# RESILIENT DATA AGGREGATION IN WIRELESS SENSOR NETWORKS

*A THESIS*

*submitted by*

**SAI THARUN REDDY T**

*for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**


**DEPARTMENT OF ELECTRICAL ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY MADRAS**

**JUNE 2014**

# THESIS CERTIFICATE

This is to certify that the thesis titled **RESILIENT DATA AGGREGATION IN WIRELESS SENSOR NETWORKS**, submitted by **Sai Tharun Reddy T**, to the Indian Institute of Technology Madras, Chennai for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

**Dr. Sheetal Kalyani**
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras 600 036

Date: June 2014
Place: Chennai

# ACKNOWLEDGEMENTS

Special Thanks to Dr. Sheetal Kalyani, Asst Professor at Dept. of Electrical Engineering; IIT Madras for providing the opportunity to work on this project and for valuable guidance to make this project successful. Thanks to Mr. Sai Shankar K P, Research Scholar at Dept. of Electrical Engineering; IIT Madras for participating in our discussion and providing valuable inputs for progress of our project. Special thanks to Late John Hunter, the author of *matplotlib* python libray, Travis Oliphant, the author of *numpy* python library and several other developers from the open source community for their contributions which helped us in simulating and analyzing our proposed methods.

# ABSTRACT

KEYWORDS: Wireless sensor networks, tree topology, distributed aggregation, System monitoring modules, Intrusion detection, and Outlier detection

Wireless sensor networks offer the potential to span and monitor large geographical areas inexpensively. Sensors, however, have significant power constraint (battery life), making communication very expensive. Another important issue in the context of sensor based information systems is that individual sensor readings are inherently unreliable. Design of reliable wireless sensor network (WSN) need to address the failure of single or multiple network components and implementation of the techniques to tolerate the faults occurred at various levels. The issues and requirements of reliability improvement mechanism depend on the available resources and application for which the WSN is deployed. The objective is to design the optimal selection of nodes for sensing and scheduling for intrusion detection such that the long term network cost incurred by all components is minimized. As part we have looked at using simple data mining and machine learning techniques like outlier detection; jointly considering data aggregation, information trust, and fault tolerance to enhance resilient distributed data aggregation in WSNs. We consider a large-scale cyber network with N nodes. Each node is either in a healthy state (1) or an abnormal state (0). Due to random intrusions, the state of each component transits from 1 to 0 over time. At each time, a subset of K ($K < N$) components are checked and those observed in abnormal states are not included in the data record while performing aggregation. We propose different approaches for making these aggregation schemes more resilient against certain attacks. As a result, we identify that median is more robust as suggested by many, and also effective in outlier elimination than mean. We propose a method to locally select a set of reliable nodes for sampling considering the cost or weightage, reliability factor and total energy available with the node, in order to increase the life time of the network and improve resilience.

# TABLE OF CONTENTS

## LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Recently there has been growing interest in providing fine-grained metering and control of living environments using low power devices. Wireless Sensor Networks (WSNs), which consist of spatially distributed self-configurable sensors, perfectly meet the requirement. The sensors provide the ability to monitor physical or environmental conditions, such as temperature, humidity, vibration, pressure, sound, motion and etc., with very low energy consumption. The development of WSNs was motivated by military applications such as battlefield surveillance; WSNs are widely used in industrial environments, residential environments and wildlife environments. With the rapid development of wireless technology, more and more people prefer to use the wireless network as their end-user network. Compared with the traditional wireless network, WSN has its own features, such as low cost and low energy consumption. They more strongly resemble embedded systems, for two reasons. First, wireless sensor networks are typically deployed with a particular application in mind, rather than as a general platform. Second, a need for low costs and low power leads most wireless sensor nodes to have low-power micro-controllers ensuring that mechanisms such as virtual memory are either unnecessary or too expensive to implement.

One major challenge in a WSN is to produce low cost and tiny sensor nodes. There are an increasing number of small companies producing WSN hardware and the commercial situation can be compared to home computing in the 1970s. Many of the nodes are still in the research and development stage, particularly their software. Also inherent in sensor network adoption is the use of very low power methods for data acquisition. To reduce cost, each sensor board has very limited on-board resources, such as computing speed, storage and energy source. Energy is the scarcest resource of WSN nodes, and it determines their lifetime. WSNs are meant to be deployed in large numbers in various environments, including remote and hostile regions, where ad hoc communications are a key component. For this reason, algorithms and protocols need to address the issues of

Lifetime maximization, Robustness and fault tolerance, and Self-configuration. Technologies enabling WSNs should be concerned of cost minimization, miniaturization and energy scavenging i.e. to recharge batteries from ambient energy sources like light, vibration etc.

Sensor networks have been proposed for scientific data collection, environmental monitoring, building health monitoring, burglar and fire alarm systems, and many other applications involving distributed interaction with the physical environment. Many of these applications involve a distributed system of sensors measuring the environment from many vantage points and then somehow aggregating the collected data to form a global summary view that can be acted upon. But a common problem in such networks is information error and loss caused by component failure, external interference, wireless transmission error, and security threats such as fake data injection [7]. Due to lower reliability and hence the accuracy of the data sensed by individual sensor nodes, collaboration among sensors is necessary for reliable event detection and prevention of faulty or fake reports. For event detection with multi-mode collaboration, the common method is data aggregation, which is leveraged not only to reduce the throughput of data transmission, thus saving energy effectively, but also to enhance the accuracy of event detection and avoid interference of the compromised nodes. Consequently, data aggregation can be viewed as an important building block in sensor networks. For wireless sensor networks (WSNs) deployed in the noisy and unattended environments, it is necessary to establish a comprehensive framework that protects the accuracy of the gathered information.

As security has been identified as a major challenge for sensor networks, recent protocols have been designed with security in mind to be able to make the aggregation resilient from attacks. When a single sensor node can be captured, compromised, or spoofed, an attacker can often manipulate the result of the aggregation operation without limit, gaining complete control over the computed aggregate which is undesirable. For instance, it is shown that any protocol that computes the average, sum, minimum, or maximum function is insecure against malicious data, no matter how these functions are computed. In response to this threat, David Wagner introduced a theoretical framework for modeling

the security of data aggregation [1]. This model insists that the aggregation function must be resilient in the presence of arbitrary changes to a small subset of sensor observations, and thus he coin the term *resilient aggregation* to refer to schemes that satisfy this condition. Wagner also introduces the concept of the breakdown point from robust statistics in the context of resilient aggregation. He introduces several techniques and principles for achieving resilient aggregation in new protocols. Wagner also recommended the use of outlier filtering techniques such as truncation and trimming for achieving resilient aggregation. Outliers can be detected and filtered out in a hop-by-hop fashion or at the base station after the sensor readings reach the base station.

If communication overhead is not a concern, we can send readings from all nodes to a base station for decent aggregation. But, exhaustively forwarding the exact data value of every node is extremely communication-intensive. Node Congestion measures how quickly the heaviest loaded nodes will exhaust. But, we require the nodes closer to Base Station not to exhaust, which in general are expected to have more communication load. Now we can understand the problem as a trade-off between 'Low communication loads on the heaviest loaded nodes' vs 'Large amount of in-network communication'.

As part of our work, we have looked at using simple data mining and machine learning techniques; jointly considering data aggregation, information trust, and fault tolerance to enhance resilient distributed data aggregation in WSNs. In later parts of Chapter 1, we look at typical WSNs in great detail, understanding the issues to be addressed for resilience. In Chapter 2, information about prior research work from which our work is adapted or motivated is provided. In Chapter 3, we define the models and detail assumptions of the network, aggregation scheme and attacker. In Chapter 4, we define the problem and outline our proposed method to address the same, propose a method to compute reliabilities and respective reliability factors at each node and also explain how we have adapted the Q-digest algorithm for distributed median computation. In Chapter 5, we compare different approaches using results obtained from simulations.

### 1.1. Typical Structure of WSNs

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices that use sensors to monitor physical or environmental conditions. These autonomous devices, known as routers and end nodes, combine with a gateway to create a typical WSN system. A typical WSN is built of several hundreds or even thousands of "sensor nodes". Each node has the ability to communicate with every other node wirelessly, thus a typical sensor node has several components: a radio transceiver with an antenna which has the ability to send or receive packets, a microcontroller which could process the data and schedule relative tasks, several kinds of sensors sensing the environment data, and batteries providing energy supply. The distributed measurement nodes communicate wirelessly to a central gateway, which acts as the network coordinator in charge of node authentication, message buffering, and bridging from the wireless network to the wired Ethernet network, where you can collect, process, analyze, and present your measurement data. A WSN consists of three main components: nodes, gateways, and software. The spatially distributed measurement nodes interface with sensors to monitor assets or their environment. . The sensors also have the ability to transmit and forward sensing data to the base station. Most modern WSNs are bi-directional, enabling two-way communication, which could collect sensing data from sensors to the base station as well as disseminate commands from the base station to end sensors. The acquired data wirelessly transmit to the gateway, which can operate independently or connect to a host system where you can collect, process, analyze, and present your measurement data using software. Routers are a special type of measurement node that you can use to extend WSN distance and reliability.

### 1.2. Architecture of WSNs

Network architectures for ad-hoc networks are in principle, relatively straightforward and similar to standard networks. Mobility is compensated for by appropriate protocols, but interaction paradigms don't change much. WSNs, on the other hand, look quite different on many levels. The topology of WSNs can vary among star network, tree network and mesh network. The need and the possibility to manipulate data as it travels through the network open new possibilities for protocol design. If a centralized architecture is used in a sensor network and the central node fails, then the entire network will collapse,

however the reliability of the sensor network can be increased by using distributed control architecture. There is also no centralized body to allocate the resources and they have to be self-organized. Characteristic requirements for WSNs will be Quality of service, fault tolerance, life-time, scalability, programmability and maintainability. The required mechanisms to meet these requirements are Multi-hop wireless communication, Energy-efficient operation, both for communication and computation, Auto-configuration, Collaboration & in-network processing (i.e. Nodes in the network, collaborate towards a joint goal with pre-processing data in the network), Data centric networking, Locality (i.e. To do things locally, on the node or among nearby neighbors, as far as possible).

In typical networks (including ad hoc networks), network transactions are addressed to the identities of specific nodes. In a redundantly deployed sensor network, specific source of an event, alarm, etc. might not be important. Thus, the network design must focus networking transactions on the data directly instead of their senders and transmitters, which can be understood as *"data-centric networking"*. Whereas energy-efficient networking protocols use multi-hop routes for low energy consumption (energy/bit) and take available battery capacity of devices into account. In many sensor applications, the data collected from individual nodes is aggregated at a base station or host computer. To reduce energy consumption, many systems also perform in-network aggregation of sensor data at intermediate nodes en route to the base station. Most existing aggregation algorithms and systems do not include any provisions for security, and consequently these systems are vulnerable to a wide variety of attacks. In particular, compromised nodes can be used to inject false data that lead to incorrect aggregates being computed at the base station. The data gathered from wireless sensor networks is usually saved in the form of numerical data in a central base station. Additionally, the Open Geospatial Consortium (OGC) is specifying standards for interoperability interfaces and metadata encodings that enable real time integration of heterogeneous sensor webs into the Internet, allowing any individual to monitor or control Wireless Sensor Networks through a Web Browser.

In-network processing [8] reduces communication overhead by applying an aggregation function in the network, but at the cost of security. Because end to end authentication can be a problem for applying popular in-network aggregation functions. Another important aspect to be considered in the design is to exploit *temporal and spatial correlation.* To reduce communication costs some algorithms remove or reduce nodes' redundant sensor information and avoid forwarding data that is of no use. As nodes can inspect the data they forward, they can measure averages or directionality for example of readings from other nodes. For example, in sensing and monitoring applications, it is generally the case that neighboring sensor nodes monitoring an environmental feature typically register similar values. This kind of data redundancy due to the spatial correlation between sensor observations inspires techniques for in-network data aggregation and mining. Depending on application, more sophisticated processing of data can take place within the network. Observed signals might vary only slowly in time and hence no need to transmit all data at full rate all the time. Signals of neighboring nodes are often quite similar, which implies its enough to transmit differences. We can also exploit the location information, activity patterns, heterogeneity etc. for improved security and efficiency. In the following sections we look at how different protocols have been designed considering these aspects to address one or many problems with WSNs.

### 1.3. Data Acquisition Systems

Data acquisition system [2, 3] for sensor networks can be classified into two broad categories on the basis of the data collection methodology employed for the application:

1. **Query-based systems:** In query-based systems, the base station (the data sink) broadcasts a query to the network and the nodes respond with the relevant information. Messages from individual nodes are potentially aggregated en route to the base station. Finally, the base station computes one or more aggregate values based on the messages it has received. In some applications, queries may be persistent in nature resulting in a continuous stream of data being relayed to the data sink from the nodes in the network. For such applications, the query broadcast by the base station specifies a period; nodes in the network send their readings to the base station after each epoch.

**2. Event-based systems:** In event-based applications, such as perimeter surveillance and biological hazard detection, nodes send a message to the base station only when the target event occurs in the area of interest. If multiple reports being relayed correspond to the same event, they can be combined by an intermediate node on the route to the base station.

Data acquisition systems can also be categorized based on how sensor data is aggregated. In single-aggregator approaches, aggregation is performed only at the data sink. In contrast, hierarchical aggregation approaches make use of in-network aggregation. Hierarchical aggregation schemes can be further classified into tree-based schemes and ring-based schemes on the basis of the topology in which nodes are organized. However, most existing data management and acquisition systems for sensor networks are vulnerable to security attacks launched by malicious parties.

### 1.4. Applications of WSNs

Wireless sensor networks (WSNs) are commonly used in pervasive and ubiquitous applications. WSNs are developed using both static (motes) and mobile (e.g. Smart phone) sensor nodes for various applications such as smart homes, tele health, surveillance, metering, and industry automation [4, 5, and 6]. In these applications, the data collected by sensor nodes from their physical environment need to be assembled at a host computer or data sink for further analysis. Typically, an aggregate (or summarized) value is computed at the data sink by applying the corresponding aggregate function to the collected data. Interaction patterns between sources and sinks classify application types as follows:

1. Event detection: Nodes locally detect events and report these events to interested sinks. Event classification is an additional option.
2. Periodic measurement
3. Function approximation: Use sensor network to approximate a function of space and/or time (e.g., temperature map).
4. Edge detection: Find edges (or other structures) in such a function.
5. Tracking: Report (or at least, know) position of an observed intruder.

### 1.5. Possible threat models for WSNs

Sensor nodes are often deployed in unattended environments, so they are vulnerable to physical tampering. Since current sensor nodes lack hardware support for tamper resistance, it is relatively easy for an adversary to compromise a node without being detected. The adversary can obtain confidential information (e.g., cryptographic keys) from the compromised sensor and reprogram it with malicious code. Moreover, the attacker can replicate the compromised node and deploy the replicas at various strategic locations in the network. A compromised node can be used to launch a variety of security attacks. These attacks include jamming at the physical or link layer as well as other resource consumption attacks at higher layers of the network software. Compromised nodes can also be used to disrupt routing protocols, and topology maintenance protocols that are critical to the operation of the network.

However, we focus on attacks that target the data acquisition protocol used by the application. Specifically, we discuss attacks in which the compromised nodes send malicious data in response to a query (or send false event reports in event-based systems). By using a few compromised nodes to render suspect the data collected at the sink, an adversary can effectively compromise the integrity and trustworthiness of the entire sensor network. In event-based systems, compromised nodes can be used to send false event reports to the base station with the goal of raising false alarms and depleting the energy resources of the nodes in the network. We refer to this attack as the false data injection attack. Similarly, in query-based systems, compromised nodes can be used to inject false data into the network with the goal of introducing a large error in the aggregate value computed at the data sink. The aggregate computed by the sink is erroneous in the sense that it differs from the true value that would have been computed if there were no false data values included in the computation. Unlike event-based systems, however, in aggregation systems the effectiveness of a false data injection attack depends on both the aggregate being computed and whether sensor data is aggregated en route to the data sink or only at the data sink, and thus the techniques used for preventing this attack differ from the techniques used in event-based systems.

In an aggregation system, a compromised node N can corrupt the aggregate value computed at the sink in four ways. First, N can simply drop aggregation messages that it is supposed to relay towards the sink. This has the effect of omitting a large fraction of sensor readings being aggregated. Second, N can alter a message that it is relaying to the data sink. Third, N can falsify its own sensor reading with the goal of influencing the aggregate value. Fourth, in systems that use in-network aggregation, N can falsify the sub-aggregate which it is supposed to compute based on the messages received from its child nodes. The input falsification cannot be addressed unless we have prior knowledge about the distribution of sensor readings.

The first attack in which a compromised node intentionally drops aggregation messages can substantially deviate final estimate of the aggregate if tree-based aggregation algorithms are used. The deviation will be large if the compromised node is located near the root of the aggregation hierarchy because a large fraction of sensor readings will be omitted from being aggregated. Countermeasures against this attack include the use of multi-path routing and ring-based topologies [9] as well as the use of probabilistic techniques in the formation of aggregation hierarchies [10]. To prevent the second attack in which a compromised node alters a message being relayed to the sink, it is necessary for each message to include a message authentication code (MAC) generated using a key shared exclusively between the originating node and the sink. This MAC enables the sink to check the integrity of a message, and filter out messages that have been altered. Hence, the effect of altering a message is no different from dropping it, and countermeasures such as multi-path routing are needed to mitigate the effect of this attack. We refer to the third attack in which a sensor intentionally falsifies its own reading as the falsified local value attack. This attack is similar to the behavior of nodes with faulty sensors, and also to the false data injection attack in event-based systems. Potential countermeasures to this attack include approaches used for fault tolerance such as majority voting and reputation-based frameworks [11, 12]. The three attacks discussed above apply to both single-aggregator and hierarchical aggregation systems, whereas the fourth attack applies only to hierarchical aggregation systems. This attack in which a node falsifies the aggregate value it is relaying to its parent(s) in the hierarchy is much more difficult to address. We refer to this attack as the falsified sub-aggregate attack.

# CHAPTER 2

# BACKGROUND

Data resilience refers to the ability of a network storing the data to recover quickly and to continue maintaining availability of data despite of any disruptions (such as equipment failure, power outage, or even malicious attack). Due to resource constraints of sensor networks such as non-renewable battery power and limited storage capacity of sensor nodes, link unreliability and scarce bandwidth of the wireless medium, and the inhospitable and harsh environments in which they are deployed, sensor nodes are often prone to failure and vulnerable of data loss. Therefore, how to ensure that data collected at sensor nodes reach the base station reliably despite all the vulnerabilities has been an active research topic since the inception of sensor network research.

The existing literature on secure aggregation can be broadly divided into two categories: the first category uses Data Authentication (verifiable sampling) to provide resilient probabilistic estimates of the aggregate result; the second category uses commitment verification, which, unlike the first category, can provide highly precise results for which any malicious tampering is immediately evident, but at the cost of availability.

1. **Data Authentication:** One-hop verification [13] and witness based verification provide good examples of the necessity of using authentication primitives such as MACs to authenticate the origin of each input and sub-result. Without such authentication, for example, an adversary could insert many false input elements and untraceably skew the aggregation result. Though One-hop verification in a tree topology works well for the single node attacker, it requires all the values to be forwarded to the parent and fails in case the child parent pair is attacked. However, it secures data from a highly powerful attacker who can observe and manipulate everything at a single node. The idea of witness based verification is to build in redundancy into the system, by including multiple redundant aggregators or witnesses which attest the aggregate reported by primary aggregator. In this sense, they are performing a pure validation function, and are hence called witnesses to the authenticity of the aggregation report.

2. **Commitment and Verification:** Each aggregator node commits to a set of inputs and a result of the computation over these inputs, which is then verified through a process of having redundant elements repeating the computation. For the one-hop verification technique, the redundant verifier is the parent of each aggregator node; for the witness-based verification scheme, the aggregation operation is replicated over several redundant aggregator nodes. This pattern of commitment and subsequent verification is the basis of more sophisticated schemes, like SIA (Secure Information Aggregation [14]). The idea of the SIA is to enable the user to verify that the answer given by the aggregator in response to a query is a good approximation of the true value even when the aggregator and some fraction of the sensor nodes are corrupted.

The basic approaches of one-hop verification and witness-based verification have a clear drawback: only a small, fixed number of verifiers need to be compromised in order for the adversary to completely break the security of the system. Despite these disadvantages, these basic schemes nonetheless have the advantages of being relatively simple to implement, and are relatively efficient for low threat applications. A number of additional extensions to One-hop verification scheme are proposed by Jadia and Mathuria [15] to provide secrecy of the data from a single in-network attacker.

Cristofaro et al [16] generalize the witness based approach to hierarchical aggregation, using multiple redundant aggregator nodes per level. They use a heuristic metric called "quality of information" (QoI) to measure the level of confidence on a given sub-aggregation result. This QoI metric is then propagated in the hierarchical aggregation computation process. In the end, the base station is presented with both an aggregation result and a measure of confidence on the result, which the base station can use as evidence to accept or reject the final aggregation result. A related approach similar to this QoI metric-based scheme is reputation based integrity checking [17], where nodes evaluate trustworthiness of other nodes based on their past behavior, and determines whether to accept a received aggregate based on the sender's trustworthiness level. Based on the multilayer aggregation architecture of WMSNs, Sun et al [18] designed a trust based framework for data aggregation with fault tolerance with a goal to reduce the

impact of erroneous data and provide measurable trustworthiness for aggregated results. By extracting statistical characteristics from different sources and extending Josang's trust model, they propose how to compute self-data trust opinion, peer node trust opinion, and peer data trust opinion.

There is another class of approaches in which accuracy of the aggregation computation is sacrificed in return for making the protocol resilient to malicious manipulation. Specifically, these approaches typically compute a statistical estimator for the aggregate value; while using the estimator is less accurate that computing the precise aggregate, it is easier to secure, because there is less reliance on the honest behavior of intermediate aggregators. Each of these algorithms [13, 19, 20 and 21] essentially selects a subset of the nodes and asks for a "representative member", a node that signifies that the set has at least one node satisfying the predicate. The representative member then returns an authenticated report attesting to this fact. An estimate of the count of nodes satisfying the predicate can then be calculated by appropriate choice of the sampled subsets.

Data aggregation, as a primitive communication task in wireless networks, can reduce the communication complexity. However, in-network aggregation usually brings an unavoidable security defect. Some malicious nodes may control a large percentage of the whole network data and compel the network misbehave in an arbitrary manner. Thus, locating the malicious nodes to prevent them from further disaster is a practical challenge for data aggregation schemes. Based on the grouping and localization techniques, Xu et al [46] proposed a novel integrated protocol to locate malicious nodes. The proposed protocol does not rely on any special hardware and requests only incomplete information of the network from the security schemes. The model is to perform Grubb's test [34] to detect group with malicious nodes and then to detect the malicious nodes within the group using area based approach to perform local estimation. Scaffer et al [22] present a statistical framework, *CORA: Correlation based Resilient Aggregation*, which is designed to mitigate the effects of an attacker who is able to alter the values of the measured parameters of the environment around some of the sensor nodes. Their proposed framework takes advantage of the naturally existing correlation between the sample elements, which is very rarely considered in other sensor network related papers.

The algorithms presented are to be applied without assumption on the sensor network's sampling distribution or on the behavior of the attacker, which otherwise is a very big assumption to make. Though there are some very efficient algorithms designed making assumptions about behavior of the attacker which are highly context specific. Sun et al [23] are the first to propose integration of system monitoring modules and intrusion detection modules in the context of WSNs. They propose an Extended Kalmann Filter (EKF) based mechanism to detect false injected data. Specifically, by monitoring behaviors of its neighbors and using EKF to predict their future states (actual in-network aggregated values), each node aims at setting up a normal range of the neighbors' future transmitted aggregated values. Using different aggregation functions they present how to obtain a theoretical threshold and further apply an algorithm of combining Cumulative Summation and Generalized Likelihood Ratio to increase detection sensitivity, illustrating how the proposed local detection approaches work together with the system monitoring module to differentiate between malicious events and emergency events. Secure tree-based aggregation protocols [24 and 25] remain vulnerable to message losses either due to node failure or compromised nodes. The performance and security tradeoffs between resilient tree-based approaches and multipath approaches such as Attack Resilient Synopsis Diffusion [26] have yet to be explored.

Distributed consensus is a common objective in WSNs, to make applications in various areas such as data aggregation [27], distributed optimization [28], and flocking [29], resilient to presence of malicious nodes. Our objective being to perform resilient data aggregation in distributed fashion and as we identified before that median is the most robust and light weight aggregate function, we adapt ideas from the model of Median Consensus Algorithm proposed by Zhang et al [30]. They design a consensus algorithm where, at each time-step, each node updates its value as a weighted average of its own value and the median of its neighbors' values. This algorithm requires no global information about the network, and is computationally lightweight. Roy et al [31] propose a protocol to compute an approximate Median and verify if it has been falsified by an adversary and design an attack-resilient algorithm to compute the Median even in the presence of a few compromised nodes. The entire network of nodes is divided into number of groups and each group is randomly sampled to collect a set of samples to

compute median. Once the base station collected all the samples it sorts them in ascending order and plots a histogram to estimate median while histogram verification is done based on the aggregation tree which essentially checks whether sum of counts in each bucket is equal to the total number of nodes. Shrivastava et al [32] proposed a data aggregation scheme that significantly extends the class of queries that can be answered using sensor networks. These queries include (approximate) quantiles, such as the median, the most frequent data values, such as the consensus value, a histogram of the data distribution, as well as range queries while each sensor aggregates the data it has received from other sensors into a fixed (user specified) size message. Aggregation scheme we proposed uses a modified version of this q-digest algorithm which is known for its accuracy, scalability and low resource utilization for highly variable input data sets.

However, only a few protocols consider secure in-network aggregation based on a prevention-based scheme, in which encryption, authentication, and key management are used. Once a sensor node is compromised, all its associated secrets become open to attackers, rendering prevention-based techniques helpless. To solve this problem, intrusion detection systems (IDSs), which serve as the second wall of protection, can effectively help identify malicious activities. Unfortunately, there is very little work that aims at addressing the secure in-network aggregation problem from an intrusion detection perspective so far. Now we understand that, to achieve resilience to the aggregation function, intrusion detection systems (IDS) are important part of the system. The objective of IDS is to locate malicious activities (e.g., denial of service attack, port scans, hackers etc.) in the quickest way such that the infected parts can be timely fixed to minimize the overall damage to the network. With the increasing size, diversity, and interconnectivity of the cyber system, however, intrusion detection faces the challenge of scalability: how to rapidly locate intrusions and anomalies in a large dynamic network with limited resources. The two basic approaches to intrusion detection, namely, active probing and passive monitoring [35], [36], face stringent resource constraints when the network is large and dynamic. Hence our objective is to design the optimal scheduling for intrusion detection such that the long-term network cost incurred by all abnormal components is minimized. In [37], the problem of intrusion recognition by classifying

system patterns was addressed based on data mining. Without resource constraint, the focus is on the best selection of system features to detect intrusion from the accessible system data statistics. Similar problems of statistical modeling of data and detection algorithms under various scenarios were considered in a number of works [38, 39, 40 and 41]. Datta et al [42] discussed the importance of distributed data mining and explains how it can be used for local approximation by distributed implementation of K-means clustering. These studies mainly address the intrusion detection problem from a machine learning or pattern recognition perspective and do not consider the constraint on the system monitoring capacity.

Our work is a stochastic control approach for intrusion detection in large networks with resource constraints, where the problem of how to adaptively allocate the limited resources for performance optimization and life time maximization is of great interest. Liu and Zhao [23] formulate the problem as a special class of Restless Multi-Armed Bandit (RMAB) process and show that for this class of RMAB, Whittle index [43] exists and can be obtained in closed form, leading to a low-complexity implementation of Whittle index policy with a strong performance. They consider a strong average-reward criterion under which not only the steady-state average reward but also the transient reward starting from an arbitrary initial arm state is maximized, leading to the maximum long-term total reward growth rate. However to compute the Whittle index in closed form, we require to have knowledge of the probability with which each component is attacked, which is a serious assumption to make. Hence we propose a more generalized approach to identify more reliable nodes and selectively sampling the nodes increasing the life time of WSN.

# CHAPTER 3

# MODELS AND ASSUMPTIONS

## 3.1. Model for WSN

WSNs are often deployed to monitor emergency events like forest fire. We assume that the majority of nodes around some unusual events are not compromised. In anomaly based detection, the normal system behavior is defined as the behavior of the majority of nodes (or similarly, the behavior of the system in the majority of its operational time). Note that the redundancy in sensor deployment is also one of the important features in many WSNs. Therefore, nodes are often densely deployed. These conditions can help sensor nodes make an accurate decision. We also assume that falsified data transmitted by a compromised node is significantly different from the actual value so that falsified data can effectively disrupt aggregation operations. If falsified data sent out by compromised nodes are only slightly different from true aggregated values, an attacker cannot cause significant impact on deployed applications. We do not assume time synchronization among nodes. Our proposed approach can tolerate the time inaccuracy caused by children nodes and parent nodes. In the context of WSNs, time synchronization still incurs expensive operations.

Since the communication range of the nodes is limited, the sink will generally not be in range of all the sensors. Therefore, the information has to be relayed from sources to the sink by means of intermediate nodes. We will assume that each sensor node has a secure channel back to the base station for reporting data measurements. Moreover, we assume these secure channels are independent: capture of one sensor node might compromise the contents of that node's channel to the base station, but it will not reveal anything about other nodes' channels. For instance, each sensor might share a per-node symmetric key with the base station and use this key to encrypt its data. As a consequence of these assumptions, we do not need to worry about interception of data in transit. This leaves only the question of whether the endpoints are trustworthy or not.

Let us assume a general multi-hop network with a set of N sensor nodes and a single base station, BS. Each of these sensor nodes are assumed to have a maximum storage limit of energy which is full at the time of deployment and gain energy from ambient sources like solar or vibration at a constant rate. The BS knows the IDs of the sensor nodes present in the network. The network user controls the BS and initiates the query. We also consider that sensor nodes are similar to the current generation of sensor nodes in their computational and communication capabilities and power resources, while the BS is a laptop-class device supplied with long-lasting power. And let this network be deployed in a vast two-dimensional geographical region. The positions of sensors are uniformly and independently distributed in the region.

We can model the network as a connected graph with N vertices. There is a special vertex BS which denotes the base station, and each of the remaining vertices each represents a sensor node. We draw an edge between every pair of vertices that correspond to sensor nodes within mutual transmission/reception range of each other. Any algorithm can be used to design the routing tree, as long as (i) it allows the data to flow in both directions of the tree, and (ii) it avoids sending duplicates. Typically, the base station broadcasts the aggregation query message throughout the network and aggregation tree is constructed then, if it does not already exist. We propose a tree building algorithm using the reliability factors at each node, which is explained in the later sections. It is advised to use different node as the root node in order to avoid node congestion and also making the network resilient to powerful attackers comprising certain set of nodes, but for our purpose we use a fixed root node and a single base station. At each discrete time, each component is either in the healthy state (1) or the abnormal state (0), and, to identify the state of each node, we use mean/median based outlier detection methods which are explained in the later sections.

### 3.2. Aggregation Scheme

The distributed aggregation problem is the problem of computing a given function over all the readings, such that the base station BS receives the correct value of aggregate. Assume an aggregation service, e.g. TAG by UC Berkley [44, 45], which aims at aggregating the data within the network in a time- and energy-efficient manner. The goal

of the aggregation service is to minimize the amount of time spent by sensors in powering their different components and to maximize the time spent in the idle mode, in which all electronic components are switched off. Indeed, the energy consumption is several orders of magnitude lower in the idle mode than in a mode where the CPU or the radio is active. Once a routing tree is set up and the nodes synchronized, data can be aggregated along the routing tree, from the leaves to the root. Each node includes its contribution in a partial state record X which is propagated along the routing tree. Partial state records may be any data structures. Partial state records are merged when two (or more) of them arrive at the same node. When the eventual partial state record is delivered by the root node to the base station, the desired result is obtained. Note that without this aggregation process, all the measurements would be routed to the base station. The root node would therefore have to send N packets per epoch. Instead, using our aggregation scheme, each node is required to send only a few pieces of data. In our work, we have not taken into account the effect of lost messages. The effect of lost messages can be mitigated to some extent in a continuous query setting where the state record is continuously updated.

In a hierarchical environment monitoring system, sensor nodes collect the environment signals according to a certain sampling mechanism and report them to higher level sensor nodes (called aggregators). An aggregator and its direct children form an aggregation set. The aggregators forward the aggregated results to their higher level aggregator recursively, and eventually to the sink node. Since the nodes participating in the process may be destroyed or the data may be manipulated, the aggregators and the sink node should have a mechanism to provide trustworthiness of aggregated results to the users. Consider a temperature monitoring network as an example. When the sensing part of a sensor node fails suddenly, the sampling value collected by the sensor changes accordingly, the node itself can judge the data abnormal immediately based on temporal correlation. But, if the sensed data from this node maintain the abnormal value from then on, as time goes by, this node may treat its reports as normal. In this case, comparing the reports from this sensor, with those of its neighbors, the aggregator should determine the trustworthiness of this sensor still low and thus reduce the weight (contribution) to the aggregation process. In any set of data it is important to be on the lookout for outliers.

Outliers are individual values that fall outside of the overall pattern of the rest of the data. This definition is somewhat vague and subjective, so it is helpful to have a rule to help in considering if a data point truly is an outlier.

### 3.3. Attack Model

We consider that the adversary can compromise a few sensor nodes without being detected. If a node is compromised, all the information it holds will also be compromised. We will use a Byzantine fault model, where the adversary can inject arbitrarily chosen malicious data readings at a few sensors. Of course, compromised nodes may behave in arbitrarily malicious ways, which means that measurements from compromised nodes are under the complete control of the adversary. We conservatively assume that all compromised nodes collude, or are under the control of a single attacker. An archetypical attack compromises nodes reporting bogus measurements in an attempt to skew the computed aggregate. Compromise of sensor nodes is indeed a real threat in real sensor networks. Because sensor nodes must be low-cost, we often cannot afford to mount them in physical packaging that provides a high level of tamper resistance. Because sensor nodes must be deployed into the environment, we cannot provide physical security or control access to them. And, because sensor nodes must be deployed in large numbers, the adversary is afforded many opportunities to compromise a sensor node.

Of course, the adversary's capabilities are not unlimited. Some cost or luck will be required for each node that the adversary wishes to compromise. Therefore, we should assume that the adversary can compromise only a limited number of sensor nodes, but not half of the network.

# CHAPTER 4

# PROBLEM AND PROPOSED SOLUTION

## 4.1. Problem Statement

Let *t* denote the discrete time variable and $X_i[t]$ be the measurement collected by the sensor $1 \leq i \leq N$, at time *t*. Furthermore, individual sensor readings are subject to environmental noise and hence we assume an additive Gaussian sampling noise to these measurements with known variance $\sigma_n^2$, as it can be estimated regularly by the base station from all the sensor readings.

$$X_i[t] = A + n_i;\ where\ A\ is\ actual\ value\ and\ n_i \sim N(0, \sigma_n^2)$$

All these nodes are provided with identical computational resources and we pick one node as root node, which communicates directly with base station. But in our discussions, for simplicity, we consider root node and base station as same identity, as root node can be varied periodically for it not to be exhausted completely. The initial energy available at each node be E0 and which is its maximum energy storage capacity and gains energy at a constant rate of e', independent of whether it's active or not. We assume an attack model, where the adversary selects certain specified fractions of nodes randomly according to a uniform distribution. To these nodes, a Gaussian random noise of zero mean and variance, $\sigma_a^2$, which is significantly higher than the noise variance, $\sigma_n^2$, which is assumed to be known as it can be periodically estimated from the sensor readings.

Essentially, we can understand the problem of determining the state of a node *i*, as a detection problem with the null hypothesis, $H_0: node\ i\ is\ healthy$ and alternate hypothesis being $H_1: node\ i\ is\ unhealthy$, described as follows:

$$H_0 :\ X_i[t] = A + n_i$$
$$H_1 :\ X_i[t] = A + n_i + a_i$$
$$where\ A\ is\ actual\ value, n_i \sim N(0, \sigma_n^2)\ and\ a_i \sim N(0, \sigma_a^2)$$

Eventually our problem is to estimate *A*, within the tolerable error bound using less power by trading the accuracy. For this once we identify a node as unhealthy we ignore

these nodes and perform aggregation. In each cycle, we are constrained to sense a limited fraction of nodes, say ‘k’ nodes, out of total ‘N’ nodes, to save energy in-order to increase the average lifetime of the network. Essentially the chance that a node among the children of a parent gets sampled is directly proportional to the product of cost, reliability and energy available. With these ‘k’ nodes, and the root node, we form a routing tree to aid data aggregation and we must ensure that all these ‘k’ nodes are included in the tree.

### 4.2. Outlier detection

To determine whether a particular node is attacked or not i.e. the state of node is 0 or 1, we use median/mean based outlier detection methods. Outliers distort the average and standard deviation and make these statistics unreliable. Deletion of outlier data is a controversial practice frowned on by many scientists and science instructors; while mathematical criteria provide an objective and quantitative method for data rejection, they do not make the practice more scientifically or methodologically sound, especially in small sets or where a normal distribution cannot be assumed. Rejection of outliers is more acceptable in areas of practice where the underlying model of the process being measured and the usual distribution of measurement error are confidently known. And hence we do not include the measurements from sensors which are identified to be attacked to make the aggregate more accurate and thus making it robust.

For mean based detection, we flag a node as attacked if it is outside the $2\sigma_n$ limit. For median based method, we use a graphical procedure called a boxplot for summarizing univariate data introduced by John Tukey [33], which includes a simple rule for flagging observations as outliers. That rule declares observations as outliers if they lie outside the interval $\left( Q_1 - g(Q_3 - Q_1), Q_3 + g(Q_3 - Q_1) \right)$; where $Q_1$ is first quartile and $Q_3$ is third quartile. The common choices for g is 1.5 for flagging "out" values and 3.0 for flagging "far out" observations. The proposed approach is especially easy to use for large data sets, which are becoming quite common. Simple adjustments make the procedure practical for smaller sample-sizes. To avoid false positives as all the measurements are close, we use this method only when the IQR (Inter quartile range given by $(Q_3 - Q_1)$) is greater than 1.5 $\sigma_n$.

### 4.3. Metrics for node comparision

At each node starting from the root, we pick a certain fraction of nodes to be sampled among its children based on their Cost function, Reliability factor and Fraction of energy remaining, the computation of which is defined as follows:

1. **Cost function**: Cost or weightage of a node, n, with respect to the entire network is directly proportional to the total number of children it has, i.e. the entire set of nodes that communicate to root node through the node n, and is defined as follows:

$$C(i) = \frac{Total\ number\ of\ nodes\ under\ the\ node\ i}{Total\ number\ of\ nodes\ (N)}$$

2. **Reliability factor:** Each node, $i$, is in state $S(i)$, either attacked or healthy, represented by 0 or 1. The state of each node can be visualized as a Bernoulli Random variable $\phi$ with a probability that it take 1 is $\rho$ and 0 is $1 - \rho$. Where as in each cycle, using our median/mean based consensus algorithm, we estimate the state (1 or 0). Since the states are subject to change at any time in the presence of an attacker, we calculate the exponential moving average of these samples, with a finite forgetting factor, λ to estimate $\rho$. This estimate of probability $\rho$ which is the probability that a node is healthy (1) can be referred as the reliability coefficient $R(i)$ of the node $i$, defined as below:

$$R(i)_t = \lambda\, S(i)_t + (1 - \lambda)\, R(i)_{t-1}$$

Let *t(n)* be the time elapsed (in number of clock cycles) from the last sampling of node n. Here we encounter two cases at each node, where the node is being sampled continuously, *t(n) = 0*, or hasn't been sampled for certain time, *t(n) ≠ 0*. Hence we need to λ accordingly, as follows:

$$\lambda_i = \begin{cases} \frac{1}{T} & ;\ t(i) = 0 \\ \frac{\frac{t(n)}{T}}{\frac{t(n)}{T} + 1} = \frac{1}{1 + \frac{T}{t(n)}} & ;\ t(i) \neq 0 \end{cases}$$

At each node, only the state information of its immediate children is available, we need to propagate the reliability information to the parent. But we only require a single bit reliability flag to be transferred from the children. Hence at each node, we compute an Effective Reliability Coefficient $\hat{R}(i)$, which is weighted average of Reliability Coefficient of the node and its immediate children, given by:

$$\hat{R}(i) = \frac{\sum_{j \in set\ of\ all\ children\ of\ i} C(j)R(j)_t + C(i)\,R(i)_t}{\sum_{j \in set\ of\ all\ children\ of\ i} C(j) + C(i)}$$

3. **Energy Remaining:** We know the initial energy available with each node $E_0$ and energy available at node $i$ at time t is $E(i)_t$. The fraction of energy remaining at node $i$ is given by $FER_i = \frac{E(i)_t}{E_0}$.

### 4.4. Query Disemmination

Base station inititates a query to pick a certain fraction (which is specified in the query from base station) of reliable nodes for sensing and performing the aggregation.

Step 0: BS sends a query to sense fixed fraction of nodes *k*

Step 1: Starting with the root node, we compute the effective reliability coefficient of its children $\hat{R}(i)$

Step 2: Compute the fraction of energy remaining $FER_i$

Step 3: Compute the product and sort the children with high to low value of the product $\hat{R}(i)FER_i$ .

Step 4: Use the first *k* fraction of these children for sensing

Step 5: Repeat step 1 to 4 for each of the selected node

Starting from the root node, each parent (*node$_i$* ) looks at their children's (*node$_j$'s* ) reliability coefficient and selects a certain fraction of nodes (*k*) according to *node_selection* algorithm given below;

**Algorithm:** *node_selection(node$_i$)*

---

*W = []* *# to store the weights for each node i.e. the product of $\hat{R}(i)$ and $FER_i$*
*K$_i$* = $\lfloor k * \textbf{len}(children(node_i)) \rfloor$
***for*** *node$_j$* ***in*** *children(node$_i$)**:***
    *W.**append(***$\hat{R}(i)\ FER_i$***)***

---

```
W.sort()
selected_nodes = indices of W[ 0 : K_i]

# Repeat the same algorithm recursively till we reach the leaf nodes
for node_k in selected_nodes:
        node_selection(node_k)
```

### 4.5. Q-digest Algorithm for median computation

We know that median is more robust than mean, but median cannot be accurately computed in distributed fashion, hence the tradeoff 'energy' vs 'accuracy'. Few algorithms are available to compute Median in distributed fashion, of Q-digest (Quantile digest) [32] is the one with improved and controllable efficiency.

The core idea behind q-digest is that it adapts to the data distribution and automatically groups values into variable sized buckets of almost equal weights. Since q-digest is aimed at summarizing the data distribution and to support quantile computation, it is useful to compare it with traditional database approaches such as histograms. The critical difference between q-digest and a traditional histogram is that Q-digest can have overlapping buckets, while traditional histogram buckets are disjoint. Q-digest is also better suited towards sensor network queries.

A q-digest consists of a set of buckets of different sizes and their associated counts. Every sensor has a separate q-digest which reflects the summary of data available to it, and this can be the partial state record that is propagated up the network as mentioned earlier. In any particular sensor, the q-digest is a subset of these possible buckets with their associated counts. The q-digest encodes information about the distribution of sensor values. The size of the q-digest is determined by a compression parameter k.

#### 4.5.1. Construction of Q-digest

The root node initiates a Q-digest algorithm while constructing the routing tree. The Q-digest data structures generated at these nodes (starting from the nodes at the bottom excluding leaf nodes) are propagated up the tree. And these Q-digests are merged at the parent node and so on. The parent can cache the q-digests received from its children and if a q-digest from a child is lost, it can replace that q-digest by the older one as these

sensor measurements doesn't change rapidly, thus addressing the problem of message losses. The idea is to have a binary tree data structure with nodes as buckets for a particular range. Starting from the root node, the range is $[1, \sigma]$, and its children have $[1, \frac{\sigma}{2}]$ and $[\frac{\sigma}{2}, \sigma]$, and so on till the leaf nodes have buckets of length 1. To construct the q-digest we will hierarchically merge and reduce the number of buckets. Given the compression parameter *k*, a node *v* is in q-digest if and only if it satisfies the following digest property:

$$count(v) \leq \left\lfloor \frac{n}{k} \right\rfloor, \qquad (1)$$

$$count(v) + count(v_p) + count(v_s) > \left\lfloor \frac{n}{k} \right\rfloor, \qquad (2)$$

$$where\ v_p\ is\ the\ parent\ and\ v_s\ is\ the\ sibling\ of\ v$$

We go through all nodes bottom up and check if any node violates the digest property. Since we are going bottom up, the only constraint that can be violated is Property (2), i.e. nodes whose parent and sibling add up to a small count. The algorithm to execute this hierarchical merge is described as ***Compress***, which takes the uncompressed q-digest Q, the number of readings *n* and compression parameter *k* as input, is shown below;

**Algorithm:** *Compress*

---

$l = \log \sigma - 1$
***While*** $l > 0$ ***:***
    ***for*** *v* ***in*** *level l**:***
        ***if*** $count(v) + count(v_p) + count(v_s) < \left\lfloor \frac{n}{k} \right\rfloor$ ***:***
            $count(v_p) += count(v) + count(v_s)$
            ***delete*** *v* ***and*** $v_s$ ***from*** *Q*
    *l = l - 1*

---

### 4.5.2. Merging of Q-digest

In a true sensor network setting we need to be able to build the q-digest in a distributed fashion. For example if two sensors $S_1$ and $S_2$ send their q-digests to their parent sensor (parent in the routing tree), the parent sensor needs to merge these two q-digests to construct a new q-digest and also add its own value to the q-digest. A single value can be

considered a trivial q-digest with one leaf node. Since merging multiple q-digests is no harder than merging two digests, the idea to merge two q-digests is to take the union of the two and add the counts of buckets with the same range *([min, max]).* Then, we compress the resulting q-digest, as shown in the *merge* algorithm below;

**Algorithm:** *merge( $Q_1$( $n_1$, k), $Q_2$( $n_2$, k))*

$Q_1 = Q_1 \cup Q_2$
*Compress( Q, $n_1 + n_2$, k)*

### 4.5.3. Adapting Q-digest for our purpose

After computing the q-digest structure, each sensor has to pack it, and transmit it to its parent. The main limitation of sensor networks is their limited bandwidth. To represent a q-digest tree in a compact fashion we number the nodes in a level by level order, i.e. root is numbered 1 and its two children are numbered 2 and 3 etc. But the problem with the algorithm mentioned above is, it assumes the values generated at each node are integers in the range of $[1, \sigma]$. In all practical purposes of WSNs, we have a closed range of values allowed and beyond which is unacceptable or sometimes unattainable. For the Q-digest algorithm to work it only requires the measurements to attain values from a limited set. Hence we assume a range of length $\sigma$ centered at an average measurement in normal conditions, say $A$, i.e. the range is $[A - \frac{\sigma}{2}, A + \frac{\sigma}{2}]$. The accuracy in aggregation is inversely proportional to the length of the range of values the nodes can take. In our case, we can round of the aggregates to one or two decimal points, to limit the number of possible values and be able to use Q-digest.

### 4.5.4. Quantile Computation using Q-digest

At each node using these Q-digest binary tree structures and by post order traversal we can compute median and other quantiles to determine whether the node is attacked or healthy with the help of Tukey's test. To find the $q^{th}$ quantile from q-digest, we sort the nodes of q-digest in increasing right endpoints (max values); breaking ties by putting smaller ranges first. This list (L) gives us the post-order traversal of list nodes in q-digest. Now we scan L (from the beginning) and add the counts of nodes as they are seen. For some node $v$ of this q-digest, this sum becomes more than $q \, . \, n$; where $n$ is the total

number of sensors branched under that sensor node, and it is suggested to report maximum of the bucket i.e. *v.max* as estimate of quantile. Notice that there are at least qn readings with value less than *v.max*, hence rank of v is at least qn. The sources of error are readings with value less than *v.max*, present in ancestors of v. These will not be counted in quantile algorithm, since v comes before its ancestors in L. This error is bounded by $\varepsilon n$ (refer: Theorem 1 [32]). So, the rank of value reported by our algorithm is between $qn$ and $(q + \varepsilon)n$. Thus the error in our estimate is always positive, i.e., we always give a value which has a rank greater than (or equal to) the actual quantile. But we report average of maximum and minimum of the range of bucket, i.e. $(v.\max + v.\min)/2$ as our estimate of the quantile thus making the error two sided. Though this increases the error in case where there is no attack, but in presence of attacked nodes, average is noted to be good estimate. However in special cases where we are only concerned of one sided attacks, the former is more sensible.

As noted earlier median based outlier elimination is more robust but as the median computed is an approximation, the aggregate is not very accurate. As we are using Q-digest, in the worst case, the count of any node can deviate from its actual value by the sum of the counts of its ancestors which implies that the maximum error is bounded (refer: *lemma 2* [32]). Furthermore, in many practical applications of WSNs like fire/burglar alarms, intrusion detection etc., it's important to accurately determine the range in which the aggregate is present rather the exact value.

### 4.6. Improvement in median based outlier detection

Our algorithm for selecting nodes to sample is based on assumption that we accurately determine whether a particular node is attacked or not. Hence, minimizing the number of false positives is important and we identified that median is more resilient in this regard than mean. In response to this vulnerability, we also tried to modify Tukey's method [33], to classify the measurement $X$ observed into 4 states (opposed to two states earlier 0/1) as: 00 – attacked if $X \in \left[ A - \frac{\sigma}{2}, Q_1 - 3(Q_3 - Q_1) \right) \cup \left( Q_3 + 3(Q_3 - Q_1), A + \frac{\sigma}{2} \right]$, 01 – not healthy if $X \in [\, Q_1 - 3(Q_3 - Q_1), Q_1 - 1.5\,(Q_3 - Q_1)) \cup (Q_3 + 1.5\,(Q_3 - Q_1), Q_3 + 3(Q_3 - Q_1)\,]$, 10 – healthy if $X \in [\, Q_1 - 1.5\,(Q_3 -$

$Q_1), Q_1)$ $\cup$ ( $Q_3, Q_3 +$ 1.5 ( $Q_3 -$ $Q_1$) ] and 11 – very healthy if $X \in (Q_1, Q_3)$. That is, we communicate additional bit to the parent signifying the reliability of children, and accordingly update the reliability factor i.e. instead of 0 or 1, now we use the values 0.0 for 00, 0.25 for 01, 0.75 for 10 and 1.0 for 11. However, we will note in later sections that no significant improvement is observed by this approach, implying single bit state information is completely sufficient in computing reliability factor.

# CHAPTER 5

# RESULTS AND ANALYSIS

Using all the mentioned ideas and algorithms we have simulated different approaches in a similar setting to understand relative efficiency. Unless mentioned the true value of the sensor measurements is centered at *A = 100*, embedded in additive Gaussian noise with variance, $\sigma_n{}^2 = 1$, which is assumed to be known at each node and sampling period is $T = 5$ which runs for 200 cycles i.e. 1000 clock cycles in total. Let us assume all float value communication is done using 32-bits, and for simplicity let energy spent for communicating a bit from one node to immediate child or parent is 1. In naïve approach of mean aggregation by forwarding all the measurements up the aggregation tree for 1000 clock cycles require 32000 units of energy. Let the energy available with each node at the time of deployment be $E_0 = 32000\ units$ and each sensor node gains energy at a constant rate, $e' = 16\ units\ per\ clock\ cycle$, that is half the energy required in naïve approach, only as a reference to compare all the approaches. For the whole network to be eternal, we need to bring down the energy requirement for each cycle to half. For approaches involving median computation using Q-digest algorithm, we assume the measurements can take values in the range $[\,90, 110]$.

At time *t*, each attacked node *i*, takes the measurements $X_i[t]$ given by;

$$X_i[t] = A + n_i + a_i \text{ where } A \text{ is actual value}, n_i \sim N(0, \sigma_n^2) \text{ and } a_i \sim N(0, \sigma_a^2)$$

The attacks used in the simulation are as follows (here *clk* is the number of clock cycles):

*100 < clk < 200:* $\sigma_a = 4$

*300 < clk < 400:* $\sigma_a = 5$

*500 < clk < 600:* $\sigma_a = 6$

*700 < clk < 800:* $\sigma_a = 4$

*900 < clk < 1000:* $\sigma_a = 5$

To simulate events I have varied the mean A of the actual distribution as follows:

*0 < clk < 750:* *A = 100.0*

*750 < clk < 850:* *A = 102.0*

*850 < clk < 1000:* *A = 103.0*

### 5.1. Naïve distributed mean aggregation

We look at naïve distributed mean aggregation as a bench mark approach to have essence of how much we have gained by implementing our ideas. Query from the base station requests to collect measurements from all the nodes, i.e. fraction of nodes to be selected is 1 and rest remains the same. Starting with the root node, each parent $node_i$ collects the sensor measurements from its children in 32-bit notation, and mean of these values is computed. Similarly these aggregates are forwarded up the tree till the root node and aggregate computed at the root is transmitted to the base station. The *aggregate* algorithm for naïve distributed mean aggregation is as shown below;

**Algorithm:** *aggregate($node_i$)*

```
values = []                          # list of values used for mean computation
values.append( node_i._mesurement)   # node_i._mesurement = x_i[t]
for node_j in children(node_i):
    if len(children(node_j) ≠ 0:
        aggregate(node_j)
        values.append(node_j._aggregate)
    else:
        # if the node is a leaf node
        values.append(node_j._measurement)
    node_i._aggregate = mean(values)
return
```

The average root mean square error using this approach is approximately 0.9, which attests the fact that mean based aggregation is not robust to strong attacks. From the plot of Error in aggregate over time in *fig 1*, the error is large when there is an attack. It's worth noting that even in the absence of adversary, we observe significant error in the aggregate implying error is not bounded in mean based aggregation. From the plot of total energy available at each clock cycle *fig 2*, we observe independent of the presence of adversary; energy is spent at a constant rate resulting in 25% reduction.

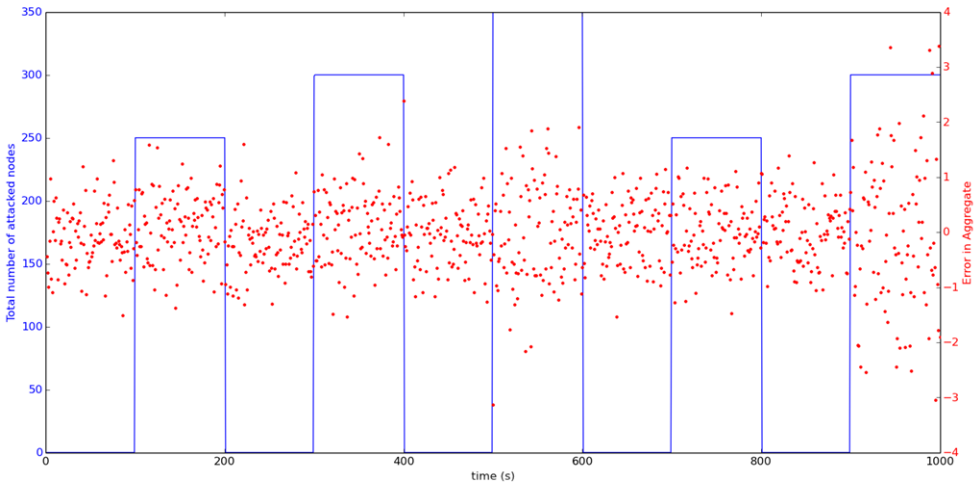


Fig. 1. Plot of total number of attacked nodes and Error in aggregate using naïve distributed mean aggregation

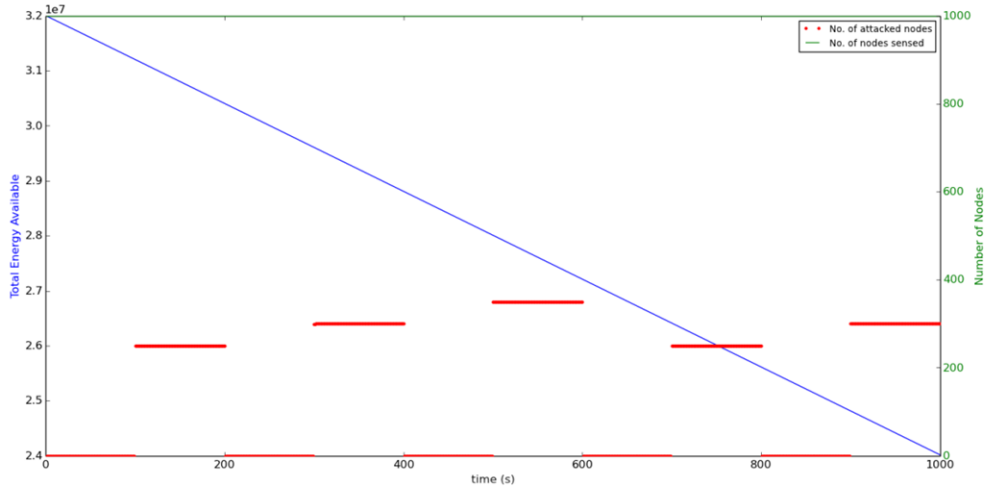


Fig. 2. Plot of total number of attacked nodes and Total energy available at each clock cycle

However as we are activating all the nodes to sense and transmit measured values, we don't save any energy by this approach (see *fig 3*). Our objective is to save energy when there is no adversary and use this energy to make the network resilient to attack otherwise, i.e. instead of total available energy line of single gradient; we need to have a one dependent on the errors.

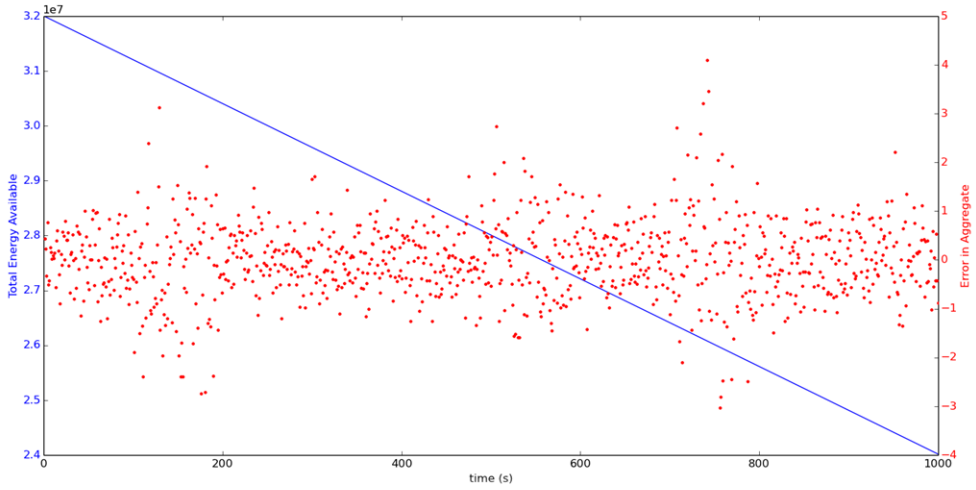


Fig. 3. Plot of total number of attacked nodes and Error in the aggregate

## 5.2. Mean aggregation of reliable nodes

Instead of using all the measurements to compute the aggregate (mean), we use mean based outlier elimination to determine whether a particular node is reliable or not (i.e. healthy or attacked). Then we only include the nodes which are noted to be reliable in computing aggregate, as shown in the *aggregate* algorithm below;

**Algorithm:** *aggregate($node_i$)*

```
values = []                                  # list of values used for mean computation
values.append( node_i._mesurement)           # node_i._mesurement = x_i[t]
for node_j in children(node_i):
        if len(children(node_j) ≠ 0:
                aggregate(node_j)
                if node_j._reliable = 1:     # include or transmit only reliable readings
                        values.append(node_j._aggregate)
        else:
                # if the node is a leaf node
                values.append(node_j._measurement)

node_i._aggregate = mean(values)
outlier_detection(node_i)                    #mean based outlier detection using 'values'
return
```

We can observe from *fig 4* that we are able to contain the error in presence of an attack, and the average root mean square error amounted to be 0.85, which is a slight improvement from the naïve approach. It is to be noted that the error in most cases is with in the $2\sigma_n$ limit, which is the bound we used to determine whether a node is reliable or not. Thus, inherently that is the best we can achieve from this approach.

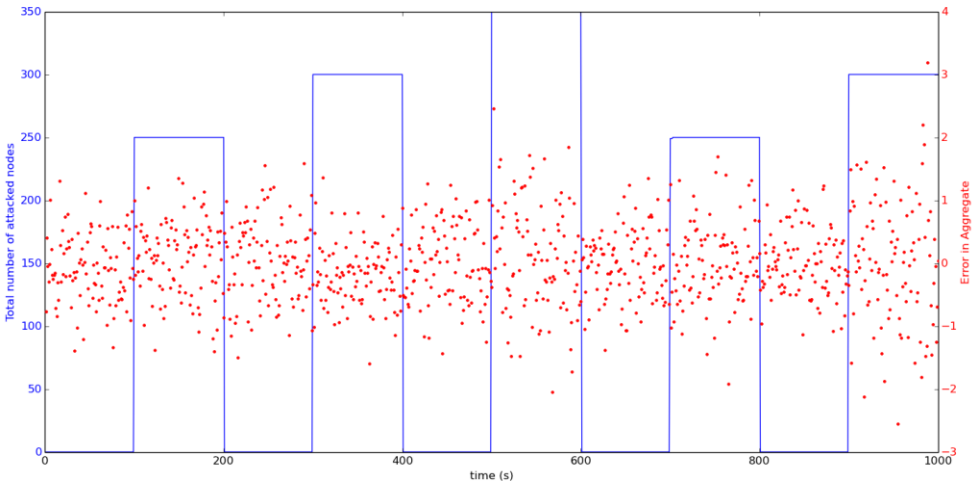


Fig. 4. Plot of total number of attacked nodes and Error in the aggregate using mean aggregation of reliable nodes

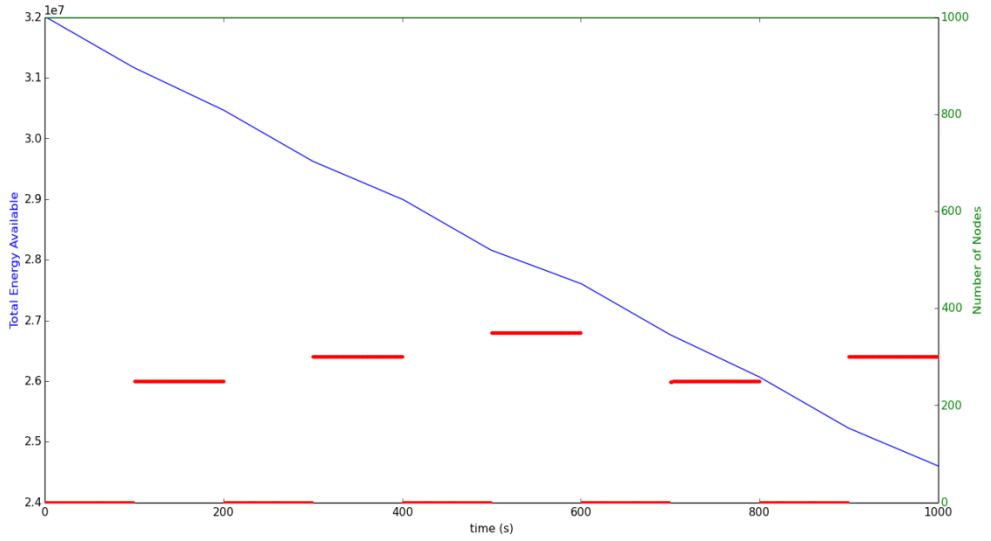


Fig. 5. Plot of total number of attacked nodes and total energy available at each cycle

According to our algorithm, if once a node is identified as attacked or unreliable, we don't require the aggregate at the node to parent. Hence, when there are attacked nodes

present in the network and are identified to be attacked, we save energy, approximately 12% of which lost in the previous approach. We can observe in the *fig 5* below that the energy loss is slowed. But actually we should save energy when adversary is not present, and use that saved energy to make the attack resilient in presence of adversary, which is not the case here. Now we try to reduce energy wastage by reducing the number of nodes we sample in our next approach.

### 5.3. Mean aggregation using selected nodes

As mentioned earlier to reduce the average amount of energy spent for each cycle, starting from the root node we select a fraction of nodes (0.5 in our case) among its children. The aggregation algorithm remains same as in the previous section, but in the query dissemination i.e. *node_selection* algorithm, we select only 50% of its children at each node. These are the ones that are considered more reliable and important, i.e. the ones with high effective reliability coefficient and energy remaining.

Using this approach, the average root mean square error observed is 0.8, which not a big change, but we can notice from *fig 7* that we save 25% energy that is spent in the naïve approach.

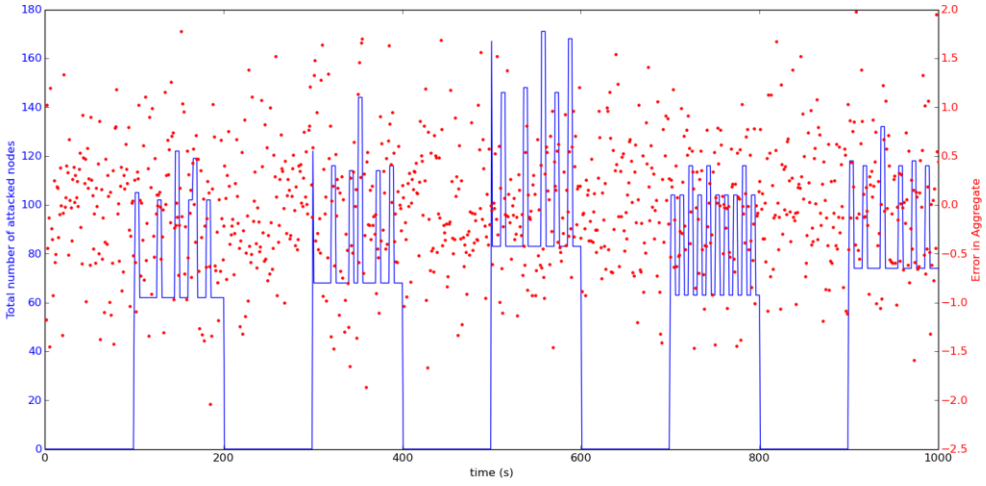


Fig. 6. Plot of total number of attacked nodes and Error in the aggregate using mean aggregation of selected nodes

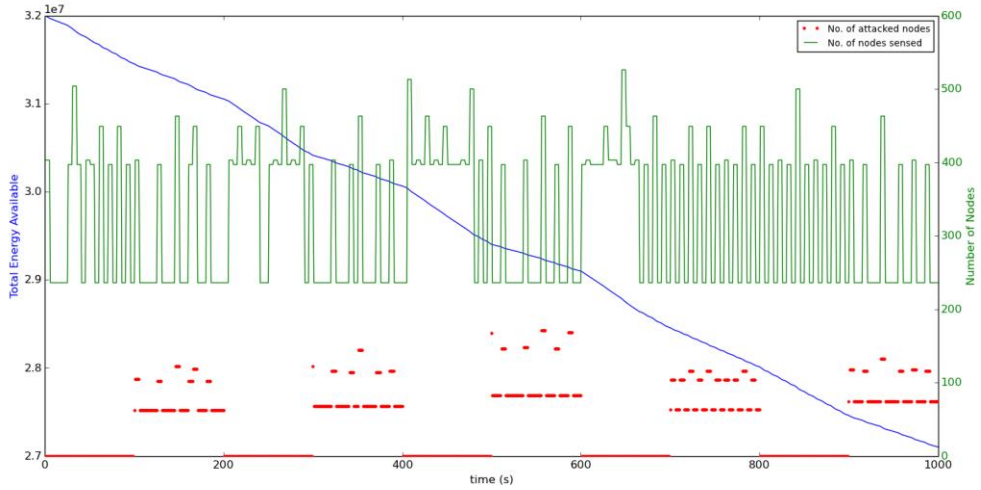


Fig. 8. Plot of total number of attacked nodes and total energy available at each cycle

Though the energy used looks dependent on the total number of nodes attacked, on close observation you will find the energy is spent more in absence of attack, because all the measurements are transmitted where as in presence of an attack, we only transmit measurements of nodes that are noted reliable. This is evident from the correlation coefficient of -0.18 between energy available and total number of nodes attacked.

### 5.4. Naïve median aggregation

Similar to the method described in 5.1, here we use median computed using Q-digest algorithm (refer: *page 23, chapter 4*) instead of mean. Using this approach, the average mean square error observed is 0.29. From the *fig 8*, we can note the error is bounded approximately by 0.3 both in the presence and absence of an adversary. As we have assumed the sensor readings can take values in the range of $[90, 110]$ while implementing the Q-digest algorithm, the minimum size of a bucket in the digest is 0.675. Since we are using the average of the bucket ranges as approximation for median, we observe an error close to $\pm 0.3$ (i.e. nearly half of 0.675), which is the trade-off made to save the energy as shown in *fig 9*. Opposed to constant message length (32-bit) in case of mean, here we use controllable message length (function of number of buckets filled) and thus save the energy.

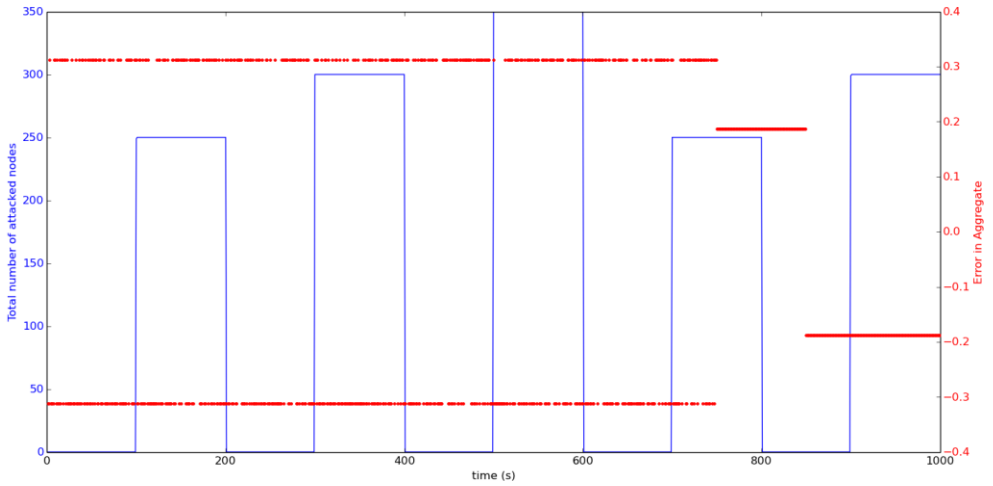


Fig. 8. Plot of total number of attacked nodes and Error in the aggregate using median aggregation of reliable nodes

Moreover, the median based outlier detection algorithm has given 0 false positives, i.e. nodes that are flagged as attacked are actually attacked. Although it wasn't quite successful in detecting all the attacked nodes among the sampled ones, it has to be noted that these nodes are within the range of the values produced by nodes with normal behavior and doesn't affect our aggregate.

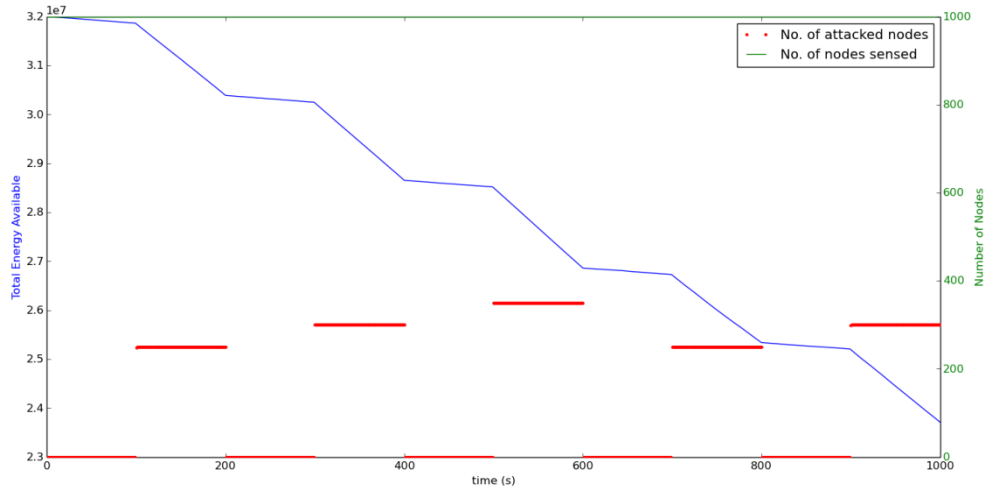


Fig. 9. Plot of total number of attacked nodes and total energy available at each cycle

It is worth noting from the energy plot (*fig 9*), that in the absence of an adversary, the gradient of total energy plot is nearly 0, which implies the nodes are accomplishing the task only by using the energy they gain from ambient sources (at rate $e' = 16$ units per cycle). However at the end of 1000 cycles, total energy lost is same as in the case of

naïve mean aggregation, which implies though we are saving energy in absence of an attack, we are wasting energy in the presence of an attacker. Hence, we need to minimize the number of nodes we activate by selecting most reliable nodes for sampling (5.5). Similar to the approach in 5.2, i.e. including only the values from reliable nodes to compute the aggregate, we can implement similar method using median. But we don't save much energy as message length doesn't vary even if we don't include value at that node if found unreliable and end up wasting energy transmitting the reliability information. However, as mentioned above, we cannot improve the accuracy either, in fact we decrease the accuracy in aggregate (observed a root mean square error of 0.45) by decreasing the number of samples used for computing the median.

### 5.5. Median aggregation using selected nodes

Similar to the method described in 5.3, here we use median based outlier detection and use the reliability information to pick the best nodes. Using this approach, the average root mean square error observed is 0.4. Furthermore, the energy usage is highly dependent on the strength of the attack, *fig 10*. It should be noted that an error of 0.33 is inherent to our median approximation algorithm even when all the nodes are behaving normally, and hence 0.4 is not huge considering the amount of energy we save using this approach, see *fig 11*.

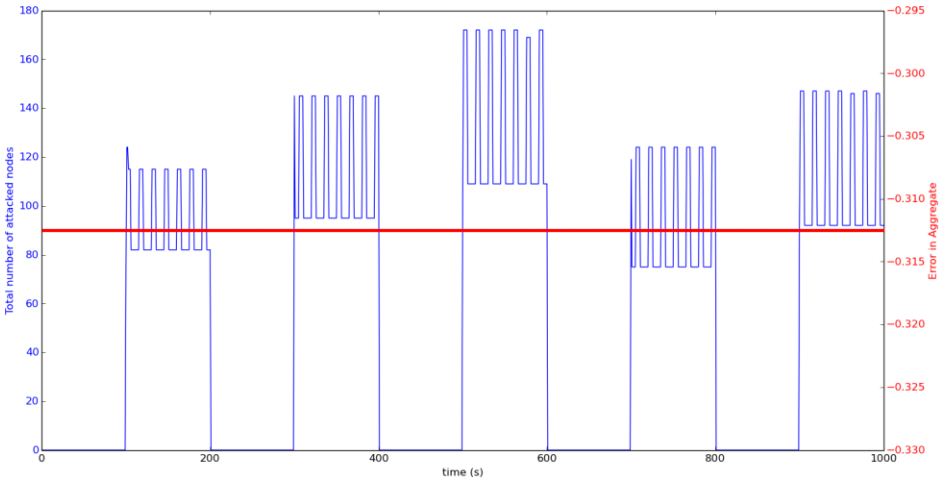


Fig. 10. Plot of total number of attacked nodes and Error in the aggregate using median aggregation of selected nodes

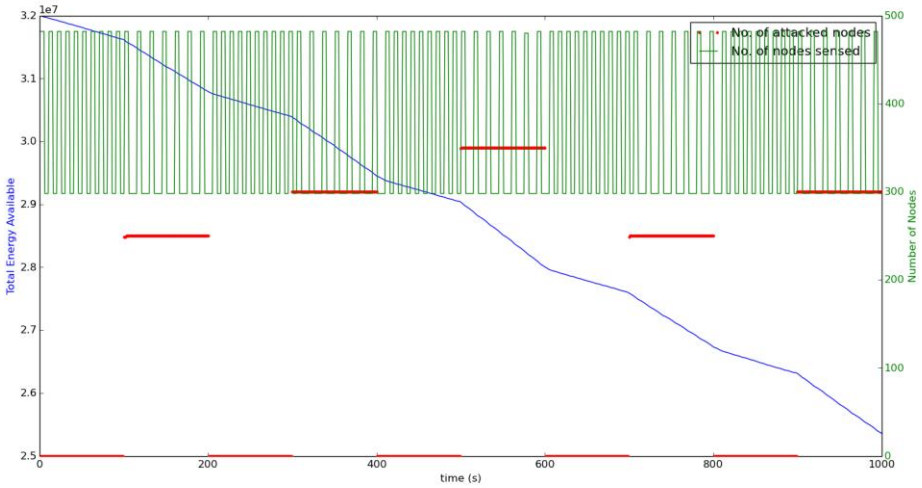


Fig. 11. Plot of total number of attacked nodes and total energy available at each cycle

To have detailed picture of the efficiency of this algorithm, we have computed the root mean square error over time period where adversary is effective and otherwise, and these values amounted to be 0.56 and 0.33 respectively.

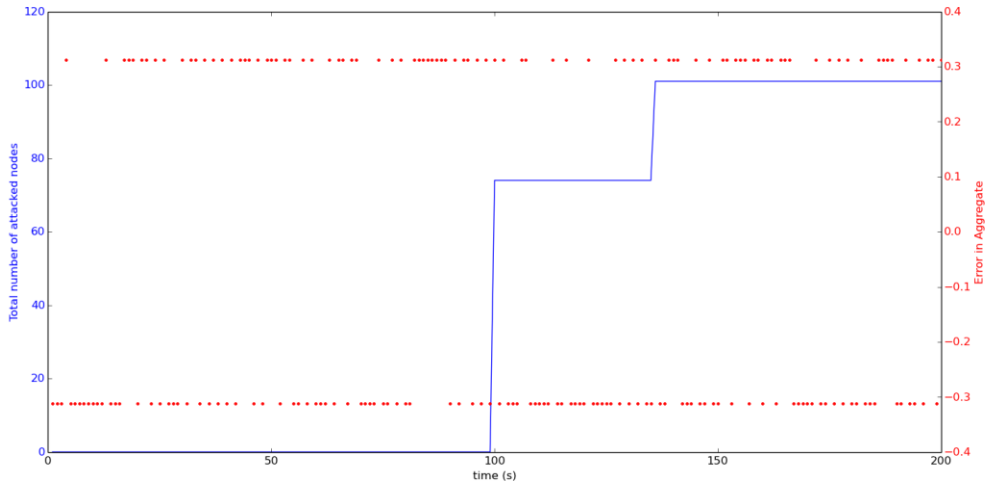


Fig. 12. Plot of total number of attacked nodes and Error in the aggregate using median aggregation of selected nodes

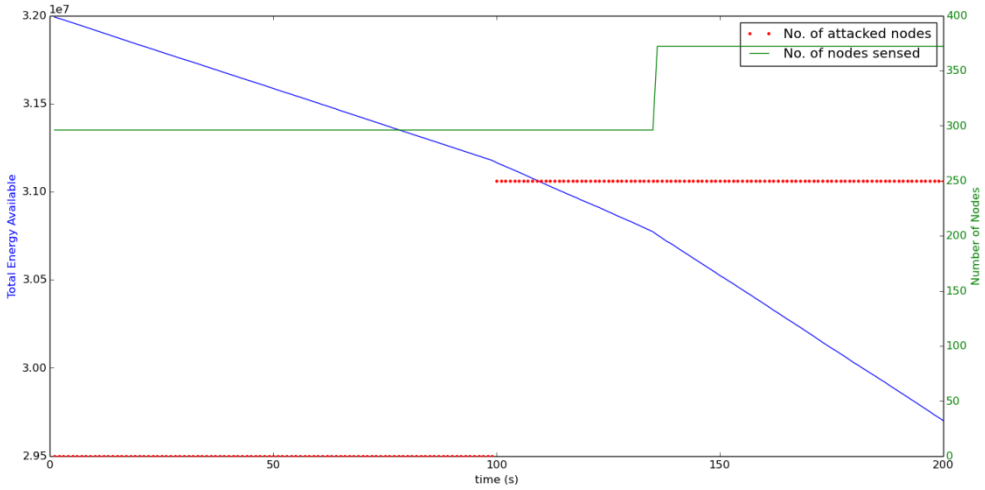


Fig. 13. Plot of total number of attacked nodes and total energy available at each cycle

# CONCLUSION

We have proposed different approaches for making these aggregation schemes more resilient against certain attacks. We have used only mean and median though there are more sophisticated algorithms available to separate additive Gaussian attack, because these are computationally light weight and can be easily implemented in distributed fashion. We also identified that median is more robust as suggested by many, and also effective in outlier elimination than mean. We have adapted the Q-digest algorithm for distributed median computation. Though we have shown in our results we save energy in communication, implementing these algorithms require high computational power, which is not addressed here. This is one wide area for research to develop computationally light weight algorithms using minimum power. There are few works addressing this issue, but the aggregate functions used are not secure, implementing which under same assumptions requires fully homo-morphic encryption system which is out of the scope of this discussion. Major contribution of our work is a method to locally select a set of reliable nodes for sampling considering the cost or weightage, reliability factor and total energy available with the node, in order to increase the life time of the network and improve resilience. Very few works have considered using the feedback from outlier detection to modify sampling query in order to make the network robust to attacks.

# REFERENCES


[1] David Wagner. Resilient aggregation in sensor networks. In Proceedings of ACM workshop on Security of Ad Hoc and Sensor Networks (SASN), 2004.

[2] Samuel Madden and Michael J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In Proceedings of Intl. Conf. On Data Engineering, pages 555–566. IEEE Computer Society, 2002.

[3] Daniel J. Abadi, Wolfgang Lindner, Samuel Madden, and J¨org Schuler. An integration framework for sensor networks and data stream management systems. In Mario A. Nascimento, M. Tamer O¨ zsu, Donald Kossmann, Ren´ee J. Miller, Jos´e A. Blakeley, and K. Bernhard Schiefer, editors, VLDB, pages 1361–1364. Morgan Kaufmann, 2004.

[4] James Reserve Microclimate and Video Remote Sensing. Http://www.cens.ucla.edu.

[5] Habitat Monitoring on Great Duck Island. Http://www.greatduckisland.net/.

[6] The Firebug Project. Http://firebug.sourceforge.net.

[7] Adrian Perrig, John A. Stankovic, and David Wagner. Security in wireless sensor networks. Commun ACM, 47(6):53–57, 2004.

[8] Y. Yao and J. E. Gehrke. The cougar approach to in-network query processing in sensor networks. ACM SIGMOD Record, 31(2):9–18, September 2002.

[9] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In Proc. Of the 2nd international conference on Embedded networked sensor systems (sensys), 2004.

[10] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In Proc. Of ACM MOBIHOC, 2006.

[11] S. Ganeriwal and M. B. Srivastava. Reputation-based framework for highly integrity sensor networks. In Proc. Of ACM Workshop on Security of Sensor and Adhoc Networks (SASN), Washington, DC, 2004.

[12] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli. Fault tolerance techniques in wireless ad-hoc sensor networks. In Sensors 2002, Proceedings of IEEE, pages 1491 – 1496.

[13] Lingxuan Hu and D. Evans. Secure aggregation for wireless networks. In Proceedings of Symposium on Applications and the Internet Workshops, 2003.

[14] Bartosz Przydatek, Dawn Xiaodong Song, and Adrian Perrig. SIA: secure information aggregation in sensor networks. In Proceedings of ACM Conference on Embedded Networked Sensor Systems (sensys), 2003.

[15] Pawan Jadia and Anish Mathuria. Efficient secure aggregation in sensor networks. In Proceedings of International Conference on High Performance Computing,2004.

[16] Emiliano De Cristofaro, Jens-Matthias Bohli, and Dirk Westhoff. FAIR: fuzzybased aggregation providing in-network resilience for real-time wireless sensor networks. In Proceedings of ACM conference on Wireless network security (wisec), 2009.

[17] Saurabh Ganeriwal, Laura K. Balzano, and Mani B. Srivastava. Reputationbased framework for high integrity sensor networks. ACM Transactions on Sensor Networks (TOSN), 4(3):1–37, 2008.

[18] Yan Sun, Hong Luo, and Sajal K. Das, A Trust-Based Framework for Fault-Tolerant Data Aggregation in Wireless Multimedia Sensor Networks, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 9, NO. 6, NOVEMBER/DECEMBER 2012

[19] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. ACM Transactions on Sensor Networks (TOSN), 4(2):7:1–7:40, 2008.

[20] Haifeng Yu. Secure and highly-available aggregation queries in large-scale sensor networks via set sampling. In Proceedings of International Conference on Information Processing in Sensor Networks, 2009.

[21] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences, 31(2):182–209, 1985.

[22] Péter Schaffer and István Vajda. CORA: Correlation-based Resilient Aggregation in Sensor Networks

[23] Keqin Liu and Qing Zhao. Dynamic Intrusion Detection in Resource-Constrained Cyber Networks

[24] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In Proc. Of ACM MOBIHOC, 2006.

[25] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In Proceedings of ACM Conference on Computer and Communications Security (CCS), 2006.

[26] Sankardas Roy, Sanjeev Setia, and Sushil Jajodia. Attack-resilient hierarchical data aggregation in sensor networks. In Proc. Of ACM Workshop on Security of Sensor and Adhoc Networks (SASN), 2006.

[27] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in Proc. 44th Annual IEEE Symp. Foundations of Comp. Sci., oct. 2003, pp. 482–491.

[28] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," IEEE Transactions on Automatic Control, vol. 31, no. 9, pp. 803–812, 1986.

[29] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," IEEE Transactions on Automatic Control, vol. 48, no. 6, pp. 988–1001, June 2003.

[30] Haotian Zhang and Shreyas Sundaram. A Simple Median-Based Resilient Consensus Algorithm. Fiftieth Annual Allerton Conference Allerton House, UIUC, Illinois, USA. IEEE 2012

[31] Sankardas Roy, Mauro Conti, Sanjeev Setia and Sushil Jajodia. Secure median computation in wireless sensor networks. ELSEVIER: Ad Hoc Networks 7 (2009) 1448–1462

[32] Nisheeth Shrivastava Chiranjeeb Buragohain Divyakant Agrawal Subhash Suri. Medians and Beyond: New Aggregation Techniques for Sensor Networks. ACM 1-58113-879-2/04/0011

[33] Tukey, J.W. (1977), Exploratory Data Analysis, Reading, MA: Addison-Wesley

[34] Grubbs, F. E. (February 1969), "Procedures for detecting outlying observations in samples", Technometrics 11 (1): 1–21

[35] S. Jajodia, P. Liu, V. Swarup, and C. Wang, Cyber Situational Awareness, Springer, 2009.

[36] H. Debar, M. Dacier, and A. Wespi, "Towards A Taxonomy of Intrusion Detection Systems," Computer Networks, vol. 31, no. 8, pp. 805-822, 2005.

[37] W. Lee and S. J. Stolfo, "Data Mining Approaches for Intrusion Detection," Proceedings of the 7th conference on USENIX Security Symposium, 1998.

[38] D.E. Denning, "An Intrusion-Detection Model," IEEE Transactions on Software Engineering, no. 2, vol. SE-13, pp. 222-232, 1987.

[39] M. Roesch, "Snort-Light Weight Intrusion Detection for Networks," Proceedings of the 13th Large Installation System Administration Conference, 1999.

[40] A. K. Ghosh, A. Schwartzbard, and M. Schats, "Learning program behavior profiles for intrusion detection," Proceedings of the 1st conference on Workshop on Intrusion Detection and Network Monitoring, 1999.

[41] T. Bass, "Intrusion Detection Systems and Multisensor Data Fusion," Communications of ACM, no. 4, vol. 43, April, 2000

[42] Souptik Datta Kanishka Bhaduri Chris Giannella Ran Wolff and Hillol Kargupta. Distributed Data Mining in Peer-to-Peer Networks.

[43] P. Whittle, "Restless Bandits: Activity Allocation in a Changing World," J. Appl. Probab., vol. 25, pp. 287-298, 1988.

[44] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TAG: a Tiny aggregation Service for Ad-Hoc Sensor Networks. Proceedings of the ACM Symposium on Operating System Design and Implementation (OSDI), 2002.

[45] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. ACM Transactions on Database Systems (TODS), 30(1):122–173, 2005.