AN ABSTRACT OF THE THESIS OF

Rahul Khanna for the degree of Doctor of Philosophy in
Electrical and Computer Engineering presented on March 11, 2016.

Title: Evolutionary Approach to Efficient Provisioning and Self-organization in Wireless Sensor Networks (WSN)

Abstract approved: ______________________________

Huaping Liu

Advances in low-power digital integration and microelectro-mechanical systems (MEMS) have paved the way for micro-sensors. These sensors are equipped with data processing capabilities along with sensory circuits. Sensor data are processed on these individual sensors and transmitted to the target (sink). Low-cost integration and small sizes of these sensors have generated special interest in the area of disposable-sensors and large scale platform management. Queries to these sensors are addressed to nodes which have data satisfying the same condition. However, these sensors may be constrained in energy, bandwidth, storage, and processing capabilities. Large number of such sensors along with these constraints creates a sensor-management problem. At the network layer it amounts to setting up the efficient route that transmits the non-redundant data from source to the sink in order to maximize one or more sensor objectives (e.g. battery (and sensor's) life, Sensor-Data yield). This is done while adapting to changing connectivity due to failure of some nodes and new nodes powering up.

First part of the thesis propose a reduced-complexity genetic algorithm (GA) for optimization of multi-hop battery-constrained sensor networks. The goal of the system is to generate optimal number of sensor-clusters with cluster-heads. It results in minimization of the power consumption of the sensor system while maximizing the sensor objectives (coverage and exposure). The genetic algorithm is used to adaptively create various components such as cluster-members, cluster-heads, and next-cluster. These components are then used to evaluate the average fitness of the system based on the sequence of communication links towards the sink. We then enhance the genetic algorithm (GA) approach for secure deployment of resource constrained multi-hop sensor networks. The goal in this case is to achieve secure coverage and improve battery life by dynamically optimizing security attributes (Like authentication and encryption).

Further, we augment the GA approach for intrusion detection of resource constrained multi-hop sensor networks. Traditional intrusion detection mechanisms have limited applicability to the sensor networks due to scarce battery and processing resources. Therefore, we propose an effective scheme that would offer a power efficient and lightweight approach to identify malicious attacks. We evaluate sensor node attributes by measuring the perceived threat and its suitability to host local monitoring node (LMN) that acts as trusted proxy agent for the sink and capable of securely monitoring its neighbors. Security attributes in conjunction with genetic algorithm jointly optimizes the selection of monitoring nodes (i.e., LMN) by dynamically evaluating node fitness by profiling workloads patterns, packet statistics, utilization data, battery status, and quality-of-service compliance.

Second part of the thesis delves into application of Information Technology (and Industrial) Systems and devices where the use of sensor networks can deliver non-intrusive and effective telemetry for group-based server management. These systems (Like Data Centers or Shipment tracking) face major challenges in seamless integration of telemetry and control data that is essential to various autonomic management functions related to power, thermal, reliability, predictability, survivability, locality and adaptability. Such systems that are supported by a dense network of sense-points operating in noisy environment (Metals, Cables) are required to deliver reliable trends, measurements and analysis in a timely fashion. The traditional approaches to provide distributed observability and control using wired solutions are static, expensive, and non-scalable. We apply the proposed GA approach for this unique environment that replaces static wired sensors with dynamically reconfigurable battery-powered wireless sensors. The proposed technique employs machine learning approach to optimize sensor node function assignment, clustering decisions, route establishment and data collection trees for improved throughput that results in effective controls.

Evolutionary Approach to Efficient Provisioning and Self-organization in Wireless Sensor Networks (WSN)

by

Rahul Khanna

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented March 11, 2016
Commencement June 2016

ACKNOWLEDGEMENTS

I would like to acknowledge **James Carr** and **Lefteris Ioannidis** who participated in the development of the reference WSN experimental setup using Intel(R) Quark(TM) D1000 MCU and Intel(R) Quark(TM) X1000 Gateway during their internship at Intel. I would also like to acknowledge **Anil Keshavamurthy (Intel)** who helped with the installation of backend connectivity tool to observe the network connectivity. I would also like to acknowledge **Prof. Hsiao-Hwa Chen** who was my co-author in many of my publications described in this theses. My data-center WSN experiment required consistent data gathering and I would like to acknowledge **Thanunathan Rangarajan (Intel)** who setup the experiment with multiple servers and helped to gather thermal and power data. Additionally, I would like to acknowledge my discussions related to extending battery life of a sensor node using low power *wireless wakeup circuit* with **Prof. Arun Natarajan (OSU)**, **Christopher Carlson (Intel)** and **Mario Alzate (Intel)**.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF APPENDIX FIGURES

LIST OF APPENDIX TABLES

Chapter 1 – Introduction

1.1 Background

Sensors typically capable of wireless communication and are significantly constrained in the amount of available resources such as energy, storage, bandwidth or computation. Such constraints make the design and operation of sensor networks considerably different from contemporary wireless networks, and necessitate the development of resource conscious protocols and management techniques. The goal of the system is to generate optimal number of sensor-clusters with cluster-heads to maximize the sensor objectives (coverage, exposure and guaranteed time-delivery). Wide range of applications such as disaster management, military, security and data-centers have fueled the interest in sensor networks during the past few years. Recent advances in wireless sensor networks have led to many new protocols specifically designed for sensor networks where energy awareness and low-effort deployment are essential consideration. Most of the attention, however, has been given to the routing protocols since they might differ depending on the application and network architecture.

Clustering of a network to minimize the distance is an NP-hard problem [2,3]. As a part of our thesis, we develop an evolutionary algorithm [4] that divides the randomly deployed sensors into an optimal number of independent clusters with cluster-head and optimal route. Cluster-head collects data from those sensors that belong to the cluster and sends them to the sink in a compressed manner via the most cost-effective route. It is assumed that while the sensors may be deployed in a non-hospitable or non-accessible environment, the sink (or Gateway) is a stationary component that is located at a safe location.

1.2 Sensor Network Architecture

The design of Sensor Network is influenced by various factors like power consumption, fault tolerance, coverage, scalability, network dynamics and the data delivery model. In this section we will discuss various Data centric Routing protocols, Hierarchical Routing Protocols and MAC protocols associated with the Sensor Networks.

1.2.1 Data centric protocols

In many applications of sensor networks, it is not feasible to assign global identifiers to each node due to the sheer number of nodes deployed. This consideration has led to datacentric routing, which is different from traditional address-based routing (like TCP/IP Routing) where routes are created between addressable nodes managed in the network layer of the communication stack. In data-centric routing, the sink sends queries to certain regions and waits for data from the sensors located in the selected regions. Since data is being requested through queries, attribute based naming is necessary to specify the properties of data. Some of the popular data-centric protocol use in sensor networks are the following:

(a) ***Flooding/Gossiping*** In flooding [5], each sensor receiving a data packet broadcasts it to all of its neighbors and this process continues until the packet arrives at the destination or the maximum number of hops for the packet is reached. On the other hand, gossiping is a slightly enhanced version of flooding where the receiving node sends the packet to a randomly selected neighbor, which picks another random neighbor to forward the packet to and so on. Gossiping reduces the implosion but adds delay to the message propagation. At the same time it doesn't guarantee that all nodes of the network will receive the message.

(b) ***Rumor Routing*** Rumor Routing is an agent based path creation algorithm [6]. Agents are randomly created entities that are circulated across the network to establish shortest path to events they encounter. When agent finds a node whose path to the even is longer than its own, it updates the node's route table. When the query is generated at the SINK, it is sent on a random walk with the hope that it will find the path (pre-established by the agent) leading to the required event. Rumor routing achieves significant energy saving over event flooding and can also handle nodes failure. However, rumor routing performs well only when the number of events is small. For large number of events, the cost of maintaining agents and event-tables in each node may not be cost effective.

(c) ***Sensor Protocols for Information via Negotiation (SPIN)*** (SPIN) [7] is among the early work to pursue a data-centric routing mechanism. It uses negotiation and resource adaptation to address the deficiencies of flooding. Negotiation reduces overlap and implosion, and a threshold based resource-aware operation is used to prolong the network lifetime.

SPIN names the data using high-level descriptors or meta-data which is exchanged among sensors before transmission, via a data advertisement mechanism. There is no standard meta-data format and it is assumed to be application specific. SPIN uses three types of messages:

(i) *ADV* allow a sensor to advertise a particular meta-data.

(ii) *REQ* message requests the specific data.

(iii) *DATA* message carries the actual data.

SPIN data advertisement (ADV) mechanism cannot guarantee the delivery of data. Data is not delivered to the destination if the nodes between source and destination are not interested in that data. Therefore, SPIN is not a good choice for applications such as intrusion detection, which require reliable delivery of data packets over regular intervals. SPIN-2 expands the SPIN protocol that reduces the participation using resource threshold.

(d) ***Directed Diffusion*** [8] aims at diffusing data through sensor nodes by using a naming scheme for the data. It consists of several elements: interests, data messages, gradients, and reinforcements. An *interest* message is a query or an interrogation which specifies what a user wants. Each interest contains a description of a sensing task that is supported by a sensor network for acquiring data. Data is named using attribute-value pairs. A sensing task is disseminated throughout the sensor network as an interest for named data. This dissemination sets up gradients within the network designed to draw events (i.e., data matching the interest). Specifically, a gradient is direction state created in each node that receives an interest. The gradient direction is set toward the neighboring node from which the interest is received. Events start flowing towards the originators of interests along multiple gradient paths. The sensor network reinforces one, or a small number of these paths. While positive gradient encourages the data flow along the path, negative gradient inhibits the distribution of data along a particular path. The strength of gradient is different in different neighbors, resulting in source-sink paths with different gradients. Query-based data delivery model is not very useful to applications that require continuous monitoring (environmental monitoring). Moreover, the naming schemes used in Directed Diffusion are application dependent and each time should be defined a priori. Moreover, the matching process for data and queries might require some extra overhead at the sensors.

(e) ***Constrained anisotropic diffusion routing (CADR)*** This protocol [9] is generalization of directed diffusion routing. It uses both information gain and communication cost to direct the data diffusion, and hence can be more energy aware and efficient than directed diffusion routing where typically only communication cost is of concern. The goal is to guide the query as close as possible towards the maximum of the objective function. This approach allows sensors to become activated when there are interesting events to report, and only those parts of the network with the most useful information balanced by the communication cost need to be active. The networks could also actively seek out information, based on predictions of when and where interesting events will be. An entire sensor network, with its in-network processing power for processing, routing, and combining distributed sensor data, is an extremely powerful distributed computing system.

(f) ***COUGAR*** [10] uses an approach to tasking sensor networks through declarative queries. Given a user query, a query optimizer generates an efficient query plan for in-network query processing, which can vastly reduce resource usage and thus extend the lifetime of a sensor network. The query proxies register the query, create a local operator tree, active relevant sensors, and return records according to the specification of the query plan. In addition, the query plan also specifies how to determine the leader of this query, a designated node where the computation of the average sensing operation will take place. The leader could be a fixed sensor with more remaining power and energy, or a randomly selected node by some distributed leader election algorithm. Such ability ensures energy efficiency especially when the number of sensors generating and sending data to the leader is huge. Some of the drawbacks of COUGAR are:

   (i) Additional query layer on each sensor node brings extra overhead in terms of energy consumption and storage

   (ii) The leader nodes have to be dynamically maintained to prevent them from failure.

1.2.2 Hierarchical Routing Protocols

The main aim of hierarchical routing is to efficiently maintain the energy consumption of sensor nodes by involving them in multi-hop communication within a particular cluster and by performing data aggregation and fusion in order to decrease the number of transmitted messages to the sink. A clustered architecture organizes the sensor nodes into clusters, each governed by a cluster-head. The nodes in each cluster are involved in communication with the cluster-head. The Cluster-Head communicates to another Cluster-Head or the SINK that is an access point connected to the wired network. Clusters can be extended to any depth in an hierarchy. Clustered architecture is useful because of its inherent property of data-fusion. Data collected by all members of the cluster can be fused at the cluster-head and the resulting information can be sent to the SINK. Cluster Networks are self-organizing, hence cluster formation and cluster-head election is an autonomous process. Some of the hierarchical protocols are:

(a) ***Low-Energy Adaptive Clustering Hierarchy (LEACH)*** [11] is one of the most popular hierarchical routing algorithms for sensor networks. The idea is to form clusters of the sensor nodes based on the received signal strength and use local cluster heads as routers to the sink. This will save energy since the transmissions will only be done by such cluster heads rather than all sensor nodes. Optimal number of cluster heads is estimated to be 5% of the total number of nodes. LEACH randomly selects nodes as cluster-heads and performs periodic re-election, so that high-energy dissipation experienced by cluster-heads in communication with SINK is spread across all nodes in the network. LEACH is completely distributed and requires no global knowledge of network. However, LEACH is not useful for large networks because it uses uses single-hop routing where each node can transmit directly to the cluster-head and the sink. Moreover, dynamic clustering introduces an extra overhead because of communication involved cluster-head re-election and advertisements.

(b) ***Power-Efficient GAthering in Sensor Information Systems (PEGASIS)*** [12] is an improvement of the LEACH protocol. Rather than forming multiple clusters, PEGASIS forms chains from sensor nodes so that each node transmits and receives from a neighbor and only one node is selected from that chain to transmit to the base station (sink). Gathered data moves from node to node, aggregated and eventually sent to the base station. The chain construction is performed in a greedy way. This approach distributes the energy load evenly among the sensor nodes in the network. For gathering data in each round, each node receives data from one neighbor, fuses with its own data, and transmits to the other neighbor on the chain. One of the drawbacks of this approach is that it introduces high delay for distant node on the chain. In order to decrease the transmission delay to the SINK, an extension to PEGASIS called Hierarchical-PEGASIS is proposed. This approach deals with data gathering problem by considering energy delay metric. PEGASIS introduces high delay for distant node on the chain. Every sensor needs to be aware of the status of its neighbor so that it knows where to route that data. Such topology adjustment can introduce significant overhead especially for highly utilized networks.

(c) ***Threshold sensitive Energy Efficient sensor Network protocol (TEEN)*** [13] is a hierarchical clustering technique where each cluster head forwards the data to an upper-level cluster head, called the super cluster-head. All the super cluster heads at the same level form an upper level cluster with an upper super cluster head and so on until the top level of the network. A hierarchical clustering structure is finally formed. The cluster formation at each level is same as that in the LEACH protocol. Because the application of super-cluster-heads shortens the transmission distance from cluster heads to the base station, TEEN is a more energy efficient scheme for large-scale WSNs than LEACH. The formation of clusters does not require centralized control from the base station. Localized coordination realizes scalability for the large-scale dynamic Wireless Sensor Networks. Due to the two features, energy-efficient and scalable, the cluster-based scheme is a promising solution for Wireless Sensor Networks. TEEN is not good for applications where periodic reports are needed since the user may not get any data at all if the thresholds are not reached. Another drawback is the overhead and complexity of forming clusters in multiple levels, implementing threshold-based functions and dealing with attribute-based naming of queries.

1.3 Challenges in Sensor Networks

There are number of challenges involved in the development of sensor networks. Unlike Ad-Hoc Networks, Sensor Networks can be of very large magnitude and require a very different handling. Sensor networks are prone of failures, non-rechargeable energy drain, sensor identification (or naming) and time-sensitive data delivery. Most routing schemes used in ad-hoc networks cannot be directly used in sensor-networks because of limited memory, computation power, limited battery life and non-scalable nature of the existing protocols. Sensor-Networks also deal with the data-aggregation and fusion from multiple sensors to avoid redundant traffic to save energy. Some of the principal design challenges are the following:

(a) Autonomic setup and maintenance of the network is required to avoid human intervention.Sensor Networks are deployed randomly and do not observe any pre-defined topology.

(b) Disposable Sensor Networks have limited available energy that cannot be recharged or replaced. This poses as major bottleneck in the development of protocols for sensor-network. Protocols designed for such network need to work within this constraint and trade off power power efficiency for network delays and reliability. This is unlike reliable transport protocols that are designed for wired networks (or ad-hoc networks) like TCP etc.

(c) Sensor Networks don't have any infrastructure, hence all routing protocols should be distributed in nature.

(d) Sensor Networks should be able to adapt to the changing connectivity and requirements. Nodes can fail, new nodes can be redeployed or certain established routes can get into congestion state because of sudden burst of activity. Sensors need to be constantly evaluated for functional allocations that is proportional to the battery usage. The goal should be to lengthen the time when the first node fails while maintaining the complete coverage and the connectivity to the SINK.

(e) Secure Communication poses an important challenge that needs to be addressed to avoid command hi-jacking that can result in denial of service, malicious traffic redirection, FAKE coverage data or un-authorized energy drain.

(f) For wireless local area networks (WLANs), several power saving approaches have been standardized for IEEE 802.11 [14] and Bluetooth. In WLANs, the problem is significantly simpler than in ad hoc networks due to the existence of a single coordination point (access point for 802.11 and the master node for Bluetooth).Sensor networks and applications motivate a MAC that is different from traditional wireless MACs such as IEEE 802.11 in almost every way: energy conservation and self-configuration are primary goals, while per-node fairness and latency are less important. On the other hand, node synchronization is required in a highly distributed network using schemes like TDMA schedules and temporal ordering of detected events.

(g) Wireless sensor networks like mobile ad-hoc networks (MANETs) involve multi-hop communications.However, the nature of the applications and routing requirements for the two are significantly different in several respects:

   (i) Communication in a sensor network observes reverse-multicast where data travels from multiple data sources to a data recipient/sink.

   (ii) Since the data being collected by multiple sensors is based on common phenomena, there is likely to be some redundancy in the data being communicated by the various sources in sensor networks. The end-to-end routing schemes for mobile ad-hoc networks are not appropriate under these settings. Data-centric technologies are needed that perform in-network aggregation of data to yield energy-efficient dissemination.

(iii) Sensors are not mobile (though the sensed phenomena may be), so the nature of the dynamics in the two networks is different.

(iv) Depending on the application a node can be dedicated to a particular special function such as relaying, sensing and aggregation since engaging the three functionalities at the same time on a node might quickly drain the energy of that node [15].

1.4 Objective and Thesis Outline

Principal objective of the thesis to to explore evolutionary approach in building sensor network hierarchy that is dependent upon the end-user applications and its attributes. Each application is unique in nature and requires special consideration. For example, disposable sensors are low-cost sensors that may be limited by computational resources and battery power. They may require to build a network hierarchy that can minimize the battery drain while achieving the sensor objectives. On the other hand, sensors used in large scale enterprise data-centers are relatively rich in computational resources and are externally powered. These sensors can measure energy, temperature, air-flow, humidity, node-utilization and other related characteristics in a burst of information. These applications require a network hierarchy that can self-organize and can deliver large amount of reliable, latency-bound data over large-scale and dense sensor networks for visualization and control objectives.

In chapter 2 we introduce energy-efficient design where we propose a genetic algorithm (GA) based optimization function for multi-hop sensor networks with a finite battery life. The objective in this case is to generate optimal number of sensor-clusters with cluster-heads to minimize power consumption of the stationary-sensor system while maximizing the sensor objectives. Therefore, GA algorithms are designed with two objectives: (1) discover the optimal clusters with cluster members and cluster head, and (2) discover low-cost path to the sink using one or more hops. We develop chromosome coding scheme that embeds the identification of each sensor along with any other specific information. This information may be related to sensor objectivity, next-hop, cluster-domain, etc. Furthermore, we also propose coding scheme for *routes representation*. While cluster adaptation creates accurate cluster boundaries due to addition, deletion, or modified sensor objectives, load adaptation creates optimal routes from cluster-heads to the sink. The adaptation process is governed by a fitness function that is specific to the network objective in a load-balanced network.

In chapter 3 we enhance the GA based sensor network design to include secure communication among sensor nodes using power efficient and lightweight approach to identify malicious attacks. The reliable functioning of sensor objectives depend on the secure communications between various functional elements (nodes and sink). This requires identifying compromised or falsely added nodes, secure re-deployment/addition of nodes, and preventing passive listening by a malicious intruder using elements of authentication, integrity, privacy (or confidentiality), and anti-playback. Ideal scheme should reduce the computational overhead while maintaining the adequate security levels by identifying the strategic nodes. Strategic nodes are optimally enabled for security by evaluating the battery status, network traffic, malformed or retries on a specific route and number of nodes in a single route handling authentication.

In Chapter 4 we propose a reduced-complexity genetic algorithm for intrusion detection of resource constrained multi-hop mobile sensor networks. We evaluate sensor node attributes by measuring the perceived threat and its suitability to host local monitoring node (LMN) that acts as trusted proxy agent for the sink and capable of securely monitoring its neighbors. Security attributes in conjunction with genetic algorithm jointly optimizes the selection of monitoring nodes (i.e., LMN) by dynamically evaluating node fitness by profiling workloads patterns, packet statistics, utilization data, battery status, and quality-of-service compliance.

In Chapter 5 we propose a genetic algorithm (GA) for dynamic deployment of resource constrained multi-hop mobile sensor networks. This allows us to achieve optimal coverage and improved battery life using dynamic power scaling (DPS) and improved fitness function. The dynamic power scaling in conjunction with genetic algorithm jointly optimizes power states and topologies by dynamically monitoring workloads, packet arrivals, utilization data and quality-of-service compliance.

Information technology systems face considerable challenges in seamless integration of telemetry and control information. These are essential to various autonomic management functions related to power, thermal, reliability, predictability, survivability, and adaptability. Sensors and control agents supporting this telemetry are a part of large multiprocessor environments that are scattered across the platform. The conventional approaches to support distributed observability and control using wired solutions are static, expensive, and non-scalable. In Chapter 6 we present an alternative approach for this unique environment that replaces static wired sensors with dynamically reconfigurable wireless sensors. It employs a genetic algorithm based approach to optimize sensor node function assignment, clustering decisions, resource distribution, and route establishment for improved control quality. Based on this new, wireless sensor network approach, we evaluate the average data-flow delay characteristics between sensor and control end-points. We also investigate the "quality of control" by measuring the conformity of the controlled objective to platform policy specifications (like power limits).

Chapter 7 onwards we build cluster-based wireless sensor network hierarchy for an enterprise data center. We propose the use of wireless sensor networks as a key technology enabler in data center monitoring and control. In a large scale data center with 10,000 servers 300,000 sense-points, dense network topologies can easily result in bandwidth problems, especially in the case where sensors are constantly generating data. These sense-points are integral part of the data center infrastructure and operate in a highly dense RACK environment. Sensor nodes that consolidate the sense data are not energy constrained and have enough storage/compute resources to sustain moderately complex functions. While we can use similar heuristics as in chapter 2-3, fitness characteristics are influenced by non-intrusive deployment, data reliability, low latency and time-bound data delivery. Sensor data produced by the network of sense nodes assist in load-balancing among nodes, workload consolidation, cooling control (chillers opening), health monitoring/logging, failure tracking, node-locality and visualization of resource trends with cost modeling. Time bound sense data observability facilitates the data center manager to (a) increase the set-points (b) minimize hot-spots (c) reduce operational cost (d) track inventory/diagnostic report thereby allowing to gain reliability, reduce energy costs, recover power and cooling capacity. We utilize GA based heuristics to build non-intrusive, orthogonal sub-nets using channel diversity. We propose data collection scheme for a given sub-tree that employs weight based time-slot allocation that further maximizes the packet-reception-ratio.

Chapter 9 describes the future work in the area of developing highly dense sensor networks for industrial applications similar to asset tracking (or shipment tracking). Such sensor networks have a life-span of 2-3 months. Therefore it is essential to develop algorithms and hardware design that is energy efficient. We propose that such wireless sensor network can utilize the GA assisted provisioning and resource utilization that we developed in this thesis. We elaborate the problem statement and possible solutions (like on-demand low-power wireless wakeup) that can be implemented in the future. We discuss research opportunities in the areas related to energy efficient sensors, data protocols and circuit enhancements like wireless wakeup and energy harvesting. Furthermore, the solutions have to be cost-effective for practical applications where the sensors may be disposable.

In the end we conclude our thesis with conclusions that summarizes the work describes in this thesis. We also provide an appendix (Appendix A) that provide sufficient details (Hardware and Software Tools) to help the maker community to start with a light-weight sensor network based on new Intel(R) Quark D1000 micro-controller and Digi(R) Xbee micro-controller.

Self-Organization of Sensor Networks Using Genetic Algorithms

— Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen —

International Journal of Sensor Networks 1.3-4 (2006): 241-252.

Chapter 2 – Self-Organization of Sensor Networks Using Genetic Algorithms

2.1 Introduction

Advances in low-power digital integration and micro-electro-mechanical systems (MEMS) have paved the way for micro-sensors [16–20]. These sensors are equipped with data processing capabilities along with sensory circuits. Sensor data are processed on these individual sensors and transmitted to the target (sink). Low-cost integration and small sizes of these sensors have generated special interest in the area of disposable-sensors. These are randomly deployed, infrastructure-less, data-centric sensors that cannot be charged or replaced. Queries to these sensors are addressed to nodes which have data satisfying the same condition. These disposable sensors find their uses in the areas of disaster recovery, target identification, reconnaissance, medical applications [21], defense applications [22] and intrusion detection, etc. However, these sensors are constrained in energy, bandwidth, storage, and processing capabilities. Large number of such sensors along with these constraints creates a sensor-management problem. At the network layer it amounts to setting up the energy-efficient route that transmits the non-redundant data from source to the sink in order to maximize the battery (and sensor's) life. This is done while adapting to changing connectivity due to failure of some nodes and new nodes powering up.

Clustering of a network to minimize the distance is an NP-hard problem [2,3]. Therefore, we develop an evolutionary algorithm [4] that divides the randomly deployed sensors into an optimal number of independent clusters with cluster-head and optimal route. Cluster-head collects data from those sensors that belong to the cluster and sends them to the sink in a compressed manner via the most cost-effective route. It is assumed that while the sensors may be deployed in a non-hospitable environment, the sink is a stationary component that is located at a safe location.

Genetic algorithm (GA) is a stochastic search technique that mimics the natural evolution proposed by Charles Darwin in 1858. GA has been successfully applied to a wide range of combination problems. They are particularly useful in applications involving design and optimization, where there are a large number of variables and where procedural algorithms are either non-existent or extremely complicated.

As a part of our research we propose a reduced-complexity genetic algorithm for optimization of multi-hop sensor networks. The goal of the system is to generate optimal number of sensor-clusters with cluster-heads. It results in minimization of the power consumption of the sensor system while maximizing the sensor objectives (coverage and exposure). The genetic algorithm is used to adaptively create various components such as cluster-members, cluster-heads, and next-cluster. These components are then used to evaluate the average fitness of the system based on the sequence of communication links towards the sink. In addition, the mechanism supports dynamically changing coverage, task requirements, failures, incremental redeployment and reconfiguration. Therefore, GA algorithms are designed with two objectives: (1) discover the optimal clusters with cluster members and cluster head, and (2) discover low-cost path to the sink using one or more hops.

2.2 Proposed GA Solution

The system consists of an *initialization module* and an *adaptation module*. The initialization module helps in coding of gene for each sensor. This gene contains the identification of each sensor and any other specific information. This information may be related to sensor objectivity, next-hop, cluster-domain, etc. The initialization module also initiates temporary clusters of the sensors with a domain identification and cluster-heads. The adaptation module is responsible for cluster adaptation and load adaptation. Cluster adaptation is responsible for creating accurate cluster boundaries due to addition, deletion, or modified sensor objectives. Load adaptation is responsible for creating optimal routes from cluster-heads to the sink. Adaptation modules are governed by a fitness function that is specific to the network objective in a load-balanced network. It prevents the flow of redundant information while maximizing the network bandwidth usage and battery life.

It is interesting to note that two competing objectives are required to create an energy-efficient sensor network. While cluster membership will keep on changing because of dead, depleted or replaced nodes, routes to sink will keep on changing to avoid high-cost paths (like multiple clusters using the same cluster-head to route the data to the sink). Therefore, we use multi-objective genetic algorithms [23]. Simple GA converges to a single solution. In problems where there are several, often conflicting objectives, a multi-objective genetic algorithm (MOGA) is used which evolves a set of solutions (the population) towards the Pareto-optimal front where trade-off analysis can be performed to select a suitable solution.

2.2.1 Node Selection Chromosome Representation

The chromosome of the GA contains all the building blocks to a solution of the problem at hand in a form that is suitable for the genetic operators and the fitness function. We propose the coding scheme where each individual sensor node is represented by a 3-bit binary number called 'gene'. These three-bit genes which define the feature of the node are called 'allele' and are represented as follows:

000 - Node Inactive (powered off).

001 - Node chosen as Cluster-Head (CH).

010 - Node chosen as Inter-Cluster Router (ICR).

100 - Node chosen as Sensor (NS).

Each cluster is represented by a cluster-head, and cluster-members are represented by inactive/active sensors and inter-cluster routers. Cluster-head is responsible for data-fusion from various node-sensors and inter-cluster router is responsible for routing cluster data (from cluster-head) to the sink.

For example, in a 25-node system, the number of bits required to represent the complete system would be $3 \times 25 = 75$. Therefore, the size of the string would be 60-bits. For the scenario shown in Fig. 2.1, this string likes as follows:

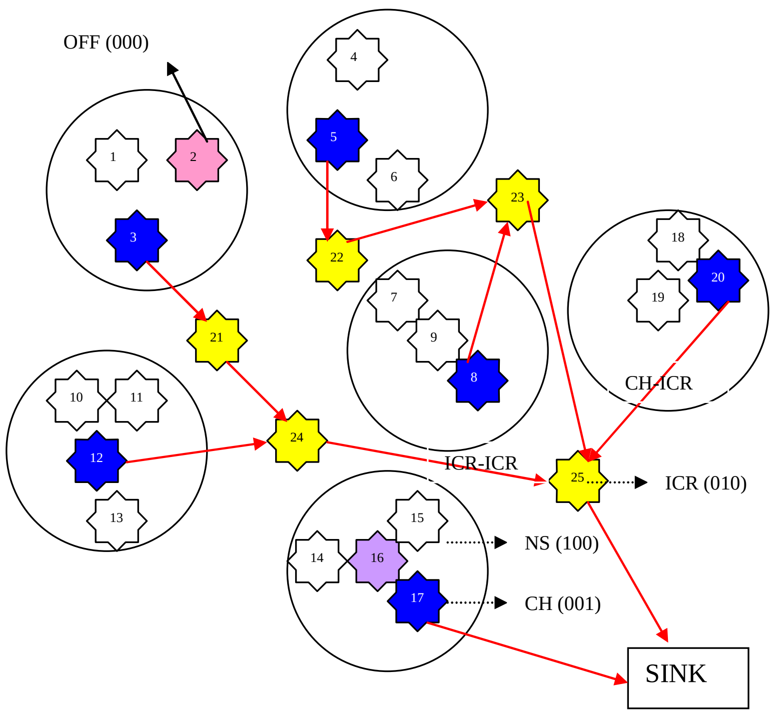


Figure 2.1: Sensor node clustering. Each node is assigned function as a result of genetic algorithm and the resulting chromosome structure. For the example below, chromosome structure is 100 000 001 100 001 100 100 100 001 100 100 001 100 100 100 000 001 100 100 001 010 010 010 010 010.

100 000 001 100 001 100 100 100 001 100 100 001 100 100 100 000 001 100 100 001 010 010 010 010 010

Upon completion of the GA algorithm, a function is assigned to each node. Once the functions are assigned, each type of nodes then performs the following functions:

2.2.1.1 Inter-Cluster Routers (ICR)

(a) Each router starts listening to 'sink' or 'Lx router', where $x = 0, 1, 2, \cdots$ represents the number of hops between sink and itself.

(b) Each router finds out the next-hop energy requirements to the sink and/or Lx routers that it can listen to by exchanging data and bounds checking. This step also involves exchanging a conflict-free proximity unique ID (PUID) with other neighboring routers (Section 2.3).

(c) Each router temporarily designates itself as Lx, where $x = 0, 1, 2, \cdots$, based on the next hop it sends the data to. $L(x)$ can send data to only an $L(x - 1)$ that is closer to the sink.

(d) Each router then sends the neighboring routers (and/or sink) information (from step-b) to the sink using the temporary router chosen in step-c.

(e) Upon cost-analysis using a parallel GA algorithm, the sink will designate a primary and fail-over path to each router and send this information using the node it received that information from. This is a periodic process that repeats at a pre-defined interval.

(f) Lx routers will update its next-hop information by replacing the temporary next-hop with that provided by the sink. Lx routers will receive this information periodically from the sink.

(g) Lx routers will start advertising (router advertisement) its presence with the cost of using this path at regular intervals. This cost is evaluated using the following metrics:

  (i) Average data flowing through this router (dynamic).

  (ii) Energy requirements to reach next hop (static).

(h) Average cost of using the next-hop (static) Lx routers will trigger an attention message when the battery reaches an attention state (battery condition in quantized steps). This attention message is carried to the sink using the current path (updated in step-f). The sink will use this message as a trigger point for re-configuration and running a new instance of node-selection genetic algorithm. In the new instance, the failing node is permanently marked “Powered Off (000b)”.

2.2.1.2 Sensor-Nodes (NS)

(a) Each sensor node starts listening to the available cluster-heads (CH advertisement).

(b) Each sensor node will calculate the cost of communicating with the available cluster-heads.

(c) Each sensor will attach to a cluster-head based on the cost as calculated in step-b and become the part of that cluster. This step also involves receiving the unique PUID from the cluster-head (Section 2.3).

(d) Each sensor will update the chosen cluster-head with the sensor data. These data include the SN-CH cost of all cluster-heads it evaluated in step-b.

(e) Sensor node will trigger an attention message when the battery reaches an attention state (battery condition in quantized steps). This attention message is carried to the sink using the cluster-head (step-b). Sink will use this message as a trigger point to re-configuration and running a new instance of node-selection genetic algorithm.

2.2.1.3 Cluster-Head (CH)

(a) Each cluster-head starts CH advertisement to invite nodes (SN).

(b) Each cluster-head sends the NS-ICR data received from the sensors to the sink.

(c) Each cluster-head listens to the router advertisement and selects the low-cost router (ICR) en-route to the sink. This step also involves receiving the unique PUID from the ICR (Section 2.3).

(d) These cluster-heads can participate in data fusion. The resulting information is then communicated to the sink using the selected router. Sink returns back the application unique ID (AUID) (Section 2.3) to the cluster-head during the setup (or re-initialization) phase.

2.2.1.4 Sink

Sink is an entity where all the event data collection and dissemination take place. This information is then processed for sensor related functions.

Sink also receives the statistical and status information from routers and cluster-heads. This information is processed in the following manner:

(a) It collects the information regarding valid router-router (ICR-ICR) communication. This information includes energy requirements for communication and the corresponding globally unique identification number (GUID).

(b) It evaluates the average data that pass through each router by processing the data received by the sink (from SN).

(c) It evaluates the cost of NS-CH communication for all valid links. This information is passed by the cluster-head during the setup-operation.

(d) It allocates an application specific AUID (Section 2.3) to the cluster-head to uniquely identify the message trace.

(e) Listens to any alert message (battery conditions).

(f) Performs a GA to evaluate the optimal route using the fitness function based on parameters obtained in step-a and step-b. This is triggered based on periodicity or an alert event.

(g) Performs a GA to designate functional unit to each node using the fitness function based on the parameters obtained in step-c. This is triggered based on alert event.

2.2.2 Route Selection Chromosome Representation

Route-selection GA uses a different chromosome structure than that used in node-selection GA (Figure 2.2). Characteristics of route-selection chromosomes are given as follows: Each node (CH and ICR) is represented by $\log_2(N)$ bits, where N is the maximum number of ICR nodes that can be reached by this node. Hence an individual in this case is represented by a string that consists of all such nodes with representation to the next ICR. For example, (0010) (0010) (001) (010) represents R12, R22, R31, R42 connections, where Rxy are the y-th route of the x-th node.

2.2.3 Node Selection Fitness Function

The node selection fitness function is a weighted function that measures the quality or performance of a solution, in this case a specific sensor network design. This function is maximized by the GA system in the process of evolutionary optimization. A fitness function must include and correctly represent all or at least the most important factors that affect the performance of the system. The major issue in developing a fitness function is the decision on which factors are the most important ones. We use the following measure.

2.2.3.1 Coverage Fitness (CF)

A sensor node has an objective to maximize the blanket coverage where the objective is to maximize the total detection area. In many applications of sensor

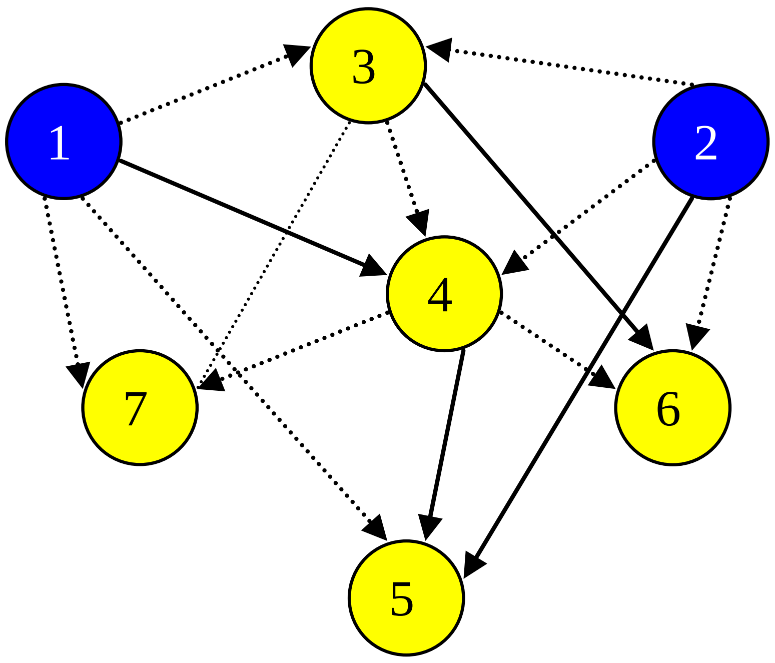


Figure 2.2: Chromosome structure of the route. For example, for nodes 1,2,3,4, chromosome string is represented by (0010) (0010) (001) (010); for node 1, route 2 is selected that connects to ICR 4; for node 2, route 2 is selected that connects to ICR 5; for node 3, route 1 is selected that connects to ICR 6.

networks, the number of neighboring nodes plays an important role. If the network is to be connected, the number of neighbors of each node needs to grow at $\Theta(\log n)$, where n is the number of nodes in the network [24]. Based on the density of deployment ρ, each node will have at least K neighbors and optimal isotropic communication range (R_c) with its neighbors (sensor nodes and cluster-heads) [25]. CF fitness depends on the percentage of nodes that have at least K neighbors.

$$\text{CF} = \sum_i (\min(1, N_i/K)) \tag{2.1}$$

where N_i is the number of fully connected sensor nodes in cluster i.

2.2.3.2 Cluster-Head Fitness (CHF)

Sensor nodes connected to each cluster-head should be uniformly distributed. This prevents cluster-head overloading. CHF defines the fitness based on the uniformity of the sensor nodes and cluster-heads:

$$\text{CHF} = 1 - \min\left(1, \left(\sum_n \frac{|\rho_n - \rho|}{\rho}\right)/N\right) \tag{2.2}$$

where n is the cluster-head number, N is the number of cluster-heads in the system, ρ_n is the number of nodes attached to this cluster-head, and ρ is the average number of nodes per cluster in a system calculated as

$$\rho = \text{Total Sensor Nodes/Total Cluster-Heads.} \tag{2.3}$$

Any cluster consisting of more than ρ sensor nodes will be penalized.

2.2.3.3 Node Communication Fitness (NCF)

A node needs power p to communicate with another node that is d distance away. The power required to communicate with the cluster-head can be computed using the path loss expressed as [26]

$$PL(d) = PL_0 + 10\mu \log_{10}(d/d_0) + S \tag{2.4}$$

where d is the distance between the sensors, d_0 is a reference distance typically chosen as $1m$ for sensor networks, PL_0 is the path loss at the reference distance d_0, μ is the path loss exponent, typical in the range of $2-4$, and S is a zero-mean Gaussian random variable that gives the deviation in path loss from its average value.

For example, sensor nodes calculates these values p by responding to CH advertisements that it can listen to during setup operation. These values are then sent to the sink via a temporary low-cost path chosen by the sensor node during the setup phase. The NCF function is obtained as

$$\text{NCF} = 1 - \min\left(1, \sum_i \sum_j \left(\max\left(0, \frac{p_{ij} - p_t}{p_t}\right)\right)/N\right) \tag{2.5}$$

where p_{ij} represents inter-node communication energy relationship (as measured by individual sensor node), p_t represents energy threshold, and N is the number of sensor-nodes in the system.

2.2.3.4 Battery Status Fitness (BF)

Anytime sensor-node communicates with the cluster-head or cluster-head communicates with inter-cluster router, there is a penalty paid in terms of battery usage. Battery is also consumed during the sensing operation or other related functions. Each node alerts the sink about its battery status (Q) when it crosses the quantized limit (or thresholds). These thresholds will be used to penalize the use of those nodes for operations that consume more battery power. Penalty for using the node with a low battery capacity depends upon the type of node and its usage. For example, a node with low battery capacity will have a greater penalty for inter-cluster routers than the cluster-head. Similarly cluster-head will have a greater penalty than the sensor-node. Therefore penalty suffered by each node depends upon the battery status and the type of node assignment. The battery status fitness function is expressed as

$$\mathrm{BF} = 1 - F(Q, \text{Node Type}) \tag{2.6}$$

where $F(\cdot)$ is the penalty with $0 \leq F(Q, \text{Node Type}) \leq 1$.

2.2.3.5 Router Load Fitness (RLF)

Inter-cluster routers participate in routing the traffic originating from cluster-heads or other ICR to the sink. Routers are penalized if they cater to more than the average number of CH and ICR. This avoids overloading routers. The RLF function is expressed as

$$\mathrm{RLF} = 1 - \sum_{n} \frac{|\varrho - \varrho_n|}{\varrho} / N \tag{2.7}$$

where n is the ICR index, N is the number of ICR in this system, and ϱ and ϱ_n are given as

$$\varrho = \frac{\text{Total(CH + ICR)}}{\text{Total ICR}} \tag{2.8a}$$

$$\varrho_n = \text{Connected CH} + \text{ICRCost}_n \tag{2.8b}$$

where ICRCost_n (n-th ICR) is updated as a result of GA function that evaluates the *cost of using* a router using route selection fitness function (Section 2.2.4).

2.2.3.6 Sensor Effector Fitness (SEF)

SEF is the fitness measure that interprets the power consumed by the sensory action of clusters. The net effect of SEF is to re-arrange the sensor nodes so that the sensor data transmission is uniformly optimized by fusion, elimination or compression methods. Similar packets from multiple nodes can generate large amount of redundant data that can be aggregated to reduce transmissions. Aggregation is done by clustering or re-clustering the nodes in order to perform data aggregation to save energy and is influenced by the following factors:

(a) Eliminating the duplicate data within the cluster [27]. Furthermore, the sensor generating the duplicate data transitions to a low-duty-cycle state based on a leaky bucket hurestics.

(b) Compressing the data by fusing the correlated information within the cluster. This operation is performed by the cluster-head that also updates the sensor node's spatial or temporal policy parameters.

(c) Compressing the data by correlating the information across the clusters. Wirelessly transmitting and receiving bits is the most energy consuming operation done by the nodes; therefore, by reducing the amount of bits that must be sent can significantly extend their lifetime [28]. Mechanisms such as *Slepian-Wolf distributed source coding* [29] can compress the content at the original sources in a distributed manner without explicit routing-based aggregation. This operation can only be performed using the sink mediation. Sink identifies the extent of correlation and updates the CH with new NS policy coding parameters. CH then applies these coding parameters to the respective NS in the associated cluster. This operation is performed using a pre-determined update timer to avoid excessive transmissions from the sink while helping to adapt to the changing correlations. Higher inter-cluster correlations introduces extra communication overhead for mediated traffic.

(d) Reducing the data globally by eliminating the duplicate paths, turning off the sensors with high degree of redundancy or simply re-clustering.

It is possible with certain probability that sensors which belong to the different clusters may have high degree of correlation. This can cause redundant information to traverse through CH and ICR to the sink. The degree of correlation is calculated by analyzing the historical data from the sensor node. This analysis is done in spatial, temporal or spatio-temporal domain. Such analysis will rank the sensor based on the power usage for sensory functions. NS power usage depends on the following factors:

(a) Degree of correlation within local sensors.

(b) Degree of correlation between cluster-heads carrying the fused data.

(c) Compression credit that reduces the redundancy in the information with some overhead. This overhead is calculated by measuring the extra information carried as side-bands or additional communication required to convey the compression parameters to the NS. Various compression schemes are addressed in [12, 30].

By exploiting the redundancy between various sensors in spatial, temporal or spatio-temporal dimensions, the average energy consumption per cluster can be optimized greatly. This energy has to be uniformly distributed among clusters. For example, a cluster allocated on the blind side of the sensory environment may not have much use as a sensor node. Sensors in that clusters may very well belong to a different clusters and be used as a Cluster-Head or an Inter-Cluster-Router (ICR).

$$\begin{aligned} \text{SEF} &= \lambda_1 \left(1 - \left(\sum_j \min\left(1, \frac{|\sum_i (E_{ij}) - E_{avg}|}{E_{avg}}\right)\right)/N \right) \\ &\quad + \lambda_2 \left(1 - \left(\sum_j \sum_i \left(I_{ij} H_{ij} / H_{max} \right)\right)/M \right) \end{aligned} \tag{2.9}$$

where E_{ij} is the average sensory power consumption of the i-th sensor node in j-th cluster, E_{avg} is the average sensory power consumed by all clusters, N

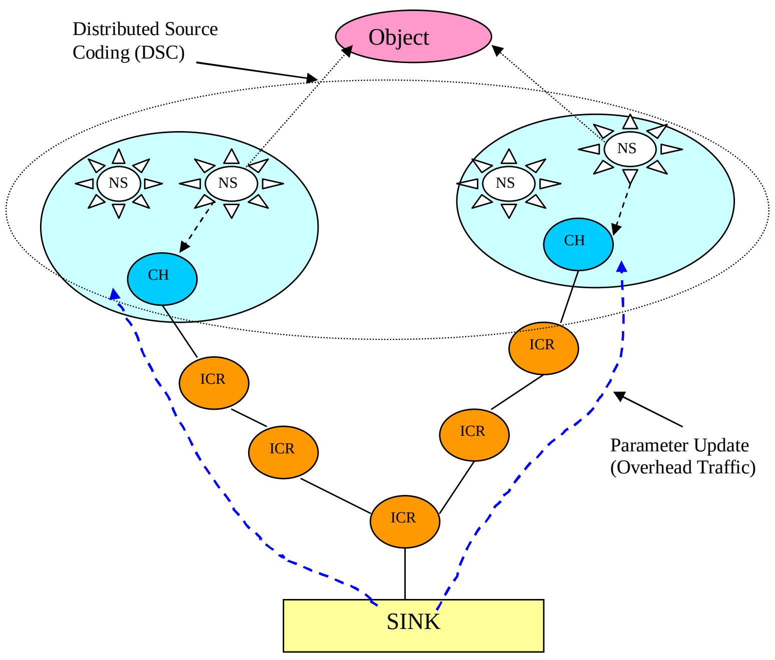

Distributed Source Coding (DSC)
Object
NS
NS
NS
NS
CH
CH
ICR
ICR
ICR
ICR
ICR
ICR
Parameter Update (Overhead Traffic)
SINK

Figure 2.3: An example of distributed source coding (DSC), where an object is monitored from two sensor nodes encapsulated across different clusters. Coding parameters are exchanged with the help of the sink that also monitors the correlation between packets arriving from different clusters. This also introduces the transmission overhead on intermediate routers and cluster-heads.

and M are the total number of clusters and sensor nodes, respectively, H_{ij} is the number of hops between j-th cluster and the sink, H_{max} is the maximum number of hops possible in the sensor network, $I_{ij} = 1$ if sensor node i of cluster j is correlated with another sensor node in a different cluster and $I_{ij} = 0$ otherwise. λ_1 and λ_2, which satisfy $\lambda_1 + \lambda_2 = 0.5$, are the contributions to the SEF fitness due to sensory power consumption and overhead traffic, respectively.

2.2.3.7 Total Node Fitness (TNF)

TNF is the final fitness that is evaluated in the GA algorithm for the appropriate node assignment. It is described by

$$\text{TNF} = \alpha_1\text{CHF} + \alpha_2\text{NCF} + \alpha_3\text{BF} + \alpha_4\text{RFL} + \alpha_5\text{SEF} + \alpha_6\text{CF} \quad (2.10)$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 = 1$ and α_i depends upon the relative significance of the component. These values can be made adaptive using an external heuristics.

2.2.4 Route Selection Fitness Function

The second objective of the MOGA is to generate balanced routes based on node allocation using GA based on node fitness function. During setup operation, both cluster-heads and inter-cluster-routers start sending the data on the most cost effective ICR. It is not guaranteed that the setup connection will remain cost-effective over a period of time. GA predicts the optimal route topology based on the cost of using an ICR for the next sampling period. CH and ICR are updated with this information in each sampling period. Route fitness function takes into account the traffic patterns, battery capacity, and transmission energy. This is accomplished because of the following properties of the sink:

(a) It is aware of the static routes that are either formed during the setup operations or updated during GA operations during a sampling period. This will help GA evaluate average load on each router (since destination of all communication is the sink).

(b) It is aware of the amount of data (bits) received from each cluster which then traverse through a static routes as in (a).

(c) Each ICR updates the sink of its battery capacity as soon as it crosses a threshold value.

(d) It is aware of the energy-cost of transmission to its nearest neighbors. This is proportional to the distance between ICH and its next hop (or sink). This information is sent by the router during the setup phase.

Based on the predicted optimal route fitness, the sink will update the cost of using this ICR for the next sampling period. The total route fitness (TRF) is given by:

$$\mathrm{TRF} = \mathrm{BF} + \mathrm{NCF} + \frac{1}{N} \cdot \sum_{k} \frac{\max(ICR(j,k)) - Curr(ICR(j,k))}{\max(ICR(j,k))} \tag{2.11}$$

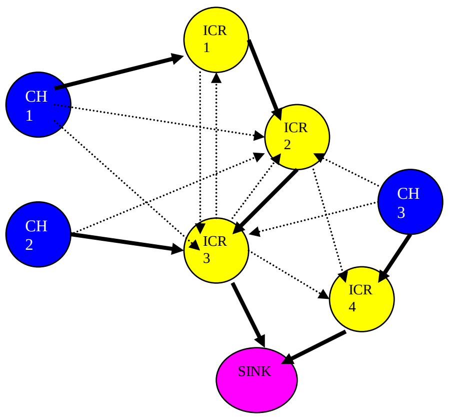


Figure 2.4: Sample output of the route-selection GA. Thick lines represent the low-cost selected route. Dotted lines represent other possible routes, but with higher relative cost (lower fitness). As the system conditions change, low-cost routes can become high-cost routes and vice versa. There is a high likelihood that ICR 3 may turn out to be a high-cost path if CH 2 becomes highly active.

where

$$
\begin{aligned}
\mathrm{ICR}(j,k) =& \text{Average bit-rate handled by ICR } j \text{ that can communicate} \\
& \text{with node } k \text{ (CH or ICR)} \\
\max(\mathrm{ICR}(j,k)) =& \text{ICR with the highest bit-rate that can be communicated} \\
& \text{by node } k
\end{aligned}
$$

$$\begin{aligned}
\text{Curr}(\text{ICR}(j,k)) =& \text{Current ICR that has been designated to} \\
& \text{communicate with node } k \\
N =& \text{Total number of nodes (CH and ICR)} \\
BF =& \text{Battery fitness of the router in question} \\
\text{NCF} =& \text{Node (CH-ICR or ICR-ICR) communication fitness} \\
\text{ICRCost}_n =& 1 - \frac{1}{M} \cdot \sum_k \frac{\max(ICR(j,k)) - ICR(j=n,k)}{\max(ICR(j,k))}
\end{aligned}$$

where $k =$ all the nodes that can communicate with ICR n. For example in Fig. 2.4, for ICR-2, $k = \{$CH1, CH2, CH3, ICR1, ICR3$\}$, M equals the total number of k nodes that can communicate with ICR n; for ICR-2, $M = 5$.

2.2.5 Node Selection Genetic Algorithm

Now that we have defined a node selection fitness, we can design the genetic algorithm for node-selection that can be represented with the following steps using the GA operators. The process of genetic algorithm takes place in the sink (or a similar centralized identity). This algorithm repeats itself upon multiple triggers. These triggers are related to battery alert, deteriorating route fitness alert, periodic action. Once the optimal fitness is achieved, the topology corresponding to that fitness is committed and the sensors are instructed to assume the new functions by relinquishing the old functions.

(a) **Initial population**: Initial chromosomes strings are seeded partially randomly using a random number generator (RNG) and partially using population of previous samples. Population uses the gene structure as defined in Section 2.2.1. This population is coded with gene structure as defined in Section 2.2.1.

(b) **Evaluation**: Each chromosome string is evaluated for the fitness using the TNF function (for node assignment) as defined in Section 2.2.2.

(c) **Reproduction**: Reproduction is a process in which individual strings are copied according to there fitness function values, which also means that individuals with larger fitness value will have a higher probability of contributing an offspring in the next generation. The algorithm uses the standard weighted roulette wheel method to select n individuals for reproduction to the mating pool. Since the TNF defines the fitness value, the chromosome with the highest fitness value means represents a better chromosome to take part in reproduction. N chromosomes will again be reproduced from the n chromosomes selected for reproduction using a crossover probability. During reproduction, we choose multiple cross-over points. Cross-over points and the locations are calculated using an RNG. As in this example, two chromosome strings having three random cross-over points will create a resultant chromosome after cross-over as below:

Parents:

100 000 001 100 001 100 100 100 001 100 **100 001 100 100 100 000 001** 100 100 001 010 010 010 010 010

100 010 010 100 100 010 100 010 001 001 **001 001 010 001 001 000 100** 010 001 100 010 000 001 100 010

Children:

100 000 001 100 001 **010 100 010 001 001** 100 001 100 100 100 000 001 **010**

001 100 010 000 001 100 010

100 010 010 100 100 **100 100 100 001 100** 001 001 010 001 001 000 100 **100 100 001 010 010 010 010 010**

(d) **Mutation**: Newly reproduced N chromosomes are transferred to the mutation pool. The mutation operator mutates chromosome in the mutation pool according to mutation probability which will make it adaptive. We will choose a maximum mutation probability p_m. In any generation, mutation probability will be inversely proportional to the average fitness of the standard number of population in any generation. Therefore

$$p_g = p_m \left(1 - \frac{N * \text{TNF}_{avg}}{\text{TNF}_{total}}\right). \tag{2.14}$$

Mutation function uses function flip (toss of a coin) to decide whether to invert the bit or not.

(e) **Selection**: Finally N chromosomes are chosen out of $2N$ chromosomes according to their fitness values. These chromosomes are carried over to the next generation. $2N$ chromosomes consist of N parent chromosomes and N children.

2.2.6 Route Selection Genetic Algorithm

Route selection GA is similar to the node selection GA with the following exceptions and an extra trigger point. This algorithm repeats itself at a regular interval to ascertain the acceptable thresholds of route-loads during the constant usage of the sensor-system.

(a) **Initial population**: Initial population is chosen partially randomly using an RNG and partially using population of previous samples. Population uses the gene structure as defined above.

(b) **Evaluation**: Each chromosome is evaluated for the fitness using the TRF function (for route selection) as defined in Section 2.2.4.

(c) **Node selection trigger point**: Route-selection GA algorithm keeps on making attempts to achieve the most cost-effective path for a given topology (as selected in node-selection) in which there can be multiple paths. At certain point certain fitness threshold is reached beyond which further conversion to higher fitness may not be possible. This condition can happen due to battery condition and bad node assignments. This will cause a node-reassignment alert, which in turn will cause the node-selection GA to run again with changed conditions.

As seen above, the route selection can sometimes act as a resisting factor for the node-selection. While nodes may have been assigned the functions based on a high fitness factor, it may not be suitable for routing the packets in a multi-hop system. This will cause a re-configuration (running route selection GA) again until both objectives reach an acceptable convergence point. This is a dynamic process and keeps on repeating over the life-time of the system.

2.3 Naming Convention

Naming convention is an important ingredient of the sensor network system architecture. An optimal naming convention will reduce the messaging overhead as well as facilitate collaborative signal processing. Each sensor supports three types of naming conventions:

(a) *Globally Unique Sensor ID* (GUID) is the uniquely identifiable node identification number that is hard-coded in the sensor hardware. This ID is used during the setup phase (or re-initialization phase).

(b) *Proximity Unique ID* (PUID) is the unique identification of the nodes contained within the neighborhood of a cluster (NS ID , CH ID, and ICR ID). This ID is used among the neighboring nodes for link-layer data exchange and collaborative signal processing. These IDs are allocated dynamically in the following manner:

    (i) ICR(s) self allocate its own PUID during the setup (or reinitialization) phase that involves negotiation between the neighboring ICR(s). This ID is used for ICR-ICR link-layer forwarding and not contained in the message header.

    (ii) CH(s) receives its PUID from ICR(s) during the CH-ICR binding operation (Section 2.2.1.3). This ID is used for CH-ICR link-layer forwarding and not contained in the message header.

    (iii) NS(s) receives its PUID from CH(s) during the NS-CH binding operation (Section 2.2.1.2). This ID is used for NS-CH link-layer forwarding. This ID can optionally be carried in the message header when the message flows through the ICR.

(c) *Application Unique ID* (AUID) is allocated by the sink to the cluster-head for message identification. This ID enables the sink to uniquely identify the origin of the message up to the node level. This is a data-layer ID that is carried in the message header and filled in by the cluster-head. This ID is for application use only, and system topology has no visibility into its construction or usage. AUID can be attribute-named data that can enable in-network processing with filters, supporting data aggregation, nested queries and similar techniques that are critical to reduce network traffic and conserve energy [27].

The addressing in this manner reduces the overhead due to addressing bits during the message transmission. This method is scalable to the size of the network. Smaller networks with fewer nodes will use less addressing bits than the larger networks.

2.4 Overhead Traffic

The GA used to perform *Node Selection* and *Route Selection* is targeted with two competitive objectives running at the sink. The introduction of this layer as a separate protocol aids in using the snoop data for predicting the fitness data. It is important to note that the sink builds up the *Node Database* tree by snooping the routed data and the setup-data (initialization step) consisting of node ID, transmission distance (to ICR and other neighboring nodes), associated CH and routes to the sink (primary and fail-over routes). This data is further evaluated at the sink for the creation of additional data-points consisting of various fitness categories defined in Sections 2.2.3 and 2.2.4. While most of the fitness data can be indirectly inferred from the regular data (normal data and setup data), battery loss due to coverage cannot be measured using this method. Moreover, changes in the associations and designations require extra messaging between sink and nodes. Hence various sources of the overhead traffic are:

(i) Setup Messaging - This is mandatory messaging [2.2.1] required for initial setup of the network. These messages are exchanged once during initialization-step and in extreme cases can also be triggered by sink.

(ii) Sink Alert Messaging - This message is performed when sink determines the need for new allocations for nodes and routes as a result of performing GA.

(iii) Node Alert Messaging - This messaging is initiated by the nodes towards sink, in order to identify critical information. Currently, this messaging is used for alerting when:

   (a) Battery levels falls below the thresholds.

   (b) A fail-over route is chosen for future routings. Each ICR identifies itself (starting with the failed over ICR).

(iv) Node Battery Status - The node battery status data flow through the pre-established path using the extra bits in the SN messaging data. Three extra bits in the header gives eight quantization levels.

(v) Correlation Data Timer - This timer updates the spatial or temporal correlation parameters of the sensor nodes that compresses the overall data in a distributed coding [29]. An update has to traverse through all the ICR(s) en-route to the co-operating nodes residing in different clusters. This is a low duty-cycle operation that can adapt to the changing environment thus limiting the expenditure involved in update overhead.

As described, most of the overhead traffic is caused during the setup-operation. A small percentage of messages also flow as Alerts between source and sink. Most of the cost of indirect inference and GA execution is pushed to the sink. As a result, cost of overhead-traffic on the nodes is greatly reduced. Increase in the number of nodes (large networks) shows a longer delay in *Initialization Setup* and *Message Propagation*. Longer delays are mitigated by sub-dividing a large network into smaller domains with identifiable boundaries.

2.5 Sensor Construction

A self-organization mechanism using a contention-free TDMA medium access protocol can be used for sensor networks [31]. During the setup phase, nodes interact in a random access manner. As the clusters are formed during the setup phase or re-clustering event (upon GA trigger), incremental TDMA schedules are formed for sensor nodes, cluster-heads, and inter-cluster-routers. Re-clustering as a result of GA evaluation (and the corresponding alert) will modify the existing TDMA schedules to accommodate the newly formed clusters. Incremental determination of the TDMA schedules is based on the known range limitations on the radio where certain nodes are expected to be outside the region of radio interference with the current node. At the same time, as a result of distributed scheduling, some nodes with similar schedules may interfere with each other. This can be remedied by using multiple channels or spreading codes, that will reduce the overhead due to transmission management. As illustrated in Figure 2.5, typical sensors used in this scenario have the following characteristics:

(a) Transmitter - Transmits the data at various power levels. It should have the ability to quickly enter and exit from one power-state to another.

(b) Receiver - Receives the data targeted toward it. It additionally comprises of a low-power listening engine. Sensor radio consumes almost the same energy as when transmitting, searching for the next packet. To reduce energy consumption during the non-transmitting periods, energy-aware protocols are used. These protocols involve extra transmission overhead in terms of extra attention bits and reduced sampling of the radio channel.

(c) Power control - Controls the power to be transmitted according to the function assigned.

(d) Function control - Performs the protocol actions related to the function. These actions are related to identification, advertising, power-control, performing bindings and associations with other nodes, and sensing, etc.

(e) Test control - Performs the test functions. Test control is transparent to the function control and does not interfere with its working. This control is required to simulate a future topology while not interfering with the current one. GA will make use of this function to evaluate the fitness before committing this topology to all nodes.

(f) Alert generation - Generates an alert action to the sink upon any critical/warning or quantized event (like battery depletion). Alert data are used by the sink in the evaluation of the fitness parameters.

(g) Memory - Limited memory is required to collect the data payload related to sensing, test-data, or route-queues. A buffer overflow can cause the

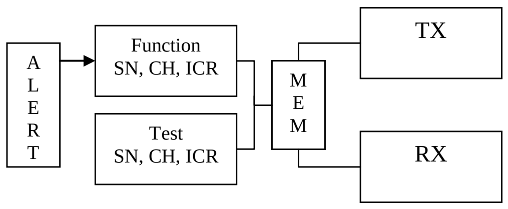


Figure 2.5: Illustration of sensor construction.

packets to be dropped. A temporary overflow memory allocation can be received from the neighboring node that belongs to the same cluster. Such allocation can be expensive as it comes at the cost of transmission overhead.

(h) Adaptive duty cycling - All ICR nodes put themselves in a low-duty-cycle state when they are not required to transmit any data through them. A Wake-on-Wireless (WoW) signal [32] can wake a sensor from the low-power state to the high-power state.

These characteristics are required for proper functioning of the self organizing sensor network using genetic algorithms. Most of these characteristics are related to the functional adaptation of the sensor based on function allocation by the sink without interrupting the sensing operation.

2.6 Experiment

Experimental setup consists of 100, 225, 400, 625 nodes placed at random positions in a 30×30 space. Each of the nodes picks up a random coordinate between $(0,0)$ and $(30,30)$ and assigns itself an UUID and a random battery

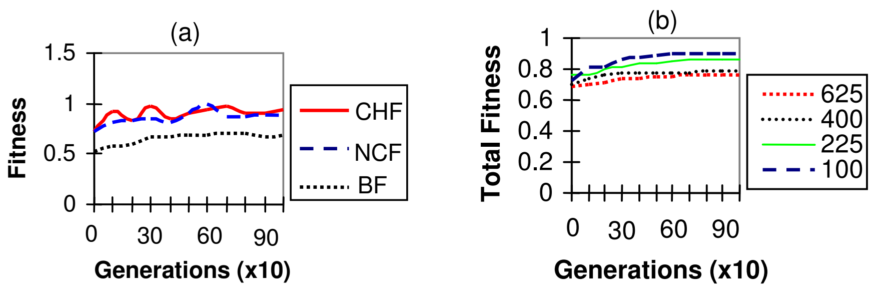


Figure 2.6: (a) Fitness chart for CHF/NCF/BF for 100 nodes; (b) Total fitness chart for 100, 225, 400, and 625 nodes setup (right).

capacity between 0 and 15. Once all the nodes have placed themselves in the listen mode, GA is run with the following parameters:

- Population size = 0.75(number of nodes)
- Crossover rate = 0.8($n -$ point cross-over)
- Mutation rate = 0.004
- Number of generations = 1000.

The experiment is simulated in an environment where each node acts as a Linux thread. Once GA run has completed, it assigns a function to each of these threads. These threads then start acting as independent nodes and initiate the node specific protocol. Each of these independent threads is capable of simulating battery depletion and transmission energy. In the experiment, it is assumed that there are no obstructions in the sensor transmit/receive path.

As seen in Fig. 2.6(b), convergence points are dependent upon the number of nodes being optimized. In all four cases convergence is reached within the first 500 generations. After that point improvement in the fitness is minimal. We

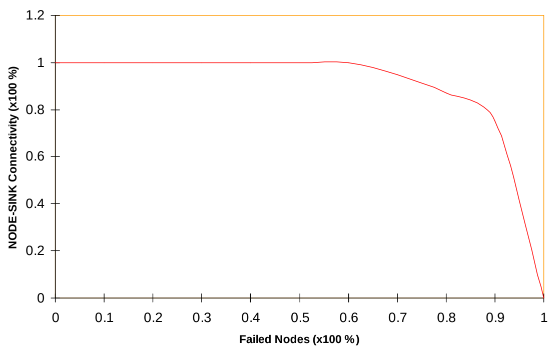


Figure 2.7: Percentage of nodes connected to the sink in the event of node failure.

can call this an 80% fitness point. After this point we may use a deterministic approach to achieve further fitness. As seen in Fig. 2.6(a), cluster-head fitness, node-communication fitness, and battery fitness increase monotonically with the number of generations. The same is true for the total fitness (Fig. 2.6(b)), which is a function of all the individual fitness. Also, as seen in Fig. 2.7, complete connectivity between non-extinct nodes and the sink can be maintained until 65% of the nodes die. While the death of sensors will reduce the coverage, the presence of efficient routing will reduce the number of orphan nodes.

Another important data point is the effect of isotropic communication range R_c of sensor nodes on the average power consumption as a result of cluster setup. Since the clustering decision is based on the density of the sensor deployment and the non-overlapping communication distance, a non-optimal distance can be expensive. This is due to the fact that the non-optimal distance will cause multiple collision, extra address bits (PUID) for unique identification in a denser environment and increased messaging. As seen in Fig 2.8, the optimal communication distance is reached at the minimum point in the valley. If the

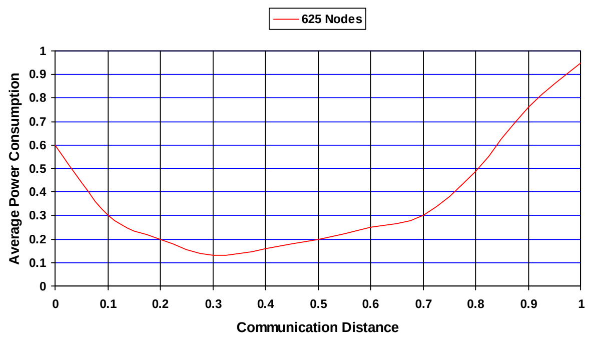


Figure 2.8: Average power consumption of the network with respect to variation in the communication distance R_c of the sensor nodes.

distance is less than the optimal, it requires more sensors to fulfill the coverage requirements. At the same time, we also observe an increased ICR messaging due to increase in the number of hops (ICR). If the distance is more than the optimal, then it reduces the coverage by turning off the neighboring sensors. This in turn increases the communication traffic due to decrease in data aggregation and fusion.

Longterm network sustainability depends upon the function distribution based on the total residual energy of the nodes. A bad allocation create pockets of no connectivity or connectivity with limited coverage. A sub-optimal function allocation can also cause frequent re-clustering and a practically unstable system. This can also result in non-useful energy expenditure. Average residual energy of the cluster-head is good measure of the effectiveness of the functional allocation of the nodes that tries to optimize the available power in the system. Re-Clustering process ensures that a node functioning as a cluster-head is re-provisioned with a different function with less energy requirements. This mechanism prevents CH from depleting its energy levels quickly by exchanging

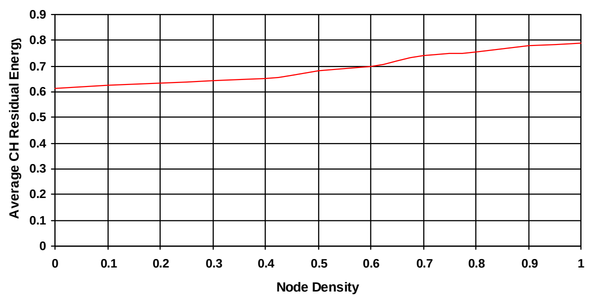


Figure 2.9: Normalized plot of node density (100-625 Nodes) and the average residual energy of the cluster-head (CH) over the period of full connectivity.

the roles with its neighboring nodes. In Fig 2.9, node population is increased from 100 nodes to 625 nodes for the same area. Average residual energy of the cluster-head (CH) varies between 61% to 79% of its total energy. For each population, measurements are made at regular intervals till the system loses its connectivity (Fig 2.7). In all 18% change in residual energy is recorced for 600% change in the node density.

Genetic Algorithm (GA) algorithm tries to balance the routes by optimally generating new routes based on the overall fitness (Section 2.2.4). This increases the overall lifetime of the system. ICR, CH & NS roles are exchanged by performing GA re-evaluation using a static timer or NODE alerts. This re-evaluation tries to re-balance the energy allocations based on the functional requirements as well as the historical traffic patterns, although with a setup transmission penalty. This penalty decreases with the overall system usage, because SINK is able to calculate the NODE parameters based on the side-band information (Section 2.4).

Fig 2.10 shows the effect of re-clustering timer (Re-Evaluate GA) on the over-

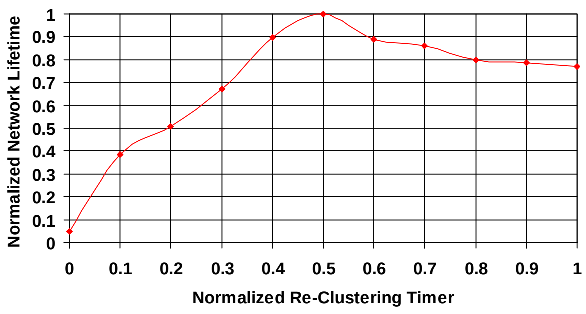


Figure 2.10: Normalized plot of Re-Clustering timer and the effect on the lifetime of the sensor network. Lifetime is measured from the start to the point where first sensor becomes unusable.

all life expectancy of the network. Once the optimal setup is reached, we don't observe any significant improvement in the life expectancy. At the same time we see a slow degradation in the life expectancy because of the increased transmission overhead. Therefore GA Re-Evaluation can be optimized by adapting the timer period based on the fitness matrix. We can evaluate the Periodicity based on the following equation.

$$\begin{aligned} Period &= K(1-(\alpha_1 \sum_{n=t-x}^{n=t} |TNF_n - TNF_{n-1}| \\ &\quad +\alpha_2 \sum_{n=t-x}^{n=t} |TRF_n - TRF_{n-1}|)/x). \end{aligned} \tag{2.15}$$

where,

K = Maximum Timer period, t = Current Instance, x = number of past instances.

TNF_n = Total Node Fitness at instance n of the GA re-evaluation (Equation 2.10).

TRF_n = Total Route Fitness at instance n of the GA re-evaluation (Equation 2.11).
α_1 and α_2 are the relative contributions and $\alpha_1 + \alpha_2 = 1$.

2.7 Summary

In this chapter we have presented a novel approach to design a self-organizing network based on genetic algorithms. Sensors that are placed at random are assigned functions (sensing node, cluster-head, router, or inactive-node) based upon the results of GA. The GA approach optimizes the network to maximize energy usage along with battery conservation with route optimization. It can be shown that the periodic run of a genetic algorithm will help conserve the overall energy of the system with maximum operability. As it can be seen from Fig. 2.6(b), individual components tends toward maximizing their fitness with the passing generations in a uniform manner. That shows that the goal of maximizing the system fitness along with individual component fitness can be achieved with a considerably reduced complexity. The algorithm also prevents the over-optimization of an individual fitness component at the cost of other components. One of the challenges in GA is to be able to converge in the shortest time possible. As an extension of this chapter, we will show the applicability of demand-based, mixed model where we run GA until convergence and then run traditional algorithms (e.g., TABU, directed diffusion, etc.) to achieve the target fitness. We will also research the prediction of system usage and the resulting topologies based on historical trends. The derivatives of these trends can then be used to define an individual fitness along with the current fitness parameters which will improve upon the uniform sensor-node usage assumption. As a part of future research, we will continue to work on improvements related to the security and the corresponding overhead. We also plan to address the challenges involved in the identification of domain boundaries in large networks which can be partitioned into multiple small network domains capable of performing GA. Another aspect that needs research is the ability to reduce the ICR traffic on cross-cluster aggregation (or fusion). While aggregation parameters are conveyed to the cluster-heads using sink mediation, this is less effective in a fast-changing environment because of overhead traffic originating from the sink.

Dynamic optimization of secure mobile sensor networks: A genetic algorithm

— Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen —

In Communications, 2007. ICC'07. IEEE International Conference on, pp. 3413-3418. IEEE, 2007

Chapter 3 – Dynamic Optimization of Secure Sensor Networks

Abstract

We propose a reduced-complexity genetic algorithm for secure and dynamic deployment of resource constrained multi-hop mobile sensor networks. Mobility and security are relatively expensive operations since they involve both communication and computation. Furthermore, these operations have to co-exist with optimal node and route assignments. The goal is to achieve optimal secure coverage and improved battery life using dynamic re-locatability. The genetic algorithm is used to adaptively configure optimal position and security attributes by dynamically monitoring network traffic, packet integrity, and battery usage. This results in minimization of the power consumption of the sensor system while maximizing the sensor objectives (coverage and exposure).

3.1 Introduction

Low-cost integration and small-size micro-sensors [16–20] have generated significant interest in the area of disposable sensors. These are motion capable, randomly deployed, infrastructure-less, data-centric sensors equipped with data processing capabilities and sensory circuits that cannot be charged (or rarely charged) or replaced. These sensors are constrained in energy, bandwidth, storage, and processing-capabilities and find their uses in the areas of homeland-security, disaster-recovery, target-identification, reconnaissance, medical applications, defense applications [22], and intrusion-detection, etc. Individual sensors process the sensory data and transmit to the target (sink) in a secure manner. Mobility reduces communication overhead (maximize the battery and sensor's life) by relocating these sensors and helping set up energy-efficient route for non-redundant secure data transmission from source to the sink. Although mobility and secure routing have been widely researched for ad hoc networks, it is still an unexplored area for resource constrained mobile sensor networks.

In this chapter we develop an evolutionary algorithm [4] that divides and positions the randomly deployed mobile sensors into an optimal number of independent clusters with cluster-head and optimal route. Once deployed, these sensors further maximize their coverage by moving (or re-orienting) themselves at the expense of battery life. Cluster-head collects data from its member sensors and sends them to the sink in a compressed and secure manner via the most cost-effective router. Sensors may be deployed in a hostile environment and may require enablement of security attributes adaptively based on observed data integrity and battery usage.

Genetic algorithm (GA) is a stochastic search technique that mimics the natural evolution proposed by Charles Darwin in 1858. GA has been successfully applied to a wide range of combination problems. They are particularly useful in applications involving design and optimization, where there are large numbers of variables and where procedural algorithms are either non-existent or extremely complicated. Simple GA converges to a single solution.

This chapter extends work on self-organization of static sensor networks [33]

(Chapter 2) by adding mobility and security to improve data integrity, coverage, and network life. The goal is to develop a long-lasting secure sensor network containing mobile nodes with non-renewal and limited energy resource. To achieve this goal we discover clustered topology, optimal locality with optimal routes to the sink. These clusters have the ability to fuse the collected data at the cluster head, which are then routed to the sink using one or more hops.

3.2 Related work and motivation

Mobile sensor networks consist of randomly deployed disposable sensors where configurable objectives cooperate with one another to maximize coverage and battery life. At deployment, sensors could diffuse into the environment via random-walk. In this chapter we use four competing objectives that create an energy-efficient sensor network: (a) *Cluster membership* that keeps on changing because of dead or depleted nodes, (b) *Routes to sink* that keeps on changing to avoid high-cost paths (like multiple clusters using the same inter-cluster router to route data to the sink), (c) *Sensor position* that dynamically adapts based on predicted optimal coverage, node traffic, and overhead traffic. The optimal position is predicted for cluster heads, routers, and sensor nodes based on factors that constitute the fitness function, and (d) *Sensor security* that dynamically enables the security attributes based on security threat and battery usage. Overall, the sensor network relies on continuous random motion to bring nodes into optimal contact for various reasons such as security, the shortest path for clusters-heads, and load migration, etc.

Previous work related to mobile sensor networks include dynamic approaches where sensor nodes are deployed one at a time, with each node making use of data gathered from previously deployed nodes to determine its optimal deployment location [34], potential fields to reduce deployment time [35], self-organization strategies and algorithms for responsive adaptation of sensor nodes to coverage of a field with multiple dynamically changing contexts [36], optimal deployment of sensors toward critical region to ensure quality of the readings of the value of interest [37]. Work related to sensor security include a solution [38, 39] using simple symmetric cryptographic algorithms. Asymmetric cryptographic algorithms are not suitable for providing security on wireless sensor networks due to limited computation, power, and storage resources available on sensor nodes. Although most of the schemes described above are promising, they do not deal with the sensor networks holistically that require optimization of the competing objectives (clustering, positioning, routing and security) for a high energy efficiency.

This chapter extends the GA approach of Chapter 2 [33] that introduced a multi-objective genetic algorithms [23] (MOGA) approach for achieving the first two objectives, i.e., *cluster membership* and *routes-to-sink*, for static sensors. The bulk of the work done focused on maximizing the coverage while minimizing the battery usage in stationary sensor networks. For problems where there are several, often conflicting objectives, an MOGA is used which evolves a set of solutions (the population) towards the Pareto-optimal front where trade-off analysis can be performed to select a suitable solution. This chapter introduces an approach to deal with a more complex problem where secure coverage per unit of power of motion-capable sensors is maximized by analyzing two additional objectives: *sensor position* and *sensor security*, using secure protocols and locomotive abilities of these sensors. These objectives help generate optimal parameters related to (a) resolving routing imbalances, (b) optimal sensor allocation for various functions, (c) resolving load imbalances, (d) reducing overhead traffic, (e) optimal positioning of sensors to avoid shadowing effect, redundant usage, sub-optimal clustering, (f) load migration, and (g) security overhead for secure communication. Data security provides a unified and efficient scheme for maximum reliability and privacy. Mobility on the other hand provides an ability to re-position (or self-repair) the sensor (nodes, routers, cluster-heads) strategically so as to maximize the overall objective (cluster membership, security overhead, and routes) with an extra degree of freedom. However motion costs battery life and therefore sensors cannot be moved very frequently. Therefore, an efficient dynamic re-positioning and security uses long-range prediction based on historical trends or generational improvement over a period of time with the primary goal of maximizing the coverage in a resource constrained environment.

3.2.1 Representation of Static Sensors

As a part of our previous work, each individual sensor node is allocated a functional assignment using using genetic algorithm. These functions are represented as (a) inactive node (powered off), (b) cluster-head (CH), (c) inter-cluster router (ICR), and (d) sensor node (NS). Each cluster is represented by a cluster-head, and cluster-members are represented by inactive/active node sensors and ICRs. Cluster-head is responsible for data-fusion from various node-sensors and inter-cluster router is responsible for routing cluster data (from cluster-head) to the sink. In later sections we will introduce the mobility and security aspect to the

GA fitness function along with its chromosome representation and co-existence with existing fitness parameters of importance. Algorithmic details regarding clustering, naming, routing using GA can be found in [33] (Chapter 2). The fitness parameters defined in the chapter 2 for optimal clustering of static sensor networks are:

1. **Coverage Fitness (CF)** optimizes the blanket coverage with an objective to maximize the total detection area.

2. **Cluster-Head Fitness (CHF)** defines the fitness based on the uniformity of the sensor nodes and cluster-heads.

3. **Node Communication Fitness (NCF)** defines the power required to communicate with the cluster-head that can be computed using the path loss.

4. **Battery Status Fitness (BF)** defines thresholds used to optimize node assignments w.r.t. battery status/usage.

5. **Router Load Fitness (RLF)** penalizes routers (ICR) if they cater to more than the average number of cluster-heads and ICR to avoid overloading.

6. **Sensor Effector Fitness (SEF)** interprets the power consumed by the sensory action of clusters. The net effect of SEF is to re-arrange the sensor nodes such that the sensor data transmission is uniformly optimized by fusion, elimination or compression methods.

7. **Total Node Fitness (TNF)** is evaluated in the GA algorithm for the

appropriate node assignment as

$$\text{TNF} = \alpha_1\text{CHF} + \alpha_2\text{NCF} + \alpha_3\text{BF} + \alpha_4\text{RFL} + \alpha_5\text{SEF} + \alpha_6\text{CF} \tag{3.1}$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 = 1$ and α_i depends upon the relative significance of the component. These values can be made adaptive using an external heuristics.

8. **Route Selection Fitness Function (RSFF)** generates balanced routes based on node allocation using GA based on node fitness function. During setup operation, both CH and ICR start sending data on the most cost effective routers.

3.3 Mobility Extensions

In this section we will discuss locomotive details of the GA with a goal to achieve energy efficient deployment. In the GA modeling, TNF and RSFF cost functions are expanded with the parameters of the locomotion architecture and encryption algorithms explained later in Section 3.4 that affect the transmission process such as available bandwidth, network bandwidth, packet size, CPU power consumption, RF distance, optimal routing, and protocol overhead. The final optimality equation is derived for the optimization and encryption decision process which is implemented by GA.

Mobility allows the nodes to seek out power optimization, request data fusion from other nodes to perform cooperative sensing, seek out repair, and locate data portals from which to report. But mobility comes with a price as locomotion is costly in terms of node size and power consumption. An optimal locomotion strategy is achieved by sensor node's ability to monitor its own power as well as its interaction with environmental dynamics. Locomotion is employed to maximize the fitness by rearranging its position to achieve the optimality of the parameters defined in the following sections.

3.3.1 Coverage Uniformity Fitness (CUF)

CUF expresses the coverage improvement by filling the coverage holes and maximizing the detection area using sensor movement. This is done by re-balancing the communication distances between member nodes of the cluster. Closely placed nodes are rewarded for moving away while farther nodes are rewarded for moving toward each other to attain coverage equilibrium. When the distances between nodes become optimal, the distance to the farthest neighboring node and the required transmission power are minimized which helps maximize the NCF. CUF is expressed as

$$\begin{aligned}\mathrm{CUF} = 1 - \frac{1}{2M}\sum_j &\min\left(1,\ |d_{j\_\mathrm{min}} - d_{j\_\mathrm{mean}}|/d_{j\_\mathrm{mean}}\right) + \\ &\min\left(1,\ |e_{j\_\mathrm{min}} - e_{j\_\mathrm{mean}}|/e_{j\_\mathrm{mean}}\right) \qquad (3.2)\end{aligned}$$

where M is the number of clusters, d_{j_min}, d_{j_mean} are, respectively, the minimum and mean communication distances between nodes in cluster j, and e_{j_min} and e_{j_mean} are, respectively, the minimum and mean communication distances between nodes and cluster-head in cluster j.

Uniformly distributed sensor nodes spend energy more evenly than nodes with an irregular topology. GA exploits the sensor motion ability by positioning the sensors in a manner to increase the coverage, reduce the inter-node interference, and minimize the power required to communicate.

3.3.2 Cluster-Node Migration Fitness (CNMF)

CNMF aids in improving the uniformity of sensor nodes and cluster-heads by rewarding the migration of sensor nodes between cluster-heads with low CHF. Migration helps to achieve higher CHF if sensor migration is from high-density clusters to those with lower density. Cluster-node migration fitness can be expressed as

$$\text{CNMF} = \frac{1}{2N} \sum_{n}^{N} (\chi_{ns} + \chi_{nt}) \tag{3.3a}$$

$$\chi_{ns} = \min(1, \max(-1, (\rho_{ns} - \rho)/\rho)) \tag{3.3b}$$

$$\chi_{nt} = \max(0, \min(1, (\rho - \rho_{nt})/\rho)) \tag{3.3c}$$

where n is the n-th migration pair (source-target cluster), N is the total number of migration pairs, χ_{ns} is the source cluster's measure of excessive number of sensor nodes, χ_{nt} is the target cluster's measure of depleted number of sensor nodes, ρ_n is the number of nodes attached to this cluster-head, and ρ is the average number of nodes per cluster in a system calculated as

$$\rho = \text{Total Sensor Nodes/Total Cluster Heads.} \tag{3.4}$$

The fitness expression rewards the migration of sensor nodes if they reside in the low CHF clusters with high diffusion gradient between source and target clusters.

3.3.3 Cluster-Head Migration Fitness (CHMF)

CHMF rewards movement of the cluster-head and inter-cluster routers with lower router load fitness. Movement of the CH and ICR can help attain higher RLF due to the following:

1. ICR or CH movement can change the membership of the ICR based on various factors defined in [33] (Chapter 2). This can result in optimizing the number of CH/ICR attached to the ICR that was moved.
2. ICR can also move to exchange roles with another functional node (cluster-head, sensor node). This can help maintain the existing topology by exchanging the nodes with higher battery capacity for router purposes (and exchanging the functional objectives).

Cluster-head migration fitness is expressed as

$$\mathrm{CHMF} = \frac{1}{N}\sum_{n}^{N}\frac{1}{1+\eta_n}((1-\mathrm{RLF}_n)+\eta_n(1-\mathrm{BF}_{ns}+\mathrm{BF}_{nt})) \tag{3.5}$$

where N is the total number of nodes-in-motion, RLF_n is the router load fitness (Section 3.2.1) of n-th node, BF_{nt} is the battery fitness (Section 3.2.1) of non-ICR node that is exchanged with n-th ICR node with BF_{ns}, η_n is the boolean that represents the presence of exchange pair for the n-th ICR.

It is evident from (3.5) that sensor movement is rewarded on ICRs and CHs with lower battery and router load fitness. Node movement influences the router's load by re-balancing and the battery capacity by exchanging functional objectives.

3.3.4 Node Motion Fitness (NMF)

The average distance traveled by a node is related to its movement at the expense of battery life. So, the expected distance is an important estimate of energy required for nodes with limited energy supply. Hence it is desired to stabilize the motion characteristics while achieving the overall system objectives (coverage and longer network life). These characteristics are related to motion frequency and oscillations.

1. *Motion Frequency* measures an average movement of the sensor in a given amount of time bounded by a threshold which is a function of the battery life defined by battery fitness. Larger movements of sensors with limited battery life is penalized which makes it highly prohibitive to achieve locomotion as the system ages.

2. *Location Stability* measures an inability of nodes to attain stable position due to competitive objectives. Nodes are penalized for having excessive movement or un-sustained oscillations.

Node motion fitness can be expressed as

$$\mathrm{NMF} = ((1 - F_i(Q, \mathrm{distance})) + (1 - \phi_i(n)))/2 \tag{3.6}$$

where $\phi_i(n)$ is the i-th sensor node's penalty measure for visiting the same location for n times ($0 \leq \phi_i(n) \leq 1$), $F_i(\cdot)$ is the i-th sensor node's penalty with $0 \leq F_i(Q, \text{Node Type}) \leq 1$, Q is the battery status represented in quantized steps, *distance* is the estimated distance traveled by the node which is estimated indirectly using energy-based localization based on multiple energy reading at different known sensor locations.

The signal energy measured on the i-th sensor over a time interval t, denoted by $y_i(t)$, can be expressed as

$$y_i(t) = \frac{G_i.S(t)}{|\boldsymbol{r}(t) - \boldsymbol{r}_i|^{\alpha}} + \epsilon_i(t) \tag{3.7}$$

where G_i is the gain factor of the i-th sensor, α (≈ 2) is an energy-decay factor, and $\epsilon_i(t)$ is the cumulative effects of the modeling error of the parameters, $S(t)$ denotes the energy emitted by the target at time t, $\boldsymbol{r}(t)$ is a $D \times 1$ vector denoting the coordinates of the target at time t, $\boldsymbol{r}_i$ is a $D \times 1$ vector denoting the cartesian coordinates of the i-th stationary sensor.

3.3.5 Sensor Data Fitness (SDF)

SDF measures sensor data efficiency with the net effect to re-position the sensor node such that its data transmission is uniformly optimized by fusion, elimination, or compression methods. This is further improved by optimizing the quality of sensing for a given SNR. Optimal sensing in a resource constrained (communication, battery, etc.) can be represented by $\theta(B, F)$, where B is the QoS requirements related to sensing operation and F is the timer policy. While QoS property is implemented to take advantage of variable data compression and fusion rules, a timer is implemented to vary the bit-rate depending upon conditions (density of sensors, etc.) of the sensor. Sensor movement is rewarded by reducing the average energy requirements in a cluster by:

1. Reduced variance in the timer activity due to load sharing by the recently moved sensor.

2. Reduction in the number of bits because of new fusion rule triggered due to recently moved sensors, and because of elimination of redundant sensing due to movement of redundant sensors.

Net result of the reward process is the optimal sensor density and bits per second for a given SNR. SDF is expressed as

$$\text{SDF} = \frac{1}{N}\sum_{n}^{N}(\lambda_1\psi(F,n) + \lambda_2\psi(B,n)) \tag{3.8a}$$

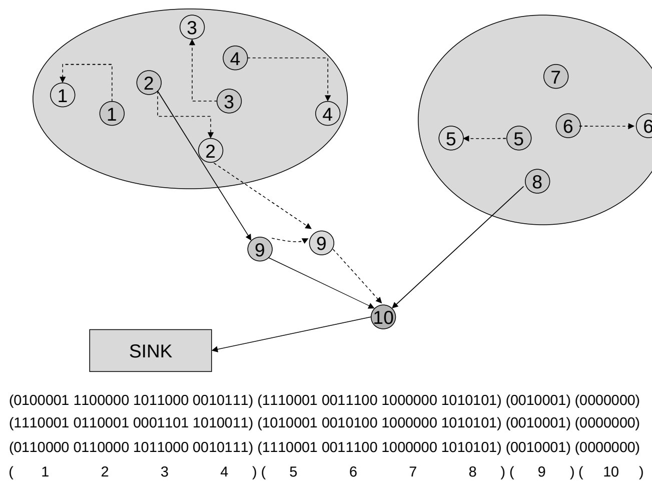
$$\begin{aligned}\psi(X,n) = &\min\left(1, \max\left(0, \frac{X_\mu^n(s-1) - X_\mu^n(s)}{X_\mu^n(s-1)}\right)\right) + \\ &\min\left(1, \max\left(0, \frac{X_\sigma^n(s-1) - X_\sigma^n(s)}{X_\sigma^n(s)}\right)\right)\end{aligned} \tag{3.8b}$$

where $\lambda_1 + \lambda_2 = 1$, $F = \{F_1, F_2....F_N\}$, and $B = \{B_1, B_2....B_N\}$ represents the average frequency and bit rate of each sensor node of the cluster n in which sensor node movement has been detected, $\psi(X,n)$ represents the improvement gain by a sensor parameter X represented by change in its mean(X_μ^n) and variance(X_σ^n) between consecutive sampling instances (s) in cluster n, λ_1 and λ_2 can be adjusted based on the sensor implementation.

The total fitness associated with node movement is given by total node motion fitness

$$\text{TNMF} = \alpha_1\text{CUF} + \alpha_2\text{CNMF} + \alpha_3\text{NMF} + \alpha_4\text{CHMF} + \alpha_5\text{SDF} \tag{3.9}$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 = 1$ and individual weight is dependent upon implementation.



```
(0100001 1100000 1011000 0010111) (1110001 0011100 1000000 1010101) (0010001) (0000000)
(1110001 0110001 0001101 1010011) (1010001 0010100 1000000 1010101) (0010001) (0000000)
(0110000 0110000 1011000 0010111) (1110001 0011100 1000000 1010101) (0010001) (0000000)
(    1       2       3       4   ) (   5       6       7       8   ) (   9   ) (  10   )
```

Figure 3.1: Node re-positioning as a result of genetic algorithm. In this example, nodes 1, 2 undergo 3 replacements; nodes 3, 4 undergo 2 replacements; nodes 5, 6, 9 are replaced only once. Other nodes do not move.

3.3.6 Node Placement Genetic Algorithm

With the TNMS, we can design the algorithm for optimal node deployment using the GA operators. The GA executes in the sink or a similar centralized entity, where it repeats upon multiple triggers. These triggers are related to battery alert, deteriorating route fitness alert, and periodic action. Once the optimal fitness is achieved, the deployment corresponding to that fitness is committed and the sensors are instructed to assume the new positions by relinquishing the old positions.

(a) **Chromosome Representation**: The chromosome of the GA is the building block to a solution of the problem at hand in a form that is suitable for the genetic operators and the fitness function. Chromosome string is formed using each individual sensor node's motion vector represented by a 7-bit binary number called 'gene' as shown in Fig. 3.1. The chromosome string hierarchy can be defined as
$((\widehat{\theta}x\theta x\widehat{S}xS)_1(\widehat{\theta}x\theta x\widehat{S}xS)_2(\widehat{\theta}x\theta x\widehat{S}xS)_3......)_1$
....
$((\widehat{\theta}x\theta x\widehat{S}xS)_1(\widehat{\theta}x\theta x\widehat{S}xS)_2(\widehat{\theta}x\theta x\widehat{S}xS)_3......)_n$
where $(\widehat{\theta}x\theta x\widehat{S}xS)_i$ represents the motion vector with the following properties:

   (a) $(\widehat{\theta}\theta)$ represents $0^o(00)$, $90^o(01)$, $180^o(10)$, $270^o(11)$ angular movement

   (b) $(\widehat{S}S)$ represents number of finite steps the sensor travels in the direction given by angular movement

   (c) Sensor is moved only if one of the x values is 1.

(b) **Initial population**: Initial chromosome strings are seeded partially randomly using a random number generator (RNG) and partially using the population of previous samples. Population uses the gene structure as defined in Section 3.2.1. This population is coded with gene structure as defined in Section 3.2.1.

(c) **Evaluation**: Each chromosome string is evaluated for the fitness using the TNMF function (for node placement) as defined by Eq. (3.9).

(d) **Reproduction**: Reproduction allows individuals (strings) with larger fitness to have a higher probability of contributing an offspring in the next generation. Since the TNMF defines the fitness value, the chromosome with the highest TNMF value has a better chance to take part in reproduction. The algorithm uses the standard weighted roulette wheel method to select n individuals to the mating pool that produces N chromosomes using a crossover probability. During reproduction, we choose multiple cross-over points whose locations are calculated using an RNG.

(e) **Mutation**: Reproduced N chromosomes are transferred to the mutation pool where the mutation operator mutates them according to adaptive mutation probability which is inversely proportional to the average fitness. We will choose a maximum mutation probability p_m.

$$p_g = p_m(1 - (N * \mathrm{TNMF}_{\mathrm{avg}})/\mathrm{NMF}_{\mathrm{total}}). \tag{3.10}$$

Mutation uses function flip (toss of a coin) to decide whether to invert the bit or not.

(f) **Selection**: Finally n chromosomes are chosen out of $N + n$ (n parent and N children) according to their fitness values and are carried over to the next generation.

3.4 Security Extensions

So far we have defined three sensor objectives that execute in parallel using genetic algorithms. The first two objectives i.e., TNF and RSFF are defined in [33] (Chapter 2) and the third objective, NMF, is defined in Section 3.3. The reliable functioning of these three objectives depend on the secure communications between various functional elements (nodes and sink). This requires identifying compromised or falsely added nodes, secure re-deployment/addition of nodes, and preventing passive listening by a malicious intruder using elements of authentication, integrity, privacy (or confidentiality), and anti-playback. All communications need to be secure to avoid data intercept, analysis and alteration by an intruder who can device methods to reduce the effectiveness of the sensor network. This has to be done in a manner such that time required to circumvent the security measures using brute-force methods takes longer than the life of the network.

In our security model, the sink is considered a trusted component that establishes a necessary trust relationship for secure forwarding of data between various node types. Nodes closest to the sink form the most trusted relationships. Farther nodes build the hierarchy of trust starting from the sink which is apparent from the pre-determined routing decisions that are created during setup and later during re-configuration [33] (Chapter 2). Ingredients of security architecture create a trust relationship between various node-types for the reasons related command/message execution, data forwarding, etc. Any authentication is mediated through sink and components of the trusted routing hierarchy. To achieve a power efficient authentication we employ certain elements of secure network encryption protocol (SNEP) [39]. Encryption portion of the protocol is executed between first-initiator (FI) and the sink with other components in the hierarchy acting as an authenticated (or un-authenticated) pass-through. First initiator is a cluster-head or an ICR that either initiates a command or participates in data fusion for delivery to sink. Elements of security are:

1. ***Master Key (MK)*** derives keys for symmetric encryption (K_{encr}), message authentication (K_{auth}), and generates pseudo-random numbers (K_{rand}) [39]. The derived keys can be changed randomly upon request by the sink. The master key is shared between a node and the sink *a priori* and used for exclusive node-sink messaging. A pseudo-random number is generated using a derived key K_{rand} and a counter C. This number is inserted in the message before encryption to avoid plain-text attacks.

$$K_{\text{rand}}^{n+1} = \text{MAC}(K_{\text{rand}}, C^n) \tag{3.11}$$

2. ***Inter-Node Communication Key (INCK)*** is the sink-mediated shared key between two nodes that authenticates (INCK_{mac}) the messages between them. Since the sink is aware of the routing hierarchy, it encapsulates an INCK=$\{(\text{INCK}_{\text{mac}}^0), (\text{INCK}_{\text{mac}}^1)\}$ for each ICR (or CH) that takes part in authentication. Each node decrypts the encapsulated packet using its K_{encr} (derived from the master key) and extracts its INCK. $\text{INCK}_{\text{mac}}^0$ and $\text{INCK}_{\text{mac}}^1$ are the MAC keys used at ports 0 and 1, respectively.

3. ***Encryption and Authentication*** Similar to the SPIN protocol [39], we use counter-mode block cypher for encryption/decryption and CBC-MAC [40] for authentication. The counter-mode block cypher requires a shared counter between a node and the sink which is incremented on each block. Since it is a stream-cipher, the message length is the same as the plain text and hence a lower communication overhead. While some routers can be used as pass-through, other routers enforce admission control using MAC based authentication. Sink can change the authentication requirements of ICR and CH depending on energy requirements and perceived security threat as measured by the battery quantization levels and the number of bad packets.

Battery limitations and computational overhead prevent us from maintaining the same threat levels by employing encryption and authentication mechanisms on all nodes. Ideal enabling reduces the computational overhead while maintaining the adequate security levels by identifying the strategic nodes. Strategic nodes are optimally enabled for security by evaluating the battery status, network traffic, malformed or retries on a specific route and number of nodes in a single route handling authentication. For the purpose of GA, we evaluate a fitness function that competes for the optimal enablement of the security ingredients on the sensor nodes.

3.4.1 Secure Node Fitness (SNF)

SNF rewards the security enabling on nodes based on the perceived threat involving data integrity for secure communication. Sink keeps track of all the ill-formed packets received on a particular route. While routes (CH$\rightarrow$sink) are penalized for carrying malformed and retried packets, they are rewarded for enabling authentication on routers (ICR) and encryption on cluster-heads. Additional penalty is awarded if the authentication is enabled disproportional to threat level quantized to M levels. While system can react proportionally to the perceived threat, it may not be enabled in an energy efficient manner. SNF rewards energy efficient enablement of security attributes that are measured against battery quantization levels and rate of battery usage. Sink uses hysteresis to compute the battery usage that is indicative of data communication, average number of connecting nodes, and locomotion, etc.

$$\mathrm{SNF} = 1 - \frac{1}{2R}\sum_{i=1}^{R}\left(\left|\frac{\theta_i}{M} - \frac{\lambda_1 K_i + \lambda_2}{N}\right| + \sum_{n=1}^{N}\frac{I_i^n F_i^n(Q,\psi)}{N}\right) \tag{3.12}$$

where $\lambda_1 + \lambda_2 = 1$ with λ_2 being the reward contribution due to encryption of the first initiator (FI), R is the total number of routes, θ_i is the threat level of route i as calculated by sink, K_i is the number of nodes (ICR(s) and CH) that are enabled for authentication and encryption in route i, N is the total number of nodes (ICR(s) and CH) in route i, I_i^n=1 (else 0) if node n in route i is enabled for authentication, and $F_i^n(.)$ is penalty for enabling admission control on node i on route j that has battery level at Q and rate of battery usage at ψ.

3.4.2 Security Enablement Genetic Algorithm

In this section we design the genetic algorithm for enabling security attributes (authentication & encryption) on nodes. These nodes are represented with a

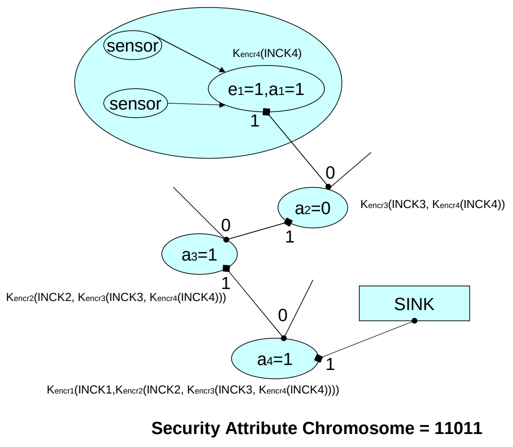

sensor
sensor
K_{encr4}(INCK4)
e_1=1,a_1=1
1
0
a_2=0
K_{encr3}(INCK3, K_{encr4}(INCK4))
0
1
a_3=1
1
K_{encr2}(INCK2, K_{encr3}(INCK3, K_{encr4}(INCK4)))
0
SINK
a_4=1
1
K_{encr1}(INCK1,K_{encr2}(INCK2, K_{encr3}(INCK3, K_{encr4}(INCK4))))
Security Attribute Chromosome = 11011

Figure 3.2: Sink mediates the INCK keys at ICR and CH ports. These keys enable the authentication based on security attribute chromosome generated by genetic algorithm (GA). For example, node 2 does not require message authentication at port 0, but requires at port 1. $\mathrm{INCK}^1_{\mathrm{mac2}} = \mathrm{INCK}^0_{\mathrm{mac3}}$.

chromosome string that is formed using each individual sensor node's security policy represented by a 2-bit binary number and defined as

$$(e_1a_1a_2..a_N)_1(e_1a_1a_2..a_N)_2..(e_1a_1a_2..a_N)_R$$

where $(e_1a_1a_2..a_N)_i$ represents the security attributes (e_n & a_n) on node n of route i, and e_n and a_i represent the encryption bit and authentication bit, respectively. It should be noted that the first node is always a CH where data encryption can optionally take place by setting $e_n\!=\!1$. Genetic Algorithm (GA) steps are similar to those defined in section 3.3.6.

Security settings competes with node-selection or locomotion. Nodes are assigned functions or locations based on the corresponding fitness factors which may be suboptimal for securing the packets due to battery conditions. This triggers re-configuration until all objectives reach an acceptable convergence. Like node/route selection, and mobility estimation, this is a dynamic process that repeats over system's life-time.

3.5 Results and Discussion

Experimental setup consists of 100 nodes at random positions in a 30×30 space. Individual node picks up a random coordinate between $(0, 0)$ and $(30, 30)$ and assigns itself an UUID and a random battery capacity between 0 and 15. For simplicity, each node is given a coverage area of 3×3. Once all the nodes have placed themselves in the listen mode, GA is run with the cross-over rate of 60% and an initial mutation of 6%. Experiment assumes line-of-sight propagation between sensor nodes. The software simulates the sink operation and runs in conjunction with NS-2 software that simulates the network traffic. It executes the GA that generates the motion path as well as security attributes. It also calculates the fitness parameters (Sections 3.3 and 3.4) based on network traffic, battery usage, and integrity of received packets. A separate process in the sink simulator runs a predictive algorithm that estimates the traffic and data integrity into the future using past hysteresis. This data is used to estimate the fitness parameters during the GA run. While each GA objective tends to compete with others to converge at the system equilibrium, the end result is to maximize the network life for optimal coverage.

Fig. 3.3 shows the coverage as a function of the number of generations for static and dynamic deployment. It is observed that coverage is increased to about 30% as a result of dynamic deployment due to locomotion. While cov-

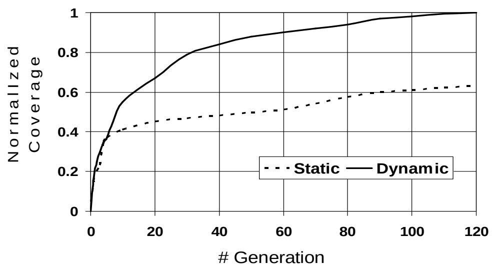


Figure 3.3: Coverage as a function of n-th generation for static and dynamic deployment.

erage is improved, energy cost may be increased due to sensor motion which affects the sensor-network life. Locomotion is accompanied with the communication overhead due to (a) encryption and authentication of motion commands and (b) temporary packet loss and data corruption due to node motion that triggers enhanced authentication attributes on the communication routes (GA Run). While battery cost of locomotion is compensated by communication cost reduction due to node-redeployment, it still reduces the overall benefit.

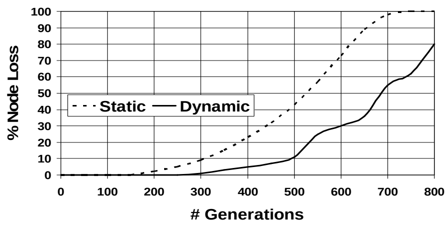


Figure 3.4: Percentage of nodes lost (due to battery) as a function of n-th generation for static and dynamic deployment (50% Threat Level, $\theta_i = 0.5$).

Fig. 3.4 shows the node loss versus of the number of generations. It is found that a dynamic deployment significantly outperforms the static one with a 15-20% reduction in the number of lost nodes. While nodes are lost exponentially in static deployment case, they die gradually in clusters for the dynamic de-

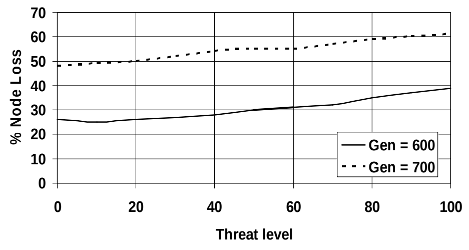


Figure 3.5: Percentage of nodes lost (due to battery) as a function of Threat level θ_i at the end of 600^{th} and 700^{th} generation.

ployment case due to better distribution of the total energy. Coverage loss due to death of statically deployed nodes results in increased transmission energy and longer routes. Genetic algorithm enhances the coverage, life and integrity of the sensor network using the security and mobility extensions in addition to optimal node assignments. Furthermore, security extensions promotes added improvement over existing methods by dynamically switching authentication and encryption based on threat levels (Fig. 3.5) as perceived by SINK. Energy savings are realized due to reduced computation and header-data overhead on safe nodes (CH and ICR).

3.6 Conclusion

This chapter presents secure, dynamic, and energy-efficient deployment of mobile sensors using a multiple-objective genetic algorithm. This approach maximizes coverage and network life by exploiting mobility which optimally relocates sensor node that further optimizes node assignments, route and security attributes. We observe incremental improvement over static deployment that involved optimal functional and route assignments using GA [33] (Chapter 2). An interesting outcome is the regional uniformity of the communication distances proportional to the sensor activity in that region. We observe a better distribution of energy among various functional nodes attributed to an extra degree of freedom that relocates the node strategically to achieve better battery utilization (fitness). This also reduces frequent re-clustering because now the roles are exchanged by just exchanging the positions while maintaining fitness parameters in equilibrium. Additionally, we develop a novel approach of adapting the security attributes proportional to the perceived threat and in a manner that promotes efficient battery usage and minimizes the effects of aberrant nodes. We will investigate the ability to exclude aberrant nodes from the network. Furthermore, we will also investigate effects of activity migration between hot and cold regions as well as better characterization of energy distribution over network life.

Reduced complexity intrusion detection in sensor networks using genetic algorithm

— Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen —

In Communications, 2009. ICC'09. IEEE International Conference on, pp. 1-5. IEEE, 2009

Chapter 4 – Intrusion Detection in Sensor Networks Using Genetic Algorithm

Abstract

We propose a reduced-complexity genetic algorithm for intrusion detection of resource constrained multi-hop mobile sensor networks. Traditional intrusion detection mechanisms have limited applicability to the sensor networks due to scarce battery and processing resources. Therefore, an effective scheme would require a power efficient and lightweight approach to identify malicious attacks. The goal of this paper is to evaluate sensor node attributes by measuring the perceived threat and its suitability to host local monitoring node (LMN) that acts as trusted proxy agent for the sink and capable of securely monitoring its neighbors. Security attributes in conjunction with genetic algorithm jointly optimizes the placement of monitoring nodes (i.e., LMN) by dynamically evaluating node fitness by profiling workloads patterns, packet statistics, utilization data, battery status, and quality-of-service compliance.

4.1 Introduction

Disposable sensors are constrained in energy, bandwidth, storage, and processing capabilities [16–20]. They are widely used in the areas of homeland-security, disaster-recovery, target-identification, reconnaissance, medical applications etc.

Each sensor processes the sensory data and transmits to the target (sink) in a secure manner. Since sensor networks operate in an unknown hostile environment and could carry sensitive information, it is essential to implement measures to detect an external attack targeted to destabilize the network. Thus we need secure communications to avoid data intercept, analysis, and alteration by an intruder who can device methods to reduce the effectiveness of the sensor network. However, security of a sensor network presents many challenges due to the hostile deployment, changing topologies, limited compute and battery resources, and competing objectives.

Intrusion is a pattern of an observed sequence and intrusion detection systems (IDS) detect security violation patterns on a system by monitoring and analyzing activity trends. It is similar to an immune system that identifies and eliminates anomalies by measuring deviations from the normal processes using distributed identifiers over the system with identifiable and adaptable relationship. Sensor networks have their unique characteristics. First, disposable sensor nodes are made up of low-cost and resource constrained hardware with little or no monitoring and logging abilities. This presents challenges in maintaining and retrieving the logged data. Second, the constantly changing topologies and volatile physical environments make it difficult to discriminate between an intrusion and normal operations. Because of the resource limitations, frequent node failures, and highly distributed nature, intrusion detection must performed in an energy efficient manner such that time required to circumvent the security measures using brute-force methods takes longer than the life of the network. Consequently, traditional IDS methods cannot be applied directly to sensor networks.

This chapter extends previous work that used a genetic algorithm (GA) based mechanisms [4] to divide and configure the randomly deployed sensors into an optimal number of independent clusters with cluster-head and optimal route [33] (Chapter 2). Cluster-head collects data from its member sensors and sends them to the sink in a compressed and secure manner via the most cost-effective route. The proposed enhancement in this paper uses GA based optimal selection of high-confidence local monitoring nodes (LMN) that act as sink proxy for distributed observability of the problem nodes. These nodes provide additional observability of the network behavior derived from the analysis of sensor events profiled in its neighborhood. The events we consider are: data message patterns, message collisions, route traffic activity trends, sensor positioning, and sync events that update the changes in the node profiles.

Genetic algorithm [4] is a stochastic search technique that resembles the natural evolution. They are particularly useful in applications involving design and optimization where there are a large number of variables and where procedural algorithms are either non-existent or extremely complicated. The potential of GA as a global optimizer permits undertaking complex optimization problems and therefore allows for greater degrees of freedom in the selection of the model's structure.

For sensor networks with a large number of nodes, it is time consuming to evaluate all possible solutions to LMN selection serially. An efficient scheme would position the minimum number of LMNs to achieve maximum coverage. Genetic algorithms reduce the complexity due to multiple offspring that come up with parallel solutions, which render them usable in real-time. Furthermore, multiple potential solutions aid in searching multiple points simultaneously and, therefore, avoids being caught in a local optimum. In general, GA can deal with highly nonlinear problems without a number of assumptions about the problem being solved.

4.2 Related Work and Motivation

Sensor networks consist of randomly deployed disposable sensors that cooperate with one another to maximize coverage and battery life. In this chapter we optimally select monitoring nodes that analyze the dynamic behavior of the sensor nodes for possible intrusion with a discrete set of observable using GA based fitness criteria. LMN nodes harden the estimation of integrity factor that results in challenge-response sync to isolate the aberrant nodes and optimal security attributes. Upon detection non-compliant nodes, triggers re-clustering event that isolates these nodes to fulfill various objectives [33, 41] (Chapter 2, 3) to create an energy-efficient sensor network by: (i) reassigning cluster membership, (ii) dynamic re-routing, (iii) and reassigning sensor security attributes. GA establishes a baseline of "‘normal"’ traffic between wireless sensor nodes over time and detects deviations that could be due to intrusion.

Various solutions have been proposed to traditional networks [42–45], but restrictions of wireless sensor network (WSN) resources make direct application of these solutions infeasible. It is impossible to have an independent IDS agent in individual node due to limited battery power. Moreover, sensor density is much higher than an ad hoc network and sensors have a high probability of failure or death due to battery constraints. For these reasons, WSNs require special handling in a more de-centralized, lightweight and energy efficient manner. Pires *et al.* [46] present an IDS method to detect message transmission as suspicious if its signal strength is incompatible with its originator's geographical position. Du *et al.* [47] present an efficient secure routing protocol that takes advantage of the powerful high-end sensors and can defend typical routing attacks. Silva *et al.* [48] propose an IDS scheme based on the inference of the network behavior obtained from the analysis of events detected by a monitor node. Agah *et al.* [49] propose a game-theoretic approach and compares it against hidden Markov model (HMM) and intuitive metric approach that increases the chance of success in defense strategy for sensor network. Liu *et al.* [50] propose the idea of insider attacker detection in wireless sensor networks by exploiting the spatial correlation among the networking behaviors of sensors in close proximity to achieve a high detection accuracy and a low false alarm rate. Doumit *et al.* [51] propose an HMM approach based on the structure of naturally occurring events that uses acquired knowledge distilled from the self-organized criticality aspect of the deployment region.

While these approaches work well, they still lack a low complexity mechanism to allocate monitoring nodes where they are most needed to cover the suspected nodes. This chapter extends our previous work [41] (Chapter 3) that optimizes the security attributes of the sensor network based on the perceived threat by the sink. We present the methodology where monitoring nodes are added as sink proxy agents to assist in the threat evaluation. These nodes act as the witness to the profile of the neighboring nodes that includes position, traffic patterns and delay patterns. Since these nodes are established on high confidence routes, they are also used by the sink for sending probe messages. Unlike many other schemes, these monitoring nodes function as lightweight additions to the sensor nodes reporting only to the sink.

4.3 Background

4.3.1 Sensor Representation

In our previous work on sensor network optimization [33] (Chapter 2), each sensor node is allocated a functional assignment using genetic algorithm. These functions are represented as (a) inactive node (powered off), (b) cluster-head (CH), (c) inter-cluster router (ICR), and (d) sensor node (NS). Each cluster is managed by a cluster-head, and cluster-members are represented by inactive/active node sensors and ICRs. Cluster-head performs data-fusion from various node-sensors while inter-cluster router (ICR) routes cluster data to the sink. Algorithmic details regarding clustering, naming, routing using GA can be found in [33]. Sensor network implements a multi-objective genetic algorithm (MOGA) with competing fitness functions defined to collectively optimize battery sensitive role (functional) assignment, optimal route selection [33], optimal positioning using locomotion and optimal security attributes [41] (Chapter 2). The net effect of control action triggered by fitness factor is the longer battery life. Optimal security attributes are assigned as a result of fitness measure that rewards energy efficient enabling of security attributes measured against battery quantization levels and rate of battery usage. While routes (CH to the sink) are penalized for carrying malformed and retried packets, they are rewarded for enabling authentication on ICRs and encryption on cluster-heads. Additional penalty is awarded if the authentication is enabled disproportional to threat level quantized to M levels.

4.3.2 Genetic Algorithm

Genetic algorithms are based on the principle of natural selection, where each possible solution is represented as a binary string (chromosomes) and an associated fitness measure. Successive solutions are built as a part of the evolutionary process where one set of selected individual solutions gives rise to another set for the next generation. Individuals with a high fitness measure are more likely to be selected into the mating pool with an assumption that they will produce a fitter solution in the next generation (next run). Solutions with the weaker fitness measures are naturally discarded. We use roulette-wheel selection to simulate natural selection, where elimination of solutions with a higher functional fitness is, although possible, less likely. There also exists a small likelihood that some weaker solutions may survive the selection process as it may include some component (genes) that may prove useful following the crossover process. Mathematically, the likelihood of selecting a potential solution is given by

$$P_i = \frac{F_i}{\sum_{j=0}^{N} F_j} \tag{4.1}$$

where P_i is the likelihood that a specific solution is going to be selected for mating pool, F_i represents the functional fitness of the candidate solution, and N is the total number of potential solutions in a population.

Genetic algorithms have proved useful for cases where the search space is large, complex and not well understood with almost no domain knowledge. They can handle large number of variables with arbitrary constraints and multiple objectives.

4.4 Intrusion Detection Approach

This chapter extends the previous work by overloading an additional functional element called *Local Monitoring node* on the sensor nodes selected by the fitness function of genetic algorithm based on the overall network integrity. In our security model, the sink is considered a trusted component that establishes a necessary trust relationship for secure forwarding of data between various node types. Nodes closest to the sink form the most trusted relationships. Farther nodes build the hierarchy of trust starting from the sink, which is apparent from the pre-determined routing decisions that are created during setup and later during re-configuration. Ingredients of security architecture create a trust relationship between various node-types for the reasons related command/message execution, data forwarding, etc. Any authentication is mediated through the sink and components of the trusted routing hierarchy.

GA based IDS approach comprises of LMNs that monitor its proximity domain and communicate any deviation from the standard profile established during initialization or re-configuration. The optimal numbers and position of LMNs maximize the monitoring coverage with minimal power overhead. Monitoring efficiency can be maximized by avoiding the allocation of monitoring nodes on *routes*, *clusters* or *routers* that are below the trust threshold. Trust threshold is established by traffic monitoring and feedback from the monitoring nodes as a function of fitness measure. LMN functionality can only be assumed by the cluster heads or inter-cluster-routers.

From the security standpoint, each node can be represented as a standard node or a monitoring node using a chromosome representation. The chromosome of the GA contains all the building blocks to a solution of the problem at hand in a form that is suitable for the genetic operators and the fitness function. It is also referred to by name ‘string’. It represents a set of parameters that defines a proposed solution to the problem that the genetic algorithm is trying to solve. Each individual CH or ICR is represented by a 1-bit binary number called ‘gene’. This one-bit genes, which define the feature of the node, are called ‘allele’ and are represented as

(a) *0 - **NOP**: No special treatment of the node*

(b) *1 - **Local Monitoring Node**: Monitoring agent for sink.*

Each string represents the collection of functional attributes of each CH or ICR as described above. The fitness function called ‘Trust Measure’ (Sec. 4.4.2.4) is evaluated for each string. A higher fitness represents an optimal functional representation that is evaluated according to the level of suspicious activity, desired coverage and residual battery power in the system. For example, a perfect system will have very few monitoring nodes and vice-versa. Trust measure of a chromosome string is defined in the following section.

4.4.1 Local Monitoring Node (LMN)

A local monitoring mode is a trusted proxy agent for the sink. The sink allocates a CH or ICR to act as a LMN. In case of a CH, it can use any of its member nodes as a pass-through for monitoring purpose, which can increase the snoop coverage of the chosen cluster-head. Upon selecting a cluster-head as LMN, the sink sends the received signal strength (RSS) profile of its member nodes that are capable of listening to the monitored (suspicious) clusters. This data is used by the CH to select one of its members as a pass-through agent to monitor the cluster or ICR under observation.

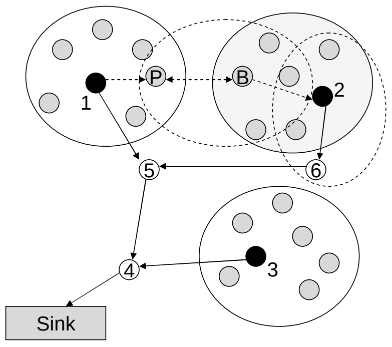


Figure 4.1: Local monitoring modes are allocated to CH-1 (Node 1) and ICR6 (Node 6) to cover the suspected cluster 2. These two LMNs provide complete coverage to the cluster-2. The LMN chromosome string is represented by 1 0 0 0 0 1, where nodes 1 (CH-1) and 6 (ICR) are allocated LMN function. The sink delivers special patterns loop-back command via LMN (CH-1) using path $4 \rightarrow 5 \rightarrow 1 \rightarrow P \rightarrow B$. Message response gets back on path $B \rightarrow 2 \rightarrow 6 \rightarrow 5 \rightarrow 4$. P is the proxy chosen by CH-1 (LMN) to monitor block of 5 nodes on cluster-2 (Represented by dotted ellipse).

Due to the broadcast nature of the sensor node communications, LMN agent evaluates the target node fingerprint by monitoring (a) the received signal strength, (b) transmission periodicity, (c) spurious transmissions from nodes that are not listed as neighbors, (d) response delay (or unresponsiveness) to test patterns, and (e) packet dropping or modification. For example, in Fig. 4.1, LMN proxy agent (P) can monitor (listen) if the data from node B is addressed to node 2 or elsewhere. Additionally, the sink uses this node as a loop-back agent to transmit special patterns through its trusted route and receives the pattern using a pre-established route. Each hop that follows the *node under observation* extracts the pattern data and hashes it with *globally unique sensor ID* (GUID) using the inter-node communication key (INCK) [41] (Chapter 3) and appends it along with the packet delay. The hashed data serves as a pattern to the next hop in the pre-established route. Packet delay on special patterns is also monitored by LMN and returned back to the sink. This process repeats through all the hops to the sink, which uses this data to identify the malicious node or to harden its integrity rank (IR).

While LMN monitors the proximity that includes neighboring nodes, the sink monitors the whole WSN for anomalies. The sink uses LMN alerts along with analytical traffic data to rank clusters and routes for integrity based on deviation from standard profiles. These structures characterize the behavior of clusters or routes in terms of statistical metrics and models of observed activity. The statistical models used by the sink may be an operational model, mean and standard deviation model, multivariate model, Markov process model, time series model, etc. [44, 45]. Various sink observables are:

(i) Detecting various network misbehavior, like selective forwarding, data inconsistency or spoofing of application data. The sink maintains up-to-date information on its neighboring routers (ICR), also called L1 routers [33] (Chapter 2). Since the sink will aggregate the application data from sensor nodes, it can detect the list of sensor nodes with missing data without difficulties. After identifying the list of sensors affected, the system can estimate the attack region. This data in conjunction with LMN can harden the intrusion detection.

(ii) Auditing the suspicious nodes by sending special patterns and comparing its hop delay to that of profiled hop delay as well as hop delay reported by LMNs.

(iii) Analyzing sensor data using data fusion to discover inconsistencies among them. Malicious nodes have larger inconsistencies between the data they return to the sink.

(iv) Auditing the locality of the sensor node under suspicion using the received signal strength indicator (RSSI) data from two or more LMNs and comparing with pre-existing data received during initialization or last reclustering event. It profiles the attenuation in radio signal strength between the sender and the receiver. The power of the radio signal falls off exponentially with distance. Multiple LMNs measure this attenuation in order to estimate the relative position using multilateration technique.

(v) Analyzing the packet data patterns related to excessive packet rate, malformed or corrupted packets, missing replies, retransmissions or repetitions and comparing it to an existing route profiles.

4.4.2 Fitness Function

Fitness function is the measure of optimality of the positioning of monitor nodes (i.e., LMN) in a sensor network as well as accurate identification of the aberrant clusters or routers. This fitness function competes with other fitness criteria like battery fitness, load balancing fitness [33, 41] (Chapter 2, 3) and acts as a dynamic process that repeats over system's life-time. Various elements of fitness function are described as follows.

4.4.2.1 Monitoring Node Integrity Fitness (MIF)

This component of the fitness function resists allocation of the monitoring agent to a cluster or a route that is suspected to be compromised. The sink evaluates each individual cluster and route for possible intrusion based on the messages that it monitors and rank them on a scale of $0 \cdots 1$ based on a set of rules. This value is called integrity rank; a low value represents high intrusion suspectability.

$$\begin{aligned}
\text{MIF} &= \frac{\sum_{\text{ch}=1}^{N} \text{IR}_{\text{ch}} \cdot K_{ch}}{\sum_{\text{ch}=1}^{N} K_{\text{ch}}} + \frac{\sum_{\text{icr}=1}^{M} \text{IR}_{\text{icr}} \cdot K_{\text{icr}}}{\sum_{icr=1}^{M} K_{\text{icr}}} && (4.2)\\
K_x &= 1, \quad \text{if} \quad x = \text{LMN}; \quad x \in (\text{ch}, \text{icr}) && (4.3)\\
\text{IR}_{\text{icr}} &= \frac{\sum_{r=1}^{R} \text{IR}_{\text{icr}}^{r}}{R} && (4.4)
\end{aligned}$$

where IR_{ch} and IR_{icr} are the 'integrity ranks' of CH and ICR, respectively, R is the number of routes, and $\text{IR}_{\text{icr}}^{r}$ is the 'integrity rank' of the route r that includes icr as one of its routers in its path.

The sink evaluates the IR by evaluating the historic traffic patterns, location estimation as recorded by current LMN, sliding window correlation analysis using covariance test (eq. (4.5)) between data packets x and y, violations of QoS parameters (e.g., guaranteed delay, packet mis-formation) and reporting of incompatible power states with respect to the sink expectation based on traffic load. An essential aspect is to choose the amount of history to use for temporal trending. Short history can be insufficient to observe the trend, whereas longer history can influence the trends based on remote past. For traffic patterns, we use exponential smoothing model (Eq. (4.6)) to predict the characterization of the packet arrival process as well as index of dispersion of count (IDC) (Eq. (4.7)) to evaluate the burstness of a specific route.

$$
\begin{aligned}
R_{x,y} &= \frac{\operatorname{cov}(x,y)}{\operatorname{var}(x)\cdot\operatorname{var}(y)}; \quad -1 < R_{x,y} < 1 && (4.5)\\
\overline{\lambda}(t) &= \alpha\cdot\lambda(t-1) + (1-\alpha)\cdot\overline{\lambda}(t-1) && (4.6)\\
\text{IDC} &= \operatorname{var}\left(\sum_{k=0}^{n}\lambda_k\right)\Bigg/ E\left(\sum_{k=0}^{n}\lambda_k\right) && (4.7)
\end{aligned}
$$

where $\lambda(t)$ is the actual number of packet arrivals in interval t, $\overline{\lambda}(t)$ is the estimated number of packet arrivals in interval t, and λ_k is the number of packet arrivals between time interval τ_k and τ_{k+1}. It provides a measure of fluctuation of the receiving rate over a given interval.

Measured deviation from the historic profile brings down the relative integrity of the route and the corresponding nodes. This integrity measure reduces further based on current LMN feedback or if the same node participates in more number of suspicious routes.

4.4.2.2 Monitoring Node Battery Fitness (MBF)

Anytime a sensor-node communicates with the cluster-head or cluster-head communicates with inter-cluster router, it pays a penalty in terms of battery usage. Battery is also consumed during the sensing operation or other related functions. Each node alerts the sink about its battery status (Q) and battery utilization rate periodically (node-sync) or when it crosses the quantized limit (or thresholds). These thresholds will be used to penalize the use of those nodes for monitoring operations that consume more battery power. Penalty for using the node with a low battery capacity depends upon the type of node and its residual capacity. The battery status fitness function is expressed as

$$\mathrm{MBF} = \frac{\sum_i^N \mathrm{BC}_i \cdot K_i}{\sum_i^N K_i}; \quad \mathrm{BC}_i = f(Q, U) \tag{4.8}$$

where Q is the residual battery capacity, BC_i is the projected battery capacity of node i (CH or ICR) that ranges between 0 and 1. Battery usage rate (U) depends upon the individual load on each node and can be estimated by observing the traffic patterns and node-sync data. Based on periodic GA evaluation, LMN dynamically shifts its position to maximize the distribution efficiency such that energy usage is uniformly utilized over the sensor network.

4.4.2.3 Monitoring Node Coverage Fitness (MCF)

Coverage fitness component rewards those LMNs that can snoop maximum the number of nodes with low estimated integrity rank. The sink maintains the integrity rank table for each existing cluster and route (to the sink). The end result of maximizing the coverage fitness is the optimal snoop coverage of suspected as well as non-suspected nodes. Though it attempts to maximize the complete coverage for the malicious nodes, non-malicious nodes are also included if convergence is possible but rewards are greater for malicious node coverage.

$$\begin{aligned} \text{MCF} &= \frac{1}{2}\left(\frac{\beta_1 \cdot \sum_i^N \psi_i}{F_1 \cdot N} + \frac{\beta_2 \cdot \sum_j^M \psi_j}{F_2 \cdot M}\right) && (4.9) \\ & \beta_1 + \beta_2 = 1; && (4.10) \end{aligned}$$

where ψ_i is the number of LMN agents monitoring malicious node i that is below the integrity rank threshold, ψ_j is the number of LMN agents monitoring non-malicious node j that is above the integrity rank threshold, and F_1 and F_2 are the desired coverage redundancy for each malicious and non-malicious nodes, respectively. Higher coverage redundancy can help the sink to evaluate the sensor localization using multilateration technique.

Coverage decisions are based upon RSS information available to the sink during initialization step or the last re-clustering step. During either of these steps the CH transmits the RSS data of the neighbors of its members to the sink.

4.4.2.4 Cumulative Trust Fitness (CTF)

The total fitness associated with local monitoring node placement is given by the cumulative trust fitness (CTF)

$$\text{CTF} = \alpha_1\text{MIF} + \alpha_2\text{MBF} + \alpha_3\text{MCF} \tag{4.11}$$

where $\alpha_1+\alpha_2+\alpha_3 = 1$ and individual weight is dependent upon implementation as well as the relative significance of the component. These values can be made adaptive using an external heuristics.

4.5 Results And Discussion

The experimental setup consists of 100 nodes at random positions in a 30×30 space. Individual node picks up a random coordinate between $(0, 0)$ and $(30, 30)$ and assigns itself an UUID and a random battery capacity between 0 and 15. For simplicity, each node is given a coverage area of 3×3. Once all the nodes have been placed in the listen mode, GA is run with the cross-over rate of 60% and an initial mutation of 6%. Experiment assumes line-of-sight propagation between sensor nodes. The software simulates the sink operation and runs in conjunction with NS-2 software to simulate the network traffic. It executes the GA to generate the monitoring node positions as well as security attributes. It also calculates the fitness parameters (Sections 4.4.2.4) based on packet traffic patterns, covariance test, integrity of received packets, battery usage, and monitoring coverage. A separate process in the sink simulator runs a predictive algorithm that estimates the traffic and data integrity into the future using hysteresis. This data estimates the fitness parameters during the GA run. While each GA objective tends to compete with others to converge at the system equilibrium, the end result maximizes the network life for optimal coverage. Additionally we introduce malicious nodes with similar characteristics but different transmission range and some with attack messages. These are deployed randomly in the sensor space or replaces a normal sensor. Attack messages are compliant with the standard protocol but carry random sensor data at random rates.

In our experiment we focus on malicious node detection rate and the false positive detection rate based on variable number of malicious node deployment. We vary the ratio of malicious nodes to non-malicious nodes between 0.05 to 0.30. We then compare the results relative to the case with no LMNs detection. All malicious nodes are inserted at the same time at random positions. Detection continues until all malicious nodes are detected. Fig. 4.2 shows the average detection time with the increase of malicious nodes ratio. The presence of LMNs decreases the detection time by $50 - 60\%$ on average. Detection time shows even greater relative improvements with larger number of malicious nodes because even though packet traffic increases the optimal presence of monitoring nodes simplifies the detection process and profile drift feedback process. Even though LMN consumes extra energy to detect profile drifts, faster detection of the malicious nodes also means a longer battery life for the rest of the nodes. To optimize the battery usage due to LMN function, we adapt the monitoring frequency based on the variation of the profile being monitored on the neighboring nodes.

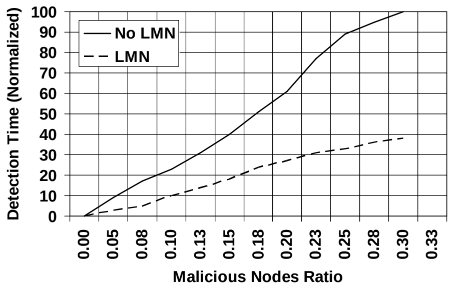


Figure 4.2: Average time to detect all compromised nodes as a function of the number of malicious nodes.

Fig. 4.3 shows the false positive and false negative detection rates as a

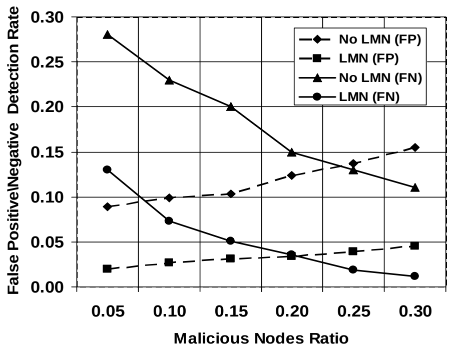


Figure 4.3: False positive (dotted lines) and false negative (solid lines) detection rate as a function of relative number of malicious nodes.

function of the number of malicious nodes. A false positive occurs when a legitimate node is identified as an intruder and a false negative occurs when a malicious node is identified as legitimate. This can be detrimental to the integrity of the system; it decreases the efficiency of the routing and clustering algorithm, since the observability and the analysis based on the compromised data from malicious nodes can trigger non-optimal clustering and optimization decisions [33,41] (Chapter 2, 3). Instead of running until all the malicious nodes are identified, the detection period is kept static in this case. False positive rate with LMN ranges from $2.5 - 5\%$ as compared to $9 - 15\%$ in the traditional case (no LMN). Similarly, false negative rate also shows a substantial improvement.

Speedier detection, low false negative rates, and adaptive LMN sampling enhance the reliability at cost of slight increased energy consumption and improve the network life in the presence of compromised nodes. Improved detection also reduces the false triggers as a result of inaccurate data analysis due to the existence of undetected compromised nodes that the sink has no knowledge of.

4.6 Conclusion

We have presented a genetic algorithm approach to enhance the intrusion detection scheme in wireless sensor networks considering the restrictions of such networks. This scheme allocates the monitoring function to the sensor nodes after evaluating its fitness based on integrity, residual battery power, and coverage. Our monitoring scheme is decentralized since the local monitoring nodes are optimally distributed in the network and feed back the profile drift data to the sink. These profiles are related to the input/output packet traffic pattern, delay profiles, and RSSI based position estimation of the neighboring nodes. The collected data is processed in the sink along with other sink-specific data collected using node-sync messages and data correlation analysis. This approach not only speeds up the detection of compromised nodes by 50%, but also reduces the false positive and false negative detection substantially. Additionally, this scheme complements our security mechanism [41](Chapter 3) that optimizes the security attributes based on accurate analysis of perceived threats as measured by the sink. To offset the monitoring overhead we implemented adaptive sampling that depends upon the measurement variability from an established profile. One of the limitations is that if we increase the number of nodes, GA convergence times increase exponentially. Future work includes improving the scalability of the algorithm as we increase the number of nodes.

Proactive power optimization of sensor networks

— Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen —

In Communications, 2008. ICC'08. IEEE International Conference on, pp. 2119-2123. IEEE, 2008

Chapter 5 – Proactive Power Optimization of Sensor Networks

abstract

We propose a reduced-complexity genetic algorithm for dynamic deployment of resource constrained multi-hop mobile sensor networks. The goal of this chapter is to achieve optimal coverage and improved battery life using dynamic power scaling (DPS) and improved fitness function. DPS exploits idle times, packet delay guarantees, performance and workload data using additional controls related to sensor power states and transmission power. The dynamic power scaling in conjunction with genetic algorithm jointly optimizes power states and topologies by dynamically monitoring workloads, packet arrivals, utilization data and quality-of-service compliance. This results in minimization of the power consumption of the sensor system while maximizing the sensor objectives.

5.1 Introduction

Low-cost integration and small-size micro-sensors [16–20] have generated significant interest in the area of disposable sensors. These are motion capable, randomly deployed, infrastructure-less, data-centric sensors equipped with data processing capabilities and sensory circuits that cannot be charged (or rarely charged) or replaced. These sensors are constrained in energy, bandwidth, storage, and processing-capabilities and find their uses in the areas of homeland-security, disaster-recovery, target-identification, reconnaissance, medical applications, defense applications [22], and intrusion-detection, etc. Each sensors process the sensory data and transmit to the target (sink) in a secure manner.

This chapter extends previous work that used an evolutionary algorithm [4] to divide and position the randomly deployed mobile sensors into an optimal number of independent clusters with cluster-head and optimal route [33] (Chapter 2). Once deployed, these sensors maximize their coverage by moving (or re-orienting) themselves at the expense of battery life and develop a long-lasting secure sensor network with variable security attributes [41]. Cluster-head collects data from its member sensors and sends them to the sink in a compressed and secure manner via the most cost-effective router. The energy dissipation of the sensor node is the sum of sensor transceiver and micro computations. As an extension to previous approach, we introduce *Dynamic Power Scaling (DPS)* and *Dynamic Transmission Scaling (DTS)* that uses the mix of proactive mechanisms and tuning parameters derived from workloads, security attributes and idle periods to optimize battery power.

Genetic Algorithm (GA) is a stochastic search technique that mimics the natural evolution proposed by Charles Darwin in 1858. GA has been successfully applied to a wide range of combination problems. They are particularly useful in applications involving design and optimization, where there are large numbers of variables and where procedural algorithms are either non-existent or extremely complicated.

Dynamic Voltage and Frequency Scaling (DVFS) [52] is key technique in exploiting the hardware characteristics of processors to reduce energy dissipation by lowering the supply voltage and operating frequency. Since performance is needed only for a small fraction of the time, the DVFS algorithms enables energy savings while providing the peak computation power in general-purpose systems by optimizing performance and battery life. The proactive scheme predicts the future work requirements and sets up the power states according to the dynamic policy with parameters related to minimum work, and maximum deferment. These policies support data bursts, runtime constraints and optimal power states.

5.2 Related Work and Motivation

Mobile sensor networks consist of randomly deployed disposable sensors where configurable objectives cooperate with one another to maximize coverage and battery life. In this chapter we use DPS and DTS in conjunction with four competing objectives [33,41] (Chapter 2, 3) that create an energy-efficient sensor network: (i) dynamic cluster membership, (ii) dynamic routing, (iii) dynamic sensor positioning, and (iv) dynamic sensor security attributes.

Related work includes dynamic voltage scaling (DVS) methodology that inserts additional information into the communication channel to guide the selection of proper voltages for data decryption/encryption and processing in order to reduce the total computational energy consumption [53], real-time DVS (RT-DVS) that modifies the OS's real-time scheduler, and task management service to provide significant energy savings while maintaining real-time deadline guarantees [54], PowerTOSSIM that proposes efficient emulation of the sensor node hardware platform coupled with careful instrumentation of the power states which generates an event-driven simulator directly from TinyOS code and emits power state transitions [55], Bult *et al.* presents advances in low-power systems spanning network design, through power management, low power mixed signal circuits, and highly integrated RF network interfaces [56], Xing *et al.* present that significant energy reduction can be achieved by jointly optimizing the transmission power and sleep time of nodes based on the network workload [57]. DVFS is an important power optimization feature in Intel and AMD class of micro-processors that provide multiple performance states using voltage and frequency scaling. ARM's Intelligent Energy Manager (IEM) voltage and frequency scaling reduces system-level power and energy consumption by as much as 15 to 20%.

5.3 Sensor Representation and GA Approach

In our previous work on sensor network optimization [33] (Chapter 2), each sensor node is allocated a functional assignment using using genetic algorithm. These functions are represented as (a) inactive node (powered off), (b) cluster-head (CH), (c) inter-cluster router (ICR), and (d) sensor node (NS). Each cluster is represented by a cluster-head, and cluster-members are represented by inactive/active node sensors and ICRs. Cluster-head performs data-fusion from various node-sensors while inter-cluster router routes cluster data (from cluster-head) to the sink. Algorithmic details regarding clustering, naming, routing using GA can be found in [33] (Chapter 2). Sensor Network implements a multi-objective genetic algorithm (MOGA) with the following fitness functions defined as follows [33, 41] (Chapter 2, 3):

(1) **Total Node Fitness (TNF)** forms weighted sum of Coverage Fitness (CF), Cluster-Head Fitness (CHF), Node Communication Fitness (NCF), Battery Status Fitness (BF), Router Load Fitness (RLF), and Sensor Effector Fitness (SEF)

$$\text{TNF} = \alpha_1\text{CHF} + \alpha_2\text{NCF} + \alpha_3\text{BF} + \alpha_4\text{RFL} + \alpha_5\text{SEF} + \alpha_6\text{CF} \tag{5.1}$$

(2) **Route Selection Fitness Function (RSFF)** generates balanced routes based on node allocation using GA based on node fitness function. During setup operation, both CH and ICR start sending data on the most cost effective routers.

(3) **Total Node Motion Fitness (TNMF)** is weighted sum of Coverage Uniformity Fitness (CUF), Cluster-Node Migration Fitness (CNMF), Cluster-Head Migration Fitness (CHMF), Node Motion Fitness (NMF) and Sensor Data Fitness (SDF). Fitness associated with node motion is given by

$$\text{TNMF} = \alpha_1\text{CUF} + \alpha_2\text{CNMF} + \alpha_3\text{NMF} + \alpha_4\text{CHMF} + \alpha_5\text{SDF} \tag{5.2}$$

(4) **Secure Node Fitness (SNF)** rewards energy efficient enablement of security attributes that are measured against battery quantization levels and rate of battery usage. While routes (CH$\rightarrow$sink) are penalized for carrying malformed and retried packets, they are rewarded for enabling authentication on routers (ICR) and encryption on cluster-heads. Additional penalty is awarded if the authentication is enabled disproportional to threat level quantized to M levels.

5.4 Power Scaling Approach

Power scaling optimizes the functional blocks of the sensors for a given Quality-of-Service (QoS) as perceived by the sink. QoS policy is defined as a function of (i) security attributes, (ii) importance and accuracy of sensor data (packet Priority), and (iii) maximum node-sink delay (sensor data, control data, etc.). QoS policies along with quantifiable observations (workloads, data arrival patterns, battery levels, node stability) adjusts the power-scaling parameters of the functional blocks of the node to enhance QoS compliance. For a given functional block, these parameters assume static conditions within a tunable observation period (T_{obs}). Following sections define various functional blocks and corresponding controls.

5.4.1 Memory Buffer Subsystem

Memory subsystem (Fig. 5.1) comprises of Transmit/Receive Buffer (TRB), Transmission Request Queue (TRQ), Performance Counters (PC), and Tunable Registers (TR). Memory buffer is divided into multiple blocks with independent power control applied according to anticipated demand. Data received by the receive buffer is processed for further action (authentication, header manipulation, etc.) before committing to the TRQ. Once in TRQ, data is transmitted to the next hop upon inactivity timer expiration or reaching burst threshold. Number of active memory blocks, observation timer, inactivity timer and burst thresholds are estimated based on QoS policies, sink feedback, and activity trends as measured by performance counters (PC). Performance Counters measure mean number of packet arrivals/serviced and IDC for packet arrivals [58] (index of dispersion for count). IDC is defined as the variance of the number of packet arrivals divided by the mean number of packet arrivals in an interval of length t

$$\mathrm{IDC} = (\mathrm{var}(\sum_{k=0}^{n}(\lambda_k))/(E(\sum_{k=0}^{n}(\lambda_k)) \tag{5.3}$$

where λ_k is the number of packet arrivals between time interval τ_k and τ_{k+1}. It provides a measure of fluctuation of the receiving rate over a given interval, which reflects considerable burstness in the received packets. Burstness is a direct indication of packet loss and buffer occupancy.

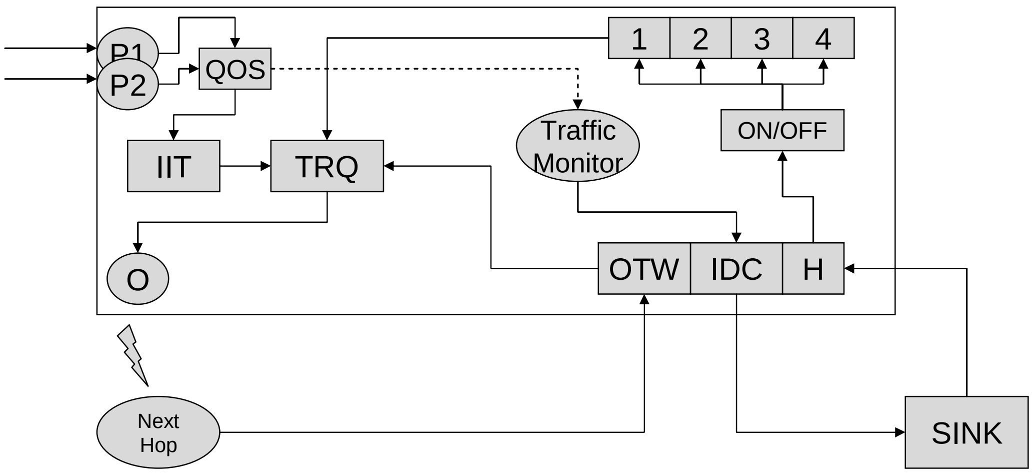


Figure 5.1: Memory buffer subsystem: it contains four memory blocks that are activated based on buffer requirements per HURST (H) parameter (updated by sink). TRQ controls the number of fragments transmitted in a single burst based on IIT and QTW parameters. P_1 and P_2 represent input ports, O represents output port (next hop). IDC value is updated upon $P_1 + P_2$ traffic.

IDC slope is further used by sink to measure the HURST parameter (H) which is the measure of the persistence of a statistical phenomenon, or the measure of the long-range dependence of a stochastic process. Buffer requirements are much higher at lower levels of utilization for higher degrees of self-similarity (higher H). Sink updates the H value periodically which is then used by nodes to estimate the number of active memory blocks.

$$\text{Buffer (B)} = \rho^{1/[2(1-H)]}/(1-\rho)^{H/(1-H)} \tag{5.4}$$

where ρ is the utilization factor given as

$$\rho(\text{utilization}) = E(\sum_{k=0}^{n}(\lambda_k)/E(\sum_{k=0}^{n}(\mu_k) \tag{5.5}$$

where μ_k is the number of packets serviced between times τ_k and τ_{k+1}. Adequate buffer activation saves power by avoiding excessive allocation or switching between ON/OFF states.

Additionally, TRQ schedule transmit requests according to maximum transmission rate, Optimal Transmission Window (OTW) (burst size), and QoS requirements relative to delay tolerances. For burst size less than the OTW, TRQ defers the transmit request for a duration equal to the inactivity interval threshold (η_k^j) for n^{th} node at t^{th} time. IIT is programmed according to QoS requirements of the sensor data.

$$\beta_i^j(x) = \max\left(-1, \min\left(1, \frac{T_i^j(x) - D_i^j(x)}{D_i^j(x)}\right)\right) \tag{5.6}$$

$$\eta_t^j = \eta_{t-1}^j + \frac{\eta_{\min}}{N}\sum_i \beta_i^j(\text{mem}) \tag{5.7}$$

where $D_i^j(\text{mem})$ and $T_i^j(\text{mem})$ represent the expected delay and measured delay, respectively, for i-th packet (route) on j-th node. While D_i^j(mem) is updated using sink's QoS feedback (Section 5.5.1), OTW is updated as a result of control message sent by target node.

5.4.2 Micro Controller (μC) Subsystem

Micro controller is an integral component of the sensor node. In this section we consider the micro-controller and its effect on sensor node power consumption. We review the factors influencing the power consumption and calculate the expected performance based on tunable parameters. Micro controllers are optimized for functions related to:

(1) *Message Handling* - Operations related to message parsing, data fragmentation, handling TRQ, data/header manipulation, and buffer management heuristics.

(2) *Security Protocols* - Operations related to data encryption/decryption and authentication protocols to support confidentiality, authenticity, unidirectional communication, and tamper resistance. The computation load (and hence execution time and energy consumption) for encryption and decryption provides an opportunity to optimize the power states.

(3) *Event Handling* - Operations related to events that are routed to a PIN assertions or triggers due to threshold crossings, periodic timers or hysteresis effects.

(4) *Performance Monitoring* - Operations related to synthesis of performance data specific to the sub-system it is monitored for. Performance data is polled at optimal sampling granularity subject to sampling variances.

Processor's dynamic power dissipation is proportional to capacitance, clock frequency, and the square of supply voltage ($P \propto C_L.V_{dd}^2.f$). This implies that to accumulate the same amount of computation, using lower voltage will consume less energy in longer time because the power level is much lower. We use discrete performance states using voltage scaling represented by P_i, where i is the state number. For X discrete states, P_0 is the highest-power/least-latency state, whereas P_X is the performance state.

Since, upon packet arrival, we are uncertain about computational requirements, any unused cycles allotted would eventually be wasted due to idling for extra processor cycles. DVS algorithm avoids wasting cycles by reducing the operating frequency and ensuring that deadline guarantees are not violated by doing so. Based on the QoS requirements of each message and its respective security attributes, P-State ceiling is set for the period of tunable interval (T_{obs}). The ceiling is adjusted according to the delay targets of the ICR or CH nodes.

$$P_t^j = min(X, P_{t-1}^j) + (\gamma/N) \sum_i \beta_i^j(\text{cpu}) \tag{5.8}$$

where X is the maximum number of discrete performance states (P-States), γ is the scaling factor ranging between 1 and 2 and $\beta_i^j(\text{cpu})$ is the QoS compliance factor. CPU state of node j is incremented or decremented according to QoS compliance of all CH packets passing through it.

5.4.3 Wireless Link Subsystem

Transmission time decreases as direct consequence of increasing bit-rate which, without increasing the data transmission decreases the radio duty-cycle. If the radio's turn-on-to-receive exit latency is high, then it becomes impossible to achieve the required duty-cycle ($< 1\%$). Furthermore, when the radio switches from sleep mode to transmit mode to send a packet, a significant amount of power is consumed for starting up the transmitter itself [59]. Therefore optimal tuning is needed to avoid reactive response to an idle slot during transmission. Power savings are realized by running micro-controller at the optimal P-State and radio at optimal frequency which spreads the computations in time and transmits the data in a quick burst. This requires decoupling between computational and transmission rate, where each can run at its optimal point using rate-matching between computational processing and data transmission. Since the instantaneous traffic load is mostly lower than the peak value, transmissions can be slowed down, to the optimal operating point. Similar to DVS [Section 5.4.2], that has shown to be effective mechanism for CPU power management, Dynamic Modulation Scaling can adapt the modulation level to match the instantaneous traffic load, as part of the radio power management.

Different multilevel modulations can be used for adaptive modulation. In this chapter we will use M-QAM modulation as it provides a lower probability symbol error compared to M-PSK for the same SNR as well as consumes lesser energy/bit. Sensor node adjusts constellation size (b) and symbol rate (R_s) to reduce the overall energy. Schurgers *et. al* [60] define the following expression for minimizing energy required to transmit one bit by choosing the correct values

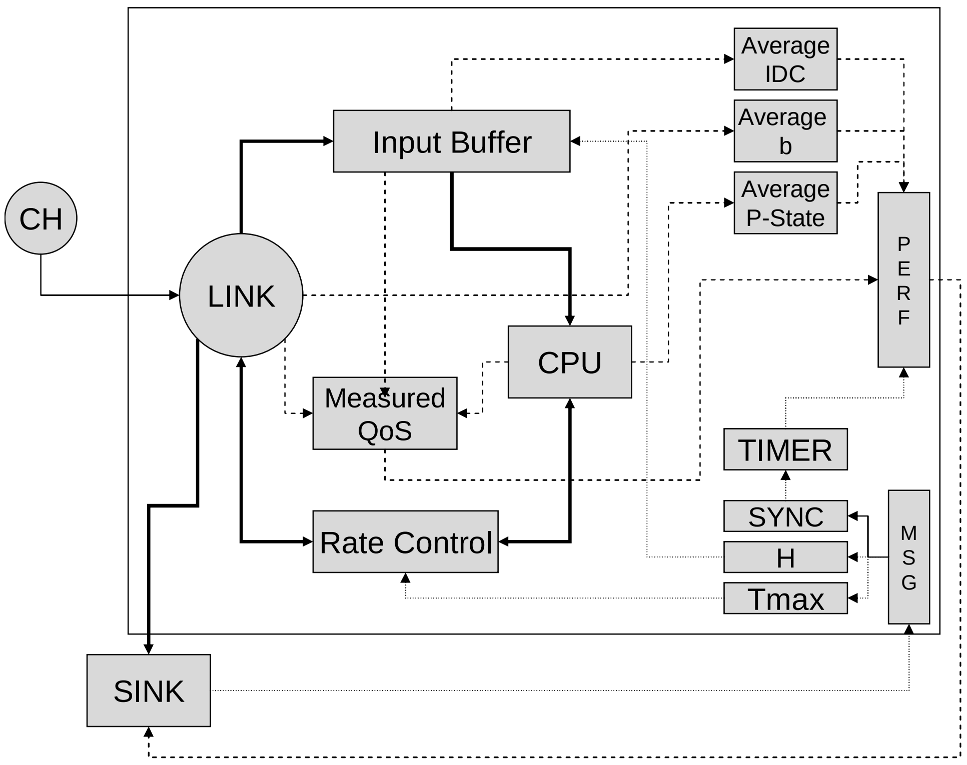


Figure 5.2: Performance counters monitor CPU power, modulation power and QoS compliance. It transmits the performance data upon timer trigger. Sink uses this data to generate performance targets for nodes. Rate controller uses this data to distribute delay targets to CPU, link and buffer control.

of b and R_s:

$$E_{\text{bit}} = \left[C_S \cdot (2^b - 1) + C_E + C_R \cdot \frac{R_{S\_\max}}{R_S} \right] \frac{1}{b} \tag{5.9}$$

where C_E and C_R are functional components that incorporate electronic circuitry for filtering, up-converting and modulating. Parts of the circuitry operates at frequencies proportional to instantaneous symbol rate (R_S), while other parts operate at frequencies proportional to maximum symbol rate ($R_{S\_\max}$). C_S

represents the function of target performance that is weakly dependent upon b. Since high value of R_S results in minimizing delay and energy transmitted per bit, it is logical to maximize this value to $R_{S\_\max}$. Hence, the constellation size b is the only option required to trade off energy versus delay. b_t^i is initially set to the maximum for the i-th node at time t and tuned dynamically according to the sink feedback and QoS targets. Similar to η [eq. (5.6)], this parameter is tuned as a function of target and measured QoS.

5.5 Parameter Tuning and Fitness Function

In this section we will discuss the aspects of coordinated tuning and fitness function (Fig. 5.2) that uses the parameters defined in Section 5.4. While tuning is necessary to maximize the performance/energy ratio of given set of nodes for a given topology (clustering, routing, etc.) and QoS, Fitness Function is required to optimize the sensor network topology to achieve the target performance/energy ratio for the entire network with optimal battery utilization. GA utilizes the optimal operating points and performance feedback of the nodes of the instantaneous topology to calculate the output of the fitness function which influences the cluster formation/placement, membership, functional attributes, and routing decisions [33] (Chapter 2).

5.5.1 Coordinated Tuning

Coordinated tuning is necessary for identifying the operating point in order to maximize the performance of the node with respect to energy consumed. Performance is measured using a QoS function that is dependent upon communication delay, messaging priority, etc. Router node j estimates the QoS error for cluster head packet i (ξ_i^j) using:

$$\xi_i^j = \frac{T_i^j - \max(\epsilon, (1 - Qp_i)) \cdot T_{i_{\max}}^j}{\max(\epsilon, (1 - Qp_i)) \cdot T_{i_{\max}}^j}, \quad 0 \leq Qp_i \leq 1 \tag{5.10}$$

where T_i^j represents the processing delay for packet i on ICR j, Qp_i is the priority of i-th packet, and $T_{i_{\max}}^j$ is the maximum delay allowed for i-th packet on j-th ICR. Periodically, sink sends the SYNC-sink message containing new parameters ($T_{i_{\max}}^j$, Qp_i and Hurst (H)) for all member nodes ($R(i)$) of route catering CH i. Each ICR and CH node implements the closed-loop-control function that minimizes the ξ_i^j by tuning inactivity interval threshold (η^j) [eq. (5.6)], CPU P-state ceiling (P^j) [eq. (5.8)] and modulation scaling parameter (b^j) [eq. (5.9)]. Each of these components contribute partially to the maximum allowed delay $T_{i\max}^j$ thereby operate within its QoS bounds. QoS contribution of each tunable component can be expressed using eq. (5.12):

$$\begin{aligned} D_i^j(x) &= \max(\epsilon, (1 - Qp_i)).T_{i_{\max}}^j(x) && (5.11) \\ \xi_i^j(x) &= (T_i^j(x) - D_i^j(x))/D_i^j(x) && (5.12) \\ T_{i_{\max}}^j(x) &= \phi_x T_{i_{\max}}^j && (5.13) \end{aligned}$$

where ϕ_x is the delay contribution factor due to processing element x =(Memory, CPU, Link). Each processing element measures QoS compliance by calculating the differential between measured ($T_i^j(x)$) and expected delay ($D_i^j(x)$). It scales the parameters accordingly to reduce the QoS error.

While early arrivals can cause a bursty traffic and memory pressures, late arrivals can cause performance issues. To avoid that situation, ϕ_x is tuned by monitoring the transmit buffer for a period of programmed interval (T_{obs}). For high bandwidth case we limit the transmit buffer utilization to 70%. This is done by first reducing the modulation delay factor ϕ_b to minimum bounds, then increasing the CPU delay factor ϕ_{cpu} and finally increasing memory delay factor ϕ_{IIT}. For low utilization case, we reduce the CPU delay factor first, followed by modulation delay factor and finally memory.

5.5.2 Power Scaling Fitness Function (PSFF)

In this section we define a new fitness function that rewards the uniformity of the power states and QoS compliance within an established route. This function also penalizes a disproportionate allocation of power-states with respect to other routes as well as non-optimal provisioning of memory buffer. Elements of fitness function are described as follows:

(1) **QoS Fitness (ω_i^1)** is defined as degree to which routes are compliant with respect to allocated delay budget. A route i is considered compliant (β_i) if the measured delay (D_i) falls within $\pm\delta\%$ of target delay (T_i). Over-compliance as well as under-compliance are both penalized though with a non-uniform penalty factor ($\widetilde{\lambda}$, λ).

$$\omega_i^1 = 1 - \frac{\sum_j (\widetilde{\lambda}|\max(0, \beta_i^j - \delta)| + \lambda|\min(0, \beta_i^j + \delta)|)}{N} \tag{5.14}$$

This parameter reflects the burst variability of multiple arrivals multiplexed locally. Variability exists due to variable message sizes, arrival rates, priorities and non-uniform compute requirements per security attributes [41] (Chapter 3)
(2) **Buffer Optimization (ω_i^2)** is the measure of effective memory utilization. Packets from different sources (CH and ICR) arrive at different times and are statistically multiplexed into the common buffer. These packets can have variable rates depending upon sampling variances and QoS. Buffer occupancy is dependent upon service times of the CPU and wireless subsystem which is optimized as a part of tuning process. It penalizes non-optimal provisioning of the memory blocks that can cause over allocation or reactive switching between memory power-states. Furthermore, inadequate buffer between CPU and transmit logic can cause coupling between CPU processing and transmission rates.

$$\omega_i^2 = 1 - \frac{\sum_j \min(1, |B_{i\_\text{prid}}^j / B_{i\_\text{act}}^j - 1|)}{N} \tag{5.15}$$

(3) **Uniform Power-State Distribution (ω_i^3)** penalizes the routes that consume disproportionate amount of power as compared to average power utilization by other routes. It uses average P-state residency ($\overline{P_i^j}$) [eq. (5.8)] and average modulation scaling ($\overline{b_i^j}$) [eq. (5.9)] as the measure of power consumption by a node i. As described above, one of the many reasons for non-uniform distribution is heterogeneous message priorities, variable rates and security attributes.

$$\omega_i^3 = 1 - \frac{\sum_j \min\left(2, \left(|\overline{P} - \overline{P_i^j}| + |\overline{b} - \overline{b_i^j}|\right)\right)}{2N} \tag{5.16}$$

Overall PSFF fitness function of route i is the weighted sum of all contributing elements. While TNF (Section 5.3) [33] (Chapter 2) incorporates communication energy as a part of NCF, it lacks the energy contribution due to other components like micro-controller and wireless logic. We modify that equation by adding PSFF contribution:

$$\begin{aligned}
\text{PSFF}_i &= \mu_1\omega_i^1 + \mu_2\omega_i^2 + \mu_3\omega_i^3 \qquad (5.17)\\
\text{TNF} &= \alpha_1\text{CHF} + \alpha_2\text{NCF} + \alpha_3\text{BF} + \alpha_4\text{RFL} + \\
&\quad \alpha_5\text{SEF} + \alpha_6\text{CF} + \alpha_7\text{PSFF} \qquad (5.18)
\end{aligned}$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 = 1$ and α_i depends upon the relative significance of the component. These values can be made adaptive using an external heuristics. Details of GA is described in [33] [41] (Chapter 2, 3).

5.6 Results And Discussion

Experimental setup consists of 100 nodes at random positions in a 30×30 space. Individual node picks up a random coordinate between $(0, 0)$ and $(30, 30)$ and assigns itself an UUID and a random battery capacity between 0 and 15. For simplicity, each node is given a coverage area of 3×3 and assumes line-of-sight propagation. After nodes placement in the listen mode, GA is run with the cross-over rate of 60% and an initial mutation of 6%. The software simulates the sink operation in conjunction with NS-2 software that simulates the network traffic. It executes the GA that generates re-clustering/re-assignment tasks. It also calculates the fitness parameters based on network traffic, battery usage and power-performance parameters (PSFF). A separate process in the sink simulator runs a predictive algorithm that estimates the traffic and data patterns (sampling rates, data Redundancies, Self-Similarity etc.) into the future using past hysteresis. This is in conjunction with the closed-loop self-optimization of the nodes itself that run optimization heuristics to distribute the delay budget between buffer, CPU and wireless link infrastructure using route-feedback (H, delay budget) from sink. While each GA objective tends to compete with others to converge at the system equilibrium, the end result is to maximize the network life for optimal coverage. The experimental scheme involves local optimizers that tunes themselves and GA optimizers that optimizes the complete topology and node assignments. Guaranteed delays are maintained by adjusting delay times between CPU and wireless link (Section 5.5.1).

In a power scaling case (PSFF), delay guarantees benefit between 10-20% over non-PSFF case (Fig. 5.3b). Due of high sampling variances, we incur frequent buffer-overflows or empty buffer queues in non-PSFF case. Furthermore, because high power-states remain static (due to reactive ON/OFF) for longer duration, it processes the traffic faster that make it burstier towards the sink and causes packet drops. PSFF not only improves the QoS guarantees, but also reduces average power consumption/node by about 27% due to proactive state determination between CPU and Wireless Link subsystem. Fig. 5.3(c) shows the node loss versus of the number of generations. It is found the Power-Scaling (PSFF) case significantly outperforms the static one with about 25% at 600-th generation. On an average it shows about 12-15% reduction in the number of nodes lost. Main power savings are realized due to voltage and modulation scaling for given delay guarantees (QoS) as well as re-clustering triggers due to

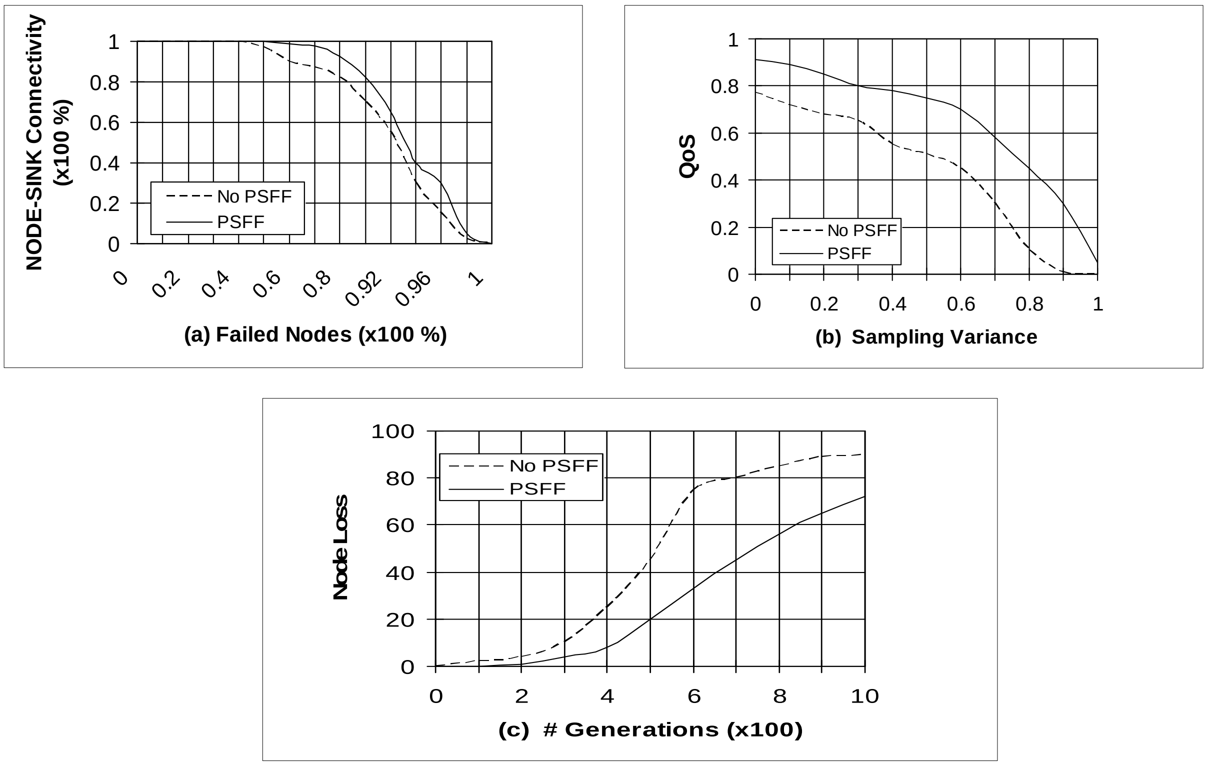


Figure 5.3: (a) Percentage of nodes connected to sink in the event of node failure; (b) QoS measure (of Route 1) as a function of variable sampling rates of the neighboring clusters; (c) Percentage of nodes lost (due to battery) as a function of n-th generation.

disproportionate allocation of packet priorities, multiplexing of heterogeneous traffic patterns with variable rates and security attributes. Fig. 5.3(a) shows 5-8% improvement in the number of nodes with a valid route to sink because residual energy saved (using PSFF and local power scaling) promotes nodes to act as ICR(s).

5.7 Conclusion

We presented dynamic energy-efficient sensor deployment using a multiple-objective genetic algorithm in conjunction with local tuning of CPU, memory and wireless link power states. This approach maximizes coverage, quality-of-service and network life by exploiting dynamic power scaling methods and re-clustering based on those methods. We observe incremental improvement over traditional approach [33] (Chapter 2) due to proactive mechanisms that predicts the future power states and buffer requirements to achieve the projected delay guarantees (guaranteed QoS). As a result of power conservation due to scaling, we also observe reduced clustering triggers as lesser nodes reach there functional thresholds. In a multi-hop network, heterogeneous traffic flows are multiplexed into an ICR which causes variations in the resource consumption. Re-clustering with power scaling fitness function bias reduce such variations and enhances the QoS compliance (up to 20%) with guaranteed delays. PSFF also prevents GA to converge to a local optimum due to uniform load requirement by biasing PSFF to about 25%. Future work include tuning the optimal bias for various components of TNF [eq. (5.17)] and an ability to optimize delay guarantees to maximize the network life.

Self-Organization of Wireless Sensor Network for Autonomous Control in an IT Server Platform

— Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen —

In Communications (ICC), 2010 IEEE International Conference on, pp. 1-5. IEEE, 2010

Chapter 6 – Self-Organization of WSN for Autonomous Control in an IT Server Platform

6.1 Introduction

Embedded in any multiprocessor platform are several distinct sensor and control devices of varying function, ranging from power/thermal/performance sensors, power control units [61] to test hardware [62]. These components provide comprehensive observability, control and protocols infrastructure, which can be used to perform performance-enhancing tasks such as *self-optimization* – adapt to improve its performance and efficiency; *self-diagnosis* – detect and diagnose complications; *self-healing* – recover from localized faults and reintegrate restored resources; *off-package testing* – analysis using stress patterns before integrating it in a system; and *self-protection* – profile normal system behavior and detect malicious use. These tasks not only streamline system performance, but also extend system lifetime, simplify maintenance operations, prevent cascading failures, optimize resource allocation and reduce cost of operation, to name just a few of the myriad of benefits. To implement these functions, each significant component within a system must be able to exchange sensor and control information with peer devices, as well as a "host" component, which enables external access to in-system data for policy enforcement.

Sensor network for this unique environment is a multi-CPU system-on-a-chip (SoC) infrastructure comprising of a hetrogeneous mix of data sensors and intelligent control devices where data flow requirements need to be brokered between end-points in an ad-hoc manner. It is time consuming to evaluate all possible solutions to optimize routing and bandwidth resources. An efficient strategy would allocate functional elements and routes to achieve maximum coverage and quality of control (QoC). The traditional approaches to support distributed observability and control uses statically configured wired solutions that are static, non-scalable and typically expensive. We propose a dynamically reconfigurable wireless network. This requires a nano-wireless sensor network (nWSN) structure with a large number of sensors (e.g., over hundreds). There are significant challenges for the organization of such network. First, the substantial amount of metallic objects inside the computing platform (e.g., along both sides of the motherboard) present propagation challenges - each sensor is expected to potentially “see” only a small subset of other sensors in the system due to signal blockage by metallic objects. Second, finding the optimal solution in terms of throughput, power, and delay of network with a large number of sensors is too complex to be practical. Third, because of the varying functions of each sensor, the amount of data to be exchanged by each sensor will be different. For example, control sensor might expect rates over 1 Gbps, whereas thermal sensors might require only Kbps data rate. These unique characteristics of such nWSN require a drastically different network organization scheme than the conventional sensor organization.

We extend our previous work on sensor network organization using genetic algorithms (GA) [33, 41] (Chapter 2, 3) that is suitable to handle such complex issue with the aforementioned unique physical-layer characteristics. Genetic algorithm [4] is a stochastic search technique that resembles the natural evolution. It targets to support dynamic re-configurability that fulfills the need for necessary ingredients required for accurate data acquisition, better data-flow rates, distributed and cooperative management, multi-objective goals and long-term observability for in-depth diagnostics functions. Organization of the target nWSN must take into consideration the physical characteristics of the channel since each sensor might potentially be able to communicate with only a subset of other sensors. The requires complex weighting procedures so that any attempts to cluster sensors who cannot see each other be minimized. GA algorithms have proved useful in tasks involving design and optimization decisions with a large variable space where procedural algorithms are either non-existent or extremely complicated. They enhance real-time usage due to multiple offspring that come up with parallel solutions. Furthermore, several possible solutions aid in searching multiple points simultaneously and, therefore, avoids being caught in a local optimum. It also acts as global optimizer that allows for greater degrees of freedom in the choice of model structure which enables programmatic infrastructure for complex optimization problems. This paper contributes to the development of a GA based network organization that seemingly achieves a high quality-of-control suitable for the complex nWSN found in autonomous systems.

We consider a typical equipment rack environment where multiple servers are stacked one above another, thereby minimizing the resource and floor space requirements. We replace the static, wired sensor network with GA based re-routable, dynamic wireless sensor network and compare the performance gain against the baseline approach. Sec. 6.2 illustrates the usage model and challenges faced by a large sensor network in an IT system. Sec. 6.3 describes the

GA approach to nWSN. Sec. 6.4 describes the fitness function used to evaluate the GA fitness related to coverage, bandwidth allocation, routing and quality of control. Sec. 6.5 evaluates the performance of the nWSN, followed by a summary of the nWSN methodology in Sec. 6.6.

6.2 Wireless Sensors in IT Systems

Each server consists of a large number of CPUs, DIMM, I/O components that need to globally optimize the operational decisions by efficient use of the available resources. Based on power, thermal, resource (CPU, memory, storage, I/O) and performance characteristics, workloads migrate between different nodes or racks with an objective to reduce operational cost while adhering to the performance requirements.

6.2.1 Usage Model

Wireless sensors with data rates at low-to-medium speeds (e.g., below 1 Gbps) allows automation using efficient state exchange (power, thermal throttling and performance states, component profile deviation), system process control (power states, tuning optimization, emergency triggers for power delivery etc.), platform test and debug in an isolated environment, off-board component testing, and cooperative tuning and control.

Fig. 6.1 shows three wireless manageability interconnects used for automating sensors to monitor platform stability, power efficiency and fault diagnosis. Self-Optimization targets platform stability using collective decision process to automate system states for maximum efficiency in terms of energy usage, per-

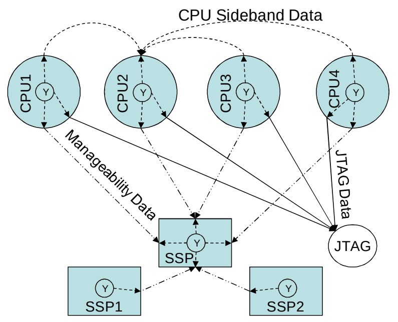


Figure 6.1: Platform usage of wireless interconnects in a PC/server environment.

formance/watt [63], thermal tuning, power budget re-balancing, and component failure analysis. Each platform component plays an optimization game. This involves multiple components which defend equilibrium strategies between them in real time using efficient telemetry for observability and control. The broadcast nature of wireless makes available the equilibrium states to all other components in real-time without adding any routing complexity. This allows them to automate themselves based on the collective strategy. Fig. 6.2 shows the ability of the system service processor (SSP) to configure the cluster-specific fabric using sensors supported by wireless interconnection. The SSP can establish basic connectivity with the cluster-specific end-points, and configure the high-speed fabric and address decode infrastructure without adding routing complexity. Similarly, each CPU can establish CPU-CPU sideband connectivity that can be used for power/thermal/performance optimization.

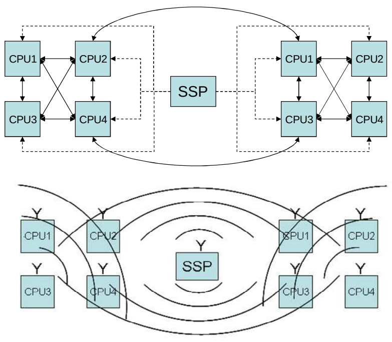


Figure 6.2: Wired manageability connectivity between System Service Processor (SSP) and end-points (CPU) are replaced by wireless. The broadcast nature of wireless can reach many CPU components simultaneously and also create CPU-CPU sideband connectivity.

6.2.2 Challenges of Wireless Sensors

Wireless interconnects as transceiver nodes in a wireless sensor network bring several significant advantages, including adaptability and re-configurability. As mentioned in Sec. 6.1, the physical structure inside a computer presents many challenges for wireless networking. Shown in Fig. 6.2 is an ideal case where wireless transceivers at each CPU can “see” one another; in reality, the tiny transceivers will likely be able to “see” only a small subset of transceivers. At the protocol level, we will need to configure the nWSN in a manner that fulfills the sensor data coverage and platform control objectives while re-configuring itself as a result of sensor node failure, changing demands, loss of connectivity due to changing channel conditions or constantly changing component states. Applications dependent upon the sensor data have little delay tolerance and therefore require real-time response to avoid sub-optimal decisions. Novel GA based approach optimizes the multi-hop path between source and destination end-points as well as assigns adequate bandwidth for efficient monitoring and control.

6.3 Sensor Network Architecture

An nWSN is built by clustering the distributed sensors from multiple servers in a manner that fulfills the data-flow requirements. It consists of sensor devices that connect via wireless infrastructure [64] along with one or many micro-controllers (μC) that monitors the received data and takes an appropriate control action in real-time. A nWSN can work using an ad-hoc approach where observability and control requirements can be at fine, medium or coarse-grain. There can be multiple intelligent micro-controllers that cooperate with each other by sharing the collected data. Fig. 6.3 represents a nWSN scenario with 3 servers in a rack. Sensors connected at the bottom of the 1st server and top of the 2nd servers constitute a single nWSN. Similarly, the bottom of the 2nd server and the top of the 3rd server constitute another nWSN. The target of all the sensor data is the micro-controller (μC) that acts as a base station (BS).

In a typical scenario, there are multiple consumers of the sensor data that can dynamically change upon changing environment. Routing and clustering decisions heavily depend upon the respective demands as well as channel limitations of ultra-short-range wireless communication environment, which is characterized by a high propensity for near-field propagation characteristics and dense multi-path. In an extreme case of nWSN, sensors and μC can reside anywhere in the 3-D space. Therefore, clustering and routing decisions are made in order to (a) reduce source-target data transmission delay, (b) reduce congestion across crit-

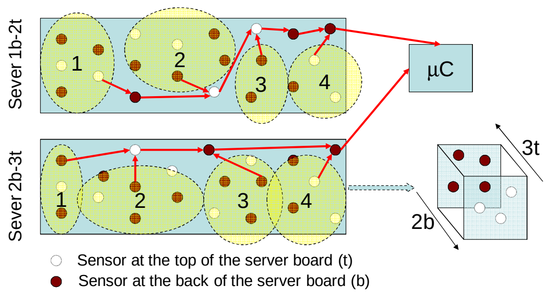

Sever 1b-2t
Sever 2b-3t
1
2
3
4
μC
3t
2b
○ Sensor at the top of the server board (t)
● Sensor at the back of the server board (b)

Figure 6.3: An example of nWSN representation in a server rack. Server board is populated with sensors on both sides. The back side of one server board and top side of an adjacent board constituent a single nWSN. Sensors are clustered with a cluster head and inter-cluster-routers.

ical routes, (c) allocate transmission bandwidth according to optimal demand, and (d) improve the efficiency of control functions within the platform. Typical types of sensors that are used for platform autonomics [65] are thermal, fan, acoustics, power, performance, control-performance, JTAG [66], and reliability sensors (bit error, etc.).

6.3.1 GA Approach: Clustering & Routing Solution

This chapter employs GA based approach [4] that configures the platform sensors into independent clusters with cluster-head and optimal route [33] (Chapter 2). Cluster-head processes the sensor data from its member nodes and transmits to the target via the most cost-effective route. Fig. 6.4 illustrates clustering process where sensors and micro-controllers orchestrate multiple connections between them. Hundreds of these cheap μC's are spread widely over the platform and connected to monitor local sensors. For optimal control, these μC's work cooperatively and exchange data from respective sensors (thermal, power, performance etc.).

Genetic algorithms follow the principle of natural selection. It encodes each possible solution as a binary string (chromosomes) and associates a fitness measure. Successive solutions build as a result of the evolutionary process where one set of selected solutions generates another set for the next generation. Solutions with high fitness are most likely to be inducted into the mating pool with an assumption that they will produce a fitter solution in the following generation. Weaker solutions are naturally discarded. Natural selection is simulated using the roulette-wheel mechanism where elimination of solutions with a higher operating fitness is less likely, although not impossible. Similarly, there exists a small possibility that some weaker solutions may get selected into the mating pool as it may include some component (genes) that may introduce diversity and prove useful following the crossover process. Mathematically, the likelihood of selecting a potential solution is given by

$$P_i = \frac{F_i}{\sum_{j=0}^{N} F_j} \tag{6.1}$$

where P_i represents the likelihood of a solution to be selected for mating pool, F_i represents the operating fitness of an individual solution, and N is the total number of solution elements in a population. GA has proved useful in solving complex problems with large search space that are less understood with little domain knowledge. Moreover, It enhances the ability to handle multiple objectives with a large number of variables and arbitrary constraints.

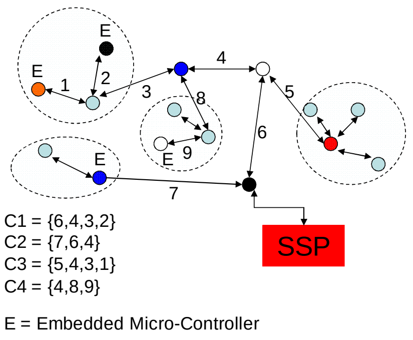


Figure 6.4: Illustration of nWSN topology. $C1$, $C2$, $C3$, $C4$ represent connection between embedded micro-controllers (E) and the respective sensors.

6.3.2 Chromosome Representation

The GA chromosome structure represents building block elements to a solution of the problem at hand that is suitable for the genetic operators and evaluation of fitness function.

6.3.2.1 Node & Route Selection

Node and route selection chromosome structure used in this chapter is similar to [33] (Chapter 2) where individual sensor node is encoded using a 3-bit binary number called 'gene'. These three-bit genes, also referred as 'allele', define the feature of the node and are represented as follows:

001 - Node selected as cluster head (CH).

010 - Node selected as inter-cluster router (ICR).

100 - Node selected as sensor (NS).

Clusters are headed by respective CH that facilitate data-fusion from various node-sensors and routes the processed CH data to the target through inter-cluster router. Sensor nodes and μC constitute the member-elements of nWSN cluster. As a part of the clustering process, sensor nodes attach to a CH by listening to the available CHs advertisement and evaluating the cost of communication. Once attached, sensor node updates its cluster-head with the sensor data. In the similar fashion, CH listens to the router advertisement and selects the low-cost ICR.

Although many possible routes exist between endpoints, only one route is permitted for any sensor data exchange. For route selection, chromosome structure of nodes (CH and ICR) is encoded by $\log_2(N)$ bits, where N is the total ICR nodes that can be reached by this node. Hence an individual in this case is represented by a string that consists of all such nodes with representation to the next ICR. For example, (0010) (0010) (001) (010) represents R_{12}, R_{22}, R_{31}, R_{42} connections, where R_{xy} are the yth route of the xth node.

6.3.2.2 Bandwidth Selection

The bandwidth selection uses *real coded genetic algorithm*, which helps in local tuning and evaluation of the bandwidth fitness with continuous variable. We use simulated binary crossover (SBX) operator [67], that represents gene crossover operator and resembles natural recombination processes. For each bandwidth variable, we choose at random between the equally likely expressions:

$$\beta_i = 0.5((1+\alpha_i)\beta_i^1 + (1-\alpha_i)\beta_i^2) \tag{6.2a}$$

$$\beta_i = 0.5((1-\alpha_i)\beta_i^1 + (1+\alpha_i)\beta_i^2) \tag{6.2b}$$

where β_i^1 and β_i^1 are the parent genes of the ith coefficient (β_i) of bandwidth variable. β_i is also regarded as offspring gene that would act as parent for the next generation. α_i is given by

$$\alpha_i = (2u_i)^{\frac{1}{1+\eta}}, \qquad 0.0 \le u_i \le 0.5 \tag{6.3a}$$

$$\alpha_i = (0.5(1-u_i)^{-1})^{\frac{1}{1+\eta}}, \quad 0.5 \le u_i \le 1.0. \tag{6.3b}$$

where u_i is a random number ($u_i \in [0,1]$) and η_i is the tunable distribution index. While a large value of η_i results in solutions (β_i) close to the parent solutions (β_i^1, β_i^2), a small value increases the likelihood of generating solutions away from parents. Since this parameter controls the spread of the offspring solutions, it can adapt itself based on the current solution using simplex optimization techniques [68]. The individual coefficient (β_i) with the best fitness is adopted for bandwidth selection by using the new coefficients (β_i).

6.4 Fitness Function

Fitness function evaluates the performance of a solution, which in this case is a nano sensor network design (nWSN). It encompasses significant factors that are necessary for the performance and survivability of the system. GA system maximizes this function as a part of evolutionary optimization. In the case of nWSN, fitness function measures the optimality of the routing, functional distribution, resource allocation (bandwidth) and quality of control in a sensor network. This fitness function competes with other fitness criteria [33,41] (Chapter 2, 3) while operating as a dynamic process that iterates over system's life-time. Necessary elements of fitness function are described as follows.

6.4.1 Coverage Fitness (CF)

Coverage fitness rewards a successful source-target connection and penalizes a missing, over-provisioned or non-existing connection. In many cases, a direct source-target connection might not be possible either due to multiple obstacles or due to disconnected nodes. Multiple routes could exist for a given connection. Routes are selected for end-to-end connection n as the result of route selection (Sec. 6.3.2.1). CF is given by

$$\chi_{nm} = \max\left(0, \frac{p_{nm} - p_t}{p_t}\right) \quad ; \; p_t \; > \; 0 \tag{6.4a}$$

$$\xi_n = \left(\prod_m^M \psi(\chi_{nm})\right) \cdot \left(2 - \frac{1}{M} \cdot \sum_m^M \min\left(1, \chi_{nm}\right)\right) \tag{6.4b}$$

$$\psi(k) = 1, \quad \text{if} \;\; k \; > \; 0 \; ; \; \text{else} \;\; 0$$

$$\text{CF} = \frac{1}{2N} \sum_n^N \xi_n \tag{6.4c}$$

where p_{nm} represents communication energy relationship between m endpoints for connection n (as measured by individual sensor node) and p_t represents energy threshold index.

6.4.2 Route Selection Fitness (RSF)

RSF rewards the optimal delay paths according to the target requirements. Higher than expected delay can result in lower quality for performance connections and may reduce the effectiveness of the received data. Every route is tested on (a) congestion – estimated by evaluating missed or delayed packets and (b) average latency – estimated by evaluating the round trip time (RTT) of sensor data transaction. RSF is evaluated as

$$RSF_d = \frac{1}{N} \cdot \sum_{n}^{N} \min\left(1, \frac{|\tau_n - \bar{\tau}_n|}{\bar{\tau}_n}\right) \tag{6.5a}$$

$$RSF_c = \frac{1}{R} \cdot \sum_{r}^{R} A \cdot \frac{T_r}{\hat{T}_r} \tag{6.5b}$$

$$\text{RSF} = 1 - \frac{1}{2} \cdot (RSF_d + RSF_c) \tag{6.5c}$$

where τ_n is the measured delay between end-to-end nWSN traffic for connection n, $\bar{\tau}_n$ is the expected delay for end-points serviced by connection n, T_r is the number of packets either missed or delayed by ICR or CH r, $\hat{T}_r$ is the total packets serviced by ICR or CH r and A is an amplification factor.

6.4.3 Bandwidth Selection Fitness (BSF)

Each connection is allowed certain bandwidth for optimal distribution of the total available wireless bandwidth. Normally, data for each connection should be transmitted at a rate not lower than the sensor sampling frequency, or the sensor data would stall at source nodes. Excessive use of bandwidth for few connections can cause non-availability or over-provisioning of bandwidth. The maximum available bandwidth for a connection is limited by the slowest path in a multi-hop connection. BSF for N required connections ($B\bar{W}_n > 0$) is defined as

$$
\begin{aligned}
BW_n =& \min(BW_{n1}, BW_{n2} \cdots BW_{nm}) && \text{(6.6a)}\\
\text{BW1} =& \frac{1}{N} \cdot \sum_{n}^{N} \frac{|BW_n - B\bar{W}_n|}{B\bar{W}_n} \quad ; \; B\bar{W}_n \; > \; 0 && \text{(6.6b)}\\
\text{BW2} =& \min\left(1, \frac{|B - \sum_n^N B\hat{W}_n|}{B}\right) \quad ; \; B \; > \; 0 && \text{(6.6c)}\\
\text{BSF} =& 1 - (BW1 + BW2)/2 && \text{(6.6d)}
\end{aligned}
$$

where $B\bar{W}_n$ is the required bandwidth of end-to-end connection i between the source and the target, BW_n is the lowest bandwidth of a single hop m (BW_{nm}) in a multi-hop connection n, $B\hat{W}_n$ is the sum of the total bandwidth allocated to connection n including all hops, B is the total available bandwidth for all connections in a nWSN. Although the available bandwidth is B, it can be over-subscribed if there is sufficient probability that not all bandwidth is going to be utilized at a single time.

6.4.4 Quality of Control Fitness (QoCF)

QoC paradigms enhance the flexibility and adaptability of nWSN with respect to the changing environmental conditions or varying demands. In an adaptive system, sensor data rate requirements increase or decrease based on the redundancy of information transmitted. QoC efficiency changes as a direct consequence of adaptation to the data rate of each source nodes at run time. Deadline misses or delayed observability can result in unstable control and missed opportunities. In many cases slow data rates may be adequate for effective control due to redundancy in the observed samples. Sensors may scale down the sampling rates based on the feedback. Although we may observe higher delay at some sensor nodes, but the QoCF may still proclaim higher fitness. The following equation represents the QoCF of controlled variable v at run-time, which is an integral of the modulus by which it deviates (D_v(t))

$$QoCF_v = \int_0^{t_n} \min|1, D_v(t)| \cdot dt \tag{6.7a}$$

$$QoCF = 1 - \frac{1}{N} \cdot \sum_{v=0}^{N} (QoCF_v). \tag{6.7b}$$

The integral (6.7a) could be calculated numerically using the control element of the nWSN. Measurements are delayed by a constant $\delta(t)$ and comparable to the variable being controlled. Optimal tuning of the nWSN radically improves the QoC of the target variable as well as the process stability.

6.5 Results and Discussion

The end goal of maximizing the platform control efficiency is achieved by maintaining on-demand connectivity and minimum end-to-end latency in a dynamic system with a large number of diverse management sensors. This can be done by sustaining low-latency routes between end-points, optimal sampling frequency at the sensor node as well optimal observability at the target. Any congestion at the ICR nodes can cause momentary delay which can cause oscillations, instability and low quality-of-control. We run GA periodically at run-time to accommodate changing telemetry demands and measure end-to-end latency and congestion on routes. In our experiment, latency is measured by calculating the average buffer storage at each sensor node. This latency data is also used as a training set for the next training cycle executed by running GA (Eq. (6.5c)).

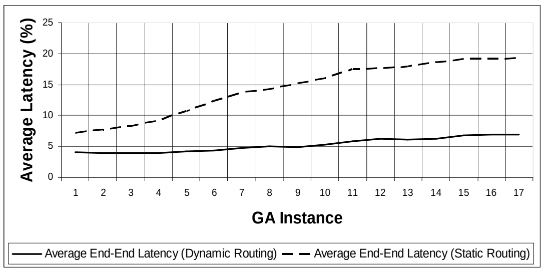


Figure 6.5: Average latency between the sensor/controller end-points with varying sensor sampling requirements.

The nWSN infrastructure simulation uses Tiny-OS component libraries. We use 100 sensors, chosen in a manner that there exists an end-to-end connectivity with only 35% of the total platform sensors. Each sensor reads 32 bits of telemetry data. GA executes on a separate machine by utilizing the training data collected between subsequent runs. This is a typical scenario in a platform where there are multiple obstructions (heat-sink, fan, wires, DIMMs, etc.) and a powerful embedded controller for manageability functions. Experimental system supports the ability to modify sampling frequency using an input to the infrastructure. Changes in the sampling duration change the traffic patterns in the nWSN that can result in route congestion, delayed observability and deteriorated quality of control. In our experiments, we simulate the dynamic variations by periodically increasing the sampling frequency by 4% for 25% of the senors. In a multi-CPU platform this scenario is evident during shutting off platform components, stressing the system such that nominal operating conditions are exceeded (temperature, power limits, etc.). Additionally, nWSN is equipped with congestion notification from the ICR. Congestion is mitigated by GA supported re-configuration trigger that either introduces new ICR or re-routes the traffic through other nodes by taking into account the latency bounds. Reconfiguration process iterates until the cumulative fitness (Equation 6.8) reaches 95%.

$$\mathrm{CU} = \alpha_1 \cdot CF + \alpha_2 \cdot RSF + \alpha_3 \cdot BSF + \alpha_4 \cdot QoCF \tag{6.8}$$

In most cases bandwidth is restricted by nature and properties of the interconnects (SMBUS, PMBUS, PECIBUS, SST, etc.). We evaluate the performance of this approach by simulating (a) the baseline condition where sensors are statically routed using pre-defined interconnects and bandwidth and (b) the proposed GA approach where nWSN clusters, sensor end-point bandwidth and routes are configured dynamically according to the changing requirements and demand.

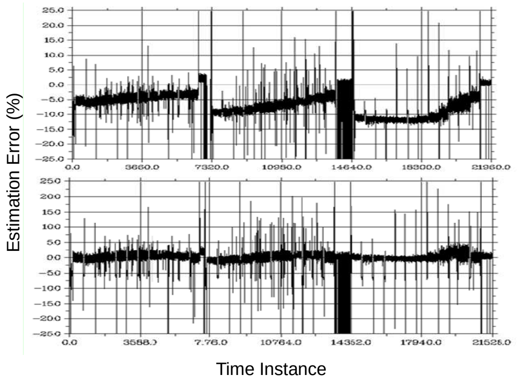


Figure 6.6: Average power estimation error by power sensors. (Top) Estimation Error with static routes (Bottom) Estimation Error with adaptive routes.

Fig. 6.5 compares the latency variation between GA supported dynamic routing and platform default static routes with each run of GA. We observe 5%−20% latency reduction with dynamic routing against a conventional static routing approach. Data flow latency reduction improves the quality of control by promoting finer grained resource monitoring. This helps detect and isolate any peak activity, which may otherwise be hidden. Improvements in peak detection present an efficient monitoring opportunity for cloud-computing environment for resource charge-back. For example, GA based solution improves the prediction function for estimating the component power consumption as a result of improved ability to monitor sensor data trends faster (15%−25%) with smaller sampling window size. Improvements in power prediction enhances the quality of service and ability to auto-provision the resources accurately for the future slots. Fig. 6.6 shows the power estimation error due to latency in recording the data. We archive a high degree of accuracy (Fig. 6.6b) as a result of improved monitoring due to reduced latency in case of adaptive routing using GA. Static case shows up to 8% error (Fig. 6.6a) in estimation.

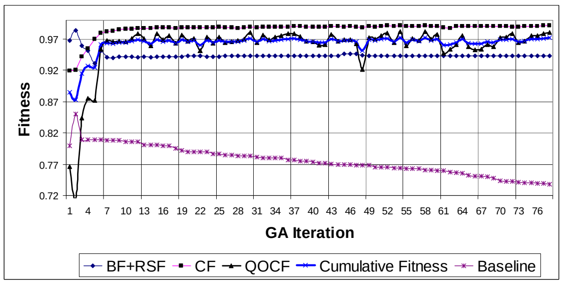


Figure 6.7: Fitness component (CF, BF+RSF, QoCF), baseline QoCF and cumulative fitness as a function of GA iteration.

Fig. 6.7 demonstrates the the fitness trend of individual objective components as a function of GA iteration. Quality of control (Eq. (6.7b)) shows on an average 97% confidence level, while bandwidth and route function attain 94% fitness. Compared with the legacy environment with pre-configured static routes, dynamic re-configuration using GA solution improves the control performance by about 20−25%. QoCF measures the stability of process control, where individual processes tries to maximize performance for given power and thermal limits for each component. Each process reads the performance data, estimates the power data and performs the control function that changes the power level to maintain desired performance levels. Additionally, each process trains itself based on the fuzzy data. A high QoCF represents an ability to maintain desired performance for a given power/thermal limits and accurate estimation of component power from trained model.

6.6 Conclusion

We described a unique, wireless sensor network infrastructure, nWSN, that could be used in IT data centers (& high-performance computing) and compute platforms that involve a large number of sensors and controllers connected in an ad-hoc manner. This infrastructure supports the manageability and autonomics backbone for observability and control of critical aspects of the platform ranging from resource estimation, performance re-balancing, power/thermal control, fault detection, resource charge-back, and failure prediction. We developed a novel scheme based on genetic algorithm that allows re-configuration and automates coverage, connectivity and bandwidth to improve the efficiency of functional control. We compared the performance of the proposed approach against the static autonomics system where routes and bandwidth are constrained by the type of interconnect. The improvement in the delay characteristics ranges from 15% to 20%; the improvement in estimation accuracy due to lower delay between end-points is $8-10\%$. High accuracy and low delays enable scalable communication infrastructure with predictable bandwidth and latency to provide demand based plug-and-play interconnection. Future work includes optimizing the routes in a manner such that they maximize the power-efficiency at variable loads in a system. This includes overprovisioning the bandwidth by controlling the sensor activity periods during which they transmit the data.

Chapter 7 – Monitoring and Control in Data Center

7.1 Introduction

Context awareness in a data-center represents the ability to detect, analyze, and respond to the changes in the local environment. It is therefore essential to create an infrastructure of sensors that monitors the physical properties of the dynamically changing environment. An efficient sensor network mandates a real-time response to a query or real-time notification of an event including the time to process the specification. It also helps to model organizational and technological choices and avoid competition for limited controls from multiple applications. Additionally we can leveraging the degrees of freedom of the underlying hardware technology and equip the system with a resource-efficient runtime support system. Various attributes of the sensor network are related to auto-discovery, addressability, event-signaling, uniqueness, abstraction model, grouping, and ubiquity. These sensors are supported by the ability for metering, data synthesis, and alerting with the following properties.

IT equipment require adequate cooling for reliable operation to avoid danger of creating potential hot spots. Therefore data centers are equipped with air-conditioning units that use low set points and very high fan speed. AC units are independently-operating cooling units that need to be operated at optimal capacity to adequately mitigate the effect of hot-spots. Unfortunately, most of the data-centers lack adequate automation and visualization tools to manage the cooling units in an efficient manner. This results in limited visibility into how the heat is generated, propagated and exchanged in the data canter. As the density of data center increases, it creates an imparitive for energy savings for cooling by efficient management and timely use of accumulated data. Real-Time sensor data monitoring allows IT operator to improve operational efficiency and reduce the power usage effectiveness (PUE) due to improved cooling control and manageability. Power Usage Effectiveness (PUE) measures the data center overhead and defines the ratio of the total facility power consumption over the power used by the IT equipment. High PUE normally reflects the lack of visibility in the data center operating conditions. On the server management front, it is also important to identify the source of thermal alarms so that the problem can be addressed effectively without over-cooling by decreasing the CRAC's temperature settings. Other factors that can improve the data-center efficiency and reduce energy cost are (a) optimal server node utilization (b) fan-speed control (c) workload distribution according to server fingerprint (d) server-node level power capping etc. This requires additional extra sense points on the server nodes and timely delivery of sense data to the management agent that controls the workload allocation among server nodes. Large number of sense-points generate enormous amount of data that needs to be delivered to the provisioning server and data-center manager in a time-bound manner. This data is then used for dynamic server provisioning strategies to control resource utilization and load fluctuation to avoid thermo-instability. Slow or delayed telemetry results in unstable-system, unscheduled shut-downs and higher PUE.

Wireless sensor network (WSN) can act as low-cost candidate for monitoring task as it is nonintrusive, can provide wide coverage, and can be easily repur-

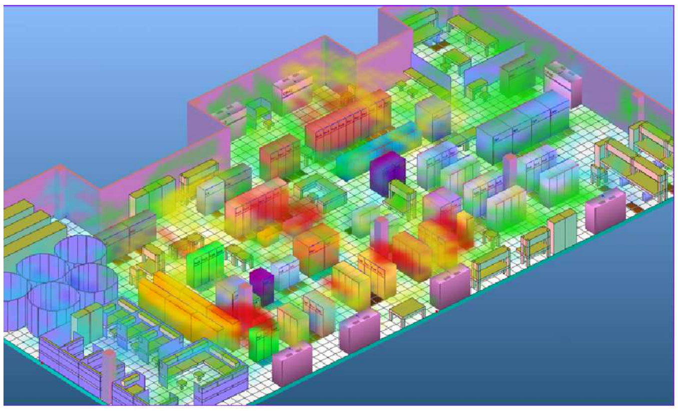


Figure 7.1: Three-dimensional temperature distribution in the data center illustrating hot and cold regions in the data center. (Courtsey: Innovative Research Inc.)

posed. In a data-center, WSN system acts as a network of devices that provide real-time monitoring ability to observe and manage space-conditioning energy. WSN can also be useful as a debugging tool to monitor hotspots, benchmarking and system forensics including alerts and alarms. The non-invasive wireless sensors can measure and trend air temperature, humidity, air particle count, current, power, node-utilization, workload performance, service-level-agreements (SLA) and other data. These trends can be summarized into graphical representations of temperature profiles within the data center. In general WSN benefits can be generalized as follows:

- Cooling Performance and Temperature profile Visualization.
- Floor Tile Tuning.
- Hot-Spot detection
- Humidification Control

- Failure Prediction
- Real-Time PUE monitoring
- Air conditioner/Air handler unit control and coordination.
- Workload Fingerprinting and migration
- REsource Monitoring

7.2 Data-Center Challenges

Technology and business challenges such as virtualization, load consolidation, real-time troubleshooting and SLA guarantees require a robust and adaptive server management plan for enterprise. Majority of data center issues are related to over-utilization of resources, application failures, data security, power usage effectiveness and infrastructure costs. This requires proactive solutions that are business intelligent and built over a network of sense-points that are guaranteed to deliver reliable trends and measurements in a reliable and timely fashion. Management solutions ensure that data center operations run smoothly 24/7 on all 365 days of the year. But managing Data Centers is becoming complex as technology advances along with the skills necessary to manage. According to 2007 estimates, cost of per square-foot construction in data center can be well over $1000. Since it is expensive to build new data-centers, the best option is to improve usage of existing facility through better power and resource management. Various methodologies are used to achieve better PUE, some of which are described below:

- *Increasing the RACK density* – Improve the RACK utilization (and available power) by using the real-time work-load measurements instead of label-power. Unused power can be capped allowing additional servers be added to the RACK.
- *Improved cooling* – Accurate server power measurements enables cooling adjustments according to the RACK temperature.
- *Load Balancing* – Sense points measure power and thermally constrained systems that are dynamically capped to avoid outage or failure.

Unfortunately, data-center control heuristics lack real-time power consumption information of servers and operate on the conservative side. The server racks are populated according to rated power from the server nameplate. This leads to unnecessary guardbands that overstate servers actual power need over-cooling than is necessary for the actual loads. This in addition to identifying alerts related to abberant nodes, resource overloading and failures makes the management of data-center a highly complex problem. Adding to the complexity, datacenters can have anywhere from 10,000 to 100,000 nodes. Not including environmental sensors, each node can generate upto 320 bytes (2.56 K bits) of data burst per sampling period. This data reflect power, thermal, utilization, reliability, QoS trends of CPU, DIMM, Fans, Network, I/O Hub, Workload performance components in the system. This data along with environmental sense-points data has to be delivered to the Supervisory Control and Data Acquisition (SCADA) system in a guaranteed time to monitor, analyze and display consolidated real-time process control data. SCADA system uses the sense point data to produce models and tools for facility management and performance optimization. Key components that are measured can be summarized as follows:

- *Facility layout*: The RACK, CRAC, and power distribution layout act as the basis for cooling efficiency and data center capacity.
- *Cooling system*: The cooling system (CRAC, water chillers, air economizers, and (de-)humidifier) consumes large amount of non-critical electrical load.
- *Power system*: Detailed monitoring of the power consumed by various server nodes is necessary for efficient control.
- *Server performance*: Server utilization is represented by key components such as CPU, Storage, DIMM, and Network. Measuring power, thermal, performance and reliability characteristics is key to characterizing heat distribution in data center.
- *Load variation*: Server application performance characteristics deliver meaningful indicators of system load, such as queries per second or concurrent users.
- *Environmental conditions*: Environmental trends related to heat distribution and hot-spot detection had been relitavely difficult to address due to lack of fine-grained visibility.
- *Alerting*: Ability to alerted on impending faults or threshold crossings for monitored health metric.
- *SLA Monitoring*: Analyze device performance metrics to identify SLA impacts and ways to improve SLA guarantees and avoid penalties.

Data accumulated from various sources (Server & Environmental) is fed into the database where it is processed for developing statistical models that correlate physical and performance characteristics. Once models are built, they are then used for analysis, prediction, optimization, classification functions. These functions can be summarized as:

- *Real-time monitoring and control*: This requires real-time thermal management, hot-spot detection and mitigation, load balancing etc.
- *Incremental Deployment*: Server nodes are replaced or deployed based on thermal distribution, power availability, cooling capacity etc.
- *Smart Provisioning*: Workloads can be consolidated to realize energy savings. It is relatively simpler to distribute load according to cooling capacity instead of controlling the cooling capacity based on varying load.
- *Failure analysis and health monitoring*: Failure analysis and prediction is an integral part of data-center management. Like any other equipment, electricals in server undergo ageing process or experience thermal stresses. This require tuning process that re-margins the circuits to an optimal value so that servers can perform according to specifications.

7.3 Platform Management Instrumentation in Server Nodes

A platform management subsystem comprises individual components that are managed based on the principals of equilibrium strategies of the platform as a whole. Platform strategies are defined relative to global platform policies, which

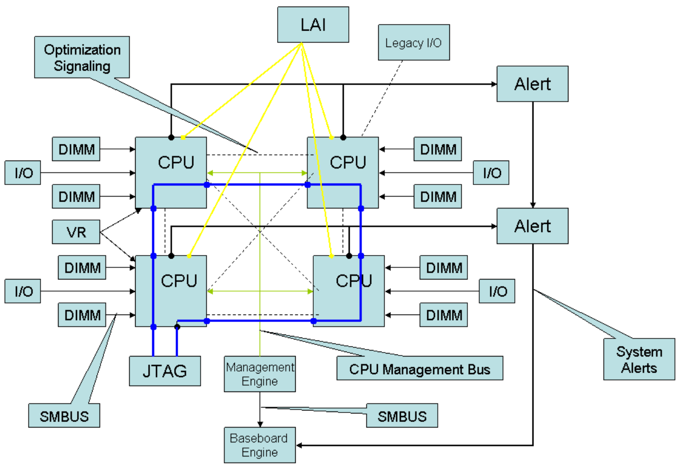


Figure 7.2: Today's state-of-the-art server containing four processors on a single blade; Block diagram of the high-speed and low-speed interconnect between the four processors on the chassis.

are then broken down into individual sub-component policies. Global and local policies are dependent upon the environmental variations and thus change dynamically for the survivability of the platform. It is also interesting to note that most of the components in an autonomics system are constantly adapting with respect to the interacting environment. Any adjustment in the operating characteristics of one component can cause imbalances and oscillations in other components. A platform management framework provides environmental monitoring agents that constitute the sensor and motor channels for monitoring and control functions (Figure 7.2). Sensor channels are used for the probing functions like temperature, voltages, current, fan speeds, utilization, state changes, and hardware errors. Control functions use motor channels in order to maintain platform homeostasis by taking appropriate actions that administer the viable limits of individual components.

Modern platform typically use centralized platform management approach that delivers the management functions through a centralized management engine (or agent). In such architecture, a platform management subsystem typically includes one or more microcontrollers that support access and control methods through sensor and motor channels. These microcontrollers can serve as intelligent controllers that serve as access points for all the sensor and motor services. In many cases, these services are abstracted using a data-link protocol specific to a channel on which it is accessed. . Slow telemetry and delayed responses can result in inefficient regulation and un-sustained fluctuations in a goal-oriented system. Various channels that are used for collecting server telemetry data can be described as below:

- *Platform Environment Control Interface (PECI)* is a one-wire bus interface [69] that provides self-clocking and a communication channel between Intel processor and chipset components to independent monitoring or control devices. It is flexible in supporting a variable data transfer rate for each message that is negotiated by the participating devices.

- *SMBUS* [70] provides a standard mechanism to connect with peripheral devices that are required for environmental monitoring and control. In recent years SMBUS has gained popularity with smart batteries, temperature, fan, voltage sensors. In many mainstream servers, the SPD EEPROM and thermal sensors are connected to the system's SMBus.

- *Management Component Transport Protocol (MCTP)* is a media independent protocol [71] for intercommunication among intelligent devices within the platform management subsystem of a managed computer system [MCTP 2016].

- *Network Controller Sideband Interface (NC-SI)* is defined as the interface between a management controller and one or multiple network controllers [72]. This interface is responsible for providing external network connectivity for the management controller, which enables a common interface definition between different management controller and network controller vendors.

7.4 Wireles Sensor Networks in Data Center

Wireless sensors can act as key enabler for efficient management of data center. There are sevaral advantages to using WSN in data centers. WSNs fascilitate real-time decisions that can help to operate a data center at peak efficiency. Various advantages that makes the WSN an ideal candidate for data centers can be enumerated as below:

- *Non Intrusive*: Data centers are senstive to any changes to the existing infrastructure that may require additional capital expenditure or retraining. It is therefore essential that sensors be deployed without any additional infrastructure or cabling requirements. These sensors should organize themselves and deliver senser data in a guaranteed time. Large number of sensors within data center can generate enormous amount of data in a bursty fashion. Wiring these sensors through TCP/IP link is not practical. Wireless sensors can fulfil these requirements as they operate out-of-band without interfering with exsiting operations.

- *Adaptive Deployment*: Data centers undergo constant deployment, replacement, provisioning, comissioning/de-comissioning of RACKS and servers in a dense environment. Wireless Sensors provide transparency to such changes and avoids any overhead of deployment, inventory monitoring, locality monitoring, asset tracking and device configuration. Furthermore, WSN provide ability for on-demand monitoring for new (or replaced) IT equipments being tuned or integrated to the existing infrastructure.
- *Low Cost of Ownership*: There may be large number of sense points in a data center. WSN provide a viable solution to fine grained sensor monitoring due to low cost of hardware, infrastructure, labor, and maintenance.
- *Seamless Integration*: Environmental sensors are organic to the data center management system. The sensor network can easily be integrated with the rest of the infrastructure, such as facility management, asset management, and performance management.
- *Context Awareness*: Wireless Sensors enable context awareness by measuring its locality relative to other nodes using traditional Time of Arrival (TOA) methodologies. Context awareness can enable self-management of RACK level policies based on local cooling capacity and power availability.

7.4.1 Challenges in Data Center Deployment

One of the objectives of this theses is to understand the technical challenges faced in this unique environment due to deployment of wireless sensor networks. Traditional sensor network research has mainly focused on increasing the life of wireless sensor network for optimal battery usage. In contrast, data centers pose unique constraints that need to be addressed for effective management through WSN. Some of these challenges can be summarized as below:

1. *Long Term Monitoring*: In a data center, primary objective of WSN is to facilitate high yield data transfer to the autonomous control system (ACS) that manages the overall health of the data-center and associated servers. Battery life is not the primary limiting factor in this case, as the sensors are externally powered and are not placed in a hostile environment. In contrast with traditional battery-limited sensors, data centers require long term monitoring with high data yield for efficient power management and control. *Therefore, one of the objective of the research is to identify the relationship between sampling period and data yield.*

2. *Sensor Placement*; In a data center, large number of sensors are packed and distributed on thousands of RACKS that may be within one-hop communication range of each other. Each sensor can generate upto several kilo-bits of data burst that needs to be delivered to the management server in a guaranteed amount of time. This requires achieving high data fidelity in a noisy channel that may further be deteriorated by the presence of metallic objects (racks and panels) that can alter the radio propagation pattern. One solution is to dynamically adapt the nodes membership on each channel according to link quality changes. *Therefore, second objective of the research is to increase data transfer efficiency in a noisy environment using channel diversity, limiting radio power and optimal configuration of cluster membership.*

3. *Zero-Touch Management*: Data center management is traditionally low-touch environment with little or no expertise to manage wireless. Therefore, wireless sensor network needs to be self-managed and should't require any manual intervention or configuration like node IDs, channels allocations, or power levels. *Therefore, essential condition of our research is to minimize any manual intervention or configuration.*

4. *Data Generation*: Sensors in data center generate continuous stream of data that needs to be delivered to collection trees in a bandwidth constraint large-scale network. This requires optimal balancing of the network routes in a manner that maximizes the throughput and minimize the number of hops. Therefore, we balance the routes such that no single route is over-utilized or under-utilized.

5. *Spatial Correlation*: Large volume of data generated by continuously active sensors can overwhelm the available bandwidth in the sensor network. Finally we propose context-aware clustering to reduce transmissions in cases where the observed data corresponds to the norm expected by the system in a given context.

Wireless data center management: Sensor network applications and challenges

— Rahul Khanna, Huaping Liu, and Thanunathan Rangarajan —

In Microwave Magazine, IEEE 15, no. 7 (2014): S45-S60.

Chapter 8 – Evolutionary Approach to Sensor Networks in Data Center

8.1 Introduction

The modern data centers are essential to fulfilling ever-evolving compute demands around cloud computing, big data and IT infrastructure. These data centers are facilities (Figure 8.1) that house computer systems and associated components, such as networking and storage systems. In order to operate a data center, it is required to have power supplies, network connections, environmental controls (e.g., air conditioning, humidity) and security infrastructure. Technology and business challenges such as virtualization, load consolidation, real-time troubleshooting and SLA guarantees require a robust and adaptive server management plan for enterprise. Majority of data center issues are related to over-utilization of resources, application failures, data security, power usage effectiveness and infrastructure costs. This requires proactive solutions that are business intelligent and built over a network of sense-points that are guaranteed to deliver reliable trends and measurements in a reliable and timely fashion. Since it is expensive to build new data-centers, the best option is to improve usage of existing facility through lower infrastructure overhead to deliver better resource management. An optimal sensor network would perform real-time sensor-data collection and deliver (a) *improved RACK utilization* (b) *improved data-center cooling* (c) *improved load-balancing* through dynamic capping of thermally constrained systems.

On the infrastructure front data centers face considerable challenges in seamless integration of telemetry and control functions. These functions are essential to management tasks related to power capping, cooling, reliability, predictability, survivability, and adaptability control. It is therefore essential to create an infrastructure of sensors that monitors the physical properties of the dynamically changing environment. The conventional approaches to support distributed sensor data collection and control using wired solutions are static, expensive, and non-scalable. Sensors and control agents supporting this telemetry are a part of a dense and noisy network that are scattered across the data centers. An alternative approach for this unique environment is to use wireless sensor network to improve data efficiency and real-time delivery.

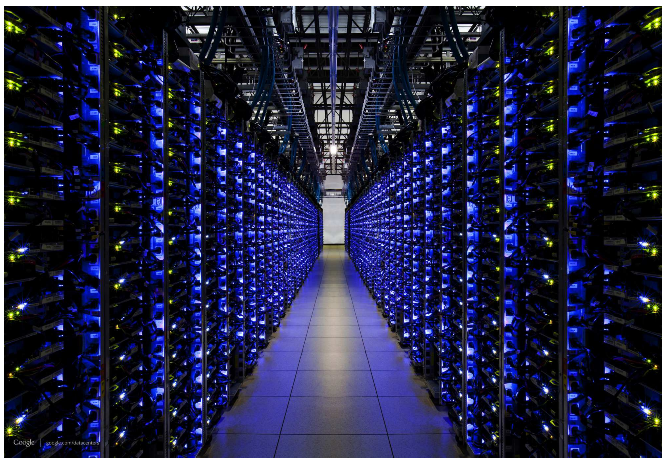


Figure 8.1: Google NC Data Center

Datacenters can have anywhere from 10,000 to 100,000 nodes and each node can generate upto several kilo-bits of data burst per sampling period. Accurate assessments and analysis of energy efficiency opportunities in a data center requires monitoring of multiple environmental parameters (such as temperature, dew point, and pressure in the data center at many locations and elevations), metering of electrical power, utilization characteristics of each compute node from the electrical sub-station to its end use. The sense-points data and environmental data is delivered to the Supervisory Control and Data Acquisition (SCADA) system in a guaranteed duration to monitor, consolidate and analyze real-time process control data. SCADA system uses the sense point data to produce models and tools for facility management and performance optimization (Figure 8.2). Monitoring so many parameters is expensive and logistically difficult through a conventional wired monitoring system. According to recent study conducted by U.S department of Energy, the cost of hard-wired systems ranges from \$1,000 to \$1,500 per sensor node. Wireless sensor networks achieves equivalent performance at a projected cost of \$100 to \$150 per node (10x Savings). Moreover, wireless systems eliminate the key logistical barrier of placing additional wiring in overcrowded racks. Because they are easily expandable and relocatable, wireless systems also provide flexibility to grow and adapt as a data center evolves over time.

8.2 Data Center: Thermal Monitoring in Dynamic Environment

A datacenter is a highly dynamic environment. In this environment, hot-spots can be created as a result of temporal events (e.g., increased workload on a set of

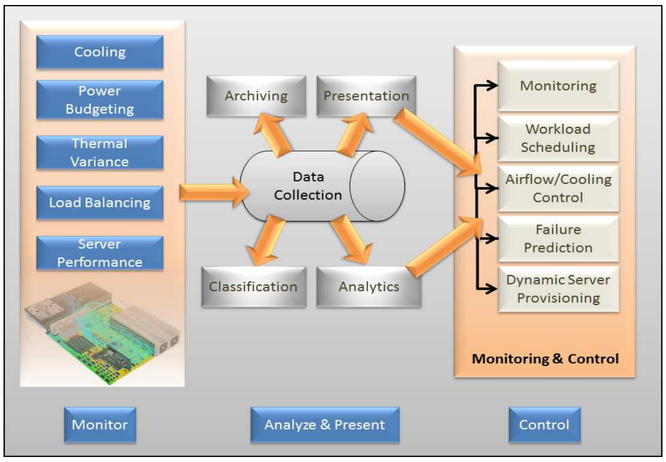


Figure 8.2: Technology and business challenges such as virtualization, load distribution, real-time troubleshooting, SLA guarantees and efficient cooling require a robust and adaptive server management plan for enterprise through efficient and timely monitoring.

servers) or spatial events (inefficiency of the CRAC units in delivering requisite cooling to a particular region in the datacenter). In particular, the dynamic nature of workloads means that a bad decision with regard to thermal management could severely impact operational costs, as side-effects like hysteresis can cause both increased energy consumption as well as unwanted workload movement. Hence, timely analysis of sensor events is vital to the successful operation of the optimization algorithms. There are three primary reasons why a naive brute-force decision-making approach would prove inadequate:

1. The dynamic nature of workloads
2. Precision and timeliness of sensing physical phenomena such as heat and air flow
3. Interplay of data center environs and running workloads

A close examination of the server platform is necessary to address the above points in the right perspective. At the heart of the platform is the microprocessor chip or the System-on-chip (SOC), which is the primary source of heat generation in the platform. On-die sensors measure heat production in the chip, and platform cooling devices such as fans then calibrate their air flow accordingly to provide cooling. As such, if the die temperature sensor is sampled periodically, one would find a net accumulation of heat due to the workload, and a net dissipation of heat due to cooling action. Predicting the heat dynamics of the datacenter hence involves understanding several dynamic factors at once, assimilating sensor data, and then constructing an instantaneous thermal snapshot of the datacenter, as shown in Figure 8.3. It can then be used to predict future thermal behavior and effect control decisions that can minimize hot-spots or optimize cooling. For example, an optimum decision is one that produces successive thermal snapshots with progressive diminishment of hot-spots. Most prediction algorithms are based on periodic sampling of the states of several entities at once: the datacenter's sensor network, sensors within servers, the datacenter's cooling infrastructure, cooling devices (e.g. fans) within servers, and finally, the set of workloads running on the servers. The prediction logic is hence a discrete-time system, and it predicts the temperature rise in the i^{th} platform at a given sampling instant T as:

$$\hat{\varphi}(T) = \sum_{t=T-d}^{T} \mu_t P(t) + \sum_{t=T-d}^{T} \omega_t T(t) - \sum_{t=T-d}^{T-1} \lambda_t C(t) \tag{8.1}$$

This prediction is based on observation of sensor and workload data (that we propose to transmit over WSN) over the last d samples for a reasonable insight

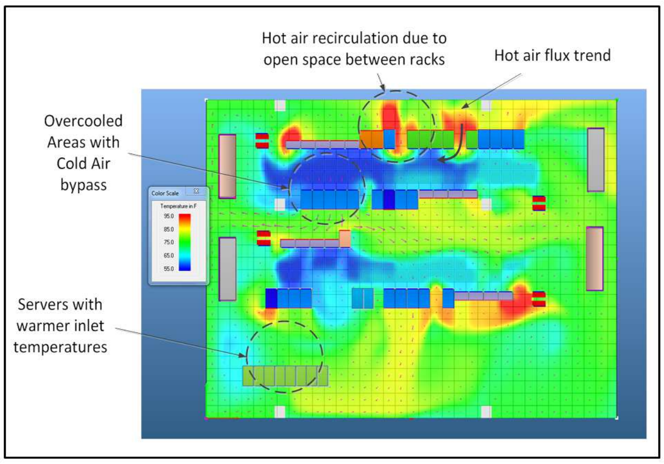

Hot air recirculation due to open space between racks
Hot air flux trend
Overcooled Areas with Cold Air bypass
Servers with warmer inlet temperatures

Figure 8.3: Thermal Snapshot of the Datacenter

into the workload and environmental dynamics. The first term on the right pertains to the accumulated power consumed by the running workload (P), which causes a rise in the junction temperature of the component, and is eventually dissipated as heat. The second term pertains to the effects of the local environment (T) in which the server is running, e.g. the effect of a hot-spot caused by other racks or other equipment in the vicinity of the server, or the redundant cooling delivered by an over-calibrated CRAC unit. The third term refers to the cooling performed purely by the server's cooling system (C), e.g. convective cooling or liquid cooling. The constants μ_t, ω_t, and λ_t must be evaluated at each sampling period and adjusted to reflect the latest thermal snapshot of the system. Equation (1) must be repeated for each server in the datacenter to yield a Prediction vector and its associated coefficient matrices, which is the basis for implementation of an accurate thermal prediction model in the datacenter management system that can forecast net cumulative temperature rise across equipment in the datacenter at each sampling instant and make decisions

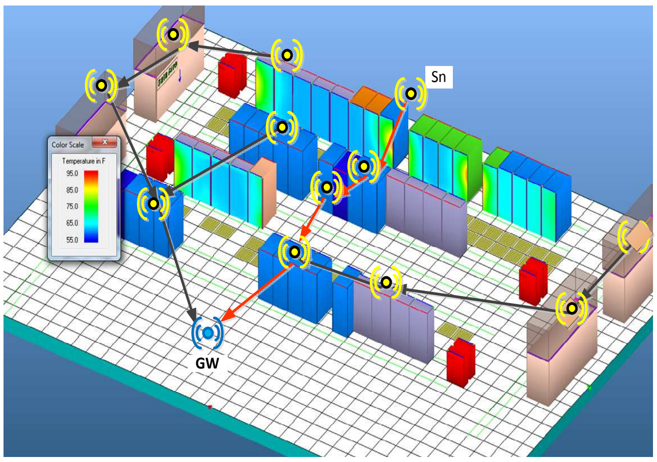


Figure 8.4: Wireless Sensor Network deployment in datacenter

related to workload placement and facilities adjustment in order to eliminate hot-spot conditions and balance workload to match the delivered cooling by the facilities infrastructure. At the beginning of each sampling period, the prediction model must assess its previous decision relative to the current thermal snapshot. A bad decision must be penalized and corrected using a machine learning approach such as reinforcement learning. The objective in such algorithms would be to reduce the error:

$$E_t(T) = [\hat{\varphi}(T) - \varphi(T)]^2 \tag{8.2}$$

A key observation here is that the availability of timely sensor data from the WSN is vital to ensure accurate decision making. Stale data could produce large deviations from the ideal conditions, resulting in severe hysteresis and complete loss of control, adversely affecting overall Total Cost of Ownership (TCO). Dense WSN installations with highly parallelized subsets enable management software with real-time data for forecasting environmental trends (e.g. direction of flow of heat from hot-spots in Figure 8.3) and perform thermal-aware workload placement.

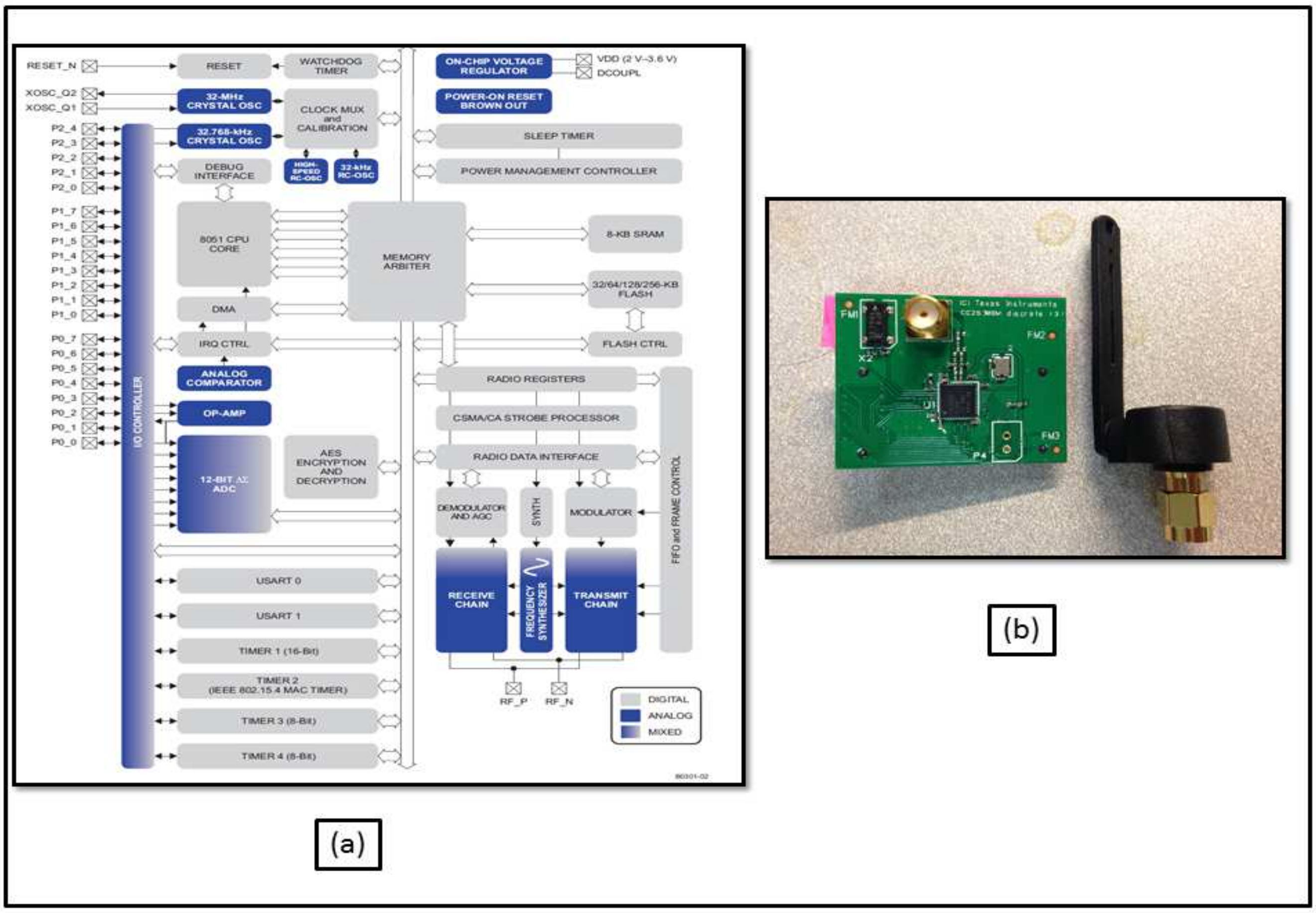


Figure 8.5: TI CC2530 SoC [1] solution for Zigbee network. (a) CC2530 SoC Block Diagram (b) CC2530 based zigbee platform with antenna.

8.2.1 Data Center: Sensors & Gateways

Datacenter wireless sensor network (WSN) comprises of a hierarchy of sensors, gateways, data sink and data analyzers. *Sensors* are edge devices that collect the data related to thermal, power, performance, locality, airflow information and transmit that to the data sink reliably. The sensor data is retained in its local memory till that data is acknowledged by the router. Figure 8.5 illustrates one such widely used System-On-Chip (SoC) “TI CC2530” for collecting and transmitting sensor information across WSN. It supports Zigbee protocol over 2.4-GHz IEEE 802.15.4 Systems. The CC2530 combines the performance of a RF transceiver with an industry-standard enhanced 8051 MCU, in-system programmable flash memory, 8-KB RAM, and many other powerful features. The CC2530 power efficiency is supported by various operating modes and short transition times between those operating modes that ensures low energy consumption. It supports RF frequency range from 2394 MHz to 2507 MHz, programmable in 1-MHz steps with 5 MHz between channels. This dynamic range facilitate channel diversity that allows the sensor node to select the best channel with low noise and high data throughput. As proposed in section 8.6, cooperative information processing by all sensor nodes in a cluster results in an optimally configured wireless sensor network.

Gateway acts an an intermediary between a sensor and a data sink. Gateways are employed to improve the data throughput, eliminate information redundancy and exploit locality information for information compression. Figure 8.6 illustrate a Galileo Platform that consists of Intel Quark SoC X1000 Application Processor, a 400 MHz 32-bit Intel Pentium-class system on a chip.

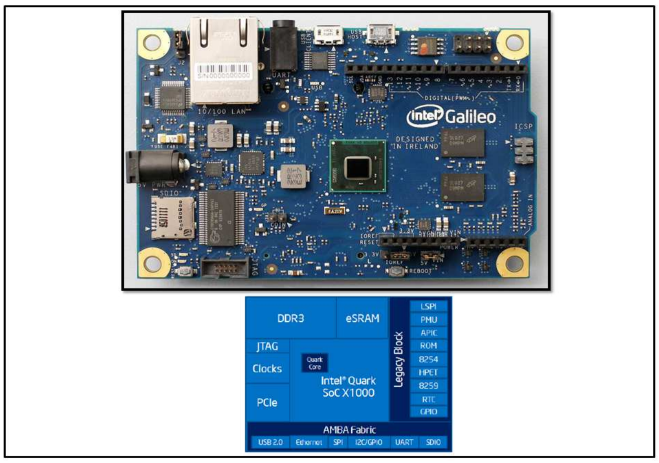


Figure 8.6: WSN Coordinator: Intel Quark SoC X1000 Application Processor

It is capable of providing back-end support for collecting and compressing the sensor information from multiple sub-nets and transmitting that information to the data sink. In addition to Quark SoC, it supports 256 MByte DRAM, 512Kb embedded SRAM, a full sized mini-PCI Express slot, 100Mb Ethernet port, Micro-SD slot, RS-232 serial port, USB Host port, USB Client port, and 8MByte NOR flash.

8.3 Data Center: Wireless Sensor Network

Through a study conducted by Lawrence Berkeley National Laboratory (LBNL) with SynapSense, it was demonstrated that a wireless sensor network could be installed rapidly and at low cost, to facilitate delivery of the projected savings. In the modern data centers, wireless sensor network (WSN) can act as low-cost candidate (10x) for monitoring task as it is non-intrusive, can provide wide coverage, and can be easily repurposed. Within a data-center, WSN system clusters a network of sensor-devices that enables real-time monitoring to observe and manage energy, thermal and performance constraints. WSN can also be useful as a debugging tool to monitor hotspots, benchmarking and system forensics including alerts and alarms. The non-invasive wireless sensors can measure and synthesize historical trends for air temperature, humidity, air particle count, current, power, node-utilization, workload performance, service-level-agreements (SLA) and other sense-data. They help to model organizational and technological choices to avoid competition for limited controls from multiple applications. Various attributes of the wireless sensor network (WSN) are related to auto-discovery, addressability, event-signaling, uniqueness, abstraction model, grouping, and ubiquity.

The wireless sensor technology in a data center comprises of sensor nodes, gateways, routers, server platforms, and software applications (similar to Data Center Manager). Once the WSN infrastructure is provisioned, it allows data center operator to perform the following functions:

1. Accurately measure real-time energy consumption and calculate Power Usage Effectiveness (PUE).

2. Interpret temperature, humidity, and sub-floor pressure differential data from various sensor nodes using live-imaging maps.

3. Accurately measure server specific performance characteristics and trends for developing statistical models that can forecast resource utilization and energy consumption.

4. Model relationship between server performance characteristics, energy con-

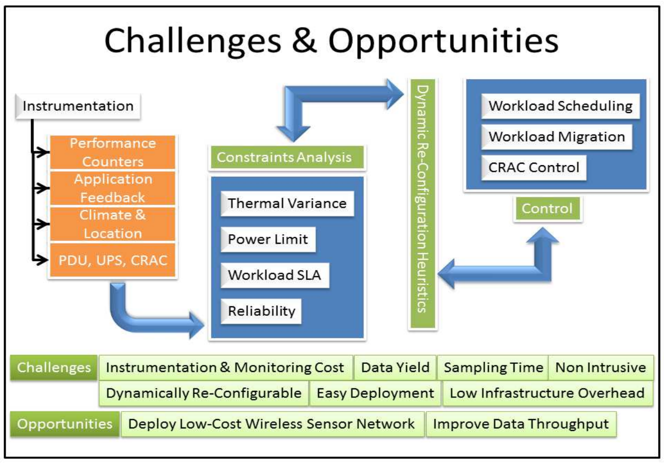


Figure 8.7: Real-Time data is analyzed for daily trends, load variations and dynamic resource demands. This along with sensor data from Each DC equipment (Server Nodes, Chillers, UPS Generators etc) is transmitted to a Data Center Manager that performs Statistical Processing and Constraint analysis related to Power, thermal, Workload & Reliability objectives.

sumption and environmental parameters (temperature, humidity, sub-floor pressure etc.)

5. Establish baseline energy consumption and identify improvement opportunities by efficient provisioning and loading of server resources.

6. Using monitoring infrastructure, develop automation strategy that performs adaptive workload-provisioning (loading, off-loading, migrating, consolidating etc.), airflow control, air-conditioning control.

7. Monitor environmental conditions to ensure compliance as per American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) and provide alerts if the ranges are exceeded.

8.4 Deployment & Monitoring Requirements

Unlike traditional wireless sensor networks, data-center WSN operation is constrained by performance issues related to facilities attributes and placement of sensors in dense wireless environment. In general, data-centers comprises of large number of wireless sensors that are densely deployed and data efficiency and delivery are primary concerns. These concerns can be summarize as:

(a) ***Sensor Density***: Unlike sparse distribution as in outdoor sensors, indoor placement of data-center (DC) sensors pose a challenging problem as they are densely deployed within one-hop communication length from neighboring sensors. This creates interference and collisions that can delay the data packet delivery.

(b) ***Data-Center Noise***: Metals are dominant composition in data centers along with server-nodes, RACKS, ducts, cables and power-distribution system. This creates disruptive conditions for reliable and latency-free data delivery.

(c) ***Data Delivery***: Sensor data from various nodes and RACK originate in bursts and amounts of several Kilo-Bits that needs to be delivered in a guaranteed time.

The unique nature of Data Canter Wireless Networks needs to fulfill certain requirements for effective data collection:

1. Wireless Network should be able to operate in an industrial environment that have a large amount of Radio Frequency (RF) noise that originates from Servers, Inverters, WiFi devices, building systems etc.

2. Time, frequency, and physical diversity should be incorporated to assure reliability, scalability, power source flexibility, and ease of use.

3. The sensor nodes should be ultra low-power wireless transceivers that transfer data to and from integrated sensors or controllers. These transceivers should be able to play a coordinated optimization role with neighboring nodes to eliminate operational interference.

4. Data latency should be minimized for optimal yield and reliability of sensor data.

5. Wireless network should be able to monitor packet throughput, collisions statistics, optimal routing, channel isolation and feedback the optimal connection to sensor nodes.

8.5 Sensor Data Collection

Data center sensor network is densely populated with aggressive latency constraints and sampling requirements for optimal cooling and workload distribution. Infrastructure is needed to allow re-configuration of sensors within a server node for optimal coverage, connectivity and bandwidth to improve the efficiency of functional control. Collection trees forms the basic building blocks of the sensor networks and the related applications. But collection trees working through traditional network protocols suffer from low delivery ratio. Couto et. al. [73] proposed ETX (expected transmission count) measure to find high-throughput paths on multi-hop wireless networks, which minimizes the expected number of packet transmissions required to successfully deliver a packet to its ultimate destination. Burri et. al. [74] proposed the protocol that coordinates MAC-layer, topology control, and routing to construct energy efficient communication subsystem. Madden et. al [75] proposed protocol that enable simple, declarative queries for efficient distribution and execution in low-power wireless sensor networks. Koala [76] proposed low duty cycles architecture that exploits the sensor-node idle periods to allow longer sleep times and proactively wakes them up upon bulk data download. Ganesan et. al [77] proposed joint optimization scheme for sensor placement and transmission structure for data gathering. Sensor nodes are placed in a field such that they aid in minimizing communication energy while reconstructing sensed data at a sink within specified distortion bounds. Jie et. al. [78] describe RACNet innovative rDCP data collection protocol for high throughput and high reliability data collection using similar concepts of channel diversity and bi-directional collection trees.

We describe a general scheme that uses a uses machine learning approach for channel-allocation using fitness function that incorporates attributes related to uniformity in allocations, number of hops, route balancing, router density, congestion aware re-allocation, data patterns, proximity patterns and sampling uniformity. A machine learning approach can facilitate sensor network provisioning and re-organization to reduce single-hop sensor node density through synthesizing interference free sub-nets for real-time data collection with latency constraints. This will require an approach to building optimal number of sub-nets for sensor data collection, and data-collection protocol for an individual sub-tree using time-slot allocation.

8.6 Data Center Sensor Network Synthesis: A Machine Learning Approach

Learning methods such as Genetic Algorithm (GA) that can facilitate allocation of each wireless sensor-node to one of the N sub-net containers by minimizing the overall interference between neighboring nodes while improving the packet delivery. Furthermore, parallel sub-nets can be synthesized that can operate independently without any interference from other trees. This allows thinning of the dense sensor network and reduce the average number of nodes that are within one-hop communication range. Genetic algorithm [4] is one such stochastic search technique that resembles the natural evolution. It support dynamic re-configurability and fulfills the need for necessary ingredients required for accurate data acquisition, better data-flow rates, distributed and cooperative management, multi-objective goals and long-term observability. They enhance real-time usage with parallel solutions that aid in searching multiple points simultaneously and, therefore, avoids being caught in a local optimum.

In a data-center with large number of sensors placed in close proximity, single-channel communication can drastically reduce the overall throughput due to collisions. In this unique environment with dense sensor-network, limited number of reusable wireless channels can cause co/adj-channel interference. This can degrade the Signal to Noise Ratio (SNR) of received packets and consequently the throughput of a data-center sensor network. For example, any degradation of data traffic can amount to over-cooling and creation of hot-spots which can ultimately result in high operational cost of cooling. Therefore in our GA approach, we evolve optimal number of sub-networks through multi-channel node allocation in a manner that minimizes the interference while meeting the data latency guarantees, thereby improving the data collection efficiency. We discuss an evolutionary approach to synthesize orthogonal sub-networks through channel diversity to enhance communication performance. We propose a multi-channel scheme for dense sensor network, which allocates channels to maximize the parallel transmissions among multiple sensor paths. In the proposed scheme we identify maximum number of non-interfering orthogonal channels (N) that can divide the sensor network (K) into sub-network represented by K_N.

$$K = (K_1, K_2 \cdot \cdot K_N); \quad K_i = (A_1^i, A_2^i \cdot \cdot A_n^i) \tag{8.3}$$

where, A_n^i represents the sensor node n that has been allocated to the i-th sub-network. Sub-networks can contain multiple trees, each leading to the gateway to achieve maximum throughput. Genetic Algorithms are employed to achieve optimal non-interfering trees. Rest of the section discuss the proposed steps in building the proposed solution

8.6.1 Sub-Network Synthesis

We describe GA based approach [4] to configure the randomly deployed sensors in a data center into an optimal number of non-interfering independent sub-networks with optimal routes and sensor membership. As discussed later in section 8.6.5, each sub-network parallelize the data collection from its member sensors and sends them to the target in a compressed manner via the most cost-effective route.

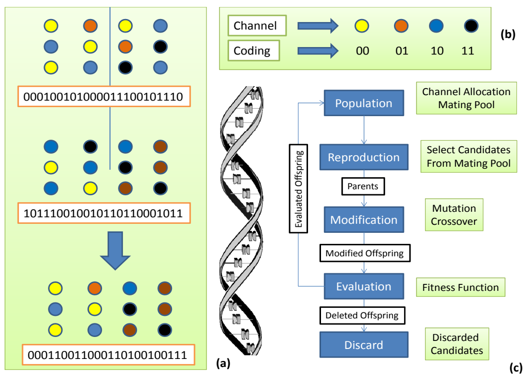


Figure 8.8: (a) Cross-Over process of two channel-allocation candidates producing an offspring candidate for new channel allocation (b) Channel Allocation 2-bit Coding Scheme (c) Genetic Algorithm for producing channel-allocation candidates.

As illustrated in Figure 8.8(c), *Genetic Algorithms* follow principle of natural selection, where each individual solution is represented as a binary string (chromosomes) and an associated fitness measure. Successive solutions are built as a part of the evolutionary process where one set of selected individual solutions gives rise to another set for the next generation. Individuals with a high fitness measure are more likely to be selected into the mating pool with an assumption that they will produce a fitter solution in the next generation (next run). Solutions with the weaker fitness measures are naturally discarded. We use roulette-wheel selection to simulate natural selection, where elimination of solutions with a higher functional fitness is, although possible, less likely. There also exists a small likelihood that some weaker solutions may survive the selection process as it may include some component (genes) that may prove useful following the crossover process. Mathematically, the likelihood of selecting a potential solution is given by

$$P_i = \frac{F_i}{\sum_{j=0}^{N} F_j} \tag{8.4}$$

where P_i represents the likelihood of a solution to be selected for mating pool, F_i represents the operating fitness of an individual solution, and N is the total number of solution elements in a population. GA has proved useful in solving complex problems with large search space that are less understood with little domain knowledge.

We propose the coding scheme where each individual sensor node channel-code is represented by a 2-bit binary number called 'gene' (Figure 8.8(b)). These two-bit genes define the sub-net to which the node belongs are called 'allele'. The chromosome of the GA represents the building blocks (allele) to a solution of the problem that is suitable for the genetic operators and the fitness function. As illustrated in Figure 8.8(a), two candidate solutions undergo a modification using cross-over function and results in a new candidate solution that undergoes an evaluation for candidacy in a new mating pool.

8.6.2 Characterization of Sensor Nodes in Data Center

First step in constructing sub-network in data center is to discover the proximity patterns between sensor nodes (Figure 8.9) originating from the WSN Gateway. While increasing the transmission power reduces the effect of non-interfering channels, reducing the transmission power increases the number of hops to reach the gateway. Either condition can degrade throughput. In or-

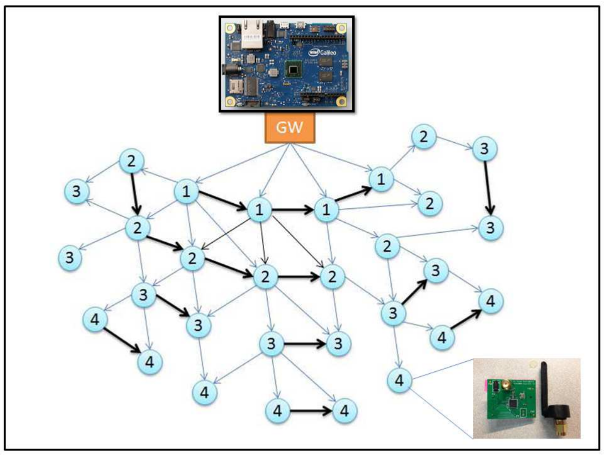


Figure 8.9: Illustration of sensor nodes characterization wrt. Gateway (GW). Thick lines represent peer-level communication, other lines represent communication between nodes in different levels of communication hierarchy. Each node-level is represented by number 1,2 3...

der to build sub-networks using GA approach, we need to identify the effect of neighboring nodes on each node. We start by broadcasting ADV message from the gateway. A generic ADV message comprises of its node-ID, parent-ID, children-list. Upon hearing the ADV messages, all the sensors within the one-hop distance of gateway tag themselves as L1 nodes, assign parent-ID and record Received Signal Strength Indicator (RSSI) into its local storage. This information, along with its node-ID is transmitted back to the parent (in this case Gateway) through contention-based approach. This Gateway-L1 process continues till all L1 nodes have responded to the Gateway. Upon completion, Gateway sends ADV_C message that identifies completion and selects child-ID that should send it's own ADV message. Upon receiving ADV message from L1 node, a new population of L2 nodes is generated that is within one-hop distance of L1 node selected by Gateway. Source-L1 nodes can also be overheard by subset of peer L1 nodes that are within one hop communication distance. L2 nodes update the corresponding information (RSSI, Self-ID, etc) to the source-L1 node. Peer L1 nodes also updates the information to Gateway. This process continues in breadth-first manner till all nodes are accounted for. At the end of process, we will have n-levels of nodes distribution. Fig 8.9 illustrates this process which builds 4-levels of distribution. Each node is characterized by its ability to communicate with its neighbors with a measured RSSI.Next step in this process is to extract N sub-nets, where each network is allocated by a non-conflicting channel. The average density of network is reduced by a factor of N. Each network can further be divided into multiple clusters that are separated by non-interfering characteristics.

8.6.3 Evolution of Sub-Networks through Machine Learning

In this step we run Genetic Algorithm (GA) [4] that allocates each sensor-node to one of the N sub-net containers by attempting to minimize the overall interference between neighboring nodes. Each subnet is a local cluster of nodes that are on the same channel and separated from other cluster by either channel diversity or by maximizing the distance that separates closest members of two clusters on same channel. We use configurable RSSI threshold to filter out the weak links, thereby removing them from evaluation. RSSI can be represented by the following equation:

$$RSSI = 10 \cdot log\frac{P_{RX}}{P_{ref}}; \quad P_{RX} \propto P_{TX} \cdot \left(\frac{\lambda}{4\pi d}\right)^2 \tag{8.5}$$

Where, λ is the wavelength of operation, P_{TX}, P_{RX} are the transmitted and received power respectively, d is the distance between sender and receiver sensor and P_{ref} is the reference power typically set to 1mW. Transmission power (P_{TX}) acts as a control parameter to isolate two interfering clusters.

We construct channel selection fitness function which is a weighted component that measures the quality or performance of a solution, in this case reduced interference between sensor-nodes. From the previous step (section 8.6.2), each node has set of neighbors defined by $A_i = (a_1^i, a_2^i \cdot \cdot a_n^i)$. Fitness function rewards the following conditions that helps to construct the initial network:

- Each node makes best effort to share channel with at least one L_{n-1} node. If such connection is not found, then the node can share channel with it's peer level node (L_n). It may be possible for the peer node to connect to L_{n-1}.
- Nodes in each hierarchy are rewarded if they can reduce the interference from L_n & L_{n-1} nodes. Although, eliminating it completely will violate the previous condition.
- If two nodes share single channel at level L_n, nodes are rewarded if they choose same parent at level L_{n-1}.
- An exclusive channel (ADV Channel) is reserved for control information. This information is related to broadcasting SINK messages as well as broadcast messages from newly added sensor. Sensors switch to this channel proactively when idle.

Once the channels are allocated, individual nodes undergo channel characteriza-

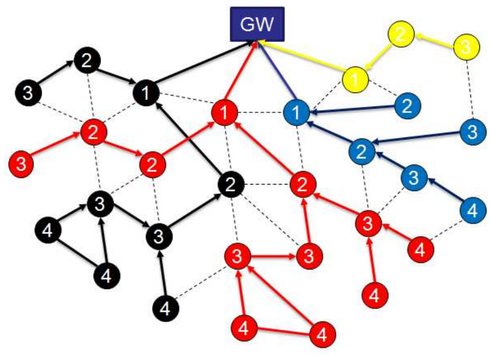


Figure 8.10: Illustration of channel distribution (represented by different colors) that builds multiple sub-nets)

tion to discover multiple subnets and the corresponding topology details. This process is similar to *First Step* as defined in section 8.6.2, except that now it is performed at the granularity of a channel (or sub-network). End result of this process is to create multiple subnet clusters for each non-interfering channel, where each cluster on similar channel is separated by more than one-hop distance.

Channel fitness function that guides the construction of sub-nets according to these conditions can be summarized as:

$$F_C = 1 - \frac{1}{3 \cdot K} \cdot \sum_{i=0}^{L} \sum_{j=0}^{N} (R_{ij} + \frac{I_{ij} - 1}{M_{ij}} + \frac{P_{ij} - 1}{\hat{I}_{ij}}) \tag{8.6}$$

where, $R_{ij} = 0$, if there exists a node k that shares channel with node j in the hierarchy i-1, M_{ij} represents the number of neighboring nodes to j at level i or below, I_{ij} represents number of shared channels with node j and level i or below, $K = (L \cdot N)$ represents total number of nodes and $\hat{I}_{ij}$ represents number of nodes sharing the channel with node j at level i and P_{ij} represents number of parents that catering to all nodes represented by $\hat{I}_{ij}$. If the parent doesn't exist for the node, it is still represented by NULL parent. In a hierarchical structure, there exists at least one candidate node in j-1 level that can act as parent. Figure 8.10 illustrates the distribution of available channels (4) among sensor nodes. The best effort methodology builds subnets to reduce the interference between neighboring nodes. Although collisions cannot be ruled out completely, they can be minimized. Nodes that are at single hop communication distance and share the same channel at the same level can identify the parent that can schedule the data delivery by using pull protocol or time-scheduling. Although, using this technique the channel distribution may be optimized for minimizing interference, it may not be optimized for load balancing. The next section enhances the fitness measure to allow balance loading through the network to minimize data-transmission latencies.

8.6.4 Building Route Trees: Synthesizing Balanced Routes

As discussed earlier, data-center sensor-nodes generate several kilo-bits of burst data in one sampling period. This data along with rest of the sensor data has to traverse several hops before reaching the gateway. Once the channel allocation solution is applied (Figure 8.10) amongst sensor nodes, the number of possible routes are reduced as well. A bad solution would result in elimination of efficient routes that could have been possible with an alternate channel distribution. We define route fitness (F_R) that rewards the channel allocation resulting in optimal delay paths according to the target requirements. Unexpected delays on routes connecting sense-points and gateway can reduce the effectiveness of the received sensor data. Potential routes are evaluated using route fitness F_R (Equation 8.7) that depends upon (a) congestion − missed or delayed packets and (b) average latency − round trip time (RTT) of sensor data transaction.

$$F_R = 1 - \frac{1}{2}\left[\frac{1}{N}\sum_{n}^{N}\min\left(1, \frac{|\tau_n - \bar{\tau}_n|}{\bar{\tau}_n}\right) + \frac{A}{R}\sum_{r}^{R}\frac{T_r}{\hat{T}_r}\right] \tag{8.7}$$

where τ_n and $\bar{\tau}_n$ are the measured delay and expected delay between end-to-end traffic serviced by connection n respectively, T_r is the number of packets either missed or delayed by intermediate nodes (proxy routers) r, $\hat{T}_r$ is the total packets serviced by intermediate nodes (proxy routers) r and A represents the amplification factor. A connection n is represented by all the routes terminating at L1 level nodes. Router node r is an intermediate node that fulfills the function of sensor as well as router (for at least one node).

Route fitness F_R factor rewards the reconstruction of routes that meets delay requirements to rebalance the overloaded nodes. An interesting property of GA algorithm is that every node seeks to attain the shortest path that is optimized for low interference and low latency to the base-station. This property comes from the fact that the algorithm first identifies interference patterns for all nodes and builds a L-level hierarchy that represents nodes hop-distance from the base-station (or Gateway). Algorithm rewards a hierarchy where each node connects to parents that uses lesser hops (one-hop less) to the base-station. First, each node builds profile structure that contains information regarding all other nodes that can overhear it and its position in the L-level hierarchy. This is a centralized operation that is performed once during the deployment and very infrequently during maintainence cycles. Deployment phase extracts parallel sub-nets that can operate independently without any interference from other trees. This allows thinning of the dense sensor network and reduce the average number of nodes that are within one-hop communication range. Furthermore, deployment phase also involves calibration process where sensor-nodes are activated and characterized for latency and other effects by running GA that combines the effect of channel allocation (equation 8.6) and route balancing (equation 8.7).

8.6.5 Dynamic Evolution of Cluster Networks

Once the wireless sensor networks are deployed for real-time monitoring it is not practical to make changes to the WSN infrastructure too often. Dynamic conditions in the data-center such as hardware provisioning, data traffic variations and noise conditions require re-configuration of WSN over time. To avoid disruption, the process of re-configuration requires an adaptation mechanism that evolves over time with least amount of intrusion to the existing configuration. Evolutionary mechanisms utilize passive measurements of the characteristic behavior of sensor-nodes over time. These measurements are used to evaluate potential modifications in the sensor network which can be summarized as:
(a) **Proximity Patterns**: As defined in section 8.6.2, these patterns define one-hop neighbors of each sensor. As sensors are added, replaced or removed, new patterns emerge and are recorded using control channel.
(b) **Data Patterns**: Sensor data from multiple sensor nodes demonstrate certain patterns that are typical of a local context (e.g. cooling devices). Sensors sharing similar context can have spatial data redundancy which can be exploited

using common models. This can reduce the amount of data flows through the network. Nodes that demonstrate steady-state sense-data can exploit temporal redundancies and transmit fraction of data that changed during the sampling interval.

(c) **Router Density**: Reassigning sensor node to an alternate channel will trigger migration of all down-stream nodes to the same channel. This can increase the probability of uncovering co-channel interference between newly-configured nodes and disrupt sense-data delivery.

Fitness function (F_D) that incorporates all the dynamic conditions listed can be represented as:

$$F_P = 1 - \frac{1}{N} \sum_{i=0}^{N} \delta_i \left[\chi \eta_i + \textsc{min} \left(1, \frac{\textsc{max}(0, (\bar{\lambda}_i - \lambda_i))}{\lambda_i} \right) \right] \tag{8.8}$$

where χ is the Amplification factor, η_i counts downstream nodes relative to node i, $\delta_i = 1$ if node i changed it's parent, λ_i and $\bar{\lambda}_i$ are the average data rate at node i's parent before and after the node changed its parent. A change in the parent can result in variation in the data compression ratio and alter the data rates. Optimal route would exploit spatial correlation between sensors that share similar behaviors and trends.

8.7 Data Collection Protocol

Data collection is initiated by the gateway by traversing the DATA_SEND message to all the nodes sequentially on the parallel tree (trees on separate channels). Data is collected in depth-first manner and cached into the parent before

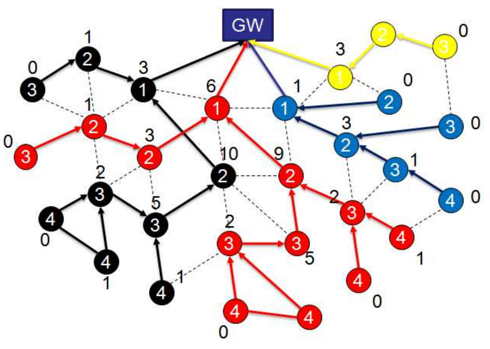


Figure 8.11: Illustration of parallel data flow through multiple sub-networks. All nodes in different colors represent member of corresponding sub-network operating on different channels and can operate in parallel)

transmitting up-stream. Periodically gateway traverses ADV_CALIB message through each single tree and ADV_SENSE through the rest of the trees. This allows a candidate node to broadcast ADV message (CALIB) so that the rest of the nodes can listen (SENSE) to that message by switching to the ADV channel, thereby calibrating there measurements with respect to each other. These measurements are delivered up-stream to the gateway as training data for executing GA function. This process is distributed over time to avoid long periods of inactive sensor measurements. This protocol maximizes the data collection by parallelizing the data flow through multiple sub-nets.

8.7.1 Total Fitness (TF)

Total Fitness (TF) is the weighted sum of individual fitness and represents optimal sub-tree construction. Optimized tree would support channel allocation that is free of interference and route congestion. Furthermore, optimal tree would aid spatial compression, scalability and low latency routes. Total Fitness (TF) function can be represented as:

$$TF = \alpha_1 F_C + \alpha_2 F_R + \alpha_3 F_P + \alpha_4 F_D; \tag{8.9}$$

where α_n is the relative weight of the fitness components.

8.7.2 DATA_SEND Protocol

DATA_SEND Protocol minimizes the amount of messages that need to traverse through the tree. We employ WAIT_SLOT attributes to all member nodes throughout the sub-trees that parallelize the data collection though the single tree. Each WAIT_SLOT represents earliest time a node can transmit a collection of all down-stream sensor node packets to its parent

Assumptions:

- Each node is aware of its weight (number of all the nodes downstream) and the weight of its children.
- Data collection can proceed in parallel on sub-nets that communicate on different channels.

Protocol

- Parent nodes enumerates all the children nodes according to number of nodes downstream.
- Parent traverses through enumerated children to send WAIT_SLOT threshold and sequence number through DATA_SEND message. This

threshold identifies the earliest time-slot the children nodes can communicate with the parent (with measurement data).

- Parent continues this process till all children on the sub-tree are enumerated.
- This process continues till all leaf-nodes are enumerated.

Figure 8.11 illustrate the process of isolating data collection among multiple tree. Data parallelism is maximized by employing channel-separation and time-slot reservation. Nodes on the same channel communicate with the parent based on time-slot reservation that is adjusted and communicated up-stream according to data-collection delay statistics. Initially, static time-slots are reserved according to maximum packet size to avoid hidden-Node collision. For example, in case of BLACK channel at level-1 the weight is set as 3. This illustrates the waiting channel at level-2 (Right) need to wait at least 3 waiting slots before transmitting its sense-data. Data collection effectiveness is quantified by the fitness that enables the sense-data collection on all sub-networks in shortest possible time. Fitness function is summarized as:

$$F_D = 1 - \frac{1}{N}\sum_{n=0}^{N}\frac{|\bar{t} - t_n|}{\bar{t}}; \quad \bar{t} = \frac{1}{N}\sum_{n=0}^{N} t_n \tag{8.10}$$

where, N is the number of sub-networks, t_n is the average sampling duration of subtree n to collect sense-data of all downstream nodes. Non uniform sampling duration results in delayed collection of sense data at the central collection server.

8.8 Sensor Network Performance

In this architecture, GA algorithm executes and trains on the Gateway Server that comprises of 32 (or more) cores and 64 (or more) Gigabyte of memory. Gateway collects the sensor data from corresponding nodes and forwards it to the SCADA controller. Additionally, gateway is capable of monitoring and analyzing data collection trends from individual nodes as well as established paths. Some of these trends correspond to inter-sensor interference patterns, Node specific packet delivery behavior (sensor data rate, burst patterns) and Route specific packet behavior (packets lost, delayed). Individual sensors do not participate directly in the training process, but configures itself according to the new responsibilities assigned to it as a part of continuously evolving solution. Gateway Server monitors and analyzes various data points that are used to evaluate the fitness function as defined by equation 8.10. GA algorithm periodically re-evaluates the existing solution based on sensor attributes and evolving training set. This results in new solution that replaces the existing solution. thereby resulting in incremental re-configuration (Channel Re-Allocation) of sensor network.

Using the feedback collection scheme as described in section 8.6.2, we discover that large number of nodes are within one-hop distance of each sensor. Therefore we employ genetic algorithm (GA) (Section 8.6.3) to allocate distinct non-adjacent channel to sensor nodes to increase the average single hop distance. After the thinning process, nodes that share same channel and are within one hop distance are allocated time-slots for interference free data collection (Section 8.7.2). Figure 8.12 illustrates the improvements in data collection latency due to GA assisted channel diversity. Sensor Network with 4-Channel diversity

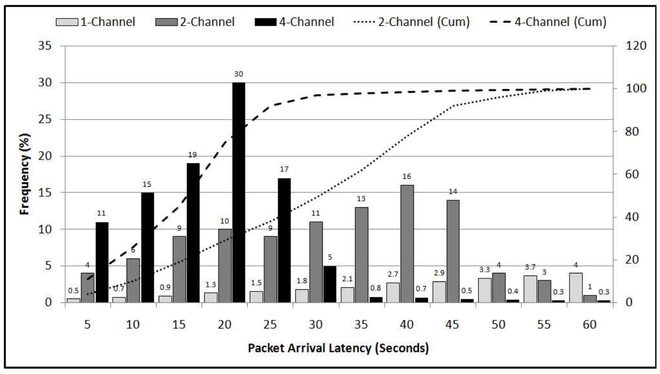


Figure 8.12: Packet collection distribution using GA algorithm for channel diversity. For 4-channel diversity, 92% of the packets are received with 25 seconds deadline.

achieves 92% packet receive ratio (PRR) in 25 seconds, compared to 2-channel case which achieves the same PRR in 45 seconds. Due to increased sensor-sensor distance, interference between neighboring sensors is reduced that results in lesser collisions and improved efficiency.

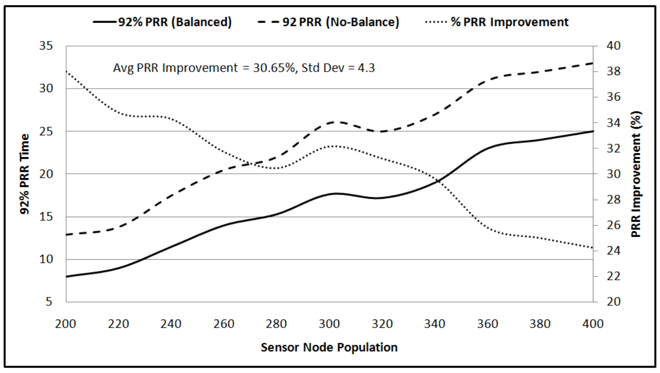


Figure 8.13: 92% Packet Receive Ratio (PRR) latency (a) Using GA assisted balanced tree fitness criteria ($\alpha_4 = 0.2$) (b) not using balanced tree criteria ($\alpha_4 = 0$) (Equation 8.10)

Although channel diversity evolves by observing interference patterns and relative signal strength (RSSI), it alone cannot solve the problems related to congestion, packet delays and route-bottlenecks resulting from traffic variances due to heterogeneous nature of sensor activity. Therefore we augment the evolution phenomenon using a fitness function (F_R) that rewards the channel diversity solution upon creating balanced routes as explained in section 8.6.4. Figure 8.13 illustrates the benefits of using sub-network balancing criteria (F_D) within the GA fitness function as described in equation 8.10. We achieve an average of 30.65% performance boost with a standard deviation of 4.3 over a scalable range of sensor population (200-400). In addition to optimal channel selection, the F_D element of genetic algorithm (GA) amplifies the evolutionary attributes of route reconfiguration that results in producing large number of non-interfering sub-networks with almost identical latency characteristics.

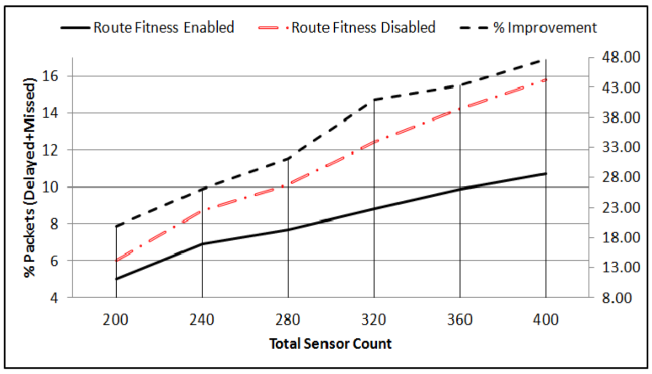


Figure 8.14: % improvement in lost and delayed packets if balanced route fitness (F_R) is enabled (Compared to disabled). F_R influences the channel allocation for optimal delay

Figure 8.14 illustrates improvements of 20-48 % in the amount of lost and delayed packets for different sensor densities. This improvement can be attributed to reduced contention on the delivery routes resulting from channel allocation that rewards contention-free routes generation.

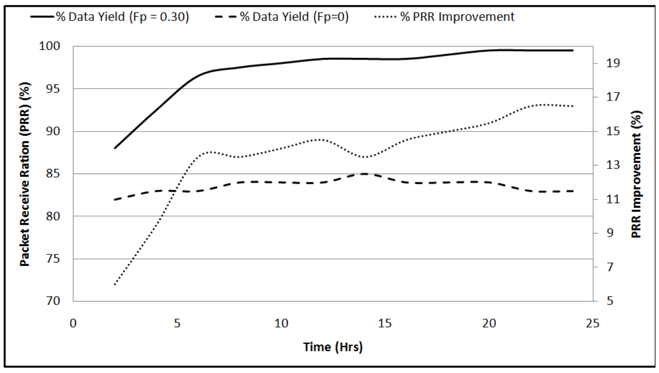


Figure 8.15: Illustration of total data yield as packet receive ratio over 24 hour collection period with 400 sensor nodes.

Figure 8.15 illustrates sense-data yield distribution over a period of 24 hours. Sense-data packets are collected with variable degrees of spatio-temporal correlation. This results in variations of the packet size due to run-time compression. Y-Axis represents packet receive ratio (PRR) based on sampling period of 25 seconds. With route balancing and congestion control (F_R) in place (Equation 8.7), we observe target PRR compliance (92% PRR over sampling period of 25 seconds) with an average of 97.125%. In case we ignore this fitness component($\lambda_2 = 0$), the PRR compliance drops down to an average of 83.58%. Route fitness factor optimizes the delay paths by reconsting routes to rebalance the overloaded nodes to meet delay requirements. During the first 6 hours, GA assisted clusters evolve and boost the yield from 92.3% to 98.7% due to dynamic re-balancing.

8.9 Conclusion

Data Centers face considerable challenges in seamless integration of telemetry and control functions. Real time monitoring and control of data center resources (Cooling, Power and workload resources) demands high fidelity monitoring of workload and environmental trends. We propose the use of genetic algorithm (GA) based approach to configure sensor network that enables time-bound sense-data collection in a data center (DC). Wireless sensor networks in data center are non-intrusive and cheaper alternative to traditional wired networks. Evolutionary Learning Techniques (like GA approach) aids in constructing parallel sub-networks by intelligent allocation of non-interfering channels to member nodes. To achieve high fitness factor, channel allocation can be geared towards balancing each subtree to deliver similar data collection timings with least amount of contention. Machine learning assisted channel diversity can deliver exponential performance boost to data collection through eliminating route congestion, parallelizing data collection and minimizing channel contention. In addition to channel distribution, we can improve the performance by introducing tree balancing and congestion detection.

Chapter 9 – Future Work: Dense WSN for Asset Tracking

9.1 Introduction

In this theses we proposed a machine learning based algorithm to provision and organize sensor network for the reasons of energy efficiency, secure communication, intrusion detection, improved throughput using channel diversity and collection trees. We applied the the combination of these algorithms for dense sensor network in a data center. Efficient data movement with densely populated sensors in a data center is essential for load consolidation, real-time troubleshooting and SLA guarantees. This helps to resolve issues are related to over-utilization of resources, application failures, data security, power usage effectiveness and infrastructure costs. We introduced channel-diversity and time-slot allocation to our basic provisioning to construct parallel data movement through multiple routes in a WSN. In this chapter we define the problems related to development of Wireless Sensor Network (WSN) for shipment tracking where the density of *asset-tracking sensors* is extremely high.

Furthermore, the battery of these disposable sensors have to last anywhere from 60-90 days. Additionally, these low-power sensors communicate through a battery powered (1800-3200 mAh) **Gateway** via 3G to the cloud with a regular periodicity. This sensor network is developed with the following constraints:

1. Sensors Nodes are disposable, low cost and low-power

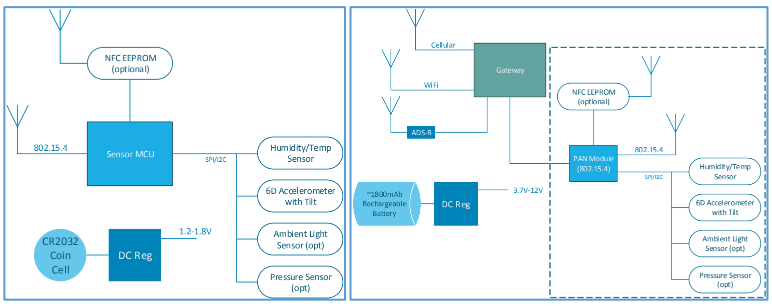


Figure 9.1: Sensor Node and Gateway for shipment tracking

2. Each Sensor Node communicates with the ”Gateway with a regular periodicity (for example, 15 Minutes)

3. Gateway is battery operated (1800-3200 mAh), cost-effective and reusable device.

4. Each Gateway collects the data and transmits to the cloud with a regular periodicity (for example, 15 minutes)

5. Gateway-Sensor tuple has to last for the life of the shipment (60-90 days)

6. Preferable communication protocol is 802.15.4 (Gateway-Sensor Nodes) and 3G (Gateway-Cloud).

Low-power and cost efficient solution mandates the development of solutions with very low power ’Idle-State’, synchronized data movement, data compression to avoid any redundant transmissions and wireless-based wake-up solution to allow ’contention-based’ emergency data movements. In this chapter we describe the high-level requirements, problem statements and possible solutions related to Sensor Node and Gateway Development. We also propose the use of GA techniques developed in this thesis for optimal data movement between intermediate nodes and Gateway. The future work will involve augmenting the "'Data Center' (Chapter 8) based provisioning algorithms to develop a solution for an extremely dense and power constrained shipment-tracking scenario.

9.2 Assumptions

Each shipment comprises upto 30 *Pallets* and each pallet contains upto 100 *Parcels*. Therefore total shipment contains 3,000 parcels, where each parcel needs to be tracked via *Gateway* throughout the life of the shipment (60-90 days) at a given interval (for example, 15 minutes). Additionally, any threshold violation (Temperature, Drop etc.) needs to be reported within 30-60 seconds to the tracking station (behind the cloud). Each pallet is fitted with a battery-operated Gateway and each Parcel is fitted with a sensor node as illustrated in Figure 9.1. Both the Gateway and Smart Sensors are instrumented with a sensor hub. The sensing requirements are not uniform across platforms. Various sensors that are connected to the sensor hub (of a sensor node) are:

1. Temperature
2. Humidity
3. Pressure
4. Intrusion Detection
5. Fall Detection
6. Collision Detection

7. Abnormal Tilting, Sliding and Wobbling detection

8. Stability Loss detection

Temperature sensor payload is 8bit, assuming temperature range from -10 to 65 with 0.5C accuracy. This leads to 125 signed increments which should be covered by one byte which can cover 256 values. One of the challenges is the develop or select the sensor components that can support (a) Wake on Threshold Violation (b) Support Hysteresis and (c) Low cost. Sometimes we can fuse multiple sensors (like Temperature, Humidity and Pressure as in BME280 [79]) into a single sensor to save cost and power.

9.2.1 Pallet

Number of boxes in each pallet in each shipment is predefined before the shipment. Maximum distance between sensor node and the pallet Gateway can be upto 1.5 meters. Each sensor in pallet reports its time-stamped status to the Gateway at a regular interval defined by the shipper. Each shipment operates on two pre-provisioned channels (Heartbeat Channel and Emergency Channel). Data transmitted by the sensors on these channels can be categorized into two categories:

1. **Heartbeat Data** (Operates on Heartbeat Channel): Minimum amount of sensor measurements reported to the Gateway if the sensor measurements are within the hysteresis limits of the pre-configured thresholds. This data moves through the sensor tree using a pre-configured route in a time-synchronized channel.

2. **Emergency Data** (Operates on Emergency Channel): Critical measurements that violate the critical-threshold limits are reported to the Gateway using this emergency channel. This channel operates in a contention mode by waking up the neighboring nodes by transmitting at higher at normal signal strength. This channel support time-critical notifications that cannot wait to be transmitted through heartbeat channel because of the critical nature of the message (Intrusion, Collision, Cooling Failure, Fall etc.)

9.2.2 Shipment

Shipment can be considered as a collection of pallets. Since each pallet comprises of a Gateway and therefore each shipment may contain upto 30 Gateways. For better efficiency and cost-reduction we may create an intermediate layer called 'Aggregator' that passes the sensor data from each pallet to a centralized 'Gateway' capable of cellular data transmission. Pallets relative position during the transportation process is not fixed. The worst case is all 30 pallets line up on one straight line with the mobile GW is on one end of the line. In this case, Gateway and Aggregator may be separated by a distance of 45 meters. It is also possible for the pallets not to be co-located and the whole network distance can grow as large as 90 meters.

9.3 Problem Statement

This section describe the problems and design issues related to operating a sensor network in a highly dense and mobile sensor network.

9.3.1 Provisioning

Through experimentation and verification of use cases, we believe that an efficient *mechanism to provision the sensor* is to implement shallow and wide tree, and the method of communication needs to implement time-multiplexing to avoid collisions at low energy-consumption. The time synchronized network can be developed using the GA based provisioning technique as described in Section 8.6.5, except that we use a single channel instead of multiple channels as in data center. This is a time synchronized sensor network that uses IEEE 802.15.4 based radio. Network time sync is achieved through beacon frame. Each box in one pallet needs to be assign a number which will be used as its slot index in the TDM frame.

The Shipment ID (SID), Pallet ID (PID) and Network ID (NID) shall be provisioned in each end nodes before its shipment. When an end node is powered on, it will be kept on listening mode until it hears the beacon message from its Gateway (Or Aggregator) by matching the SID and PID. Then it will sync its clk with beacon message and respond at the allocated time slot corresponding to its NID. When an Gateway (or Aggregator) is powered on, it will start to broadcast the beacon message (Node Discovery) and then listen to response from the nodes in the pre-configured number of slots till it has heard response from all nodes. The response from Nodes comprises of the necessary attributes (Neighbor RSSI, TX Strength etc.) that are then used to construct the fitness function to optimize the tree depth and connecting nodes of sub-networks using GA as described in Chapter 8.

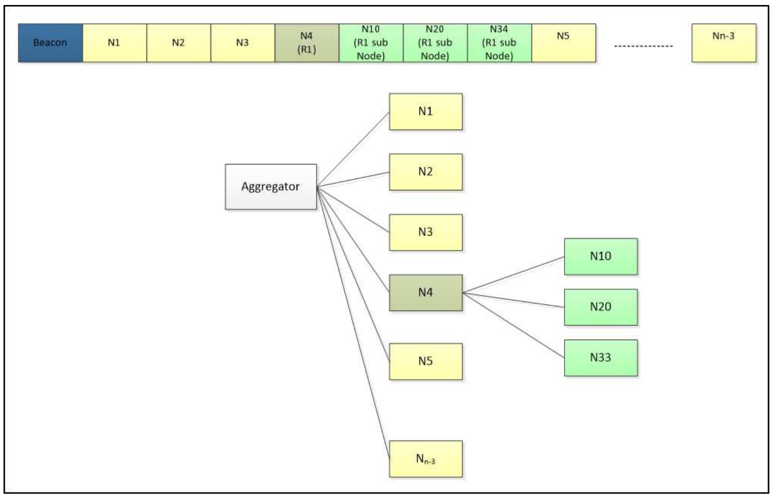


Figure 9.2: Illustration of timeslot allocation during node discovery process.

A pallet Gateway (Or Aggregator) (X) contains a list of all the nodes that are part of the pallet. It sends a broadcast packet to locate all of its nodes. Every nodes that receives the packet will respond with necessary attributes at their time slot. X builds a list with these nodes. At the end of this first round, nodes that responded are the ones that can communicate to its Gateway. We call these nodes $(N1, N2, N5 \cdots)$ as L[1] nodes (or Level 1 Nodes). For each L[1] node, Gateway X sends a command to start listening for sub-nodes, so the node $(N4)$ at L[2], sends a broadcast command at the beginning of the round to find any nodes that respond to it, building also a list L[i]. When the end of the round is reached, then the current node sends to X its list of sub-nodes L[i].

When all nodes with non-zero values are finished, the pallet Gateway builds and optimize its tree. The maximum number of nodes on a sub-tree is defined by a configuration variable. If this number is reached, the list must be trimmed down. the tree depth should be the weighted component of the fitness function. After all the subtrees have been calculated, the time-multiplexed slot allocation must be re-created, since every node and its children must have consecutive time slots, so the parent node has to be on for the shortest possible amount of time.

9.3.2 Inter-Shipment Collisions

At a warehouse or waypoint there may exist multiple shipments, and collisions should be minimized. The Management Information System on the cloud, based on the location of each shipment on the floor (assuming that all parts of a shipment should be around each other), will assign a frequency channel for each shipment, and in cases where the number of shipments exceeds the number of channels, these will be allocated so two shipments with the same channel that are farthest from each other to minimize collisions. Although a pre-configured channel is assigned to each shipment, in case of interference the active nodes move to an available channel that is non-adjacent and non-interfering. This process of moving to a new channel is managed by the Gateway based on the "‘Lost Data Packets"’, "‘Received Signal Strength"’ coupled with its own selection criteria (Noise measurements of available channels).

9.3.3 Time Budgeting

If we assume the Gateway reporting interval of 15 minutes, that allows us 900 seconds of total time allocated for data collection from each sensor in a shipment. Since we have maximum of 30 pallets in a shipment, that allows a 30 seconds time-multiplexed window based for each pallet ID. For each pallet, its window is subdivided in two regions, a short period where an Gateway (Or Aggregator) sends messages and every node listens, followed by a much larger region where every node has been assigned a time-multiplexed slot based on its node ID and only a node-pair is powered up to communicate with its Gateway (Or Aggregator).

Assuming that beaconing is implemented for proper synchronization, and knowing that only two nodes should be communicating at any time, so there is no contention, to transmit a data packet with a 12-byte payload, with a total packet size of 43 bytes, accounting overhead, at 250kbps, it takes 1.376mSecs. The ACK frame, with 11 bytes, takes 0.352mSecs, and the turn-around from transmit to receive requires 0.192mSecs, for a total of 11.92mSecs.

9.3.4 Critical Events

If a node detects a critical event, it activates the emergency channel by waking up its companion node (if possible selectively) and transmits the event message to that node. This process continues till the message reaches the gateway. In a simplest case, the wake-up signal is sent at higher than normal signal strength. It is possible that other nodes may also wake up while waking up the companion node. But the message is received and forwarded by the node that is the next node in the sub-network to which the node belongs to. It is possible that multiple nodes may experience the critical event. In this case we may observe many wake-messages waking up every sensor node. We may have to resolve this rare event condition by using an emergency channel that gets activated by the wake-up event and acts as a "'Contention Channel"'. Some of the future work in this area may be needed in the development of low-power *wakeup-on-demand* mechanism that can selectively wake up a sensor node. Such mechanism would improve the battery life of the sensor by operating it in a low-power "'Idle-State"' for majority of the time. The sensor only wakes up when it is transmitting, interrupted to receive or interrupted by the sensor (Threshold Violation). When a threshold is met on a tag sensor, the node must wake-up and transmit this event within a minute. For a large number of packages an acceptable approach would be to generate a wake-on packet that could be sent to nodes in the vicinity: When a sensor generates an event, an interrupt wakes the processor, a packet OOK (On-Off-Key encoded) consisting of a preamble of 1s and 0s, followed by the ID of the sub-route of the node, is created and transmitted by the 15.4 radio.

As illustrated in Figure 9.3, a passive circuit can implemented on every node, with an antenna, a filtering circuit followed by amplification, This signal is passed to a comparator on an ADC line of the micro-controller. When a transition from 1 to 0 is recognized by the comparator, an interrupt wakes the MCU, and it starts counting the number of 1s and 0s. If this number is higher than a pre-established number, it means that a wake-on packet preamble has been identified, it receives the following ID and compares it to the ID stored on the current node. If there is a match, then the MCU wakes-up the 15.4 radio

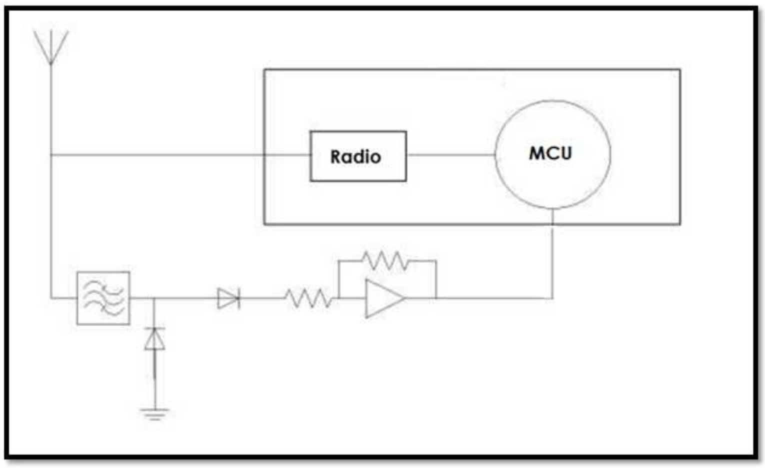


Figure 9.3: Wakeup On Demand signal is passed to the comparator of ADC to enable selective wake-up.

to receive the packet that will be broadcasted by the node that generated the event, if not, the MCU goes back to sleep. Experimental results demonstrate that wake-up circuit can save upto 66% of the total energy that is wasted during idle periods. The energy consumed during wakeup may offset the savings to some extent, if the wakeups are too frequent.

9.4 Summary

As a part of future work we would like to augment the algorithms developed in Chapter 8 for dense sensor network provisioning for asset tracking where performance and throughput may not be the quality requirement. Instead battery utilization efficiency in a sensor network with periodic transmission of sensor data over a long period (2-3 months) is important. This leads us to research on energy efficient sensors, data protocols and circuit enhancements like wireless wakeup and energy harvesting. Furthermore, the solutions have to be cost-effective for practical applications where the sensors may be disposable. The cost of the sensor may prevent its use in applications where the cost of the asset itself is low. While majority of the algorithms developed in this theses can be applied to dense asset tracking (or shipment tracking), certain enhancements in the *wireless wakeup and energy harvesting* can help improve the life of the sensors and improve it practical applicability to many more applications.

Chapter 10 – Conclusions

10.1 Summary

In this thesis we introduced evolutionary mechanism supported by Genetic Algorithm (GA) that supports self-organization of sensor networks in a battery constrained as well as bandwidth constrained environment. We develop algorithms for (a) Efficient provisioning (b) Secure communication (c) Power Optimization (d) Compute Resource allocation and (e) Intrusion detection. These algorithms can be applied to a sensor network according to the requirements of the application.

In our base approach [33], randomly placed sensors are assigned functions (sensing node, cluster-head, router, or inactive-node etc) based upon the results of GA. The GA approach optimizes the network to maximize energy usage along with battery conservation with route optimization. It can be shown that the periodic run of a genetic algorithm will help conserve the overall energy of the system with maximum operability. Furthermore, we apply the concepts to a *Datacenter Network* scenario, where we enhanced the concept of self-organization of sensor networks using channel diversity and data collection trees into the fitness criteria.

For battery-constrained environment the GA approach is extended to optimize the network to maximize energy usage along with battery conservation with route optimization [80]. It can be shown that the periodic run of a genetic algorithm will help conserve the overall energy of the system with maximum operability. Individual components tends toward maximizing their fitness with the passing generations in a uniform manner. The goal of maximizing the system fitness along with individual component fitness can be achieved with a considerably reduced complexity. The algorithm also prevents the over-optimization of an individual fitness component at the cost of other components.

Further, we apply the sensor network to a manageability and autonomic backbone for observability [81] and control of critical aspects of the platform ranging from resource estimation [82], performance re-balancing [83], workload scheduling [84], power/thermal control [85], fault detection [62], resource chargeback, and failure prediction. We augment the genetic algorithm (GA) with fitness criteria that allows re-configuration and automates coverage, connectivity and bandwidth to improve the efficiency of functional control. We compared the performance of the proposed approach against the static autonomics system where routes and bandwidth are constrained by the type of interconnect. The improvement in the delay characteristics ranges from 15% to 20%; the improvement in estimation accuracy due to lower delay between end-points is $8-10\%$. High accuracy and low delays enable scalable communication infrastructure with predictable bandwidth and latency to provide demand based plug-and-play interconnection.

Since security is an important concern in the wireless sensors, we augment the basic approach with a novel scheme of adapting the security attributes proportional [41] to the perceived threat and in a manner that promotes efficient battery usage and minimizes the effects of aberrant nodes. Battery limitations and computational overhead prevent us from maintaining the same threat levels by employing encryption and authentication mechanisms on all nodes. Ideal enabling reduces the computational overhead while maintaining the adequate security levels by identifying the strategic nodes. Strategic nodes are optimally enabled for security by evaluating the battery status, network traffic, malformed or retries on a specific route and number of nodes in a single route handling authentication. Ingredients of security architecture create a trust relationship between various node-types for the reasons related command/message execution, data forwarding, etc. This scheme further allocates the monitoring function [86] to the sensor nodes after evaluating its fitness based on integrity, residual battery power, and coverage. The monitoring scheme is decentralized since the local monitoring nodes are optimally distributed in the network and feed back the profile drift data to the sink.

Finally in a bandwidth-constrained environment, such as enterprise datacenters, we propose a GA approach that instruments low-latency high-yield data transmission through a Wireless Sensor Network. This approach augments the battery-constrained approach [87] by introducing the channel-diversity and time-slot allocation to construct parallel data movement through multiple routes in a WSN that is bandwidth constrained. Efficient data movement in an environment with densely populated sensors is important to fulfill technology and business challenges such as virtualization, load consolidation, real-time troubleshooting and SLA guarantees require a robust and adaptive server management plan for enterprise. This helps to resolve issues are related to over-utilization of resources, application failures, data security, power usage effectiveness and infrastructure costs. To achieve data-delivery effectiveness, we divide the sensor-network into multiple sub-nets, each of which operates on one of the non-interfering channels.

The selection of nodes that attaches to one of the many sub-trees is constructed using GA function that evaluates various aspects of the node membership wrt. overloading, route-balancing etc. Each sub-tree introduces distributed time-slot allocation according to the number of down-stream nodes for each parent node. This further reduces the contention between nodes to achieve high packet-reception-ratio (PRR). We achieve an average of 30.65% performance boost with a standard deviation of 4.3 over a scalable range of sensor population (200-400). We use the GA assisted balanced routes that produce large number of non-interfering sub-networks with almost identical latency characteristics. Furthermore, we improve the number of lost or delayed packets by 20-48% for different sensor densities. This improvement is attributed to reduced contention on the delivery routes resulting from channel allocation that rewards contention-free routes generation.

Bibliography

[1] Texas Instruments. A true system-on-chip solution for 2.4-ghz ieee 802.15. 4 and zigbee applications. *CC2530 datasheet, February*, 2011.

[2] Panos M Pardalos, Henry Wolkowicz, et al. *Quadratic Assignment and Related Problems: DIMACS Workshop, May 20-21, 1993*, volume 16. American Mathematical Soc., 1994.

[3] Drew Fudenberg and Jean Tirole. Game theory mit press. *Cambridge, MA*, page 86, 1991.

[4] David E Goldberg et al. *Genetic algorithms in search optimization and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989.

[5] Sandra M Hedetniemi, Stephen T Hedetniemi, and Arthur L Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.

[6] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 22–31. ACM, 2002.

[7] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185. ACM, 1999.

[8] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67. ACM, 2000.

[9] Maurice Chu, Horst Haussecker, and Feng Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 16(3):293–313, 2002.

[10] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod Record*, 31(3):9–18, 2002.

[11] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on*, pages 10–pp. IEEE, 2000.

[12] Stephanie Lindsey and Cauligi S Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *Aerospace conference proceedings, 2002. IEEE*, volume 3, pages 3–1125. IEEE, 2002.

[13] AMADP Agrawal. Teen: a protocol for enhanced efficiency in wireless sensor networks. *IEEE, San Francisco*, 2001.

[14] IEEE 802 LAN/MAN Standards Committee et al. Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Standard*, 802(11), 1999.

[15] Mohamed Younis, Moustafa Youssef, and Khaled Arisha. Energy-aware routing in cluster-based sensor networks. In *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*, pages 129–136. IEEE, 2002.

[16] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

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

[18] Rex Min, Manish Bhardwaj, Seong-Hwan Cho, Eugene Shih, Amit Sinha, Alice Wang, and Anantha Chandrakasan. Low-power wireless sensor networks. In *VLSI Design, 2001. Fourteenth International Conference on*, pages 205–210. IEEE, 2001.

[19] Jan M Rabaey, M Josie Ammer, Julio L da Silva, Danny Patel, and Shad Roundy. Picoradio supports ad hoc ultra-low power wireless networking. *Computer*, 33(7):42–48, 2000.

[20] Kahn RHKJM and KSJ Pister. Mobile networking for smart dust. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 99), Seattle*, 1999.

[21] BG Celler, T Hesketh, W Earnshaw, and E Ilsar. An instrumentation system for the remote monitoring of changes in functional health status of the elderly at home. In *Engineering in Medicine and Biology Society, 1994. Engineering Advances: New Opportunities for Biomedical Engineers. Proceedings of the 16th Annual International Conference of the IEEE*, pages 908–909. IEEE, 1994.

[22] Loren P Clare, Nikolai Romanov, and A Twarowsk. Wireless sensor networks for area monitoring and integrated vehicle health management applications. *American Institute of Aeronautics and Aeronautics (AIAA)*, 1999.

[23] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 82–87. IEEE, 1994.

[24] Feng Xue and Panganamala R Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless networks*, 10(2):169–181, 2004.

[25] Sameera Poduri and Gaurav S Sukhatme. Constrained coverage for mobile sensor networks. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 165–171. IEEE, 2004.

[26] Qun Li, Javed Aslam, and Daniela Rus. Hierarchical power-aware routing in sensor networks. In *Proceedings of the DIMACS workshop on pervasive networking*. Citeseer, 2001.

[27] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 146–159. ACM, 2001.

[28] Jim Chou, Dragan Petrovic, and Kannan Ramchandran. Tracking and exploiting correlations in dense sensor networks. In *Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference on*, volume 1, pages 39–43. IEEE, 2002.

[29] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[30] Samir Goel and Tomasz Imielinski. Prediction-based monitoring in sensor networks: taking lessons from mpeg. *ACM SIGCOMM Computer Communication Review*, 31(5):82–98, 2001.

[31] Katayoun Sohrabi and Gregory J Pottie. Performance of a novel self-organization protocol for wireless ad-hoc sensor networks. In *Vehicular Technology Conference, 1999. VTC 1999-Fall. IEEE VTS 50th*, volume 2, pages 1222–1226. IEEE, 1999.

[32] Eugene Shih, Paramvir Bahl, and Michael J Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 160–171. ACM, 2002.

[33] Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen. Self-organisation of sensor networks using genetic algorithms. *International Journal of Sensor Networks*, 1(3-4):241–252, 2006.

[34] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots*, 13(2):113–126, 2002.

[35] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

[36] Shiow-yang Wu, Chao-Hong Liu, and Chen-Kuang Tzeng. Self-organization strategies for dynamic context coverage in capability-constrained mobile sensor networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 1, pages 8–pp. IEEE, 2006.

[37] Goce Trajcevski, Peter Scheuermann, and Herve Brönnimann. Mission-critical management of mobile sensors: or, how to guide a flock of sensors. In *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, pages 111–118. ACM, 2004.

[38] Hasan Çam, Suat Özdemir, Devasenapathy Muthuavinashiappan, and Prashant Nair. Energy efficient security protocol for wireless sensor networks. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 5, pages 2981–2984. IEEE, 2003.

[39] Adrian Perrig, Robert Szewczyk, Justin Douglas Tygar, Victor Wen, and David E Culler. Spins: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[40] NIST FIPS PUB. 81-des modes of operation, 1980.

[41] Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen. Dynamic optimization of secure mobile sensor networks: A genetic algorithm. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 3413–3418. IEEE, 2007.

[42] Koral Ilgun, Richard A Kemmerer, and Phillip A Porras. State transition analysis: A rule-based intrusion detection approach. *Software Engineering, IEEE Transactions on*, 21(3):181–199, 1995.

[43] Phillip A Porras and Peter G Neumann. Emerald: Event monitoring enabling response to anomalous live disturbances. In *Proceedings of the 20th national information systems security conference*, pages 353–365, 1997.

[44] Rahul Khanna and Huaping Liu. System approach to intrusion detection using hidden markov model. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 349–354. ACM, 2006.

[45] Rahul Khanna and Huaping Liu. Distributed and control theoretic approach to intrusion detection. In *Proceedings of the 2007 international conference on Wireless communications and mobile computing*, pages 115–120. ACM, 2007.

[46] Waldir Ribeiro Pires, Thiago H de Paula Figueiredo, Hao Chi Wong, and Antonio A Loureiro. Malicious node detection in wireless sensor networks. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 24. IEEE, 2004.

[47] Xiaojiang Du, Mohsen Guizani, Yang Xiao, and Hsiao-Hwa Chen. Two tier secure routing protocol for heterogeneous sensor networks. *Wireless Communications, IEEE Transactions on*, 6(9):3395–3401, 2007.

[48] Ana Paula R da Silva, Marcelo HT Martins, Bruno PS Rocha, Antonio AF Loureiro, Linnyer B Ruiz, and Hao Chi Wong. Decentralized intrusion detection in wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, pages 16–23. ACM, 2005.

[49] Afrand Agah, Sajal K Das, Kalyan Basu, and Mehran Asadi. Intrusion detection in sensor networks: A non-cooperative game approach. In *Network Computing and Applications, 2004.(NCA 2004). Proceedings. Third IEEE International Symposium on*, pages 343–346. IEEE, 2004.

[50] Fang Liu, Xiuzhen Cheng, and Dechang Chen. Insider attacker detection in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1937–1945. IEEE, 2007.

[51] Sarjoun S Doumit and Dharma P Agrawal. Self-organized criticality and stochastic learning based intrusion detection system for wireless sensor networks. In *Military Communications Conference, 2003. MILCOM'03. 2003 IEEE*, volume 1, pages 609–614. IEEE, 2003.

[52] Kinshuk Govil, Edwin Chan, and Hal Wasserman. Comparing algorithm for dynamic speed-setting of a low-power cpu. In *Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 13–25. ACM, 1995.

[53] Lin Yuan and Gang Qu. Design space exploration for energy-efficient secure sensor network. In *Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference on*, pages 88–97. IEEE, 2002.

[54] Padmanabhan Pillai and Kang G Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 89–102. ACM, 2001.

[55] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 188–200. ACM, 2004.

[56] K Bult, Amit Burstein, D Chang, M Dong, M Fielding, E Kruglick, J Ho, F Lin, T-H Lin, William J Kaiser, et al. Low power systems for wireless microsensors. In *Low Power Electronics and Design, 1996., International Symposium on*, pages 17–21. IEEE, 1996.

[57] Guoliang Xing, Chenyang Lu, Ying Zhang, Qingfeng Huang, and Robert Pless. Minimum power configuration in wireless sensor networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 390–401. ACM, 2005.

[58] Riccardo Gusella. Characterizing the variability of arrival processes with indexes of dispersion. *Selected Areas in Communications, IEEE Journal on*, 9(2):203–211, 1991.

[59] Andrew Wang, SeongHwan Cho, Charles Sodini, and Anantha Chandrakasan. Energy efficient modulation and mac for asymmetric rf microsensor systems. In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 106–111. ACM, 2001.

[60] Curt Schurgers, Olivier Aberthorne, and Mani Srivastava. Modulation scaling for energy aware communication systems. In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 96–99. ACM, 2001.

[61] Inki Hong, Darko Kirovski, Gang Qu, Miodrag Potkonjak, and Mani B Srivastava. Power optimization of variable-voltage core-based systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(12):1702–1714, 1999.

[62] Jay Nejedlo and Rahul Khanna. Intel® ibist, the full vision realized. In *Test Conference, 2009. ITC 2009. International*, pages 1–11. IEEE, 2009.

[63] Luca Benini and Giovanni de Micheli. System-level power optimization: techniques and tools. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 5(2):115–192, 2000.

[64] Changhui Hu, Steven Redfield, Huaping Liu, Rahul Khanna, Jay Nejedlo, and Patrick Chiang. Transmitter equalization for multipath interference cancellation in impulse radio ultra-wideband (ir-uwb) transceivers. In *VLSI Design, Automation and Test, 2009. VLSI-DAT'09. International Symposium on*, pages 307–310. IEEE, 2009.

[65] Lenitra M Durham, Milan Milenkovic, and Phil Cayton. Platform support for autonomic computing: A research vehicle. In *Autonomic Computing, 2006. ICAC'06. IEEE International Conference on*, pages 293–294. IEEE, 2006.

[66] IEEE Standards Committee et al. Ieee standard test access port and boundary-scan architecture. *IEEE Std*, pages 1149–1, 1990.

[67] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[68] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[69] Intel Corporation. i7-900 desktop processor extreme edition series data sheet.(july 2010). http://download.intel.com/design/processor/datashts/320765.pdf.

[70] Smart Battery System Implementers Forum (SBS-IF). *[SMBUS 2.0] System Management Bus (SMBus) Specification, Version 2.0*, August 2000. Available online at http://www.smbus.org/specs/smbus20.pdf.

[71] Inc. (DMTF) Distributed Management Task Force. *[MCTP 2016] Management Component Transport Protocol (MCTP) Overview White Paper, Version 1.0.0a*, July 2007. Available online at http://www.dmtf.org/sites/default/files/standards/documents/DSP2016.pdf.

[72] Inc. (DMTF) Distributed Management Task Force. *[NCSI 0222] Network Controller Sideband Interface (NC-SI) Specification, Version 1.0.0*, July 2009. Available online at http://www.dmtf.org/sites/default/files/standards/documents/DSP0222_1.0.0.pdf.

[73] Douglas SJ De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[74] Nicolas Burri, Pascal Von Rickenbach, and Roger Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 450–459. IEEE, 2007.

[75] Samuel Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[76] Răzvan Musăloiu-E, Chieh-Jan Mike Liang, and Andreas Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*, pages 421–432. IEEE, 2008.

[77] Deepak Ganesan, Răzvan Cristescu, and Baltasar Beferull-Lozano. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. *ACM Transactions on Sensor Networks (TOSN)*, 2(2):155–181, 2006.

[78] Jie Liu, Feng Zhao, Jeff O'Reilly, Amaya Souarez, Michael Manos, Chieh-Jan Mike Liang, and Andreas Tersiz. Project genome: Wireless sensor network for data center cooling. *The Architecture Journal*, 18:28–34, 2008.

[79] Bosch Sensortec GmbH. *BME280 Integrated Environmental Unit, Document Number BST-BME280-DS001-10, Document Version 1.1*, May 2015. Available online at `https://www.adafruit.com/datasheets/BST-BME280_DS001-10.pdf`.

[80] Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen. Proactive power optimization of sensor networks. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 2119–2123. IEEE, 2008.

[81] Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen. Self-organization of wireless sensor network for autonomous control in an it server platform. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010.

[82] Rahul Khanna, Kshitij A Doshi, Christian Le, John Ping, Martin P Dimitrov, Mariette Awad, and Melissa Stockman. Dynamic energy allocation for coordinated optimization in enterprise workloads. In *Energy Aware Computing (ICEAC), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[83] Rahul Khanna, Jaiber John, and Thanunathan Rangarajan. Phase-aware predictive thermal modeling for proactive load-balancing of compute clusters. In *Energy Aware Computing, 2012 International Conference on*, pages 1–6. IEEE, 2012.

[84] Piyush Gupta, Shashidhar G Koolagudi, Rahul Khanna, Mrittika Ganguli, and Ananth Narayan Sankaranarayanan. Analytic technique for optimal workload scheduling in data-center using phase detection. In *Energy Aware Computing Systems & Applications (ICEAC), 2015 International Conference on*, pages 1–4. IEEE, 2015.

[85] John Ping, Zhiqiang Gao, and Rahul Khanna. A novel control design approach for sever subsystems: the concept of active disturbance rejection and a case study. *Intel® Technology Journal*, 16(2):98, 2012.

[86] Rahul Khanna, Huaping Liu, and Hsiao-Hwa Chen. Reduced complexity intrusion detection in sensor networks using genetic algorithm. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–5. IEEE, 2009.

[87] Rahul Khanna, Huaping Liu, and Thanunathan Rangarajan. Wireless data center management: Sensor network applications and challenges. *Microwave Magazine, IEEE*, 15(7):S45–S60, 2014.

[88] Intel Corporation. *Intel(R) Quark(TM) microcontroller D1000 Datasheet, Document Number: 332910-1.0*, September 2015. Available online at `http://www.intel.com/content/www/us/en/embedded/products/quark/mcu-d1000/quark-d1000-datasheet.html`.

[89] Intel Corporation. *Intel(R) Quark(TM) SoC X1000 Datasheet, Document Number: 329676-005US*, August 2015. Available online at `http://www.intel.com/content/www/us/en/embedded/products/quark/quark-x1000-datasheet.html`.

[90] Digi International Inc. *XBee/XBee-PRO ZigBee RF Modules User Guide, Document Number: 90000976 X*, December 2015. Available online at `http://ftp1.digi.com/support/documentation/90000976.pdf`.

[91] SparkFun Electronics. *SparkFun XBee Explorer USB*. Available online at `https://www.sparkfun.com/products/11812`.

[92] Intel Corporation. *Intel(R) Quark(TM) microcontroller D1000 Customer Reference Board v3, Document Number: 333097-001US*, September 2015. Available online at `http://www.intel.com/content/www/us/en/embedded/products/quark/mcu-d1000/quark-d1000-crb-user-guide.html`.

[93] Intel Corporation. *Tools for Intel(R) Quark(TM) Microcontroller Software Developers*. Available online at `http://www.intel.com/content/www/us/en/embedded/products/quark/mcu/d1000/software-and-drivers.html`.

APPENDIX

Appendix A – Development of Dense Sensor network using Intel(R) Quark(TM) D1000 and XBEE

A.1 Development of Dense Sensor network using XBEE

Exceptionally large clusters (300-3000 nodes) in a high density computing environments are prone to noise and interference that can result in degraded throughput at increased energy consumption. This can lead to non-optimal feedback loop (Decision Process) which otherwise has to be fast, optimal and automated. This section proposes a full-stack reference solution based on Intel®Quark(TM) D1000 MCU [88] and XBEE that creates a self-configuring wireless sensor network which can be used to experiment with high-density large scale sensor network. This reference solution can detect variations in temperature as well as other atmospheric variables and upload them to the Intel IoT Analytics Dashboard®. This allows to use an analytic platform together with the data collected by Wireless Sensor Network to provide feedback to the control system for efficient control.

A.1.1 Design

Our goal for this reference wireless sensor network is for it to be self-configuring, scalable and low-power. To satisfy these goals we tune all wireless devices around the following parameters.

A.1.1.1 Channel

The Xbee S1 offers 16 available channels, starting with `0x0B` all the way to `0x1A`. Each of these channels has it's own bandwidth with sufficient distance from neighboring channels to ensure minimum interference across channels. The Xbee S1 also allows for jumping between different channels with some considerable delay between jumps. In a small-scale demonstrations, we may use 2 or 3 of the 16 available channels, due to the fact that 3 channels are enough to scale up to 40 sensors. In the case of large Datacenters more channels can be used to maximize performance, although our network scales exponentially to the number of channels used. In theory, if all 16 channels were used, we could scale all the way to 65535 online sensors.

A.1.1.2 Role

In our design we created three different modes of operation for all participants in the wireless network. We call them `Roles`.

1. Gateway
2. Router
3. Edge (Sensor Node)

The Gateway role is statically assigned only to the Intel(R) Galileo board (Intel(R) Quark(TM) SoC X1000 [89]), together with a static 16-bit UID, which is `FF EE`. The Gateway is responsible for creating the graph model of the wireless sensor network as well as acting as a bridge to the cloud. The role of the

Gateway is statically assigned before the network powers on, as the Gateway is responsible for assigning roles to all other nodes in the network. Except for roles, the Gateway assigns Channel information and Energy levels to all devices. In the reference prototype we describe a stand-alone Gateway, although it would be entirely possible to establish a form of inter-Gateway communication and thus incorporate more Gateways in our existing design.

The Router role is assigned to nodes which are strongly connected, nodes without which our network would have to split in two or more sub-networks. Routers are important as they enable all nodes to reach the Gateway, so they always operate in a higher Power level than Edge nodes and sleep more rarely.

The Edge device (or Sensor Nodes) role is assigned to nodes which are not strongly connected and they operate solemnly as data transmitters. They can sleep in regular intervals to conserve battery and if they fail the network will continue operating normally.

A.1.1.3 Energy Levels

The Xbee S1 operates on 4 different power levels, ranging from $1-4$, where 1 is the lowest energy consumption level and 4 is the highest. Here it would be useful to notice that the power level of the Xbee S1 wireless nodes is proportional to the signal strength. Meaning that if two nodes are close, they should use the lowest energy level, to save on battery life and also reduce noise in other nodes' communication. So there are two reasons for which we are tuning the power levels of our nodes, conserving battery power and lower interference across nodes who are at a moderate distance.

A.1.2 Protocol

In the communication between edge nodes and the Gateway there are two phases; The introduction phase and the stable phase. When all wireless sensors start operating, the Gateway enters the Introduction Phase, where all nodes are in the same channel (0x0B) and advertise their Global Unique Identifier (GUID). A flooding algorithm makes sure all advertised packets reach the Gateway, who in turn, creates a graph of the network topology. When the graph is done, all sensors are configured to use the channel and power level that minimizes noise and maximizes communication with their immediate neighbors.The Gateway is responsible for gathering signal strength data and neighbors relationship data from all sensors, before assigning them a channel and a power level. When the final assignment is made, our distributed system moves from the Introduction phase to the Stable Phase, for both of which we will elaborate further.

A.1.2.1 Introduction Phase

The introduction phase starts with the edge nodes, whose goal is to be discovered and assigned a role by the Gateway at the end of the phase. The default Xbee S1 configuration has two modes of addressing available, 16-bits and 64-bits. 16-bit addressing should be enough to address 65535 different nodes, while retaining a small packet overhead, so that was our prefered method of addressing for this project. However, the 16-bit address of the XBee is assigned to 0000 by default for all nodes, but luckily their 64-bit address is set to a 64-bit Unique ID. Instead of dealing with the problem of distributing UIDs for a low-energy distributed system, we may instead hash the 64-bit address to a 16-bit UID. Our hashing function for the beta drops the high bits of the 64-bit SL keeping only the low 16-bits. This hashing function does not scale well due to the high possibility of UID collisions, therefore we included a hashing function using CRC16, for which the Intel(R) Quark(TM) D1000 has inbuilt support.

After a 16-bit UID is generated from the 64-bit Xbee address, it is assigned to the Xbee by issuing the local AT command `ATMY` followed by the generated 16-bit UID and then the local AT command `ATWR` which makes the change persistent in the Xbee firmware across hardware resets. Now that the 16-bit serial number is set, edge devices flood empty Introduction packets with their serial number targeted to the Gateway. To help addressing the Gateway in the pre-Introduction phase period, we have assigned a static serial ID to it, which is `FF EE`. During the introduction phase, all nodes flood messages with `FF EE` as the recipient, so the Gateway will get N packets at the beginning if there are N total devices. Some of these could be duplicates, so we look-up their checksum and drop duplicates to save bandwidth.

Our flooding algorithm gets over the issue of infinite loops inside the network by using a TTL counter variable embedded in each flooded packet. This design is very similar to how TCP/IP solves network loops, with the difference that our TTL counter starts at 4 and decreases by 1 with each hop.

Now, the Intermediate sensors that receive serial numbers from their neighbors have to forward them, by re-packing them into a new packet, the full Introduction ('I') packet. The only difference between a full Introduction packet and an empty Introduction packet is that the former includes an adjacency list of the nodes from which our current node has heard from. This way data are consolidated across hops in the wireless network. Example packets are given at the end of this section. All of these adjacency lists are received eventually by the Gateway. Galileo's 16 bit serial is also in the adjacency list, so first order neighboring nodes of the Gateway are the first to be assigned a role, usually that of a Router ('R').

A.1.2.2 Stable Phase

In the stable phase our only goal is to route data packets with as little overhead as possible, taking advantage of the network topology. The Intel(R) Quark(TM) D1000 based sensor nodes start sending sensor data and RSSI information to the Gateway, using targeted addressing this time. Packet types can have either an uppercase or lowercase type. If it is lowercase, it is setting an attribute. If it's uppercase, it is a full-size packet (see below). During the stable phase the Intel(R) Quark(TM) D1000 Nodes are asleep most of the time and wake up only on a UART interrupt, triggered by incoming Xbee Frames, or WDT interrupt, that comes periodically and samples the ADC.

A.1.2.3 Full-packets (uppercase)

Each packet has a source and destination bytes, (consuming total of 32 bits) as described below:

I: **Serial number**: "I" is transmitted (Table A.1, Table A.2) even if there are no neighbors. Followed by "this" serial and a list of any neighbors (sometimes 0, if it's the first transmission)

S: **Setup**: Sent by Gateway (Table A.3) at the end of receiving ”I” packets, S(role, channel, power).
D: **Data packet**: Sent by Sensor Node (Table A.5). A(attr, value) (could be temp or RSSI etc)

A.1.2.4 Attribute Packet (lowercase)

This packet can be transmitted by either Gateway or a Sensor Node. This packet is configured for Destination address. D(attr, value) (key-value pairs that every device has, including gateway)
n: new neighbor found by edge device
d: neighbor lost by edge device
c: channel set by Gateway (Table A.4)
r: role set by Gateway (edge, router)
p: power level set by Gateway

A.2 Galileo Software

The Galileo Gateway executes 3 pieces of software continuously, consisting of modular Python Gateway behavior software, the Intel IoTkit Agent in the background, and an HTTP server hosting d3.js visualizations and live graphics.

A.2.1 Python Gateway Software

The Galileo runs Yocto Linux, allowing us to attach XBee hardware on USB ports as Linux ttyUSB devices. This allows serial communication with the python program that handles packets, dispatches tasks, and sends out configuration messages to the entire network. On top of this serial communication layer enabled by the pyserial library, the network discovery and formation code runs the algorithm that decides the ideal network topology. During the operation of the Galileo as the gateway of the network, There are four phases: Galileo hardware initialization, node discovery, setup/configuration, and normal operation.

A.2.1.1 Galileo Hardware Initialization

The user specifies a target number of channels for the completed network. This target number should ideally match the amount of physical XBee radios that are connected to the Galileo. If there is a mismatch, however, the Galileo will auto-detect the connected hardware and adjust the target number of channels accordingly. The devices are then set to unique channels, with the first being the setup channel for the entire network. All of the nodes will start this channel, and all of the data will flow through the gateway-attached XBee that has just been configured to operate on this channel. The remaining XBee devices will be activated later, when the network has been reconfigured to operate on multiple channels.

A.2.1.2 Node discovery

During the node discovery process, the Galileo collects a range of packets that allow it to form an adjacency list of the devices in the network. This comes in as a 60-second continuous stream of data on a single channel. The packets can just declare the existence of the node, or could include a node and several of its neighbors within one hop (direct communication between these two nodes).

The information comes with RSSI (Received Signal Strength Indicator) as well, so the gateway software takes note of the signal strength of the connections between nodes. The Gateway also supports a link strength threshold, allowing fine control over the connections deemed valid for reliable transmission. Some of this information is hopped several times to get to the Gateway, while other nodes can communicate directly with the gateway's radio hardware. Once all of the raw information is gathered and sorted, the algorithm to setup and configure the nodes in the most optimal way proceeds. Figure A.1 illustrates a low-power mesh-connectivity orchestrated by the gateway that create sub-networks using channel diversity.

A.2.1.3 Node Setup and Configuration

Nodes within direct range of the gateway are "'roots"' of the network which can serve as the branching points for separate channels. The Galileo first decides how many channels can be created within the user-defined target number of channels, set as a parameter in the python configuration. Although a network could have a large number of root nodes within range of the Galileo, the algorithm deciding

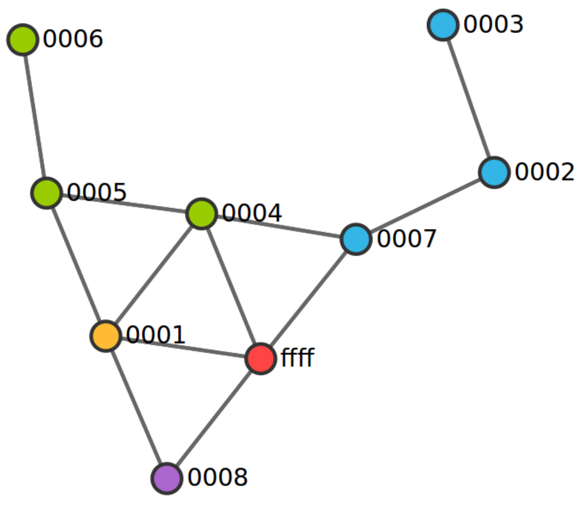


Figure A.1: Mesh connectivity between 8 Xbee Sensor Nodes and Galileo Gateway

the best arrangement of nodes takes into account several factors. The best roots are those with a strong connection to the gateway and strong connections to as many other nodes as possible. A bad root has a weak connection to the gateway and can't communicate with a significant number of nodes. Once the number of appropriate roots has been determined, and the best roots have been prioritized, the Galileo knows how many channels the network will create. It then sends directed setup packets to each node, assigning the role of that node as well as its new channel and power level. All of the nodes apply these packets after a configuration timeout, and the newly-created network is now online. Because the network is now split between multiple channels, the data submission is much faster because of the lack of full-network flooding (as occurs during the Node Discovery process).

A.2.1.4 Calculating the best graph using Djikstra's algorithm

In this section, we describe a simple algorithm that can be used for fewer objectives in lieu of genetic Algorithm as described in Chapter 8. In addition to considering the absolute number and strength of connections in building sub-channels, Gateway assisted sensor-provisioning process may use Djikstra's algorithm to calculate the shortest path if there is more than one way to include a node in a channel. The edge weights for the algorithm are the RSSI values that are constantly forwarded and updated from the nodes to the Gateway, providing up-to-date information about the strength of the connection between nodes. As a result, we can prioritize two strong hops over a single weak hop based on thresholds. These thresholds can be tuned in the sensor network environment to produce the best results.

The goal is to find the point where adding a hop becomes more reliable than trusting a weak connection. These are all factors that we account for when using Djikstra's algorithm to calculate the network. The internal Galileo code that calculates the nodes and the best paths is efficient and memorized on multiple simultaneous threads, ensuring that the network (re)calculation is effective and reliable during normal operation, when the Gateway is taxed with hundreds of packets a minute. Because it can be inefficient to recalculate an entire network's configuration when a node status changes, we also support partial network reconfiguration within sub-channels and an allowed number of dropped nodes before recalculation (useful for giant networks). This allows small changes to the overall network to keep nodes online without recalculating large portions of the graph too often.

A.2.1.5 Stable Phase Operation

After the nodes have been configured, the Galileo begins using all of its radios on all of the just-configured channels. This works as an asynchronous multi-threaded python process on Galileo, with separate threads performing packet acquisition and handling for each network channel. During this time, the gateway handles packets of all types on all of the channels, mostly composed of data packets from individual nodes. These can contain temperature and other data center metrics. Other packets that are handled during normal operation include "'diff"' packets, which announce a network topology change (node disappeared, new node discovered). In these cases, part of the network will be reconfigured to adapt to the change in topology.

A.3 Analytics Backend

A.3.1 IoTKit-Agent and Backend

The Intel IoTKit Agent is a node.js service that sends data to Discovery Peak Cloud Dashboard. This gives us server-side data for multiple nodes and parameters, and an adjustable timeline for the window of data. The Python program sends received multiple-channel datapoints to the IoTKit-agent, which supports multiple key-value pairs for data. In our submissions, the keys are the unique serial number-based IDs of the nodes on the network, and the values are time-coded temperature data-points.

A.3.2 HTTP Server on Galileo

A unique feature of our network is the ability to monitor the complete network topology and show a live visualization. Because of this information that is live-updated on the Galileo, we are able to have a multiple-client live webpage showing the D3.js live network topology.

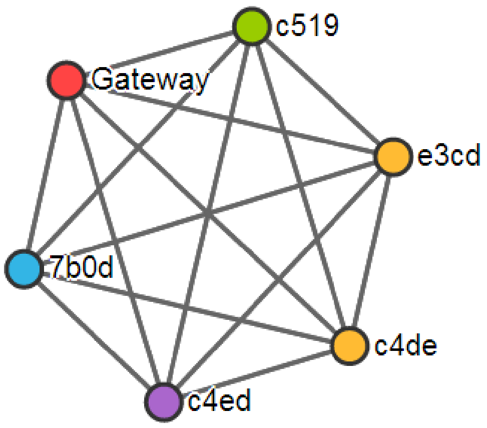


Figure A.2: Sensor Nodes with allocated channels (colors) and the links between the channels represent the connectivity attributes (Like RSSI) corresponding to the default channel.

Figure A.2 illustrates multiple channels with different colors where the links between channels represent the collection of link attributes (RSSI as edge labels). These attributes are used for optimal distribution of channels and development of optimal sub-networks using one of the many optimization algorithms

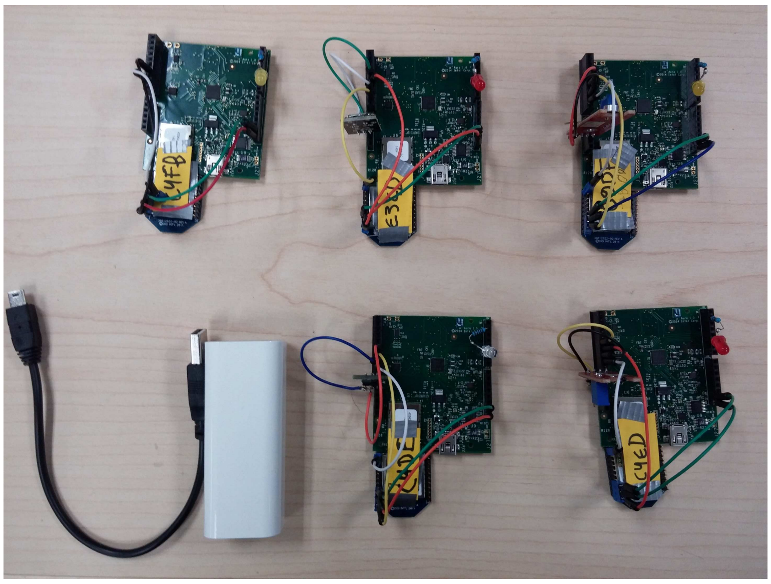


Figure A.3: Low-Power Intel(R) Quark(TM) D1000 Development Board connected to XBee that uses UART connectivity for AT command execution.

A.4 Hardware Implementation

This section enumerates the steps involved in creating an end-end connectivity using Intel Galileo Gateway, Intel Quark D1000 MCU and Xbee 802.15.4 controller. Figure A.3 illustrates Intel(R) Quark(TM) D1000 as a low-energy x86 microcontroller that acts as our edge nodes.

A.4.1 Required Hardware for WSN sensor

- 1x Xbee S1 [90]

  1x Xbee Explorer (with FTDI and mini-USB) [91]

- 1x Intel(R) Quark(TM) D1000 Board CRB-V3 [92]

- 1x ISSM 2015 for Intel(R) Quark(TM) D1000 [93]

- 1x LED

- 1x 1K Resistor

- 1x Analog Sensor (3 pins; -, + and S)

A.4.2 Assembly

A.4.2.1 Status LED

1. Start with the ISSM 2015 for Intel(R) Quark(TM) D1000 SDK.

2. Connect XPB2 (GPIO[0]) to the positive pin of the LED, in series with a 1K resistor and to the GND of the Intel(R) Quark(TM) D1000 CRB-V3. This LED is used for debugging and activity control. It should be blinking frequently in normal operation mode.

A.4.2.2 Xbee S1/Xbee Explorer

1. Connect the $V_{3.3}$ pin to the 3.3V pin on the Xbee Explorer.

2. Connect the GND pin to the GND pin on the Xbee Explorer.
3. Connect the TXD-XPD1 pin of the Quark D1000 Development board (CRB-V3) to the DIN pin of the Xbee Explorer.
4. Connect the RXD-XPD0 pin of the CRB-V3 development board to the DOUT pin of the Xbee Explorer.

A.4.2.3 Analog Sensor

1. Connect the + and - of the Analog Sensor to the 3.3V and GND of the Quark D1000 CRB-V3 development board.
2. Connect the S of the Analog Sensor to pin XPC1 (GPIO[4]).

Now compiling and installing the firmware with Intel(R) ISSM 2015 SDK should create a working edge node.

A.4.3 Galileo Hardware

Galileo is a reasonable choice as the Gateway device because of its built-in internet connectivity (WiFi and Ethernet) and its high-powered Intel Atom processor.

A.5 Example packets

This section illustrate examples of the packet format that are required for provisioning as well as establishing data communication within the sensor-network as

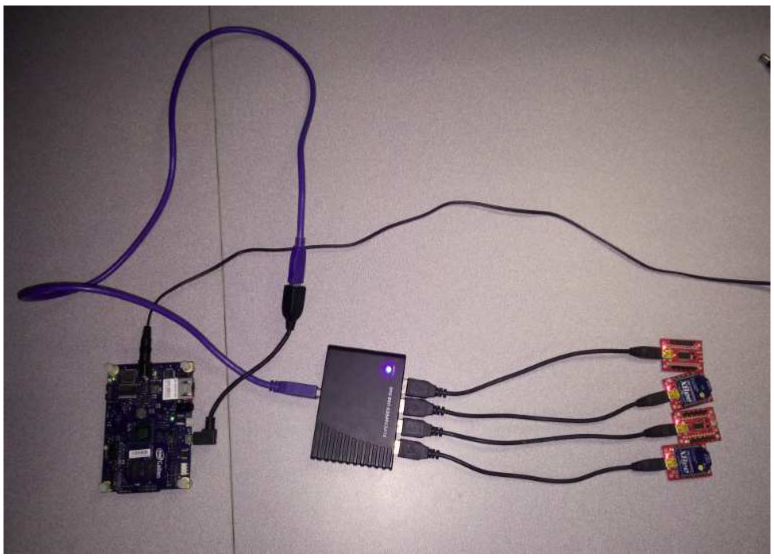


Figure A.4: Galileo (Intel(R) Quark(TM) X1000) based Router with 4 802.15.4 Channels configured using Xbee

well as the gateway. These packets vary from "'Introductory Packets"' originating from *sensor nodes* and *edge nodes* to introduce themselves to the gateway to *Setup Packets* from Gateway to assign specific roles to the sensor nodes. Additional packets are assigned to *Change Channels*, *Send Data Packets* and *Receive Sensor Attributes*.

7E	Xbee TX Code
00 0B	Xbee Length
01 01 00 00 00	Padding
49	*I*
FF FF	dest
00 01	src
04	TTL
B1	checksum

Table A.1: Empty Introduction(I) Packet from edge node with ID = 00 01 to the Gateway (FF EE)

7E	Xbee TX Code
00 0E	Xbee Length
01 01 00 00 00	Padding
49	*I*
FF FF	dest
00 01	src
00 03 00 04	Adjacent nodes
9F	RSSI
04	TTL
B0	checksum

Table A.2: Introduction Packet (I) Packet from edge node 00 01 to Gateway (FF FF) with adjacency list { 00 03, 0 04 }.

7E	Xbee TX Code
00 0E	Xbee Length
01 01 00 00 00	Padding
53	*S*
00 03	dest
FF FF	src
45 0F 04	Role: *E*, Channel: 0F, Power: 04
04	TTL
CA	checksum

Table A.3: Setup(S) packet from Galileo to node 00 03 setting role = E (edge), channel = 0F, power = 04

7E	Xbee TX Code
00 0B	Xbee Length
01 01 00 00 00	Padding
53	*c*
00 03	dest
FF FF	src
0E	New Channel
04	TTL
B1	checksum

Table A.4: Change channel(c) of node 0003 to channel 0E (instead of issuing a full S command like above)

7E	Xbee TX Code
00 0C	Xbee Length
01 01 00 00 00	Padding
44	*D*
FF FF	dest
00 03	src
74	't'
58	88°
04	TTL
A4	checksum

Table A.5: Sensor data(D) packet from sensor 0003 to Gateway FFFF, temperature(t)=0x74, value: 88°F = 0x58

7E	Xbee TX Code
00 08	Xbee Length
81 00 00	Padding
20	RSSI
00	padding
63	*c*
00 03	dest
FF FF	src
0E	New Channel
B3	checksum

Table A.6: The receiving radio gets a the packet response with RSSI bundled, and the same payload: (for example, in response to the c packet from Table A.4)