Asynchronous Covert Communication in Wireless Networks

Submitted To

Mr. Youngok Kim
Dr. Sanjay Shakkottai
Associate Professor, University of Texas at Austin

Prepared By

Ania Kacewicz
Adam Pridgen

EE464 Senior Design Project
Electrical and Computer Engineering Department
University of Texas at Austin

Summer 2005

CONTENTS

CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

In today's wireless network infrastructure, clandestine individuals can engage in covert communications using known attributes of these channels wireless networks. These individuals can utilize jamming as a communication technique by intentionally corrupting network traffic. In order to accomplish the illegitimate communication, these cloaked users use jamming with random coding to adapt to the traffic in the area. Individual packets are corrupted in such a way that information is encoded in the pattern of corruptions. The desired message is first preprocessed with a random code so that the covert message better blends in with the network traffic in the area.

The primary objective of this project is to develop an asynchronous covert communication protocol for wireless networks called *Jade Coin*, using the mechanism described above. After we developed this communication technique, our group wanted to observe a sender's throughput, and at the same time identify network performance impacts and how visible these impacts would become during communication.

For our design implementation, our group needed to choose several error correcting codes (ECC) that would help protect our secret message from noise, occurring in the form of naturally corrupted packets, in the channel. Also, this noise could result in an unsynchronized clock, which could occur in the event of high propagation delays between nodes. Due to time constraints, our group chose to utilize the following ECCs: repetition, hamming, and convolutional codes. In our final product, only hamming and repetition codes were fully implemented.

Due to the limitations on time, our group chose to prototype this protocol in a network simulator, more specifically, Network Simulator-2 (NS-2). To implement the protocol in NS-2, slight modifications were made to the IEEE 802.11 media access control (MAC) standard to accommodate our protocol. These modifications allowed us to send and receive packet jams as binary data. After the MAC was modified, our group implemented the *Jade Coin* protocol and developed the asynchronous clock based on traffic patterns.

During the testing and evaluation phase, NS-2 showed some limitations which debilitated the asynchronous communication; however, performance metrics could still be performed on the sender. Since we were using inefficient codes with high overhead, the throughput is heavily impacted. Also, from the results, also observed a considerable impact on the network as we varied the contention and jamming window periods.

Jade Coin is an extraordinary protocol that has a limited applications space. This technique can be applied in the intelligence community with minimal liability, but in the US, the FCC has declared radio jamming illegal. Also, like any other technology *Jade Coin* takes no sides, and the protocol could be used by hostile entities to hide their communications. Overall, this prototype is very successful. If further development is undertaken, our group recommends efficient coding schemes for reduced detection or modifying collision representations. We also recommend development for either a Linux or Unix platform. Development on these systems will ensure access to device drivers, which is essential for hacking the device at the MAC.

1.0 INTRODUCTION

The purpose of this document is to discuss relevant material about the design, development, and implementation of our novel protocol *Jade Coin*. *Jade Coin* is a covert communication protocol for use in wireless networks, specifically the IEEE 802.11 standard. This communication technique has relevant applications in the intelligence and military communities, where this activity is necessary to carry on sensitive missions. *Jade Coin* can be used as a fundamental stepping stone for building not only a more secure and robust communication solution, but this protocol can be used to begin research in the development of detection techniques for identifying anomalous noise that could be interpreted as covert communication.

Primary investigators in our group include Ania Kacewicz and Adam Pridgen. The supervising faculty member is Dr. Sanjay Shakkottai. Ania is especially fluent in Math, ECC theory, and digital communication, as well as real-time DSP communication. Ania also has an acute ability for interpreting mathematical theory applying this information to an engineering implementation. Adam is an expert in networking and security, and has taken courses in these fields at the graduate level. His academic experience includes real-time DSP communication and network protocol implementation, while his professional experience is in the area of network security, and he has been recognized by Cisco for his networking and security acumen.

In the following sections of this report, the steps taken to design, develop, and implement our covert communication protocol will be discussed. For evaluations of performance, *Jade Coin's* throughput and network impact will also be analyzed. Some safety and ethical analysis with regard to *Jade Coin's* applications will be identified and described. In conclusion, some recommendations for future progress and developments will be made.

2.0 DESIGN PROBLEM STATEMENT

Everyday new technology is developed and released with the intentions of common good or wealth; however, even though these products are developed for a particular purpose or need, these uses may ignite other ideas of application, or they could spur malicious intent to exploit this technology. In today's technology prevalent society, the population in large is vulnerable to these malicious attacks of both a physical and cyber nature. People can become victims of not only identity theft, but also violent acts of terrorism. In the recent weeks, agencies have begun to take notice of organizations spreading tactical and malicious information via the Internet, and this activity has only been noticed recently. As dedicated malicious organizations become more ambitious and advanced, what will stop them from utilizing other popular technologies such as peer to peer file sharing or our own wireless networks against us. Our motivation for the *Jade Coin* protocol is to prototype a covert communication scheme in a wireless network for future developments in detection of related anomalies and enhancements in the field intelligence communication.

Covert communication is conversation over a network, between two malicious users, which may go undetected by legitimate network users. The clandestine communication strategy takes advantage of the basic attributes found in wireless networks. Our protocol is based on inducing packet collisions in a wireless network. The communication uses a binary technique in which the malicious transmitter causes a collision, also known as a jam, between legitimate packets to signify a '1' and does nothing to convey a '0'. There can be instances, where the malicious user intends to send a '0', but a legitimate collision occurs, changing the '0' into a '1'. Many legitimate collisions can occur during periods of heavy congestion or contention. In order to mediate this noise, the secret message being sent by the clandestine node is encoded using error correcting codes (ECC) prior to transmittal.

2.1 DESIGN PARAMETERS AND SPECIFICATIONS

Jade Coin is a covert communications protocol that relies on wireless packet corruption to send messages between clandestine agents. For this reason, the project was performed in a simulator to protect us from liability issues and allow us more control over the operation and events in the network environment. This decision also helped to alleviate any stress that might have occurred when trying to debug not only the protocol and the device driver but also the host operating system.

Used assertively, the jamming can appear to the common users as congestion in the network, resulting from contention or thermal noise. Since *Jade Coin* is a covert communication protocol, the agents must operate under stringent criteria to evade detection, while delivering meaningful messages. Since our implementation required us to develop close to the hardware in the MAC layer, we decided simply modify the existing IEEE 802.11 media access control (MAC) in Network Simulator – 2 (NS-2) rather than develop our own. Other parameters and specifications used in the project can be found in Table 1.

Table 1. Jade Coin Project Specifications

Parameters	**Specifications**
Number of Users Per Simulation	4 Nodes
Frame Size	1500 Bytes
Time Slots Per Frame	32
Slot Time	20 μs
IEEE 802.11 Bandwidth Frequency	11MB/s
Frame arrival rate (λ)	1563 Frames/sec
Probability of legitimate user sending packet (p)	0.1-0.9
Probability of Error	10^{-4} b/s
Message Encoding	Error Correcting Codes and Interleaving
Clandestine Agent Message Synchronization	Pattern Marker Using Auto-Correlation (Not Completed)
Clandestine Agent Throughput	160 b/s, tentatively

3.0 DESIGN PROBLEM SOLUTION

To ensure that the effects of noise are diminished, the original message needs to be encoded with a proper error-correction scheme. On top of this consideration, this encoding scheme must appear random, since the output of the encoder will be used as the jamming pattern. The randomness and uneven distribution, where the non-jamming bit is dominant, is absolutely necessary to prevent network participants from recognizing the anomalous behavior or degraded network performance due to excessive packet collisions. The components necessary for this solution will include an error correcting and encoding scheme, as well as subtle modifications to the MAC for the conjoined layer *Jade Coin* protocol.

For the network solution, a MAC capable of jamming must be implemented, and the MAC must also be able to synchronize most of the data communication. The synchronization will assist the receiver with properly decoding the message. Inside the *Jade Coin* portion, callable functions must exist to encode and decode the data, properly mark and identify the beginning and end of the messages, and handle timeouts in a reasonable manner. Also, the *Jade Coin* protocol must have a method of acquiring, handling, and presenting any data that needs to be modified.

3.1 ERROR CORRECTING SCHEME

Initially, the plan was to find an optimal encoding scheme through the use of MatLab simulations. After performing some initial research, two error-correcting codes, which are close to achieving the Shannon Limit, were chosen to be simulated in MatLab. Cross-Interleaved Reed-Solomon Codes (CIRC) was one of the ECC schemes. These codes are intended to correct burst errors, and they are most commonly found implemented in CD and DVD players for data correction[1]. The next ECC scheme considered was Turbo codes, which are commonly used in CDMA standards such as UMTS and cdma2000. Turbo codes are a combination of two similar convolutional encoders separated by an interleaver.

3.1.1 Cross Interleaved Reed-Solomon Codes

The CIRC MatLab simulation was helpful for identifying which errors were being corrected. From our research, we found that Reed-Solomon codes were good at correcting burst errors, but after simulating an ECC method that uses two RS codes, it was clear that the error-correcting capabilities of these codes was limited. A common channel was simulated in MatLab, and generally, these errors did not occur in bursts. In a wireless channel, there are some moments where the network becomes extraordinarily congested, but in general, the occurrence of legitimate collisions is extremely random, resulting in an inefficient encoding method. Also, the use of MatLab made the analysis of RS codes much more difficult. In this environment, the RS codes are equivalent to Hamming codes in higher dimensions, resulting in RS codes using Galois field arithmetic to encode and decode. Unfortunately, in MatLab it is impossible to convert matrices in Galois field form back into a binary format. This made it difficult to compare the two ECC schemes that we decided to analyze. Thus, the only plots that could be made were in Galois form. The RS decoding in MatLab also passes back a parameter, letting the user know how many errors were corrected. This allowed plots for the number of errors corrected versus the number of errors introduced in both layers of the CIRC encoding. A drawback to this feature is introduced by MatLab when the errors in the received noisy message are not all erased. This causes the number of errors corrected to be -1, but one can just assume that the value -1 represents a case where the number of errors removed has not changed.

3.1.2 Turbo Codes

Turbo codes are composed of two convolutional codes in parallel, separated by an interleaver. A convolutional code is known for correcting independent, randomly spread errors, and for this reason, these codes can correct a variety of errors. An interleaver distributes bits in an anomalous but prescribed fashion, allowing the original data bits to be in an arbitrary order. As a result, the de-interleaving at the decoder allows burst errors to become randomly distributed in the received message, and hence, the convolutional code can optimally decode errors. This attribute makes Turbo codes exceptional in correcting

errors, because they have error-correcting capabilities for correcting both random and burst errors. The encoder structure that was simulated is shown on page 4 in figure 2. Due to the two memory registers, this encoder has 2^2, or four possible states. This quality means that the trellis diagram, which is made up of all the possible codewords, has four nodes at each possible time instant. The codewords are obtained by traversing the paths at each time instant and appropriately going to nodes according to the input sequence.

The decoding process uses a dynamic programming algorithm called the Viterbi Algorithm. This algorithm traverses trellis paths and picks the maximum- likelihood sequence. The trellis paths, also known as codeword paths, are contained in a trellis diagram. An example of a trellis diagram is shown below in figure 1.

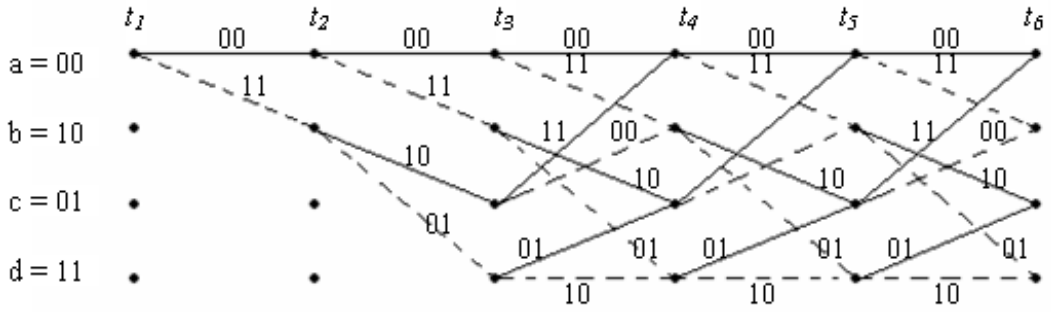

Figure 1. Trellis Diagram

Viterbi Algorithm runs through all possible codewords and determines the codeword or "path" that has the minimum Hamming distance from the received codeword. This is dynamic since the minimum Hamming distance from the source is determined for each node, and through the course of the algorithm, updates occur each time a smaller weight path is found for any node. So as the algorithm progresses, the shortest path for each particular node may have changed several times. For example, suppose there is a path of Hamming distance four from the source node (node S) to node A at time t_1, and at time t_2 when node B is reached, the Hamming distance from node S to node B is two. If the Hamming distance from node B to node A is one, then this particular path from node S to node B and then to node A is of Hamming distance three, which is shorter than the

pervious shortest path. In this case, the shortest path to node A is appropriately updated to the one that goes through node B. Since two tail bits are appended to each message prior to going into the encoder, the encoder always starts in state 00. The source node always occurs at time t = 0, and the tail bits cause 00 to become the source node. After the maximum-likelihood codeword is chosen, a trace back occurs across the trellis diagram and picks the corresponding input sequence. One drawback of Viterbi decoding with trellis diagrams is computational inefficiency. First of all, to be stored, trellis diagrams require a significant portion of memory. Also, the Viterbi algorithm runs through all possible codewords resulting in running time that is proportional to the amount of codewords. The codewords are periodic due to the limited number of states in the encoder, and fortunately, helping reduce the running time when the received message is dreadfully long.

3.2 NETWORK SOLUTION

The network solution contains two conjoined layers with different functionality. The first layer is the MAC layer, or more specifically the IEEE 802.11 protocol. Inside this layer, node jamming and listening is executed. This functionality is placed in this location, because the MAC layer is typically one layer above the hardware, meaning this location is where all the raw data is received. The next layer is where the *Jade Coin* protocol operates. At this layer, data is encoded and appended with a header and tail in the sender, and the data is auto correlated to find the header and tail and ten decoded in the receiver.

3.2.1 Media Access Control Solution

In order to facilitate the data transfer in our covert communication, the MAC must be able to corrupt the packets that are incoming on all network nodes. This concept means the principal reason for carrier sensing is being reversed. Rather than sensing to avoid collisions, the interface sensing to cause collisions. When a an incoming packet is detected, the response is to immediately send another packet, or a jam, to indicate a ‘1’, or let the packet pass through the channel uncorrupted by the clandestine node to represent a ‘0.’ At this point a ‘1’ may still occur, because another node may randomly send a packet to cause a collision, which would be registered by the receiver as a ‘1’ due to the collision.

The next function for the MAC will be communication synchronization. This is important, because, our node may not be capable of listening to every packet that passes through the wireless channel. In the solution, an elegant and efficient clock must exist between the nodes as a means to synchronize the transmission of data. Another reason for placing the synchronization methods in the MAC is due to the software delay introduced by moving this functionality further away from the hardware. One such method of synchronizing is monitoring traffic patterns for idle periods and transmission bursts. When an idle time is observed, for the first packet after the idle period to either send a jam or listen. We refer to theses as windows, where the idle period is the *contention window* and the period of time for listening or sending is the *jamming window*.

3.2.2 Jade Coin Protocol Solution

Jade Coin is the layer where data is manipulation occurs. On the sender side several functions need to take place before data is sent. First data must be acquired, either as a string or as a file. The solution should use swap files for the actual encoding and decoding, because the size of the message is arbitrary and can vary significantly. After we obtain our data, we will perform the user specified encoding scheme passed in with the data during the initialization of the covert node. Once the encoding is complete, the sender node will append a header and tail to the messaging, signifying the beginning and end of the message [2]. After this process, the sender will begin sending data through the MAC jamming functionality.

At the receiver, we will receive the secret message bit by bit and place the data into a swap file. Currently, the end of the transmission is signaled by some predetermined out of band method. Once the transmission is complete, the receiver will begin process of decoding by performing an autocorrelation to identify the start and end of the message. Once the boundaries for the message are known, the rest of the received bit stream is omitted, and the decode process specified during initialization is used on the message. The user can

supply an output file, or the data can go to a default output file. If the data received is small enough, the decoded message may also be printed to the standard output, or monitor.

A timeout function would be necessary to handle extended durations of transmission failures. The transmission failure could result from too many nodes consuming too much bandwidth or too few nodes not using enough bandwidth. In either case, a decision in the algorithm should be made about whether to abort or to continue waiting for the next transmission opportunity. If there is a significant portion of the message to still transfer and channel contention is extremely small high or small, an abort should be performed. The adaptive algorithm needs to rely heavily on the number of nodes and the collective contributed bandwidth, when creating the timeout window.

4.0 DESIGN IMPLEMENTATION

Once the research was complete, the design implementation phase began started. Based on our simulations and research, Turbo codes were chosen for the ECC scheme. The turbo encoder and decoder were going to be written in the language of C. We started writing them and realized how difficult it was to write the decoder for Turbo codes. It is complicated to write a convolutional decoder, so the Turbo recursive decoder, which can be seen in figure 2 [3], would be very difficult to implement.

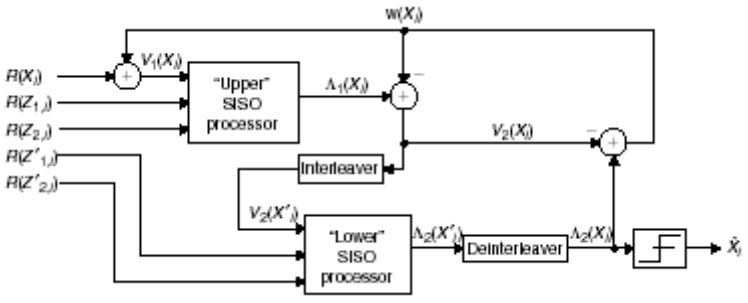

Figure 2. Generic Turbo Decoder

We spoke to our sponsoring professor Sanjay Shakkottai, and he told us to start with simple ECC such as repetition and hamming codes. He said after those were implemented to move to more complex ECC's such as convolutional codes.

Also after the research, we began modifying the MAC and developing *Jade Coin*

4.1 ERROR-CORRECTING CODE IMPLEMENTATION

The following codes are simple, yet they allow us to analyze the impact that different ECC's might have on the network performance. Also, it can be determined which ECC's have optimal skills that allow them to decode the message correctly a majority portion of the time.

4.1.1 Repetition Code

A repetition code is one which repeats the current input bit by the amount specified in the rate. If the rate is eight, and the current input bit is a '1', then that '1' is repeated eight times and transmitted. This code has excellent error-correcting capabilities if the rate is large enough, but unfortunately, with a high rate there is a tradeoff with efficiency in the transmission. If the same bit is repeated eight times, this means there is one input per every eight outputs, giving us a transmission rate of 1/8, which is significantly below the channel capacity. Another disadvantage of using this code in our covert communication method is that the transmitted message will not seem random to the legitimate users and system. If an input bit is a '1', then the malicious transmitter will jam rate amount of times and as a result will bring attention to the constant collisions in the network. It is necessary that the collisions appear like noise or congestion in the channel. It is not probable for a collision to occur numerous times in a row, since bit error rate in a wireless channel is 10^{-4}. For the decoding, the correct bit is found by finding the predominant bit found in a block of rate size.

4.1.2 Hamming Code

The Hamming code can be thought of as a check sum code. This is a one-dimensional code, meaning that everything is done on a bit-by-bit basis. We used an [8,4] code which is an extension of the [7,4] code. A [7,4] code means that for every four input bits, there are three output bits. The extended [8,4] code adds one more parity bit that is used primarily for error-detection. This last bit is the sum of all the parity bits (excluding itself) and input bits. In our implementation, we do not use the extended bit for error-detection or correction purposes, it is just there to make the decoding easier, because all variables are sizes that are multiples of eight. The parity bits are found by summing three input bits modulo two. For example, the first parity bit is equal to the sum of input bit 1, 2, and 3 modulo two, the second bit is equal to the sum of input bit 1,2 and 4 modulo two, and the third parity bit is the modulo two sum of input bit 2,3, and 4. This scheme allows for syndrome decoding. Syndrome decoding uses modulo two arithmetic to its advantage, by taking the parity bits and adding them modulo two to the input bits that formed the parity bits in the first place. If parity bit 1 is the sum of input bits 1,2, and 3, then the first syndrome bit is the sum of parity bit 1 and input bits 1,2, and 3 modulo two. Clearly, if one of the syndrome bits is not zero, then an error occurred and a particular input bit needs to be altered. Syndrome decoding can be though of as sphere-decoding. A negative aspect of this code is that it is only capable of correcting one error per every eight received bits.

4.1.3 Convolutional Code

Finally, we implemented a convolutional code. The structure of our code is shown below in figure 3.

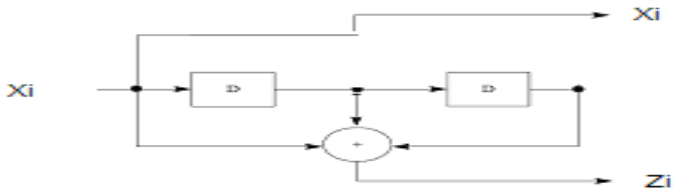

Figure 3. Convolutional Encoder Structure

This code uses Viterbi decoding, which was explained earlier in the report. The encoder is non-recursive and systematic meaning that the input bit appears at the output and there is no feedback. The rate of this convolutional code is ½, because for every input bit there are two output bits. There are four possible encoder states because there are two memory registers.

4.2 NETWORK IMPLEMENTATION

The covert communication was implemented in a simulated IEEE 802.11 network, created using NS-2. The protocol for this particular network was already written, but several key modifications were made this MAC layer protocol. These changes were made to allow the malicious users to radio jamming on user packets. To further improve the covert communication, an asynchronous clock based on traffic patterns was incorporated into the MAC functionality. As discussed earlier, two windows were used, a contention window and a jamming window.

The *Jade Coin* protocol is responsible for creating the covert communication pattern from the specified input. The protocol is conjoined to the MAC layer for fast and efficient operation. The error correction codes are applied to the original data message to form the covert message, during the encoding phase, and sent through the network as jams or no jams. In the next two sections, the key points will be discussed about the MAC modifications and Jade Coin development.

4.2.1 NS-2 IEEE 802.11 MAC Modifications

The networking solution for the covert communication required a MAC that would permit packet jamming, but when development began for *Jade Coin*, there were no such implementations in NS-2. This deficiency meant we had to develop a protocol amd modify an existing one to permit the intentional jamming or reception of collisions. After several attempts at implementing our own Slotted Aloha MAC protocol, we decided

because of concerns for time and practicality, the best step would be to modify the IEEE 802.11 MAC protocol.

The first modification incorporated into the MAC was jamming. This functionality was added by waiting for a packet to be received and then, when a reception occurred, to jam. To jam a packet, the node needs to send a packet simultaneously as one is being received. This should result in a packet collision for all nodes within a specified delta, or propagation delay plus interface delay. A collision can be observed by other nodes if this delay is less than that of transmit and delay time on the sending node. If the packet is a standard size of 1500 bytes, then the transmit time would be high enough such that jamming could almost always take place; however if the average packet sizes are small, jamming becomes a more difficult race condition to win. The method used to send a jam essentially polls the wireless network interface every 50μs to see if a packet is being received. After a jam is sent, the *Jade Coin* protocol is signaled, using a semaphore residing in static memory space.

Another modification made to the IEEE 802.11 MAC revolves around the asynchronous clock mentioned before. To accomplish this task, our covert nodes rely on two windows. The *contention window* is used to identify an idle period of data communication, but when an idle period exceeds the *contention window*, the covert nodes will prepare to listen or jam on the first packet after this time period. For example, if traffic is bursty, the nodes wait for an idle time that exceeds the *contention window*. After the *contention window*, when the first packet is transmitted, this event will signify the beginning of the *jamming window*. During this period, the sender jams or does not jam on any packet in this window, while the receiver listens until the *jamming window* expires. This listening function would act under a timer, and when the scheduled timer expired would pass the observed bit to *Jade Coin*. If a jam is observed, then a one is recorded, otherwise a '0' is recorded.

Unfortunately, due to constraints in the operation of NS-2, the asynchronous operations would only work partially. The problems stem from the fact that NS-2 is simply an event based simulator, and does not operate well when almost simultaneous events occur. After extensive debugging and investigation, we identified a very large and unexplainable delay of 1500μs, when the only delay that should have existed should have been in the area of 314μs. This limitation prevented us from comparing bit error, but since our sender primarily affected the network and performed sends, we could gauge performance and impacts based on only this node.

4.2.2 NS-2 Jade Coin Implementation

Inside *Jade Coin* there are several key components. These components consist of an encoder, decoder, autocorrelation and timeout protocol functions. Due to our time constraints, we were unable to complete the functions for timeout and autocorrelation; however, these are still part of the protocol none the less, and they are necessary for a properly working field solution. For the purposes of this projects goal, however, they will not be implemented or necessary.

For the encoder and decoder pair, we intended on allowing the covert users to choose an encoding method of their own, but there are default methods built-in to the system. This attribute was not implemented, but can be easily modified and added using filenames as direct input. The decoder and encoders we implemented into the protocol were repetition and hamming. The logic exists for the convolutional encoding method, but it is not currently included in the implementation.

The auto correlation functionality would permit us to sandwich the message in between a header and a tail. These additions would be transmitted in the clear, and then before decoding the received secret message, the receiver would find the header and tail using auto-correlation. The receiver would then decode the data between these two markers. Before implementing this procedure, we wanted to determine how much noise might exist in the environment before choosing the header and tail lengths.

The timeout algorithm was going to be significantly dependent on network topology and bandwidth being observed by the nodes. This adaptive algorithm would have been formulated to adapt to the contention of other nodes as well as network performance. Considerations for this algorithm, however, depend on channel characteristics and performance, not to mention how many nodes are sitting in the network.. The goal for this would not only minimize the necessity to re-start transmission, but also try to help maximize throughput. This function would have cloak the nodes even further, but our time ran short and we excluded this feature from the list.

Other features were also necessary for the implementation of *Jade Coin* and the MAC. One of the major limitations built into this system is the inability to handle that is not a multiple of 8. This concept was simply used to ensure an easy to debug system at first, and then we would repair this in the future. Another problem is the separation of the *Jade Coin* and the IEEE 802.11 MAC. To get around the separations, commands were added to Jade Coin permitting the passing of pointers to both the *Jade Coin* layer and the IEEE 802.11 MAC. For example, *Jade Coin* would directly receive a jam, and the MAC would directly send a jam on behalf of the *Jade Coin* protocol. These were necessary to combat confusion and reduce delay by sidestepping some of the NS-2 architecture. In order to implement these direct calls, the objects were linked via pointers, but these pointers can be considered extensions allowing the protocols to interoperate quickly and efficiently.

5.0 TEST AND EVALUATION

Testing of the software was performed on two fronts. All of the software was tested vigorously, but as modifications were introduced, new bugs appeared as older ones were found. In the case of the encoding functions, this software was introduced as a callable function, so testing could be separate and verified by hand, in the cases of the hamming and convolutional codes. However some bugs were introduced as this software was migrated into the *Jade Coin* implementation.

For the test and evaluation, our group originally intended to observe not only the network performance impacts and sender throughput, but we also intended to focus on bit error rate by comparing the output of the receiver to the input of the sender. As mentioned before, NS-2 could not handle events in a more asynchronous fashion, so the bit error test became infeasible to perform before our project deadline. We also could not perform the test, because errors would be artificially injected into the final output, reducing the merit of our results. However, since we were confident in the sender's functionality, the throughput and network impacts can be observed and evaluated.

5.1 TESTING

Functionality testing of *Jade Coin* followed a very different approach. We first ran simulations using the TCL scripts, which are the simulators primary interface. These scripts contained key functional commands that helped us determine if jams and no-jams were being sent. These scripts also helped test and validate input for what we wanted to do, but there was no security testing to hunt down and identify possible buffer overflows.

Debugging the *Jade Coin* implementation required built-in debugging statement that could be turned on and off with C-language pragmas. These statements were a seires of printf() functions identifying protocol state, time, and variable values. During this debugging process several mistakes were identified. The first one was the initialization of the swap files. After the first iteration, the file would retain this data output this rather than the new input. The debugging helped us identify the limitations in NS-2. By knowing what our delay should be from propagation delay, interface transmit time, and actual transmit time, we found that NS-2 could not handle our asynchronous implementations due to the event scheduling. However, this process took many hours to identify and validate the cause.

5.2 EVALUATION CRITERIA AND RESULTS

To conduct our performance evaluations, we sent the secret message "bah bah sheep" across the network using only the repetition encoding method. The secret message breaks down into 13 bytes with 40-'1' bits and 64-'0' bits. In the testing phase the nodes were

placed on top of each other to reduce the propagation delay to essentially zero, but in our evaluation simulations our nodes were spread out in a more practical configuration, as shown above in figure 4.

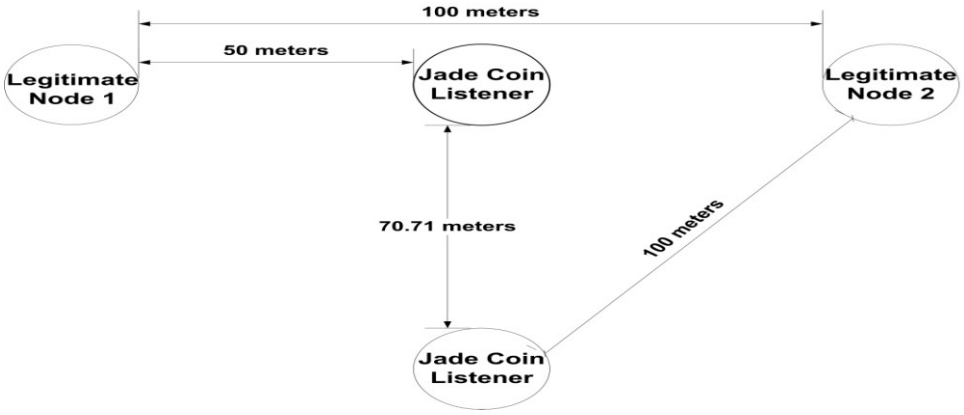

Figure 4. Node Positioning for Evaluation

For the senders encoding method, we decided just performing the repetition codes were enough to satisfy the performance and network impact evaluation, so we varied the coding rates by factors of 8, so the rates used were 8, 16, and 32. If any attempts use a higher coding rates were made, the test would not complete in a sufficient amount of time and the results would also be difficult to compare. The first simulation conducted set the jamming window at 100μs and the contention window at 60μs, and the second simulation set the jamming window at 360μs and the contention window at 100μs. The results from both tests can be found on this and the next page in tables 2 and 3 respectively.

Table 2. Transmission Rate: J. Window=100μs and C. Window=60μs

REPETITION RATE	SIMULATION RUN TIME	SENDER THROUGHPUT	TRANSMISSION RATE
8	2.01 s	413.93 bits/sec	51.74 bits/sec
16	6.62 s	251.36 bits/sec	15.67 bits/sec
32	34.55 s	96.32 bits/sec	3.01 bits/sec

Table 3. Transmission Rate: J. Window=360µs and C. Window=100µs

REPETITION RATE	SIMULATION RUN TIME	SENDER THROUGHPUT	TRANSMISSION RATE
8	6.25s	133.12 bits/sec	16.64 bits/sec
16	85.48s	19.47 bits/sec	1.21 bits/sec
32	969.45s	3.43 bits/sec	0.10 bits/sec

In our final results, there is an obvious gain in the throughput as the contention and jamming window are decreased. However, this throughput will come at a cost, where the overall network performance is impacted. Since there are only two legitimate nodes communicating, the affects of the covert communication becomes readily apparent in figure are exaggerated even more. In figure 5 on the next page, a dramatic rate of change is shown, as the code rate is increased from 8 to 16 bits.

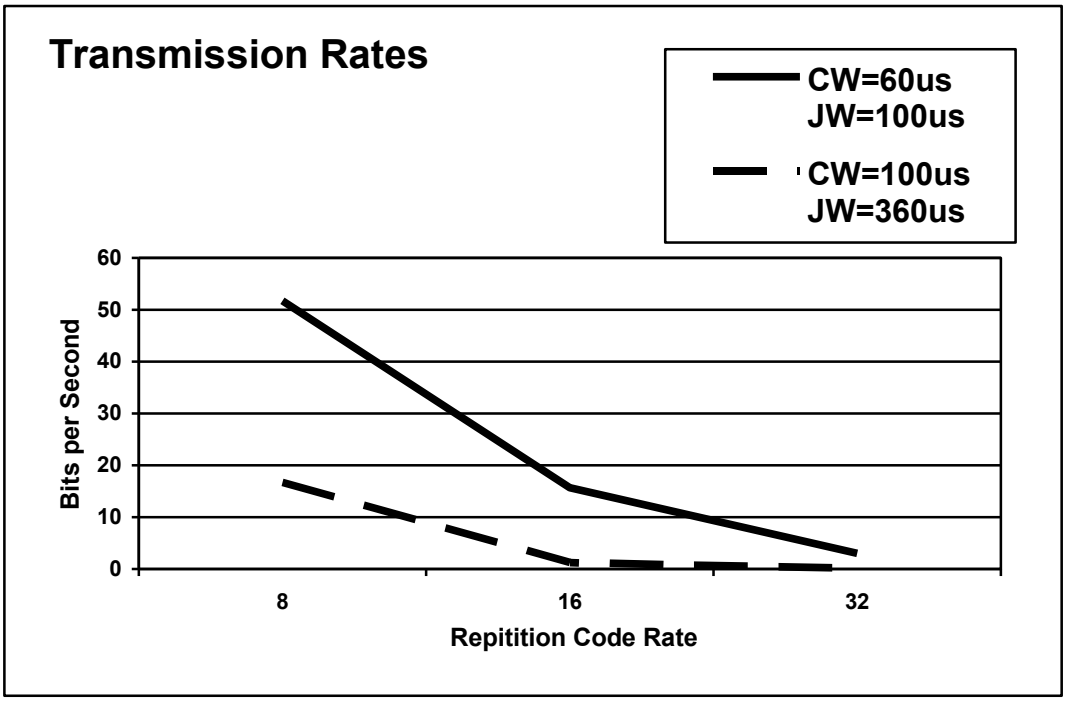

Figure 5. Transmission Rates Comparison

The phenomenon occurs because of the exponential back-off implemented in IEEE 802.11 protocol, also found in IEEE 802.3. This back-off was developed to prevent more than one collision from occurring successively. When the collision happens, the nodes experiencing the collision perform an exponential back-off. This process involves the host waiting for a random period time where the period of time is dependent on the contention of the wireless medium. This contention window is incremented during node collisions and decremented during idle periods [2]. This contention window must reach zero before a retransmit may occur.

6.0 TIME AND COST CONSIDERATIONS

Since summer sessions are limited in time, we had a few unrealized goals. A proper time-out function was not implemented, and even though the convolutional encoder/decoder was written, it was not employed in NS-2. Optimally, we would have liked to append a pseudo-noise marker to the beginning of each covert message. Then, the receiver would be able to determine the beginning of the message by using an autocorrelation function and properties of psuedo-noise sequences. Turbo codes have optimal properties since they are able to get within half a decibel of Shannon's Capacity, and unfortunately, we were not able to execute this code in software [3].

7.0 SAFETY AND ETHICAL ASPECTS OF DESIGN

Jade Coin protocol is an extraordinary design in terms of safety and ethical issues due to its nature. There is no true ethical application for this protocol. *Jade Coin* is used to hide communication, and in the process of hiding this communication, *Jade Coin* can disrupt network services paid for by legitimate users. Since *Jade Coin* is used for disguising communication, intelligence services may find the application worthwhile; however, the same features that appeal to these intelligence services will also appeal to hostile entities or terrorist organizations. This protocol definitely is safe physically for people and animals, but the application could be indirectly detrimental to human life.

Some ethical concerns raised by our products stems from uses by unauthorized citizens. These people may unlawfully disrupt wireless network traffic. By FCC regulations, radio jamming is illegal, so people with no authority would be intentionally breaking the law. If this technique is applied in public, customers paying to use wireless network services may be impacted. Their service could unfairly reduce the bandwidth that they have paid for and are entitled to use. As mentioned before, this protocol has a strong application in the intelligence community, which could be used indirectly to sustain national security.

8.0 CONCLUSIONS AND RECOMMENDATIONS

In this project, we achieved some fantastic results fro design and implementation. Our evaluations also showed that this is a viable stepping stone for further research by intelligence agencies. From our work, we have shown that two covert entities can communicate and pass small messages through a wireless network. We developed an asynchronous clock from the traffic patterns of the network, and then developed software encoders to protect the covert messages from the very corruption being induced in the channel.

From the results of our project, we feel safe in making our recommendations for further development. While the prototype of this protocol is not complete with timeouts and message markers, this protocol is ready for real system testing. Some improvements are definitely necessary for this system and should include a more efficient encoding scheme.

Turbo Codes would be an excellent code for this application, because this technique applies a convolutional code to the message, runs the result through an interleaver, and then through a convolutional encoder. This process would most certainly be effective for randomly distributing bits in the original message, but the receiving system will need to be capable of performing Viterbi Algorithm, due to the computational intensity of the Markov Modeling.

Another possible modification to the encode-and-jam pattern would be to dynamically decide whether to send '1' or '0' bits as jams. This decision could be sent along with the header an tail unencoded, so the receiver could decode the message. This reversal of bits would help alleviate any excessive jamming in short periods of time, which reduces the possibility of being detected and caught. This could also increase throughput by reducing the number of exponential back-offs that must take place to send the message.

If this communication scheme is ported to a mobile machine, this *Jade Coin* should be implemented on an open source operating system, because this factor will help ensure access to open source wireless networking device drivers. Access to these drivers is essential, because access to the physical layer of the network is required to jam communication, and this access is controlled through the MAC. The device driver is one of the closest layers to the hardware, but some reverse engineering may be necessary to dodge the embedded firmware on the card itself.

Microsoft Windows is not an option for two prevalent reasons. First, the time investment for hacking through Windows software is poor, due to the many layers the hacker will need to cross through just to get to the hardware abstraction layer. The other main point involves the fact that it is closed source and obfuscated, with little documentation for assistance.

REFERENCES

[1] Reed-Solomon Codes: *An introduction to Reed-Solomon codes: principles, architecture and implementation*, http://www.4i2i.com/reed_solomon_codes.htm.

[2] C. R. Johnson Jr. and W. A. Sethares, *Telecommunication Breakdown*, New Jersey: Pearson Education, Inc., 2004.

[3] K. Park, "IEEE 802.11 MAC," http://www.cs.purdue.edu/homes/park/cs536-wireless-3.pdf, August 2005.

[4] M. C. Valenti and J. Sun, "Turbo Codes," Ch. 12 of *Handbook of RF and Wireless Technologies*, http://www.csee.wvu.edu/~mvalenti/documents/DOWLA-CH12.pdf.

[5] Flemming, Chip, "Tutorial on Convolutional Coding with Viterbi Decoding," http://pweb.netcom.com/~chip.f/Viterbi.html.