9-30-2009

Model for Virtual Physical Layer Communication over Deployed Wireless Sensor Networks

Thomas Childers
Naval Postgraduate School, Monterey, California, USA

Yow Thiam Poh
Naval Postgraduate School, Monterey, California, USA

John McEachen
Naval Postgraduate School, Monterey, California, USA

Murali Tummala
Naval Postgraduate School, Monterey, California, USA

3rd International Conference on Signal Processing and Communication Systems
Omaha, Nebraska, 28-30 September 2009

A Model for Virtual Physical Layer Communication over Deployed Wireless Sensor Networks

Thomas Childers, Yow Thiam Poh, John McEachen and Murali Tummala
Department of Electrical and Computer Engineering
Naval Postgraduate School, Monterey, California, USA
{mceachen, mtummala}@nps.edu

Abstract—**A method for file transfer utilizing forward error correction (FEC) to pass traffic over deployed wireless sensor networks is studied. The entire wireless sensor network is modeled as an error-prone virtual physical link. Previous work in the area of terminal communication across the sensor network is expanded upon to include file transfer in order to provide a more capable channel and a basis for testing the performance obtained through erasure coding. The results of the FEC implementation are examined using multiple sensor network configurations. While the error correction method proved effective, larger topologies presented congestion issues due to the sensors' use of CSMA. Recommendations for future improvements are proposed.**

Keywords-wireless sensor networks; forward error correction; erasure codes; block codes.

I. Introduction

Sensor networks were designed to incorporate small, lost cost, portable devices that could collect and report on physical or environmental conditions. This type of information reporting suits these devices well as it requires less power and lower bit rates.

Among other things, military operational requirements drive the development of new technologies or the modification of current technologies to meet applicable mission objectives. The latter is the more desirable of the two as it usually requires lower developmental costs to come up with a working solution. By looking at sensor networks in this light, it is be possible they may provide capabilities outside of what they were originally designed for.

By configuring the sensor nodes as repeaters, it may be possible to use the devices, on a limited basis, as a means to extend network communications into regions normally difficult to access. Their small profile and portability would allow them to be nearly invisible to the enemy, meanwhile serving as a pipe for vital communications to forces in theater. While limited in power and throughput, they may provide a temporary solution where no others exist.

In order to utilize sensor networks as a bridge between communications networks, it is necessary to look more closely at how sensor networks work and decide what measures must be taken to facilitate this implementation. For example, TCP/IP traffic is based upon the principle of assuring that the transmitted packets will be received at the desired location. Sensor networks are normally passing information considered non vital and thus handle the data accordingly. On top of that, when working with a wireless medium, a large amount of loss can be expected. In order to consider using a sensor network to pass TCP/IP traffic, particular emphasis must be placed on data reliability across the network.

Two means of providing better reliability were considered. First, using Automated Repeat Request, or ARQ, was analyzed. Considerable research has been already been done in this area. While it would be an effective means of verifying whether or not packets were received properly, the difficulties of employing this method with multiple sensor nodes were daunting. The work that this thesis was based on used a simple broadcast of packets that each node repeated until received by the destination. Changing this scheme would present two major problems. By adding a feedback channel, the throughput would be significantly decreased. If the end goal was to be, for instance, passing Voice over IP (VOIP) traffic over the network, throughput would be a major concern. Something in the order of 90 kbps would be required for a reliable channel depending on the codec chosen [1]. Second, the use of an ARQ response would be fairly straightforward when using a simple, one hop network, but would increase exponentially in difficulty as more nodes were added to the topology.

Because of the difficulties of using acknowledgements, Forward Error Correction (FEC) was chosen to add reliability to the communication path. Throughput was a concern but the implementation was considerably easier. Based on the available FEC schemes available, a form of block coding was chosen for this effort. The implementation of the block coding scheme and the results of its implementation are discussed later.

Previous work [2] successfully set up a Java based implementation of a text messaging service across a sensor network. Messages that were typed in to the sending terminal were packaged into packets and sent over the network to the destination terminal which assembled the packets and displayed the message. To improve the capabilities of the channel, the next step was to add the ability for transporting files across the network. Specifically, the ability to send Joint Photographic Expert Group (JPEG) encoded images was desired. Adding this capability would improve the usefulness

978-1-4244-4474-8/09/$25.00 ©2009 IEEE

of the channel and make it possible to evaluate the effectiveness of an FEC algorithm.

Adding FEC to the channel would improve the reliability but also decrease the throughput. It was necessary to look at the effectiveness of the chosen algorithm and the throughput that resulted from its implementation. During experiments, the FEC implementation would be compared against two different transmission schemes. One would be transmitting the data from a representative JPEG image without any redundancy. The second would be transmitting a copy of the packets along with the original packets. This second scheme would send roughly the same number of packets as the FEC scheme and provide a better basis for comparison.

Several useful studies of error correction in sensor networks have been conducted. Researchers in [3] conducted tests of single and double error correcting codes in outdoor and indoor tests. Another interesting research topic involved using an adaptive FEC code control algorithm for sensor networks [4]. In the study, they identified the need for something other than fixed correction codes for channels with constantly varying bit error rates. Finally, Terry Norbraten's work with erasure codes and detailed explanation of the Java FEC Library from Onion Networks were extremely helpful during this research [5].

One of the realizations after examining the results from this research was that simply using error correction alone to improve the reliability of a channel is not sufficient. With increasing network topology complexity, additional measures should be considered. One such measure deals with modifying medium access control (MAC). A few different protocols were examined. Z-MAC is an exciting MAC protocol that achieves high efficiency by acting as hybrid between TDMA and CSMA [6]. Z-MAC behaves like CSMA during periods of low contention and like TDMA during periods of high contention. By using this approach, it aims to maximize efficiency during all phases of network activity. Two other protocols, T-MAC [7] and S-MAC [8], represent hybrids between TDMA and CSMA, although these protocols put more emphasis on energy efficiency while Z-MAC aims to maximize network throughput.

II. WIRELESS SENSOR NETWORKS AS A NETWORK BRIDGE

An exciting application worth considering is the temporary use of sensor networks as a bridge between communications networks. Although throughput and power limitations prevent these devices from performing more intensive data transfers, temporary use to aid the military units in difficult environments may be worthwhile.

A. Benefits of Implementation

Wireless network communications are becoming more prevalent for military operations. Intelligence reports, imagery, and general communication are reaching further into the battlefield than ever before. Typically, the infrastructure found in these environments is limited or non-existent.

Wireless sensor networks were originally designed for the purpose of reporting on the environmental and physical conditions of the battlefield. Using these networks to temporarily extend vital TCP/IP network communications might be possible despite the limitations of the sensor nodes. A few specific applications of interest are Voice-Over-IP (VOIP) and the transmission of time sensitive data and imagery. In order to consider this as a possibility, the difficulties involved must first be considered

B. Barriers for Implementation

Sensor networks were designed to transmit small amounts of data with limited frequency. As a result, the sensor nodes typically have limited onboard memory. The MICAz motes used in this research have 128 kbytes of flash program memory, 512 kbytes of flash log memory, and only 4 kbytes of RAM. Since the data they transmit is considered non time-sensitive and non vital, the networks do not have to incorporate many of the assurances necessary in TCP/IP networks. Tunneling over the sensor network may involve increasing the onboard memory to support an enhanced network stack or finding ways to more efficiently use the smaller amount of memory.

Power is another limitation that must be addressed. As sensor motes were designed for low power, autonomous operation, batteries or solar power are typically employed. Increasing the data amounts and rates would increase the power demand and threaten the longevity of the device.

In order for this application of sensor networks to be successful, the throughput of the network would have to provide a certain level of performance to meet the needs of the user. To use the network to pass VOIP traffic, for example, a minimum data rate of about 90 kbps would have to be supported to provide adequate communications. For transmission of time sensitive data or imagery, throughput would have to meet specific mission requirements.

By testing the transmission of image files across a sensor network during the course of this research, it was desired that a general idea of the throughput capabilities would be found. Although the forward error correction implemented would reduce the channel's throughput, it was considered necessary for the channel's reliability.

III. THE SENSOR NETWORK ADAPTATION INTERFACE LAYER (SNAIL)

As mentioned before, the work in [2] involved the development of a sensor network channel that allowed a form of text messaging. Three Java applications that would work in coordination with the sensor mote hardware from Crossbow to achieve this task were developed. The suite of applications was referred to as the Sensor Network Adaptation Interface Layer (SNAIL). SNAIL consisted of separate Client and Server modules that were used on a transmitting laptop, and a Listen module that was used on the receiving laptop. These SNAIL modules were modified extensively to achieve the goals of this research.

In order to incorporate a more robust channel by adding error correction, the first step was in modifying the SNAIL software to incorporate file transfer. The larger amounts of data associated with file transfer would allow the improvements provided by error correction to be observed. The file types that were chosen for implementation and testing were JPEG images, test files, and MS Office documents. These were chosen due to their popularity and everyday use. How each of the SNAIL modules was modified is explained below.

A. SNAIL Client Module

The SNAIL Client module now presents the user with three options upon running the application. Choices are now for text message transfer, standard file transfer, or JPEG image transfer. Before passing off the data to the SNAIL server module, a few modifications were necessary.

Previously, the Client module allowed the user to input a text message from the terminal to be transferred over the sensor network. Using a blocking reader module, upon receiving a message from the user, the message was read in as a string and then converted to a character array. The use of a character array was chosen to allow some flexibility with the data stream. Essentially, this allowed adding the user's selection to the data stream before passing it on to the Server module. Prior to the IO operation, the character array had been converted back to a string.

To accommodate the transfer of files, the first modification was changing the module to work with byte arrays. Should the user choose to send a text message, the message is still read in as a string, but is converted to a byte array. The option selected is added to the byte array by using concatenation of arrays. By using byte arrays, the data is now compatible with the file transfer options. Strings could have been used for both, but ultimately that would have limited the file length to roughly 64 kbytes.

Upon the choice of either standard file transfer or JPEG image transfer, the user is presented with a file selection box. Upon selecting the desired directory and file, the file data is read in to a byte array. The two file transfer options are executed differently. For standard file transfer, a fileinputstream is opened to pull in the data to a byte array. To pull in a JPEG image correctly, the Java ImageIO tools were used to read in the file as a bufferedimage. After the file is read in, it is converted to a byte array.

By utilizing byte arrays for each of the three options, all three are assured to be compatible and the same flexibility to modify the data stream that character arrays afforded is maintained. Finally, the data residing in the byte array is transmitted to the SNAIL Server module via an ObjectOutputStream. Figure 1 presents the process decisions of the SNAIL Client module as a flow diagram.

B. SNAIL Server Module

The SNAIL Server module required more extensive modifications to allow file transfer and error correction. Error correction will be covered later. For now, data handling and packetizing for sensor network transport will be covered.

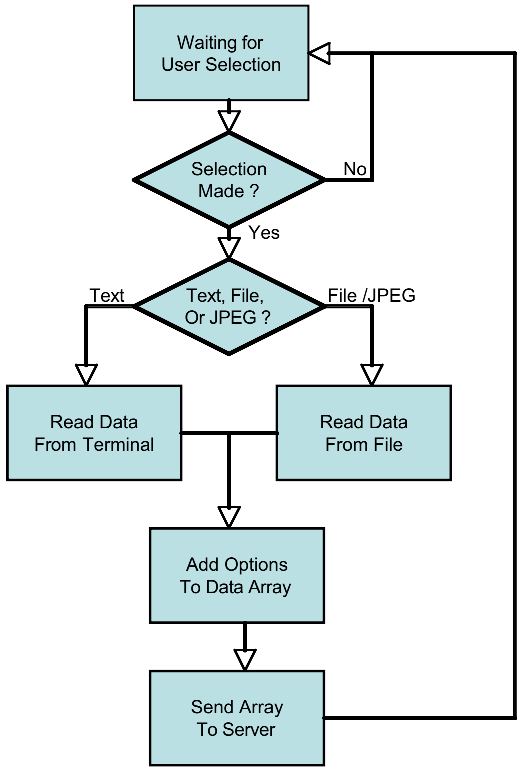

Figure 1. SNAIL Client Flow Diagram

After reading the data sent from the SNAIL Client module, the data is stored in a byte array. One of the main changes is the handling of all data as byte arrays from start to finish. From the data array, the option that was selected is obtained. The option will ultimately be removed from the data and transmitted as part of a handshake packet. The handshake packet is the first packet that will be transmitted by the server and contains much of the amplifying information needed by the receiver.

The destination bitmap approach takes advantage of the existing slot structure used to support the non-contention mode of the protocol. The slot assignment process can be either distributed or centralized and the results must be disseminated to all nodes identified within the transmission frame.

Header(5)	Payload(29)	CRC(2)

Figure 2. Packet structure of TinyOS packets

Should encoding not be desired by the user, the number of packets to be sent is calculated from the array size. The array size is a key piece of information needed by the receiving terminal and is included in the handshake packet. Of the 29 bytes of data available in the Active Messaging (AM) packet, two of the bytes will be used for Terminal ID and packet number for this non-encoding case. Figure 2 above shows the standard AM packet structure used by TinyOS. The packet number will help the SNAIL Listen module keep track of which packets have been received and facilitate dropping redundant packets.

The use of the header packet was changed only with regards to content. Because of the new error correction option, additional information was required at the receiver. One new addition to the transmission process was the use of a terminating packet. After all of the data packets are sent out, the terminating packet is sent out which contains an identifying byte sequence. This packet was added to correct the condition where packets are dropped and the receiving end is stuck in a loop waiting for packets. Once the terminating packet is read in, the Listen module is free to move on to analyzing the data.

Figure 3 illustrates the flow diagram for the SNAIL Server module.

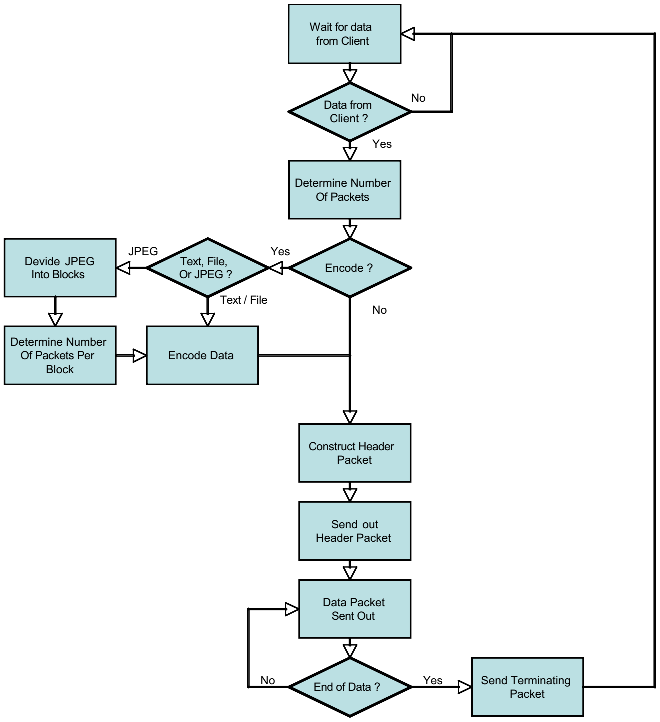

Figure 3. SNAIL Server Flow Diagram

C. *SNAIL Listen Module*

Upon receiving the handshake packet, the SNAIL Listen module knows whether or not the data will be encoded, what option was selected, and the length of the data array involved. Using the array length, the number of packets to be expected is calculated.

A loop is entered in which each of the packets is read in. Both the handshake and terminating packets contain flags to help identify them from normal packets. Also, as the packets are read in, the packet number is obtained which is used to identify the packet. If the packet is redundant, it is dropped. A change was made to help keep track of which packets are read in. Previously a packet counter expired when the expected number of packets was reached. This was changed due to the encoding option. This will be covered further in the next chapter.

Once all of the data has been read in and decoding has been completed, if necessary, the option selected determines how the data is handled. If a text message was selected, the message is displayed on the terminal. If a standard file was transferred, the file is saved to the location chosen by the user using a fileoutputstream. For a JPEG image, the byte array is converted back into a bufferedimage and then the image is stored using the ImageIO utilities.

Figure 4 below presents the flow diagram for the SNAIL Listen module.

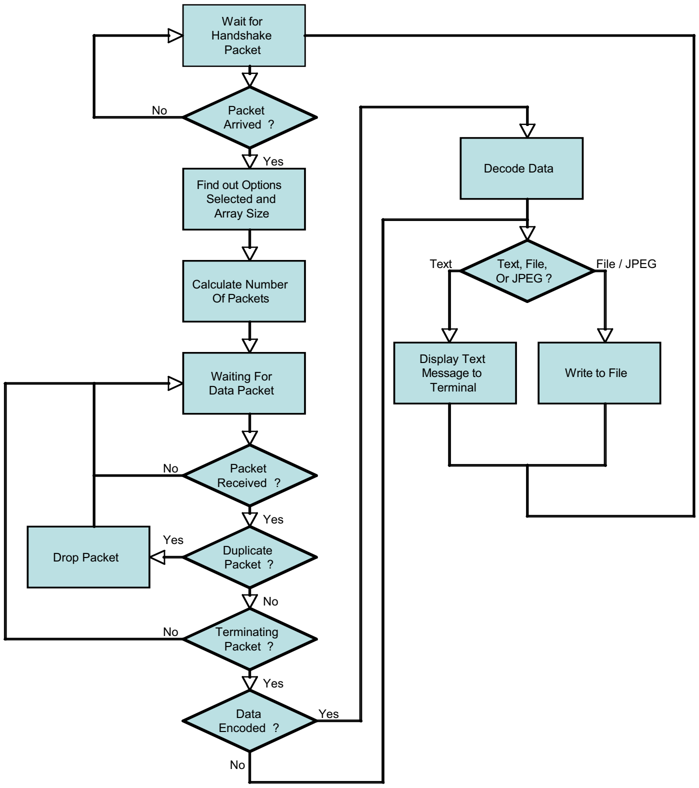

Figure 4. Flow diagram for SNAIL Listen Module

IV. FORWARD ERROR CORRECTION IN WIRELESS SENSOR NETWORKS

In wireless networks, packet loss is inevitable. In order to combat this packet loss, either Forward Error Correction (FEC) or Automatic Repeat Request (ARQ) or a combination of the two techniques are used. ARQ is an attractive option since it is relatively inexpensive in that it requires no manipulation of the data being transferred. Under conditions of increasing losses, ARQ does suffer significant reductions in overall throughput. Tradeoffs exist between complexity of implementation and data throughput. ARQ also rapidly becomes more complicated as the number of clients grows.

FEC, on the other hand, detects and corrects losses incurred by a noisy channel by including redundant information with the data it passes. This has the advantages of allowing the correction of errors more quickly than with ARQ and by simplifying the network traffic scheme. Some kind of feedback channel could be included but may not be necessary.

A. Overview of FEC Correction Methods

The available types of FEC are broken down into two categories. These are block codes and convolutional codes. Block codes work on fixed-size blocks of bits or symbols of a fixed size. Convolutional codes work on bit or symbol streams of various sizes.

1. Block Coding

Many different forms of blocks codes exist. A few examples are the Hamming code, BCH code, and Reed Solomon code. The latter is the most widely used due to its near optimal coding qualities.

A Reed Solomon code encodes a data message block as points in a polynomial function plotted over a finite field [9]. The polynomial coefficients are the data symbols of the block. These codes can work to correct errors at either the bit-level or the packet level. Lost data packets are corrected from encoded packets, otherwise known as repair packets. These repair packets represent a set of linearly independent equations. By solving this set of equations, the lost packets are recovered. One drawback to the Reed Solomon coding scheme is encoding and decoding time which is O(n2) and O(n3) respectively. Another is the memory requirement resulting from the polynomial operations.

2. Convolutional Coding

Convolutional coding involves taking an m-bit message that will be encoded and converting it to a n-bit symbol. The code rate for the encoding process is m / n where n m. The constraint length of the code, k, determines the error correction capability and the complexity. As k increases, the correction capability increases but the complexity also increases exponentially. For decoding convolutional codes, the Viterbi algorithm is commonly used. The Viterbi algorithm uses maximum likelihood estimation to make decisions regarding the underlying probability distribution of the bits received [10]. For effective correction, a constraint length of at least 7 and typically below 9 is used while a code rate m / n of at least 1 / 2 is required. A convolution code has reduced complexity over a Reed Solomon code but suffers higher coding redundancies. For this reason, convolutional codes are more ideal for communication channels with a lower signal-to-noise rate (SNR). A convolution code is also not typically used for the recovery of lost packets as is Reed Solomon.

B. Erasure Coding

For this research, an implementation of erasure coding was chosen. Erasure coding is basically a form of block coding that takes a number of data packets, or blocks, and encodes them into a larger number of encoded data packets. The larger the number of encoded data packets, the more redundancy allowed. As long as a minimum number of the transmitted packets reach the destination, the source data can be reconstructed. The key to erasure coding is that the destination knows exactly which packets have been dropped. Without that knowledge, this coding scheme would not work.

Erasure coding was chosen due to the inclusion of CRC in the wireless sensor network packets. Since the lower layers of the protocol stack would check arriving packets for errors, only allowing error free packets to reach the application layer, this method of FEC coding seemed appropriate. Packets lost in transmission or dropped due to errors would not prevent the successful transmission of the source data. An open source JAVA implementation of erasure coding created by Onion Networks was implemented during the experiment [11].

1. Erasure Coding Fundamentals

The basis behind erasure coding is that *k* blocks of source data are encoded producing *n* blocks of encoded data [11]. If any subset of the n encoded blocks is received at the destination, the receiver is able to reconstruct the source data. This code is referred to as an *(n, k)* code. In this scheme, up to $n - k$ losses are acceptable.

A subset of the erasure codes, called linear codes, can be analyzed using the properties of linear algebra. If $\underline{x} = x_0 \ldots x_{k-1}$ represents the source data and G is an $n \times k$ generator matrix, then $\underline{y} = G\underline{x}$ is the (n, k) linear code resulting from the matrix multiplication. As long as k components of $\underline{y}$ are received, $\underline{x}$ can be recovered.

If the encoded data contains an exact copy of the source data, this is referred to as a systematic code. With a systematic code, a portion of the generator matrix, *G*, will contain the identity matrix. Systematic codes can be very advantageous if very few losses are expected in the link. Reconstruction of the source code would be greatly simplified.

The generator matrix G is a $n \times k$ matrix of rank k. Because of this, only k of the n encoded packets are necessary. Each column of G can be composed of a maximum of $(k - 1)$ nonzero elements. For the systematic code example, since the columns already have $(k - 1)$ zero elements due to the identity matrix, all of the remaining elements are required to be nonzero.

For the reconstruction process, along with the encoded packets of data, the identification of those packets must also be known. This will add overhead to the process as the transmitting end will have to include this information with the transmission. This is a negligible amount of overhead though and the packet identification will also aid in with identifying redundant packets so they may be ignored. The recovery is performed by solving the linear system:

$$\underline{y}' = G'\underline{x} \rightarrow \underline{x} = \left(G'\right)^{-1} \underline{y}' \qquad (1)$$

For the above equation, $\underline{x}$ represents the original source data and $\underline{y}'$ is a subset of k encoded packets. G' is the corresponding subset of columns from the generator matrix. To reconstruct the original data, the inverse of G' is taken and then multiplied by the subset of encoded packets, $\underline{y}'$. The cost of inversion is somewhere in $O(kl^2)$, where $l \le \min(k, n-k)$. The value of l represents the minimum number of packets that must be received.

2. Erasure Code Based on Vandermonde Matrices

An example process for the creation of a generator matrix can be shown with the use of a Vandermonde matrix. This matrix has coefficients of the form

$$g_{i,j} = x_i^{\,j-1} \tag{2}$$

where the x_i 's are elements of extension fields, or $GF(p^r)$.

Extension fields are a subset of finite fields that allow basic arithmetic to be performed on data much like it is done with integers. They help resolve problems associated with handling the number of bits needed to represent the result of computations. Mapping data elements into field elements prior to arithmetic operations and then applying the reverse mapping to get the desired results avoids this trouble. If finite fields are not used and the results of the coding arithmetic operations are rounded prior to transmission, exact reproduction of the data would not be possible.

Seen in matrix form, the $n \times k$ Vandermonde matrix is

$$G = \begin{pmatrix} 1 & x_1 & x_1^{\,2} & \cdots & x_1^{\,k-1} \\ 1 & x_2 & x_2^{\,2} & \cdots & x_2^{\,k-1} \\ 1 & x_3 & x_3^{\,2} & \cdots & x_3^{\,k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^{\,2} & \cdots & x_n^{\,k-1} \end{pmatrix}. \tag{3}$$

The determinant of a square Vandermonde matrix is defined as

$$\prod_{i,j=1\ldots k, i<j} (x_j - x_i). \tag{4}$$

The matrix will have a non-null determinant and thus be invertible if all of the x_i 's are different. As long as $q > k$ and all x_i 's are not equal to zero, $q - 1$ rows at a maximum can be created, where q is the number of finite field elements. If the identity matrix is added, a suitable generator can be created for a systematic code.

Considering a few special cases for the code, a (*n*, *1*) code would simply create copies of the single packet. This is essentially the same thing as making multiple copies of the same packet to be sent out. The work that this thesis built upon utilized this simple method of improving the link performance by sending multiple packet copies. Unfortunately, this type of code is inefficient compared to codes with higher values of *k*. A (*k+1*, *k*) code is another simple case. This would include the *k* packets plus one packet that would represent the sum of the others. Once again, this case is not very useful except for channels with small amounts of loss.

3. Erasure Codes for Wireless Sensor Networks

Once again, erasure codes were chosen to combat the relatively high amount of packet loss that can be expected with wireless sensor networks. While the software implementation of erasure codes is somewhat computationally expensive, low to medium speed applications, up to the 100 KB/s range, could be supported with fairly low amounts of overhead. Given the limitations of power and throughput with these networks, erasure codes may be a very useful tool. Combining this technique with a simple form of ARQ might be the best course of action.

V. EXPERIMENT DETAILS

The performance of the link was the primary concern while conducting the experiments. Since a more robust link was desired, the performance of the sensor network link was tested with forward error correction implemented. In order to provide a comparison for the results, experiments without the FEC coding were also conducted. These results will hopefully provide a good foundation for what performance enhancements can be achieved using the FEC scheme selected.

Three different scenarios were chosen for experimentation. These include a direct terminal to terminal test, a terminal to terminal test via one sensor mote hop, and a terminal to terminal test via two sensor mote hops. For each of these scenarios, three different data transmission schemes were tested. First, the erasure coding scheme was tested. The second scheme involved testing the link by sending only the image data across. No redundancy was used during this test. For the last scheme, redundancy was incorporated by sending a copy of each packet along with the original packets. This last scheme was chosen since it more closely approximates the total number of packets being sent out during the FEC test.

To test the three transmission schemes, a small 8 kbyte image was chosen to be transferred. The small file was chosen to allow for large number of tests to provide an adequate amount of results for a comparison. For a successful transmission, the entire image had to be transferred without error.

A. *Experiment setup*

The experiment was originally conducted in an academic building hallway to allow enough space to separate the base stations and motes in order to drive the link to the edge of its performance capabilities. By adjusting the power on the motes and investigating the resulting transmission range, the motes were placed to force a considerable amount of packet loss. This was necessary to prove that the FEC method would handle more challenging conditions better than the schemes lacking error correction.

After initial trials that were conducted, it was determined that multipath effects encountered in the hallway environment would negatively influence the results. Due to this realization, the experiment was relocated to an anechoic chamber which would help by eliminating a majority of the undesired signals.

1. Hardware

Each of the base stations incorporated a laptop and a Crossbow MICAz sensor mote connected through the USB port using a MIB520 sensor board. Individual MICAz motes were used for the repeater stations simulating the sensor network.

The mote can be used with or without an optional sensor board providing capabilities as a wireless sensor platform or as a wireless node. A variety of sensor and data acquisition boards can be connected to the MICAz by means of a 51-pin expansion connector. The RF transmit power for the MICAz is user selectable from -24 dBm to 0 dBm. This feature was particularly helpful in allowing tests to be conducted in a small laboratory sized environment. The CC2240 Transmitter datasheet power levels are shown in Table 1. Experimental tests found that the power levels lower than those shown were also possible.

These MICAz motes implement a CSMA based protocol. This was determined by the CC2420 radio installed on the board and the version of TinyOS installed. A more advanced CSMA protocol, B-MAC, became available in the 1.1.3 version of TinyOS, but was not available for this experiment. Among the changes in this protocol was a variable noise floor over the fixed floor originally used. This noise floor is used in the determination process of when the mote can transmit. B-MAC's improvements on performance might be a valuable topic for further work.

B. Terminal to terminal experiment

The first scenario for the experiment involved setting up a simple terminal to terminal link as pictured in Figure 5.

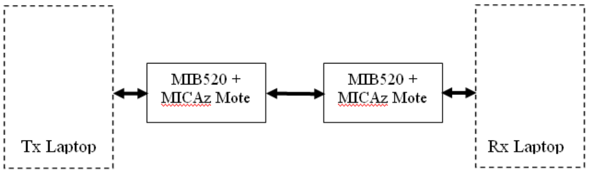

Figure 5. Scenario 1 – Terminal to Terminal Communication

This, the most simple of the arrangements to be tested, was used for the initial testing of the FEC code and would provide a good basis of comparison for the one hop and two hop tests to be conducted later. The Tx laptop shown in Figure 11 uses the SNAILServerTest_fec and SNAILClient_fec applications. Simultaneously running is the SerialForwarder application that is also used on the Rx laptop. In addition to SerialForwarder, the Rx laptop runs the SerialListenTest_fec application. Both MICAz motes were programmed with the TOSBase software. The TOSBase software was modified to set an RF power level of -24 dBm.

The transmitting and receiving motes were placed at a distance of 110 inches from each other. This distance was chosen to promote some loss of packets at the receiving end in order to test the effectiveness of the FEC algorithm. The FEC encoded transmissions were the best performing of the three tested methods. Of the twenty-five runs performed, the non-encoded scheme was unable to successfully transfer an image. Without any redundancy, losing a single packet constitutes a failure to transfer an image. Averaging 282 packets received out of the original 299 packets sent, this method of transfer did not provide for a robust link at the chosen distance.

The second transmission scheme sent a copy of each data packet or a total of 598 packets. This provides for the ability to lose random packets but consecutive lost packets could be a problem. This method proved successful for 60% of the transmissions, averaging 288 of the necessary 299 data packets. A counter was created to keep track of the extra or redundant packets that were received. An average of 260 redundant packets was received.

The performance enhancement of the FEC coding was obvious. On average 88% of the runs were successful; 589 of 640 sent packets were received on average. Looking at run number twenty, the image was saved successfully even though only 466 of the 640 packets were received. The key of the erasure code algorithm is that for each block of image data being transferred, as long as *k* packets out of *n* transferred are received, the file can be reconstructed.

C. One hop experiment

The general arrangement for the one hop scenario is shown in Figure 6.

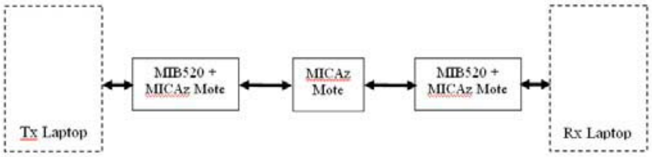

Figure 6. Scenario 2 - Terminal to Terminal via One Hop

To facilitate adding a hop between the two terminals, the power of the transmitting mote was reduced to a reference level two, which is believed to be equivalent to roughly -35 dBm. This equated to a transmission range of 1 ft. Due to the size limitation of the anechoic chamber, this short first hop would be necessary later for the two hop experiment. The power level for the mote used for the hop was set to -24 dBm.

Once again the layout of the motes was selected to force dropped packets to occur. Both of the non-encoded tests yielded no successful transmissions while the FEC encoded test was successful 100% of the time. Without redundancy, an average of 224 of the 299 necessary packets was received. Doubling the number of packets increased the average to 267 with an average of 141 redundant packets, although no successful transmissions were obtained. The FEC approach yielded an average of 459 of the 640 encoded packets received.

D. Two hop experiment

The last of the scenarios, the two hop arrangement, is pictured in Figure 7.

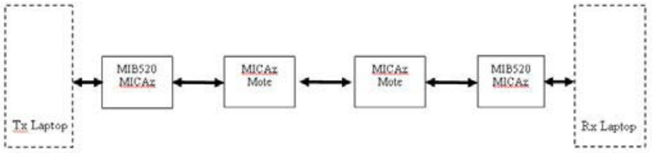

Figure 7. Scenario 3 - Terminal to Terminal via Two Hops

The arrangement of the terminal mote and first hop mote remain unchanged from the previous one hop test. The second hop mote, like the first, was set to -24 dBm for transmit power and located roughly 100 inches from the first mote.

Once again the FEC encoded data tests outperformed the non-encoded data tests. Adding redundancy to the non-encoded tests raised the success rate from 0% to 4%, or an average number of packets received from 166 to 270. The average redundant packets increased from 154 to 415. By using the FEC encoding, the success rate was increased to 52%. An average of 403 packets out of the 640 sent was received with an average of 200 redundant packets.

The two hop arrangement added a large number of redundant packets to the link and caused a considerable reduction of link performance as a result. The increase of repeated packets seemed to cause a large number of necessary packets to be dropped during transmission and even the encoded tests showed difficulty in transferring complete JPEG images.

E. FEC Performance Over Varying Distances

The previous scenarios that were run were setup to purposefully cause a considerable amount of packet loss in order to test the FEC effectiveness. Each of the transceivers were positioned at a certain location and set to a power level that would force this condition. While the tests did show that the FEC scheme provided improved performance over the other tests run, another series of tests was needed in order to give a more qualitative comparison of the three schemes.

Table 1. Effect of Varying Distances Upon Transmission Success

Distance (in)	No of Trials	Non-Encoded			Non-Encoded (Redundant)			FEC Encoded		
		Avg Packets Received	Avg Red Packets	% Success	Avg Packets Received	Avg Red Packets	% Success	Avg Packets Received	Avg Red Packets	% Success
43	15	299	0	100						
49	15	298.8	0	80						
55	15	298.3	0	67						
61	15	298.6	0	73						
67	15	298.4	0	73						
73	15	298.1	0	67	299	299	100	640	0	100
79	15	298.5	0	80	298.9	297.5	100	638.7	0	100
85	15	298.2	0	67	299	297	100	638.4	0	100
91	15	297	0	33	298.9	296.8	87	635.5	0	100
97	15	295.6	0	0	298.9	296	93	635.2	0	100
103	15	288.1	0	0	298.9	292.6	87	615.5	0	100
109	15	220	0	0	298.1	275.7	27	473.9	0	87
115	15				292.1	228.9	0	398.3	0	53
121	15				272.2	173.4	0	282.6	0	7
127	15				287.7	211.5	0	370.7	0	60
133	15				75	12	0	78	0	0
139	15									

In order to get a better idea of how the performance was falling off for each of the transmission methods as the distance increased, the final test was performed. The configuration of hardware used for this test was that of the original terminal to terminal test. The transmitting terminal, set to a power level of -24 dBm, remained fixed while the receiving terminal was varied to record the packet transmission effects. Each of the non-encoded and encoded methods were tested fifteen times at varied distances to find the transition from 100% image transfer success to the distance that resulted in a failure to transfer a single image.

Table 1 displays the results from the test. The non-encoded scheme showed a gradual decrease in link performance starting at a distance of 43 inches until it totally failed at 97 inches. At that distance, the non-encoded scheme that included redundant packets was still showing a 93% success rate. At 115 inches, the redundant packet scheme completely failed. At this distance the FEC encoded scheme still performs at a 53% success rate. Not till 133 inches did it fail to transfer a single image. There was an unexpected spike in performance for the FEC scheme at the next to last distance that was unexplained.

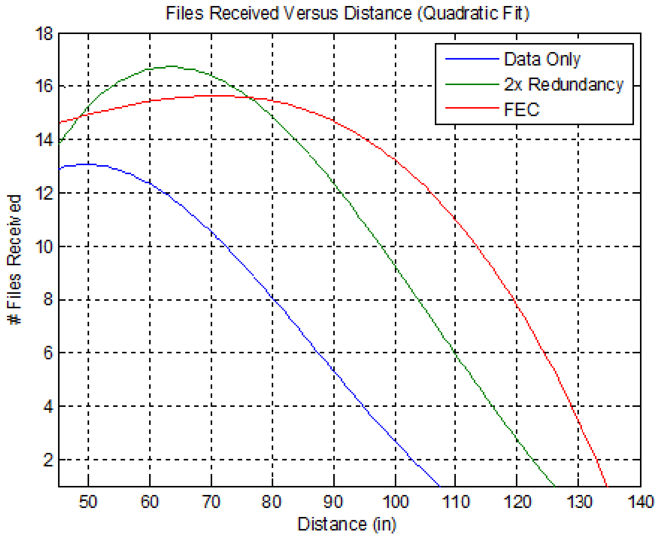

Figure 8. Quadratic Fit Curves for Experimental Results

In order to get a better picture of the performance falloff, the plot shown in Figure 8 was created from the recorded data. To achieve the smooth curves shown, quadratic fit curves were created from the data. To compare the three transmission schemes, a reference for acceptable loss of 25% was chosen. This translated to receiving 11.25 out of the 15 files sent. By that standard the non-redundant scheme that included redundancy represented a 42% increase in transmitting range from the non-redundant scheme. The FEC encoded scheme represented a 66% increase in transmitting range.

VI. Conclusions

For each of the experiment configurations used, the erasure code outperformed the non-FEC schemes. The larger the amount of data to be sent, the more effective erasure code would be for the transmission process. While the TinyOS AM packet structure is very limiting, the positive effects of the coding scheme were still observed.

While the error correction method did improve the performance of the channel, the two hop experiment pointed out flaws with the underlying architecture that must be addressed. During the two hop experiment, a large amount of packet loss was observed that can be attributed to unnecessary congestion in the network. The CSMA scheme used by the motes may be the reason for the packet loss observed.

Maybe the most limiting factor observed was the channel throughput. During the terminal to terminal tests an average transfer time of 7 sec was recorded for the non-redundant file transfer. This equated to a transfer rate of 12.3 kbps for the 299 packets. Considering that the CC2420 radio on the MICAz mote was rated for 250 kbps and the serial forwarder program was set for a transfer rate of 57.6 kbps, the cause of the slow transmission speed was unknown. Researching the TinyOS documentation led to the discovery that the MAC protocol was actually the limiting factor. The CSMA protocol limited the number of packets sent to the radio each second to 43. Implementing a TCP/IP bridge to pass multimedia or VOIP traffic would require a much higher transmission rate. While the equipment and software used in this experiment might not be able to achieve the desired results, other MAC schemes may be available that could provide a transmission rate closer to the radio's capabilities.

References

[1] "Implementing VOIP over Wireless Networks", Retrieved from the Alvarion site: http://www.alvarion.com/upload/contents/291/VoIP over wireless networks 060706.pdf Last accessed 01 Nov 2008

[2] Yow Thiam Poh, "Tunneled Data Transmission Over Wireless Sensor Networks," Master's Thesis, Naval Postgraduate School, Monterey, California, December 2007

[3] J. Jong and C. T. Ee, "Forward Error Correction in Sensor Networks," UCB Technical report, May 2003. http://nest.cs.berkeley.edu/papers/FEC_report.pdf, Last accessed 01 Nov 2008

[4] J. S. Ahn, S. W. Hong, and J. Heidermann, "An Adaptive FEC Code Control Algorithm for Mobile Wireless Sensor Networks," Journal of Communications and Networks, 7 (4), pp. 489-499, 2005. http://www.isi.edu/~johnh/PAPERS/Ahn05a.pdf, Last accessed 01 Nov 2008

[5] T. D. Norbraten, "Utilization of Forward Error Correction (FEC) Techniques with Extensible Markup Language (XML) Schema-Based Binary Compression (XSBC) Technology," Master's Thesis, Naval Postgraduate School, Monterey, California, December 2004

[6] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: A Hybrid MAC for Wireless Sensor Networks," IEEE Communications Magazine, Volume 16, Issue 3, pp. 511-524, June 2008.

[7] T. Van Dam and K. Langendoen. An Adaptive Energy Efficient MAC Protocol for Wireless Sensor Networks. In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys), Los Angeles, CA, November 2003.

[8] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. IEEE/ACM Trans. Netw., 12(3):493{506, 2004.

[9] Y. Xu, J. Xu, and W. C. Lee, "Analysis of a Loss-Resilient Proactive Data Transmission Protocol in Wireless Sensor Networks." Proc. the 26th IEEE INFOCOM '07, Anchorage, Alaska, USA, May 2007. http://www.comp.hkbu.edu.hk/~xujl/Papers/infocom2007.pdf, Last accessed 01 Nov 2008.

[10] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory, IT(13):260–269, 1967.

[11] L. Rizzo. Effective erasure codes for reliable computer communication protocols. SIGCOMM Comput. Commun. Rev., 27(2):24–36, 1997.