Wireless Sensor Networks Scaling and Deployment in Industrial Automation

Samuel Zats

University of California, Berkeley

May 13, 2010

Abstract

This project performs analyses and simulation of a comprehensive wireless sensor network for the Industrial Automation application. Utilizing the WirelessHART architecture, the proposed network consists of 1 million motes within an area of 10 km^2. Each mote contains a single sensor, which samples vibration, pressure, temperature, flow, tank level, or corrosion. This project creates an end-to-end open source simulator to evaluate the scaling limits and challenges for deployment of a reliable (characterized by at least 99.9% reliability), periodically sensing (measuring levels every 10 s), low power (7+ year lifetime power), secure wireless network (encrypted payload) for the specific environment in order to simplify control and increase operating efficiency. In order to meet these goals, the simulation includes the following modules: positioning, connectivity, routing, scheduling, and packet simulation. Real-life data form assumptions and validation in all applicable simulations.

1 Introduction

Until now, wireless sensor networks have been limited to networks with nodes numbering in the thousands. With the release of WirelessHART architectures and increasing maintenance and operational costs for fixed wired networks, wireless sensor networks have been turned to in order to decrease costs and maximize efficiency. An application which arguably presents the greatest challenges and limitations is industrial automation and process control. Topologically, industrial automation environments typically span large areas with varying sensor densities. Networks must ensure the highest possible level of reliability while delivering sampled data at periodic intervals. In this project, we explore the end-to-end elements of such a complex network by creating an interactive two-part simulation for both network topology and packet flow.

In this introduction, we outline wireless sensor networks as a whole and lay out the industrial automation application along with the requirements defined by the environment. Additionally, we introduce the network and packet simulator. The report reviews the WirelessHART architecture and explores both the network simulator and packet simulator at the modular-level. Utilizing the simulators, we present analysis for the simulated networks and overview findings. Finally, we present our future work and conclusions.

1.1 Wireless Sensor Networks

While wired sensor networks are seeing skyrocketing labor, installation, and maintenance costs, wireless sensor networks in the past decade have seen an immense decrease in costs for computation, sensing, and communication [10]. The industrial automation application provides the optimal environment for utilizing the potential of these wireless sensor networks through reliable, high-volume, low-powered, and flexible to save costs, simplify control, and increase operating efficiency. Recent research and deployment into various environments have led to successful deployments in manufacturing plants and engineering facilities [1].

1.2 Industrial Automation

Industrial automation presents the ideal application for wireless sensor networks as complex processes require data sampling and operational management. In this project, we focus on the oil refinery application where existing network infrastructures consist of 750,000 to 1 million motes. Although this is one prominent example, industrial automation can include any factory or control management where a high-volume of sensors periodically report process data to a coordinator. In this space, wireless sensor networks provide the promise to deploy and maintain networks at a fraction of the cost. Specifically, safety regulations greatly impact the cost to expand wired networks for additional sensor nodes. Given the nature of wireless, the associated costs can be decreased by a factor of 100.

Although wireless sensor networks have led to the development of a number of alliances and organizations including ISA, HART, WINA, and ZigBee, to encourage deployment and adoption of the new technology, WirelessHART became the first industrial automation and process control technology standard for wireless communication, upon ratification by the HART Communication Foundation in September 2007[5, 11]. In this report, and subsequently the simulation, we focus on proven WirelessHART technology.

1.3 Environment Requirements

As previously mentioned, we focus on the scaling and deployment of industrial automation sensor networks for oil refinery applications. In oil refineries, wireless sensor networks provide extraordinary financial and safety advantages. Not only are the installation costs dramatically decreased as the high-cost wire infrastructure is no longer necessary, but also the time to install and deploy additional nodes is greatly reduced. More so, the wireless rollout provides improved response for these installations and also allows for monitoring in remote or cost-prohibitive locations, where wired networks create insurmountable challenges.

Figure 1: An aerial photograph of the Shell Refinery in Martinez, CA

Review of national oil refineries shows the following attributes are characteristics and goals for a deployable wireless sensor network:

- Area of 10 km^2
- Network of 10^6 motes
- Sensor densities from 1/cm^2 to 1/100 m^2
- Average mote density of 1/10 m^2
- Each mote contains a single sensor for vibration, pressure, temperature, tank level, or flow rate
- Reporting frequency of 10 s
- Node lifetime of 7 years

For the purposes of the simulation, we have taken these environment requirements and proceed to scale them to 1% due to computing limitations. Thus, we explore a wireless network of 10,000 motes within an area of 0.1 km^2, a square of 316 m by 316 m. Reporting frequency is defined by a constant 10 s and a maximum node lifetime is sought.

1.4 WirelessHART

Even prior to the release of WirelessHART, numerous other wireless communication standards were publicly available, including ZigBee [12], Bluetooth [13], and WiFi's IEEE 802.11 [14]. Although these technologies have been adopted in various other applications, they are unable to provide the strict timing and data collection while meeting the high security requirements needed for industrial automation. For example, both Bluetooth and ZigBee are considered unreliable since there is no end-to-end delay guarantee for communication [11]. Bluetooth is predominantly designed for point-to-point communication and is entirely unable to scale to handle high-volume node networks. ZigBee, which does not channel hop, is expected to have low reliability and delivery rates due to the intensity and harshness of some industrial automation environments. WiFi, the dominant technology for home and business based internet sharing networks, is extremely resource intensive and requires high processing and power devices. Since no technology addresses these specific environment requirements, WirelessHART was formed to solve the inadequacies of existing technologies for industrial automation.

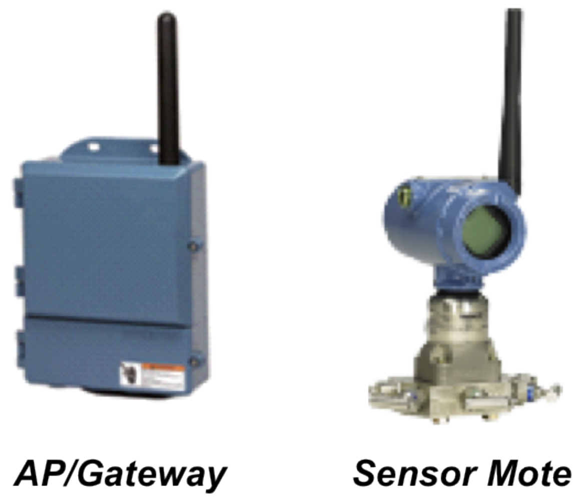


Figure 2: Typical WirelessHART-compliant Gateway and Sensor Mote outfitted for industrial automation, manufactured by Emerson/Rosemount.

WirelessHART, along with its communication protocol Time-Synchronized Channel Hopping (TSCH), provides a comprehensive solution for industrial automation and process control throughout five layers with the following goals and motivations [8]:

- Reliability: Packet delivery rate exceeding 99.9%.
- Low Power: Node battery lifetime exceeding seven years of average use.
- Scalability: Ability to handle thousands to millions of nodes through entire mesh network. Each Access Point must manage hundreds of nodes.
- Flexibility: Various time traffic provided by different types of sensor nodes
- Security: Full end-to-end security and encryption for every packet in the network.
- Environment: Coexist with other wireless technologies that produce RF interference. Handle environmental variations in temperature, sound, vibration, pressure, etc...

1.5 Network Simulator

As the first component of our deployment simulation, the Network Simulator runs a variety of modules in order to take user-inputted network sizes and outputs a WirelessHART-based Superframe schedule. Additionally, the user is able to view a visual representation of the network where motes, APs, and routes are drawn in a coordinate plane.

The Simulator consists of a series of modules including: Positioning, Connectivity, Routing, and Scheduling. On its own, the Simulator will perform random positioning and lowest-cost load-balanced routing to output a Superframe schedule for the network. At each step, the user is able to replace any of the modules to develop and optimize the selected component.

1.6 Packet Simulator

As the second component of our network simulation, the Packet Simulator accepts a Superframe schedule and Packet Delivery Ratio (PDR) link-map to virtually run the packet flow for the network. Based upon the Generation rate, data packets are created and timestamped. Following the Superframe schedule, packets flow through the network with PDR as rates given in the link-map. The Flow Analysis module summarizes the simulation with reliability, latency, and power consumption metrics.

2 WirelessHART

In this section, we overview the WirelessHART standard to provide a comprehensive understanding and consider its strategic design for industrial automation deployment. The protocol, itself, uses the OSI 7-layer communication model to define a five-layer protocol stack[5]. In the following subsections, we explore the standard as it is applicable to wireless sensor networks deployed in industrial automation.

2.1 Layer 1: Physical

WirelessHART provides minimal changes to the standardized IEEE STD 802.15.4-2006 physical layer at 2.4 GHz and incorporates DSSS. As the same physical layer used by ZigBee [12], 15.4 defines the electrical bits and physical device specifications to communicate with the medium. By this standard, radio communication is operated in the ISM band between 2400 and 2483.5MHz at a data rate of 250 kbit/s [11]. Channels are numbered from 11 to 26 and separated by 5MHz intervals.

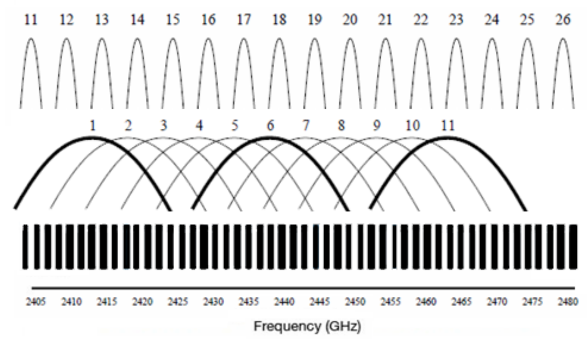


Figure 3: Plot of spectrum with channels delineated, from top to bottom: 802.15.4 HART, 802.11b/g WiFi, 802.15.1 Bluetooth.

In the above figure, we can see that WirelessHART has to share spectrum frequencies with two common technologies including WiFi and Bluetooth. These presents the case for Time-Synchronized Channel Hopping, minimizing the potential for collision and interference.

2.2 Layer 2: Data Link

In order to solve the application challenges, WirelessHART applies a Time Synchronized Mesh Protocol (TSMP) at the second layer [8]. Under TSMP, time-specific *slots* are sent periodically. Slots remain at a constant length, typically 10ms, and each network node must agree on the *Absolute Slot Number*, or ASN, the number of *slots* which have passed since the creation of the network. This process synchronizes all nodes to the same count. As *slots* occur in a repeating pattern, a single period is identified as a *superframe*.

Since TSMP transmits across multiple channels, each slot is further divided into channel-based *cells*. Thus, for each 10ms time *slot*, numerous packets can be sent as long as they are across various channels and occupy separate *cells*. By this structure and the pseudo-random process of selecting a *cell*, TSMP guarantees no collision or interference between *cells*.

Each transmission event occurs in a *slot*, which is defined by a series of properties including: *frame_id* - index of the superframe, *index* - index of the actual slot within the superframe, *type* - identifies the event (TX, RX, or idle), *src_addr* - address of the source node, *dst_addr* - address of the destination node, *channel_offset* - channel utilized for transmission[11].

As TSMP not only transmits across various channels but also hops channels for a given link, a formula is used to calculate an actual channel from a given cell:

$$\textit{Channel} = (\textit{ChannelOffset} + \textit{ASN})\ \%\ \textit{NumChannels}.$$

Even though the standard allows for channels 11-26, channels that are found noisy or have a low probability of successful delivery are avoided by the Network Manager. By blacklisting them, channel use is disabled and results in *NumChannels* to range from 1 to 16, inclusively [4].

2.3 Layer 3: Network and Layer 4: Transport

Under the WirelessHART network architecture, the standard defines a networking hierarchy which includes at least one **Network Manager**, one **Gateway** that collects information, **Motes** that take samples via attached sensors, and optional **Field Devices** that allow for maintenance for process control. The **Network Manager** is responsible for forming the mesh, allowing devices to join, setting the schedule, and monitoring the network. In the middle, the **Gateway** bridges the mesh network to the automation network and allows data to follow among both. Finally, **Field Devices** could be connected to a process device, a routing device, or a portable handheld support device for monitoring and maintenance [5].

In order to create a mesh, each node within the WirelessHART must have the potential to route packets for any other device. The two protocols used include **Graph Routing -** where each node contains a pre-calculated graph of all the other nodes in the network and forwards the packet based on the destination address, and **Source Routing -** with each packet header containing an itinerary of nodes to travel through the network, the device simply forwards the packet to the next node on the list. In our simulation, we leverage Source Routing to create a routing table that assigns each mote to transmit to another.

2.4 Layer 7: Application

Although only the fifth layer in the WirelessHART standard, the application layer is at the top of the OSI model. Through this layer, WirelessHART permits dialogue through various commands and inquires which can be established between the **Field Devices** and **Gateway**. The layer handles the processing and computation of any requests and responses needed per the application [11].

3 Network Simulator

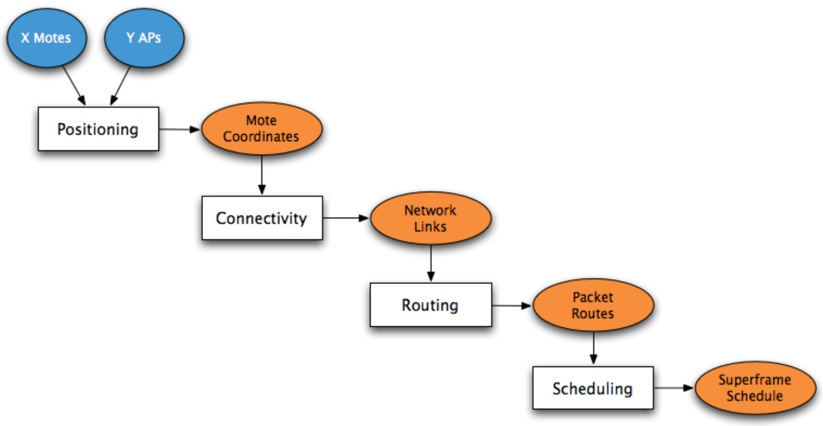


Figure 4: Network Simulator flow chart of Modules.

As the primary simulation tool, we have constructed the Network Simulator to generate a network of motes and perform routing and scheduling of its links. The Simulator consists of the Positioning Module, which creates nodes and randomly selects coordinates, Connectivity Module, which uses the Friis Transmission Model to realistically generate a connectivity matrix, Routing Module, which performs a modified Dijkstra least cost algorithm to create routes, and the Scheduling Module, which assigns links to cells in a schedule.

Upon launching the Network Simulator, the user enters the number of AP/Gateways and Motes as well as Load Factor, used for load balancing in the Routing Module. In its current form, the Simulator operates upon the following customizable assumptions:

- Timeslot duration of 10 ms.
- All motes have a sample generation rate of 10 s.
- Provisioning factor of 3x.
- Superframe of 3.33 s (derived from the previous two statements).
- 15 offsets, one for each channel, as defined by the WirelessHART standard.

Due to computational limitations, we scale both the area and network size to 1% of the actual environment of 10^6 motes in 10 km^2 to a square area with length of 316 m and 10^3 motes.

Additionally, it is important to note that the existing simulator is a comprehensive framework with end-to-end wireless sensor network simulation. Each module can be modified and customized based on the user's intent.

3.1 Positioning Module

The first module, Positioning, creates a mapping from AP/Gateway or sensor motes to 2-dimensional coordinates within our scaled area. As previously noted, our 1% scale of the typical oil refinery produces an area of 0.1 km^2. Currently, the module assumes a square area of side length 316 m. Both APs and motes are randomly distributed throughout the entire area by regenerating a random seed to select coordinates for each device. As the first step, the Module outputs a hash mapping from node to coordinates.

Alternatively, we can replace this module by loading a file of node coordinates for both APs and sensor motes.

3.2 Connectivity Module

In the Connectivity Module, we begin by creating a connectivity model based upon distance and probability. Including an element of randomness to account for multi-path and potential interference, the Module produces an N-by-N Connectivity Matrix that signifies whether or not pairs of nodes are c̈onnectedänd thus, a link can be established.

We have selected to use a model which sets the probability of connectivity for two nodes based upon distance. Taking the node coordinates from the Positioning Module as an input, the Module calculates the distance between each node pair and looks up it's respective connectivity probability. Using a typical weighted coin-flip, we reasonably assume if the nodes are connected and update the Connectivity Matrix.

3.2.1 Friis Transmission

As the basis of our current Connectivity Module, the Friis Transmission equation is coupled with a random probabilistic constant to account for multi-path and interference. As typically presented, the Friis equation of transmission is defined as:

$$P_r/P_t = G_t/G_r(\lambda/4piR)^2.$$

In this equation, P is power in dBm, and G represents gain for transmitter and receiver, respectively. Considering a standard isotropic radio, both G values are 1. R's value is distance between nodes in meters. λ is the wavelength which is simply calculated by:

$$\lambda = c/f.$$

where c represents the speed of light, $3 * 10^8 m/s$ and f is transmission frequency of 2.4 GHz.

Once the coordinates are looked up from the Connectivity Module input, the radius, R, for the link is computed by the distance formula between the two points. To account for multi-path, an additional randomized constant between -40 and 0 dB is added [9]. As all links are now delineated by received power, the threshold of -85 dBm is selected to divide the links into connections and inactive links. For 2.4GHz radios, sensitivity thresholds are typically between -85 dBm and -100 dBm. In our simulation, we have decided to select the conservative threshold of -85 dBm.

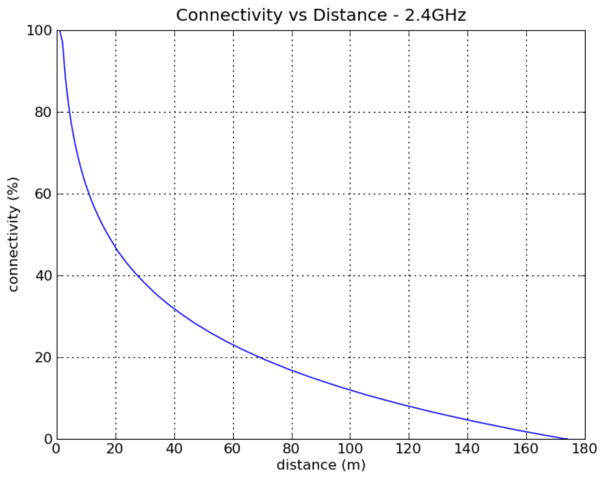


Figure 5: The resulting probability that a link of the given distance will be connected, based upon the Friis Transmission and the Hack Model.

Running the Friis Transmission and Hack Model over a series of 100 trials for each distance between 0 m and the theoretical limit of 175 m, we calculate the probability for a link to be connected. Thus, we are now able to flip a weighted coin to determine link connectivity. The two-step process accounts for highly unpredictable wireless connections and is based upon link stability experiments.

3.3 Routing Module

The Routing Module performs all path assignments by utilizing a commonly-used algorithm, Dijkstra, to select a parent, whether it is AP or sensor mote, for each node in the network. The routing component begins by receiving the Connectivity Matrix from the previous module, Load Factor, and optional Link-Packet Delivery Ratio (PDR) Mapping as inputs and outputs the node to parent mapping based on the algorithm outlined in the following section. In order to achieve the least-cost paths that also load-balance AP assignments, we create temporary structures including to hold link costs based on the Cost Function as well as a running load counter for each AP, incrementing the respective AP upon node assignment.

In the following subsections, we outline the modified routing algorithm and detail the existing cost function for the Routing Module.

3.3.1 Load-Balanced Least-Cost Algorithm

As described above, we modify the Dijkstra's Algorithm to produce our own Routing Algorithm which optimizes upon both AP load balancing and least cost, as calculated by the Cost Function. While Dijkstra's Algorithm locates the least-cost route between a source and destination node, we modify the algorithm to similarly begin from a node but to consider any AP/Gateway as the destination. Thus, the route will incrementally explore the least cost neighbor until an AP is located. This requires us to initialize all APs with a cost equal to 0 and sensor motes with infinity (or a value of reasonably large size. We have selected 1,000). As new paths are explored, the infinity cost is replaced as neighbors are evaluated. Here are the steps for the Algorithm:

1. Assign each AP Node with a cost of 0, Sensor Node with a cost of infinity.
2. Select a mote to be the Initial Node.
3. Using the Connectivity Matrix, look up all of the Initial Node's neighbors and calculate the new Cost by the formula (a summation of link ETX and weighted current load of the AP).
4. Select the node with the Least Cost.
5. Continue Steps 3 and 4, replacing Cost Assignments with lower costs until a Destination AP is found.
6. Continue Steps 2-5 for all potential nodes.

Performing the Algorithm, the Module outputs a routing table that maps each node to a parent.

3.3.2 Cost Function

As part of the Algorithm, the Cost Function for a given route is dependent on the existing Cost plus the additional link Cost and a weight of the assigned AP's load. The Cost formula is defined as:

$$Cost_x = Cost_{current} + ETX_{link} + (\lambda * Load_{AP}/200).$$

In this equation, the new Cost is a summation of current Cost plus the Expected TX Cost of the link and the AP's load by the Load Factor, λ. Each link in the network is assumed to have an ETX value, determined by taking the reciprocal of the PDR. The Link-PDR Mapping is either available as the optional input parameter or by default, a conservative constant of 0.80 used. Link PDR equal to 0.80 has been experimentally verified through numerous network deployments [6]. This PDR constant presents an ETX value of 1.25, which signifies that on average 1.25 transmissions need to be attempted for successful packet delivery.

3.4 Scheduling Module

As the final component to the Network Simulator, the Scheduling Module utilizes the Routing Table generated by the prior module to fill cells in a WirelessHART-based Superframe. Specifically, the WirelessHART standard defines the structure of the resultant schedule, but not the algorithm assignment. In order to assign the links to cells, we have created our own Layer Scheduler. Once all links are assigned, the Module outputs a WirelessHART-compliant Schedule table, indexed by time slot.

3.4.1 WirelessHART Scheduling

The WirelessHART standard specifies that scheduling consists of Superframes that allow for Time-Synchronized Channel Hopping. Given 15 channels in the spectrum, WirelessHART guarantees that 15 links can transmit simultaneously on separate channels, without collision or interference.

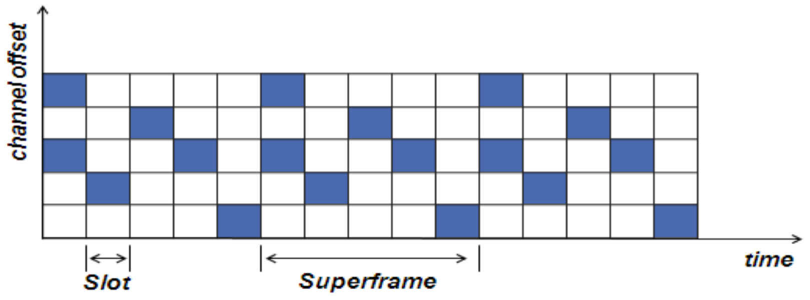


Figure 6: Simplified visual representation of 3 WirelessHART Superframes, consisting of 5 time slots spanning 5 channels.

For our Simulator, we have selected the standard time slot length of 10 ms. Given the 802.15.4-PHY, this duration allows for both packet transmission and a return ACK. As mentioned as assumptions, the Simulation employs a constant 10 s reporting rate. Since we seek a 3x provisioning rate to provide for increased reliability, the network will use a Superframe of 3.33 s to match packet flow with the data sample rate. As time slots are 10 ms, each Superframe includes 333 slots across the standard 15 channels. Thus, each Superframe in our Simulator is able to schedule up to 5,000 cells.

Since the project goal is to scale to 10,000 motes, we will need to schedule 10,000 links at the least. This signifies that we will need an algorithm to assign multiple links to a single cell, considering interference potential. Our algorithm is outlined in the following section as the Layer Scheduler.

3.4.2 Layer Scheduler

As a scheduler is undefined by the WirelessHART standard, we have created the Layer Scheduler. This simple looping algorithm forms the basis for the Scheduler Module. Suggested by its name, the Scheduler assigns in row-major ĺayers' to avoid unpredictable concentration in cell assignment, as witnessed by prior scheduler versions. The term *layer* refers to the number of assignments in a cell for the current *pass*, or loop through the schedule.

Before outlining the algorithm, let's consider the assumptions and restraints in link assignment. First, we assume that the entire network is built on SISO antennas, single input single output. This restricts each device to either receive or transmit in a given time slot. Second, we need a method to model interference. Since we already have a Connectivity Matrix of all possible links, we restrict two links from being assigned if any two nodes, not in the link pair, are connected.

The Layer Scheduler Algorithm operates the following steps:

1. Loop through all nodes creating a list of full paths from source to AP.
2. Order the paths by in decreasing hop count.
3. Start by assigning the first layer of links by row-major, across the entire time interval.
4. Proceed to schedule links, checking that nodes are available and do not interfere within cell assignments.

5. Continue uniform distribution of links per cell through row-major assignment until all links are scheduled.

Once the Scheduler has completed assignment of all paths, the Module outputs a schedule, indexed by slot number and channel offset.

4 Network Analysis

Utilizing our Network Simulator, we step back to investigate individual modules and implications for our ultimate network goal of 1 million motes within an area of 10 km^2. In the following sections, we analyze the effect of each component for deployment.

4.1 Connectivity

As outlined in the Network Simulator section, our Connectivity Module relies on the Friis transmission equation and an added random constant to arrive at a mapping of link distance to its probability of connection. The probability value forms the weight for a coin that is flipped to check for connectivity.

Given our goal density, we scale our area to a square of side length 316 m and disperse an increasing number of motes randomly. Performing the Connectivity Module on each node, we count the average number of neighbors and divide by the entire network size. Thus, we arrive at the following figure:

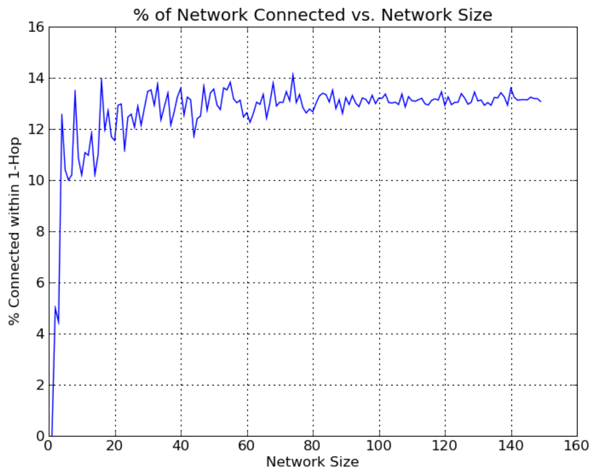


Figure 7: The average percent of network neighbor to a single node as a function of network size.

The graph demonstrates that with a very small network, we see ranging connected percentages. Once we exceed a network size of 20 motes, the percentage stabilizes rapidly at 13% of the network. This suggests that given any mote in the network, we can expect it to be connected to 13% of the nodes in the network.

Given our scaled area, the figure represents the average number of motes connected to a given mote in the network. This figure also represents the degree as a percent of the entire network.

Additionally, we look to explore how the Connectivity Model increases the number of neighbors as a function of distance. For this analysis, we ran a series of trials where all 10,000 motes were randomly placed in our 0.1 km^2 area. Running the probabilistic Connectivity Module, we observed how many nodes are reachable with every additional meter in link distance:

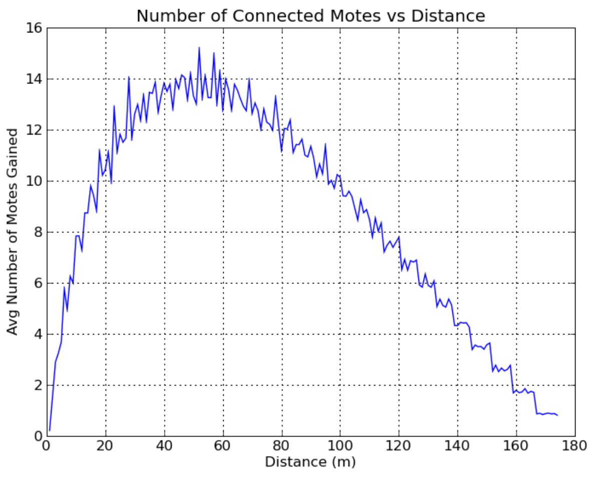


Figure 8: The average number of motes gained as potentially connected as a function of distance.

Examining the results, we see that the number of motes gained peaks at approximately 54 m. At this distance, our connectivity model produces a probability of 22% connectivity. The results are expected as increasing distance correlates directly with the number of potential neighbors, yet the model's probability resembles logarithmic decrease. Thus, the optimal point lies closer to the middle, with a slight skew to the lower distances due to the model's curve.

4.2 Hop Counts

Expanding from the Connectivity to the Routing Module, we begin by exploring the number of hops to reach a node from another. Since the Simulation treats APs and Sensor Motes as randomly placed devices, the first analysis monitors the effect of network size on hop count. Averaging over a series of ten trials, we create small networks in our full area of 0.1 km^2 and calculate the mean number of hops in a path between any two nodes. The resulting graph is displayed below:

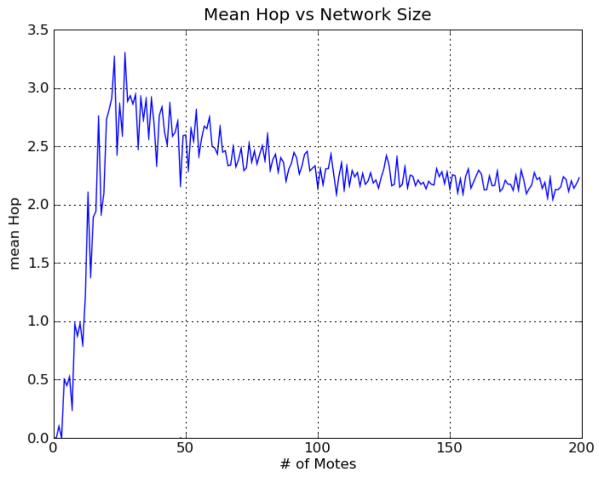


Figure 9: The average number of hops between any two nodes as a function of number of motes.

It is interesting to note that the mean hop fluctuates greatly for small networks. Given that we disregard node pairs where no link can be established, we see the large run up to roughly 30 motes, reaching as high as 3.3 hops. At this point, the entire network is connected and all nodes are able to reach others in the network. With 120 nodes, the network's mean hop count stabilizes at 2.3 hops per node.

The basis for our Routing Module, the modified Djisktra's algorithm seeks to minimize costs along with AP-load factor. In this analysis, we simulate the entire 10,000 sensor network in 0.1 km^2. To exclusively explore potential connectivity and resulting hops, we set the load-factor to 0. To explore one of our original goals, minimize infrastructure costs, we map the relationship between number of APs and the resulting route distribution of hops:

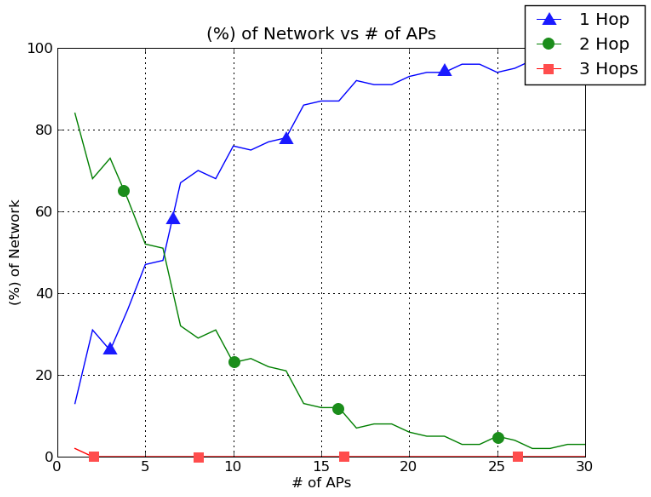


Figure 10: The relationship between hop counts as a function of number of APs.

Given our least cost Dijkstra's Algorithm, the figure represents the relative minimum hop-counts to ensure that the entire network is connected. While almost all nodes are reached with one or two hops even with a low number of APs, the majority of nodes (54%) can be reached by a direct link once the AP count exceeds 7. It is important to note that this analysis does not include any load balancing and only defines routes to connect a sensor mote to an AP.

5 Packet Simulator

As the second simulator for our framework, the Packet Simulator performs a virtual deployment of the network and follows the schedule to study network metrics. Although the Network Simulator created a Coordinate Map, Connectivity Matrix, Routing Table, and Schedule, we only need two inputs to simulate packet flows: Superframe Schedule and Link-PDR Table. Although both the Schedule and PDR Table are outputted by the Network Simulator, the user can alternatively test with any network configuration. This single-file Simulator consists of two components. First, it actually simulates generating and transmitting the packets for a given time period. Next, we analyze the flow by calculating network reliability, packet latencies, and current consumption.

The Packet Simulation Module

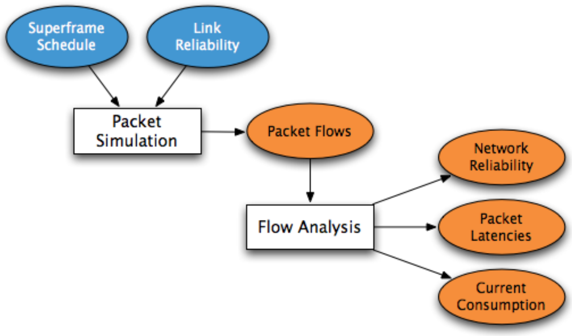


Figure 11: Packet Simulator flow chart of Modules.

In the following sections, we explain how the Simulator performs the packet flow and analysis.

5.1 Packet Simulation

In order to accurately simulate packet flows, the Simulator needs both the Schedule and Link-PDR Table. Before beginning, each sensor mote is allocated a 1.25 KB buffer, which is able to store at most ten 128 byte packets. This buffer is considered FIFO and sampled packets are enqueued at its tail. Consistent with prior assumptions, all sensor motes generate samples every 10 s. It is important to note that no packets *fail* unless a node samples data and the buffer is full. Once packets enter the queues, they can not be dropped. Packet remain at the head of the buffer until the transmissions is ACKed and thus, considered successful.

Packet flows are simulated through the following steps:

1. Check if this is a generation slot. If so, create packet and enqueue into the node's buffer. Drop the packet if the queue is full.

2. Lookup link assignments for the time-slot from the Schedule.
3. Lookup the PDR value for each link assigned in the time-slot.
4. For each link, check if the Destination Node's buffer is not full. If it is full, the assignment is forfeited.
5. Flip a coin weighted with the PDR value to account for packet failure. If it fails, the packet remains at the Source Node.
6. Transmit the packet by dequeueing at the Source Node and enqueueing at the Destination Node.

The Simulator will repeat these actions for each time slot in the duration of the simulation.

5.2 Flow Analysis

Our simplest Module, Flow Analysis calculates network and node statistics from the flow of packets in the network. The analysis currently consists of Network Reliability, Packet Latencies, and Current Consumption.

5.2.1 Network Reliability

Network Reliability is the ratio of successfully received packets to the overall number of generated packets. Given the WirelessHART standard, the network can only drop packets when generated, in the event that the buffer is full. Any packet that is enqueued will remain in the buffer until it is successfully ACKed at the Destination Node. Thus, we simply count network reliability by the following equation:

$$NetworkReliability = DroppedPackets/(\frac{SimulationTime}{GenerationRate} * Motes).$$

In order to calculate the Network Reliability, a counter is incremented with each packet that is dropped.

5.2.2 Packet Latencies

Another critical analysis metric is the latency of a packet. This is calculated by the Simulator by taking the difference of the time the packet arrives at the AP destination node and the generation time.

$$PacketLatency = APArrivalTime - GenerationTime.$$

Latency analysis is easily accomplished by time-stamping each packet when it is created and taking note of it's arrival time at the AP destination node.

5.2.3 Current Consumption

Since one of our goals for the Simulator is to study battery lifetime, we must also calculate the current consumption for each node. For each scheduled assignment, a node can either be in transmit or receive mode with either active or idle state. Based on typical 802.15.4 radio specifications, the following table outlines the amount of charge burned by each action.

Consumed Charge		
T_i	0 μC	Empty queue
T_x	100 μC	Transmit packet
R_i	25 μC	Listen for packet
R_x	75 μC	Receive packet

For each node, we calculate current consumption by summing all actions and averaging over the time period. The analysis provides us the average current draw for the node.

6 Packet Analysis

For the analysis of the Packet Simulator, we first ran our Network Simulator with the specific scaled-environment conditions of 10,000 sensor motes in our 0.1 km^2 area. Drawing analysis on a reasonable goal of 200 motes/AP, we input 50 as an AP count and chose 10 for a load factor. This produced AP node assignments ranging from 168 to 215 (see the Network Simulator screenshot in Section 7) with 40% and 60% for 1 and 2-hop paths, respectively. At the conclusion of the Network Simulator, outputs of the Superframe schedule and Link-PDR table are exported as files. Having input both as parameters into the Packet Simulator, we run it for a time period of 100 samples or 300 Superframes. Mirroring modern buffer sizes, each node holds a buffer for up to ten packets, dropping generated packets when it becomes full. To put the figures into context, we re-ran the Network Simulator for varying AP counts and analyze the three network metrics from the packet flow in the following sections.

6.1 Latency

An indicator of the success of the end-to-end network components, packet latency is crucial to minimize in order to increase operating efficiencies and decrease response times. Running the Packet Simulator over a series of discrete AP counts (25, 50, 100, 200, and 500 APs), we average the latency in time slots, where each has a duration of 10 ms:

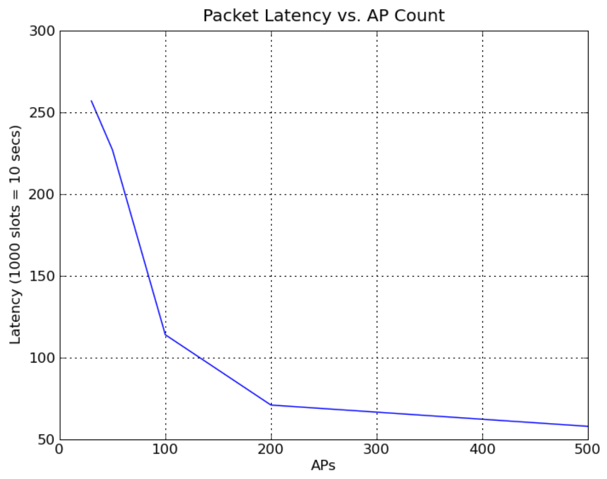


Figure 12: Packet latencies as a function of APs.

Simulated for 300 superframes, the graph shows a worst case of 2.5 s at 25 APs. Reaching an average packet latency of under 1 s for approximately 130 APs, the figure stabilizes beyond 200 APs at 0.5 s latency.

6.2 Reliability

As the goal of each wireless sensor network deployment is data collection. Network reliability is critical in order to ensure that each sample is received and eventually processed at the AP. Observed for the same AP counts as in the earlier experiment, we chart the reliability percent, which is represented by subtracting the drop rate from perfect reliability, 100%:

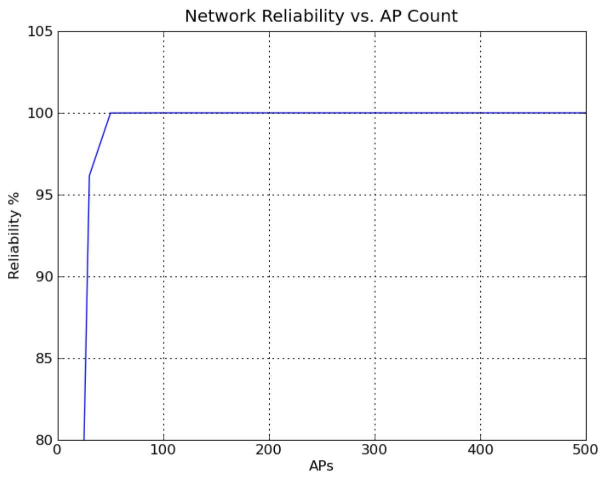


Figure 13: Network reliability as a function of APs.

Simulated for 300 superframes, the figure represents the packet reliability rate given a number of APs in the network. While the network has reliability performance issues at low AP counts, the reliability reaches 96% at 25 nodes and over 99.9% at 50 APs.

6.3 Power Analysis

Finally, the last component of the Flow Analysis is the Current Consumption calculation. As described, we monitor the activity of the packet flow of four states: T_i, T_x, R_i, and R_x. With the target network simulated, we arrive at very high reliability of 99.9% and latency averaged 2.5s with 50 APs. Now, we compute the power analysis for this deployment. Using the consumption table as listed in Section 5.2.3, we sum all power consumed and average over the total simulation time, arriving at the following current distribution:

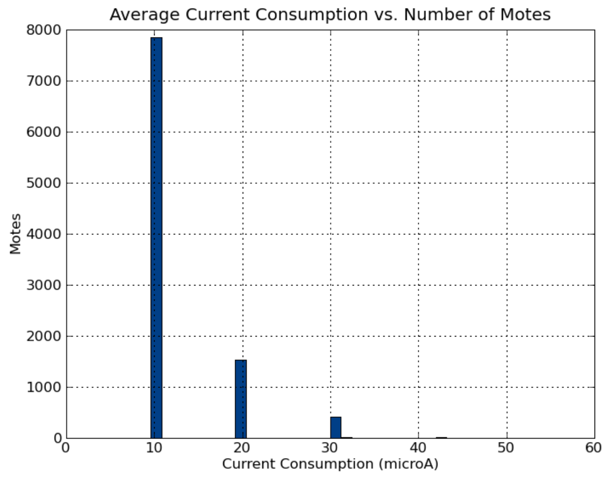


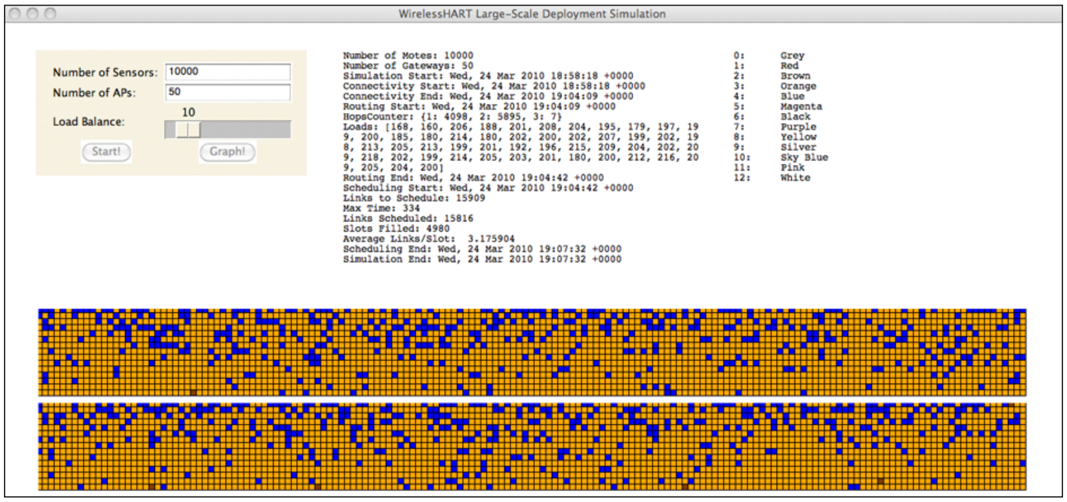
Figure 14: The distribution of power consumption for the simulated network.

Analyzed for all 10,000 motes, the figure represents the average current consumption in microAmps after simulating packets by the proposed schedule from the Network Simulator. As expected, the values are discrete. With almost 80% of nodes at 10 μA consumption, we see that this correlates to successfully transmitting a single packet every ten seconds. Subsequently, the greater currents occur with nodes that require two attempts to transmit, 20 μA, as well as those that not only transmit but also must forward a packet from a child, 30 μA. It is important to note that the analysis also showed that there were a few nodes that were out of range due to routing that bottlenecks at a single parent. Recalling our Routing Module, only APs were load-balanced. Thus, in areas were a hop was needed, the parent node would have an extremely high number of children. We do not display these consumption figures since they skew the analysis but realize is an issue presented by the Routing Module.

7 Screenshots

7.1 Network Simulator

7.1.1 Main Interface



Number of Sensors: 10000
Number of APs: 50
Load Balance: 10

```
Number of Motes: 10000
Number of Gateways: 50
Simulation Start: Wed, 24 Mar 2010 18:58:18 +0000
Connectivity Start: Wed, 24 Mar 2010 18:58:18 +0000
Connectivity End: Wed, 24 Mar 2010 19:04:09 +0000
Routing Start: Wed, 24 Mar 2010 19:04:09 +0000
HopsCounter: {1: 4098, 2: 5895, 3: 7}
Loads: [168, 160, 206, 188, 201, 208, 204, 195, 179, 197, 19
9, 200, 185, 180, 214, 180, 202, 200, 202, 207, 199, 202, 19
8, 213, 205, 213, 199, 201, 192, 196, 215, 209, 204, 202, 20
9, 218, 202, 199, 214, 205, 203, 201, 180, 200, 212, 216, 20
9, 205, 204, 200]
Routing End: Wed, 24 Mar 2010 19:04:42 +0000
Scheduling Start: Wed, 24 Mar 2010 19:04:42 +0000
Links to Schedule: 15909
Max Time: 334
Links Scheduled: 15816
Slots Filled: 4980
Average Links/Slot:  3.175904
Scheduling End: Wed, 24 Mar 2010 19:07:32 +0000
Simulation End: Wed, 24 Mar 2010 19:07:32 +0000
```

```
0:      Grey
1:      Red
2:      Brown
3:      Orange
4:      Blue
5:      Magenta
6:      Black
7:      Purple
8:      Yellow
9:      Silver
10:     Sky Blue
11:     Pink
12:     White
```

Figure 15: Screenshot of the main screen for the Network Simulator.

7.1.2 Network Graph

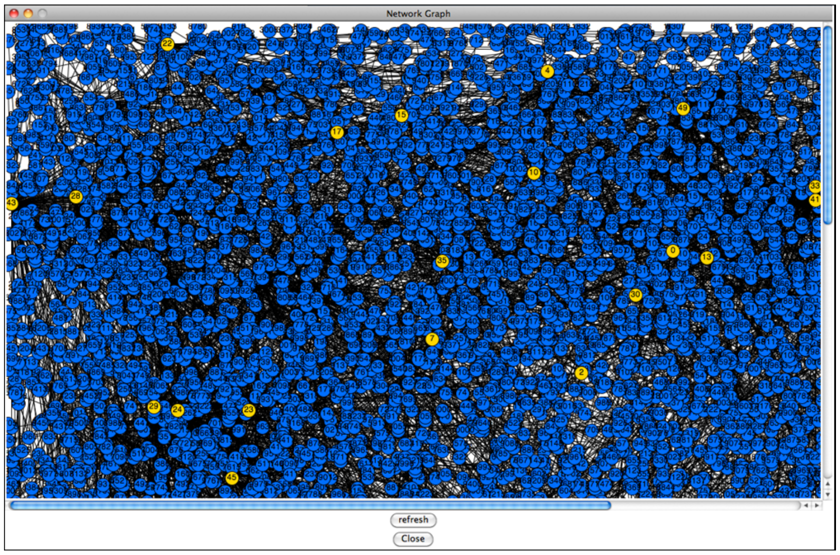
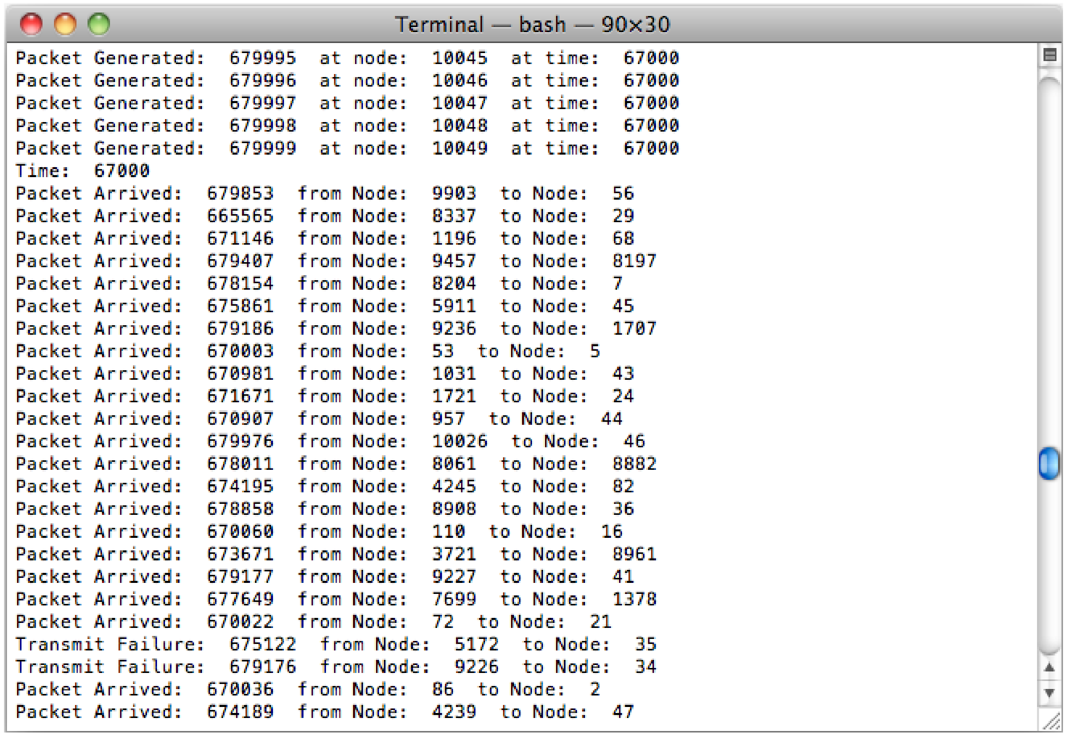


Figure 16: Screenshot of the resulting visual representation of the simulated network.

7.2 Packet Simulator

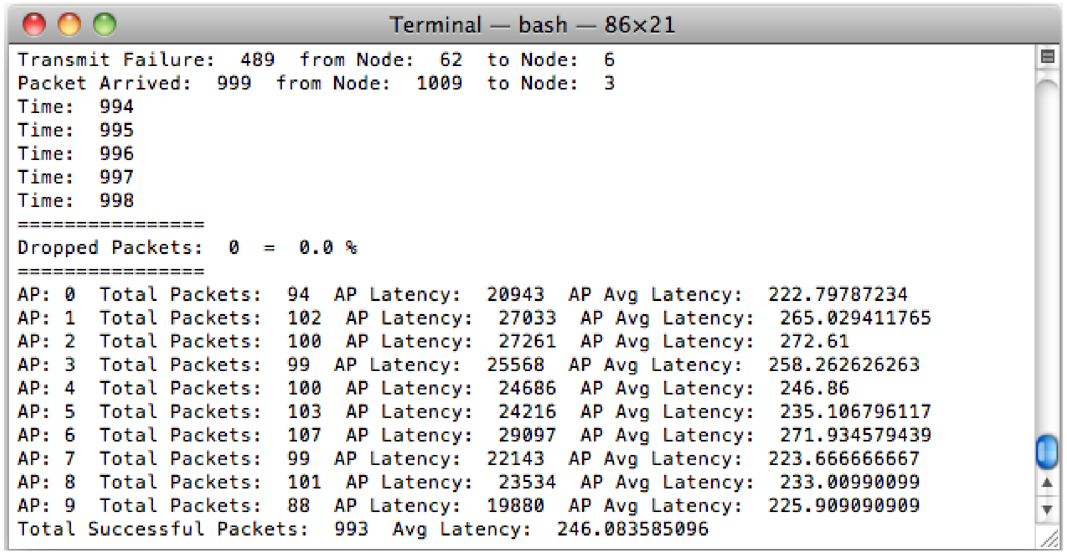
7.2.1 Packet Flows



```
Packet Generated:  679995  at node:  10045  at time:  67000
Packet Generated:  679996  at node:  10046  at time:  67000
Packet Generated:  679997  at node:  10047  at time:  67000
Packet Generated:  679998  at node:  10048  at time:  67000
Packet Generated:  679999  at node:  10049  at time:  67000
Time:  67000
Packet Arrived:  679853  from Node:  9903  to Node:  56
Packet Arrived:  665565  from Node:  8337  to Node:  29
Packet Arrived:  671146  from Node:  1196  to Node:  68
Packet Arrived:  679407  from Node:  9457  to Node:  8197
Packet Arrived:  678154  from Node:  8204  to Node:  7
Packet Arrived:  675861  from Node:  5911  to Node:  45
Packet Arrived:  679186  from Node:  9236  to Node:  1707
Packet Arrived:  670003  from Node:  53  to Node:  5
Packet Arrived:  670981  from Node:  1031  to Node:  43
Packet Arrived:  671671  from Node:  1721  to Node:  24
Packet Arrived:  670907  from Node:  957  to Node:  44
Packet Arrived:  679976  from Node:  10026  to Node:  46
Packet Arrived:  678011  from Node:  8061  to Node:  8882
Packet Arrived:  674195  from Node:  4245  to Node:  82
Packet Arrived:  678858  from Node:  8908  to Node:  36
Packet Arrived:  670060  from Node:  110  to Node:  16
Packet Arrived:  673671  from Node:  3721  to Node:  8961
Packet Arrived:  679177  from Node:  9227  to Node:  41
Packet Arrived:  677649  from Node:  7699  to Node:  1378
Packet Arrived:  670022  from Node:  72  to Node:  21
Transmit Failure:  675122  from Node:  5172  to Node:  35
Transmit Failure:  679176  from Node:  9226  to Node:  34
Packet Arrived:  670036  from Node:  86  to Node:  2
Packet Arrived:  674189  from Node:  4239  to Node:  47
```

Figure 17: Screenshot of the packet transmission for the Packet Simulator.

7.2.2 Summary



```
Transmit Failure:  489  from Node:  62  to Node:  6
Packet Arrived:  999  from Node:  1009  to Node:  3
Time:  994
Time:  995
Time:  996
Time:  997
Time:  998
================
Dropped Packets:  0  =  0.0 %
================
AP:  0  Total Packets:  94  AP Latency:  20943  AP Avg Latency:  222.79787234
AP:  1  Total Packets:  102  AP Latency:  27033  AP Avg Latency:  265.029411765
AP:  2  Total Packets:  100  AP Latency:  27261  AP Avg Latency:  272.61
AP:  3  Total Packets:  99  AP Latency:  25568  AP Avg Latency:  258.262626263
AP:  4  Total Packets:  100  AP Latency:  24686  AP Avg Latency:  246.86
AP:  5  Total Packets:  103  AP Latency:  24216  AP Avg Latency:  235.106796117
AP:  6  Total Packets:  107  AP Latency:  29097  AP Avg Latency:  271.934579439
AP:  7  Total Packets:  99  AP Latency:  22143  AP Avg Latency:  223.666666667
AP:  8  Total Packets:  101  AP Latency:  23534  AP Avg Latency:  233.00990099
AP:  9  Total Packets:  88  AP Latency:  19880  AP Avg Latency:  225.909090909
Total Successful Packets:  993  Avg Latency:  246.083585096
```

Figure 18: Screenshot of the packet summary of network latency and reliability.

8 Conclusions and Future Work

In this project, we have set out to both build a framework for the exploration of the challenges in scaling and deploying large wireless sensor networks. Specifically, our goal entails a target network of 10^6 motes in an area of 10 km^2. Due to computing limitations, the Simulators have been adjusted to study at the same density, yet scaled to 10^3 motes in a square, with a side length of 316 m. The work creates end-to-end Network Simulation and Packet Simulation tools for analyzing the network deployment in oil refineries.

As we simulate, we leverage the potential of WirelessHART as the first wireless technology protocol designed for industrial automation. Since the application requires highly dense large-area networks, WirelessHART must meet the criteria of Reliability, Low Power, Scalability, Flexibility, and Security. While each of these criteria is crucial to the long term success of WirelessHART, we measure the reliability, latency, and power consumption over the network's nodes in varying AP distributions.

To realistically simulate the entire network, we create a series of Modules for both Simulators including Positioning, Connectivity, Routing, Scheduling, and Packet Flow. Remaining compliant to WirelessHART, we developed simple algorithms that match the needs of the application. In Routing, a modified Dijkstra Least-Cost with load-balancing algorithm allowed us to minimize hop counts while balancing mote assignments among APs. In Scheduling, our Layered Scheduler provided row-major assignment to evenly distribute links in cells and guaranteeing for collision avoidance. With any Module in the framework, we can replace it with either a separate algorithm or insert real-life data for greater accuracy.

Revisiting our target and deployment challenge, we find that a million mote sensor network is potentially possible with just 5,000 APs (as shown by our 1% scaled simulation with 50 APs). With 80% PDR, the expected packet latency is roughly 2.25 s and reliability matches the 99.9% goal. This was confirmed by our simple Simulator and while it is an end-to-end solution, there are many areas that we can optimize upon. To achieve greatest effect, we can add greater sophistication in the Routing Algorithm and Scheduler Algorithm. Mentioned before, greater diversity in path and assignment would enhance packet flow.

Although this is the first step in producing a Simulator, we admit there are areas for improvement and additional work:

1. Full Scale Simulation - Expanding from 1% to 100% in area and devices.
2. Variable Generation Rates - Increasing variability of generation rates beyond the 10 s constant.
3. Time-Varying Network Conditions - Accounting for potential failures or external interference from WiFi or Bluetooth.

Additional Management Requirements:

4. Redundancy - Adding path diversity to ensure delivery and testing links.
5. Power Aware Routing - Selecting routes to avoid nodes with low remaining power.
6. Network Maintenance - Scheduling health reports.
7. Downstream Cell Allocations - Assigning links to the schedule from AP to nodes.

Continuing to pursue with these remaining elements of the end-to-end network elements will further strengthen the Simulator and prepare us for real-life deployment of highly dense large-area wireless sensor networks.

References

[1] Adler, R., Buonadonna, P., Chhabra, J., Flanigan, M., Krishnamurthy, L., Kushalnagar, N., Nachman, L. and Yarvis, M. (2005), 'Design and deployment of industrial sensor networks: experiences from North Sea and a semiconductor plant', *Proceedings of ACM Sensys 2005*, November.

[2] Cisco Systems Inc. (2008), 'Cisco Secure Wireless Plant: Security and Quality of Service for Industrial Environments', `http://www.cisco.com/web/strategy/docs/manufacturing/SWP_Industrial_Environments.pdf`.

[3] Crossbow Technology Inc. (2004) 'Avoiding RF interference betweenWiFi and Zigbee', `http://www.xbow.com/Support/Support_pdf_files/ZigBeeandWiFiInterference.pdf`.

[4] Gnawali, O., Yarvis, M., Heidemann, J., and Govindan, R., 'Interaction of Retransmission, Blacklisting, and Routing Metrics for Reliability in Sensor Network Routing'. In *Proceedings of the First IEEE Conference on Sensor and Adhoc Communication and Networks*, pp. 34-43. Santa Clara, California, USA, IEEE. October, 2004.

[5] HART Communication Foundation. `http://www.hartcomm2.org/index.html`.

[6] Lanzisera, S., Mehta, A. M., 'Reducing Average Power in Wireless Sensor Networks Through Data Rate Adaptation', *Proc. IEEE International Conference on Communications, Jun. 2009.*

[7] Musaloiu-E., R. and Terzis, A. (2008) 'Minimising the effect of WiFi interference in 802.15.4 wireless sensor networks', *Int. J. SensorNetworks*, Vol. 3, No. 1, pp.43-54.

[8] Pister, K.S.J., Doherty, L. (2008), 'TSMP: Time Synchronized Mesh Protocol', *Proceedings of the IASTED International Symposium DSN 2008)*, November.

[9] Pister, K.S.J.,'Pister Hack Model' *EE290Q-2, UC Berkeley, February 17, 2009.*

[10] Pister, K.S.J.,'Wireless Sensor Networks' *EE290Q-2, UC Berkeley, May 7, 2009.*

[11] Song, J., Han, S., Mok, A.K., Chen, D., Lucas, M., Nixon, M., and Pratt, W. 2008. ''*WirelessHART*: Applying Wireless Technology in Real-time Industrial Process Control'. Real-Time Technology and Applications Symposium (Apr 2008), 377-386.

[12] Kinney, P., 'ZigBee Technology: Wireless Control that Simply Works', *Communications Design Conference 2003)*, October.

[13] Bluetooth SIG, *Specification of the Bluetooth System*, November 4, 2004.

[14] LAN/MAN Standards Committee of the IEEE Computer Society, *IEEE Std 802.11-2007*, September 16, 1999.

[15] IEEE Standard for Information Technology, *IEEE Std 802.15.1-2002*, June 14, 2002.