

FAULT TOLERANCE IN WIRELESS SENSOR NETWORKS

Farinaz Koushanfar¹, Miodrag Potkonjak², Alberto Sangiovanni-Vincentelli¹

¹*Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA, US 94720*

²*Department of Computer Science
University of California, Los Angeles
Los Angeles, CA, US 90095*

Abstract: In this Chapter, we address fault tolerance in wireless sensor networks. In order to make the presentation self-contained, we start by providing a short summary of sensor networks and classical fault tolerance techniques. After that, we discuss the three phases of fault tolerance (fault models, fault detection and identification and resiliency mechanisms) at four levels of abstractions (hardware, system software, middleware, and applications) and four scopes (components of individual node, individual node, network, and the distributed system). The technical cores of the chapter are two case-studies on heterogeneous fault tolerance and discrepancy minimization-based fault detection and correction. We conclude the chapter with a brief survey of the future directions for fault tolerance research in wireless sensor networks.

Key words: Wireless multi-hop networks, Wireless Sensor networks, fault-tolerance, error resiliency.

1. INTRODUCTION

1.1. Motivation

The reliability of computer, communication, and storage devices has been recognized early as one of the key issues in computer systems. Since the 1950's, techniques that enhance the reliability of computer and communication systems were developed both in academia and industry. It has been also recognized that as complexity of computing and communication devices increases, fault-tolerance will gain more importance. Surprisingly, fault tolerance has never been the major design objective. While there are a number of reasons for this situation, the most important is that the reliability of individual components has been increasing at a much more rapid pace than it was expected. In addition, creative packaging and cooling schemes tremendously reduced the stress factor on computation and communication systems.

The only component of fault tolerance that has received a great deal of attention in industry was off-line testing. The modern testers are \$10+ million systems that are contributing increasingly to the cost of modern microprocessors. In addition, the percentage of logic that supports testing has been rapidly increasing in the last 10 years, from less than 1% to more than 5% of the total transistor count.

The rapid growth of the Internet in the last 10 years was the first major facilitator of the renewed interest in fault tolerance and related techniques such as self-repair [Bro01a, Bro01b, Bro02, Can02]. Internet requires the constant mode of operation and therefore special effort has been placed to develop fault tolerant data centers. We believe that the emergence of wireless sensor networks will further increase the importance of fault tolerance. At the same time, wireless sensor networks will impose a number of unique new conceptual and technical challenges to fault-tolerance researchers. There are at least three major groups of reasons why research in fault tolerant sensor networks should receive a significant attention. The first one is related to the technology and implementation aspects. At least two components of a sensor node, sensors and actuators, will directly interact with the environment and will be subject to a variety of physical, chemical, and biological forces. Therefore, they will have significantly lower intrinsic reliability than integrated circuits in fully enclosed packaging. In addition, wireless sensor nodes themselves are exceptionally complex systems where a variety of components interact in a complex way.

Furthermore, hundreds or maybe thousands of these nodes will form a distributed embedded network system that will handle a variety of sensing,

actuating, communicating, signal processing, computation, and communication tasks. Wireless sensor networks will be often deployed as consumer electronic devices that will put significant constraints on the cost and therefore, quality of used components. More importantly, nodes operate under strict energy constraints that will make energy budget dedicated to testing and fault tolerance very limited.

The second reason is that applications will be equally as complex as the involved technology and architectures. More importantly, sensor networks will often operate in an autonomous mode without a human in the loop. Furthermore, security and privacy concerns will often prevent extensive testing procedures. Note that not just testing and fault tolerance will be adversely impacted, but also related tasks such as debugging, where reproduction of specific conditions under which fault has occurred will be difficult. Also, applications will require that sensor nodes are often deployed in uncontrolled and sometimes even hostile environments. Finally, and maybe most importantly, many applications of sensor networks will be safety critical and can have very adverse impact on humans and the environment, in particular when the actuators are used.

The final reason is that wireless sensor networks themselves are a new scientific and engineering field and it is not still quite clear as to what is the best way to address a particular problem. In this situation, it is also difficult to accurately predict the best way to treat fault tolerance within a particular wireless sensor network approach. In addition, both technology and envisioned applications for wireless sensor networks are changing at a rapid pace. For example, if we consider power consumption, each particular scheme will depend significantly on the relative power consumption of different approaches. Specifically, if communication energy is significantly higher than the computation energy, then it is important to develop localized algorithms that will require only a limited amount of communication. Therefore, with respect to fault tolerance, it is important to consider schemes that conduct error detection using only local information. Or, if one wants to ensure fault tolerance during the sensor fusion, the goal is to design fault tolerant techniques that do not significantly increase the communication overhead. On the other hand if the computation energy is significantly higher than the communication requirements, it is a good idea to support communication resources at one node with the computation resources at other nodes. It is preferable to develop fault tolerant sensor fusion approaches that require little additional computation regardless of any additional communication requirements.

1.2. Objectives

Our primary goal is to survey the field of fault tolerance in sensor networks. We consider fault tolerance at four different levels of abstraction, starting from hardware and system software and going to the middleware and application layers. We consider fault tolerance at each level of six individual components of a node: computing engine, communication and storage subsystems, energy supply, sensors, and actuators. We also consider fault tolerance at the level of a node itself, as well as the network level. Finally, we consider resiliency against errors where wireless sensor networks are treated as embedded distributed systems.

We consider three aspects of fault tolerance, namely, fault models, error detection and diagnosis techniques, and resiliency mechanisms. In order to provide in depth treatment of specific approaches we present two case studies. One on error detection in sensor networks and one on heterogeneous built-in-self-repair (BISR) based fault tolerance.

Finally, in order to provide a global vision, we discuss the relationship to sensor networks and traditional fault tolerance techniques as well as a set of our predictions of future research directions in this field.

The remainder of the chapter is organized in the following way. In the next two sections, we provide relevant preliminary information. After that, we discuss fault tolerance at the node and at the network levels. Next, we present two case studies where we address in more comprehensive way several technical details with respect to fault tolerance in sensor networks. Finally, we address the future directions along the three dimensions of fault tolerance.

2. PRELIMINARIES

2.1. Sensor network

A wireless sensor network is a system of small, wirelessly communicating nodes where each node is equipped with multiple components. In particular, each node has a computation engine, communication and storage subsystems, a battery supply, sensing, and in some cases actuating devices. Such a network is envisioned to integrate the physical world with the Internet and computations. The power supply on each node is relatively limited, and frequent replacement of the batteries is often not practical due to the large number of the nodes in the network. Therefore, energy is the most constraining factor on the functionality of such networks. In order to save energy, nodes only use the short range

communications which is proven to be much less energy consuming than the long range [Rab00]. The short range communication between the nodes implies localized interaction in the network.

There is a need to model the different components of a sensor network. Sensor networks are often abstracted and mapped into a graph, where each vertex of the graph corresponds to a wireless node and there is an edge corresponding to the communication between two nodes. If the communication between the nodes is bidirectional, the mapped graph of the network will be non-directed. However, if the communication between the nodes is asymmetric, then the mapped graph becomes directed. The communication model between the nodes can be either one-to-one, or one-to-many. In the one-to-one communication model, each node sends and receives messages from only one of the communication edges. In the one-to-many communication model, each message sent out by a node can be heard by all of its neighbors. Providing a reasonable and practical model for sensors and actuators is a much more complex task. This is due to the great variety of different sensors, both in terms of their functionality and in terms of their underlying technologies.

There are a lot of potential applications that are envisioned for sensor networks. For example, they can be used in a battlefield, where they can detect and spy on the enemies or they can support the positive forces. Also, they can be used in intelligent security systems in buildings and security critical applications. They can be used for habitat monitoring applications where they can monitor and study the changes in phenomena for a long time. Comprehensive surveys on sensor networks include [Aky02, Fen02, Int00].

3. EXAMPLE FOR FAULT-TOLERANCE IN A SENSOR NETWORK SYSTEM

The problem of fault-tolerant multimodal sensor fusion for digital binary sensors can be informally introduced using the example shown in Figure 1a-b. A sensor network recognition system is deployed in an office to identify the people in that office as they walk in through the main door. Six people, named A, B, C, D, E, and F, work in the office. The system consists of two different types of sensors: (1) a height sensor that is a set of light sensors in series, (2) a voice recognition sensory system that requires everybody entering the room saying out a given pass phrase in a microphone. Figure [1a] shows the selected identification characteristics of people in the office. Figure [1b] shows the same characteristics mapped to a two-dimensional plot.

It is easy to see that the system can distinguish between two persons P_1 and P_2 , if they fall into different squares, when mapped to the chart shown in figure [1b]. If all of the sensors work properly, each person will naturally fit into a different square, according to the figure. For most of the cases, even if one of the height sensors or voice sensors fails, the recognition of the right person is still possible. This is accomplished using heterogeneous fault tolerance, where the failed sensor of one type, can be replaced by the functionality of a sensor of another type. However, for the case of the persons B and E who are the same height, voice figure is the only way to distinguish the two persons, so the system does not have any fault tolerance to the failure of the sensor v_3 that distinguishes between the objects B and E. If the office had only 5 people (excluding either B or E), then it would be completely fault tolerant. Complex sensor network systems can be designed in such a way that the heterogeneous fault tolerance scheme can enable the system resilient to the failure of a specified number of sensors of specified modality.

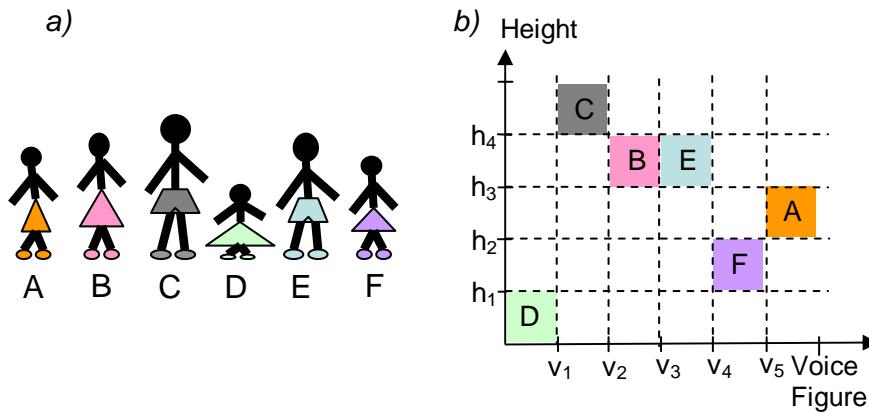


Figure 1 – Example of a multi-modal sensor network system

4. CLASSICAL FAULT TOLERANCE

Fault tolerance emerged early as a major concern during design of digital computing systems. For example, the Bell Lab's relay-based computer was performing multiple computations for the same inputs using the same program, in order to compare results and detect potential temporary malfunctioning. Also, the UNIVAC 1, the first computer that was commercially available in 1951, utilized both parity checking and arithmetic unit replication to enhance its reliability. At the same time, fault tolerance

received research attention in the mid 50s. Both Moore and Shannon [Moo56] and von Neumann [von56] conducted studies on how to design systems that preserve functionality after a subset of components experience failures. Also, embedded computing systems such as Bell Lab's electronic switching system had extensive support for fault tolerance mainly through enhanced serviceability features [Dow64]. Furthermore, APOLLO's mission to Moon was using triplicated computers in order to maximize the chances of success [Coo76].

During 70's, research on fault tolerance started to diverge along several lines. Initially, the two most important and influential were fault tolerance in VLSI-based systems and fault-tolerance in distributed system [Lam82, Lam83, Pea80, Sch84]. More recently, fault tolerance in data base [Gra93, Bit88], internet (e.g. reliable multicast and reliable distributed storage, peer-to-peer), and self-repair have been topics of major importance.

In the VLSI systems, fault-tolerance has been addressed at all levels of abstraction, including circuit level, logic level, register transfer level, program level, and system level. It is common that reliable system design is discussed within three stages of life of a product: design, manufacturing and operational. Before we go into more technical details, it is important to define the basic entities.

Error is the manifestation of a fault inside of a program. It is important to note that error can occur not just at the fault site, but also at some distance. Fault is an incorrect state of hardware or a program as a consequence of a failure of a component. Permanent faults are the ones that are continuous and stable in time. For example, permanent hardware faults are consequences of irreversible physical alteration within a component. An intermittent fault is one that has only occasional manifestation due to unstable characteristic of the hardware, or as a consequence of a program being in a particular subset of space. Finally, a transient fault is one that is the consequence of temporary environmental impact on otherwise correct hardware. For example, often the impact of cosmic radiation may be transient.

There are three main types of concerns that fault tolerance considers: fault models, fault detection and diagnosis, and resiliency mechanisms. Each level of abstraction has its own types of faults [Sie92]. For example, at the gate level, several fault models have been successfully used in the testing phase. One model is "stuck at" where the logical value on interconnect gate or pin is permanently set to a value stuck at one or stuck at zero. Another model is the "bridging" fault model, where two or more neighboring signal lines are physically connected introducing either wired AND or wired OR functions depending on the used logic family. Shorts and opens are another class of faults and they correspond to missing or additionally introduced

connections respectively. Finally, unidirectional faults are faults where an error occurs in the same logical direction when a certain single failure occurs. Typical examples are where an open circuit in a memory-select line results that a particular word is incorrectly read as all ones, instead of all zeros. It is interesting and important to note that almost all testing approaches assume a single fault model, regardless of which type of just mentioned fault is considered.

Reliability techniques compromise of the following phases. The first one is fault confinement, where limits of fault effects are established over a particular area and therefore, contamination of other areas is prevented. Fault detection is a phase where it is recognized that an unexpected event has occurred. Fault latency is the time that passes between the fault occurrence and the moment when the fault is detected. Traditionally, fault detection techniques are classified into offline and online detections. Most often, for offline detection, special diagnostic programs are employed, either during idle periods of time, or using multiplexing with a regular mode of operation.

Online detection targets real-time fault identification and is performed simultaneously with a real work load. Typical online detection techniques include parity checking, duplication and triplication. Diagnosis is a stage where the exact occurrence of a fault is attributed to a specific atomic piece of hardware. Reconfiguration is the stage that is entered after diagnosis and where the system is restructured in such a way that faults do not have an impact on the correct output. Graceful degradation is a reconfiguration technique where performance of the system is reduced, but the correct functionality is preserved. Recovery is a stage where an attempt to eliminate the effects of faults is conducted. Two most widely used recovery techniques are fault masking and retry. The fault masking approach is one where redundant correct information is used to eliminate the impact of incorrect information. In retry, after the fault is detected, a new attempt to execute a piece of a program is made in the hope that the fault is transient. Restart is the stage that is invoked after the recovery of correct undamaged information. In cold-restart, a complete resetting of the system is conducted. Repair is the stage where the failed component is substituted with the component that is operational. There exist a number of excellent surveys, special issues, and textbooks on fault-tolerance [Lee90, Jal94].

5. FAULT TOLERANCE AT DIFFERENT LEVELS OF THE SENSOR NETWORK

5.1. Physical layer

The physical layer is responsible for establishing communication in a given medium between two nodes. Typical tasks at this level include modulation-demodulation, and encoding-decoding. Traditionally fully hardwired solutions have been used in order to minimize the cost and maximize the energy efficiency. A software radio is a wireless communication device in which parts or all of the physical layer functions are realized in software [Man99].

Software radios are a way to extend programmability into the physical layer and to enable adaptation to channel conditions. It has been demonstrated that adaptive link layer techniques can significantly improve the performance of wireless networks [Eck96, Eck98a, Eck98b, Fra97]. The first commercially available radio was the speakeasy device [Lac95]. Comprehensive surveys about software radios and related technologies include [Mit95, Mit99, Tut99]. Currently the major commercial application driver is incompatibility between the number of cellular and PCS communication standards. While the primary reason for the deployment of hardware and software radios has been to solve interoperability problems and to enhance performances in noisy media, there are also ideally suited for realization of a variety of fault tolerance techniques at the physical layer. For example, if some components of the software radio are not functional, one can switch to modulation and encoding schemes that can be realized with the still operational hardware resources. Note that adaptation to noise characteristics can also be considered from the fault tolerance point of view.

5.2. Hardware

At the hardware level, components can be divided into two groups. The first group consists of a computation engine, storage subsystem and power supply infrastructure that are all very reliable. It is not just that there are exceptionally reliable systems available that incorporate sophisticated fault tolerance techniques, but even off-the-shelf microprocessor, DSP processor and controllers are very reliable devices that have a very low rate of malfunctioning. There exist at least three main reasons why this does not necessarily imply that computational subsystems of sensor nodes will be exceptionally reliable. The first is that sensor nodes are very cost sensitive and therefore will not always be able to design using the highest quality components. The second is that strict energy constraints imply that repeated

computations are often not realistic options. The third is that these systems are often deployed in much harsher environments than today's computers. Although programmability and flexibility are of high importance in sensor networks, strict energy constraints will result in extensive use of application specific designs that can have up to two orders of magnitude less energy consumption for the same functionality. For these subsystems, we expect that the heterogeneous BISR fault tolerant schemes will provide the targeted level of fault tolerance and simultaneously low energy consumption [Gue98].

With respect to storage components, there exist several options. Memories can be divided into volatile and non-volatile. Volatile memories are used for short term storage. The best known and most widely used are SRAM and DRAM. Due to their relatively simple and regular structure, both types are very reliable, in particular DRAM. Starting with 4 Megabit components, BISR has been often used to enhance the yield of DRAM memories. Note that memories are designed using the standard semiconductor processes. Non-volatile memories such as flash and MRAM are also very reliable. Flash has the restriction that one cannot write too many times at the same location and therefore it will be mainly used for storage of system programs. Therefore, we can conclude that the storage systems are generally fault resilient and it is very rarely that they will be a fault tolerance bottleneck.

With respect to energy supply, we can distinguish two parts of the energy supply subsystem. Traditional energy sources are rechargeable batteries and fault tolerance is achieved by providing a back up battery. The first commercial fuel cells and subsystems that leverage on energy scavenging have been recently demonstrated. While it appears that fuel cells will be very reliable, some energy scavenging subsystems such as the ones converting light into energy, can have very volatile performance. For energy distribution the standard solution to enhance fault tolerance is to deploy multiple distribution networks.

Another component that greatly depends on the surrounding environment is the wireless radio. The standard way to enhance the performance of radios is to use aggressive error correction schemes and retransmission. These two schemes are examples of time redundancy. In addition, several schemes have been proposed where two or more radios are used. Although the primary goal of these approaches is to save energy, they can be also used to enhance fault tolerance.

In addition to the schemes that operate on the physical and link layer, it is important to mention techniques that operate at the network layer. However, these schemes are more naturally considered and implemented at the system software level.

As we have already mentioned, sensors and actuators are subsystems that are most prone to malfunctioning. In the case of sensors, we can distinguish three types of faults: (1) calibration systematic error, (2) random noise error and (3) complete malfunctioning. While the first two can be addressed through time redundancy, the last one is enhanced using hardware redundancy. Depending on the type of data and information processing, in particular effective fault tolerance scheme for sensors can be accomplished using heterogeneous BISR techniques [Kou02]. Currently, no scheme other than hardware redundancy is envisioned for actuators.

5.3. System software

System software consists of the operating system (OS) and utility programs. There are several ways how one can address fault tolerance at the system software level with respect to the computational subsystem [Bir85, Bir87, Bir94]. Probably the most promising is through software diversity, where each program is implemented in n different versions in hope that different versions will not have identical bugs [Avi84, Avi85]. The subsystem that can most benefit from fault-tolerance realized at the system software level is the communication unit. For example, one can reroute messages using different paths in the multihop network. With respect to sensors and actuators, the most important piece of system software is the one related to calibration. Recently, a number of schemes have been proposed for this task [Whi02, Byc03]. A very important component of system software is the one that supports distributed and simultaneous execution of localized algorithms. For example, in the case of energy minimization under functionality constraint requirements, several protocols have been developed for the coordination of distributed actions [Kou03b, Che01]. It is important to note that when communication protocols are considered, there is a clear trade-off between its complexity and its effectiveness.

5.4. Middleware

While at the system software level, in addition to the OS of the individual nodes, networking (communication) plays the most dominant role. Starting with the middleware level the emphasis is shifted toward data aggregation, data filtering and sensor fusion. These are the tasks that are mainly related to sensor readings. Since it is difficult to provide fault tolerance in an economic way at the level of a single sensor, we expect that numerous fault tolerant approaches for this task will appear at the middleware level. While currently the majority of applications are very simple, in order to address the real-life applications, there is a need to develop much more complex middleware. In

order to combat software faults, n-versioning is one of the options [Avi84, Avi85]. In particular, we believe that heterogeneous approaches that can substitute the readings of one type of sensors with the readings of another type of sensor are very important due to their low overhead. Another important issue that is solely related to middleware is how many sensors of each type should be placed on a particular node and on which positions. If error resiliency of communication is much higher than the error resiliency of sensors, then solutions where the sensors of the same node are placed on the same node will be favored.

5.5. Application

Finally, fault tolerance can be addressed also at the application level. For example, if one wants to identify a particular person, he can try to measure using the sensors a variety of biometric features of that person. Each feature and possibly a combination of features will be sufficient to identify that person. While addressing fault tolerance at the application level maybe very efficient, unfortunately any given application will require a customized way to properly address the issue. On the other side, an additional advantage of application level fault tolerance is that it can be used to address faults in essentially any type of resource.

6. CASE STUDIES

6.1. Heterogeneous fault detection

Before we describe this case study, we briefly outline key facts and assumptions about fault models, fault detection, and embedded sensor networks.

Each sensor node has five components: computation, communication, storage, sensors, and often actuators. Widely accepted fault and error models for processors, FPGA-based components, SRAM and DRAM, non-volatile memory and disks, and communication systems are readily available. However, the situation for actuators and sensors is very different. Both types of resources are conceptually more complex and intrinsically more diverse to allow for simple, yet realistic and widely applicable fault and error models.

In the work presented by Koushanfar et. al. [Kou02], they restrict our attention on faults in sensors. They adopt two fault models. The first one is related to sensors that produce binary outputs. In this case, obviously, one can envision a number of applicable fault models. For example, one model can capture probability or statistics of erroneous reported result.

Nevertheless, it appears that the most logical and with potentially largest applicability range is the permanent fault model where only possible outcomes are that either the sensor is functional or not. For this fault model, the fault detection procedure is often straightforward: usually just observing the output of the sensors.

The second fault model is related to the sensors with continuous (analog) or multilevel digital outputs. The fault models for this type of sensors are even additionally more complex and diverse. They propose to measure the level of discrepancy of the output of individual sensor with the multimodal model used for fusion as the indication of the level of error in that sensor.

The approach has two key advantages. The first is that the fault tolerance approaches are such that the developed technique is applicable to great variety of fault models. The approach is in particularly well suited for addresses transient errors and errors in measurements. The second advantage is that the approach simultaneously addresses fault detection and correction. Overall, they made only mild assumptions: the main being that majority of sensors are functioning correctly.

The SENSOR RESOURCE ASSIGNMENT (SRA) PROBLEM can be formulated in the following way.

INSTANCE: Set A_1 of points p_i (x_{i1}, \dots, x_{im}), in m -dimensional space where $1 \leq i \leq n$, a positive integer J_1 , set H that consists of $m(n-1)$ [$m-1$]-dimensional hyperplanes that are perpendicular to one of the m axes, such that each hyperplane is separating two points p_i and p_j that have the closest coordinates along the axis to which the hyperplane is perpendicular.

QUESTION: Find a subset of selected hyperplanes H , such that any two points p_i and p_j are separated by at least one of the selected hyperplanes and also the cardinality of H is at most J_1 .

Claim: SENSOR RESOURCE ALLOCATION (SRA) is NP-complete.

Next, they present their approach and algorithms for fault tolerant sensor assignment. Although it is easy to envision a monolithic solution that simultaneously considers fault tolerance requirements and sensor allocation and assignment problem, following principles of separation of concerns and orthogonality, they designed a fully modular system that has separate optimization mechanisms for the subtask: sensor assignment, sensor allocation, and fault-tolerance. These three steps are addressed in the following way.

They employ two different algorithmic engines to RSA problem: ILP-based and simulated annealing based. The rationale behind the integer linear

programming (ILP) approach is that although ILP solvers are often not fast, they are attractive since they guarantee optimal solution. In addition, for many smaller instances of practical importance can be solved using this approach. The points is that they have to find the solution to the SRA problem before the deployment, so it is one time expense in computational time on workstation and may be acceptable. In the cases when ILP is not applicable, they provide the option of using simulated annealing as the optimization mechanism.

The ILP formulation for the SRA problem can be stated in the following way.

INPUTS: set of n , m -dimensional points $p_i(x_{i1}, x_{i2}, \dots, x_{im})$, $1 \leq i \leq n$. Set of all possible tests T , with elements t_k ($1 \leq k \leq m(n-1)$), where the $(l(n-1)+1)$ to $(l+1)(n-1)$ tests are in dimension l , $1 \leq l \leq m$, each separating two closest point in that dimension. The cost of each test t_k is c_k .

They define the variable X_k as followed:

$$X_k = 1 \text{ if test } t_k \text{ is selected}$$

$$X_k = 0 \text{ otherwise.}$$

The objective function is to minimize the total cost of all of the selected tests. In another word:

OF:

$$\sum_{k=1}^{m(n-1)} X_k \cdot c_k$$

The constraint of the problem is that for each pair of points p_i and p_j , there should be at least one test that has a different outcome when applied to these two points. They define an auxiliary matrix $A[n \times k(m-1)]$ with constant elements a_{ik} ,

$$a_{ik} = 1 \text{ if the test } t_k \text{ produces 1 on point } p_i$$

$$a_{ik} = 0 \text{ otherwise.}$$

Using the matrix A and the variables, they find a linear expression that produces zero, if a test produces similar results on the two points p_i and p_j and one otherwise. One such expression is $X_k \times (a_{ik} + a_{jk}) \times (1 - a_{ik} \times a_{jk})$ that has the required property. Therefore, to have a different test result on each set of two points p_i and p_j , they write the following constraints.

CONSTRAINTS: For each pair of points p_i and p_j ,

$$\sum_{k=1}^{m(n-1)} X_k \cdot (a_{ik} + a_{jk}) \cdot (1 - a_{ik} \cdot a_{jk}) \geq 2$$

They used a standard simulated annealing code. The four components of simulated annealing (moves - neighborhood structure, objective function, cooling schedule, and stopping criteria) are defined in the following way. Move is the replacement of one sensor with another sensor of the same type. The goal is to maximize objective function. They use the standard geometric cooling schedule. Finally, as stopping criteria, we use the user specified number of steps in which the improvement did not occur.

The resource allocation is conducted in the following way. They first propose as the initial solution as the number of sensors that is lower bound on the potential solution. The bound is calculated assuming that all dimensions have the same number of sensors and each n-dimensional compartment will eventually contain one point. After that, they run the simulated annealing RSA algorithm. During this running process, they modify the move so that one type of sensor can be replaced with another type of sensor. They accumulate statistics about which type of sensor helps the most to improve objective function after each move and use this information to decide which type of sensor to add or remove.

For fault-tolerance, one can envision three different mechanisms. The first is to specify in the ILP formulation or in the simulated annealing code that each two points have to be separated by at least r hyperplanes. Since this approach essentially doubles the redundancy, they did not accept this alternative. The second alternative is to add exactly one extra sensor of each type to the solution generated by the sensor resource allocation problem. When large number of sensor nodes of each type is used, the overhead is relatively low. Also, in this case the need for storing or communicating more than one resource assignment solution is eliminated. Therefore, if moderate levels of fault tolerance are needed, this can be an attractive alternative.

The final and most attractive alternative in terms of overhead is to leverage on heterogeneous back up of sensors of different modality. Here they generate allocation in the following way. They first calculate for each type of sensors for all allocations k from 1 to smaller than the number allocated in the best resource allocation solutions, the cost of overall solution. After that they plot the cost of all these solutions on y-axis on the graph where the x-axis is the number of allocated sensors of analyzed type. In such a way they obtain m graphs, where m is the number of sensors of different modality. Obviously, now they have to use the RSA algorithm to analyze only allocations that are worse in terms of cost than the optimal solution and better than the solution from the second alternative. They conduct this analysis in the order dictated by increasing cost of the proposed solution.

There are a number of ways to generalize and therefore enhance the applicability of the technique presented in the above. One possibility is to characterize objects using statistical data and to build statistical model for decision making using data from sensors. Another, equally important and with equally large application domain option is to conduct multimodal sensor fusion in order to support decision process. As a matter of fact, the multimodal multilevel sensor fusion has emerged as one of canonical problems in sensor networks. Informally it can be defined in the following way. A number of sensors, some of them with different modalities, are given. The goal is to extract as accurately as possible the information requested by a user from noisy measurements.

Although it the problem seems too general to be efficiently solved using a single approach, there is a systematic way to address the problem. What is needed is to develop or even better to find some already developed analytic models that are related to the measured quantities. Once the equations of an analytical model are assembled, the intriguing and important question is to try to figure out which measurements are faulty or have high degree of noise. One way to answer this question is to try to find a subset of measurements that produce consistent set of analytic models. Using this set of equation, the value for all quantities of interest can be calculated. Therefore, the key for providing fault tolerant multimodal sensor fusion is to generate rich enough model of physical world and in particular to ensure that the system is solvable even when some of the equations are not used. The main difficulty is that the systems of equations are often nonlinear and therefore it is very difficult to say in advance when the system is well defined in a sense that it can be uniquely solved.

Probably the best way to clarify the introduced approach is to take a closer look at an example. For this purpose we will use scenario illustrated at Figure 2. We see an object O that moves along its trajectory that includes points p_i in embedded sensor network that consists of a number of nodes each represented by a shaded circle n_i . We have four types of sensors: RSSI-based distance discovery, speedometer, accelerometer, and compass that are used to measure the angle in 2D physical space. Three RSSI-based measurements can be used to locate the object O in any particular moment. Euclidian space, Newton mechanics, and trigonometry laws can be used to establish relationships between measurements. Specifically, Equations 1-9 are trilateration equations, Equation 10-13 are the Newton law equation and the Equations 14-15 are trigonometry laws. The key observation is that we have more equations (15) than variables (12) that may have errors. So, if one of sensor is not functioning, we can calculate it from the established system of equations. Also, for each variable, we can find how much it has to be altered in order to make the whole system of equations maximally

consistent. The variables that have to be altered the most are most likely measured by faulty sensors. Therefore, one way to identify and correct sensor measurements is to try all scenarios where exactly one type of sensor measurements is not taken into account and compare the maximal error in the system. Another very important observation is that by sampling all operational sensors more often, we can compensate for faulty sensors.

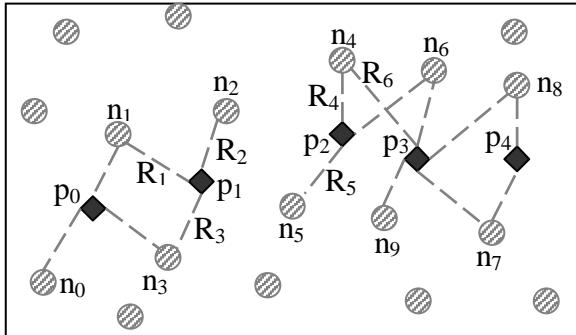


Figure 2 - tracking of an object by the sensors

$$(x_1 - s_1)^2 + (y_1 - t_1)^2 = R_1^2 \quad \text{Eq 1}$$

$$(x_1 - s_2)^2 + (y_1 - t_2)^2 = R_2^2 \quad \text{Eq 2}$$

$$(x_1 - s_3)^2 + (y_1 - t_3)^2 = R_3^2 \quad \text{Eq 3}$$

$$(x_2 - s_4)^2 + (y_2 - t_4)^2 = R_4^2 \quad \text{Eq 4}$$

$$(x_2 - s_5)^2 + (y_2 - t_5)^2 = R_5^2 \quad \text{Eq 5}$$

$$(x_2 - s_6)^2 + (y_2 - t_6)^2 = R_6^2 \quad \text{Eq 6}$$

$$(x_3 - s_7)^2 + (y_3 - t_7)^2 = R_7^2 \quad \text{Eq 7}$$

$$(x_3 - s_8)^2 + (y_3 - t_8)^2 = R_8^2 \quad \text{Eq 8}$$

$$(x_3 - s_9)^2 + (y_3 - t_9)^2 = R_9^2 \quad \text{Eq 9}$$

$$\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2} = \frac{1}{2} \cdot a_1 (\Delta t)^2 + v_1 (\Delta t) \quad \text{Eq 10}$$

$$\sqrt{(x_3 - x_2)^2 - (y_3 - y_2)^2} = \frac{1}{2} \cdot a_2 (\Delta t)^2 + v_2 (\Delta t) \quad \text{Eq 11}$$

$$a_1 \cdot \Delta t = v_1 - v_0 \quad \text{Eq 12}$$

$$a_2 \cdot \Delta t = v_2 - v_1 \quad \text{Eq 13}$$

$$\alpha_1 = \tan^{-1} \left(\frac{y_2 - y_1}{x_2 - x_1} \right) \quad \text{Eq 14}$$

$$\alpha_2 = \tan^{-1} \left(\frac{y_3 - y_2}{x_3 - x_2} \right) \quad \text{Eq 15}$$

6.2. Discrepancy-based fault detection and correction

The work by Koushanfar et. al. [Kou03] introduces a cross-validation-based technique for online detection of sensor faults. The approach can be applied to a broad set of fault models. They define a fault as an arbitrary type of inconsistent measurement by a sensor, which cannot be compensated systematically. In particular, they consider faults associated with the incorrect measurements that cannot be corrected using calibration techniques. The approach is based on two ideas. (i) comparing the results of multi-sensor fusion with and without each of the sensors involved. (ii) using non-parametric statistical techniques to identify the measurements that are not correctable, regardless of the used mapping function between the measured and accepted values.

Sensor measurements are inevitably subject to errors. One can identify two types of errors: (i) random fluctuations in data due to a noise in a sensor or in a sensed phenomenon, or (ii) gross errors - faults. A practical method to distinguish a random noise is to run maximum likelihood or Bayesian approach on the multi sensor fusion measurements. A random noise would exist, if running these procedures improves the accuracy of final results of multi-sensor fusion. While there have been several efforts to minimize random errors, very little has been done for fault detection. In multi-sensor fusion, the measurements from different sensors are combined in a model for consistent mapping of the sensed phenomena. Although the new fault detection technique is generic and can be applied to an arbitrary system of sensors that use an arbitrary type of data fusion, for the sake of brevity and clarity they focus on equation-based sensor fusion [Kou02].

Assume a set of sensors s_i ($0 \leq i \leq n$), each measuring a value x_i at a time t . The multimodal sensor fusion model equations are f_1, \dots, f_p are typically non-linear functions, and have the following forms: $f_j(x_1, x_2, \dots, x_n) = 0$, ($0 \leq j \leq p$). The system of equations is over-constraint. They solve the system $n+1$ times. First, they solve with all the equations in the original format and then, they ignore each variable and solve a least constrained system with $n-1$ variables (n times). They compare the values for each variable x_n in all $n+1$ scenarios. In order to improve accuracy of fault detection, the system can be

solved for m measurements by each sensor. At last, they conduct statistical analysis on the data for each sensor. If the obtained values for a sensor are not consistent within a confidence interval calculated by the percentile method [Efr82], that sensor is considered faulty.

7. FUTURE RESEARCH DIRECTIONS

In this section, we outline some of the potential future directions related to fault tolerance in sensor networks. It is well-known that it is very difficult to predict anything, in particular the future. Nevertheless, we believe that certain directions will inevitable attract a higher level of research interest, due to their intrinsic importance. We group future research directions in four groups. The first two are related to fault models and testing. The third one is related to resiliency mechanisms and the last one to analogies between fault tolerance and other domains such as power minimization and security.

The development of theoretically attractive and realistic fault models is one of the key prerequisites for the development of sound and real-life relevant fault tolerance techniques for sensor networks. While apart from fault tolerance models for components such as computation, storage and to serious extent communication are available, there is very little published in terms of fault models for sensors and actuators. At the same time, these components are the most important for overall system fault tolerance. The development of fault models for sensors will be in particularly difficult due to a great variety of their types, environments in which they will be deployed, and requirements in terms of fault tolerance of various applications. For example, it is clear that electromagnetic and mechanical based sensors will have very different fault characteristics than biological and chemical sensors. Also, sensors deployed in harsh environment such as nuclear plant will have very different characteristics than the ones deployed in friendly environments such as offices and residential areas. There is an intrinsic trade-off between more complex fault models and relatively simple ones. In the VLSI domain, only the simplest fault models, such as stuck at one and stuck at zero, have been extensively used. However, we expect that in addition to these models, more complex models will be required for sensors. In particular, it will be interesting to see what kind of fault models will be developed for sensors that can be used only once for reading. Note that these sensors are common in a number of biological and chemical sensors. In the VLSI domain, testing received significantly greater attention and application range than fault tolerance. Note that there is a need to address testing not just on the components and the individual node level, but also at the network and distributed system level.

Closely related to testing is calibration. Calibration can be defined as the process of mapping raw sensor data to a new set of data that is according to some statistical measure more accurate than the initial readings. Note that calibration can be done both off-line and on-line. In the former case, the emphasis will be on the accuracy and the strict interval of confidence, while in the latter case; the focus will be on the localized mode of operation. Also, there is a need to conduct not just calibration of sensor readings but also of other parameters that are relevant to the operation of sensor networks, including timing and the available energy level at each node. At the application level, we expect that fault resiliency mechanisms required for common applications such as sensor fusion, data filtering and data aggregation will be of primary importance. It is important to observe that for each of these applications there will be a significant variety of approaches that will be used. For example, in addition to equation based sensor fusion, we will see graphical based sensor fusion; statistics based sensor fusion and stochastic based sensor fusion. Each of these techniques has a number of unique peculiarities. While we do expect that specific fault resiliency techniques will be developed for each of them, we believe the primary importance will be on fault tolerance techniques that can be applied to multiple classes of approaches.

There is an interesting relationship between fault tolerance and several other fields. One of the main constraints in the deployment and the operation of the wireless sensor networks is the energy. The most effective way to prolong the lifetime of the network is to place a subset of the nodes in sleep mode. For example, power consumption of the software radio in three operational modes (Transmission, Receiving, and idle) rarely differs for more than a factor of two. At the same time, energy consumption in sleep mode is most often by two orders of magnitude, sometimes even more, lower. A simple and powerful observation is that a node in sleeping mode can be treated as faulty and vice versa. It will be possible to relatively easily retarget theoretical results and algorithms, and even software for one objective to the other. One of the major concerns is security and privacy. For example, the key question is to what extent one can trust the results obtained using sensor fusion or data aggregation in a particular scenario in a particular sensor network, under the assumption that one or more nodes are compromised. In order to study this question, one has to develop a threat model and models of attacks in one or more nodes in a sensor network. Now, again we can make a simple but important observation that these attacks can be modeled as the worst case fault models. Another interesting and important observation is that any technique that is resilient against non-intentional faults could also be retargeted to intentional faults.

8. CONCLUSION

Due to the potential deployment in uncontrolled and harsh environments and due to the complex arch, wireless sensor networks are and will be prone to a variety of malfunctioning. Our goal in this Chapter was to identify the most important types of faults, techniques for their detection and diagnosis, and to summarize the first techniques for ensuring efficiency of fault resiliency mechanisms. In addition to a comprehensive overview of fault tolerance techniques in general, and in particular in sensor networks, we discussed techniques that ensure fault resiliency during sensor fusion as well as the approach for heterogeneous built-in-self-repair fault tolerance. We concluded by outlining the potential future research directions along several dimensions.

ACKNOWLEDGEMENTS

This material is based upon work supported in part by the National Science Foundation under Grant No. ANI-0085773 and NSF CENS Grant.

REFERENCES

- [Aky02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cyirci, "Wireless Sensor Networks: A Survey." Computer Networks, vol. 38, no.4, pp. 393-422, 2002.
- [Avi84] A. Avizienis, J.P.J. Kelly, "*Fault Tolerance by Design Diversity: Concepts and Experiments*", IEEE Computer, vol. 17 no. 8, August 1984, 67-80.
- [Avi85] A. Avizienis, "*The N-Version Approach to FaultTolerant Software*," IEEE Trans. Soft. Eng., vol. SE-11, no.12, pp.1491-1501, 1985.
- [Bir85] K. Birman. "*Replication and fault-tolerance in the ISIS system*." In Proceedings of the 10th ACM Symposium on Operating Systems Principles, pp. 79-86, 1985.
- [Bir87] K. Birman, T. Joseph. "*Reliable communication in the presence of failures*." ACM Transactions on Computer Systems, 5:47--76, February 1987.
- [Bir94] K. Birman, R.V. Renesse, "*Reliable Distributed Computing with the Isis Toolkit*", IEEE Computer Society Press, 1994.
- [Bit88] D. Bitton and J. Gray. Disk shadowing. VLDB, pp 331-338, 1988.
- [Bro01a] A. Brown, D. A. Patterson. "*Embracing Failure: A Case for Recovery-Oriented Computing (ROC)*." 2001 High Performance Transaction Processing Symposium, Asilomar, CA, October 2001.

- [Bro01b] A. Brown, D. A. Patterson. “*To Err is Human.*” Proceedings of the First Workshop on Evaluating and Architecting System dependabilitY (EASY '01), Göteborg, Sweden, July 2001.
- [Bro02] A. Brown, D. A. Patterson. Rewind, Repair, Replay: “*Three R's to Dependability.*” 10th ACM SIGOPS European Workshop, Saint-Emilion, France, September 2002.
- [Bro96] R. R. Brooks, S. S. Iyengar, “Robust Distributed Computing and Sensing Algorithm.” IEEE Computer, vol. 25, no. 6, pp. 53-60, June 1996.
- [Byc03] V. Bychkovskiy, S. Megerian, D. Estrin, M. Potkonjak. “*Calibration: A Collaborative Approach to In-Place Sensor Calibration.*” 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03), pp. 301-316, April 2003.
- [Can02] G. Canea, J. Cutler, A. Fox, R. Doshi, P. Garg, R. Gowda. “*Reducing Recovery Time in a Small Recursively Restartable*” System. Proc. International Conference on Dependable Systems and Networks (DSN-2002), Washington, D.C., June 2002
- [Che01] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. “Span: An energy-efficient coordination algorithm for topology maintenance in ad-hoc wireless networks.” In ACM MobiCom, July 2001.
- [Che02] S. Chessa, P. Santi, “Crash Faults Identification in Wireless Sensor Networks”, Computer Communications, vol. 25, no. 14, pp. 1273-1282, Sept. 2002.
- [Dow64] R.W. Downing, J.S. Nowak, L.S. Tuomenoksa, “No 1 ESS Maintenance Plan.” Bell Sys. Tech. J. 43, no. 5, pt. 1, pp. 1961-2019, September 1964.
- [Eck96] D. Eckhardt, P. Steenkiste, “Measurement and Analysis of the Error Characteristics of an in Building Wireless Network.” In Proceedings of the SIGCOMM, pp. 243-254, 1996.
- [Eck98a] D. Eckhardt, P. Steenkiste, “A trace-based evaluation of adaptive error correction for a wireless local area network.” Journal on Special Topics in Mobile Networking and Applications (MONET), 1998.
- [Eck98b] D. Eckhardt, P. Steenkiste, “Improving Wireless LAN Performance via Adaptive Local Error Control.” INCP, 1998.
- [Efr82] B. Efron, The Jackknife, The Bootstrap, and Other Resampling Plans. S.I.A.M., Philadelphia, 1982.
- [Fra97] C. Fragouli, P. Lettieri, M. Srivastava, “Low Power Error Control for Wireless Links.” ACM/IEEE MobiCom, pp. 139-150, 1997.

- [Gra93] J. Gray and A. Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, San Mateo, California, 1993.
- [Gue93] L. Guerra, M. Potkonjak, J.M. Rabaey, "High Level Synthesis Techniques for Efficient Built-In-Self-Repair." Intl. Workshop on DFT in VLSI Syst., pp. 41-48, 1993.
- [Int00] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks." ACM/IEEE MobiCom, pp. 56-67, 2000.
- [Jal94] P. Jalote. Fault Tolerance in Distributed Systems. P.T.R Prentice Hall, Englewood Cliffs, NJ 07632, 1994.
- [Kou02] F. Koushanfar, M. Potkonjak, A. Sangiovanni-Vincentelli, "Fault Tolerance in Wireless Ad-Hoc Sensor Networks." IEEE Sensors, vol. 2, pp. 1491-1496, June 2002.
- [Kou03] F. Koushanfar, M. Potkonjak, A. Sangiovanni-Vincentelli, "Online Fault Detection in Wireless Sensor Networks." IEEE Sensors, 2003, to appear.
- [Kou03b] F. Koushanfar, A. Davare, D. T. Nguyen, M. Potkonjak, A. Sangiovanni-Vincentelli, "Low Power Coordination in Wireless Ad-hoc Networks." ISLPED, Aug 2003.
- [Lac95] R. Lackey, D. Upmal, "Speakeasy: the Military Software Radio." IEEE Communications Magazine, vol. 33, no.5, pp. 56-61, May 1995.
- [Lam82] L. Lamport, R. Shostak, and M. Pease. "*The Byzantine generals problem.*" ACM Transactions on Programming Languages and Systems, vol. 4, no. 3, pp.382-401, July 1982.
- [Lam83] L. Lamport, "*The weak Byzantine generals problem,*" J. ACM, vol. 30, pp. 668--676, July 1983.
- [Lee90] P. A. Lee and T. Anderson. Fault Tolerance: Principles and Practice, Second Edition, Springer-Verlag, 1990.
- [Man99] N. Mandayam, "A Software Radio Architecture for Linear Multiuser Detection." IEEE JSAC, vol. 17, no. 5, pp. 814-823, May 1999.
- [Mit95] J. Mitola, "The software radio architecture." IEEE Communications Magazine, May 1995.
- [Mit99] J. Mitola, "Technical Challenges in the Globalization of Software Radio." IEEE Communications Magazine, February 1999.
- [Moo56] E. F. Moore, C.E. Shannon, "Reliable Circuits Using Less Reliable Relays." J. Franklin institute, vol. 262, pp. 191-208, September 1956.

- [Par97] V. D. Park, M. S. Caron, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks." Infocom 1997.
- [Pea80] M. Pease, R. Shostak, and L. Lamport. "*Reaching Agreement in the Presence of Faults.*" Journal of ACM, vol. 27, no. 2, pp. 228-234, 1980.
- [Pra91] L. Prasad, S. S. Iyengar, R. L. Kashayap, R. N. Madan, "Functional Characterization of Fault Tolerant Interaction in Distributed Sensor Network." Physical Review E, vol. 49, no.2, April 1994.
- [Sch84] F. B. Schneider, "*Byzantine Generals in Action: Implementing Fail-Stop Processors*", ACM Transactions on Computer Systems, vol. 2, no.2, pp.145-154, May 1984.
- [Tut99] W. H.W. Tuttlebee, "Software-Defined Radio: Facets of a Developing Technology," IEEE Personal Communications Magazine, vol. 6, no. 2, pp. 38--44, April 1999.
- [von56] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components." In C. E. Shannon and J. McCarthy eds. Automata studies, Princeton university press, 1956, pp. 43-98.
- [Whi02] K. Whitehouse, D. Culler, "Calibration as Parameter Estimation in Sensor Networks." ACM WSNA, 2002.