Winter 2019

Mesh node communication system for fire figthers

Eric E. Hamke
University of New Mexico, ehamke@unm.edu

Trace Norris
University of New Mexico

Jessica E. Ladd
University of New Mexico

Jeffrey K. Eaton
University of New Mexico

Jicard J. Malveaux
University of New Mexico

See next page for additional authors

Authors

Eric E. Hamke, Trace Norris, Jessica E. Ladd, Jeffrey K. Eaton, Jicard J. Malveaux, Manish Bhattarai, Ramiro Jordan, and Manel Martinez-Ramon

Mesh node communication system for fire figthers

Eric E. Hamke, Trace Norris, Jessica E. Ladd,
Jeffrey K. Eaton , Jicard J. Malveaux, Manish Bhattarai
Ramiro Jordan, Manel Martínez-Ramón
Department of Electrical and Computer Engineering
The University of New Mexico

October, 2019

Contents

1 Introduction

This report describes the prototype and demonstration of a reliable local area network (LAN) of devices able to interconnect a crew of fire fighters with a communications node. The data consists of images for thermal cameras at a given rate and other low bandwith data taken from the different sensors of the fire fighter gear, that are connected to the transmitter using a Bluetooth personal area network (PAN). The physical environment is indoors and it is expected that all nodes of the communication networks are distributed in different positions into a building. The implemented network is schematically depicted in Figure 1.

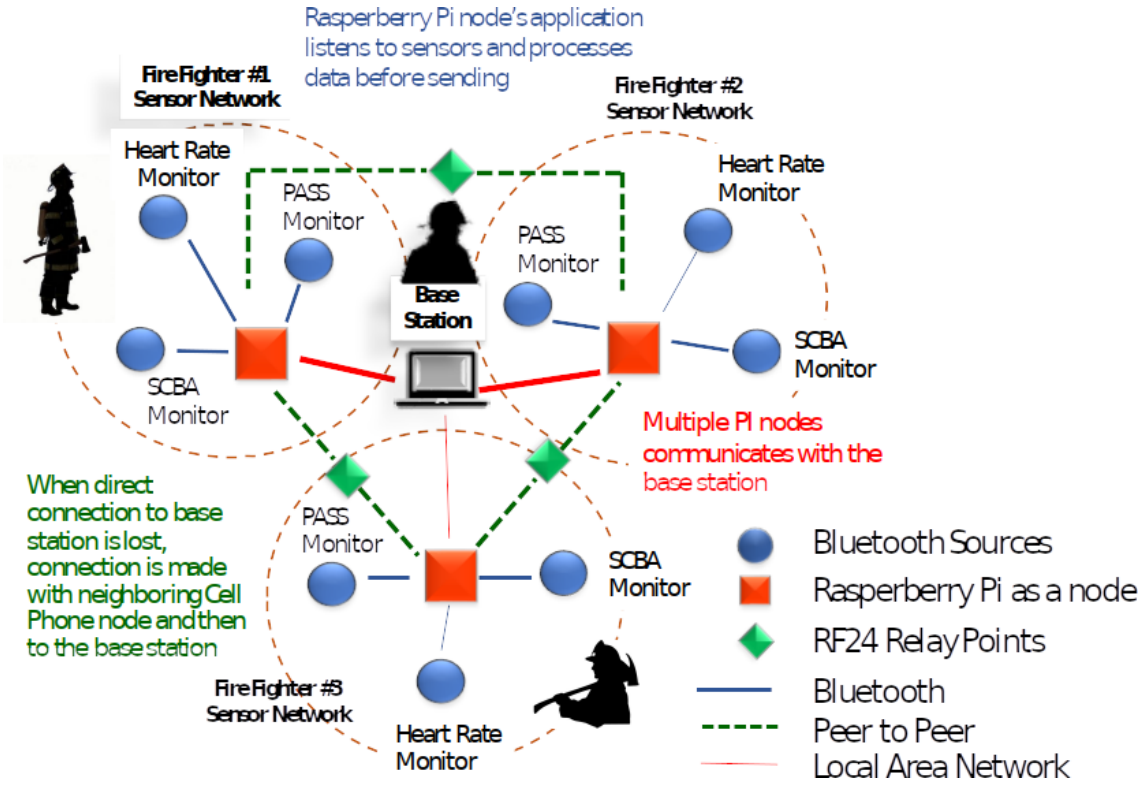

Figure 1: System schematics of the deployed fire fighter communications prototype. Blue dots are the different sensors including a SCBA monitor and others. They are copnnected via Bluetooth to the single board computers (red squares), which connect with the resto of boards using a radio module. Thermal cameras are assumed to be wired to the board. The green diamonds are single board computers endowed with a radio that act as additional relay nodes and they can be left in strategic positions.

The purpose of this system is to manage and transmit relevant data to a node in order to create a situational awareness environment based on image and signal processing and machine learning. The multiple layers of communication devices in the system are designed to ensure robustness of data transmission. The first physical layer of devices is attached to the fire fighters. Each one has a single board computer that communicates with the sensors of the fire fighter using Bluetooth. This is the PAN. The single board computer used in the demonstration is simple Raspberry PI, which is also connected to a radio device type RF24L01+. These radios, interconnected together are the physical layer to the LAN, and

Figure 2: Hardware use for the PAN emulation.

they create a mesh network that communicates with a base station where data is logged and processed. Additionally, existing relay points constructed with additional Raspberry Pi boards endowed with an RF24L01+ radio can communicate with the rest of nodes in order to complete the mesh network. All radios are configured in broadcasting mode. In this study it is assumed that not all receivers can listen to all broadcasts because of the inconsistent nature of indoor scenarios, where some nodes can be totally or partially shielded by walls or obstacles. The system is therefore intended to ensure that all data broadcast by any of the nodes is able to reach the hub through mesh transmission. We believe the study should have 2 components. The first component involves determining a threshold of network performance in support of the image processing algorithms and to determine what kind of behaviours the network needs to have to adapt to the standard.

The first part can be performed by corrupting an image stream with a loss data packets and then assessing the image processing algorithm's ability to perform its tasks.

The second part is to explore the application of machine learning to create an adaptive network. The Adaptations focus on trying to optimize the network's connectivity or preserving bandwidth. We will develop a simulation of both behaviours and with the intent of determining the best approach to meet the needs of the network. The simulations will be based on the python library http://mininet.org/. We would use actual data from the performance characteristics of the current implementations of the Raspberry Pi Personal Area Network interface and the Raspberry Pi relay node performance to provide the statistics needed to characterize the nodes in the simulations.

The following sections describe the organization and structure of the network; address the use of node behaviours, and possible study topics.

2 Physical elements

In order to reproduce the production of data coming from a PAN, an emulator has been constructed that uses a Bluetooth transmitter, an Arduino UNO microcontroller and a small battery. The elements are illustrated in Figure 2.

The Bluetooth has been setup so the Bluetooth frame (Figure 3) carries all the needed information. In particular, a frame contains allocations for the time stamp, heart rate, skin

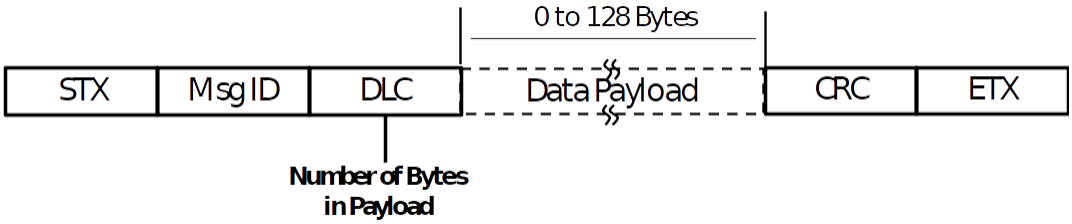

Figure 3: Bluetooth frame

temperature, posture, activity level, battery level, breathing wave amplitude, ECG noise, acceleration (to plug an IMU device) and others. The details of the frame are included in Table **??** in the Appendix.

The STX field (Start of Text) is a standard ASCII control character (0x02) and denotes the start of the message. The Message ID uniquely identifies each message and is in binary format. A response message uses the same Message ID as the corresponding request message. The Data Length Code (DLC) is used to specify the number of bytes within the data payload field of the message. Valid values range between zero and 128 (inclusive). Payload Data Field contains the actual data sent between the local and remote units and can contain anywhere between zero and 128 bytes of data. The number of bytes in this field is dictated by the DLC field. An 8-bit CRC is used and has a polynomial of 0x8C (CRC-8), with the accumulator being initialised to zero before the CRC calculation begins. The ETX field (End of Text) is a standard ASCII control character (0x03) and denotes the end of the message. Note that although this is the last field within the request message, the response message has no ETX value, instead using an ACK/NAK in its place.

The LAN network consists of a set of nodes carried by the fire fighters, which are connected to the PANS and that use a radio module that broadcasts the packets to other nodes, which are configured as a mesh structure. Some nodes are set up as standalone relay points to extend the number of nodes of the network. These relay points are intended to be dropped at specific positions in the fireground.

The nodes and relay points consists of a radio module (model RF24L1+) connectad to a Raspberry PI. The set is powered with a battery pack (Figure 4). The devices emulating the relays and nodes are therefore programmable and they host the ad-hoc network software that controls the mesh. The software embedded in these nodes and relay points include a means to record the time of packet reception and retransmission of the packages as well as the number of received and retransmitted packages for analysis and evaluation.

The Bluetooth® communication interface shall conform to the BioHarness Bluetooth Comms Link Specification 2014-MAY-01, BioHarness BT Android API User Guide 2012-03-27. The Bluetooth® data in the system shall be in conformance to the BioHarness 3 Logging System Interface 2012-07-12. Raspberry Pi Node software shall support the transmission of a standard set of image files (IR Camera Images at 100x200pixels). Each transmitted image shall have a CRC included in the file image transmission.

The mesh is connected to a base station emulator, that consists of a desktop computer connected to a radio module through a simple Arduino UNO microcontroller that establishes a serial communication between the radio and the computer. The base station records all the

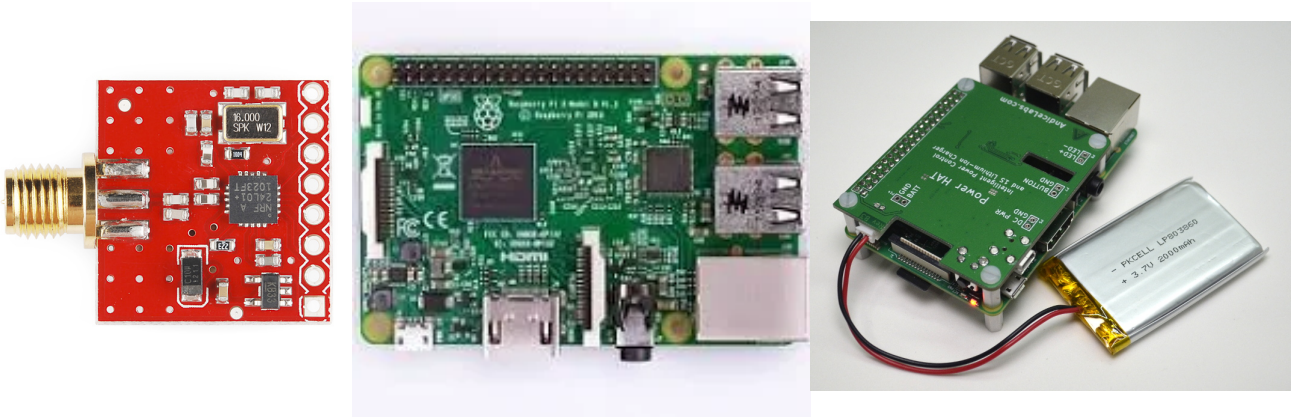

Figure 4: Hardware used for the fire fighter nodes and the relay points to be placed as standalone transceivers.

data received fro the radio through the serial interface. The Base Station Emulator software includes the computation of the number of sensor data packets corrupted versus those sent.

3 Network Description

The RF24L01+ radios can be organized into a mesh network. The network layer takes advantage of the fundamental capability of the nRF24L01+ radio to listen actively to up to 6 other radios at once. The network is arranged in a Tree Topology (Figure 5), where one node is the base (Base Station), and all other nodes are children either of Base Station (Relay Points), or of another node such as children of the relay points (Personal Area Networks (PAN)). The maximum connectivity of the network is depicted in Figure 6, where it is assumed that all nodes and relays can communicate with eacho other.

Each node must be assigned an address by the administrator, consisted of a n octal number. This address exactly describes the position of the node within the tree. Each digit in the address represents a position in the tree further from the base. [1] An example of address set is given below:

- Node 00: base station.
- Nodes 01-03: relay points or nodes whose parent is the base station (Relay Point # 1, Relay Point # 2, and Relay Point # 3, respectively).
- Node 011: PAN # 1 node since it is the first child of the Relay Point # 1,
- Node 012: PAN # 2 node since it is the first child of the Relay Point # 2, and
- Node 013: PAN # 3 node since it is the first child of the Relay Point # 3.

The largest node address is 055, so 45 nodes are allowed on a single channel. The maximum connectivity of the topology is shown in Figure 6, with 5 nodes in direct communication with the Base Station, and multiple leaf nodes spread out at a distance, using intermediate

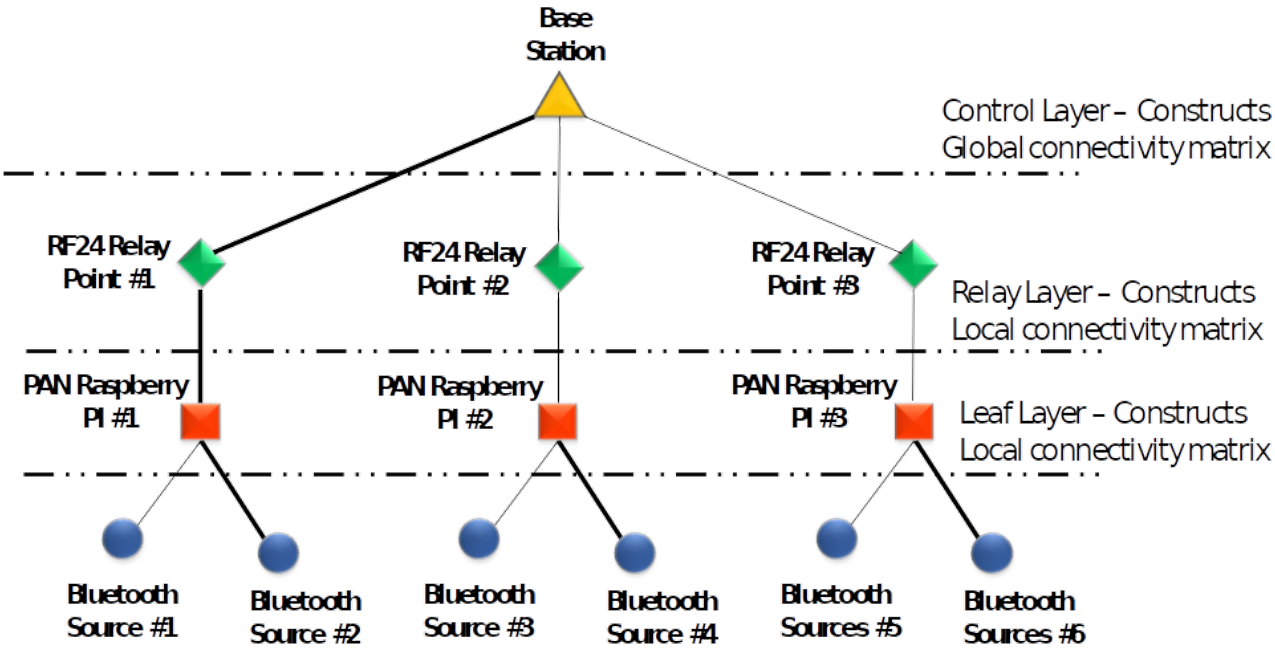

Figure 5: Example of data flow diagram of the mesh network

nodes to reach other nodes. This shows the flexibility of the topology that allows for the movement of data from any leaf (PAN # 1, PAN # 2, and PAN # 3) to the Base Station through multiple intermediate nodes. Note that the base station has connectivity to all nodes in the network. This is represented as a six-sided star with nodes at points.

The yellow triangle (base station) connected with all nodes and relays (represented by the continuous lines) represents the best communications case, where the PANs talk directly to the base station, but this is not what is to be expected in a fireground scenario, where many of the channels may be jeopardized by physical obstacles. To deal with this circumstance, we created a secondary network between relays and nodes. It is intended that the PAN Raspberry Pi route its traffic to a relay node and then on to the base station. We also anticipate that when multiple firefighters are in a room, it may be possible for the PAN Pi's to route traffic between themselves and out to a relay node or the base station.

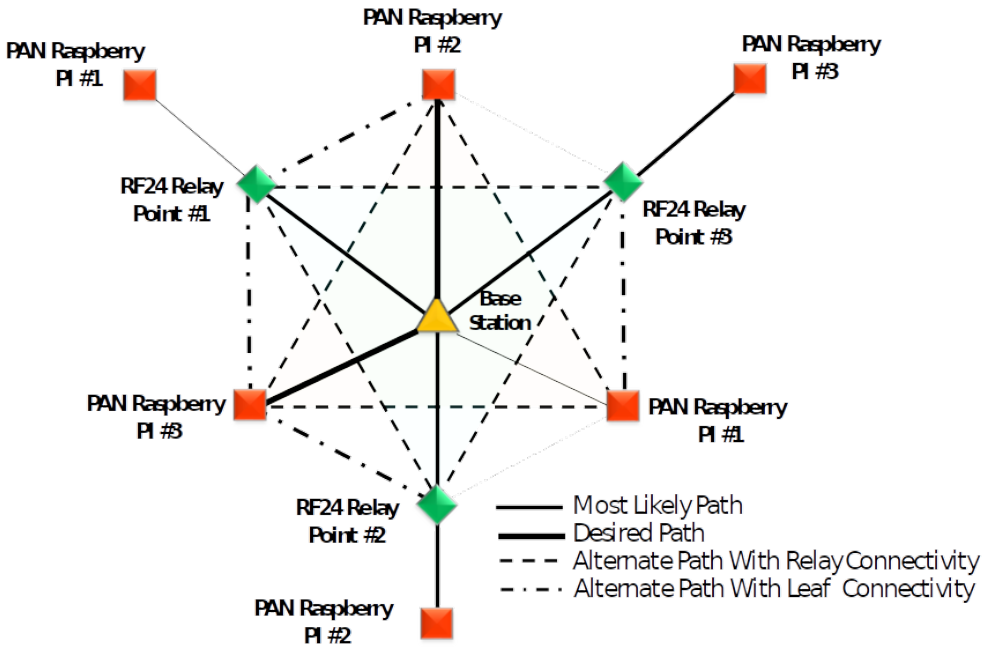

Figure 6: Maximum Interconnectivity Schematic – Note multiple instances of PAN nodes were used to simplify the drawing. So, all nodes labelled PAN Raspberry PI # 1 are the same physical location.

3.1 Node Architecture

The communication node structure is shown in Figure 7. The system requires the use of a supervisor application that monitors the different elements in a communication node. If one of the components experiences a failure such as a radio going offline or a time out failure between the radio and the Transmission Communication Protocol (TCP) server, the supervisor will provide the necessary system recovery actions.

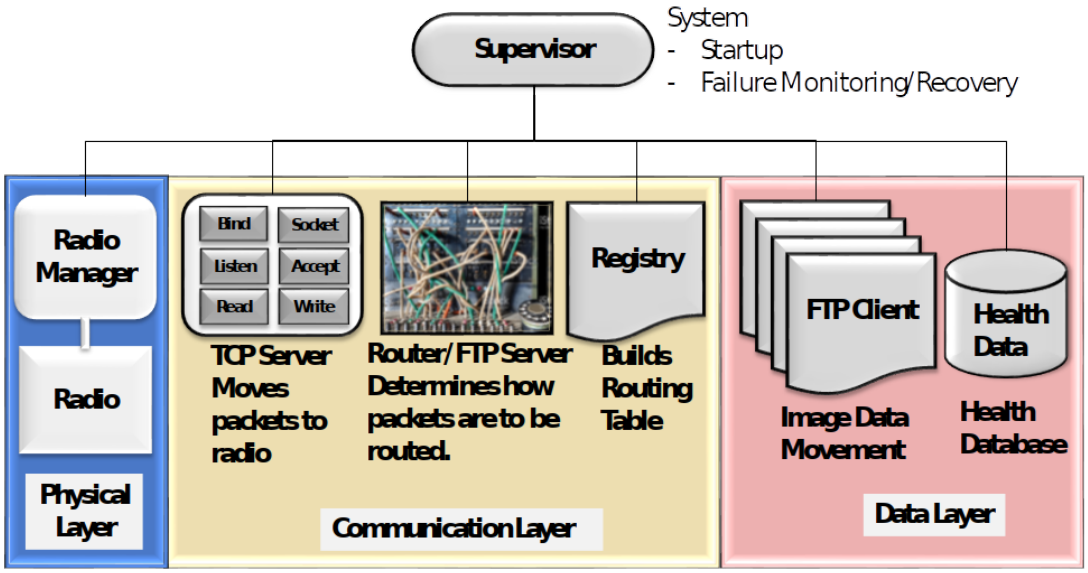

Figure 7: Figure 3. Communication node structure.

The network is implemented using the Elixir/Erlang programming language [1], which is designed to create communication networks with fault tolerant behavior. Elixir is a dynamic, functional language designed for building scalable and maintainable applications. It leverages the Erlang Virtual Machine (see e.g. [2]), known for running low-latency, distributed and fault-tolerant systems, while also being successfully used in web development and the embedded software domain.

One of the key features of the Erlang/Elixir interface is the ability to build supervisory structures (Figure 8) independent of the applications being supervised. The node supervisor monitors the supervisors for each of the applications. If a supervisor fails, the node supervisor will initiate a recovery process for that application. In this way we can manage application failures that impact the other applications. Similarly, each application supervisor monitors its application and the failure recovery processes needed to restart each application. In the case of the radio interface the supervisor will also restart the radio as needed.

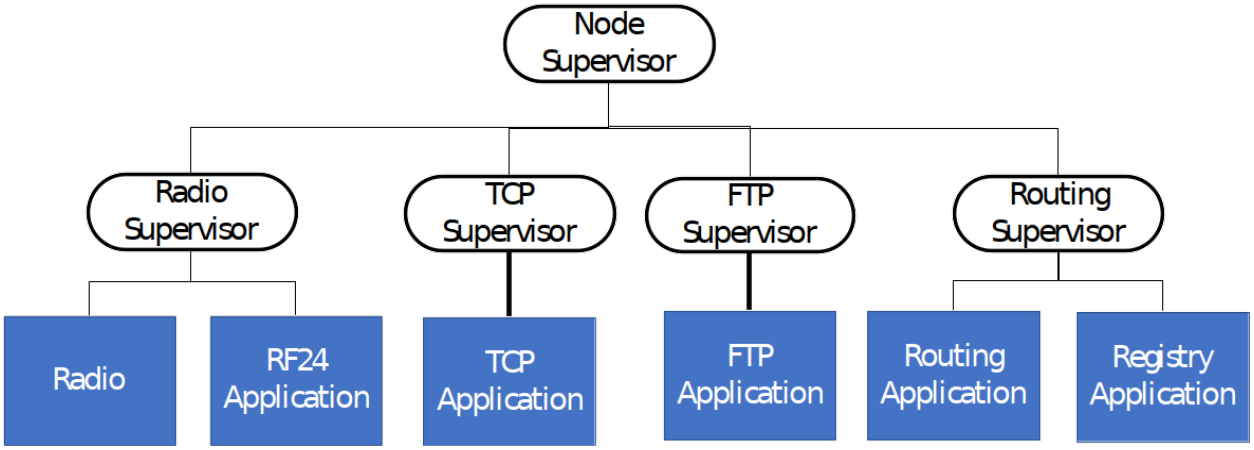

Figure 8: Node supervisor structure.

The Radio Manager application manages the radio hardware to include configuring the radio, monitoring its status and placing the radio in Transmit or Receive mode. The configuration includes current radio channel, data rate, Cycle Redundancy Check (CRC) size, and transmit power setting. The TCP server manages the transmission and receipt of packets. The interface between the Radio Manager and the server will support sending one packet at a time and the receipt of packets. The TCP server maintains and manages queues of packets awaiting transmission and receipt processing/rerouting. The Radio Interface has been implemented in Python and C-code. These applications will be incorporated using NIFs (Natively implemented functions).

3.2 Registry

The registry application maintains a connectivity matrix for the Router application. Upon startup of the application, it will generate a system ping message. The router will then monitor the responses (pongs) to build the matrix. After startup, the registry application monitors the headers from all incoming packets and records the From node and Last node information in a connectivity matrix structure. Each connection also has a latency timer. At the expiration of the timer the entry in the table will be reset to 0 indicating the link is no longer in use. Another factor we need to address is answering the question, "If a node starts leaning heavily on path B vs C or D, how is path C or D updated given that it is not sending enough (or any) traffic their way?"

We have considered two approaches for dealing with a silent interval. In the *active approach*, the base station sends out a heartbeat every few seconds to the layer one nodes (see Figure 5). This heartbeat cascades down the network with individual nodes updating their transmission probabilities as they receive their parent nodes heartbeat. This approach keeps very up to date transmission probabilities and connection states, but it increases the network traffic. The alternative approach is a *passive mode*. In this approach, if a node has not received packages from a given node in a time interval, it sends out a ping for that node, and it updates the connectivity matrix as 1 if it receives a pong or 0 otherwise. This

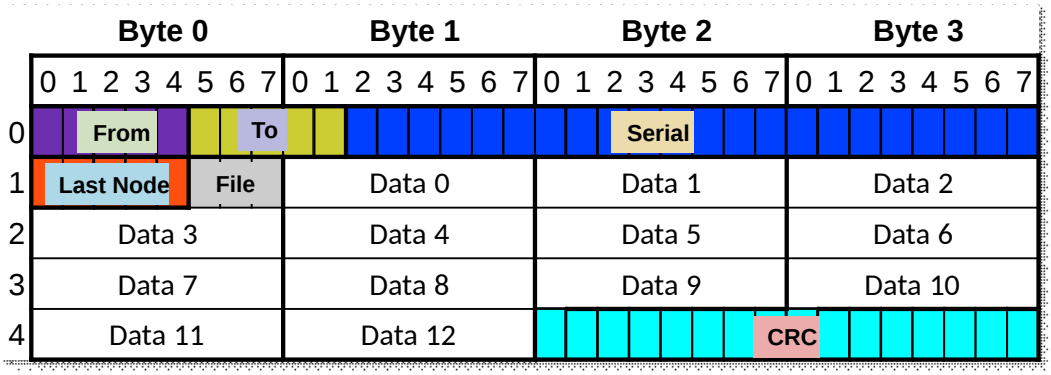

Figure 9: Health Information packet structure.

is a minimal traffic approach, but it may potentially lead to outdated routing options and it could lead to excess transmission failure and re-tries.

Each node will assess its connectivity using a short time interval. A node is considered connected to another node if during the time interval it receives a message from the other node. The connectivity probabilities are updated using a longer sliding time-window. The ratio computed as the number of intervals with a received message divided by the number of intervals in the sliding window. The Router/File Transfer Protocol application manages the generation of packets.

3.3 Packet Structures

There are three types of data provided in the network. They are

1. Health telemetry (Figure 9) usually contained in a single packet;
2. Back channel coordination traffic (Figure 10) as the active heartbeat or passive inquiry to determine the mesh connectivity;
3. The movement of video data from the IR cameras (Figure 11). These images will consist of a series of 20 byte elements or file chunks. Assuming the image has 3.4Kbytes this means an uncompressed image will use 20 chunks. This leads to the question of network performance such as how many of these packets make it to the destination in a timely and consistent manner.

RF24 Radios supports the transmission of 32-byte packets. The fields in the packet are as follows

- From: 5-bit field with unique radio ID – implied IP Address 127.0.0.<radio id>
- To: 5-bit field with unique radio ID – If connected to base station this is base station address otherwise the intermediate relay point address.

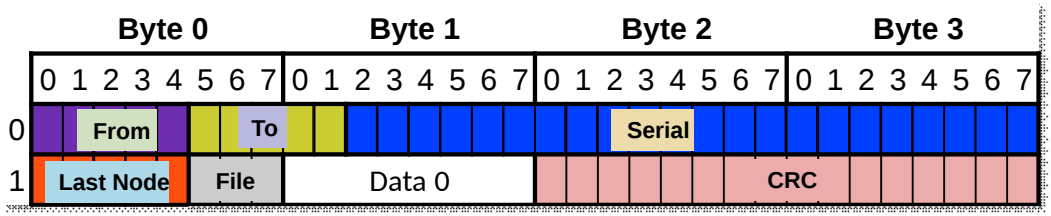

Figure 10: Ping/Pong packet structure for back channel coordination. Note Data 0 byte could be used to identify pong or ping message type.

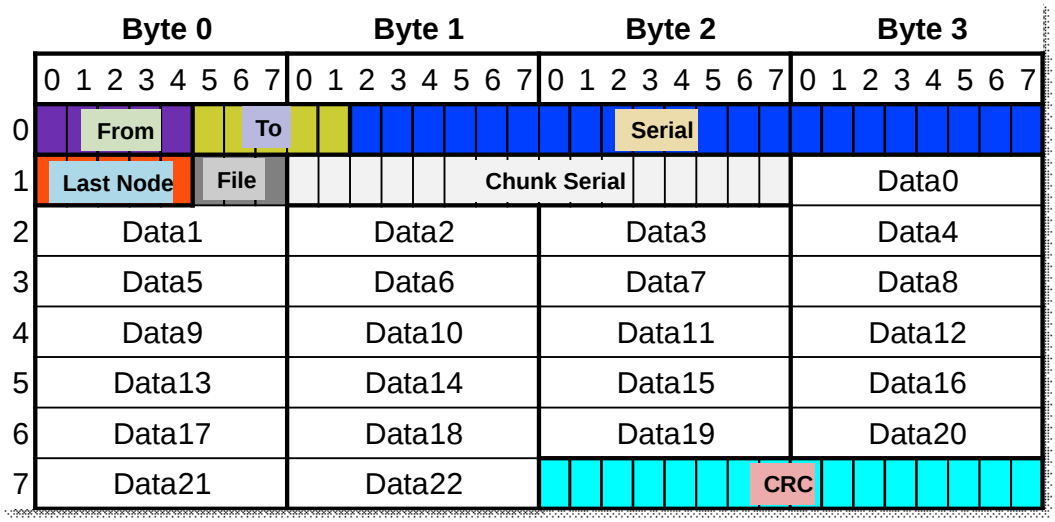

Figure 11: File Transfer Protocol packet structure.

- Last Node: 5-bit field of last node to transmit package (used to support building routing tables; may be different from source node if packet routed through alternative nodes to Base Station)
- File: This is used to indicate the first file chunk (000), intermediate file chunk (001), last file chunk (010) from File Transfer Protocol Client application, health record (011), system ping (100), and system pong (101).
- Chunk Serial – used to serialize the chunks being sent and received should an interruption of the file transfer occur.
- Serial: Is a unique number in the range from 0 to 2^{22} used to determine missing packets and reassembling files sent using FTP server. Each Node will maintain its own serial counter.
- Data Bytes – These will be filled in based on data source. The FTP server will send 24-byte chunks. A health record should consist of 12 bytes. The Data byte is unused in the Ping/Pong packet but is there for future growth.
- RF24 has built-in CRC generation functionality.

With the use of a serialized packet structure for sending video image data, it is necessary to consider consolidation of packages into an image. Given the potential for packet loss when transmitting an image, and the desire to minimize network traffic an end of file signature is needed as well as a total file size. This way the receiver can determine what packets are received and which ones are lost, and request them.

Assuming the probabilistic nature of routing methodology, we would need to determine how long we should consider data to be relevant. A packet or set of packets could be routed to node where the connection is lost for some time interval. When the connection is restored, the contents of the queue are serviced. The packets waiting in the queue reflect data that may not be relevant to the current image being sent and may wind up consuming transmission time that could have been used to send packets related to the current time frame. To deal with this, packets are given a life span. When a packets life span expires, it is removed from the queue.

3.4 Routing Function

To understand how registry application's connectivity matrix is used, let us suppose that we are sending data from PAN # 3. The PAN # 3 column indicates we are connected to PAN # 1 and PAN # 2. The PAN # 3 row has a connection to Relay Point # 1. The PAN # 3's routing application could then send the data packets to PAN # 1, PAN # 2, and Relay Point # 1. The choice of a destination is based on the connectivity of the destination node. Looking at PAN # 1, is not connected directly to the Base Station. This means the message would be forced to hop to an intermediate destination before reaching the base station. Examining PAN # 2 connections shows a direct connection to the Base Station. This is better since the message will spend less time in network. Relay Point # 1 has no direct

correction to the Base Station. Like PAN # 1, it will have to be sent on to an intermediate node before reaching the Base Station. Ideally, the best destination is PAN # 2.

The network topology is such that we expect it to change over time. As the topology changes or evolves over time, it will be necessary to re-evaluate that route to the base station at each intermediate node. The primary motivating factor is to reroute packets around links no longer present or to take advantage of new routes that may not available when the packet entered the network. The evolution can be captured using several approaches.

The first approach discussed involved using a connectivity matrix where the entries designate if a node has a connection or not. Instead of just looking at basic connectivity (i. e.in Figure 12, X is 2 hops from the base station, Z and W are both 2 hops, so the solution send to Z or W).

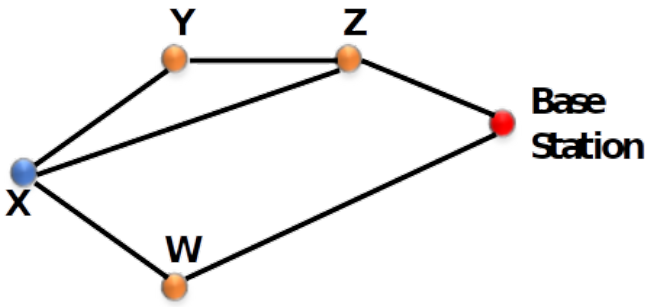

Figure 12: Typical network topology.

It is possible that when the packet reaches node Z (the first hop from X), node Z's connection to the base station is lost (Figure 13).

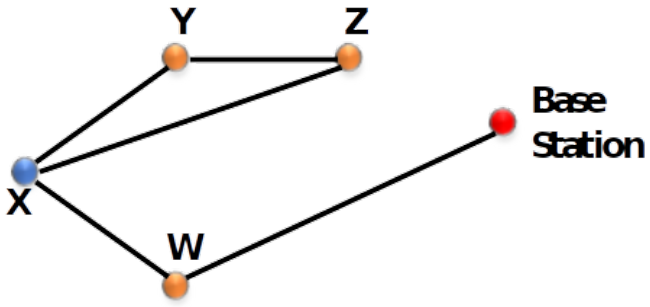

Figure 13: Typical degraded network topology.

Reevaluating the route, the one with the least number of hops is Z, X and W. In this case sending the packet back to X. This backtracking or circular routing behaviour can be checked for given the current connection table but cannot be anticipated.

One problem with this approach is that there is no measure of likelihood that with the connections will be there to complete the routing or path. An initial attempt at developing a

soft measure would be the ratio of the time the connection is present over a fixed time window. With the connection probabilities, it would be possible to perform a maximum likelihood calculation of the possibility of the packet reaching its destination (the base station) over the routing choices.

As an example, nodes would provide a probability that a packet sent from them could successfully reach the next node. In a more abstract sense, given the current network topology it is possible to compute a path probability as shown in Figure 14 . Note that the first hop for any path is assigned a probability of 1 given that that connection is active at the time of transmission. Path 1 would have a probability of 0.096. This path involves sending the packet to node Y and then to Z. Now suppose we are in touch with node Z. The probably would be 0.06. Finally, examining path 3 we would have a probability of 0.0375. The path through Y would be the most likely path.

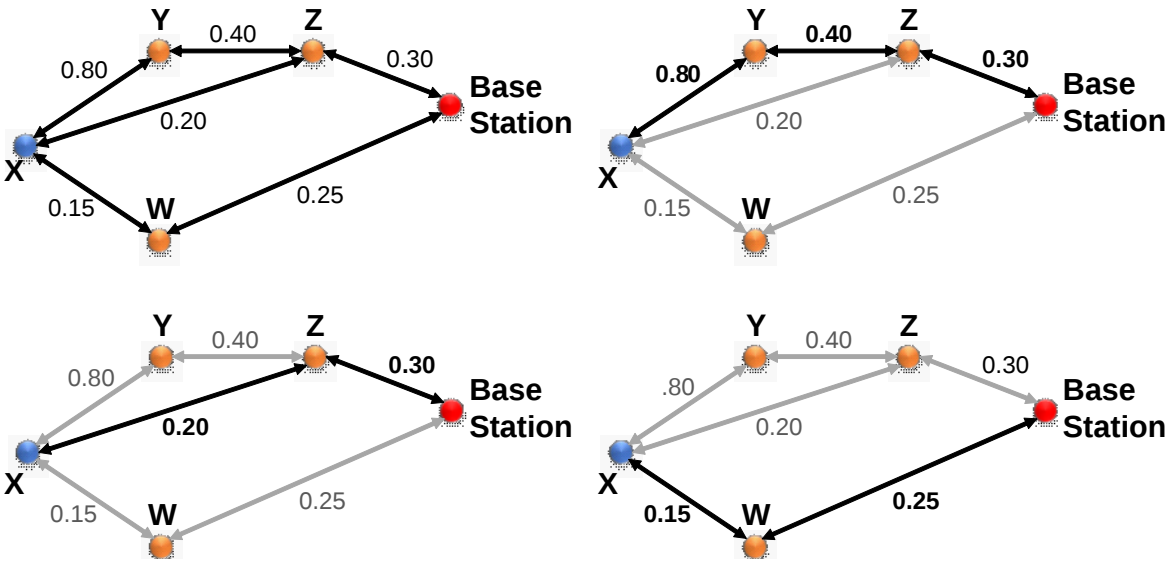

Figure 14: Using probability metadata.

In calculating the expected transit times in the network, it may be best to evaluate an optimistic route through the network and a pessimistic route. The actual time through the network is bounded by these times. The optimistic time is based on the expectation calculation with a minimum wait time and the probability the link is available. The pessimistic time is the time assuming a maximum weight time the probability that the link is not available.

Additional metadata such as an average node throughput time (Figure 15) can be used to calculate an expected time to not reach the base station. Computing the expected value for the first path is (1-0.80)10+(1-0.40)15+(1-0.3)6 or 15μs; path 2 has a value of 10.6μs and the last path time is 17.45μs. So maybe we would send the packet to node Z.

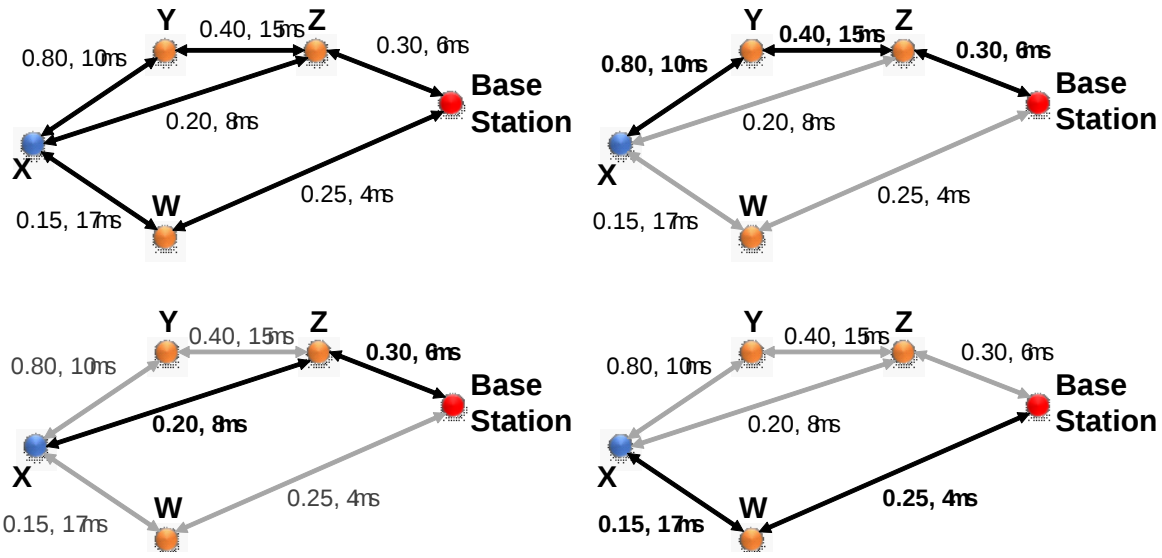

Figure 15: Using probability and throughput time metadata.

Additional Metadata considered could link could include data about the bandwidth saturation. The Radios offer a single channel at a fixed rate of 1mbs or 2mbs. Each radio runs in broadcast mode. So only one radio can use the frequency at a time may broadcast at a time. Thus, there is a physical limit to the maximum usage of the broadcast channel. In practice each cannel can move 1 or 2 megabits of data in one second. In terms of bytes it is 131,072 or 262,144 bytes. Since all nodes use this frequency, the frequency resource must be shared.

Frequency allocation to each node could be evenly divided. For example, in a five node configuration in a 2 Mbps system (Table 1), the allocation would be 262,144/5 or approximately a maximum of 52,428 bytes/sec for each node. If we assume an image file has 3.4kBytes or 174 thirty-two byte packets. So, an image uses 5,568 bytes when accounting for packet data overhead. This means in one second the node can send 9 images. This leaves 2,316 bytes for health status packets and network coordination traffic.

Now if we need to send more image data we could consider sending the data in a compressed format such as MPG4. This should compress the file to one tenth the size of the original image. The 3.4kbyte file becomes 349 bytes. This means the system need to use 18 thirty-two byte packets or 576 bytes to send the compressed image accounting for packet overhead. Again, if we examine a one second interval the system could send 87 compressed images with 2,316 bytes for health status packets and network coordination traffic (Table 2).

Other arguments could be made if we examine performing image processing at the nodes. This additional processing would be moved from the base station to the PAN area networks or distributed to the relay nodes. The data being sent back to the base station could then be reduced to a data set smaller than a compressed image.

Another consideration in allocating bandwidth would be to examine what data the individual node is sending. For example, a node sending just health information and using coordination traffic could use 2,316 bytes of the bandwidth. These nodes could then sur-

Table 1: Byter allocation in a 5 node configuration example at a speed of 2Mbps.

	File Size	20 byte File Chunks/Packets	Total Bytes for 32 Byte Packets	52,428 bytes/sec bandwidth allocations
Uncompressed Image	3.400 bytes	174	5,568 Bytes	9 image files + 2,316 bytes
Compressed Image	349 bytes	18	576 Bytes	87 image files + 2,316 bytes

Table 2: Allocation for the 5 node communication but using MPEG image compression at an average rate of 10:1.

	Event Allocation	Functional Allocation
node W (Fire Fighter No Camera)	52428	2316
node X (Fire Fighter With Camera)	52428	52428
node Y (Fire Fighter No Camera)	52428	2316
node Z (Relay Node)	52428	152652
Base Station	52428	2316

render 50,112 bytes each to be reallocated to the relay nodes. If we assume two nodes have an imaging camera then we could allocate these nodes 52,428 bytes. The relay nodes have a need to be able to move 2 sets of images each. So, we could add the additional bytes to both nodes evenly and thus they could have 102,540 (52,428 + 50,112) bytes from the bandwidth. Since the base station generates minimal traffic, mostly network coordination traffic we could also reduce its bandwidth allocation to 2,316 bytes. The additional 50,112 bytes could be allocated evenly to each relay node as well giving the nodes 127,596 (52,428 + 50,112 + 25,056) bytes (see Table 2 and Figure 16).

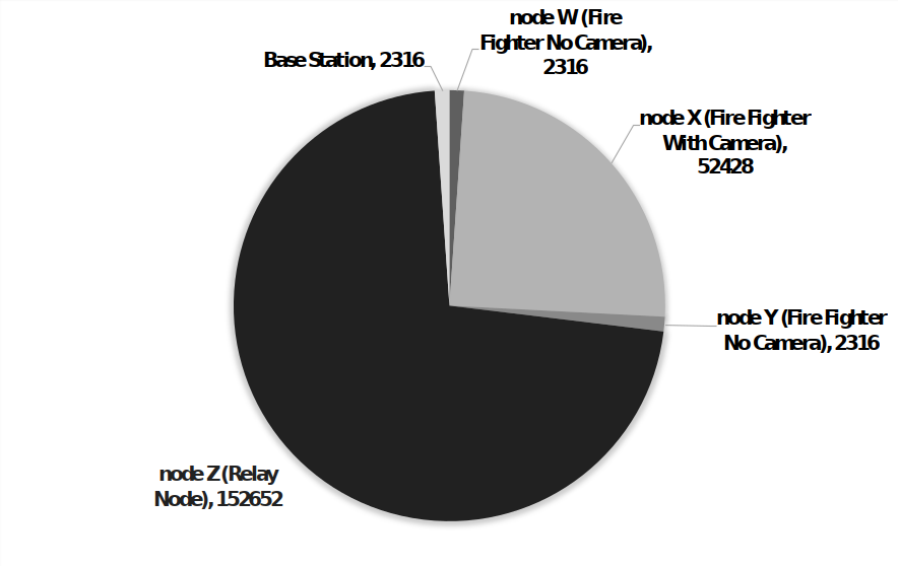

Figure 16: Bit allocation for a 5 node communications mesh with some nodes transmitting commpressed images.

In a more optimized setting we could have the nodes examine their queues and enter bids as to the allocation required and an arbitration node could allocate the limits based on the bids.

Another Optimization could involve looking at the placement of relay points based on RSSI information and throughput statistics.

3.5 FTP Server

The Erlang FTP server supports the movement of files to a destination location by first sending a file configuration packet that contains the destination or remote file location on the receiving node.

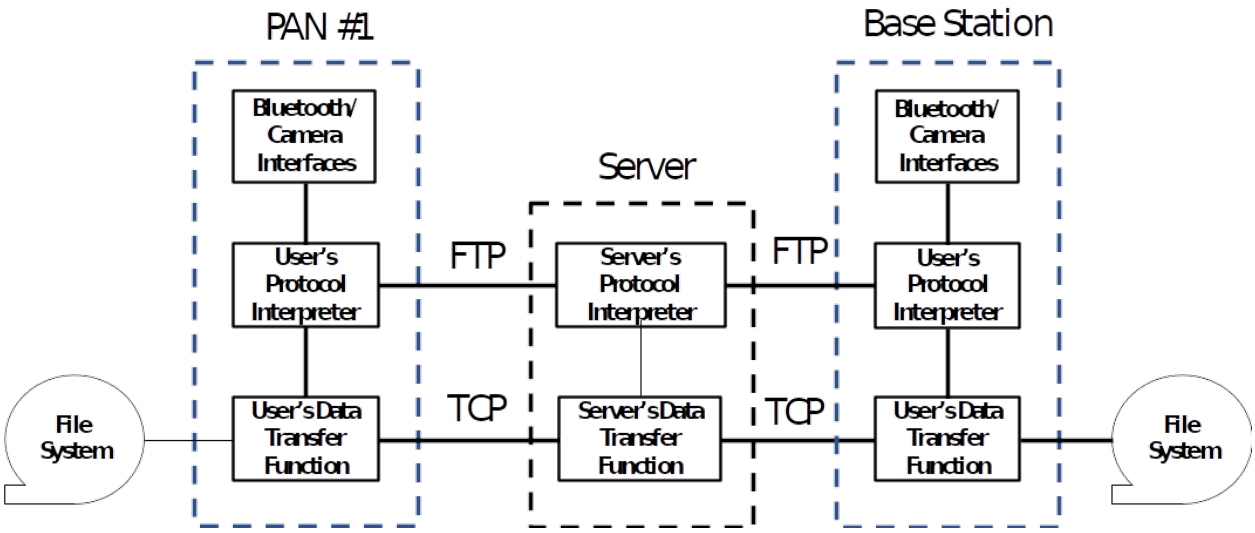

Figure 17: Figure 13 FTP file transfer mechanism

3.5.1 FTP Client Function

This application manages the breaking up image files into 22-byte chunks and reassembling the files at the receiving end. Typically, the FTP Image mode (commonly called Binary mode): The client application sends each file byte by byte, and the recipient stores the byte stream as it receives it.

4 Study Measurements

Given that the nature of the data in the network the primary concern is the movement of thermal imaging cameras (TICs) back to the base station. We will assume for the moment that there will be one TIC per engine company. For the image processing's convolutional neural network algorithm to perform effectively it needs to receive 10 images a second. This implies that each image must be sent and received in its entirety every 0.1 seconds. Thus, all the file transfer packets involved in the transmission of a thermal image must be received within a tenth of a second of starting to send the image.

4.1 Measure 1

4.1.1 Data

- Receipt time by base station, $t_{n,k}$, of all N_k packets associated with an image k from start of transmission of the image at the originating node. This forms the set transmission times $T_k = \{t_{n,k}, \forall n \in N_k\}$ for image k. Let the set of all images sent be K.

4.1.2 Analysis

1. The estimated distribution of for the set of all maximum receipt times for all the image's packets received by the base station. The set of maximum receipt times for a given image is defined as $t_{max}|_k = max\,(t_n)\,\forall n \in N_k$. The set of maximum times $T_{max,k} = t_{max}|_k, \forall k \in K$. Thus, the distribution would represent $P(T_{max,k}|\forall k \in K)$.

2. The estimated distribution of number of packets for any given image, c_k, received by the base station within a tenth of a second from start of transmission of the image k . Let $c_k = t_{n,k}|t_{n,k} \leq 0.1$. The resulting distribution is $Pc_k|\forall k \in K$

Another aspect of the system will be a study of how to implement smart queueing/routing disciplines. It stands to reason that if a packet for an image that is older than a tenth of a second from start of transmission utilizes queuing and transmission resources need for transmission of the current image. The study will examine different approaches to removing old or packets whose life time expires to improve the throughput and ensure timely delivery of the current image's packets.

4.2 Measures 2 and 3:

4.2.1 Data

1. Time spent in network queues, $t_{q,n,k}$, for n image packets of image k. This time is the time from when a packet is received to when it is transmitted, where q represents the queue/router identifier of a transmission node or the base station and Q is the set of all queues.

2. Number of image packets (regardless of image), $m_q\,[p]$ being routed through a node q for each 0.1 second time interval p. The total simulation time has P time intervals. So, the set of T_p is the set of all observation times.

4.2.2 Analysis

1. The estimated distribution of number of packets for any given image m_k routed through a given router/queue during the simulation. Let

$$m_{k,q} = \{|t_{n,q,k}||n \in N_k, q \in Q, \forall k \in K\}$$

   The resulting distribution is $P\,\{m_{k,q}|\forall k \in K, q \in Q\}$.

2. The estimated distribution for the maximum time $t_{max,k}|_q = max\,(t_{n,q,k}|n \in N_k, q \in Q, \forall k \in K)$ for any image's packets spend in queue at any given location in network. This includes queues at originating nodes during the simulation. The set of maximum times

$$T_{max,q,k} = (t_{max,k}|_q|\forall q \in Q, \forall k \in K)$$

   The resulting probability should be $P\{T_{max,q,k}|\forall k \in K, q \in Q\}$.

3. Determine the distribution of $m_q[p]$ across the set of observation times T_p . This distribution will represent the $P\{m_q[p] | p \in T_p\}$.

4. Comparison of estimated distributions dependent on queue and routing strategies to determine the degree of similarity.

If we look at the network topology as being mutable as it changes with time. We can introduce the loss of single and multiple nodes and study the networks response. Some of the measures proposed include examining the number of redundant paths to the base station and the average connectivity between nodes. This analysis could be driven by graph theory as proposed by Trace.

4.3 Measures 4 and 5

4.3.1 Data

1. Packet transmission times, ($x_{n,k,q}$) for packet n of image k of node q .

2. Packet receipt times, ($r_{n,k,q}$) for packet n of image k of node q .

3. Node connectivity table ($NC_q[p]$) for a data node, q , for each 0.1 second time interval, p . This table summarizes the nodes that the current node has connection with at time interval p as described in the following section.

4.3.2 Analysis

1. The estimated distribution of the number of 1 hop routes ($h_{1,q}[p]$) between the base station and node, q , for each 0.1 second time interval, p . A hop is defined as the transmission between a sending sending/receiving node pair. The total simulation time has P time intervals H_1 is the set of all observation times. The analysis will use the packet transmission and receipt times to check for nodes that are directly linked to the base station at time interval p . Essentially, this measures the likelihood of a packet reaching the base station without having to travel through intermediate nodes.

2. The estimated distribution of the number of 2 hop routes ($h_{2,q}[p]$) between the base station and node, q , for each 0.1 second time interval, p . The total simulation time has P time intervals. So, the set of H_2 is the set of all observation times. The analysis will use the packet transmission and receipt times to check for nodes that are linked to the base station using one intermediate node at time interval p . Essentially, this measures the likelihood of a packet reaching the base station one intermediate nodes.

3. The estimated distribution of the number of 3 hop routes ($h_{3,q}[p]$) between the base station and data node, q , for each 0.1 second time interval, p . The total simulation time has P time intervals. So, the set of H_3 is the set of all observation times. The analysis will use the packet transmission and receipt times to check for nodes that are linked to the base station using two intermediate nodes at time interval p . Essentially, this measures the likelihood of a packet reaching the base station using two intermediate nodes.

4. The estimated distribution of a node q being connected to another node in the network as identified in the node connectivity table ($NC_q[p]$) for a data node. Essentially, this measures the likelihood of a packet being passed on as a function of node connectivity.

5 Mininet

Mininet is a *network emulator*, or perhaps more precisely a *network emulation orchestration system*. It runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. It uses lightweight virtualization to make a single system look like a complete network, running the same kernel, system, and user code. A *Mininet* host behaves just like a real machine, so the user can use a Secure Shell (SSH) to enter into it (by starting up an OpenSSH Daemon (sshd) and bridge the network to your host) and run arbitrary programs (including anything that is installed on the underlying Linux system.) The programs running can send packets through what seems like a real Ethernet interface, with a given link speed and delay. Packets get processed by an emulated Ethernet switch, router, or middlebox, with a given amount of queueing. When two programs, like a client and server, communicate through *Mininet*, the measured performance should match that of two (slower) native machines.

In short, *Mininet*'s virtual hosts, switches, links, and controllers are created using software rather than hardware, and for the most part their behavior is similar to discrete hardware elements. It is usually possible to create a *Mininet* network that resembles a hardware network, or a hardware network that resembles a *Mininet* network, and to run the same binary code and applications on either platform [**Lan**].

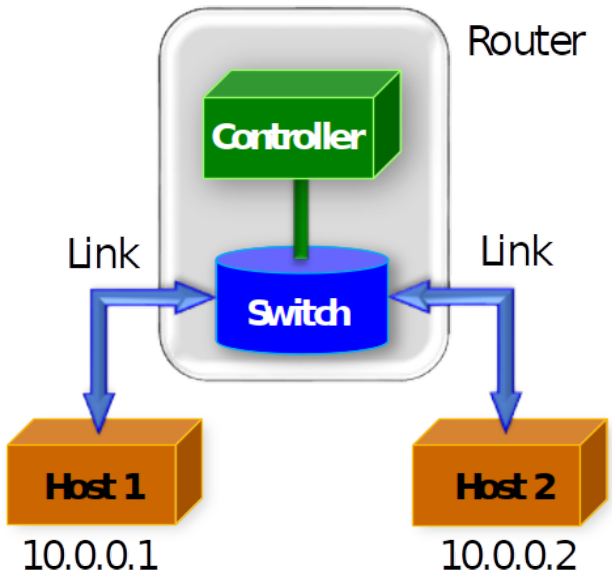

Figure 18: *Mininet* uses 4 components (hosts, links, switches, and controllers) to represent an Eithernet network.

The emulated host in Mininet is a group of user-level processes moved into a network

namespace (a container for network stat)e. Network namespaces provide process groups with exclusive ownership of interfaces, ports, and routing tables (such as ARP and IP). Each Host is a Linux shell that allows the node to run processes within network namespace such FTP transfers between hosts. Mininet uses CPU Bandwidth Limiting to limit the fraction of a CPU available to each process group.

Links are virtual Ethernet pairs. The data rate of each link is enforced by Linux Traffic Control (TC), which has a number of packet schedulers to shape traffic to a configured rate. Each emulated host has its own virtual Ethernet interfaces (created and installed with IP link add/set). A virtual Ethernet (or VEth) pair, acts like a wire connecting two virtual interfaces, or virtual switch ports; packets sent through one interface are delivered to the other, and each interface appears as a fully functional Ethernet port to all system and application software.

5.1 Multi Path TCP (MPTCP)

Multipath TCP allows a particular TCP flow to simultaneously send data across multiple paths to the receiver. This increases the connection's resilience to downed links across any particular path and allows for higher throughput by utilizing multiple network interfaces available on hosts. For example, a cellphone could simultaneously send data via a MPTCP connection by starting a subflow over both the WiFi link and the cellular connection (3G/4G). If either signal deteriorates, the flow can survive. If both signals are good, MPTCP will take advantage of the combined bandwidth to improve overall performance.

We use this setup to down and then reinstate links. The timing is managed as a relative sequence. That timing is dependent on the preceding instruction and not on some absolute clock reference. In this case, after completing the start transmission command for the sender, we wait 20 seconds and force the Ethernet (Sender.s1) link to fail. After the link command is completed we wait 10 more seconds and fail the link between the WiFi (Sender.s2) link. Again, waiting for the link to shutdown we wait another 10 seconds and bring the WiFi (Sender.s2) link back up. After reestablishing the link, we wait another 10 seconds and reinstate the Ethernet (Sender.s1) link.

The throughput/bandwidth is shown in Figure 19. The 3G link (red line) continues despite the other downed links, indicating that the overall TCP connection has survived the loss of the other two links. Note some interfaces may stop earlier than others.

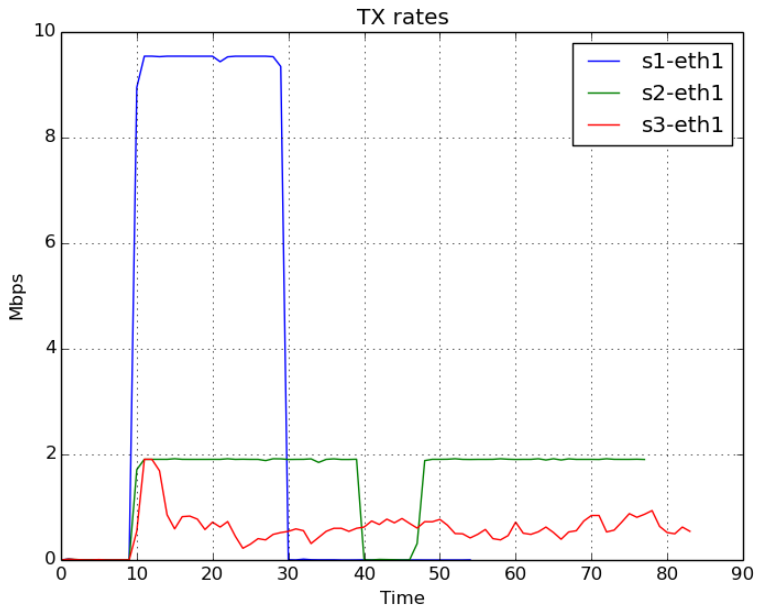

Figure 19: Typical Handoff Experiment Output

5.2 Fire fighter Model

The previous experiment is expanded into a 3-fireman and 3-relay node network topology. The experimental setup is in Figure 20. Blocks $PixRY$ are the links between fighters and relays. Blocks RX_B are the links between relays and base station. Blocks PiX_Y are links between nodes X and Y and blocks RX_Y are the links between relays.

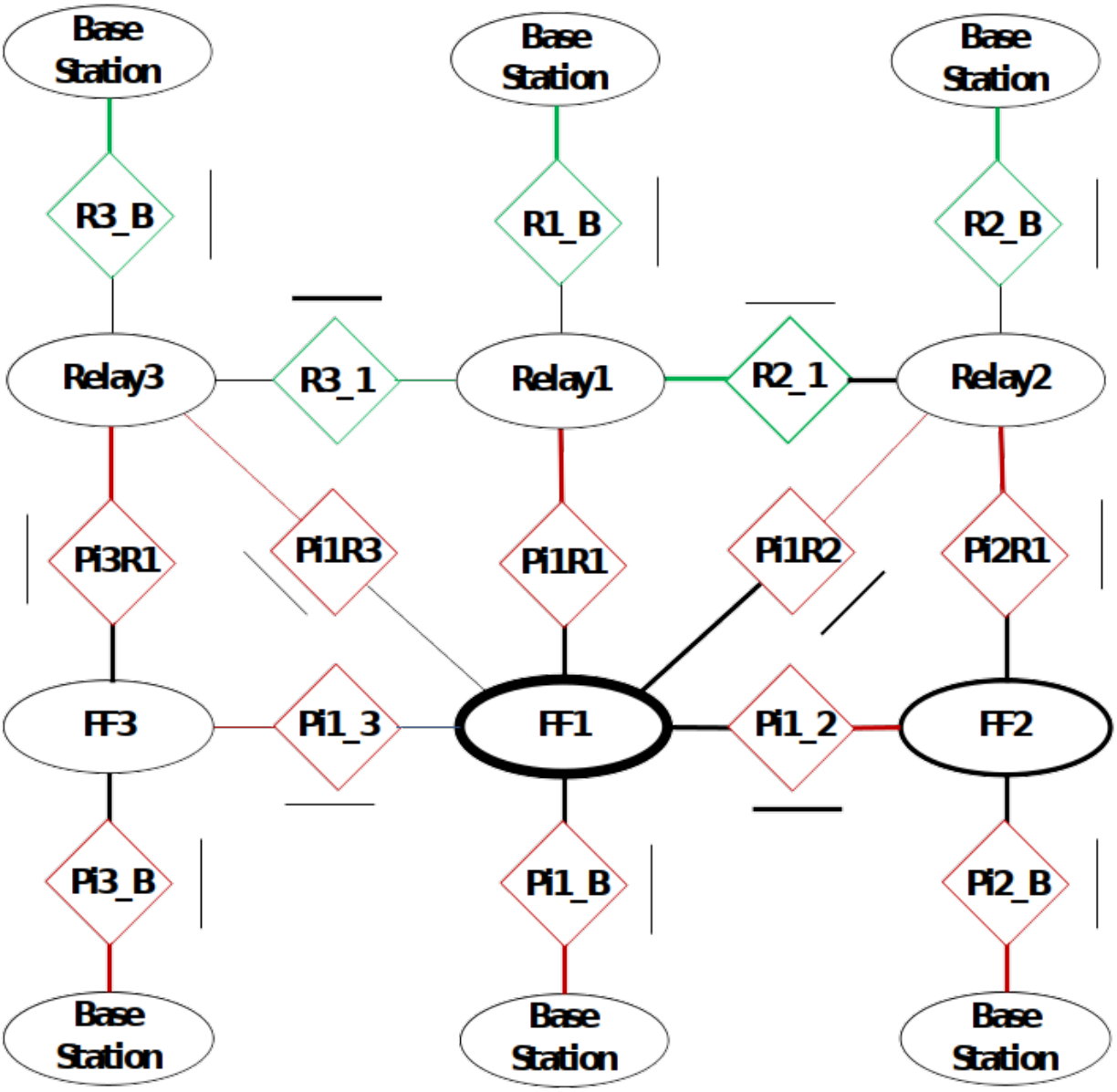

Figure 20: Mininet Topology for a fire fighter node sending data to the base station.

Initially each red and green link are modeled as having the same bandwidth of 2 Mbps, 1 ms delay, and a 3% dropped packet rate. Each node has 2 redundant links. Each node in the above network are modeled shown in Figure 21.

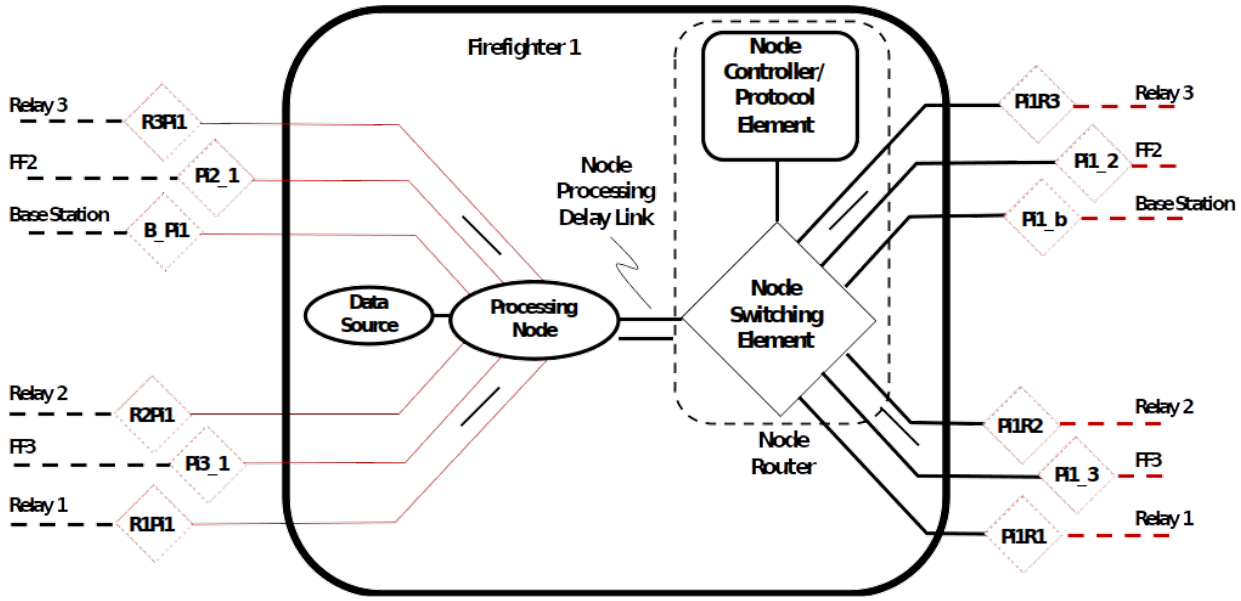

Figure 21: Mininet Firefighter node Topology.

6 Model Verification

The system has been tested in order to determine whether the performance meets the required characteristics. The physical environment was the Electrical and Computer Engineering Department of the University of New Mexico. The different nodes were positioned in different rooms of the building and the relays were randomly placed.

The frames sent have the following header parameters: The image serial number, from 0 to 4095 images until it needs to be reset to zero and the image chunk serial number, in order to reconstruct the image and to report for losses. After the data bytes, a checksum code of 2 bits is generated. The total number of bytes per image was of 53694 bytes per image. The target speed was 1 byte per 0.0333 ms or 3000 bytes per 99.9 ms, which translates in a rate of 10 images per second. Table 3 shows examples of the time needed to transmit images. The results show that the needed time was about twice the targeted time for this project. Nevertheless, the functionality of the network was correct (see Table 4).

7 Conclusion

This report shows the structure of the prototype of a nodal communication system designed to establish a mesh network for its use in fire fighter scenarios. The system is intended to transmit images for their analysis in order to detect objects of interest in the fireground, plus voice, respiration and other physiological signals form the fire fighter.

The system consists of a personal area network constructed over Bluetooth, that connects devices carried by the fire fighter to a node. The devices include a microphone and other low bandwidth sensors. A thermal image camera can be also connected via BT or

Table 3: Examples of data rates and packet loss in a transmission.

Test #	Time	Time to transmit 3000 bytes	Packets Lost, Ratio
1	35926 ms	200.73 ms	0/2065
2	35943 ms	200.82 ms	0/2065
3	36108 ms	201.74 ms	0/2065
4	36355 ms	203.12 ms	1/2065
AVG		AVG Time: 201.6025 ms	
		AVG IPS: 4.96 images/s	

wired. All elements of the PAN network have been emulated through the use of an Arduino microcontroller. Each node has been emulated using Raspberry PI B single board computers connected to the Arduino and to a radio board that broadcast packets to other nodes. The system has been complemented with relays, which are standalone nodes that can be placed at strategic positions to enhance the communications channel. The boards are powered using LiPO battery packs. The software defined radio has been constructed using the Erlang programming language in order to construct a nodal communications system that is able to route packets form any of the nodes to the hub, which has been emulated with a desktop connected to a radio device.

The system has been tested in order to check the ability to transmit packages in an indoor inconsistent communications channel. The system shows a high robustness in terms of packet loss. It is estimated that the image rate needed at the hub for image processing is of 10 images per second. Nevertheless, the current hardware only allows a rate of 5 images per second per node when 5 nodes have been included in the tests. Future work includes the use of more advanced hardware, including radios with a higher bandwidth (at least 2MBPS), single boards with higher computational power and reasonable power consumption.

Acknowledgements

This work has been supported by NSF Award 1637092 (The Next Generation Smart and Connected Fire Fighter System).

Table 4: Network functionality results

Tested Functionality	Description	Observations
Unique Addresses	Address of base station is unique	Successful, base,station assigned the address ID 0
	Each node is assigned a unique address	Successful, other,nodes assigned unique addresses as they connected
Transmission of Data Between Nodes	Information can be transmitted between source and destination nodes	Successful, information was routed from a specified source to a specified destination
Relay Point Software	Relay points are programmable and will be used to host the network software	Successful, relay points were programmed with the network software
	Relay point software will be based on RF network examples	Successful, software was based on RF network examples
Image Transmissions	Each transmitted image will have a CRC included in the transmission	Successful, network appends a CRC to each transmission
	Images will be transmitted at a rate of 10 per second (3000 bytes per image)	Unsuccessful, transmitted 5 images per second while maintaining reliable transmissions
Personal Area, Network (PAN) Interface testing	Bluetooth communication between Bluetooth node simulators and RF network nodes in a PAN	Successful, was able to establish a PAN supporting the reception and transmission of messages
		Network included multiple BT node simulators
		Multiprocessing/multithreading in Python script on node's processor.
PAN/WAN Interface	Establishing communication between PAN and RF network nodes proved to be viable	Successful, able to transmit and receive messages between 2 nodes with Python
		The RF24 radios can transmit a maximum of 32 bytes of data from the PAN.

References

[1] Francesco Cesarini and Simon Thompson. *ERLANG Programming*. 1st. O'Reilly Media, Inc., 2009.

[2] Martin Logan, Eric Merritt, and Richard Carlsson. *Erlang and OTP in Action*. 1st. Greenwich, CT, USA: Manning Publications Co., 2010.