A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking

A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao[1]
M. Gouda, Y. Choi[2]
T. Herman[3]
S. Kulkarni, U. Arumugam[4]
M. Nesterenko, A. Vora, and M. Miyashita[5]

[1] Department of Computer and Information Science, The Ohio State University
{anish,duttap,bapat,vinodkri,zhangho,naik,mittalv,caohu}@cis.ohio-state.edu
[2] Department of Computer Sciences, The University of Texas at Austin
{gouda,yrchoi}@cs.utexas.edu
[3] Department of Computer Science, University of Iowa
herman@cs.uiowa.edu
[4] Department of Computer Science and Engineering, Michigan State University
{sandeep,arumugam}@cse.msu.edu
[5] Department of Computer Science, Kent State University
{mikhail,avora,mmiyashi}@cs.kent.edu

Abstract. Intrusion detection is a surveillance problem of practical import that is well suited to wireless sensor networks. In this paper, we study the application of sensor networks to the intrusion detection problem and the related problems of classifying and tracking targets. Our approach is based on a dense, distributed, wireless network of multi-modal resource-poor sensors combined into loosely coherent sensor arrays that perform *in situ* detection, estimation, compression, and exfiltration. We ground our study in the context of a security scenario called "A Line in the Sand" and accordingly define the target, system, environment, and fault models. Based on the performance requirements of the scenario and the sensing, communication, energy, and computation ability of the sensor network, we explore the design space of sensors, signal processing algorithms, communications, networking, and middleware services. We introduce the influence field, which can be estimated from a network of binary sensors, as the basis for a novel classifier. A contribution of our work is that we do not assume a reliable network; on the contrary, we quantitatively analyze the effects of network unreliability on application performance. Our work includes multiple experimental deployments of over 90 sensors nodes at MacDill Air Force Base in Tampa, Florida, as well as other field experiments of comparable scale. Based on these experiences, we identify a set of key lessons and articulate a few of the challenges facing extreme scaling to tens or hundreds of thousands of sensor nodes.

1 Introduction

Deeply embedded and densely distributed networked systems that can sense and control the environment, perform local computations, and communicate the results will allow us to interact with the physical world on space and time scales previously imagined only in science fiction. This enabling nature of sensor actuator networks has contributed to a groundswell of research on both the system issues encountered when building such networks and on the fielding of new classes of applications [1, 2]. Perhaps equally important is that the enabling nature of sensor networks provides novel approachs to existing problems, as we illustrate in this paper in the context of a well-known surveillance problem.

Background. The instrumentation of a militarized zone with distributed sensors is a decades-old idea, with implementations dating at least as far back as the Vietnam-era Igloo White program [3]. Unattended ground sensors (UGS) exist today that can detect, classify, and determine the direction of movement of intruding personnel and vehicles. The Remotely Monitored Battlefield Sensor System (REMBASS) exemplifies UGS systems in use today [3]. REMBASS exploits remotely monitored sensors, hand-emplaced along likely enemy avenues of approach. These sensors respond to seismic-acoustic energy, infrared energy, and magnetic field changes to detect enemy activities. REMBASS processes the sensor data locally and outputs detection and classification information wirelessly, either directly or through radio repeaters, to the sensor monitoring set (SMS). Messages are demodulated, decoded, displayed, and recorded to provide a time-phased record of intruder activity at the SMS.

Like Igloo White and REMBASS, most of the existing radio-based unattended ground sensor systems have limited networking ability and communicate their sensor readings or intrusion detections over relatively long and frequently uni-directional radio links to a central monitoring station, perhaps via one or more simple repeater stations. Since these systems employ long communication links, they expend precious energy during transmission, which in turn reduces their lifetime. For example, a REMBASS sensor node, once emplaced, can be unattended for only 30 days.

Recent research has demonstrated the feasibility of ad hoc aerial deployments of 1-dimensional sensor networks that can detect and track vehicles. In March 2001, researchers from the University of California at Berkeley demonstrated the deployment of a sensor network onto a road from an unmanned aerial vehicle (UAV) at Twentynine Palms, California, at the Marine Corps Air/Ground Combat Center. The network established a time-synchronized multi-hop communication network among the nodes on the ground whose job was to detect and track vehicles passing through the area over a dirt road. The vehicle tracking information was collected from the sensors using the UAV in a flyover maneuver and then relayed to an observer at the base camp.

Overview of the paper. In this work, we define, investigate, design, build, and field a dense, distributed, and 2-dimensional sensor network-based surveillance system using inexpensive sensor nodes. Such an approach relaxes the 1-dimensional constrained motion model and instead offers fine-grained detection and tracking within an area but along any arbitrary 2-dimensional path. In this model, intrusion data are processed locally at each node, shared with neighboring nodes if an anomaly is detected, and communicated to an exfiltration gateway with wide area networking capability. The motivation for this approach comes from the spatial- and temporal-locality of environmental perturbations during intrusions, suggesting a distributed approach that allows individual sensor nodes, or clusters of nodes, to perform localized processing, filtering, and triggering functions. Collaborative signal processing enables the system to simultaneously achieve better sensitivity and noise rejection, by averaging across time *and* space, than is possible with an individual node which averages only across time.

Our approach thus demonstrates how dense, resource-constrained sensor networks yield improved spatial fidelity of sampling the environment. More specifically, we introduce a spatial statistic called the influence field, realize an estimator for it using a binary sensor field, and use it as the basis for a new type of classifier. Informally, the influence field is the spatial region surrounding an object in which the object causes fluctuations in one or more of the six energy domains. In other words, the influence field is the region surrounding the object in which the object can be sensed using some specific modality. We are unaware of prior work that estimates the influence field from a set of spatially diverse samples or uses the influence field to classify an object in this manner.

Our approach complements and improves upon existing unattended battlefield ground sensors by replacing the typically expensive, hand-emplaced, sparsely-deployed, non-networked, and transmit-only sensors with integrated collaborative sensing, computing, and communicating nodes. Such an approach will enable military forces to blanket a battlefield with easily deployable and low-cost sensors, obtaining fine-grained situational awareness enabling friendly forces to see through the "fog of war" with precision previously unimaginable. A strategic assessment workshop organized by the U.S. Army Research Lab concluded:

> "It is not practical to rely on sophisticated sensors with large power supply and communication [demands]. Simple, inexpensive individual devices deployed in large numbers are likely to be the source of battlefield awareness in the future. As the number of devices in distributed sensing systems increases from hundreds to thousands and perhaps millions, the amount of attention paid to networking and to information processing must increase sharply."

Our work focuses attention on the question of whether existing sensor systems can be simply augmented with networking to realize the benefits of sensor networks. Indeed, much of the research in sensor networks is aimed at addressing key networking problems like time synchronization, node localization, and routing in the context of constrained cost and power. However, as this paper demonstrates, the simple addition of "networking" to an application may not achieve the desired level of performance. Instead, we must address *simultaneously* several additional topics including data compression, information exfiltration, and network tuning. In other words, co-design of the entire system is an essential, but often ignored, element of sensor network *system* design. This paper emphasizes co-design and demonstrates the subtle dependencies between the various subsystems. Our approach enables top-to-bottom requirements traceability.

The main contribution of our work is that it demonstrates, through a proof of concept system implementation, that it is possible to discriminate between multiple object classes using a network of binary sensors. We then demonstrate, through a proof of performance, that our implementation provides a tunable level of classification quality based on the reliability of the network. To achieve these goals, we realize the concept of an influence field. Although influence fields have been used in other contexts like tracking, we believe our work represents its first use as the basis for classification. We have demonstrated both theoretically and experimentally that the influence field provides a basis for distributed classification. We have also demonstrated the robustness of this feature in a real system, even in the presence of node failures and severe network unreliability. Each node can send out as little as one bit of information about the presence or absence of a target in its sensing range and only requires local detection and estimation, but no computationally complex time-frequency domain signal processing.

Organization of the paper. Section 2 reviews related work on detection, classification, and tracking using sensors networks. Section 3 describes the user requirements of a surveillance system, formulates more precise metrics, and establishes the target, system, environment, and fault models. This problem formulation, along with the special constraints of sensors networks, guides the exploration of the design space in Section 4. Section 5 identifies potential sensors which could be used to detect the target classes of interest, analyzes the suitability of these sensors for use in wireless sensor networks, and ties together the problem specification, design considerations, and sensing modalities to identify an appropriate sensor suite for achieving the detection and classification requirements. Section 6 considers signal detection and parameter estimation. Section 7 describes the goals of the classification system and introduces the influence field as a spatial statistic suitable for classification purposes. Section 8 discusses the goals of tracking and describes an influence field-based approach to the problem. Section 9 establishes the requirements for neighborhood- and network-wide time synchronization based on the demands of classification and tracking. Section 10 describes the communications, networking, and routing aspects of our application. Section 11 describes the system architecture, sensor network nodes, sensor boards, packaging, and other implementation details. Section 12 discusses some of the challenges and failures we encountered during the development and fielding of this system and outlines approaches to mitigate some of these problems. Finally, Section 13 summarizes our results, discusses our future plans, and provides our concluding thoughts.

2 Related Work

Detection, classification and tracking of targets is a basic surveillance or military application, and has hence received a considerable amount of attention in the literature. Recent developments in the miniaturization of sensing, computing, and communications technology have made it possible to use a plurality of sensors within a single device or sensor network node. Their low cost makes it feasible to deploy them in significant numbers across large areas and consequently, these devices have become a promising candidate for addressing the distributed detection, classification, and tracking problem. A variety of approaches have been proposed that range over a rich design space from purely centralized to purely distributed, high message complexity to high computational complexity, data fusion-based to decision fusion-based, etc. The spatial density and redundancy that is possible due to the diminishing cost of a single node favors highly distributed models. The constraints of network reliability and load permit sending only a limited amount of data over the network, and in some extreme cases, even a single bit of data. Such binary networks have been used in previous work for tracking [4–6]. Even classification schemes based on low dimensional decision fusion [7] require significant local signal processing.

In contrast to our work, much of the work on target classification in sensor networks has used a centralized approach. This typically involves pattern recognition or matching using time-frequency signatures produced by different types of targets. Caruso, et. al. [8] describe a purely centralized vehicle classification system using magnetometers based on matching magnetic signatures produced by different types of vehicles. However, this and other such approaches require significant a priori configuration and control over the environment. In [8], for instance, the vehicle has to be driven directly over the sensor for accurate classification and at random orientations and distances, the system can only detect presence. Such an approach also imposes high computational burden on individual nodes. Meesookho, et. al. [9], describe a collaborative classification scheme based on exchanging local feature vectors. The accuracy of this scheme, however, improves only as the number of collaborating sensors increases, which imposes a high load on the network. By way of contrast, Duarte et. al. [7] describes a classifier in which each sensor extracts feature vectors based on its own readings and passes them through a local pattern classifier. The sensor then transmits only the decision of the local classifier and an associated probability of accuracy to a central node that fuses all such received decisions. This scheme, while promising since it only slightly loads the network, requires significant computational resources at each node.

We also use the notion of an influence field for tracking. We successfully demonstrate the robustness of our tracking scheme in the presence of network unreliability (i.e. the accuracy of our tracking is not greatly affected even if some nodes in the influence field of the target fail or messages from some nodes are lost). The influence field also allows us to track multiple objects separated in space or time. Zhao, et. al. [10] note that the influence field can be used for tracking multiple objects separated in space, however, we believe we are the first to actually demonstrate the robustness of this approach in tracking in a network in which the unreliability is as high as 50%. The topic of distributed tracking using sensor networks has received a considerable amount of attention recently. Most of this work is based on collaborative signal and information processing, sequential Bayesian filtering, and extended Kalman filtering [10–12]. Other solutions for tracking in a sensor network are based on a or Kalman filter based approach [5, 13, 14]. These approaches attempt to estimate the future position of a target given its past and present positions. However, such estimation tends to require considerable computational resources and we are unaware of implementations that can run on the class of devices we consider for our sensor nodes [15].

3 Problem Formulation

The operational problem that this work addresses is enabling military personnel to “put tripwires anywhere.” This section specifies the user requirements of the surveillance system, formulates the system’s performance metrics, and establishes the target, system, environment, and fault models. Adlakha, et. al. [16] identified four key, sufficient, and independent user level quality-of-service (QoS) parameters appropriate for sensor networks including density, spatial-temporal accuracy, latency, and lifetime. All of these parameters are central in our work, although we do relax somewhat the lifetime parameter by not specifying a minimum system lifetime. Issues of cost and size are also important since they can directly affect the QoS parameters.

3.1 User Requirements and Performance Metrics

We consider a surveillance application scenario called “A Line in the Sand.” The objective of this scenario is to identify a breach along a perimeter or within a region. The intruding object, or target, may be an unarmed person, a soldier carrying a ferrous weapon, or a vehicle. The three fundamental user requirements of this application are target detection, classification, and tracking. The system user specifies several QoS parameters that affect how well the system detects, classifies, and tracks targets. In addition to these QoS parameters, the user defines the area or border to be protected.

Detection requires that the system discriminate between a target’s absence and presence. Successful detection requires a node to correctly estimate a target’s presence while avoiding false detections in which no targets are present. The key performance metrics for detection include the probability of correct detection, or P_D, the probability of false alarm, or P_{FA}, and the allowable latency, T_D between a target’s presence and its eventual detection.

Classification requires that the target type be identified as belonging to one of several classes including person, soldier, and vehicle. More generally, classification is the result of M-ary hypothesis testing and depends on estimation, which is the process of determining relevent parameters of the detected signal including, for example, its peak amplitude, phase, duration, power spectral density, etc. Successful classification requires that targets are labeled by the system as being members of the class to which they actually belong. The key performance metrics for classification are the probability of correctly classifying (labeling) the i-th class, $P_{C_{i,i}}$, and the probability of misclassifying the i-th class as the j-th class, or $P_{C_{i,j}}$.

Tracking involves maintaining the target's position as it evolves over time due to its motion in a region covered by the sensor network's field of view. Successful tracking requires that the system estimate a target's initial point of entry and current position with modest accuracy and within the allowable detection latency, T_D. Implicit in this requirement is the need for target localization. The tracking performance requirements dictate that tracking accuracy, or the maximum difference between a target's actual and estimated position, be both bounded and specified, within limits, by the user. The system is not required to predict the target's future position based on its past or present position.

Table 1 summarizes the overall performance required from the system. Table 2 provides a detailed set of classification performance requirements.

Metric	**Value**	**Description**
P_D	> 0.95	Probability of Detection
P_{FA}	< 0.10	Probability of False Alarm
T_D	< 10	Detection Latency (s)
$P_{C_{i,j\|i=j}}$	Table 2	Probability of Correct Classification
$P_{C_{i,j\|i\neq j}}$	Table 2	Probability of Misclassification
$(\hat{x}, \hat{y})$	$\in (x, y) \pm (2.5, 2.5)$	Position Estimation Error (m)

Table 1. Summary of the performance requirements.

	Person	**Soldier**	**Vehicle**
Person	$P_{C_{P,P}} > 90\%$	$P_{C_{P,S}} < 9\%$	$P_{C_{P,V}} < 1\%$
Soldier	$P_{C_{S,P}} < 1\%$	$P_{C_{S,S}} > 95\%$	$P_{C_{S,V}} < 4\%$
Vehicle	$P_{C_{V,P}} = 0\%$	$P_{C_{V,S}} < 1\%$	$P_{C_{V,V}} > 99\%$

Table 2. Summary of the required classification confusion matrix. Vertical labels are the true class labels and horizontal labels are the classifier labels. A person is considered a small threat; a soldier is a greater threat than an unarmed person; a vehicle is the greatest threat. Consequently, it is more important that a greater threat not be misclassified as a lesser threat than vice versa.

3.2 Target Models

This section specifies kinematic motion models of the three target classes: an unarmed person, a soldier carrying a ferrous weapon, and a vehicle. The target motion models are 2-dimensional random walks with normally and/or uniformly distributed speeds, accelerations, and bearings. The probability distributions are largely unconstrainted with the notable exceptions of bounded velocity, V_{min} which is necessary to ensure track continuity and V_{max} which is necessary to compute sampling rates, and realistically bounded accelerations. Despite the relatively unconstrained target motion models, we assume the existence of prior probabilities for each target class. For example, a normally distributed walking speed and generally constant heading are assumed for an unarmed person. A soldier, however, is equally like to crawl, walk or run, and may change directions frequently, resulting in a uniform distribution of the prior probabilities for speed, and wider tails on a normally distributed bearing model than unarmed persons. Vehicles exhibit a greater range of speeds but more constant and constrained headings than either unarmed persons or soldiers, resulting in a wider distribution and greater mean velocity, but a tighter distribution for bearing. Table 3 summarizes these constraints.

We assume that all targets actually belong to one of the specified classes. In other words, we do not consider questions of misclassifying, for example, a non-human mammal as an unarmed person. Furthermore, the track entanglement that results from the presence of multiple targets simultaneously occupying the same

Constraint	Value	Description
V_{max}	25	Maximum Velocity (kmph)
V_{min}	1	Minimum Velocity (kmph)
V_P	$\sim \mathcal{N}(5, 1)$	Person Speed (kmph)
A_P	$\sim \mathcal{U}[-1, 1]$	Person Acceleration (m/s^2)
θ_P	$\sim \mathcal{N}(0, 1)$	Person Bearing (rad)
V_S	$\sim \mathcal{U}[1, 20]$	Soldier Speed (kmph)
A_S	$\sim \mathcal{U}[-3, 3]$	Soldier Acceleration (m/s^2)
θ_S	$\sim \mathcal{N}(0, 2)$	Soldier Bearing (rad)
V_V	$\sim \mathcal{U}[1, 25]$	Vehicle Speed (kmph)
A_V	$\sim \mathcal{U}[-5, 5]$	Vehicle Acceleration (m/s^2)
θ_V	$\sim \mathcal{N}(0, 0.25)$	Vehicle Bearing (rad)

Table 3. Summary of the target motion models. The acceleration distributions are relative to the target's current speed and the bearing distributions are relative to the target's current bearing.

space and time is complex, and we are unaware of efficient distributed algorithms to disentangle these tracks, despite active research in this area [17, 14]. We also observe the impossibility of disentangling the tracks of multiple targets of the same class without additional constraint information like target motion models or individual target velocities. Consequently, we make the simplifying assumption that if multiple targets are present in the sensor network, their trajectories will not coincide in both space and time. Section 5 considers the phenomenology of the target models in the six energy domains.

3.3 System Model

The system consists of a large number of nodes distributed over an extended geographic area that is to be monitored. We do not assume careful placement of these nodes so the nodes can be deployed with some degree of randomness as they might be in a typical military deployment scenario. We do assume, however, that the nodes are deployed with generally uniform density, λ, subject to some local variations. We also assume that this density is sufficient to guarantee redundant coverage of the region to be monitored and that a localization service exists that can provide each node's relative or absolute position.

Each node in the network has a unique identifier and consists of a processing unit, memory, radio, power source, and one or more sensors of different types. The capabilities of these nodes are limited due to size, cost, and lifetime constraints. A single node has limited processing power, memory, and energy so that complex or computation intensive algorithms cannot be executed on an individual node. The communication range of these nodes is also limited such that the entire network cannot be traversed in a single hop. For purposes of exfiltration of the classification and tracking results, one or more of the nodes may be attached to a relay which can transmit these results over longer distances or over a satellite link to a remote base station.

The communication medium is wireless and broadcast is the basic communication primitive. In the wireless broadcast model, messages are subject to fading and other propagation losses. Messages from nearby nodes may collide with each other if they are sent at the same time. Even if the transmitting nodes are not in each other's communication radius, their messages could still collide at a receiver node and be lost.

3.4 Environment Model

Eventually, we expect that derivatives of our work will find themselves being deployed on real battlefields by actual military personnel. As a result, we make our environment model reality itself and accept the harshness that comes with this decision. While we cannot account for all of the environmental factors that might affect the system, we address the ones that are most likely to adversely affect the correctness and performance of our sensors, protocols, and algorithms. Cases in which we must make decisions between extensive engineering and a milder environment, we usually pick the latter and note the decision and the system's shortcomings. In the remainder of this section, we discuss weather effects, geographic variations, and noise model.

Wind can affect the sensors and cause a flurry of false positives by directly moving the sensor, indirectly through wind "noise," or by moving nearby objects like grass, bushes, and trees. Since the probability of false alarm, or P_{FA}, is an important system performance metric, we are motivated to engineer the sensors to withstand wind gusts and their attendant effects on the nearly environment. Rain can adversely affect both sensors and signal propagation. Furthermore, rain occurs frequently enough in nature that we were concerned enough with it to design waterproof containers. We allow for the possibility of snow but expect that it will melt away quickly enough that the sensors do not run out of stored energy before getting a chance to recharge using energy harvested from the sun. However, we do not actually test our system in the snow. Military specifications typically call for a wide operating temperature range spanning -40C to +85C but designing such systems is a well-understood concern of electrical and mechanical engineering, and does not contribute to the novelty of this research. As a result, it is not considered.

Both uneven terrain and the presence of obstacles can affect dramatically the quality of communications between nodes as well as the quality of sensing at a node. We make two assumptions in regards to terrain and obstacles. The first assumption is that the nodes remain sufficiently connected such that the network does not partition into multiple connected components that are disjoint from each other. The second assumption is that terrain does not affect a statistically significant number of sensors. We also note that the Earth's magnetic field constantly varies with a time-varying rate of change. This phenomenon requires that the system adapt to a changing ambient magnetic field.

The noise parameters are unknown but we assume that noise power is upper bounded. Furthermore, we assume that the noise has an unknown probability density function, is not wide sense stationary, and the noise samples are not independent and identically distributed. We choose such a mathematically intractable noise model, in contrast to a Gaussian noise model, because our early experiments indicate that environmental noise tends to have more spikes or outliers than Gaussian noise, that some of these outliers tend to be correlated, and that the noise statistics change with time.

3.5 Fault Model

Sensor networks are subject to a wide variety of faults and unreliability. Inexpensive hardware, limited resources, and extreme environmental conditions all contribute to causing these faults. In this section, we describe the types of faults that may affect the correctness and performance of our system.

Node Failures and Hardware Faults. During deployment, sensors may be dropped from high altitudes so some nodes may not survive the fall. In some cases, the sensors may become debonded from the node due to ground impact and cause intermittent or continuous false alarms or misses, resulting in seemingly Byzantine behavior. Some nodes may run out of power due to the limited onboard energy resources. Nodes may be displaced from their original positions by the targets themselves or due to environmental factors. Sensors may become desensitized due to heat or moisture and report readings that are railed high, railed low, or even arbitrary.

Communication Faults. Broadcast, the basic communication primitive in the network, leads to message losses from collisions when two nodes within range of each other transmit simultaneously. Even if the transmitting nodes are not in each other's transmission range, messages can still collide and be lost at a receiver due to the hidden terminal effect. Even in the absence of collisions, messages may still be lost as a result of fading during propagation over the wireless medium. The inter-node distance, altitude difference, antenna polarization, environmental conditions, and presence of obstacles are all factors that contribute to the fading characteristics of a wireless link.

Software Faults. The limited computational resources available on a node impose some restrictions on the amount of processing that can be successfully performed at the node. If this limit is exceeded, processing tasks may run to completion causing non-deterministic behavior and various kinds of failures. Pointers and memory locations may get corrupted, message buffers may be overwritten, and certain sensing and processing events might get lost. The node might even be forced into deadlock or livelock states from which it cannot recover on its own.

4 Design Considerations

This section considers some constraints in translating the user requirements into a functional system.

4.1 Energy

Ultimately, our systems must survive in the real world and consequently, their designs are constrained by practical matters. One such fundamental constraint is energy. Wireless sensor nodes must use either stored energy (e.g. batteries) or harvested energy (e.g. solar cells). The rate at which energy can be consumed is constrained by either the node's required lifetime for stored energy or by the average rate of energy collected through harvesting.

There are four main ways in which nodes consume energy: sensing, computing, storing, and communicating. Each of these processes consumes a different amount of energy for each unit of useful work that it performs. In order to directly compare these processes we might relate units of useful work to corresponding energy consumed as shown in Table 4.

Process	**Unit of Useful Work**	**Metric**	**Units**
Sensing	sample	E_{sens}	Joules/sample
Computing	instruction	E_{comp}	Joules/instruction
Storing	bit	W_{stor}	Watts/bit †
Communicating	bit	E_{comm}	Joules/bit ‡

Table 4. Relating units of useful work to the energy consumed. (†) Note that the metric is Watts/bit (Joule/bit/second) since a bit must be stored for a period of time. Assuming static memory allocation and the absence of random access memory that allows bit storage cells to be powered down, each bit stored contributes directly to the continuous power budget of the system. Conversely, memory comes in fixed size units so we might as well use as much memory as available. (‡) Joule/bit transmitted has been suggested as a communications unit of work but perhaps a more *useful* unit of work is message communicated, with the corresponding metric of Joule/message, since the size of the message headers can dwarf the size of the data.

Designing an acceptable system is equivalent to finding a weighted mix of these processes that minimally meets the system's requirements and ideally optimizes the system's overall performance. Recall that in Section 3, we relaxed the node lifetime requirement by not specifying an actual value. Instead, we will order different algorithms based on their complexity along these processes and choose the one that is likely to maximize a node's lifetime.

4.2 Complexity

Previously, we equated designing a good system to finding a weighted mix of the sensing, computing, storing, and communicating processes. The algorithms which perform detection, estimation, classification, tracking, time synchronization, and routing are the ones that will draw on the sensing, computing, storage, and communications subsystems. Therefore, we should focus our attention on optimizing the time, space, and message complexity of these algorithms with respect to their input parameters.

For example, we might be interested in the complexity of our signal detection algorithm as a function of sample size, n, or our tracking algorithm as a function of the number of messages, m. We also need to consider carefully our choice and method of collecting features for classification. For example, a classifier based on centralized data fusion would have a high message complexity since high dimensional data must be communicationed. Conversely, a distributed classifier may have a low message complexity, transmitting a message only when a target is detected, but a high time or space complexity due to the classification algorithm's computing or storage requirements.

4.3 Reliability

The unreliability of sensor networks has a significant impact on the system design for classification and tracking, particularly while selecting the feature that serves as the basis of classification. There are two fundamental approaches to choose from while doing feature selection - centralized and distributed. The centralized approach typically involves doing a time-frequency series analysis followed by some kind of signature matching algorithm. However, since the nodes in our system have limited computational power, performing these computation-intensive tasks would have overburdened an individual node. Also, the phenomenon we were trying to detect and classify, viz. an target moving through the field, itself was distributed both in space and time. For this reason, we concentrated our efforts on coming up with a distributed feature for the problem at hand.

Selecting the right distributed feature, however, is not an easy problem and involves several design tradeoffs. For example, the constrained resources of a single node impose restrictions on the features that can be extracted locally. Moreover, the unreliable nature of the network and the degradation of network performance under load forces us to reduce the amount of data sent out over the network. These constraints on local and network load forced us to look for a distributed feature whose projection on a single node could be efficiently calculated, whose calculation did not overload the network and yet gave us the desired accuracy of classification and tracking. Furthermore, the feature needed to be robust to network failure, i.e. it needed to still work if a few nodes failed or messages from a few nodes were lost. The resource constraints at a single node and limitations on the bandwidth and reliability of the network thus guided our feature selection into coming up with a binary network comprising of local one bit detection decisions from each node being sent over the network to a classifying and tracking module.

5 Sensing

The selection of sensors is an important task in the design of sensor networks. Choosing the right set of sensors for the job at hand can improve dramatically the system's performance, lower its cost, and improve its lifetime. However, there is a fundamental tension between the richness of a sensor's output and the resources required to process the signals it generates. For example, even small cameras have tens of thousands of pixels that provide an immense amount of information but the vision processing algorithms needed to process this vast amount of information often have high space, time, or message complexity and therefore requires significant computational resources.

In this section, we consider the sensing modes appropriate for detecting our target classes – unarmed person, armed soldier, and vehicles – based on the fluctuations they cause in the six fundamental energy domains. First, we identify the target phenomenology (i.e. the perturbations to the environment that our potential targets are likely to cause). Then, we identify a set of sensors that can detect these disturbances and discuss the difficulty of the signal processing task, using the metrics of space, time, and message complexity, required to extract meaningful information from these signals.

5.1 Phenomenology

Phenomenology is the study of the essence of things. In this section, our goal is to find a set of essential features whose values are very similar for objects in the same categories and very different for objects in different categories. We identify features in all six fundamental energy domains including optical, mechanical, thermal, electrical, magnetic, and chemical. We take such a broad view because there is considerable research underway developing MEMS sensors for each of these domains. We also note that a variety of sensors could detect different aspects of the same energy domain. For example, microphones, accelerometers, and scales all measure mechanical energy, but along acoustic, seismic, and potential dimensions.

Person. An unarmed person is likely to disrupt the environment thermally, seismically, acoustically, electrically, chemically, and optically. Human body heat is emitted as infra red energy omnidirectionally from the source. Human footsteps are impulsive signals that cause ringing at the natural frequencies of the ground. The resonant oscillations are damped and propagated through the ground. Footsteps also create impulsive acoustic signals that travel through the air at a different speed than the seismic effects of footsteps travel through the ground. A person's body can be considered a dielectric that causes a change in an ambient electric field. Humans emit a complex chemical trail that dogs can easily detect. Specialized sensors can detect certain chemical emissions, as anyone who has used public restrooms recently can attest. A person reflects and absorbs light rays and can be detected using a camera. A person also reflects and scatters optical, electromagnetic, acoustic, and ultrasonic signals.

Soldier. An armed soldier is likely to have a signature that is a superset of an unarmed person's signature. We expect a soldier to carry a gun and other equipment that contains steel or other metal. As a result, we would expect a soldier to have a magnetic signature that most unarmed people would not have. A soldier's magnetic signature is due to the disturbance in the ambient (earth's) magnetic field caused by the presence of ferro-magnetic material. We might also expect that a soldier would better reflect and scatter electromagnetic signals like radar due to the metallic content on his person.

Vehicle. A vehicle is likely to disrupt the environment thermally, seismically, acoustically, electrically, magnetically, chemically, and optically. Like humans, vehicles have a thermal signature consisting of "hotspots" like the engine region and a plume of hot exhaust. Both rolling and tracked vehicles have detectable seismic and acoustic signatures. Tracked vehicles, in particular, have highly characteristic mechanical signatures due to the rhythmic clicks and oscillations of the tracks. Vehicles contain a considerable metallic mass that affects ambient electric and magnetic fields in an area much larger than a soldier. Vehicles emit chemicals like carbon monoxide and carbon dioxide as a side effect of combustion. Vehicles also reflect, scatter, and absorb optical, electromagnetic, acoustic, and ultrasonic signals.

5.2 Sensing Options

This section reviews a subset of sensors that are well suited for *wireless* sensor networks in general and our application in particular, owing to their low power, small size, and low cost. However, some of these sensors may be unsuitable from a signal processing perspective, but those considerations have been postponed until a later section. Despite the plethora of available sensors, no primitive sensors exist that detect people, vehicles, or other potential objects of interest. For such phenomena, sensors can be used to detect various features like thermal signature or ferro-magnetic content. It can be inferred from the presence of these analogues that, with some probability, the target phenomenon exists. However, it should be clear that this estimation is an imperfect process in which multiple unrelated phenomena can cause indistinguishable sensor outputs. Additionally, all real-world signals are corrupted by noise which limits a system's effectiveness. For these reasons, in addition to sensor classification and selection, we will discuss the related topics of signal detection and parameter estimation in Section 6.

Passive sensors detect and measure various analogues of a target including its magnetic, thermal, or acoustic signature. Active sensors, such as ultrasonic and radar, can measure a target's presence, range, velocity, or direction of travel by how the target modifies, reflects, or scatters a signal transmitted by the sensor. We consider the following sensors in our selection. A more detailed analysis of these sensors can be found in [18].

Magnetic. Strengths include well defined far-field target phenomenologies, discrimination of ferrous objects, no line-of-sight requirement, passive nature. Weaknesses include poorly defined near-field target phenomenologies, limited sensing range.

Radar. Strengths include no line-of-sight requirement, ability to operate through obstacles, estimate velocity, resist jamming. Weaknesses include active nature, interference.

Thermal. Strengths include excellent sensitivity, excellent selectivity, passive nature. Weaknesses include Fresnel lens requirement, line-of-sight requirement.

Acoustic. Strengths include long sensing range, high-fidelity, no line-of-sight requirement, passive nature. Weaknesses include poorly defined target phenomenologies, moderately high sampling rates, high time and space complexity for signal processing.

Chemical. Strengths include no line-of-sight requirement, unique ability to detect gaseous compounds, passive nature. Weaknesses include lack of availability for most chemicals.

Electric. Strengths include no line-of-sight requirement, non-contact sensing of non-ferrous, fast or slow-moving, cool, quiet, odorless, steady, camouflaged objects. Weaknesses include electrode placement, nuisance parameters, active nature, interference.

Seismic. Strengths include long sensing range, no line-of-sight requirement, passive nature. Weaknesses include signal propagation variations due to ground composition, moderately high sampling rates, high time and space complexity for frequency domain analysis.

Optical. Strengths include long sensing range, high-fidelity, passive nature. Weaknesses include poorly defined target phenomenologies, line-of-sight requirement, high pixel sampling rates, high time and space complexity for signal processing.

Ultrasonic. Strengths include multi-echo processing allow sight beyond small obstacles. Weaknesses include signal propagation variations due to temperature and humidity, line-of-sight requirement, active nature, interference.

5.3 Sensor Selection

Section 3 described the user requirements of our surveillance system, formulated more precise metrics, and established the target, environment, deployment, fault, and system models. This problem formulation then drove the exploration of the design space in Section 4. Earlier parts of Section 5 identified potential sensors which could be used to detect the target classes of interest and analyzed the suitability of these sensors for use in wireless sensor networks. We now relate the problem specification, design considerations, target phenomenology, and sensing modalities to select our sensor suite. Our metrics are summarized in Table 5. We select a set of sensors that are both necessary and sufficient for the detection and classification requirements. Table 6 lists the sensors that we considered earlier along with how they compare with our metrics in Table 5.

1. Orientation Invariant: The sensor can operate regardless of its azimuthal and zenith orientations.
2. No Special Packaging: The sensor does not need to be exposed to the environment nor does it need special mechanical hardware (e.g. lenses, mirrors, etc.).
3. Reasonable Signal Processing: The algorithms required for signal detection and parameter estimation are reasonable given the constraints of the platform.
4. Established: The sensors are well characterized, commoditized, and available from multiple sources.

Table 5. Summary of the sensor selection criteria.

Sensor Type	**Orientation Invariant**	**No Special Packaging**	**Reasonable Sig. Proc.**	**Established**
Magnetic	√	√	√	√
Radar	⊘	√	√	√
Thermal	⊘	⊘	√	√
Acoustic	⊘	⊘	√	√
Chemical	√	⊘	⊘	⊘
Electric	⊘	⊘	√	⊘
Seismic	⊘	⊘	⊘	√
Optical	⊘	⊘	⊘	√
Ultrasonic	⊘	⊘	⊘	√

Table 6. Summary of the strengths and weaknesses of the sensors under consideration.

Having ranked the potential sensors according to the criteria of Table 5, we turn to the problem of identifying a sufficient set of sensors that will allow us to detect all of our target classes and discriminate among them. Magnetic sensors ranked highest in Table 6 and allow us to detect soldiers and vehicles. Since in our model, the only difference in phenomenology[6] between a person and a solder is that a soldier carries a ferrous weapon, magnetic discrimination is clearly necessary.

Even though magnetic detection is necessary, it is *not* sufficient, since a person will not be detected at all. Radar sensors ranked second highest in Table 6, and can detect all of the target classes of interest. Magnetic and radar sensors together provide sufficient information for detection of targets *and* discrimination among them. Recall that one of the differences between the soldier and vehicle phenomenologies is that a vehicle affects the magnetic field in an area much larger than a soldier does. It should follow, then, that the sensing

[6] However, we do assume some differences in the kinematic models.

range of a vehicle is greater than the sensing range of a soldier. If we can determine either the target's range from the sensor at the time of first detection, or the size of the area in which a target is detected, we can discriminate between soldiers and vehicles. We will return to the topic of discriminating between targets in Section 7.

6 Signal Detection and Parameter Estimation

The signal processing subsystem is responsible for sensing, detection, estimation, classification, and tracking. This section focuses on the detection and estimation aspects of signal processing. From a signal processing perspective, signal detection is the process of determining when a signal of interest is present and estimation is the process of determining the relevent parameters of the signal.

To bridge the notion of detecting a target's presence (as described in Section 3) with the notion of detecting a signal's presence (as described in this section), we return to our discussion of phenomenology. Specifically, we derive an analytical model for each target and sensor type, and use this model to determine the sensor output that would result if a target were present. We verify our models with empirical data from dozens of field experiments.

6.1 Magnetic

In this section, we model both a soldier and a vehicle as a magnetic dipole and provide analytical and empirical results to support our model. A moving soldier or vehicle, or more generally, a moving ferromagnetic object can be modeled as a moving magnetic dipole centered at (x_m, y_m, z_m). Still more generally, the dipole position can be described as a function of time if x_m, y_m, and z_m are replaced with $x_m(t)$, $y_m(t)$, and $z_m(t)$, respectively.

The dipole is modeled as two equal but opposite equivalent magnetic charges $+q_m$ and $-q_m$, separated by a distance $l = 2r$, where r is the radial distance from the dipole center to a charge. The orientation of the dipole is given in spherical coordinates relative to the dipole center. The zenith (polar) angle ϕ and the azimuthal angle θ, together with r, fully specify the position and orientation of the magnetic dipole, as illustrated in Figure 1. As before, a more general description of the dipole's orientation as a function of time can be given by replacing the angles ϕ and θ with time dependent versions $\phi(t)$ and $\theta(t)$ or position dependent versions $\phi(x_m, y_m, z_m)$ and $\theta(x_m, y_m, z_m)$.

A non-uniform ferromagnetic object like a vehicle will have a more complex magnetic signature composed of dipoles of varying strengths and moment arm lengths representing the engine block, axles, spare tires, roof, and other parts. These finer-granularity components can be modeled in an additive manner due to superposition effects. However, for a first order analysis sufficient for vehicle detection, a single dipole should provide an adequate estimate of the magnetic flux density $\mathbf{B}$ at the origin

$$\mathbf{B} = \frac{\mu}{4\pi} q_m \left(\frac{\mathbf{r_1}}{r_1^3} - \frac{\mathbf{r_2}}{r_2^3} \right) \tag{1}$$

where

$$\mathbf{r_1} = (x_m + r \sin\theta \cos\phi)\hat{\mathbf{x}} + (y_m + r \sin\theta \sin\phi)\hat{\mathbf{y}} + (z_m + r \cos\theta)\hat{\mathbf{z}} \tag{2}$$

and

$$\mathbf{r_2} = (x_m - r \sin\theta \cos\phi)\hat{\mathbf{x}} + (y_m - r \sin\theta \sin\phi)\hat{\mathbf{y}} + (z_m - r \cos\theta)\hat{\mathbf{z}} \tag{3}$$

The magnetic flux density $\mathbf{B}$ is composed of the following components:

$$\mathbf{B} = B_x \hat{\mathbf{x}} + B_y \hat{\mathbf{y}} + B_z \hat{\mathbf{z}} \tag{4}$$

where B_x, B_y, and B_z are the rectangular components given by:

$$B_x = \frac{\mu}{4\pi} q_m \left[x_m \left(\frac{1}{r_1^3} - \frac{1}{r_2^3} \right) + r \sin\theta \cos\phi \left(\frac{1}{r_1^3} + \frac{1}{r_2^3} \right) \right] \tag{5}$$

$$B_y = \frac{\mu}{4\pi} q_m \left[y_m \left(\frac{1}{r_1^3} - \frac{1}{r_2^3} \right) + r \sin\theta \cos\phi \left(\frac{1}{r_1^3} + \frac{1}{r_2^3} \right) \right] \tag{6}$$

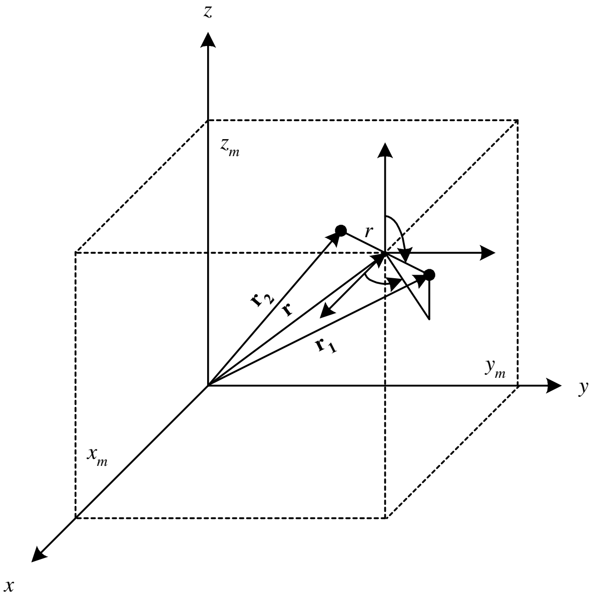

Fig. 1. Magnetic dipole model.

$$B_z = \frac{\mu}{4\pi} q_m \left[z_m \left(\frac{1}{r_1^3} - \frac{1}{r_2^3} \right) + r\cos\theta \left(\frac{1}{r_1^3} + \frac{1}{r_2^3} \right) \right] \tag{7}$$

and where r_1 and r_2 are the magnitudes of $\mathbf{r_1}$ and $\mathbf{r_2}$, respectively:

$$r_1 = \sqrt{(x_m + r\sin\theta\cos\phi)^2 + (y_m + r\sin\theta\sin\phi)^2 + (z_m + r\cos\theta)^2} \tag{8}$$

and

$$r_2 = \sqrt{(x_m - r\sin\theta\cos\phi)^2 + (y_m - r\sin\theta\sin\phi)^2 + (z_m - r\cos\theta)^2} \tag{9}$$

The component graph of the detection signal chain for magnetic targets is shown in Figure 2. The modules of this signal chain include a limiter, low pass filter (noise filtering), decimator, moving statistic, constant false alarm rate (CFAR) detector, low pass filter (hysteresis), and energy estimator. The outputs of the detector include signal duration and signal energy content. Addition modules could include a peak estimator and a hidden markov model for finer-grained signature analysis supporting classification. This signal chain operates independently on each of the two magnetometer axes and fuses the readings together at each node. Such local fusing reduces false alarms caused by spurious noise along only one axis. The design philosophy of the signal detection subsystem is that all operations occur in response to the arrival of new data samples or timer events. Sampling occurs with a certain predetermined frequency, f_s, which is computed as a function of V_{max} and the noise PSD.

The module of the signal chain perform the following functions:

Limiter. The limiter is a non-linear module that acts to limit the samples that are large in magnitude in an effort by the detector to reduce the effect of noise outliers. Without the limiter, the P_D could be reduced substantially.

Low Pass Filter. The FIR low pass filter computes a moving average of the signal to reduce noise and improve the P_{FA}.

Decimator. The decimator allows downsampling to rates more appropriate for our signal phenomenology.

Moving Statistics. The moving statistics module estimates the signal mean and variance over the previous n data samples every time a new sample becomes available.

CFAR Detector. The output of the CFAR detector is *true* during the interval in which a target passes by the sensor and *false* otherwise. This module estimates the signal duration. The module implements a Neyman-Pearson detector that works by building a histogram, which serves as a proxy for the probability

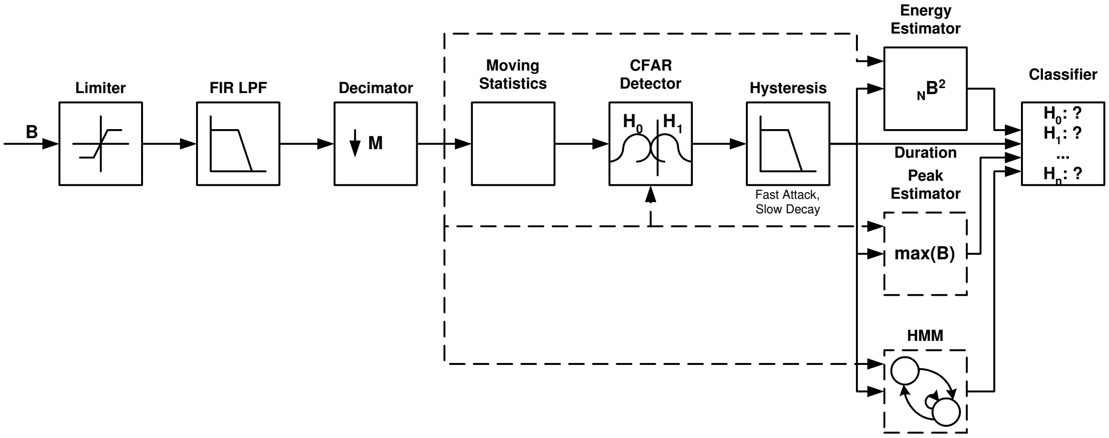

Fig. 2. Magnetometer Signal Chain.

density function, of the signal variance in the noise over a long period of time and then compares it to the (nearly) instantaneous signal variance as reported by the moving statistics.

Hysteresis. The hysteresis is implemented using an IIR low pass filter over the signal variance. The IIR filter provides a "fast-attack, slow-decay" response (i.e. a non-constant phase shift). Such a response was desired in order to avoid breaking up a single detection event into multiple smaller detections. The downside, however, is that stronger signals cause longer decay times and biases the duration estimation non-linearly before it is reported.

Energy Estimator. The energy estimator module determines the energy content of the signal of interest. The estimator begins computing the energy content of the signal when the output of the signal detector is *true* and stops when the output of the signal detector is *false*. The interval over which the energy content is computed is the duration of the signal. The energy is computed by subtracting the moving average, or bias, from the signal and then summing the squares of the samples over the period of the signal. An event is signaled upon completion of the energy content computation.

Classifier. The classifier fuses the various parameters (duration, energy, etc.) and, optionally, can attempt to categorize the target into one of several classes.

6.2 Radar

We now demonstrate that it is possible to estimate a target's distance using the radar sensor *provided the target moves with a constant velocity and heading.*

We use the TWR-ISM-002 sensor from Advantaca as our radar platform. This sensor detects motion up to a 60ft radius around the sensor but this range is adjustable to a shorter distance using an onboard potentiometer. The units sensitivity can also be adjusted in a similar fashion, depending on environmental considerations like proximity to the ground and presence of clutter.

These sensors output an analog signal that varies from 0V to 2.5V and is nominally centered at 1.25V when there is no motion. The analog output varies between 0V and 2.5V when there is motion toward or away from the radar. The output signal is a sinusoid (potentially clipped) at the Doppler frequency and is proportional to the radial component of the target's velocity. (in the direction toward or away from the radar. The clipped sinusoidal output signal encounters a zero crossing (or 1.25V bias, in this case) for every $\lambda/2$ units of distance the target travels. The radar sensor's operating frequency $f = 2.4$GHz, giving us a wavelength $\lambda = c/f = 3.0 \times 10^8 / 2.4 \times 10^9 = 12.5$ cm. Therefore, the analog output encounters a zero crossing (1.25V) for every $\lambda/2 = 6.25$ cm of travel toward or away from the sensor. The analog output returns to 1.25V when motion stops. The elapsed time t_z between successive zero crossings is inversely proportional to the target's radial velocity (the target's velocity along the sensor's line of sight) with a proportionality constant of $\lambda/2$. Therefore, the radial velocity $v_r = \lambda/2t_z$.

The radar sensors available for this study were motion sensors and not rangefinders. It is possible to use the analog output of the radar motion sensors to determine a target's range, provided we assume the following:

1. The radar sensor's field of view is a hemisphere whose projection onto the xy-plane is a circle of radius r with the sensor located at the center point O. The area enclosed by the circle is the sensitive area of the sensor and sometimes we call this area the detection circle or detection zone. The maximum detection range of the sensor is equal to the radius r of the circle.
2. The target's trajectory passes through a secant or diameter of the detection circle. If the target's trajectory is a tangent or non-intersecting line, the sensor does not detect the target's passage.
3. A single target at a time moves through the sensor's field of view at a relatively constant speed v_0 and with an arbitrary but constant heading. We can tolerate small variations in speed, as long the target's speed remains above a certain lower bound. An implication of the prior requirement is that the target does not reverse direction. Violating these assumptions leads to poor range estimation at best and completely useless range estimation at worst.
4. The initial detection time t_0 corresponds to the target's entry point into the detection circle and the final detection time t_1 and corresponds to the target's exit from the detection zone.

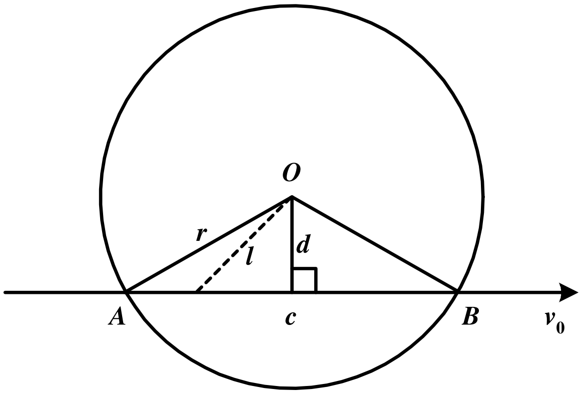

Fig. 3. Doppler ranging model.

Figure 3 depicts the above assumptions geometrically. We will refer to this model during our derivation of the range estimation equations.

Even though the radar does not directly provide a range measure to the target, it is possible to compute this information from the analog output of the MIR sensor. Let c be the length of the chord $\overline{AB}$ traversed by the target and let d be the shortest distance from the center of the detection circle O to the chord $\overline{AB}$. The chord length $c = 2 \times (r^2 - d^2)^{1/2}$. At the point where the target first enters the detection circle, the target's range l from the sensor is equal to the detection radius r. At the closest point of approach, or CPA, the target's range from the sensor is d. And, at the point of exit from sensor's field of view, the target's range is once again r.

We can determine d by recognizing that the target travels a distance of $r - d$ in the radial direction toward the center of the circle and then an equal distance away from the center, giving a total radial distance traversed of $2 \times (r - d)$. Recall that the analog output produces a zero crossing for every $\lambda/2$ units of distance the target travels in the radial direction. If we let k be the number of zero crossings counted during the interval from t_0 to t_1, then $2 \times (r - d) = k\lambda/2$. Rearranging we have $d = r - k\lambda/4$. Since r and l are constant, and k is countable with very little signal processing (i.e. simple zero crossing detection), d is easily computable ex post facto.

Now, let us compute the target's range as a function of time. We begin by determining the target's progression along the chord cp as function of time. Since the target travels with velocity v_0 and enters the circle at time t_0, the progression of the target along c is given by $c_p(t) = v_0(t - t_0); \forall t | t_0 \leq t \leq t_1$. To simplify our derivation, assume that the sensor is located at $O = (0, 0)$. Then, we can write the target's position along the x-axis as $x(t) = v_0(t - t_0) - (r^2 - d^2)^{1/2}; \forall t | t_0 \leq t \leq t_1$. This function's range is $[-c/2, c/2]$.

Let θ denote the angle enclosed by l and d. We can easily compute θ as a function of time, giving $\theta(t) = \arctan(|x(t)|/d); d \neq 0$. In the interest of symmetry and a cleaner derivation, we rewrite this equation to let $\theta(t)$ be an odd function whose range is $(-\pi/2, \pi/2) : \theta(t) = \arctan(x(t)/d)$.

It should be clear that $\forall t | t_0 \leq t \leq t_1$

$$l(t) = d/\cos\theta(t) \tag{10}$$

where

$$d = r - k\lambda/4 \tag{11}$$

$$\theta(t) = \arctan(x(t)/d) \tag{12}$$

$$x(t) = v_0(t - t_0) - c/2 \tag{13}$$

$$c = 2(r^2 - d^2)^{1/2} \tag{14}$$

$$v_0 = c/(t_1 - t_0) \tag{15}$$

Combining these equations into a function consisting of only the variables t, t_0, t_1, k, r, and λ yields

$$\hat{l}(t, t_0, t_1, k, r, \lambda) = \frac{r - k\lambda/4}{\cos\left\{\arctan\left[\frac{2\times\sqrt{r^2+(r-k\lambda/4)^2}((t-t_0)/(t_1-t_0)-1/2)}{r-k\lambda/4}\right]\right\}} \tag{16}$$

Range estimation from Doppler signals is a promising approach when targets maintain constant heading but non-constant speed, as might be the case along a fixed stretch of roadway. In our case, however, the kinematic model for both soldiers and vehicles allow for changes in speed and bearing. As a result, even though we can compute range for a constant heading target, we cannot do so robustly in the general case of arbitrary motion using the available Doppler-based radar motion sensors. Instead, we estimate the area in which a target is detected using sensors operating in a binary mode by providing presence and absence detection. We develop this estimator in Section 7 and use it as the basis for our classifier.

7 Classification

The goal of classification is to correctly label a target as belonging to one of several classes including person, soldier, and vehicle. In Section 5, we noted that vehicles contain significantly greater amounts of metallic mass than soldiers and consequently vehicles magnetically affect much larger areas than soldiers do. In general, each ferro-magnetic target class has a minimum and maximum area in which it disrupts Earth's ambient magnetic field in a manner that is detectable by our sensors. In this section, we formalize the notion of this area, called the target's "influence field," develop a distributed estimator for it, and analyze the estimator's performance for varying degrees of network reliability and latency. Finally, we address the question of discriminating a person from a soldier and vehicle through the use of sensor fusion.

7.1 The Influence Field as a Spatial Statistic

In Section 6, we informally described the need for and benefit of measuring the influence field, or the size of the area in which a target can be detected. In practice, the influence field is (the union of) the area(s) bounded by the curve(s) of equipotential field strength where the signal-to-noise ratio exceeds the sensor's minimum detectable threshold. Since the size and shape of this area could change as a function of sensor calibration and sensitivity, noise power, and other nuisance parameters like target orientation, the area is bounded by a maximum and minimum value. The influence fields of different target classes may be different from one sensing modality to the next: a vehicle will have a larger magnetic influence field than a soldier but a soldier may have a larger acoustic influence field than a vehicle if the solder is firing a gun.

The following example will serve to further illustrate this concept. Assume that nodes are deployed with a density of λ and that the nominal area of the influence field is A. Then, the number of sensors, n, that can simultaneously detect the object is given by $n = A\lambda$, or more generally by the range $[A_{min}\lambda, A_{max}\lambda]$. If target classes do not have overlapping influence field ranges, then the system can discriminate between target classes by examining the value of n. For example, assume a regular grid on 5-foot centers giving a deployment density of $\lambda = 1/5^2 = 0.04$ sensors/sqft, and a target with a minimum and maximum influence radius of 12 ft and 15 ft, respectively. Then, $A_{min} = \pi 12^2 = 452$ sqft and $A_{max} = \pi 15^2 = 707$ sqft. We find that between $n_{min} = A_{min}\lambda = 0.04 \times 452 = 18$ and $n_{max} = A_{max}\lambda = 0.04 \times 707 = 28$ sensors will detect the target's presence at a single point in time. Given the same deployment density, assume a second target with a minimum and maximum influence radius of 5 ft and 8 ft, respectively. Then, $A_{min} = \pi 5^2 = 79$ sqft and $A_{max} = \pi 8^2 = 201$ sqft. We find that between $n_{min} = A_{min}\lambda = 0.04 \times 79 = 3$ and $n_{max} = A_{max}\lambda = 0.04 \times 201 = 8$ sensors will detect the second target's presence at a single point in time.

For practical reasons, we associate with the influence field the notion of a window of time in which the target is detected. There are several factors that influence the choice of the size of this window. The number of nodes that can detect a moving target in a given interval of time may depend upon the size of the object, the amount of metallic content and hence the range at which it can be detected by a magnetometer, the velocity of the target, and the number of sensors in the region around the target. Therefore, we must consider the density of node deployment and the size and speed of the target types. Referring back to the target motion models in Section 3, we identify the smallest and the slowest moving ones as well as the largest and fastest moving ones. The amount of time required to process the data for a given window must be less than the window duration in order to meet the needs of a real-time online system. We are also concerned with the concurrent detection of the same event at different sensors because of differences in the hardware, the sensitivity of sensors, or the parameters of the detection algorithm running at the sensor node. For instance, a fast-attack, slow-decay detector, like the one used in our signal detection software, can affect sensors in a non-linear and non-deterministic manner, causing perceived time differences between the starts and ends of detections at different nodes for the same event. This uncertainty in detection duration, which can be as large as 500ms, also affects the size of the influence field window. Finally, we also have to factor in the effect of network unreliability on the number of messages corresponding to the detection events that are actually received at the classifier. Based on these factors, we selected 500ms as the width of the influence field window.

Given classification's required window size, we must verify that detection events occurring at the same physical time are timestamped accordingly with values that will fall within the same window of global network time at the classifier. In order to achieve this common timebase, a time synchronization service is needed that will maintain an estimate of global time at each node. This service also needs to guarantee that the maximum difference in the estimates of any two nodes in the network does not exceed some fraction of the classifier window size. For instance, for a window size of 500 ms and a classification accuracy of exceeding 99%, we require that the accuracy of the time synchronization service to be within 1% of 500ms or 5ms.

7.2 Classifier Design

The classifier collects data received from the network and partitions it into windows of global time. Once the incoming data has been partitioned into windows based on global time, the classifier counts the number of nodes that have detected the presence of an target in that window. To conserve the network bandwidth, nodes simply report the start and the end of a detection event. Hence, the classifier has to maintain a history of nodes that have started detecting an event but have not yet stopped detecting it. The classifier carries forward the count of such *active* nodes from one window to the next. Further, all detection events in a classification window need not belong to the same target. For instance, if multiple targets simultaneously in the network, each target will be detected by the nodes in the region surrounding it. The classifier distinguishes multiple targets and does not combine these simultaneous detections into a single larger target.

Wind and other sources of noise can cause nodes to report false detections. The classifier identifies such outliers that could skew the classification. Consequently, the classifier uses localization information about the reporting nodes and knowledge of the target motion and phenomenological models. For example, if the classifier receives only two detection events and the influence field for the smallest target type, the soldier, is expected to be between 4 and 9 for the given density considering 50% network reliability, the classifier identifies these nodes as outliers and does not generate a classification. If, on the other hand, the influence field for a soldier is 9 while that for a car is 36, and if 4 soldiers walk through the network at the same time such that they are at sufficient distance from one another, the classifier identifies that the corresponding events belong to different targets and accurately classifies the targets as 4 soldiers rather than a single car. Note that the data association problem is automatically addressed because of the fine-grained spatial locality of the detections. The output of the classifier at the end of each classification window is one or more classification decisions along with the supporting evidence, in the form of a set or sets of nodes, that are associated with the given target.

Classifier latency is an important tunable parameter that governs the length of time that the classifier waits between receiving the first detection event and providing the first classification result. The classifier masks detection events until it has received a sufficient number of samples to achieve the desired probability of false alarm, P_{FA}. In addition, the classifier introduces a delay while waiting for enough samples to report a meaningful classification. Consequently, the classifier latency is only one of two components that contribute to the overall system latency. The other component of system latency is the latency between detecting a target at the sensor node and reporting that detection to the classifier. We investigate the classifier performance as a function of latency in the next section.

7.3 Validation

A theoretical model of a target's influence field, parameterized with the target's size, speed, heading, inclination, location, and ferro-magnetic content can be used to demonstrate that the probability density functions of the various target classes are discriminable. However, an accurate model of a target's influence field may be difficult to achieve without incorporating many nuisance parameters and even still may require time consuming finite element methods to compute. Due to the spatial, temporal, and class-conditional variations of the numerous nuisance parameters, we present a simple lumped parameter model of the computed strength and shape of the influence field for both a soldier and a vehicle in Figure 4. The soldier is modeled as carrying a gun of length 3 ft at an azimuthal angle of 20°. The vehicle is modeled as the superposition of the influence fields of the engine, front axle, rear axle, transmission, spare tire, and steering wheel. In both cases, the positive y-axis is pointing toward the northwest and the field strength displayed is the planar projection of the magnetic field at ground level across the displayed area. These models should convey a sense of the relative shape of the influence fields and an intuitive understanding of their discriminability.[7] Since we do not know the true weights and variations of the parameters, we will use empirical methods to validate the model.

As with any real system, experimental validation of performance is necessary. In the case of the influence field as an estimator of the target's class, this validation exists at three levels: the theoretical influence field, the influence field as measured by the sensor nodes, and the influence field as reported to the classfier. Due to the complexity of the theoretical model, the remainder of this section will focus on empirical measurements of the influence field at the sensor nodes and its estimate at the classifer. The key distinction is that the estimated influence field is a noisy function of the measured influence field, network reliability, latency, probability of detection and false alarm, detector hysteresis, and nuisance parameters described earlier. How much "noise" is added and in what quantities as a result of these additional parameters in reality is not clear. Consequently, we vary a few easily controllable parameters such as speed, heading, network reliability and latency, and lump the remaining parameters.

Figure 5 shows the influence field probability distributions for a soldier and a car, as actually measured by the sensors. The measured influence field of a soldier has a mean, μ, of 12.2 and a variance, σ^2, of 0.44 while a car has a mean of 43.5 and a variance of 0.49. We overlay on this figure a pair of curves $\sim \mathcal{N}(12.1, 1)$ and $\sim \mathcal{N}(43.5, 1.5)$, approximating the influence field of a soldier and a vehicle, respectively. It should be clear from Figure 5, that there exists a clear separation between the measured influence fields for a soldier and a car. Since the distributions have nearly identical variances, we can compute the discriminability, d', as

$$d' = \frac{|\mu_2 - \mu 1|}{\sigma} = 67 \tag{17}$$

which indicates a practically infinitesmal probability of mis-classification.

[7] Note: the field strength is not scaled the same for both plots. Hence, these plots demonstrate the shape but not the relative size of the influence fields.

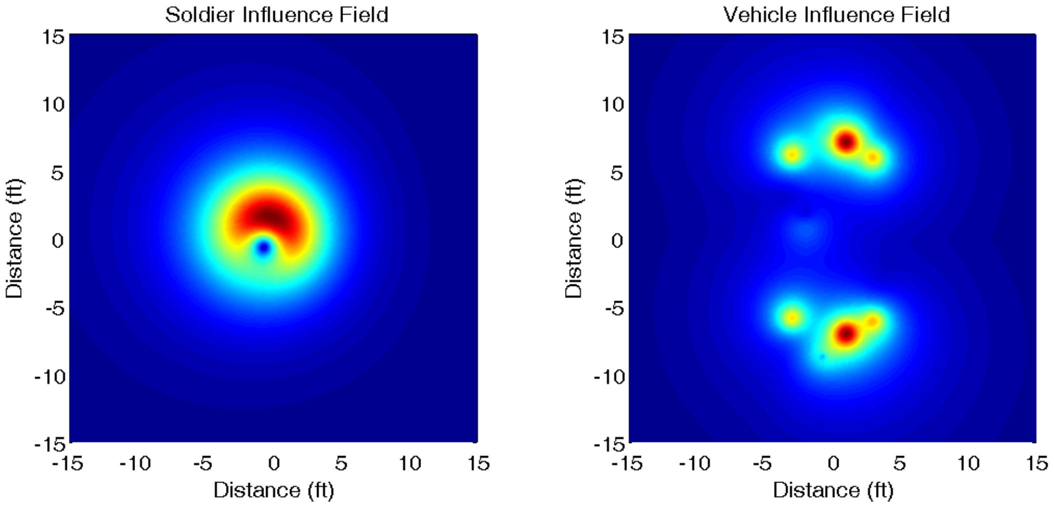

Fig. 4. The shape of the influence field of a soldier vs. a vehicle.

We note that these influence field values are ideal in the sense that they are based on data obtained directly at the nodes and not necessarily the data that is actually available to the classifier. In order to evaluate system performance, we study the effects of network unreliability and latency on the classifier performance by varying the transmission power level (3, 6, 9, and 12)[8], the number of transmissions per message (1, 2, 3, 4, 5), and the classifier latency (5, 10, and 15 seconds). In all, we run a total of 280 experiments using 16 different parameter configurations (a subset of the possible parameter space). Analyzing the timestamped measurements at the nodes and at the classifier, we identify in Figure 6 the parameter values which best demonstrate the variability in the estimator performance as a function of reliability and latency. We use the notation MAC(P,T,L), where P is the power setting, T is the total number of transmissions, and L is the latency in seconds.

The confusion matrix for MAC(9,1,5) is shown in Table 7. The classifier performance is clearly unacceptable with this optimistic single transmission per node and a classifier latency of 5 seconds.

We find that increasing the latency to 10 seconds, as shown in MAC(9,1,10), improves the classifier performance to 100%. Recall, however, that we used a lumped parameter approach that did not attempt to characterize the influence field over the entire range of all possible parameters. Consequently, the probability distributions of the influence fields would likely have greater variance than our test cases indicate and would likely overlap, reducing the discriminability to an unacceptable level. Since increasing the latency from 5 to 10 seconds improves the classifier performance, we are motivated to further increase latency, to 15 seconds, as

[8] See Section 11 for a description of the hardware used

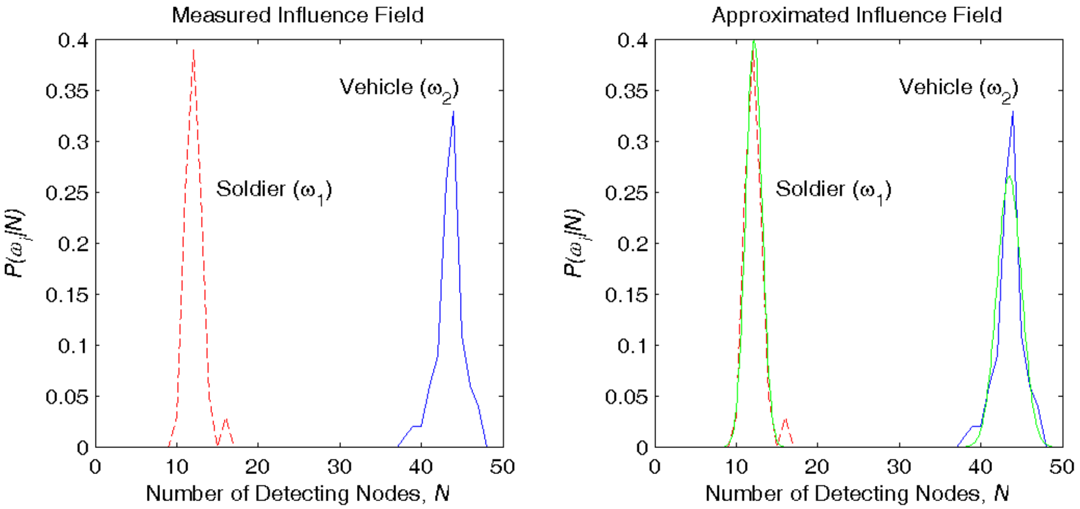

Fig. 5. The influence field of a soldier and a car as measured at the sensor nodes, and their Gaussian approximations.

shown in MAC(9,1,15). We find, however, that no further improvement in discriminability results from this increase in latency. We still remain interested in improving the classifier performance, so we must consider other parameters.

Turning our attention to reliability, we attempt to transmit each message at most three times (i.e. two retransmissions) and with a latency of 5 seconds. The results are shown in MAC(9,3,5). We find that our attempt to increase network reliability has actually decreased the discriminability of the target classes since the probability distributions now overlap even more so than with MAC(9,1,5). We attribute this poor performance to the increased traffic generated from retransmissions and its attendant effects on collision and congestion. Next, we try MAC(9,3,10) and notice an improvement over MAC(9,3,5) in that the discriminability has improved. We also note the MAC(9,3,10) results in a distribution whose variance is smaller than MAC(9,1,10). Encouraged by the positive trends in both separating the means and decreasing the variances, we investigate MAC(9,3,15) and find that it offers the best discriminability.

	Soldier	**Vehicle**
Soldier	$P_{C_{S,S}} = 31\%$	$P_{C_{S,V}} = 69\%$
Vehicle	$P_{C_{V,S}} < 1\%$	$P_{C_{V,V}} > 99\%$

Table 7. The confusion matrix for MAC(9,1,5) assuming the $P_{C_{V,V}} > 99\%$ requirement is met.

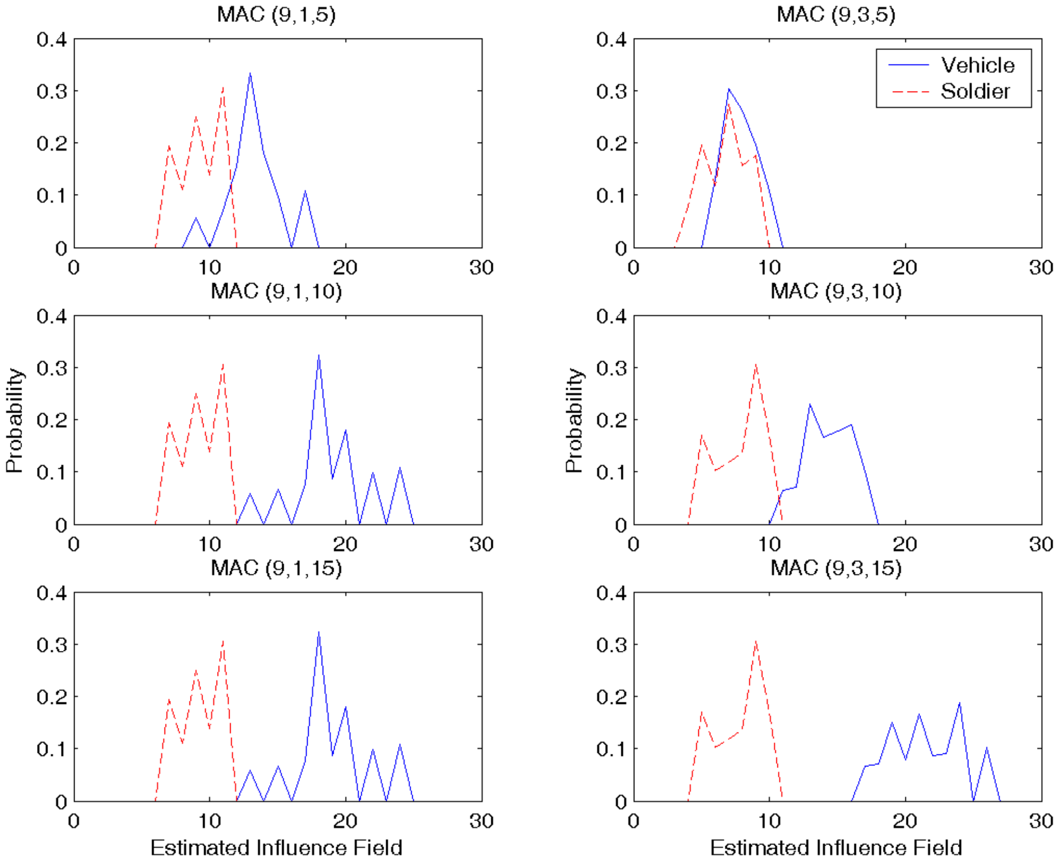

Fig. 6. Probability distribution of the estimated influence field as a function of media access control (MAC) power, transmissions, and latency. MAC(P,T,L), where P is the power setting, T is the total number of transmissions, and L is the latency in seconds.

We notice that while MAC(9,3,15) offers the best overall performance, the worst case network reliability is approximately 50%, since only one-half of the detections at the nodes are actually available to the classifier. We also note that the network loss rate varies according to target class. There is less variation in the soldier's influence field than in the vehicle's, as estimated at the classifier across all of our tests. The soldier's estimated influence field ranges from 9 in MAC(9,1,5) to 6.4 in MAC(9,3,5), a decrease of 27% and 48%, respectively, from the measured value of 12.2. In contrast, the vehicle's estimated influence field ranges from 21.2 in MAC(9,3,15) to 7.7 in MAC(9,3,5), a decrease of 51% and 82%, respectively, from the measured value of 43.5. In other words, the network delivers approximately twice the reliability for a soldier than it does for a vehicle. We attribute this wide variation to the different levels of traffic that are generated isochronously for the different target classes. Assuming a 10% margin of error in the influence field estimation and greater than 99% classification accuracy, we find that the acceptable lower bound on network reliability for classification is approximately 50%.

We have focused our attention on discriminating soldiers from vehicles. We now turn to sensor fusion in order to address the classification of persons, and to make our detection of soldiers and vehicles more robust. Recall that the radar motion sensors detect all of the target classes but only the ferro-magnetic targets are detected by the magnetic sensors. Consequently, we can fuse these two sensing modalities into a single predicate that the classifier evaluates at the end of each window to determine whether a person, and only a person, is present

$$person = radar \wedge \neg magnetic \tag{18}$$

We also use sensor fusion to make the detection of soldiers and vehicles more robust

$$soldier = radar \wedge magnetic \wedge (\hat{f} < x^*) \tag{19}$$

$$vehicle = radar \wedge magnetic \wedge (\hat{f} \geq x^*) \tag{20}$$

where $\hat{f}$ is the estimated influence field and x^* is the decision boundary that minimizes the classification error rate. In our case, $x^* = 14$.

8 Tracking

Recall our tracking requirement is an accuracy of less than 5m for soldiers, unarmed persons and vehicles moving through the network. Moreover, we need to track multiple simultaneous targets moving through the network assuming they are separated by a sufficient distance, say 10m.

Algorithm. We use the same influence field concept as used for classification for tracking intruders moving through the network. The classifier calculates the influence field of an intruder or multiple intruders moving through the network. It also performs the task of separating false alarms from real intruders and distinguishing multiple intruders moving through the network at different locations, using localization information. The classifier then passes to the tracking module in each window of time, a set or sets of nodes constituting each classified intruder and the intruder type. When the tracking module receives such a classifier output for the first time, it tags it as a new intruder. It then estimates the most likely location for each intruder as the centroid of the convex region enveloping all the nodes detecting it. Depending on the type of intruder and the current estimated position, the tracking module also computes an expected region for the intruder to be located in the next window, based on the velocity of the intruder type. It then correlates the tracked objects from successive windows in order to construct a continuous track per intruder for the entire time it spends in the network. If the estimated location of an intruder does not lie in the expected regions of any of the currently tracked intruders, a new intruder is detected and a new track is created.

Implementation considerations. The accuracy of tracking the intruder location using a centroid approach is fairly good in case of a perfectly reliable network. However, due to network unreliability, the set of nodes received by the tracking module is only a non-deterministic subset of the nodes that actually detect the intruder. Hence, the tracking module has to first perform some kind of a smoothing or filling algorithm to reduce the effects of network noise. To achieve this, we use a heuristic called the *Bounding Box*, which is defined as the tightest convex region that can be fit around the set of nodes detecting the intruder whose messages are received. The soundness of the bounding box method lies in the fact that the classifier performs detection of false positives and outliers, hence the input to the tracking module consists of legitimate intruder detections. The tracking algorithm also uses information from the intruder model about the spacing between multiple intruders moving concurrently through the network in order to accurately compute bounding boxes for multiple intruders and not confuse them as part of a single large intruder. The tracking module now estimates the location of each intruder as the centroid of its computed bounding box.

Results. Based on the influence field approach and using the bounding box heuristic, we were able to meet the prescribed accuracy requirements. The system is able to track soldiers and unarmed persons with an accuracy of 1-2m and vehicles with an accuracy of 3-4m. The system can also accurately track multiple soldiers or vehicles separated by a distance greater than 10m moving simultaneously through the network.

9 Time Synchronization

As described in the derivation for the influence field and the size of the classifier window, we require that observations occurring at the same physical time need to be timestamped with values differing by not more than 5 ms in order to achieve the desired classification accuracy. This imposes an accuracy requirement on the time synchronization service, i.e., the time values of any two nodes in the network cannot differ by more than 5 ms at any time instant, which translates to a per-hop accuracy of less than 500 μs.

An important requirement of the time synchronization service is that it should be robust to node and link failures. Our first implementation of time synchronization was based on the traditional spanning tree based algorithm for synchronizing clocks. In this algorithm, a distinguished node periodically sends time values along a spanning tree structure. However, this algorithm, while providing good accuracy, suffers from the drawback of depending on a relatively static structure for time dissemination and hence performs poorly in the presence of unreliable nodes and links and network partitions. For this reason, we designed a truly distributed time synchronization service that is robust to these commonly occurring faults in sensor networks.

9.1 Algorithm

The basic idea behind our time synchronization algorithm is that of locally synchronizing to the fastest clock. In this scheme, each node maintains a local clock which is never changed and an estimate of network time which is stored as an offset with respect to the local clock. Each node broadcasts its network time value periodically and receives time values from its neighbors. If a node receives a time value greater than its own, it adjusts it local time to that value, otherwise it ignores the value received. Thus, the entire network synchronizes to the maximum clock value in the network. This scheme also guarantees that the timestamp values at every node are monotonically increasing.

This time synchronization algorithm is purely distributed, local, and does not depend on a fixed structure for disseminating time values. Consequently, even if a single node or link fails, the other nodes can receive time values from the rest of their neighbors. The protocol is thus robust to individual node and link failures. Node joins are also easy to handle as the new nodes have lower time values and hence catch up with the rest of the network and do not force other nodes to roll back their clocks. The algorithm can also handle network partitions in the sense that the nodes in each partition synchronize to the maximum time value in that partition. These properties of robustness to network dynamics like failures, joins, and partitions make this protocol suitable for mobile environments as well. We demonstrated the robustness of the time synchronization scheme in a mobile setting by partitioning a group of sensor nodes into two subgroups, each of which synchronized to the maximum value in its partition. We showed that whenever a node moved from one partition to another, it would either catch up with the time in the new partition or the new partition would catch up with the moving node's time and consequently the original partition. In the presence of node mobility in both directions, the protocol converges to synchronize both partitions with high accuracy even if no two nodes in either partition can communicate with each other directly.

9.2 Accuracy

The accuracy provided by the time synchronization service depends on the node and network resources available to the service. On the network side, accuracy increases as the frequency of the periodic message exchange increases. On the node side, accuracy can be improved using skew calculations. In the basic version of the protocol, whenever a node receives a higher time value than its own, it copies the received value. However, this is inaccurate because the time value at the sender of that time has already moved forward by the time the value is copied. The amount by which its clock has moved forward is the sum of the time taken by the node to send out the message after timestamping it, the message transmission time and the time spent in receiving the message, comparing its time value and copying it. To complicate matters, this elapsed time may be non-deterministic in case of random waiting or backoff strategies for channel access. The accuracy of the time synchronization service can be greatly improved by estimating this elapsed time. This calculation requires the sending of additional information in each message and additional processing at each node. In order to avoid errors due to non-deterministic delays, the stamping and copying of time values can be done at the MAC level just before the message is transmitted and just after it is received.

9.3 Results

Our experiments demonstrate that the basic time synchronization algorithm meets the level of accuracy desired by the application, as can be seen from Table 8.

Our results also show that accuracy improves significantly when time synchronization is implemented as close to the hardware level as possible. Moreover, accuracy can be further improved using skew compensation techniques.

Time synchronization scheme	Per-hop accuracy
Basic Algorithm (local maxima based)	305.1 μs
Basic Algorithm + low-level processing	19.25 μs
Basic Algorithm + low-level processing + skew compensation	10.9 μs

Table 8. Accuracy of the time synchronization protocol.

It should be noted that in large scale networks, where the amount of message traffic received is high, processing time synchronization values at the level closest to hardware can be risky. Because of the overhead of extra computations at the lowest level, processing of time synchronization messages might be preempted by other low level events resulting in arbitrary state corruptions if not programmed or scheduled carefully.

10 Communications and Networkworking

Section 7 demonstrated the importance of network reliability. In this section, we consider both routing and reliable communications between neighboring nodes.

10.1 Routing Algorithm

The limited transmission range of sensor nodes implies that all nodes are not within direct transmission range from the classifier. We thus need a routing service to direct messages from any node in the network to the classifier. The dynamic nature of the wireless communication medium often results in the formation of unstable long links in the network [19, 20]. These unstable links, if part of the routing structure, result in frequent route changes. One approach to deal with such links involves dynamic link estimation to determine the set of stable links [20]. We opt for a simpler routing protocol that is both reliable and lightweight in terms of communication, computation and memory requirements. By tuning the transmission power level of the sensor nodes, we find that it is possible to guarantee that links in the routing structure will be stable. For these reasons, we devised a simpler routing protocol than the more traditional distance vector protocol available. We call this routing protocol the Logical Grid Routing Protocol or LGRP.

LGRP uses the localization or neighborhood detection service to determine a set of reliable nearby neighbors. These nodes are called the logical neighbors. A node can determine which of its logical neighbors are closer to the root than it is. Neighbors closer to the root are called a node's low neighbors, and neighbors farther from the root are called a node's high neighbors. A node limits its choices for selecting a parent in the routing structure to these logical neighbors. This simple scheme for neighbor selection helps avoid the formation of unstable long links in the routing structure. The root node periodically transmits beacons in order to construct and maintain a routing tree. The logical neighbors of the root receive these beacons and set the root to be their parent. These one-hop neighbors of the root then start propagating their own beacons. Upon receiving a beacon from a logical neighbor, a node selects it as its parent if it is a low neighbor. In case a node does not receive any beacons from low neighbors, a node selects a high neighbor as its parent. However, since such inverse links can lead to loops in the routing structure, the protocol limits the maximum number of inverse links that a path from a node to the root can contain.

LGRP is self-stabilizing and can tolerate node failstops. If a node misses a certain threshold of beacons from its parent, it suspects that its parent is no longer available and tries to select a new parent according to the protocol. The protocol is simple and lightweight, and does not tax the resources of the node or the network by exchanging frequent messages and performing dynamic estimations. Finally, LGRP provides a reliable and stable routing structure in a wireless sensor network, and hence is suitable for the requirements of our system.

10.2 Reliable Communications

The influence field feature and its validation mandate that the network has to provide a worst case end-to-end reliability of about 50% under the traffic generated by the different target types to satisfy the desired classification goals. However, messages sent over the wireless network is subject to loss as a result of collisions and fading of the radio signals during propagation through the medium. These sources of message loss adversely affect the end-to-end reliability of the network and we observed an effective reliability of less than 50%, resulting in poor performance of our classifier. Hence, we designed a reliable communications service, called ReliableComm, whose goal was to improve per-hop and end-to-end reliability in the presence of fading, collisions, and congestion.

ReliableComm makes extensive use of implicit acknowledgements. The implicit acknowledgement scheme is known to be well-suited to wireless sensor networks because of the broadcast nature of the medium. Whenever a node forwards a message to its parent on behalf of its child, this forwarded message is also received by the child, thereby serving as an acknowledgement for the child's message to the node. Every node in the network maintains a queue of messages. A node transmits the message at the head of its queue and starts a timer which fires after a timeout interval. Any messages received after this period are enqueued for later transmission. If the node receives an implicit acknowledgement from its parent before this timer expires, it knows that its parent received the message and hence removes the corresponding message from its own buffer. If the acknowledgement is not received before the timeout, the node assumes that the parent did not receive the message, retransmits it and starts a new timer and waits for an acknowledgement. This process is repeated until either the node receives an implicit acknowledgement from its parent or the node has attempted a certain threshold number of retransmissions, which is a tunable parameter of this service. At the last hop, the root does not have to forward the message and so it explicitly acknowledges receipt of the messages on the last hop.

ReliableComm provides an efficient implementation of the implicit acknowledgement idea and hence meets the requirement of being lightweight. The concept of logical neighbors in LGRP bounds the number of children that a node can have in the routing structure. Consequently, the size of the queue maintained by a node in ReliableComm is also bounded, making it efficient in memory utilization. It is possible in the implicit acknowledgement scheme that despite having correctly received a message from a child, the parent has not yet forwarded the message due to a other messages in the queue. In order to avoid unnecessary retransmissions in the case of successful reception, ReliableComm uses an optimization by which a node cumulatively acknowledges all the messages in its queue with each outgoing message. Here again, the bounded size of the node queue ensures that all messages in the queue can be acknowledged in each outgoing message. In the event that a node misses this implicit acknowledgement and retransmits the message, ReliableComm maintains a sequence number of the last message received from each child, thereby avoiding unnecessary requeueing and retransmission of duplicate messages. Once again, the bounded number of children bounds the memory required at each node. ReliableComm thus provides an efficient implicit acknowledgement scheme that exploits the broadcast nature of communication to piggyback the acknowledgement on the forwarded message, thereby conserving network bandwidth.

The level of reliability provided by the ReliableComm service depends on several tunable parameters. The overall end-to-end delivery increases in general, as the number of retransmissions is increased, although in the worst case the reception latency also increases when the network is congested simply because messages may arrive at the base station after being retransmitted an increasing number of times for each hop. The transmission power level is also a key factor affecting the reliability of the network. Clearly, nodes have to transmit messages with enough power so that these messages will reach a node's parent with high probability despite the effects of fading in the channel. However, our experiments show that the interference range of a node is almost two or more times its high-probability transmission range. Hence, transmitting at a power level higher than necessary may lead to increased collisions in the network, thereby reducing network reliability.

10.3 Experimental Results

As with the influence field feature, we performed several experiments in order to validate that the instrumentation of end-to-end reliability met the requirements imposed by the desired classification accuracy. The experimental setup was similar to the one used for the influence field calculations, except that each node upon detecting an intruder in its range, now also sent a start and a stop message indicating the start and end of the detection event, over the network to the classifier connected to the root of the routing structure. This message consisted of a single start or stop bit along with a timestamp and source information. Because of the network unreliability, the classifier received data from only a subset of the transmitting nodes.

Moreover, this data was out of order and possibly duplicated because of retransmissions at the last hop in the network. In order to negate the effects of message reordering and accurately calculate the influence field for an intruder, we introduced a delay in processing at the classifier. This processing latency ensures that the classifier receives the desired number of messages over a moving window before processing the messages that originated in the network during that window. In our most detailed experiment, we performed 280 trials consisting of 16 different combinations of parameters including transmission power level, maximum number of retransmissions, retransmission timeout, etc. The key results of these experiments are described in Section 7, supporting the notion of co-design.

Figure 5 with MAC(9,1,x) shows the experimentally measured probability distributions of the influence fields for a soldier and a car as aggregated at the classifier. Figure 5 with MAC(9,1,10) is the best distribution obtained without using ReliableComm while Figure 5 with MAC(9,3,15) is the distribution under the best network reliability obtained using ReliableComm with the number of maximum retransmissions at each hop being set to 2. The transmission power level in both these cases was the same[9] and it yielded the best performance for the inter-node spacing in our experiments.

We were able to meet the desired goals of classification accuracy only by instrumenting reliability in the network. However, this accuracy came at the cost of increased latency overhead of 13 seconds as a result of using ReliableComm, causing the system to perform with a latency that did not meet the original specification. The increased latency caused much consternation among observers but given the choice between latency and classification error, increased latency was seen as less troublesome. These results demonstrate a tradeoff between speed and accuracy that is central to system co-design.

11 Implementation

During the spring, summer, and fall of 2003, we repeatedly deployed a 90 node sensor network at three different sites in Ohio and Florida to test the entire system described in this paper. We also deployed dozens of networks consisting of a smaller number of nodes to test various subsystems including detection, classification, tracking, time synchronization, and routing at several additional sites in Ohio, Michigan, Iowa, and Texas. This section examines the system architecture, sensor node hardware, packaging, and visualization used in some of these experiments.

11.1 System Architecture

Our experiments are based on a network of 78 magnetic sensor nodes arranged in a 60 ft by 25 ft layout as shown in Figure 7. Overlayed on this network is 12 additional radar sensor nodes co-located at nodes 2, 3, 6, 7, 28, 29, 34, 35, 67, 68, 76, and 77. The magnetic sensor nodes are distributed uniformly in this region except for along two “trails” where the nodes are “pushed aside” to the edges of the trails. The trails are wide enough to allow a vehicle to drive through without running over the nodes. The nodes in the network are numbered 0 to 77 and the network is connected to a remote computer via a base station (node 0) and a long-haul radio repeater.

11.2 Network Nodes

The Mica2 mote, a derivative of the Mica family of motes developed at U.C. Berkeley [15], served as our network node. The Mica2 offers a 4MHz Atmel processor with 4KB of random access memory, 128KB of FLASH program memory, and 512KB of EEPROM memory (for logging) . The motes run the TinyOS operating system [21], and are programmed using the NesC language [22]. The Mica2 mote is shown in Figure 8.

11.3 Sensor Boards

Sensing of magnetic fields was accomplished using the Mica Sensor Board that is a part of the Mica family. We also integrated the TWR-ISM-002 radar motion sensor from Advantaca to detect moving persons, soldiers,

[9] A power level setting of “9” on our sensor network nodes

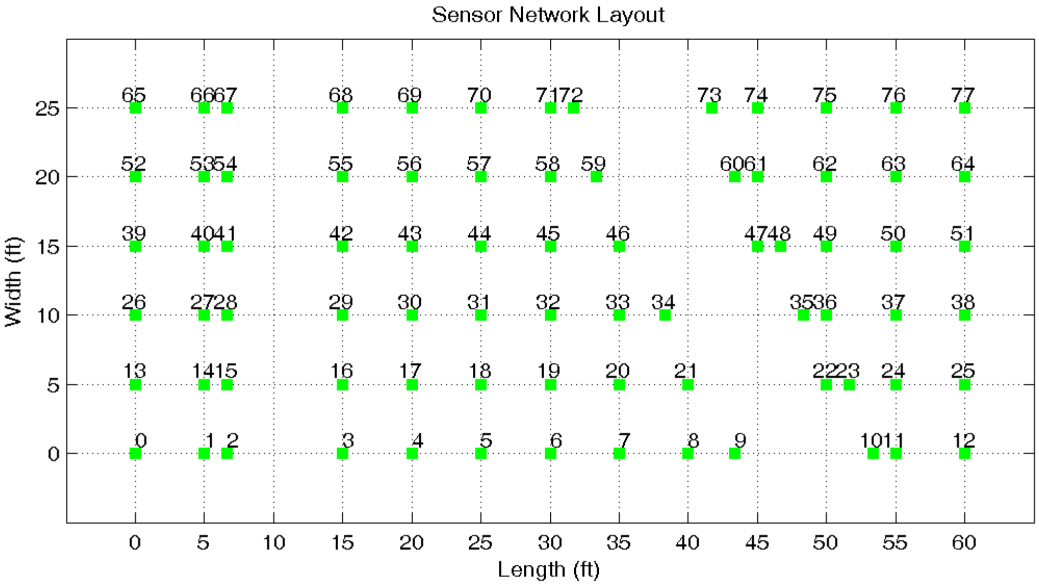

Fig. 7. Sensor Network Layout.

and vehicles. In order to integrate the radar sensors with the Mica2 motes, we developed the Mica Power Board [23] which contains a pair of boost regulators and is capable of powering the radar board. The Mica Sensor Board, Mica Power Board, and Radar Board is shown in Figure 8.

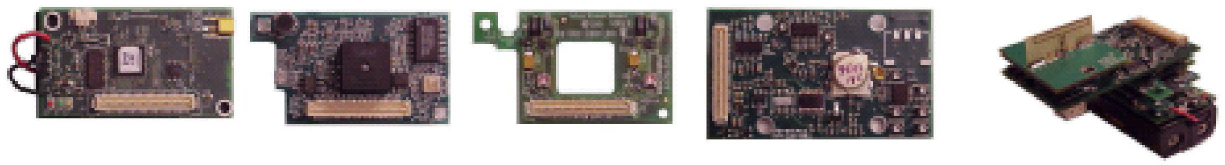

Fig. 8. Sensor hardware from left to right: (a) Mica2 network node, (b) Mica Sensor Board, (c) Mica Power Board, (d) TWR-ISM-002 Radar Board, and (e) All of the boards attached together.

11.4 Packaging

The environment in which the sensors operate was described in Section 3. The sensor enclosure was designed to minimize the likelihood of environmental effects of wind, rain, snow, and terrain. For example, the final enclosure design is a smooth cylindrical capsule that minimizes wind resistance. The enclosure is manufactured from a clear acrylic or Lexan$^{(R)}$ material to allow sunlight to pass through and illuminate a solar cell. The end caps are screwed down at four points and could be augmented with O-rings to ensure a water-tight seal. The electronics are mounted on a single gimbal mechanism which, when coupled with the rotation degree of freedom of the cylindrical enclosure, significantly increases the likelihood that the electronics will be co-planar with the ground. The enclosure design is shown in Figure 9.

11.5 Visualization

Figure 10 shows the software for visualizing the sensor nodes and targets. This software includes support for zooming in and out, replaying recent history, viewing the network topology, and displaying estimated target tracks.

12 Key Lessons

As shown in earlier sections, network unreliabilty, resource constraints, and network size yield nontrivial challenges in the design, implementation, testing, and analysis of the application and the network services. Several of these challenges are not readily accommodated by traditional methods. Towards focusing attention on methods that need to be better addressed for fielding large scale sensor network applications, we identify in this section a number of lessons that we learned during this project.

12.1 Impact of Network Unreliability

Existing methods for a number of problems in sensor networks, including classification and tracking, often assume that the network is reliable. However, our experience and recent literature [20, 24, 25] corroborate that communication in dense sensor networks is significantly unreliable. While the unreliability can be partly attributed to the quality of hardware, available resources, and the nature of the wireless media, we find that message collision – both direct and indirect – is a key underlying factor. Collisions are positively correlated with the scale and density of the sensor network and the amount of traffic that is generated.

We learned that network unreliability cannot be ignored in the design of the network services or the application. In fact, satisfying the application requirements required the hand-in-hand selection, design, and

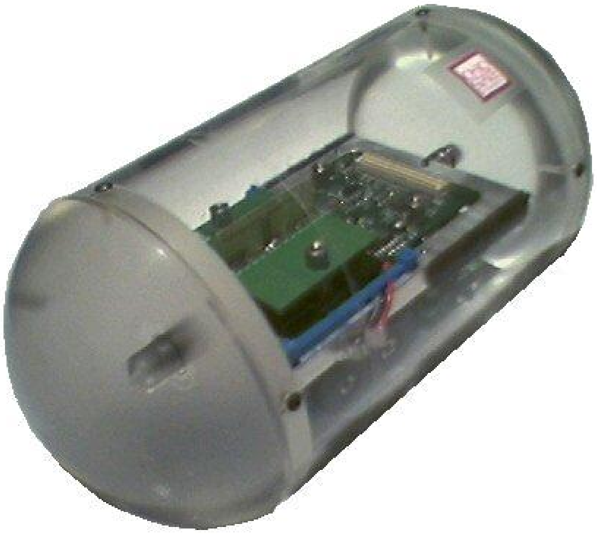

Fig. 9. Enclosure.

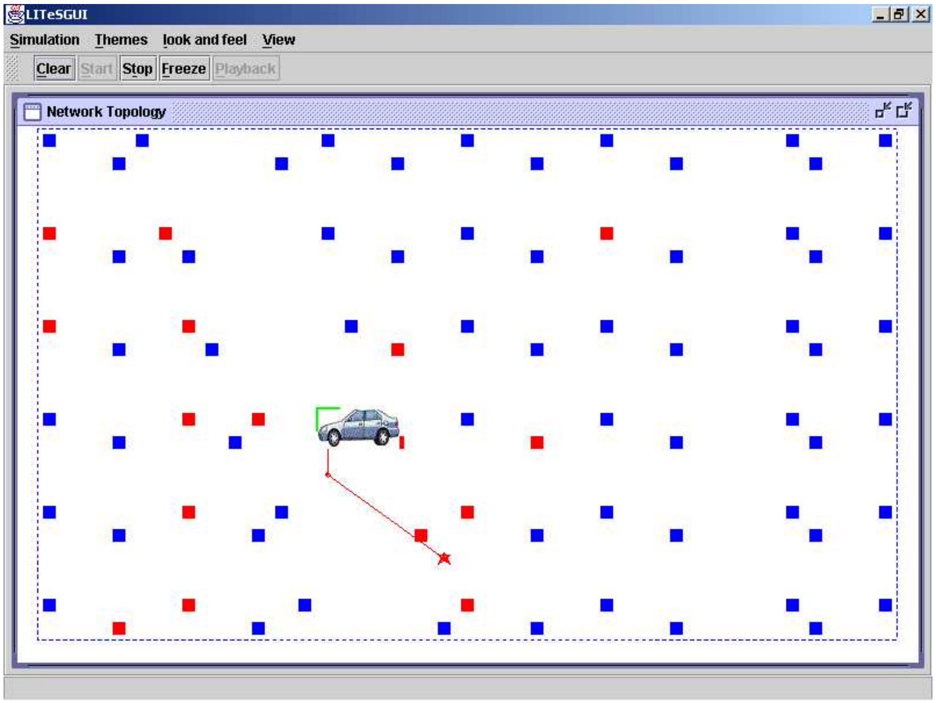

Fig. 10. The software for visualizing the network and targets.

tuning of the network services and the application components. By way of a simple illustration, Figure 6 shows that the latency parameter selected for the classification application depends on our choice of the end-to-end message delivery protocol and the number of retransmissions parameter selected in the latter depends on the accuracy requirements of the former. We thus argue that it is important to develop methodology for co-design of network services and application components and to study sufficient conditions under which the two can be systematically decoupled.

12.2 Unanticipated Faulty Behavior

Sensor nodes fail frequently and in complex ways. Failures range from node failstops to transient state corruptions to unanticipated, unpredictable program behavior. We experienced several of failures of the last category, as a result of undetectable incorrectly downloaded programs, depleted energy levels, debonding of sensors or other components, desensitization of sensors over time, and extreme environmental conditions (e.g., heat, moisture, and water). While all of our protocols could self-stabilize in the presence of node failstops and transient faults, they could not always self-stabilize in the presence of unanticipated, unpredictable behaviors. The result in those cases was sometimes quite serious, with a few misbehaving nodes affecting the entire network or application. For example, a node would constantly detect false events when its sensor board was overheated or debonded. Such faults led to the incorrect generation of a large amount of detection message traffic at a single node. “Jabbering” nodes effectively disabled all communications to the operational nodes in its neighborhood.

We learned that some complex faults are readily dealt with by better packaging, hardware and by incorporating redundancy techniques, but for the rest it is necessary to provide support to detect, contain, and correct them. In particular, the detection and correction of local invariants or contracts was useful, e.g. to control jabbering nodes to not communicate events at a rate higher than normal. We note that there is a considerable body of theoretical work that deals with node failstops, transient state corruptions, and Byzantine behavior, and argue that it is important to demonstrate the applicability of this work to sensor networks.

12.3 Testing at Scale

Our experience contains multiple instances in which network services and applications tested for a certain number of nodes failed to satisfy their requirements when they were scaled up by an order of magnitude. For instance, the reliable communications service achieved better than 95% reliability when tested with 9-16 nodes but its reliability fell to 50% when tested with 100 nodes where the background and application traffic levels increased significantly.

We learned that system scale must be considered at the time of testing. Fine-grain testing is desirable, whereby individual network/application-level implementations can be tested in the presence of simulations of the other services and in components. Realistic modeling of the application, the environment, and the network (topology and services) is important to this end. We thus argue that it is important to develop an integrated simulation and testbed environment early in the application development process.

12.4 Quality of Network Reprogramming

As the number of network nodes grows and the complexity of the application scales, the need for node reprogramming – even during operation – grows. The distributed nature and scale of large sensor networks will make reprogramming by hand practically if not actually impossible. Reprogramming via the network is thus highly desirable.

We learned that on-the-fly network reprogramming must contend with the network unreliability before it can be efficient, reliable, and scalable. Message losses and corruptions during program download over the network resulted in poor performance and arbitrary node failures in the first generation of sensor network reprogramming services. Specifically, the amount of time it took to program each node was too long, the number of nodes that could be simultaneously programmed was not high enough, and, as we noted above, there were scenarios in which nodes were undetectably programmed in an incorrect manner, which yielded

Byzantine behavior. We note that very recently there have been some promising first steps that have been taken towards addressing aspects of this problem [26, 27].

Incidentally, while reprogramming is desirable when the node program evolves in an unanticipated way, we encountered several scenarios in which observation and control of known node parameters was desirable. As mentioned above, network and application level parameters are co-dependent, and so the monitoring and tuning was an on-going process that benefited from a light-weight alternative to reprogramming. We were thus led to developing services for systematically exposing, accessing, and controlling parameters via monitoring and tuning interfaces. Such fine-grained services were successful in enhancing the observability and controllability of our sensor network application.

13 Conclusions and Future Work

This paper reports on our experiences developing and fielding an experimental wireless sensor network for distributed intrusion detection. The key elements of the problem include detection, classification, and tracking. The influence field, a novel spatial statistic useful for classification is defined, and a distributed estimator for it is developed and tested. We find that the estimator performance varies for different target classes and network reliability levels. Consequently, we treat the unreliablility of the network as a first-class parameter and simultaneously evolve the network reliability and application performance to achieve acceptable results. We believe our work is unique in that it explicitly details the degradation in application performance in sensor networks as a function of network unreliability. Our experimental observations are based on more than a thousand empirical tracks tested at a dozen sites including multiple deployments at MacDill Air Force Base in Tampa, Florida. Based on this nearly year-long study, we identify several lessons that will be of practical value to sensor network designers.

It is a widely held notion that someday there will be sensor network deployments consisting of hundreds of thousands of nodes. The challenges involved in scaling to a network of this size are quite different than the ones encountered in fielding much smaller networks. Obviously, the fundamental services of detection, estimation, classification, tracking, time synchronization, localization, and routing will remain important. However, the key challenges will center around robustness in the face of unreliability and incomplete information. Research efforts into self-stabilization [28], recovery-oriented computing [29], and autonomic computing [30] may provide a foundation for robust and self-regulating sensor networks that:

1. Are roughly aware of their constituent components, the approximate state of these components, and their connections to external systems.
2. Adjust and aggregate the behavior of the individual components into a cohesive whole.
3. Adapt to dynamic environmental conditions on their own.
4. Actively seek configurations that optimize their performance without causing instability.
5. Heal themselves in the face of transient and permanent failures.
6. Upgrade themselves by disseminating new code that they becomes aware of through their connections to external systems.

Our future research efforts will explore the new regime of *autonomic sensor networks* that are truly self-regulating in a manner similar to the autonomic nervous system. To motivate initial investigation into this area that is well grounded in experimental work, we plan to design, build, field, and test a sensor network consisting of more than 10,000 nodes and spanning several miles during the next year. This goal represents a scale that is impossible to achieve without incorporating many elements of *autonomic sensor networks.*

References

1. A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, “Habitat monitoring: Application driver for wireless communications technology,” *In Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Apr. 2001.
2. Alan Mainwaring, Joseph Polastre, Robert Szewczyk, and David Culler, “Wireless sensor networks for habitat monitoring,” *ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.
3. Mark Hewish, “Reformatting fighter tactics,” *Jane's International Defense Review*, June 2001.

4. J. Liu, P. Cheung, L. Guibas, and F. Zhao, "A dual-space approach to tracking and sensor management in wireless sensor networks," *In Proc. 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications*, pp. 131–139, Apr. 2002.
5. J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *Journal on Applied Signal Processing*, 2002.
6. Javed Aslam, Zack Butler, Florin Constantin, Valentino Crespi, George Cybenko, and Daniela Rus, "Tracking a moving object with a binary sensor network," *Proceedings of ACM Sensys 2003, Los Angeles, California*, 2003.
7. Marco Duarte and Yu-Hen Hu, "Vehicle classification in distributed sensor networks," *Journal of Parallel and Distributed Computing*, 2004, to appear.
8. Michael J. Caruso and Lucky S. Withanawasam, "Vehicle detection and compass applications using amr magnetic sensors," .
9. C. S. Raghavendra C. Meesookho, S. Narayanan, "Collaborative classification applications in sensor networks," *Second IEEE Sensor Array and Multichannel Signal Processing Workshop*, Aug. 2002.
10. Feng Zhao, Jie Liu, Juan Liu, Leonidas Guibas, and James Reich, "Collaborative signal and information processing: An information directed approach," *Proceedings of the IEEE*, 2003, to appear.
11. Ashwin D'Costa and Akbar Sayeed, "Collaborative signal processing for distributed classification in sensor networks," *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, pp. 193–208, Apr. 2003.
12. Dan Li, Kerry Wong, Yu Hu, and Akbar Sayeed, "Detection, classification and tracking of targets in distributed sensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17–29, Mar. 2002.
13. Donal McErlean and Shrikanth Narayanan, "Distributed detection and tracking in sensor networks," *36th Asilomar Conf. Signals, Systems and Computers*, 2002.
14. Feng Zhao, Jaewon Shin, and James Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, Mar. 2002.
15. Jason Hill and David Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, 2002.
16. S. Adlakha, S. Ganeriwal, C. Schurgers, and M. B. Srivastava, "Density, accuracy, delay and lifetime tradeoffs in wireless sensor networks: A multidimensional design perspective," *Proceedings of ACM Sensys 2003, Los Angeles, California*, 2003.
17. Juan Liu, Jie Liu, James Reich, Patrick Cheung, and Feng Zhao, "Distributed group management for track initiation and maintenance in target localization applications," *Proceedings of 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, Apr. 2003.
18. Prabal K. Dutta and Anish K. Arora, "Sensing civilians, soldiers, and cars," *The Ohio State University Department of Computer and Information Science Technical Report OSU-CISRC-12/03-TR66*, 2003.
19. Mohamed G. Gouda, Young-Ri Choi, Moon C.Kim, and Anish Arora, "The mote connectivity protocol," *Proceedings of the International Conference on Computer Communication and Networks (ICCCN-03)*, 2003.
20. Alec Woo, Terence Tong, and David Culler, "Taming the underlying issues for reliable multhop routing in sensor networks," *Proceedings of ACM Sensys 2003, Los Angeles, California*, 2003.
21. J. Hill, "A software architecture supporting networked sensors," *Master's thesis, U.C. Berkeley Dept. of Electrical Engineering and Computer Sciences*, 2000.
22. David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer, and David Culler, "The nesc language: A holistic approach to networked embedded systems," in *Proceedings of Programming Language Design and Implementation (PLDI) 2003*, 2003.
23. Prabal K. Dutta and Anish K. Arora, "Integrating micropower radar and motes," *The Ohio State University Department of Computer and Information Science Technical Report OSU-CISRC-12/03-TR67*, 2003.
24. Jerry Zhao and Ramesh Govindan, "Understanding packet delivery performance in dense wireless sensor networks," *Proceedings of ACM Sensys 2003, Los Angeles, California*, 2003.
25. Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin, and Stephen Wicker, "Complex behavior at scale: An experimental study of low-power wireless sensor networks," *UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013*, 2003.
26. Thanos Stathopoulos, John Heidemann, and Deborah Estrin, "A remote code update mechanism for wireless sensor networks," *UCLA CENS Technical Report # 30*, 2003.
27. S. S. Kulkarni and M. (U) Arumugam, "Tdma for sensor networks," *Assurance in Distributed Systems and Networks*, 2004, to appear.
28. A. Arora and M. G. Gouda, "Distributed reset," *IEEE Transactions on Computers*, vol. 43, no. 9, pp. 1026–1038, 1994.
29. D. A. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman, and N. Treuhaft, "Recovery-oriented computing (roc): Motivation, definition, techniques, and case studies," *UC Berkeley Computer Science Technical Report UCB//CSD-02-1175*, Mar. 2002.

30. J. Kephart and D. Chess, “The vision of autonomic computing,” *IEEE Computer*, pp. 41–50, Jan. 2003.