase in time for dissemination occurs mostly because Freshet loses some coverage in the network by turning off the radios during the quiescent phase. Fig. 4.12 shows that Deluge still outperforms Freshet after only 50% of the pages are downloaded, but by less than 5% on an average. This indicates a tradeoff – if marginal loss in time for dissemination can be tolerated in order to save energy, the design point would favor Freshet.

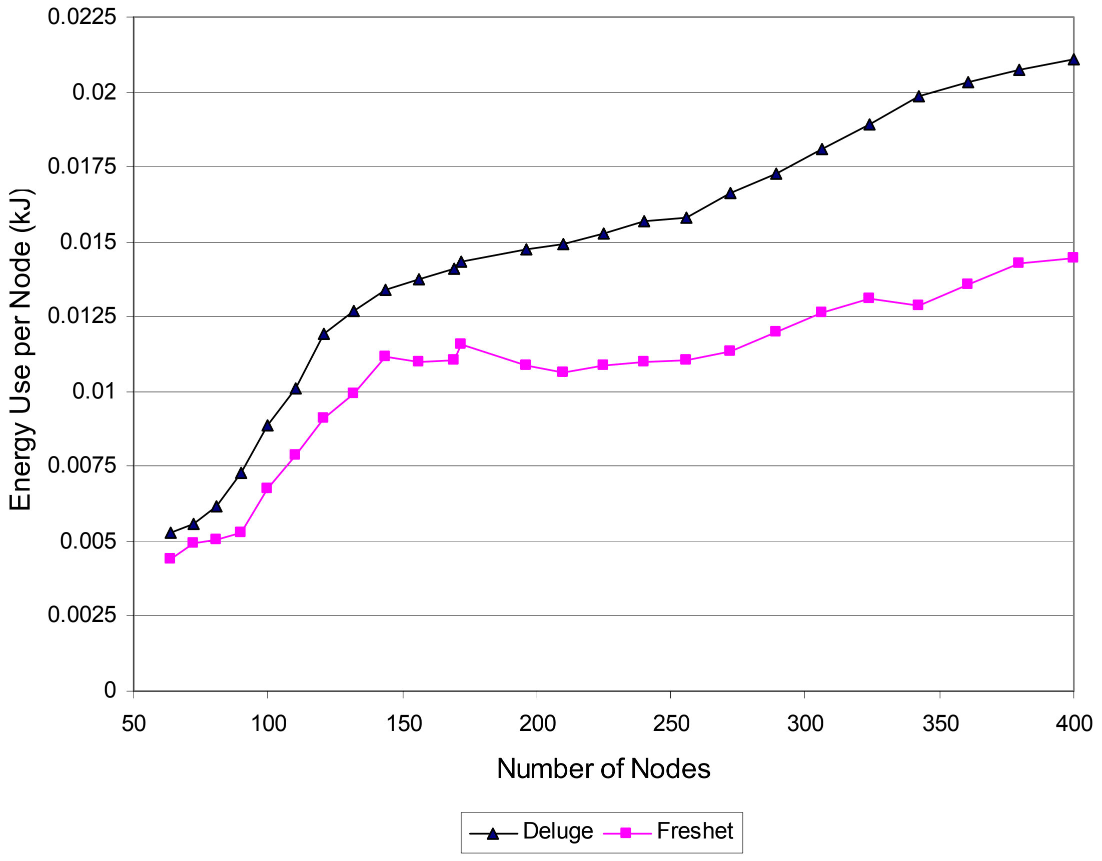


Fig. 4.10 Energy usage per node of Deluge and Freshet

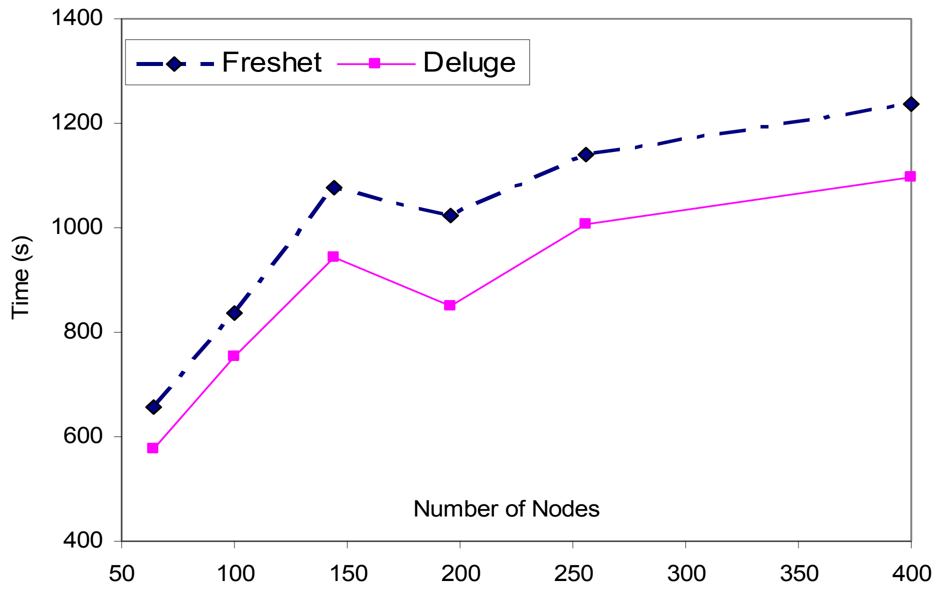


Fig. 4.11 Time to complete 92% of pages

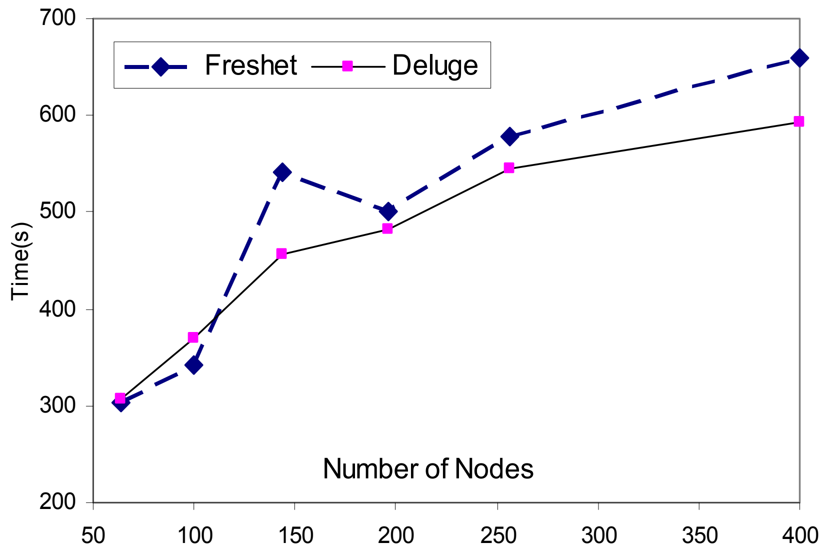
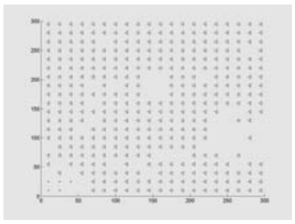
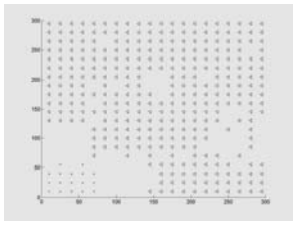
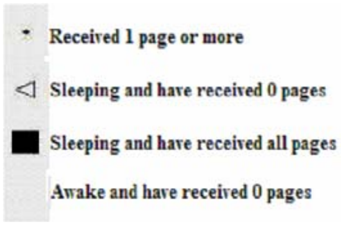


Fig. 4.12 Time to complete 50% of pages

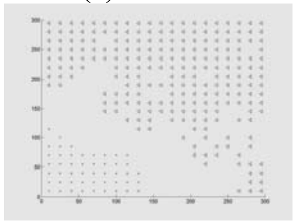
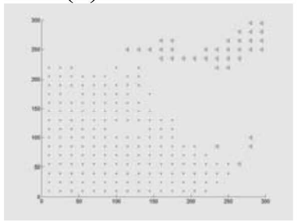
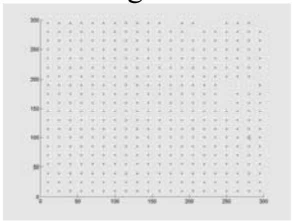
The next part of our analysis centers on the network's behavior over time. Fig. 4.13 shows the positions of sleeping nodes in the network as time progresses. The originator node is in the bottom left corner of the area. The small dots represent the nodes that have at least one page, the bigger dots (small solid triangles) represent nodes that are asleep, and the lack of any dot at a grid point represents a node that is awake but does not have a page yet.

Fig. 4.13(a), (b), and (c) show that initially most of the network is asleep. In (d) most of the nodes have now turned their radios back on, and by (e) nearly all nodes in the network have at least one page. (f) shows the transfer of the code image to be complete, and in (g) we find that the nodes near the originator have now begun to sleep in the quiescent phase. By (h) a larger fraction of the network is sleeping in its quiescent phase.

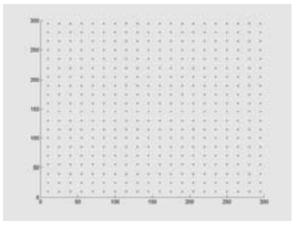
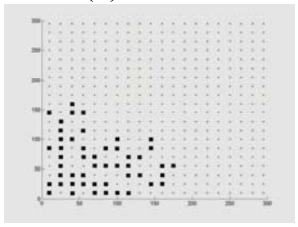
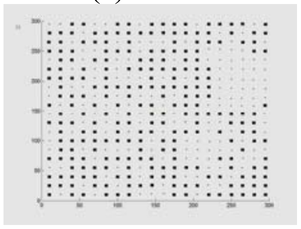
These figures show that Freshet can reliably predict when to turn its motes' radios on and off, thereby saving substantial amounts of energy. While in some cases we see that motes that are near those that have already obtained a complete page and should be ready for beginning the distribution phase, are actually asleep (some nodes to the right in (d)). However, this is the exception rather than the norm, implying that network coverage is generally unaffected.




• Received 1 page or more
◁ Sleeping and have received 0 pages
■ Sleeping and have received all pages
Awake and have received 0 pages

(a) *t*=15 (b) *t*=30 Legend

(c) *t*=75 (d) *t*=150 (e) *t*=300

(f) *t*=450 (g) *t*=650 (h) *t*=900

Fig. 4.13 Nodes sleeping in the network over time. Triangles are sleeping nodes, dots have at least 1 page

Fig. 4.14 and Fig. 4.15 demonstrate the substantial energy savings that can be obtained through the short term use of the quiescent phase (in this experiment the completion time is 1500 s, so 150 s for the quiescent phase is approximately 10% of the total completion time). Fig. 4.14 shows the distribution of node energy savings when 75% of the network has got the complete code. The energy savings at this point are due to the estimate of the time between the blitzkrieg and the diffusion phases and sleeping for part of it. Fig. 4.15 shows the same network 150 s after 92% of the network is completed. It is clear that a much larger percentage of the network has increased its energy savings in this time since the quiescent phase has set in.

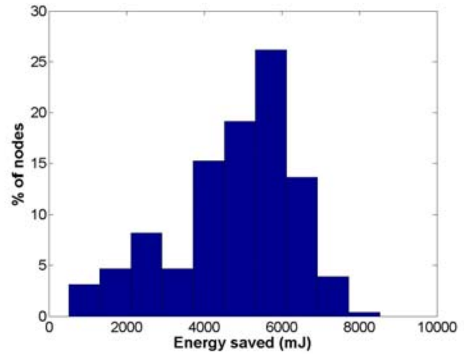


Fig. 4.14 Shows the energy saved at 75% network completion (mJ)

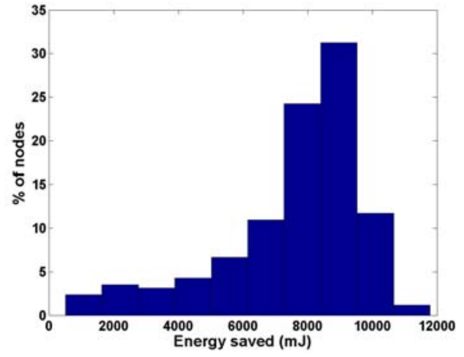


Fig. 4.15 Shows the energy saved 150s after 92% of the pages were completed(mJ)

4.3.2 Multiple originator results

Our second set of experiments was run with two originators at the top left and bottom right corners and code size of 4 pages in networks consisting of 8 through 12 nodes square. We compare the performance of Deluge, with one and two originators and Freshet, also with one and two originators. The two originators used in the native Deluge implementation are identical in all aspects except location; in the case of Freshet, one originator is set to prioritize distribution of even numbered pages and the other odd numbered pages.

Fig. 4.16 summarizes our results with the two Freshet bars to the left of the two Deluge bars. Our results show that multiple originators always improve performance in networks with 100 nodes or more. Specifically, when the originators are farther apart due to the larger network, the interleaving of pages outperforms both the native Deluge implementation and Deluge using two sources. This result occurs because the hidden terminal problem limits the functionality of Freshet in networks with less than 100 nodes; this problem's effects are lesser in networks with 100 nodes or more.

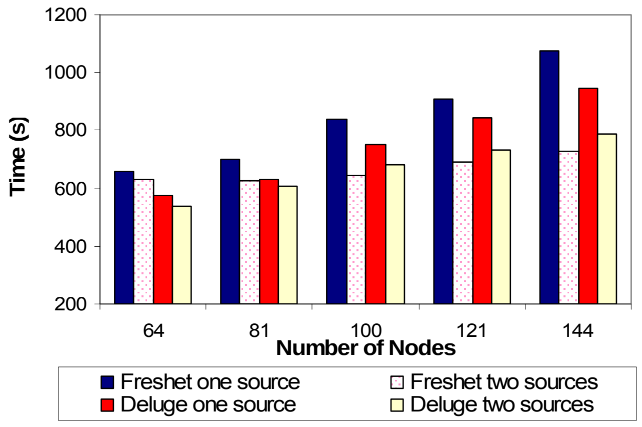


Fig. 4.16 Time to completion of various distribution techniques

Generally, interleaving looks to distribute two different pieces of code (odd and even pages) on opposite ends of the network. In a small network these messages will tend to collide, and therefore increase contention and result in a performance loss. For a sufficiently large network, however, interleaving of pages with the proper contention resolution procedures as in Freshet enables nodes near the middle of the network to complete downloading their code images earlier. This consequence enables nodes not in the middle of the network and also not near the originators the ability to download complete code images earlier than they would through Deluge.

4.4 Small Hardware Implementation

To show that Freshet functions in the physical world, a small sensor testbed was constructed to demonstrate that in small networks Freshet performed with comparable time to Deluge.

The network was constructed through four mica2 motes placed in a line (Fig. 4.17). The radio was set to the lowest power setting so that approximately 10 feet produced one hop communication. The motes were placed 10 feet apart, effectively creating a 3 hop network. While this network is still too small to demonstrate the energy savings potential of Freshet, it can demonstrate that Freshet performs similarly to Deluge, thereby making it a viable protocol.

The next step was to make five runs each for both the native Deluge protocol and Freshet. Each run was set up with a new code image injected into the network. The code sample used in this experiment was 20 pages in size, each page having 48 packets of 36 bytes each. This code would then propagate through the network as per the network reprogramming protocols. Time to completion of each page was observed through motes within range of each Freshet mote. In particular, two motes were used for measuring the time to completion, one set between A and B, another between C and D. These motes observed the messages sent out by each of motes A, B, C, and D. When a mote sent an advertisement message that indicated it had all 20 pages downloaded, then its upload was marked complete.

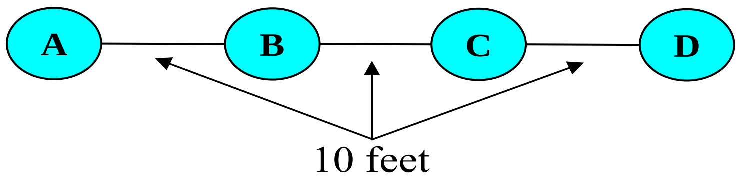


Fig. 4.17 Experimental set up. Node A is the code image source

Since the network is approximately three hops, this set up examines the multihop capabilities of both Freshet and Deluge and makes a valid comparison of their comparative speeds.

The results for these experiments are shown in Table 4.2. Bear in mind that each data point represents the average of 5 runs. These results show that over a very small network Freshet performs just as well as Deluge in disseminating data objects.

Table 4.2
Average time to disseminate 21 pages to each node

Protocol	To Node B	To Node C	To Node D
Deluge	250 s	373 s	475 s
Freshet	247 s	381 s	471 s

4.5 Summary

This chapter presented Freshet, a means of reliably and efficiently distributing code throughout a sensor network. Freshet has two distinct schemes: single source code distribution and multiple source code distribution. TOSSIM simulations show that single-source Freshet is between 20-45% more efficient in energy compared to the Deluge protocol, while requiring about 10% more time for propagating the code. In the case of smaller networks the multiple-source Freshet is shown to be 5% faster than Deluge with multiple sources. Finally, this chapter presented a small-scale physical implementation of Freshet, and it was shown that in small networks Freshet performs the same as Deluge.

5. SENSOR NODE LOCALIZATION WITH DIRECTIONAL ANTENNAS

In this chapter we discuss a technique for localization of an unknown sensor node using one or more nodes with a known location. All nodes are also equipped with directional antennas, which are then used to acquire location estimates.

5.1 Directional Antenna Model

One of the simplest semi-directional antennas is the patch antenna. This antenna is used as a representative example of an antenna that may be used for localization and which will still fit on a mobile form factor. The ideal patch radiation model is a hemispherical radiator which allows for semi-directional radiation. The typical gain of a patch antenna is on the order of 3.5 to 6 dBi, depending on the dielectric substrate used in the design. A representative angular variation of the gain for a typical microstrip antenna will be in the range of $\cos^2\left(\frac{\beta l}{2}\sin(\theta)\right) < G(\theta) < \cos\left(\frac{\beta l}{2}\sin(\theta)\right)$, where β is the free-space constant and l is the longest length of patch, assuming the lowest order mode of operation [44]. The gain is defined as the ratio of the intensity, in a given direction, to the radiation intensity that would be obtained if the power accepted by the antenna were radiated isotropically.

In the E-plane cut, the antenna's radiated e-field from a standard patch radiator is ideally $E = \cos\left(\frac{\beta l}{2}\sin(\theta)\right)$. This pattern dependence is in relation to a coordinate system with the z-axis perpendicular to the microstrip patch radiator. This is the ideal solution for a patch antenna with an infinite ground plane and is only slightly altered using a finite size ground plane. The ground plane is used to shield the radiating field from the rest of the circuitry and the other radiators. Unshielded radiators, such as those that are standard with the motes, are susceptible to parasitic radiating currents which result in asymmetric patterns.

The received power at an antenna is given by $P_r = \frac{P_t G_t(\Theta_t) G_r(\Theta_r)}{r^2}\left(\frac{\lambda}{4\pi}\right)^2$, where Θ_t and Θ_r are the transmitting and the receiving angles, respectively, and r is the distance between the transmitter and the receiver. λ is the RF wavelength of the carrier frequency. Since $(\lambda/4\pi)^2$ is a constant, we will exclude it from future expressions. It was however included in calculating the results.

A realistic antenna radiation pattern obtained from the design of patch antennas in the HFSS simulation package is used to model the gain for the experiments. The simulations were validated through actual experiments in an anechoic chamber. For the sensor network simulations in this paper we use an antenna model given by $G_t(\Theta) = G_r(\Theta) = \cos\left(\frac{\beta L}{2}\sin(\Theta)\right)$. This is the upper bound of the possible antenna gain given in Section 5.3 and is chosen in our analysis and simulation so that an anchor has a larger number of target nodes within its transmission range. However, the proposed techniques are equally valid for any other antenna model.

5.2 Aligned antennas

In a number of practical applications it is reasonable to expect that the sensors will be manually deployed. Sensors set up to monitor a bridge's health have to be placed by construction workers on the bridge for example. In such scenarios even though it may not be possible to know the precise location of the sensor, it is possible to place these sensors in a pre-determined orientation.

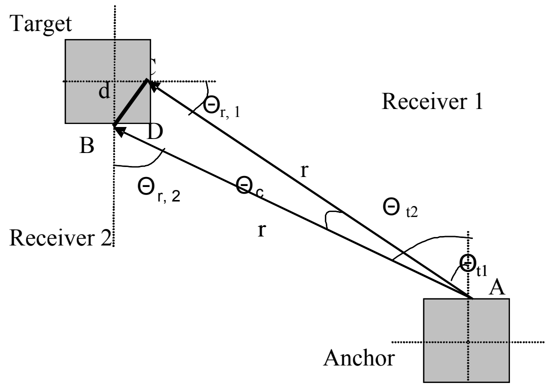


Fig. 5.1 Location determination with aligned nodes

If the antennas of a target sensor node are aligned, then we can use the power received at multiple receiving antennas of the target from a single transmitting antenna on an anchor for position estimation. Without loss of generality consider that an anchor node is placed to the south-east of the target node as shown in Fig. 5.1. The size of the sensor would usually be much smaller than the transmission distance. So $d/r=\Theta_c$. Then received power at the two receiving antennas of the target node is given by equations (5.1) and (5.2) in two variables Θ_1 and r. Since, these are nonlinear equations it is difficult to get a closed form solution for Θ_1 and r in terms of the input variables $P_{r,1}$ and $P_{r,2}$. However, these equations can be numerically solved by standard methods to obtain Θ_1 and r.

$$P_{r,1} = \frac{P_t}{r^2} G_t(\Theta_{t,1}) G_r(\Theta_{r,1}) = \frac{P_t}{r^2} G_t\left(\frac{\pi}{2} - \Theta_1\right) G_r(\Theta_1) \tag{5.1}$$

$$\begin{aligned} P_{r,2} &= \frac{P_t}{r^2} G_t(\Theta_{t,2}) G_r(\Theta_{r,2}) = \frac{P_t}{r^2} G_t(\Theta_2) G_r(\Theta_2) \\ &= \frac{P_t}{r^2} G_t\left(\frac{\pi}{2} + \frac{d}{r} - \Theta_1\right) G_r\left(\frac{\pi}{2} + \frac{d}{r} - \Theta_1\right) \end{aligned} \tag{5.2}$$

Where $\Theta_1 = \Theta_{r,1}$ and $\Theta_2 = \Theta_{r,2}$.

Alternatively, if the orientations of these sensors are not perfect, Θ_1 in (5.1) and (5.2) can be replaced by $\Theta'_1 = \Theta_1 - \Phi_{unaligned}$, where $\Phi_{unaligned}$ can be obtained from a digital compass [51] or some other simple algorithms [50]. A possible approach is mounting an omni-directional antenna with the four directional antennas on the same node and estimating $\Phi_{unaligned}$ from the difference of the received power strength between the directional antennas and the omni-directional antenna.

A baseline experiment for this is with the anchor node having omni-directional dipole antennas. In this case the gain of the transmitter, $G_t(\Theta)$, is constant over all Θ and denoted G_{omni}. Fig. 5.2 shows this configuration. Now, since we know the distance as well as the relative direction of the target with respect to the anchor, we can estimate its position. This estimate is based on measurements from just one neighboring anchor node whereas triangulation requires measurements from at least three anchors.

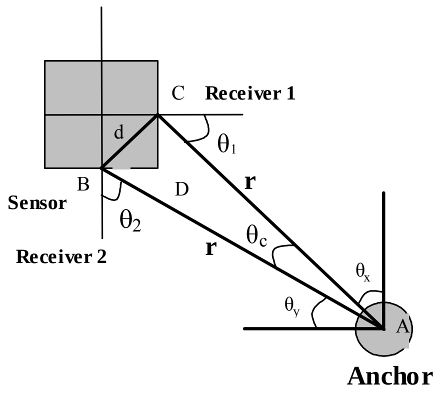


Fig. 5.2 Location determination with an omni-directional transmitter and directional receiver

The estimates from multiple anchors can be averaged to obtain a better estimate of the position. Alternatively, the information about Θ_1 could be discarded and the range measurements (r) can be used to triangulate the position of the sensor in a least squares manner. Both these strategies have been evaluated in our simulations and the averaging strategy has yielded better results.

5.3 Generalization to Unaligned Antennas

In cases where it is not possible to ensure a global orientation of all nodes of a network, additional measurements can be used to estimate position. Received power at two different antennas of the target node from two transmitting antennas of the anchor node is measured. Such an arrangement is shown in Fig. 5.3.

Geometric relations between the various transmission and receiving angles can be derived from the figure.

$$\Theta_2 + \Theta_6 = \Theta_4 + \Theta_8 = \Theta_3 + \Theta_5 = \Theta_1 + \Theta_7 = \frac{\pi}{2} + \frac{d}{r}$$

$$\Theta_1 + \Theta_2 + \Theta_3 + \Theta_4 = \pi$$

$$P_{r,11'} = \frac{P_t * G_t(\pi - \Theta_2 - \Theta_3 - \Theta_4) * G_r(\Theta_2)}{r^2} \tag{5.3}$$

$$P_{r,21'} = \frac{P_t * G_t(\frac{\pi}{2} + \frac{d}{r} - \Theta_3) * G_r(\frac{\pi}{2} + \frac{d}{r} - \Theta_2)}{r^2} \tag{5.4}$$

Let $P_{r,ij}$ denote the power received by antenna *i* on the target node when antenna *j* is transmitting on the anchor node. We can use these equations to simplify the received power equations as follows.

$$P_{r,12'} = \frac{P_t * G_t(\frac{d}{r} + \Theta_2 + \Theta_3 + \Theta_4 - \frac{\pi}{2}) * G_r(\frac{\pi}{2} + \frac{d}{r} - \Theta_4)}{r^2} \tag{5.5}$$

$$P_{r,22'} = \frac{P_t * G_t(\Theta_3) * G_r(\Theta_4)}{r^2} \tag{5.6}$$

Equations (5.3) through (5.6) in the four variables Θ_2, Θ_3, Θ_4, and *r* can again be numerically solved to estimate the location of the target node.

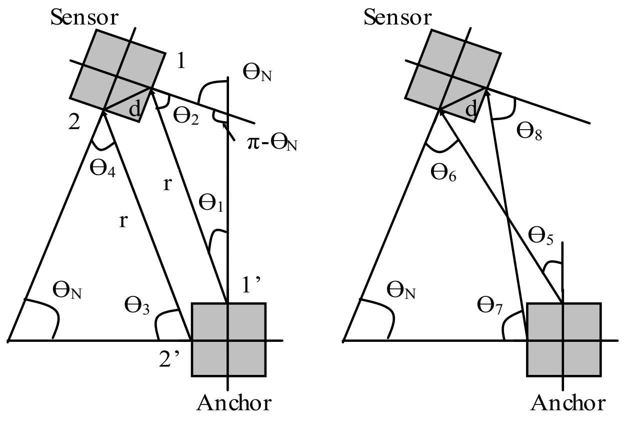


Fig. 5.3 Location determination for unaligned antennas

This scheme requires that two target antennas be able to simultaneously receive transmissions from two anchor antennas. This would require a transmitter beam width of 180˚. This is non-optimal for four antennas covering a 360˚ plane but is a tradeoff for increased degrees of freedom in the orientation of the nodes. Besides, the increased beam-width will lend greater fault tolerance to the system by providing greater redundancy in the areas reached by multiple transmitting antennas. It will also make the antenna design easier since high directionality, i.e. narrow beam width is not needed.

5.4 Aligned Antennas with Two Anchors

The two location determination methods described earlier rely on the difference in power received at two antennas of a sensor from the antennas on the same anchor node. The error in the power received can become correlated due to the proximity of the two antennas, even if they are pointed in separate directions. In a real life scenario the correlation can significantly reduce the accuracy of the location estimate, especially for a very small sensor node. To investigate the performance of the location determination with increasingly uncorrelated channels, two transmitted signals were sent from two motes substantially removed from each other. This scheme is also useful in situations in which more than one directional antenna would not fit on a single mote. The arrangement is shown in Fig. 5.4.

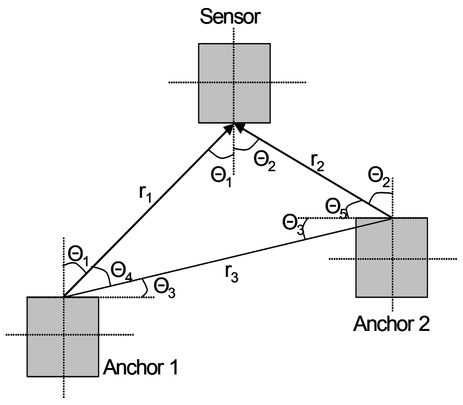


Fig. 5.4 Location determination using measurements from two anchors

Since the location of the two anchors is known, the parameters r_3 and Θ_3 can be determined. Using geometric properties of the system we get the following relations between the various angles

$$\Theta_5 = \frac{\pi}{2} - \Theta_2 \qquad \Theta_1 + \Theta_3 + \Theta_4 = \frac{\pi}{2}$$

The equations for the received power are given by

$$P_{r,1} = \frac{P_t}{r_1^2} G_t(\Theta_1) G_r(\Theta_1) \tag{5.7}$$

$$P_{r,2} = \frac{P_t}{r_2^2} G_t(\Theta_2) G_r(\Theta_2) \tag{5.8}$$

Using the law of sines along with relations between the angles derived earlier we get two more equations

$$\frac{r_2}{\cos(\Theta_1 + \Theta_3)} = \frac{r_3}{\sin(\Theta_1 + \Theta_2)} \tag{5.9}$$

$$\frac{r_1}{\cos(\Theta_2 - \Theta_3)} = \frac{r_3}{\sin(\Theta_1 + \Theta_2)} \tag{5.10}$$

This gives us four equations in four unknowns r_1, r_2, Θ_1, and Θ_2, which can be numerically solved. Thus, the distance and the angle with respect to each of the two anchors are determined. The sensor node's location can be estimated using either distance, angle pair and the final location estimated using averaging of each estimate.

In this section, we have provided the mathematical solution to the problem of location estimation with directional antennas in three different scenarios. The node specifications and the deployment conditions will determine which scenario is applicable.

5.5 Localization Experiments and Results

Most of this work was completed with Nipoon Malhotra and Chin-Lung Yang, so the experimental results are included and explained in appendix A. The next section describes work that was accomplished primarily by the author, the simulation results of directional antenna networks.

5.5.1 Simulation results

We are interested in evaluating the accuracy of the location determination protocols with varying number of anchor nodes. The accuracy is measured for between 2 and 30 anchors of different kinds: omni-directional, directional, unaligned, and the two anchor case. This range of experiment would be beyond the hardware resources of our test bed and hence a simulation methodology is used. These simulations are only intended to highlight the behavior of the schemes with increasing node densities. The distortion in received power is simulated using a Rician fading channel. Each data point is based on the average of 50 different samples. For each sample, the anchor nodes are placed randomly within a 10x10 meter grid. The transmitted power is -10 dB, equivalent to that used in hardware, and the size of the sensor node is set at 10 cm.

Fig. 5.5 is shown with respect to the increasing number of neighbors. From top, the different curves correspond to: single anchor unaligned, three omni-directional anchors and omni-directional target node, single anchor aligned antenna with least square error aggregation, single anchor aligned antenna with averaging for aggregation, two anchors with aligned antenna. Except for the omni-directional case, all the others have patch antennas on the target node. The first three form one group and the last two form another. The errors in the first group are noticeably higher than those in the second group. The estimates in both groups show a horizontal trend beyond 14 anchor neighbors indicating that higher density is not required for location estimation purposes. The generalized orientation, however, requires up to 20 neighboring anchor before stabilizing. The highest error is observed for the generalized orientation, indicating the importance of approximate alignment at the least. The next highest error is for the omni-directional anchor antenna, followed by the single anchor with least square error aggregation. This indicates the value of directional antennas and averaging as the method for aggregation. The two anchor case with aligned antenna and using averaging as the aggregation technique outperforms the single anchor case at lower densities but is statistically equivalent at higher node densities. The error bars corresponding to 95% confidence interval are shown.

We observe that averaging as the method for aggregating gives much better results than least square estimation. One of the primary reasons for this phenomenon is that averaging cancels the errors in estimates in individual measurements while the effect of error in least squares estimation is additive.

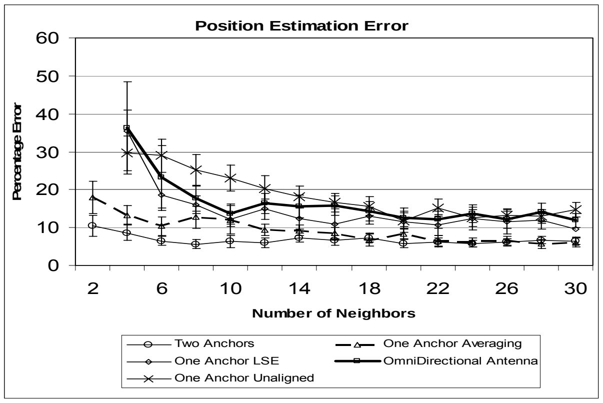


Fig. 5.5 Evaluation of estimation error for varying number of neighboring anchors

The results shown in Fig. 5.5 have slightly lower error rates than those found in hardware. This is most likely due to a limited ability to account for all multipath propagation, fading, and other environmental affects. Moreover, the aligned cases were perfectly aligned in the simulation, while the experiments had relatively aligned antennas, but within a margin of error that is very difficult to determine.

5.6 Summary

This chapter has presented various techniques for location determination in ad-hoc networks using directional antennas. The combination of the schemes is designed to meet the variety of requirements and degrees of freedom for real life applications. The solution approach can form the foundation for location determination protocols for the increasingly popular directional antennas. Experimental results are shown in appendix A and simulation results are shown in this chapter. These results bring out the fact that location estimation with omni-directional antennas requires at least three anchors and is less accurate than with directional antennas. Also, using uncorrelated communication channels through two geographically separated anchor nodes produces better results than multiple antennas on the same anchor node. The error is reduced from 27.5% in an omni-directional system to 11.6% with two directional anchor nodes.

6. TRUST-BASED FAULT TOLERANCE

In this chapter we discuss a protocol for masking faulty or malicious nodes within the network. It is possible that reprogramming motes may require a means to avoid or remove nodes with faulty behavior patterns. This chapter presents a protocol that may be applied to such a case, as well as more general event-detection schemes.

Recent innovations made in the fields of electronics and wireless communication have enabled the advent of sensor networks. These networks comprising of thousands of inexpensive sensor nodes can be set up with relative ease by placing the nodes in predefined locations manually or through the use of robots, as well as by random deployment of self-organizing nodes. A wide gamut of applications ranging from health, home, environmental to military and defense make use of sensor nodes for collection of appropriate data. The sensor nodes comprising of data collecting, processing, and transmitting units are very small in size and can be densely deployed owing to their low cost.

Sensor nodes have serious limitations in available resources, such as power, memory, and processing ability [54]. The sensor nodes and wireless links are prone to failure, while the network is also open to various malicious attacks. While significant research has been done in the areas of communication architecture, routing, and energy conservation in sensor networks, development of fault tolerance in this highly volatile scenario remains an interesting open research issue. Conventional fault tolerance and intrusion tolerance protocols do not translate well to the sensor network domain due to its large scale and the resource constraints on the sensor nodes.

As stated, this chapter considers fault tolerance in an event driven model for sensing. An event driven model of behavior for sensing finds many applications in civilian, military as well as industrial scenarios. Examples could be seismic monitoring to detect and locate tremors in a given area, or military applications to sense any movement within a cordoned-off area. The inherent unreliability of sensor nodes makes fault tolerance in such an environment an important concern. The problem is essentially one of aggregating data from multiple sensor nodes to decide if an event has occurred and determining the location of the event, in the face of natural and malicious failures in both the sensing nodes as well as the aggregating nodes. In particular, our approach looks at arbitrary faults in the sensor networks, whether natural or malicious. Natural arbitrary faults may arise suddenly and intermittently in sensor networks, thereby causing a node to miss reporting an event (missed alarms) or falsely reporting an event that has not occurred (false alarms). Malicious faults occur when some nodes in the network have been compromised by an adversary. This adversary can make the nodes send out corrupt information intended to adversely affect the data gathering role of the network. These malicious nodes, depending on their level of intelligence, may have some knowledge of how the network functions and can to behave in a manner to escape detection.

The goal of the proposed TIBFIT protocol involves event detection and location determination in the presence of faulty sensor nodes, coupled with diagnosis and isolation of faulty or malicious nodes. The accuracy of the system is defined in terms of fraction of instances when an event occurrence is correctly detected, and its location determined within the given error bound.

The approach followed by the protocol is to maintain state of the sensing nodes in terms of the fidelity of their previous sensing actions, and use this information in making decisions involving those sensing nodes. Sensor nodes report the occurrence and location of events to a data sink, and remain silent otherwise. The data sink then decides on whether the event occurred and where based on the aggregated data. To determine the location of the event the data sink must aggregate all reports from nodes within the detection radius. The aggregation could be a simple voting scheme. However voting is a stateless approach and does not reflect on the past performances of the sensing nodes. TIBFIT introduces a new parameter called *trust index* for this purpose. The *Trust Index* (referred to as TI) of a node is a quantitative measure of the fidelity of previous event reports of that node as seen by the data sink. In a system comprised of sensing nodes, the data sink assigns and maintains a TI for each node in its domain, and does voting in a stateful manner. As the system runs over a longer time, more state is built up concerning the performance of the associated sensing nodes, and hence tolerance for faults also goes up accordingly. So while the simple voting approach falls apart when more than 50% of the nodes within detection range of the event are corrupted, TIBFIT can tolerate faults in a network with more than 50% of its nodes compromised *after* it has built up adequate state of the nodes.

To demonstrate the effectiveness of TIBFIT, we use an event-driven simulation with ns-2. All nodes are considered liable to fail, whether in a natural or a malicious manner. We group the nodes into four categories: a) non-faulty nodes that naturally fault some percentage of the time; b) faulty nodes that err randomly; c) malicious nodes working independently that err occasionally and attempt to subvert the system but also try to remain undetected; d) malicious nodes that collaborate and err occasionally and attempt to subvert the system but also try to remain undetected. We show through simulation that TIBFIT is capable of accurately detecting and determining locations of events even when more than 50% of the network is compromised. Finally we also simulate a system that has a gradually increasing number of malicious nodes and analyze the accuracy of the system.

The main contributions of this protocol and its evaluation are the following:

1. TIBFIT tolerates nodes that fail both naturally and maliciously, and makes decisions on event occurrence as well as location**.** Under several scenarios, accurate event determination and localization can be done even with more than 50% of the network compromised. We also demonstrate diagnosis and limited recovery in the system.

2. No nodes are considered immune to failure, whether they are sensing nodes or the data sink.

3. We have come up with an adversary model with increasing levels of sophistication and demonstrate the effectiveness of the protocol in each case.

4. The protocol is generic and can be applied to any data sensing and aggregation application in sensor networks.

6.1 Related Work

As in any sensor networks problems, we require a great deal of related material to ensure that our model accounts for the many challenges of creating a functioning wireless sensor network. For instance, [69] gives an algorithm that guarantees reliable and fairly accurate output from different types of sensors when at most *k* out of *n* sensors are faulty. [68] gives a fault tolerant way of averaging sensor data, and the author also gives a control process to deal with individual sensor failures. [70] deals with multi-sensor data fusion and assumes that the biggest loss in sensor network efficiency is from sensor readings. They propose a method of handling sensor failures through substitution of another on-board sensor. [32], [33], and [35] provide techniques of localization for finding node position, such as triangulation and lateration. Nodes within sensing range of this mobile node must be able to determine the location of this node. Location determination efforts with directional antennas can aid in finding the location of such a mobile node. In [65] it is shown that given signal strength and attenuation model one can estimate sensor location. Given enough fixed anchor nodes Bagchi *et al.* present a technique for finding an unknown node within some range of error [40].

There appears to be a dearth of existing work related to our specific topic of data fault tolerance in sensor networks. Schaeffer *et al.* discuss decision making concerned with propagating an alert through a network [59]. They set a threshold for event propagation, where if a node hears more than *n* nodes announce an alert then that node sounds the alert. They analyze the characteristics of this network with false alarms and missed alarms, where the evaluation is on whether the event notification reaches some data sink. They address natural faults exclusively and do not consider cases with faulty nodes colluding.

Wagner discusses aggregation of data in a sensor network with malicious intruders in [62]. The author presents a mathematical framework for analyzing the vulnerabilities of common aggregation functions and then presents the mathematical basis for secure aggregation functions, such as average with trimming. The work presented here can complement this by providing trimming of some failing nodes so that the aggregation can work on the remaining data set. However, the paper does not address the problem of in network aggregation, which is covered here through the analysis of failure prone CHs. It admits the aggregation functions break down with more than half the network compromised. Also, the paper presents the case for aggregation with redundant deployments of cheap, crude sensor nodes.

Koo shows an upper bound on the tolerance of a broadcast decision process as approximately 1/π of the network being compromised [53]. This model is proven theoretically with arbitrarily powerful malicious nodes.

6.2 System Model

All nodes in the network are identical and are arranged into disjoint clusters, each with a set of *cluster heads* (CHs), only one of which is active at any point in time. The CH serves as the data sink for its particular cluster. The nodes in a cluster are within one hop communication of the CH. The clusters themselves are formed randomly around the elected CHs. The CHs are rotated over time and CH election is based on energy-related parameters of the constituent nodes. In each cluster, the node that is chosen to be the CH knows the topology of the cluster. Nodes that are within the detection range of an event are called *event neighbors* for that event. This topology is illustrated in Fig. 6.1.

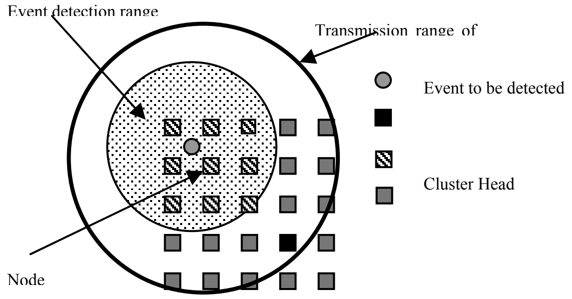


Fig. 6.1 Event detection

When an event occurs, all the event neighbors are expected to report the occurrence of the event to the CH. The CH makes a decision on whether the event has occurred based on the reports received from the event neighbors and their *trust indices*. A detailed description of the TI model follows in Section 6.3.

The sensor network is deployed by placing the nodes randomly in the network. It is assumed that the nodes have the ability to determine their own locations. This can be accomplished through GPS mechanisms, deploying nodes with deterministic mobility in known locations and using triangulation methods to compute their positions as functions of time, etc. Further discussion is beyond the scope of this paper. The locations of the nodes at a given time are known to the CHs, but not necessarily to the non-CH nodes. The network could be stationary or mobile, as long as it is possible for the CH to estimate the positions of its cluster nodes during decision making. The sensor nodes function in an event-driven model, that is, they sense the environment for occurrence of a particular detection-level event and transmit data only if they sense such an event. We will assume that the event is typically detectable by multiple nodes, which makes our protocol practical. This assumption is not unreasonable for many practical sensor deployments.

We adopt the low energy, adaptive hierarchical clustering protocol (LEACH), for cluster formation as well as CH election [55],[56]. This protocol architecture aids in the formation of self-organizing clusters, with dynamically chosen CHs. Each node is assigned a probability of becoming a CH at the beginning of each round, which depends on the number of times it has been made CH previously and the energy available in the node. These properties help spread energy usage equally throughout the network. We have also incorporated the TI of the node as an additional parameter to be considered for CH election. The TI of the node has to be higher than a threshold value to ensure that only sufficiently trusted nodes can become CHs. This is not a property of the original LEACH protocol.

Each node independently decides if it wishes to be a CH. Once a node decides to become a CH, it broadcasts this information. Any node that receives advertisements from *n* different contending CHs, affiliates itself with a single CH based on the strength of the signal received. If a node's TI is below a certain threshold, the central base station will cancel this node's effort to become a CH and re-initiate CH election. A CH that reaches the end of its leadership period sends the aggregate TI information that it has gathered for all nodes in its cluster to the base station before ending its leadership. A newly CH elected for an existing cluster requests the base station for TI information for nodes in its cluster.

We group event detection into two categories – binary event detection and event detection with location determination. Binary event detection leads to the system recognizing the occurrence of the event with a binary decision about whether it happened or not and not being concerned with the location of the event. An example could be detection of a forest fire based on the temperature reaching a critical threshold. Location determination is when the coordinates of the event are also reported by the sensing node. In the forest fire example, the sensor can detect environmental changes such as wind and variation in light intensity in a direction and estimate the location of the oncoming fire.

6.2.1 Failure Model

The nodes in the network may fail due to accidental failures or may be compromised by an adversary and therefore exhibit failure due to malicious causes. Three types of failure scenarios are possible. A node may have a missed alarm where it does not report an event within its sensing radius to the data sink within a specified time. A node may provide a false alarm where it either reports an event outside of its sensing radius or reports an event within its sensing radius that did not occur. A node may exhibit a location faults where it reports an event but at the wrong location. Flooding based denial of service (DoS) attacks are not considered in this paper.

Four categories of sensing nodes are identified. *Correct nodes* are not assumed to be 100% accurate, but are expected to make errors within a specified bound referred to as *natural error rate. Faulty nodes* form the superset for nodes with natural or malicious failures. A faulty node can exhibit *naïve* behavior in terms of randomly sending out corrupt information following no specific pattern. The node lies arbitrarily, either in dropping an event report, falsely reporting an event, or reporting a faulty location (level 0). A *smart* faulty node is aware partially of the system model and tries to retain its TI at a reasonably high level where it estimates it will not be detected and isolated. If a malicious node's TI is reaching a level at which it will either be dropped from the network or its vote has too little influence on the event decision, then the node will stop lying until its TI is raised sufficiently. The *smart* faulty nodes may lie independently (level 1) or in collusion (level 2). The colluding nodes are assumed to be connected in a way that is undetectable by the reliable nodes in the network.

6.3 TibFit Design

The goal of the TIBFIT protocol is to determine whether an event has occurred from analyzing reports from the event neighbors. To combat failures in the reporting nodes, each node is assigned a TI, maintained at the CH, to indicate its track record in reporting past events correctly. The TI is a real number between zero and one and is initially set to one. For each report a node makes that is deemed incorrect by the CH, the node's TI is decreased. Similarly, for each report a node makes that is deemed correct by the CH, the node's TI is increased, but not beyond one. Thus correctly functioning nodes will have a TI approaching one while faulty and malicious nodes will have a lower TI.

We assume that correct nodes are allowed to make occasional errors due to natural causes. The rate of these errors is denoted the natural error rate (NER). The TI is decremented exponentially. Nodes that make mistakes are penalized more for earlier mistakes, and find it more difficult to regain their previous trust levels. This is considered better than a linear model where a node that lies 50% of the time would still occasionally have the trust index value of one. If a node errs more frequently than its NER its index decreases, while if it errs less frequently then its index increases.

The TI is calculated as follows. Let the natural error rate be f_r (<1). A variable v is maintained for each node at the CH. Each time a node makes a report deemed faulty by the CH its v is incremented by the expression $1-f_r$. Each time a node makes a report deemed to be correct by the CH its v is decreased by f_r if v is larger than zero. The TI is calculated as

$$TI = e^{-\lambda v}$$

where λ is a proportionality constant that is application dependent. An uncompromised node's TI is expected to remain at the same value. It can be expected to suffer a fault at the rate of one per every $1/f_r$ events and the expected change in v is:

$$E[\Delta v] = (1 - f_r) - \left(\frac{1}{f_r} - 1 \right) * f_r = 0$$

The design of the protocol is explained next by successively relaxing some simplifying assumptions.

6.3.1 Binary events

Let us initially assume that event reports are binary in nature simply specifying whether the event has occurred or not. All the nodes in the cluster, say k, are event neighbors for any event detected by the cluster. A sensing node can detect the occurrence of an event perfectly for events that happen within a radius r_s surrounding the node. All the nodes within radius r_s of an event E are called event neighbors for E.

After the CH receives the first event report, it calculates the k event neighbors for the event. The CH then waits for a predefined interval of time T_{out} for event reports to be received from these nodes. After T_{out} has elapsed, the CH partitions the event neighbors into two sets R and NR based on whether they have reported the occurrence of the event or not, respectively. The trust indices of each group are summed and the group with the higher cumulative TI (CTI) wins out. The trust index values of nodes in the *winning* group are increased while the index values of nodes in the *losing* group are decreased according to the formula given above. It should be noted that a smaller group of reliable nodes can win the vote against a larger group of unreliable nodes based on higher TI for the individual reliable nodes earned over past events. This process provides detection, diagnosis, and masking of the fault.

It is evident that we do not need a TI model for a system with faulty nodes in the minority. A simple voting would suffice to mask the decision of the faulty nodes. However, consider a system where the density of faulty nodes increases over time. Examples could be batteries of the nodes dying out with time, or existing nodes being compromised by adversaries. The faulty nodes which have been in operation for a while would have had their TIs reduced to low values. Hence even when the total number of faulty nodes is in a majority, their CTI may still be lower than that of the correct nodes. Hence, TIBFIT can lead to correct aggregation as well as diagnosis even with more than 50% of the nodes compromised. It is obvious that if the initial condition consists of faulty nodes being in the majority, then the protocol will be unsuccessful in tolerating faults. After time, the system can identify a faulty node when its TI falls below a certain threshold. It can then be removed from the network.

6.3.2 Location determination

In this section we build on the previous model by adding location details to the event reports. The event report consists of location in terms of (r, Θ) with respect to that node. The nodes do not sense the location of the event perfectly and the CH must determine the actual location of the event. One sensor network problem that can be solved through this extension is where a network is attempting to track a mobile sensor node that is transmitting a signal as it moves throughout the network.

Simplifying Assumptions: Let us assume there is a time difference of at least T_{out} between any two events to avoid overlapping event neighbors. A correct event report sent in by a sensing node reports the location of an event to within a radius r_{error} surrounding the event.

Once time T_{out} has elapsed after the first event report, let there be k other reports that have come in from the nodes in the cluster during this time. The CH performs a clustering algorithm based on K-Means which groups these k event reports into a number of *event clusters* based on the locations indicated by the reports [66]. Each event cluster represents a possible location where the event could have occurred, as indicated by the reports. The clustering algorithm is a heuristic based on K-Means, so as to minimize the sum of squares error.

Goal of the algorithm presented below is to organize the event reports into disjoint event clusters of radius $r_{error.}$ Let C be the set of all event clusters consisting of elements $\{C_1, C_2 \ldots C_r\}$. Let $\{c_1, c_2 \ldots c_r\}$ be the centers around which the event clusters $\{C_1, C_2 \ldots C_r\}$ are formed. Let d(x, y) denote the distance between two points x and y. $d(c_i, c_j) > r_{error}\ \forall\ C_i, C_j \in C$. $C_k.cg$ (Center of gravity) denotes the average location indicated by all event reports in cluster C_k.

Event clusters are created using the following procedure.

(1) The clustering algorithm is started once T_{out} has elapsed after the first event report. The set of all event reports in this time T_{out} is referred to as E. The distances between each pair of event reports are computed and sorted in a 2D array.

(2) Let E_1 and E_2 be event reports from the set E with the greatest distance between them. Event clusters C_1 and C_2 are created with E_1 and E_2 as centers, and C_1, C_2 are added to C.

(3) Condition for any event report E_k to form a separate event cluster is that $d(E_k, c_i) > r_{error}\ \forall C_i \in C$. The set E is iterated through and the set of all cluster centers are identified, so that the remaining event reports are at a distance of less than r_{error} from at least one element in C, i.e., the remaining event reports cannot form separate event clusters.

(4) Once the initial set of clusters in C are formed, remaining event reports in E are added to one of the clusters in C based on which cluster center it is nearest to. $C_k.cg$ for the clusters are updated appropriately.

(5) If the centers of two or more clusters lie within r_{error} of each other the clustering algorithm is repeated by forming a new cluster center at the weighted average of these centers. The rounds are executed until no change in cluster constituency takes place in a new round.

The final elements in C represent the set of all events. $C_k.cg$ represents the location of the event *k.* The event neighbors can be determined for the location determined and a determination of whether an event has occurred is made based on the trust indices of the associated nodes as in Section 6.3.1. This design successfully throws out event reports from nodes which make a localization error of more than r_{error}.

6.3.3 Concurrent events

Additions: In this section we build on the previous model by assuming that multiple events can occur within T_{out} of each other (referred to as concurrent events from here on). We however assume that concurrent events cannot occur closer than a distance of r_{error}.

(1) When the CH receives the first event report *E1*, a symbolic circle of radius r_{error} is drawn around it. A new timer $E1.T_{out}$ is started for the associated event reports from other event neighbors to come in. All subsequent events that lie within r_{error} of *E1* reported within time T_{out} are added to the same circle.

(2) If any subsequent event report E_k received lies outside this circle, a new circle of radius r_{error} is formed with this event report E_k as its center. Associated $E_k.T_{out}$ is started.

(3) Once time $E_k.T_{out}$ has passed from the reception of event report E_k that is the center of a circle, all the event reports inside this circle are put into a group and the clustering algorithm described in the previous section is performed on them to determine the location of the event.

(4) However if one or more other circles overlap with this circle, then the CH must wait until time T_{out} has elapsed for all such overlapping circles. The clustering algorithm is performed on the union of all event reports in all the overlapping circles to determine the event clusters and thus how many events have actually occurred.

6.3.4 Unreliable Cluster Heads

Though the CHs are chosen based on high TI values, it is still possible for a selected CH to fail. To combat this problem we assign two additional shadow cluster heads (SCH) to each cluster such that the SCHs can monitor all input and output traffic associated with the selected CH. The SCHs themselves may be considered to be reliable as they are chosen based on the fact that they have the highest trust indices among nodes within one hop of the CH. The SCHs listen in to the communication going in and out of the CH and perform all the functions as the CH except transmitting the aggregated event reports to the base station. On perceiving a wrong conclusion being drawn at the CH based on the input data, the SCHs also send the result of their own computations to the base station. The base station, on receiving data from all CHs in the cluster, does a simple voting to arrive at the right conclusion. It also prompts CH election in that cluster to pick a new CH and reduces the TI of the previous faulty CH. Thus, only a single CH failure can be tolerated.

TIBFIT can also be extended to scenarios where the sensing nodes are more than one hop away from the data sink. The data sink still needs to know the location of the constituent node and reliable data dissemination primitive needs to be introduced to ensure that the data sent out by the sensing nodes reliably reach the data sink without alteration[14],[67].

6.4 TibFit Analysis

In this section we analyze the probability associated with the CH successfully identifying a binary event in the presence of faulty nodes.

Consider a baseline model with no trust indices assigned to the nodes. Let us assume that there are *N* event neighbors, of which *m* are faulty. The probability of a successful report from a correct node is *p*, and the probability of a successful report from a faulty node is *q*. Let **X** be the random variable that is the number of correct reports from correct nodes, and **Y** be the random variable indicating the same for the faulty nodes. They are defined:

$$P\{X = k\} = \binom{N-m}{k} p^k (1-p)^{N-m-k}$$

$$P\{Y = k\} = \binom{m}{k} q^k (1-q)^{m-k}$$

The probability that the *N-m* correct nodes make *k* or more correct reports is therefore the sum of the probabilities from *k* to *N-m*, and from *k* to *m* for faulty nodes.

Define the random variable **Z**=**X**+**Y**. We wish to know the probability that **Z** has a majority of the N votes which is the probability that the event is successfully identified. The expressions are shown in equations (6.1), (6.2), and (6.3). These expressions map to Fig. 6.2 with N=10, q=0.5, and p=0.99, 0.95, 0.90, 0.85.

The accuracy begins to fall off steeply once fifty percent of the network is compromised. TIBFIT can tolerate both an increase in faulty nodes over time and more initial nodes being faulty, and will therefore outperform this baseline case. Next we will show how TIBFIT performs over time.

Consider the TIBFIT model. Assume the network initializes with N nodes with 1 faulty node and N-1 correct nodes. We will corrupt the nodes in the network at a constant rate of one after every k events and show how the system still functions with 100% accuracy till N-3 nodes are corrupted, thereby outperforming the baseline case which drops in accuracy once 50% of the nodes in the system are compromised. Without loss of generality, let us assume that N is odd. We also make the simplifying assumption that correct nodes are always correct and the faulty nodes always fail. Let $\mathrm{CTI}_{\mathrm{correct}}$ be the CTI of the set of correct nodes and $\mathrm{CTI}_{\mathrm{faulty}}$ be the CTI of the set of faulty nodes.

After every k events a good node is compromised. After *(N-2).k* rounds, total number of correct nodes is 3, and faulty nodes is *N-3*. $\mathrm{CTI}_{\mathrm{correct}}$ is 3 as correct nodes are always correct and each have a TI of one. After the first faulty report, the TI of a node becomes $e^{(-\lambda)}$. Therefore after k rounds, the TI of the faulty node would be $e^{(-k\lambda)}$. So, $\mathrm{CTI}_{\mathrm{faulty}}$ for *(N–3)* faulty nodes when the newest addition to the faulty set has made k errors would be $e^{-k\lambda} + e^{-2k\lambda} + \ldots + e^{-(N-2)k\lambda}$.

$$P(success) = P\left(\overline{Z} \geq \left\lfloor \frac{N}{2} \right\rfloor + 1\right) = \sum_{j=1}^{\lceil N/2 \rceil} P\left(\overline{Z} = \left\lfloor \frac{N}{2} \right\rfloor + j\right), \text{now let } i = \left\lfloor \frac{N}{2} \right\rfloor + j - k \qquad (6.1)$$

$$P(success) = \sum_{j=1}^{\lceil N/2 \rceil} \left(\sum_{k=\left\lfloor \frac{N}{2} \right\rfloor + j - m}^{\min\left(\left\lfloor \frac{N}{2} \right\rfloor + j,\, N-m\right)} \left(\binom{N-m}{k} * p^k (1-p)^{N-m-k} * \left(\binom{m}{i} * q^i (1-q)^{m-i} \right) \right) \right) \quad m \leq N-m \quad (6.2)$$

$$P(success) = \sum_{j=1}^{\lceil N/2 \rceil} \left(\sum_{k=\left\lfloor \frac{N}{2} \right\rfloor + j - (N-m)}^{\min\left(\left\lfloor \frac{N}{2} \right\rfloor + j,\, m\right)} \left(\binom{m}{k} * q^k (1-q)^{m-k} * \left(\binom{N-m}{i} * p^i (1-p)^{N-m-i} \right) \right) \right) \quad m > N-m \quad (6.3)$$

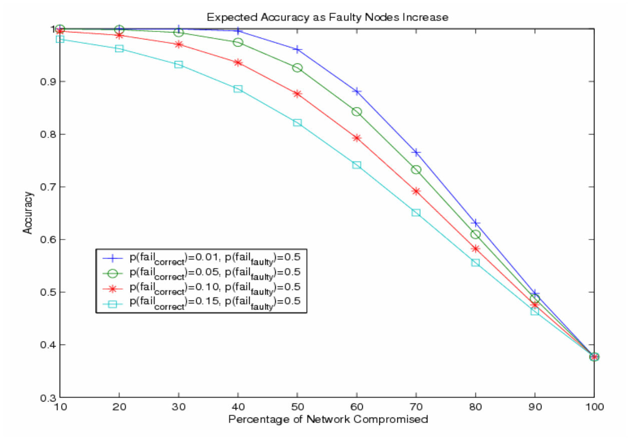


Fig. 6.2 Expected accuracy of the network as the percentage of faulty nodes increases

For the system to be 100% accurate, CTI of correct nodes ($CTI_{correct}$) should always be greater than CTI of faulty nodes (CTI_{faulty}). For a correct node to be corrupted, CTI_{faulty} should be infinitesimally close to 1, so that $CTI_{correct}$ -1 > CTI_{faulty}+1 (a node is transferred from the good side to the bad side). We have the following expression:

$3-1>1+e^{-k\lambda}+e^{-2k\lambda}+\ldots+e^{-(N-2)k\lambda}$, or $2=\dfrac{1-e^{-(N-1)k\lambda}}{1-e^{-k\lambda}}\rightarrow 0=e^{-k\lambda(N-1)}-2e^{-k\lambda}+1$, which can be solved with Matlab.

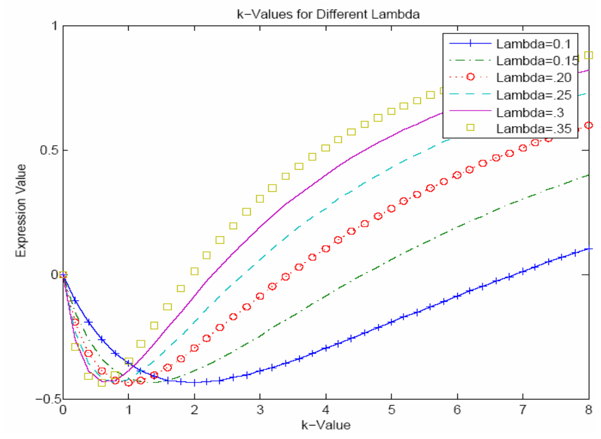


Fig. 6.3 Variation of k with different λ values

Fig. 6.3 shows this expression for several different λ values. Wherever a given line crosses the x-axis that is the value of *k* and the number of rounds after which a good node can be made into a faulty node. Expectedly as λ increases, the frequency of nodes failing that can be tolerated increases since the TI degrades more rapidly with failures. It is for this reason we chose λ=0.25 for our simulations, so that we could create a fair number of data points but without needing a very large number of events to show the beneficial effects of TIBFIT.

The upper limit on k is the k necessary to make three good nodes tolerate an additional failure. We stop the analysis at two because once the system has two good nodes left then the sum of the faulty nodes' trust indices must be less than zero to allow the addition of a bad node, which is impossible. When there are 3 good nodes left in the system, then $3 > CTI_{faulty}$, where $CTI_{faulty} = 3-\varepsilon$, $\varepsilon>0$. After k_{max} rounds from this state, let us assume that one more correct node can be transferred to the faulty side. Therefore after k_{max} rounds the value of CTI_{faulty} should be = $1- \varepsilon$ before the transfer. Solving $3*e^{-kmax\lambda}=1-\varepsilon$ gives us $k_{\max} = \frac{1}{\lambda}\ln 3$ as $\varepsilon \to 0$. Hence, the maximum number of rounds needed to tolerate another faulty node is $\frac{1}{\lambda}\ln 3$.

6.5 TibFit Simulation and Results

The TIBFIT protocol is simulated using the *network simulator – ns-2* [58]. A sensing radius of 20 units is considered. Events are generated at regular time intervals by the *event generator,* using a uniform random variable to generate X and Y coordinates uniformly distributed in the network. The event generator informs the event neighbors of the event and its location.

We run three different experiments. In experiment 1 we show the accuracy of the binary event model versus percentage of the network compromised by level 0 faulty nodes. In experiment 2 we show the accuracy of the location event model versus percentage of the network compromised by level 0, 1, and 2 faulty nodes. In experiment 3 we show the accuracy of the location event model versus time, where the percentage of the network compromised increases linearly over time.

For each simulation we use either the TIBFIT system that uses the trust index, or we use the baseline system, which uses majority voting to make event decisions.

Experiments are run with faulty nodes belonging to only one level for a given experiment. Nodes are stationary in all experiments.

6.5.1 Experiment 1 – binary events

A cluster of ten nodes is formed, and all nodes are considered event neighbors for every randomized event. Level 0 faulty nodes are used for the fault model, generating both missed alarms and false alarms. The CH makes a decision regarding occurrence of the event based on the data forwarded to it from the sensing nodes.

Table 6.1

Parameters for Experiment 1

Type of Event	Binary Event Model
Independent Variable	Percentage Faulty Nodes: varied from 40%-90%
Correct Nodes NER	0, 1, and 5%
Faulty Nodes NER	Level 0:Missed Alarm 50% False alarm 0,10, and 75%
Size of network	10 sensing nodes, 1 CH
Number of Event Neighbors	10
Events per simulation	100
λ	0.1
Fault rate (f_r)	Same as NER

For this experiment we started simulations with 40% of the network compromised. As Section 6.4 shows, even for the baseline system, the probability of failure with less than 40% of the network compromised is very small, and therefore not simulated.

The results in Fig. 6.4 include only missed alarms. The most noteworthy result from this experiment is that the network can have 70% of its nodes compromised and still maintain over 85% accuracy. This result is superior to the analytical results shown in Fig. 6.2 in Section 6.4.

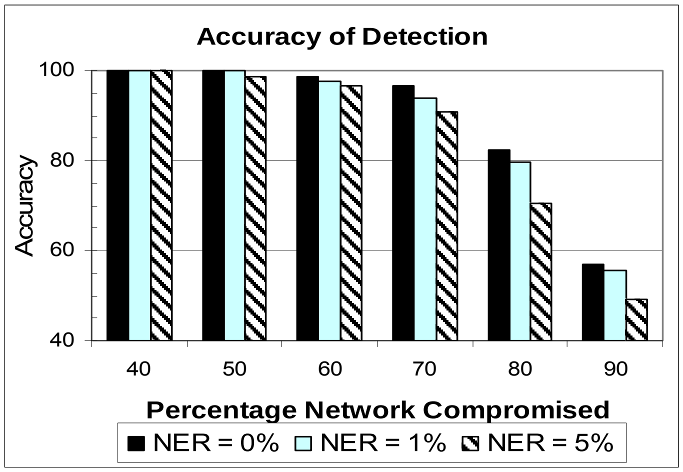


Fig. 6.4 Experiment 1 – 50% accurate faulty Nodes, missed alarms only

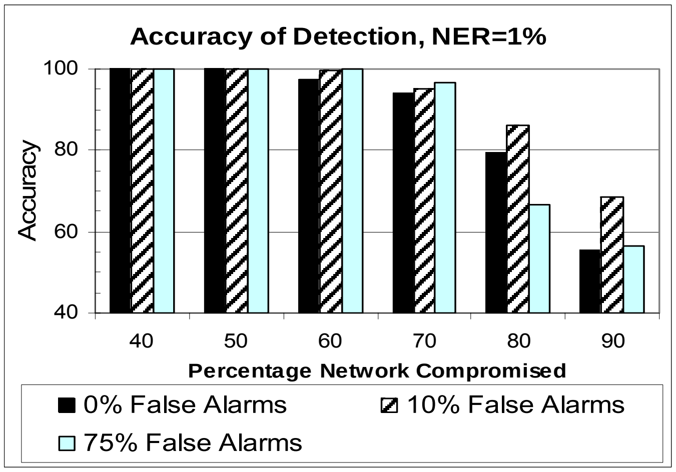


Fig. 6.5 Experiment 1 – 50% accurate faulty nodes, missed alarms and false alarms

Fig. 6.5 shows the simulation with both false alarms and missed alarms from faulty nodes. All correct nodes have 1% NER. Again, the network performance starts to degrade with 70% faulty nodes. The interesting results is that 75% false alarms shows the best accuracy when less than 80% of the network is compromised, indicating that the excessive false alarms lower faulty nodes' TIs and therefore increase system reliability. At 80% faulty nodes with 75% false alarms, accuracy falls dramatically, as the system is no longer able to tolerate the excessive false alarms. 10% false alarms maintains the highest accuracy at this point, indicating that occasional false alarms lower faulty nodes' trust indices enough to outperform 0% false alarms.

6.5.2 Experiment 2 – location determination model

In the second type of simulation, 100 nodes are placed uniformly on a 100X100 grid. The CHs and event generator are two other entities present in the network. The CH decides on both the occurrence of the event as well as its location. The network is a single cluster, and the CH knows the positions of all 100 nodes. All nodes can reach the CH in a single hop. For location estimation r_{error} is 5 units. Table 6.2 shows various experimental parameters for this experiment. Due to the ns-2 wireless model, correct nodes' packets are naturally dropped less than 1% of the time.

A lower threshold (lowerTI) of 0.5 is used for level 1 and level 2 nodes to ensure their trust indices do not fall too low. If they reach the lower threshold they behave like a correct node until they reach an upper threshold (upperTI) of 0.8, after which they begin erring again. Each node reports an event with error in both the X and Y directions as dictated by a Gaussian random variable with standard deviation σ.

The error percentage indicated in Table 6.2 is calculated as the joint probability distribution of the two Gaussian rv's, which are Rayleigh distributed, and it indicates the probability a node reports an event more than 5 units away from the actual event location. The standard deviation for a correct node is much less than that for a faulty node. Level 1 nodes work independently, while level 2 nodes collude with each other and all either send the event report for the same location or do not send the event report.

Table 6.2
Parameters for Experiment 2

Type of Event	Location Determination Concurrent or single events
Independent variable	Percentage faulty nodes, varied from 10%-58%
Error rate for correct nodes	Location report has std. deviation of 1.6 or 2.0
Error rate for faulty nodes (levels 0, 1, and 2)	Location report has std. dev. of 4.25 or 6.0, drop packets 25% of the time
Size of network	100 sensing nodes, 5 CH
Number of event neighbors	Variable on location
λ	0.25
Fault rate (f_r)	0.1 (different from NER for wireless channel model losses)

This experiment initialized a network with a percentage of the network compromised by Level 0, 1, or 2 malicious nodes. 58% was the upper limit for the compromised network as past this point the system did not work with much accuracy. The output accuracy metric was the number of events detected by the CH within r_{error} of the actual event. Simulations are run with both concurrent and single events. The legend format for all the result figures from this point on is "*Lvl M W-Z [TIBFIT or Baseline]*", where *M* is the type of malicious node used, *W* is the standard deviation of the correct nodes, *Z* is the standard deviation of the malicious nodes, and the final parameter is whether the TIBFIT or the baseline model was used.

The results in Fig. 6.6 show that at low percentages of the network compromised, the TIBFIT system and the baseline system perform similarly. However, after 40% of the network is compromised, the TIBFIT model performs better than the baseline model by at least 7% percent, and by as much as 20% percent. More importantly, TIBFIT has accuracy near 80% even with faulty nodes having errors 70% of the time. A consequence of the execution of the network with TIBFIT is that the trust index values of the faulty nodes continue to decrease and once they reach the threshold, the nodes can be removed from the network, thus eliminating them from causing future damage.

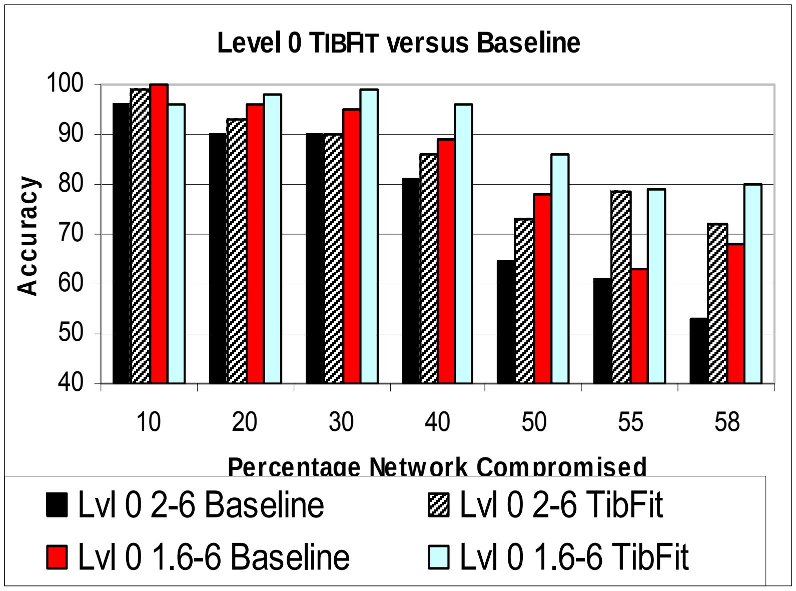


Fig. 6.6 Experiment 2 – Level 0 faulty nodes

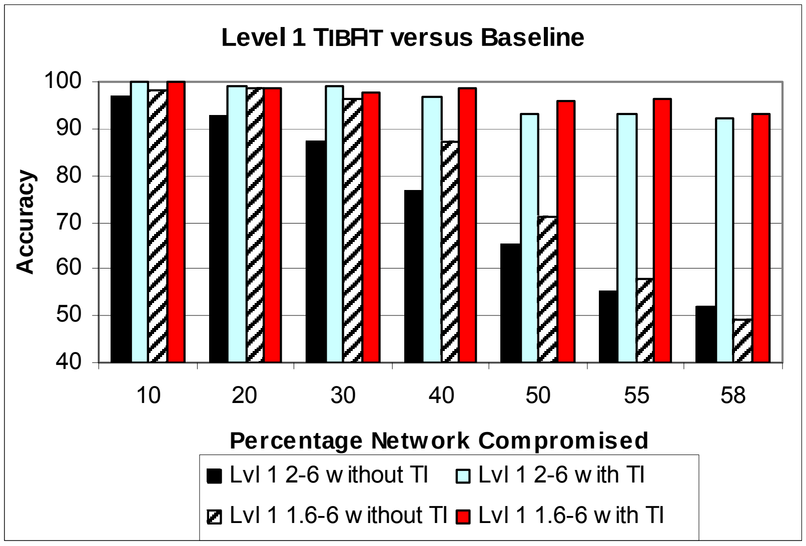


Fig. 6.7 Experiment 2 - Level 1 faulty nodes

The second graph for location estimation, shown in Fig. 6.7, is for level 1 nodes. The result shows that even with 58% of the network compromised, TIBFIT's accuracy remains over 90%. In contrast, the baseline model falls well below that level once the network reaches 40% malicious nodes. The reason for this trend is that the level 1 nodes lie with intention to keep them from being detected. In effect, the trust index forces the malicious nodes to lie less frequently and therefore helps to improve the accuracy of the event determination.

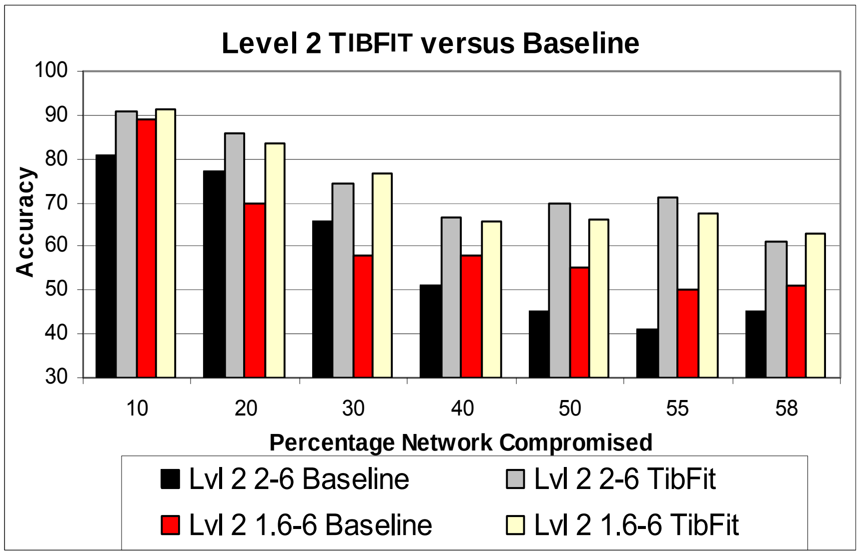


Fig. 6.8 Experiment 2 – Level 2 faulty nodes

Fig. 6.8 shows results for level 2 malicious nodes. It shows that these nodes dramatically reduce the accuracy of the network, although the TIBFIT still outperforms the baseline model. It is clear from this figure that even the trust index has trouble tolerating level 2 type faults due to the collaborative nature of the nodes.

Fig. 6.9 shows level 0 nodes with concurrent events compared to single events, both simulations using TIBFIT. The concurrent events occur with uniform distribution simultaneously, although never within r_{error} of each other. The graph indicates that tolerating concurrent events does not significantly alter the success of the nodes in accurate detection of events.

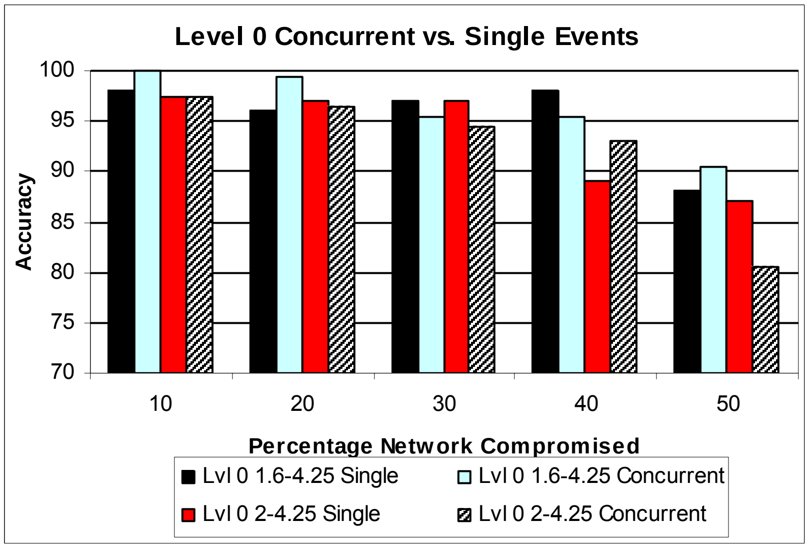


Fig. 6.9 Experiment 2 – Single and concurrent events

6.5.3 Experiment 3 – decay of network

The next simulation increases the percentage of the network compromised by malicious nodes linearly over time. The network is initialized with 5% of the network compromised by level 0 faulty nodes. After every 50 events 5% more of the network is compromised until 75% of the network is compromised.

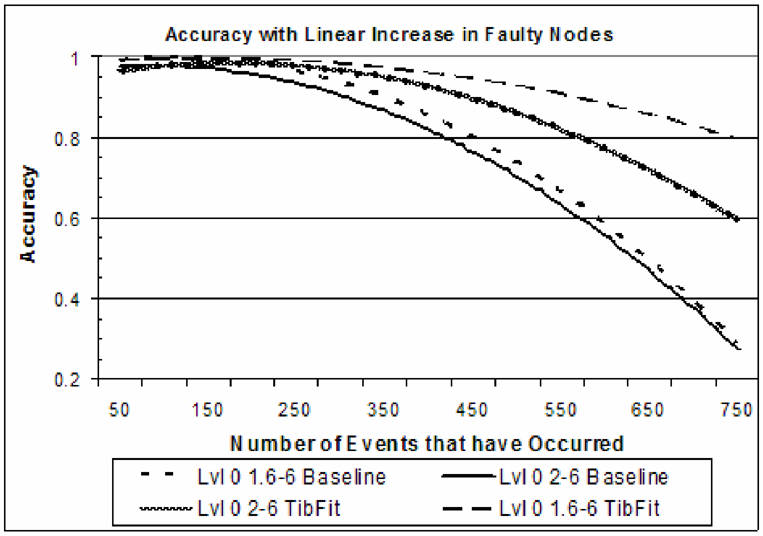


Fig. 6.10 Experiment 3 – Linear increase in number of faulty nodes

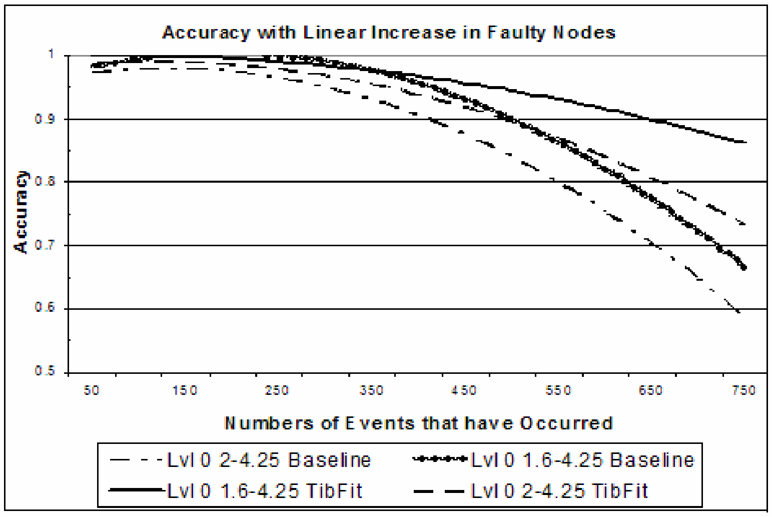


Fig. 6.11 Experiment 3 – Linear increase in number of faulty nodes

Fig. 6.10 and Fig. 6.11 show that over time TIBFIT outperforms the baseline model in all cases. This occurs because the trust indices of the faulty nodes decrease over time and the system can then handle the transition of some correct nodes to faulty nodes. It is important to compare only the lines with the same standard deviation parameters, because for some time the baseline model with 1.6-4.25 outperforms the TIBFIT 2-4.25 case, although after a longer period of time the TIBFIT line does better, even though it has a higher fault rate in its correct nodes. What is also notable is that the TIBFIT network maintains nearly 80% accuracy even with 60% of the network compromised.

6.6 Summary

This chapter presented a protocol called TIBFIT that maintains state for event decisions in a sensor network. This protocol can handle both binary event detection and event location estimation with high accuracy in the face of natural and malicious node failures within the network. The protocol outperforms the standard voting scheme for event detection. We also define two types of intelligent malicious fault models that can disrupt a network, and find that using TIBFIT malicious nodes acting independently are successfully tolerated. However, the accuracy of TIBFIT in a system of colluding nodes is not as high though it outperforms the baseline voting scheme.

7. CONCLUSIONS

This work aims to create more dependable sensor systems. There are multiple aspects to this problem, but this work has chosen to focus on three specific problems: sensor network retasking, sensor node localization, and sensor node fault tolerance.

Through TOSSIM simulations and some hardware results it is shown that network reprogramming can be both reliable and energy efficient. This work presents a protocol called Freshet that builds on the standard network reprogramming protocol Deluge. Freshet shows nearly 40% more efficiency in energy usage than Deluge, primarily due to effective radio management. Freshet also proposes a means of expediting disseminating code with multiple senders through interleaving of pages. All these results tend towards a system that will operate longer and more reliably than one using the original Deluge protocol.

The next aspect of dependable middleware considered is that of sensor node localization. Location information inevitably improves sensor network performance by informing the network of things that it is otherwise unaware. In some cases, especially removing faulty nodes, localization is necessary for any effective functionality. However, due to localization equipment's high cost, it is necessary to find means of disseminating localization information without expensive equipment attached to each node. This work shows how inexpensive directional antennas can be applied to this problem. A hardware implementation of these antennas shows over two times less error in localizing nodes than do the traditional omni-directional nodes.

Finally, we deal with fault tolerance in sensor networks. Inevitably some reliability or security concerns will arise with deployed nodes, and thus it is necessary to provide a means to combat these problems. This thesis presents TIBFIT, a trust-based fault tolerance protocol for masking and detecting faulty nodes within a network. Simulations in NS-2 show that TIBFIT can tolerate faulty and malicious nodes with great reliability. In certain fault models TIBFIT can tolerate a network that is over 50% compromised, a non-intuitive and important result. If nodes fail frequently enough within a TIBFIT network then those nodes may be removed; this removal process is improved through directionality, which could be provided by the localization scheme proposed earlier.

In summary, this thesis provides three steps towards more reliable sensor network middleware. Additional work remains to be done as far as hardware implementations are concerned. It would be very valuable to test Freshet in an extensive hardware implementation and hopefully then employ TIBFIT and localization schemes to tolerate potential faults that may arise within this network. A system equipped with all of these tools would provide a very flexible and robust network of nodes that could potentially prove very valuable in practical use.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Jonathan W. Hui and David Culler, “The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale,” in *Proceedings of the 2nd International Conference on Embedded networked sensor systems*, Nov. 2004, pp. 81-94.

[2] Philip Levis, Neil Patel, David Culler, and Scott Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation*, 2004, pp. 15-28.

[3] J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, “Building efficient wireless sensor networks with low-level naming,” in *Symposium on Operating Systems Principles*, 2001, pp. 146-159.

[4] P. Levis and D. Culler, “Maté: a Virtual Machine for Tiny Networked Sensors,” in *Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[5] S. R. Madden, “The Design and Evaluation of a Query Processing Architecture for Sensor Networks,” PhD thesis, UC Berkeley, December 2003.

[6] Thanos Stathopoulos, John Heidemann and Deborah Estrin, “Remote Code Update Mechanism for Wireless Sensor Networks,” CENS Technical Report # 30, http://lecs.cs.ucla.edu/~thanos/moap-TR.pdf.

[7] S. S. Kulkarni and Limin Wang, “MNP: Multihop Network Reprogramming Service for Sensor Networks,” in *Proceedings of the 25th International Conference on Distributed Computing Systems*, June 2005, pp. 7-16.

[8] Jaein Jeong and David Culler, “Incremental Network Programming for Wireless Sensors,” *The First IEEE International Conference on Sensor and Ad hoc Communications and Networks*, Oct. 2004, pp. 25-33.

[9] Andrew Tridgell, “Rsync,” At: http://samba.org/rsync/

[10] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 151–162.

[11] S. K. Kasera, G. Hjálmtýsson, D. F. Towsley, and J. F. Kurose, “Scalable reliable multicast using multiple multicast channels,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, 2000, pp.:294–310.

[12] J. Luo, P. Eugster, and J. P. Hubaux, “Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks,” *IEEE INFOCOM*, 1-3 April 2003, pp. 2229-2239.

[13] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, “Negotiation-based protocols for disseminating information in wireless sensor networks,” *Wireless Networks*, vol. 8, no. 2, 2002, pp.169–185.

[14] Gunjan Khanna, Saurabh Bagchi, and Yu-Sung Wu, “Fault Tolerant Energy Aware Data Dissemination Protocol in Sensor Network,” in *Proceedings of the IEEE Dependable Systems and Networks Conference*, June 28-July 1, 2004, pp. 739-748.

[15] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurthy, “PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks”, *First ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, September 28, 2002.

[16] S.-J. Park, R. Vedantham, R. Sivakumar and I.F.Akyildiz, “A Scalable Approach for Reliable Downstream Data Delivery in Wireless Sensor Networks,” *ACM International Symposium on Mobile Ad hoc Networking and Computing*, May 2004, pp. 78-89.

[17] F. Stann and J. Heidemann, “RMST: Reliable data transport in sensor networks,” in *Proceedings of the First IEEE International Workshop on Sensor Net Protocols and Applications*, April 2003, pages 102–112.

[18] University of California, Bereley, “TinyOS,” At: http://www.tinyos.net/.

[19] Crossbow Technology, Inc., “Mote In-Network Programming User Reference Version 20030315, 2003,” At: http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/Xnp.pdf.

[20] G. Bianchi, “Performance analysis of the IEEE 802.11 Distributed Coordination Function,” in *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, March 2000, pp. 535-547.

[21] J. H. Kim and J. K. Lee, “Performance analysis of MAC protocols for wireless LAN in Rayleigh and shadow fast fading,” in *Proceedings of the IEEE Global Communications Conference*, vol. 1, Nov 1997.

[22] V. Shnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh. “Simulating the Power Consumption of Large-Scale Sensor Network Applications,” in *Proceedings of the 2nd International Conference on Embedded networked sensor systems*, 2004, pp. 188-200.

[23] Y.-B. Ko and N. H. Vaidya, “Location-Aided Routing (LAR) in Mobile Ad Hoc Networks,” *ACM/BaltzerWireless Networks Journal*, nol. 6, no. 4, 2000, pp. 307-321.

[24] Jörg Widmer, Martin Mauve, Hannes Hartenstein, Holger Füßler, “Position-Based Routing in Ad-Hoc Wireless Networks,” in *The Handbook of Ad Hoc Wireless Networks*, Mohammad Ilyas, ed., CRC Press, Boca Raton, FL, U.S.A., 2002.

[25] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein, “Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet,” in *Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002, pp. 96-107.

[26] J. M. Kahn, R. H. Katz and K. S. J. Pister, “Mobile Networking for Smart Dust,” in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, WA, August 17-19, 1999, pp. 271-278.

[27] Garmin’s eTrex, at www.garmin.com/products/etrex/spec.html

[28] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, “The cricket location-support system,” in *Mobile Computing and Networking*, 2000, pp. 32-43.

[29] L.-C. Kuo, “ 3-D FDTD Design Analysis of A 2.4 GHz Polarization-Diversity Printed Dipole-Antenna with Integrated Balun and Polarization-Switching Circuit for WLAN and Wireless Communication Application,” *IEEE Trans. Microwave Theory and Techniques*, vol. 51, no.2, Feb 2003, pp.374-381.

[30] J. Preiss et al., “Polarization diverse antenna for portable communication devices,” U.S. Patent 6 031 503, 2000.

[31] M. A. Jensen, and Y. Rahmat-Samii, “Performance Analysis of Antennas for Handheld Transceivers Using FDTD”, *IEEE Trans. Antennas and Propagation*, vol. 42, 1994, pp. 1106-1113.

[32] Jeffrey Hightower and Gaetano Borriello, “Location sensing techniques,” Technical Report of the University of Washington CS Department, UW-CSE-01-07-01, July 2001.

[33] J. Hightower and G. Borriello, “Location systems for ubiquitous computing,” *IEEE Computer*, August 2001, pp. 57-66.

[34] Andy Harter, Andy Hopper, Pete Steggles, Any Ward, and Paul Webster, “The anatomy of a context-aware application,” in *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking*, August 1999, pp 59-68.

[35] N. Bulusu, J. Heidemann, and D. Estrin, “GPS-less low cost outdoor localization for very small devices,” *IEEE Personal Communications Magazine*, Oct. 2000, pp. 28-34.

[36] J. Hightower, R. Tower, and G. Borriello, “SpotON: An indoor 3d location sensing technology based on RF signal strength,” Technical Report of the University of Washington, CS Department, February 2000.

[37] D. Wood and J. A. Stankovic, “Denial of service in sensor networks,” in *IEEE Computer*, vol. 35 no. 10, Oct. 2002, pp. 54–62.

[38] Jan Beutel, “Geolocation in a PicoRadio Environment,” M.S. Thesis, ETH Zurich, December, 1999.

[39] Chris Savarese, Jan Rabaey, Koen Langendoen, “Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks”, *USENIX Technical Annual Conference,* Monterey, CA, June 2002.

[40] S. Cabuk, N. Malhotra, L. Lin, S. Bagchi, and N. Shroff, “Analysis and evaluation of topological and application characteristics of unreliable mobile wireless ad-hoc network,” in *10th Pacific Rim Dependable Computing Conference*, March 2004.

[41] N. Sundaram and P. Ramanathan, "Connectivity based location estimation scheme for wireless ad hoc networks," in *Proceedings of Global Communications Conference*, vol. 1, Nov. 2002, pp.143-147.

[42] L. Doherty, L. El Ghaoui, K. S. J. Pister, “Convex Position Estimation in Wireless Sensor Networks,” *IEEE INFOCOM 2001*, April 2001.

[43] Kay Römer, “The Lighthouse Location System for Smart Dust,” in *ACM/USENIX International Conference on Mobile Systems, Applications, and Services***,** May 2003, pp. 15-30.

[44] P.J.B. Clarricoats, Y. Rahmatt-Samii, J.R. Wait, *Handbook of Microstrip Antenna*, Volume 1, IEEE Electromagnetic Waves Series 28, 1989.

[45] M. Vossiek, L. Wiebking, P. Gulden, J. Wieghardt, C. Hoffmann, P. Heide, "Wireless local positioning," *IEEE Microwave Magazine*, Dec. 2003, pp. 77-86.

[46] HFSS simulation tool. Available at: http://www.ansoft.com/products/hf/hfss/overview.cfm

[47] L. Boccia, G. Amendola, G. Massa, "Proper modeling using an FEM electromagnetic simulator leads to the design of a low-cost, lightweight GPS patch antenna capable of excellent multipath rejection," *Microwave and RF*, Jan. 2003.

[48] Crossbow Technology Inc., MICA2 Mote: http://www.xbow.com/.

[49] D. Niculescu, B. Nath, "Ad Hoc Positioning System (APS) Using AOA," *IEEE INFOCOM,* 2003, pages 1734-1743.

[50] L. Fang, P. J. Antsaklis, et. al, "Design of a wireless dead reckoning pedestrian navigation system – The NavMote experience," Technical Report, ISIS Lab, University of Notre Dame. [Online]. http://www.nd.edu/~isis/techreports/isis-2004-001.pdf

[51] "DMC-SX Digital Magnetic Compass – Operator Manual," Leica Vectronix AG, Switzerland.

[52] K. Chintalapudi, A. Dhariwal, R. Govindan, G. Sukhatme, "Ad-Hoc Localization Using Ranging and Sectoring," *IEEE INFOCOM*, 2004, pages 2662-2672.

[53] C-Y Koo. "Broadcast in Radio Networks Tolerating Byzantine Adversarial Behavior," in *ACM Symposium on Principles of Distributed Computing*, 2004, pp. 275-282.

[54] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "A Survey on Sensor Networks," *IEEE Communications Magazine*, Aug. 2002, pp. 102-114.

[55] W. Heinzelman, J. Kulik, and H. Balakrishnan. "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," in *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 174-185.

[56] W. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660-670, Oct 2002.

[57] J. Lu, T. Suda. “Coverage-aware Self-scheduling in Sensor Networks,” *IEEE Computer Communications Workshop,* Oct 2003.

[58] Ns-2 simulator. http://www.isi.edu/nsnam/ns.

[59] S. E. Schaeffer, J. C. Clemens, P. Hamilton. “Decision Make in a Distributed Sensor Network,” in *Proceedings of the Santa Fe Institute Complex Systems Summer School*, Santa Fe, NM, USA, 2004. Santa Fe Institute. To appear.

[60] G. J. Pottie and W. J. Kaiser. “Wireless Integrated Network Sensors.” *Communications of the ACM*, vol. 43 no. 5, May 2000, pp. 51-58.

[61] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan. “Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks,” in *Proceedings of the International Conference on Mobile Computing and Networking*, 2001. pp. 272-287.

[62] D. Wagner. “Sensor networks: Resilient aggregation in sensor networks,” in *ACM Workshop on Security of ad hoc and sensor networks*, 2004. pp. 78-87.

[63] Hoblos G., Staroswiecki M., Aitouche A. “Optimal Design of Fault Tolerant Sensor Networks” in *International Conference on Continuous Applications*, Sept. 2000, pp. 467-72.

[64] S. Cabuk, N. Malhotra, L. Lin, S. Bagchi, and N. Shroff, “Analysis and evaluation of topological and application characteristics of unreliable mobile wireless ad-hoc network,” in *Proceedings of the 10th Pacific Rim Dependable Computing Conference*, March, 2004.

[65] J. Hightower, R. Tower, and G. Borriello, “SpotON: An indoor 3d location sensing technology based on RF signal strength,” Technical Report of the University of Washington, Computer Science Department, February 2000.

[66] Sanjiv K. Bhatia, “Adaptive K-Means Clustering,” in *Proceedings of Florida Artificial Intelligence Research Symposium,* 2004.

[67] "Design and Analysis of Hierarchical Key Management for Scalable and Energy Efficient Secure Communicationon Sensors," Issa Khalil, Ness Shroff, and Saurabh Bagchi. CERIAS Tech Report TR-2003-33, Purdue University.

[68] Loren Schwiebert, Sandeep K.S. Gupta, “Research challenges in wireless networks of biomedical sensors,” in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001, pp. 151-165.

[69] K. Marzullo, “Tolerating failures of continuous valued sensors,” *ACM Transactions on Computer Systems*, vol. 8, no. 4, pp. 284-304, November 1990.

[70] F. Koushanfar, M. Potkonjak, A. Sangiovanni-Vincentell, “Fault tolerance techniques for ad-hoc sensor networks,” in *Proceedings of IEEE Sensors*, vol. 2, June 2002, pp. 1491-1496.

APPENDIX

A. LOCALIZATION EXPERIMENTAL RESULTS

A.1 Experimental Setup

Crossbow MICA2 motes MPR400CB operating at 900 MHz and running TinyOS as the programming environment are employed as the sensor nodes for our testbed. Two kinds of omni-directional antennas are used - quarter wave whip antennas (MMA400CA) (on the transmitting anchor node, comes off-the-shelf with the motes) and quarter wavelength monopole antennas with an expanded ground plane (on the receiving target nodes, fabricated for easy interfacing and co-existence with the patch antennas). Directional antennas are fabricated and used on both the target sensor node, whose location is to be determined, and the anchor sensor nodes, whose location is assumed to be known. Patch antennas fabricated on duriod substrates, Rogers RO3010 whose dielectric constant is 10.2, are chosen as the semi-directional antennas in this testbed due to their simple fabrication, small size, and low-cost. The other candidate designs such as Yagi antennas, horn antennas, and antenna arrays, would have more directionality but are larger and more expensive and therefore not applicable. Sensor motes with four directional antennas controlled by a switching network according to the received signal strength indicator (RSSI) at the antennas are employed to be the target motes (functioning as receivers). The switching network is implemented by a GaAs MMIC SP4T switch. Software executing on the target motes monitors the received signal power on the four antennas and selects the requisite ones (the best, or the two best) for its location computation. The motes are tested in an outdoor environment to observe the performance of the location determination system with different kinds of wireless fading. The goal of the experiments is to determine the estimation error of the location determination system.

Three experiments are set up with transmitting anchor motes that initially have omni-directional (dipole) antennas (see Fig. A.2(a)). If necessary, aligned patch antennas are mounted on the anchor motes (see Fig. A.2(b) and (c)) to have a complete directional transmitting and receiving system. The target motes, functioning as receivers, are equipped with the switched directional antennas and collect the data which is forwarded to a laptop through a Universal Asynchronous Receiver/Transmitter (UART) Interface. Matlab programs are written to solve the equations given in Sections 5.2, 5.3, and 5.4 numerically. The software utilizes a gain pattern derived from the patch antenna from Ansoft's High Frequency Simulation Software (HFSS) package. HFSS, a full wave electromagnetic simulation commercial software, is the default industry standard RF design simulation tool [46][47]. The radiation pattern is simulated and compared with the measurement of the fabricated patch (Fig. A.1). It is observed that the two are in close agreement and hence the HFSS model is used for further analysis. The experiments in Section 0 have 150 samples for each position, taken in three separated time intervals. These are averaged for the reported number. The largest 95% confidence interval for these experiments was 12% relative error. Matlab simulation is used to iteratively solve the equations to minimize the error in the radial distance and the angle of reception.

If the iterative Matlab simulation is considered too expensive to execute on the motes, a static lookup table of signal strength versus radial distance and angle of reception can be created and uploaded into the motes. This cuts down on the latency of the location estimation as well as the computational expense on the simple processors of the sensor nodes, at the expense of the accuracy of the solution. This approach will be further investigated in the future.

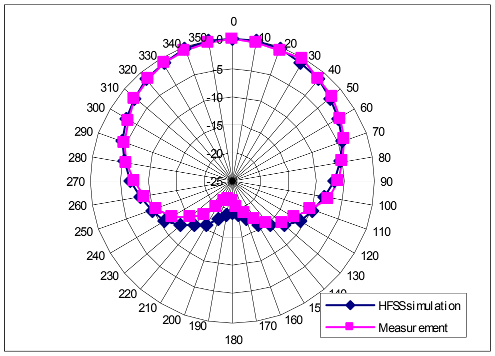


Fig. A.1 Radiation pattern of the patch antenna from HFSS and the measurement in anechoic chamber room

The location error is defined by the error distance from the estimated target position $\hat{r}_{target}$ to the actual target position $\vec{r}_{target,0}$ divided by the known actual distance between the anchor mote and the target mote ($R = \left|\vec{r}_{anchor,0} - \vec{r}_{target,0}\right|$) where $\vec{r}$ is a two-dimensional position vector. Location error R_{error}

$$= \frac{\left|\hat{\vec{r}}_{target} - \vec{r}_{target,0}\right|}{\left|\vec{R}\right|} = \frac{\left|\hat{\vec{r}}_{target} - \vec{r}_{target,0}\right|}{\left|\vec{r}_{anchor,0} - \vec{r}_{target,0}\right|} = \frac{\sqrt{(\hat{x}_{target} - x_{target,0})^2 + (\hat{y}_{target} - y_{target,0})^2}}{\sqrt{(\hat{x}_{anchor} - x_{target,0})^2 + (\hat{y}_{anchor} - y_{target,0})^2}} \text{(A.1)}$$

A.1.1 Experiment 1: aligned case with single omni-directional antenna anchor, dual patch antenna target

In Experiment 1, an anchor node is used with an omni-directional or dipole transmit antenna. The target node has a dipole antenna and two directional patch antennas. Test 1 is the case with a dipole antenna on the target node. Test 2 has the directional antennas on the target node. The target node is placed in the center of a circle of radius d = 8 feet and d = 24 feet. Received power measurements are made with the anchor node at different points on the periphery of the circle. Calculations were computed using equations (5.1) and (5.2) with G_r independent of Θ.

The relative distance error is shown in Fig. A.3 for both tests. The x-axis is the angle of the anchor node with respect to the north-south axis drawn from the target node, with the anchor node being moved on the circumference of a circle with the target node at the center. When using the directional antennas, the target node selects the two strongest signals on its receiving antennas to execute the location estimation algorithm.

When using the omni-directional antenna, the position cannot be determined since at least three anchor nodes would be needed and hence, the distance estimate is used as the basis for comparison. In this case the Friis' formula is used with an experimentally determined parameter for the exponent in the power loss, R^N, N=1.89 in the measurement environment. This parameter was determined using the omni-directional antenna and then generalized for the experiment setting and applied in all calculations. It is reasonable that the value of N would be constant over a section of the sensor field and does not have to be calibrated for each source-destination pair.

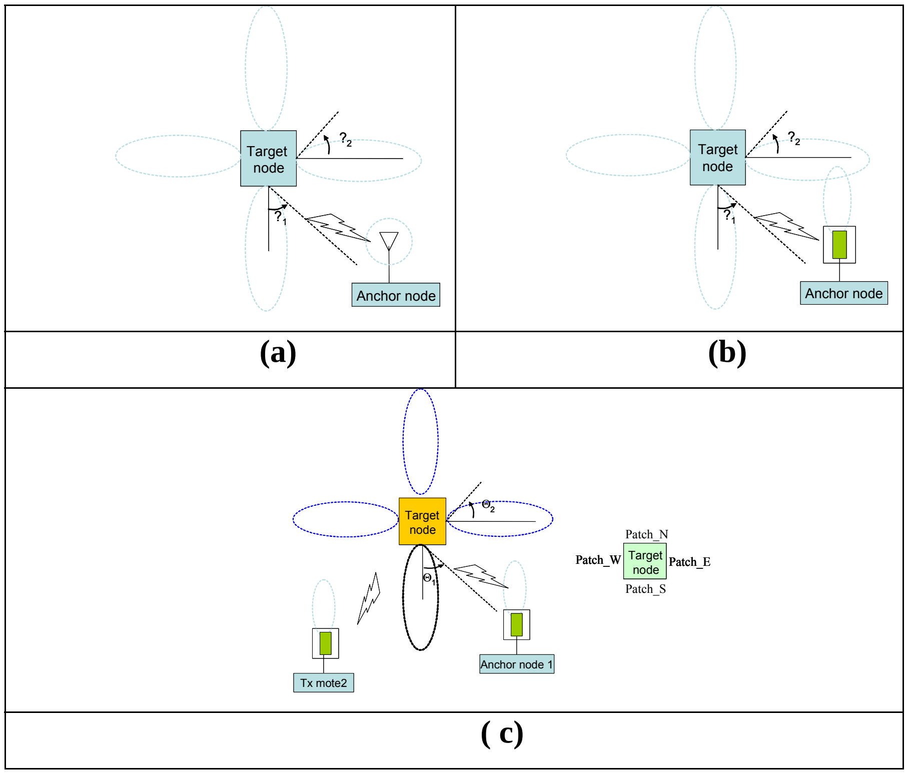


Fig. A.2 Antenna configurations for (a) Experiment 1 (b) Experiment 2 (c) Experiment 3

This is classified as a line-of-sight and relatively multipath free environment and is applied to estimate $\left|\hat{\vec{R}}\right| = \left|\hat{\vec{r}}_{TX,0} - \vec{r}_{RX,0}\right|$. The distance error in the entirely omni-directional antenna system is in the range of 0-105% with a mean of 34%, while this error is reduced to 0-90% with a mean of 23% in the case of the directional antennas.

This result can be explained by the fact that in omni-directional antennas, there are more multi-path effects leading to more interference and more fluctuation in received signal strength. Thus, even if triangulation is used, directionality of the antennas is useful since it leads to more accurate estimates of individual distances.

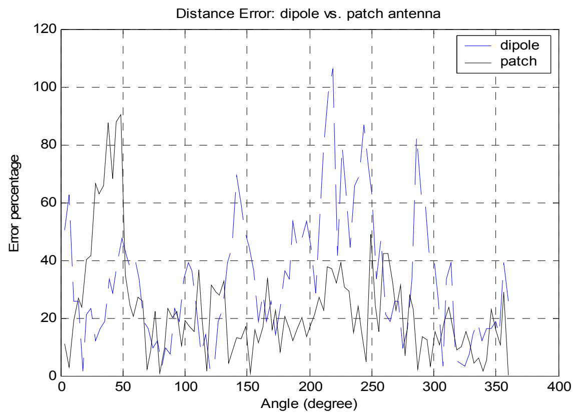
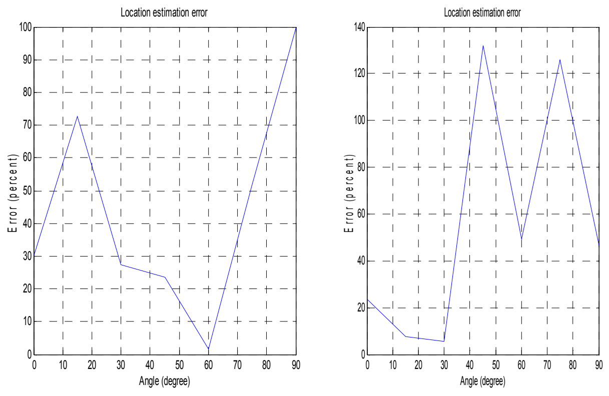


Fig. A.3 Relative error in distance estimation for Experiment 1



(a) (b)

Fig. A.4 Location estimation error for Experiment 1, Test 2 for (a) 8 feet, (b) 24 feet

However, by using the angular information that is gained from multiple receiving antennas, the location of the target node can be determined with a single mote without the use of triangulation (see Section 5.2). The location error of test 2 in Experiment 1 is shown in Fig. A.4. The error is larger than the radius error because the angle estimation error also contributes to the location error. On average, the location errors are (a) 43.15% and (b) 55.75%. This motivates the need to use dissemination of location information through multi-hop communication for large distances. This error can be further mitigated by using other estimation anchors, as Experiment 3 shows.

A.1.2 Experiment 2: single patch antenna anchor with dual patch antenna target

In this experiment directional patch antennas are employed in the transmission motes (see Fig. A.2(b)). The receiving antennas are all directional. The analytical model is as shown in Section 5.2. The errors in measured distance and angle are shown in Fig. A.5 and the maximum errors in the two metrics are seen to occur at different positions. The location estimation error calculated from actual measurements is shown in Fig. A.6. These calculations were made using equations (5.1) and (5.2). The location error grows larger as Θ_1 increases because the transmission antenna is not oriented towards the target node.

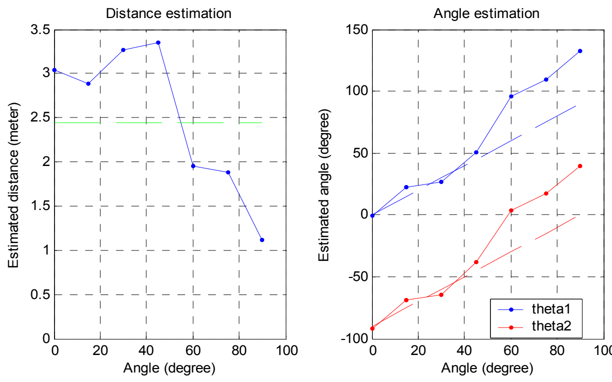


Fig. A.5 Distance and angle estimation error for Experiment 2. [The solid line is the estimation and the dashed line is actual measurement.]

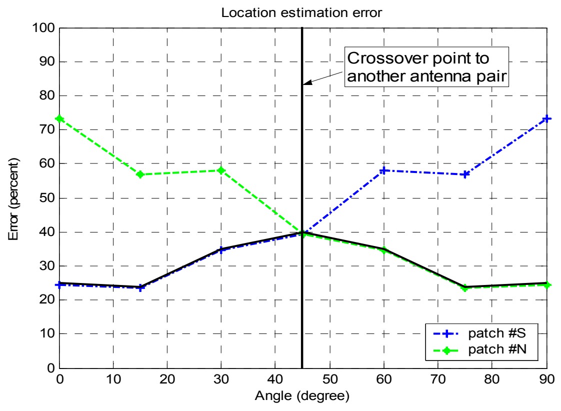


Fig. A.6. Location estimation error for Experiment 2

Compared to the experiment with omni-directional transmitting antenna, the location error is reduced when the angle is smaller than 45 degrees because the radiation pattern of directional transmission antenna overlaps with the receiving antennas. After this cross-over point, the patch #W on the anchor node which is oriented to the west can be used as the transmission antenna. Hence, the location error can be controlled to be less than 40% in a single anchor. The average error is 29.18%.

A.1.3 Experiment 3: two single patch antenna anchors and single patch antenna target

Two anchors can be used to provide additional data for location estimation, as shown in the analysis of Section 5.4. The configuration of Experiment 3 is shown in Fig. A.2(c). In this experiment only the two patches receiving the strongest signals are used while the others are turned off. Calculations for this experiment were made using equations (5.8), (5.9), and (5.10). The outdoor measurement result is shown in Fig. A.7. For this experiment, measurements are taken individually from each directional anchor node and not at the same time. This does not affect the validity of the results since the stochastic model for the channel is time invariant. When an additional anchor provides another power measurement, the estimation error reduces from 29.18% in Experiment 2 to an average of 17.78% in Experiment 3. The first cause of this improvement is the increase in the separation of the two transmission patch antennas in Experiment 3 compared to the relatively small distance of the two patches in Experiment 2. This means that the error in the radial distance R and the reception angle Θ are now uncorrelated. Second, the fading channels are more uncorrelated in Experiment 3 because the distance between the two anchors is much greater than the wavelength of the RF wave. Thus, we conclude that significant location estimation improvement can be obtained by using multiple anchors with directional antennas.

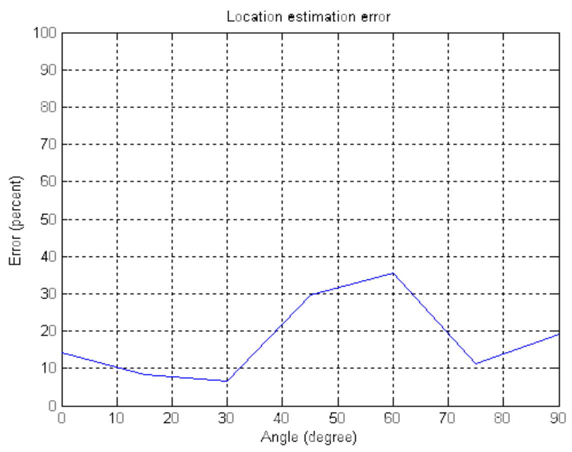


Fig. A.7 Location estimation error for two anchor nodes with directional patch antennas

perform statistical analysis on the raw measurements to quantify the errors using information from three anchor nodes. For test 1 of experiment 1 (pure omni-directional), the error is 27.5%, for experiment 2 20%, and for experiment 3 11.6%. The improvement is more striking with more neighbor anchors, as the simulation results show.