

Dynamic Jamming Mitigation for Wireless Broadcast Networks

Jerry T. Chiang Yih-Chun Hu
Department of Electrical and Computer Engineering
University of Illinois, Urbana-Champaign
1406 W. Green St., Urbana, IL 61801-2918
{chiang2, yihchun} @uiuc.edu

Abstract—Wireless communications are inherently symmetric; that is, it takes an attacker the same amount of power to modulate a signal as it does for a legitimate node to modulate the same signal. As a result, wireless communications are often susceptible to the jamming attack in which the attacker injects a high level of noise into the system. Spread spectrum has long been used to resist jamming attacks in unicast environments, or when the jammer has less information than the other receivers. Recently, we proposed a scheme for broadcast jamming mitigation based on spread spectrum and a binary key tree and showed some improvements over a multiple-unicast system. In this paper, we extend our previous work in five ways. First, we provide a theoretical result that under our scheme, jammers can cause only a limited number of losses. Second, we develop a dynamic tree-remerging scheme that achieves higher power efficiency than previously proposed schemes, and scales to an arbitrary number of receivers without increasing the number of codes in use; in particular, we send each transmission on at most $2j + 1$ codes, where j is the number of jammers. Third, we show that our scheme is close to optimal, demonstrating that under certain realistic restrictions, the system cannot escape jamming without using at least $j + 1$ codes. Fourth, we provide a detailed analysis of false alarm rates, showing both experimental and theoretical results. Finally, we perform a more extensive analysis of our system using both a chip-accurate MATLAB simulation and a bit-accurate event-driven simulation in the *ns-2* network simulator; these simulations demonstrate that our scheme approaches the best possible performance.

I. INTRODUCTION

Wireless communications are inherently symmetric; that is, it takes an attacker the same amount of power to modulate a signal as it does for a legitimate node to modulate the same signal. As a result, wireless communications are often susceptible to the jamming attack in which the attacker injects a high level of noise into the system. By injecting noise into the channels, a jammer effectively reduces the *signal to noise and interference ratio* (SNIR), thereby reducing the probability of successful message reception.

An effective countermeasure against jamming is *spread spectrum* [9], where a transmitter redundantly encodes information using a *code*, allowing a receiver to reject signals that do not come from the transmitter. One common spread spectrum system is *code division multiple access* (CDMA), which spreads data across a wide frequency band in a manner described by a code. Two distinct implementations of CDMA are *direct-sequence* (DS-) and *frequency-hopping* (FH-) CDMA.

In a DS-CDMA system, each bit is mapped to either 1 or -1 and each *pseudo-random code* is of length η . To send a 1 bit, the sender transmits the pseudo-random code, and to send a -1 bit, the sender transmits the additive inverse of the code. To decode a bit, the receiver takes the inner product of the code and the signal it received; if the inner product is positive, then a 1 bit was sent, and if the inner product is negative, then a -1 bit was sent. (Intuitively, the inner product computes the correlation between the input and the code). The DS-CDMA system relies heavily on the property of *orthogonality*. Two vectors are said to be orthogonal if their inner product is zero. Moreover, two pseudo-random codes of sufficient length have been shown to be asymptotically orthogonal [9]. Messages sent on orthogonal codes do not interfere with each other, because during decoding, if one sender sends bit b_1 on code c_1 and another sender sends bit b_2 on code c_2 , then the message received by any receiver is $\alpha_1 b_1 c_1 + \alpha_2 b_2 c_2$, where α represents the *path loss* from each sender to the receiver. A receiver using code c_1 will compute $c_1 \cdot (\alpha_1 b_1 c_1 + \alpha_2 b_2 c_2) = \alpha_1 b_1 (c_1 \cdot c_1) + \alpha_2 b_2 (c_1 \cdot c_2) = \alpha_1 \eta b_1$, which is not affected by the transmission of b_2 .

In an FH-CDMA system, each user is assigned a set of *frequency-hopping patterns*, according to which the user hops between frequency channels from time slot to time slot. As long as two users use different channels in each time slot, the two users can transmit without interfering with each other.

Spread-spectrum codes are inherently *symmetric*; that is, the sender and receiver use the same information for encoding and for decoding. Though spread spectrum systems can be highly effective against jamming in a point-to-point system, they are more easily subverted in a broadcast system, because each potential broadcast recipient must share the code, and once the jammer acquires the code, any advantages provided by spread-spectrum are erased. We recently proposed a tree keying scheme to circumvent jamming in a broadcast system [3] by using a balanced binary tree where each node of the tree corresponds to a spread spectrum code. Each user is assigned a leaf and has access to the codes corresponding to that leaf and all its ancestors. For example, in Figure 1, user N_2 would have access to codes C_2 , C_{23} , C_{03} , and C_{07} . Our system transmits on a set of codes such that all users can decode using exactly one code; such set is referred to as a *disjoint cover*. Besides transmitting on the disjoint cover, the system

also transmits on a set of *test* codes. We call a code in the disjoint cover a *detectable* code if it is the ancestor of any of the test codes. For example, if the current disjoint cover in use is $\{C_{03}, C_{45}, C_{67}\}$, then the test code set of $\{C_{01}, C_4\}$ would make the detectable set be $\{C_{03}, C_{45}\}$. If any user receives message on a test code but not on the corresponding detectable code, then that user reports jamming on the detectable code, and the system responds by removing the detectable code from the cover and inserting its two children codes in its place. For example, in Figure 1, when jamming is detected on code C_{07} , the transmitter splits the cover into $\{C_{03}, C_{47}\}$. If jamming is further detected on C_{47} , the resulting cover would be $\{C_{03}, C_{45}, C_{67}\}$.

For purposes of simplicity, we will restrict the description of our work to its application in DS- and FH-CDMA systems. However, our scheme can be generalized to any spread spectrum system with bidirectional communication including orthogonal frequency-division multiplexing (OFDM), and can be applied to any existing networks that use spread spectrum technologies such as IEEE 802.11 [5], Bluetooth [8], and IS-95 cellular system [7].

In this paper, we extend our previous work in five ways. First, our previous work does not show that the scheme eventually stops jamming; here we provide a theoretical result that shows that our previously proposed scheme can mitigate jamming after a limited number of losses. Second, we develop a dynamic tree-remerging scheme that achieves higher power efficiency than previously proposed schemes and scales to an arbitrary number of receivers without increasing the number of codes in use; in particular, we send each transmission on at most $2j+1$ codes, where j is the number of jammers. The optimized remerging scheme also has the property of mitigating jamming after a limited number of losses. The first two results are given in Section III. Third, we show that our scheme is close to optimal, demonstrating that under certain realistic restrictions, the system cannot escape jamming without using at least $j+1$ codes; we prove this in Section IV. Fourth, because false alarms can result in a substantial departure between our strong theoretical results and actual real-life performance, we analyze false alarms using both experimental and theoretical techniques. In Section V, we demonstrate that false alarms can be reduced to negligible levels while retaining a reasonable level of detection. Finally, in Section VI, we perform a more extensive analysis of our system using both a chip-accurate MATLAB simulation and a bit-accurate event-driven simulation in the *ns-2* network simulator; these simulations demonstrate that our scheme approaches the best possible performance, as shown by our results in Section IV.

II. RELATED WORK

Interference prevention using CDMA has been studied at length [9], and random codes of sufficient length have been found to be asymptotically orthogonal [9]. We rely on this property and choose random codes and assume that they are orthogonal. Other physical layer techniques, such as the use of multiple antennas, have also been studied, but those do not

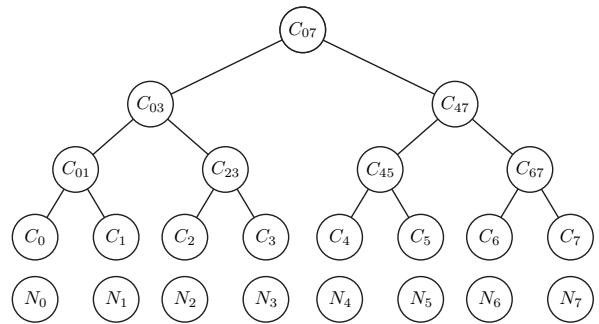


Fig. 1. Example Code Tree

make use of higher-layer feedback and are orthogonal to our approach.

The effectiveness of jamming [2] and the difficulty of differentiating jamming from congestion [11] have previously been discussed, but the authors do not propose solutions to traverse the jammed area. In particular, Xu et al [11] try to detect and avoid jammed regions.

We recently proposed a tree keying scheme that achieves asymmetry in a broadcast wireless network [3]. However, no theoretical justifications were given about the validity of the protocol. Section I describes the new contributions in this paper.

III. TREE REMERGING SCHEME

A. Types of Covers

When there are no jammers, a transmitter can transmit on a single code; specifically, it would choose the code corresponding to the root of the key tree. Transmissions on this code can be decoded by any legitimate receiver. For example, in Figure 1, the transmitter would transmit on code C_{07} . In general, in order to ensure that every receiver can decode the packet while ensuring power efficiency, the transmitter wants to transmit on a set of codes such that any user can decode using exactly one code in the set. We call such a set a *disjoint cover*. Once jamming has been detected on some codes, the transmitter should avoid using such codes in the future. Given a set of codes J on which jamming has been detected¹, we call a disjoint cover C *safe* relative to J if $C \cap J = \emptyset$; that is, if C contains no codes on which jamming was previously detected. Because each code used for transmission either increases the power consumption or reduces the received signal strength, we want to transmit on the smallest possible safe set. We call a safe disjoint cover *minimal* if no safe disjoint cover has fewer codes in it. In particular, for all codes $c \in C$, either c is the root or $\text{parent}(c) \in J$. We define κ to be the set of all non-leaf codes known to the jammer. Table I summarizes the definitions in this paragraph.

B. Loss Due to Jamming Is Limited

In an idealized environment, for any $\varepsilon > 0$ we show that any user will lose, due to jamming, at most a finite number of messages with probability higher than $1 - \varepsilon$.

¹We assume J never includes leaf codes, because each leaf code is known only to a single receiver. Thus jamming the leaf code is equivalent to random jamming.

TABLE I
TYPES OF COVERS

Term	Definition	Explanation
Cover	$\forall \text{leaf } c, C \cap \text{ancestors}(c) \neq \emptyset$	Each receiver can decode at least 1 code
Disjoint	$\forall c_1, c_2 \in C, c_1 \notin \text{ancestors}(c_2)$	Each receiver can decode at most 1 code
Safe	$C \cap J = \emptyset$	No previously jammed code is used
Minimal Safe Cover	$\forall c \in C, c = \text{root} \vee \text{parent}(c) \in J$	No smaller safe cover exists

Theorem III.1. *For every normal user u and every $\varepsilon > 0$, there exists a finite loss limit λ such that u will lose at most λ packets with probability at least $1 - \varepsilon$, in the following idealized environment:*

- 1) *Jamming is successful on a code only if a jammer knows that code (that is, jamming cannot overcome the processing gain)*
- 2) *Each subset of the leaves of the tree is a possible test set (that is, each subset is tested with non-zero probability)*
- 3) *When only normal users know a test code, and jamming is perpetrated on the corresponding detectable code, at least one normal user detects the jamming with probability at least $p > 0$ (that is, jamming detection sometimes works)*

This idealized environment is not necessary for correct protocol behavior, but is merely provided to formally characterize the properties of our protocol. Section VI shows simulation results from a realistic environment.

Proof: Intuition. The full proof is omitted due to space constraints; we present the intuition of our proof here. We consider first the set of packets on which the jammer might attempt jamming (with non-zero probability). We show that for each such packet, every time jamming results in packet loss to a normal user, it also results in a probability $p\alpha > 0$ of being caught. As a result, for every $\xi > 0$, there exists a finite loss limit L such that before a single jammed code is detected, a normal user will lose at most L packets with probability at least $1 - \xi$. Because a jammer knows a finite number of codes, it can be detected a limited number of times, and each detection has this loss limit, so the aggregate system has a loss limit with probability $1 - \varepsilon$ for all $\varepsilon > 0$. ■

C. Remerging Trees

In this section, we describe a scheme that allows a transmitter to split and re-form a tree to reduce the number of codes in the disjoint cover. We will show that even with such remerging, jamming will be detected at most $j \lceil \log_2 n \rceil$ times before the jammer can no longer affect normal users. (The proof of Theorem III.1 shows that this limits the losses any user can experience). We will also show that a transmitter performing this remerging scheme sends on at most $2j + 1$ codes simultaneously.

a) Intuition: Instead of keeping a disjoint cover, each transmitter keeps a set of trees, \mathcal{R} . The cover consists of the root of each of these trees (and is clearly disjoint), and \mathcal{R} is chosen such that jamming has not been detected on any root in \mathcal{R} (which shows it is safe). When jamming is detected on the root of some tree T in \mathcal{R} , we remove T from \mathcal{R} and insert its two subtrees, T_L and T_R , into \mathcal{R} .

We define a binary tree to be *full* if the number of leaves is 2^h , where h is the height of the tree. When a tree is non-full, we require its empty leaves to be flush to the right, as shown in Figure 2(a). To improve power efficiency, two trees can sometimes be merged together, thereby reducing the cardinality of the cover. In particular, two trees T_1 and T_2 can be *safely* merged (that is, such reinsertion will not nullify the jamming-prevention aspects of our protocol) when they are both *sibling-free*; that is, neither T_1 nor T_2 has their most recent siblings in \mathcal{R} . This is because T_1 and its most recent sibling T_1' must have been simultaneously added to \mathcal{R} due to jamming at the root of their shared parent T_1^p . Once we know that one jammer is in T_1' , we treat the detection of jamming at T_1^p as having come from a jammer in T_1' , even if a jammer exists in T_1 . The safety of this approach is proven in Lemma III.2.

To insert a full tree T into a non-full tree T' , T is placed into an appropriately-sized gap in T' , and subtrees are swapped as necessary, as shown in Figure 2(b).

b) Detailed Description: Our algorithm for keeping the set of trees \mathcal{R} is as follows: we start with the collection containing only the original tree. When jamming is detected on the root of a tree, the root is deleted and the two children of the tree are inserted into the set of trees. Because we insert a tree into \mathcal{R} only when the parent of the root of this tree has been jammed, the roots of \mathcal{R} form a minimal safe disjoint cover.

If a subtree T is in \mathcal{R} , but its most recent sibling $T' = \text{sibling}(T)$ is not in the set of trees, then T' must contain a jammer. This is because the jamming event that caused T to be placed in \mathcal{R} must also have caused T' to be placed in \mathcal{R} . The fact that T' is no longer in \mathcal{R} indicates that jamming was detected on the code at the root of T' ; that is, the root of T' is known to a jammer. It is therefore possible to remerge all sibling-free subtrees T into a new tree; Lemma III.2 will show that even with such remerging, jamming will be detected at most $j \lceil \log_2 n \rceil$ times before the jammer can no longer affect normal users.

To ensure that the resulting remerged trees are balanced, we impose requirements on the empty subtrees contained in the tree. Any binary tree can be represented as a complete binary tree on which every leaf node is at equal depth. Some of these leaf nodes will be empty; if their sibling is also empty, the parent will likewise be empty. In general, we call a subtree *empty* if every leaf of that subtree is empty. We call a tree *full* if it contains no empty subtrees.

We represent each non-full tree in this way, and build it such that a tree of height h has at most one empty subtree of each height from 1 to $h - 1$ when each empty subtree is counted only once (that is, a subtree of height h' is not counted as two subtrees of height $h' - 1$). In addition, we do not allow an empty subtree of height $\lceil \log_2 n \rceil$ or greater, where n is the number of legitimate users. This ensures that no tree can be of height greater than $\lceil \log_2 n \rceil$, because there are at most n leaves in any tree.

To keep insertion efficient, we ensure that all the empty leaves are contiguous and at the far right. This ensures that, when viewed from right to left, there is at most one empty

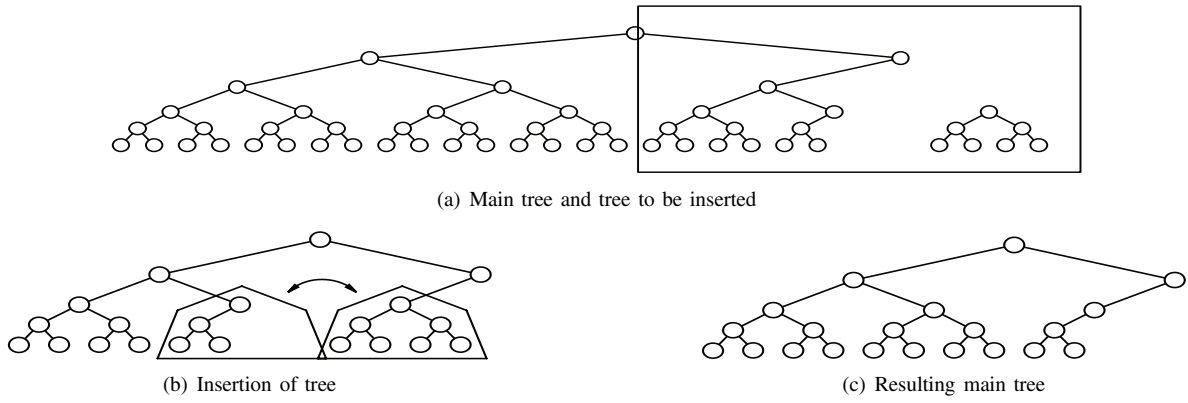


Fig. 2. Merging Code Trees. Figures 2(b) and 2(c) show the right half of the tree in Figure 2(a).

subtree of any height (because any two empty subtrees of the same height would be next to each other, and would therefore constitute an empty subtree of a height one greater than either subtree). Also, the heights of the empty subtrees are in descending order. All our algorithms will require the tree to start this way, and will return the tree to this condition when complete. Our tree operations may change the ancestors of some tree nodes; in this case, we randomly generate new codes for all affected ancestor nodes, and disseminate the new codes as described later in this section.

To insert a full tree T' of height h' into a tree T of height h with e empty leaves, we first determine whether or not T' will fit (that is, whether $e \leq 2^{h'}$). If not, we increase the height of T by adding an empty subtree of height h to the right of T , making T have height $h+1$ and $e+2^h$ empty leaves. We then recursively apply this algorithm.

At this point, an empty subtree of height at least h' exists. Figure 2(a) shows an example of a tree being merged into the main tree. We insert T' into the leftmost place that it will “fit;” that is, on the left side of the smallest empty subtree of height at least h' (it may have height greater than h' , in which case we will take an empty subtree of height $h'+\delta$ and turn it into δ empty subtrees of heights $h', h'+1, \dots, h'+\delta-1$). The leftmost side of this tree will be the $e - (e \bmod 2^{h'})$ -th leaf, counting from the rightmost leaf. In our example, there are 10 empty leaves, and $h' = 2$, so the leftmost leaf of the inserted tree is placed in the $10 - (10 \bmod 2^2) = 10 - 2 = 8$ th position counting from the right, as shown in Figure 2(b). To maintain the property that all empty leaves reside at the right, if there are any empty leaves to the left of where T' was inserted (that is, whether $e \bmod 2^{h'} > 0$), we swap the root of T' with the node at the same level that is one to the left of the root of T' . We know that at most one such swap is necessary to ensure that all empty leaves are on the right hand side, because if there were more than $2^{h'+1}$ empty leaves to the left of T' , we would have placed T' farther left. In our example, this swap is shown in Figure 2(c).

To insert a non-full tree T into a tree T' , we break T into maximally-sized component subtrees and add each such subtree into T' . When the root of a non-full tree is found to have been jammed, we split the tree in half and insert both the left child

and the right child into the tree set \mathcal{R} . If an empty subtree is in \mathcal{R} , we leave it in \mathcal{R} until its sibling is removed from \mathcal{R} ; at that point, we discard the empty subtree rather than remerging it.

c) Limit on Number of Transmitted Codes: We call a tree $T \in \mathcal{R}$ *main* if T does not have a sibling in \mathcal{R} . A tree set can only contain one main tree; otherwise, the two trees that both do not have siblings could merge. To see that a transmitter sends on no more than $2j+1$ codes, we observe that every pair of siblings in \mathcal{R} must contain at least one jammer (because jamming was detected on the parent of those two siblings). This means that excluding the main code, there are at most $2j$ codes in \mathcal{R} , for a total of $2j+1$.

d) Practical Limitations: When new tree nodes are created, new codes are also created. To disseminate these codes, the transmitter periodically broadcasts new ancestor codes on codes that were part of the original tree. For example, if c was a code on the original tree, and the main tree has new ancestors for c , the transmitter will broadcast the codes of those new ancestors using the code c , so that all users with code c can learn the new ancestor codes. Because of mobility and varying wireless propagation, these broadcasts need to be repeated periodically.

Lemma III.2. *When tree remerging is done in the manner described in this section, after $j \lceil \log_2 n \rceil$ detections, the jammer is unable to jam further (that is, it will only know roots of \mathcal{R} that are leaves).*

Proof: Intuition. The full proof is omitted due to space constraints; we present the intuition of our proof here. Given a set of jammers, we define a cost and potential for a tree set \mathcal{R} . The cost of a single tree is the height of the tree times the number of jammers in the tree, and the cost of a tree set is the sum of the costs of the trees in the set. That is, $\text{cost}(\mathcal{R}) = \sum_{T \in \mathcal{R}} \text{height}(T) \cdot \text{jammers}(T)$. The potential of a tree in \mathcal{R} depends on whether or not the sibling of a tree is in \mathcal{R} . If the tree is main (that is, it does not have a sibling in \mathcal{R}), its potential is $\text{potential}(T) = (\lceil \log_2 n \rceil - \text{height}(T)) \cdot \text{jammers}(T)$, where n is the total number of legitimate receivers. Otherwise, the tree T and its sibling T' together have potential $\text{potential}(T, T') = (\lceil \log_2 n \rceil - \text{height}(T)) \cdot (\text{jammers}(T) + \text{jammers}(T') - 1)$.

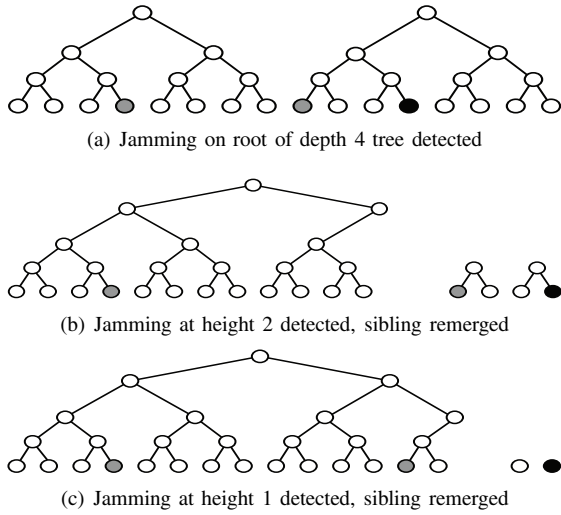


Fig. 3. Illustration of Potential and Cost of Merging Trees (from the proof of Lemma III.2). Darkened nodes correspond to jammers. In this illustration, after jamming is detected $\log_2 n$ times, the solid jammer is isolated.

Whenever jamming is detected, the splitting and remerging process reduces the cost plus potential of the tree set by at least 1. Because the tree starts with $j \lceil \log_2 n \rceil$ cost and 0 potential, jamming can be detected at most $j \lceil \log_2 n \rceil$ times. ■

IV. LOWER BOUND ON NUMBER OF CODES REQUIRED

Because a transmitter must divide its transmission power amongst all the codes in the cover, smaller covers are better as long as they exclude all jammed codes. Our tree remerging scheme ensures that no more than $2j + 1$ codes are ever used (assuming no false alarms); in this section we show that any scheme that provides both availability and termination must use at least $j + 1$ codes in the worst case. By *availability* we mean that each non-attacking node must eventually know at least one code in the cover, and that node must not have reported that code as being jammed. Intuitively, we want to ensure that the non-attacking node is able to receive each broadcast on an unjammed code. By *termination* we mean that the cover eventually converges; that is, given any specific jammer strategy, there exists a maximum number of changes to the set of codes used, above which the transmitter makes no further changes to the cover. Both our previously proposed scheme and our tree remerging scheme achieve termination: if a code c in the cover is known to a subset of nodes $\mathcal{N}_c \subseteq \mathcal{N}$ and is reported as being jammed, then never again is another code c' included in the cover when $\mathcal{N}_{c'} = \mathcal{N}_c$. Because the power set of nodes $2^{\mathcal{N}}$ is finite, the cover eventually converges.

Theorem IV.1. *In a network of n nodes where we do not a priori know the number of attackers, if we allow the network to exclude at most $k - 1$ non-attacking nodes, then j jammers can force the use of $\lfloor \frac{j}{k} \rfloor + 1$ codes.*

Proof: In network A , we consider j jammers a_1, a_2, \dots, a_j ; these jammers divide into sets of k elements

$s_1 = \{a_1, \dots, a_k\}, s_2 = \{a_{k+1}, \dots, a_{2k}\}, \dots, s_{\lfloor \frac{j}{k} \rfloor}$. All jammers always jam, and a jammer $a_f \in s_g$ reports only when the detectable code c is known to a jammer that is not in s_g . Let R_A denote the set of reports made in network A , so that $(N, r) \in R_A$ if and only if N made a report r in network A .

In network B , we consider k normal users $s'_g = \{a'_{gk+1}, \dots, a'_{g(k+1)}\}$ with the rest being jammers who jam continuously, and report exactly that subset of R_A that they can report. (Specifically, if $(N, r) \in R_A$ and $N \notin s'_g$, then N reports r in network B , so $(N, r) \in R_B$). Furthermore, because every cover element is jammed, a normal user $a'_f \in s'_g$ will report jamming on detectable code c whenever c is known to any of the jammers (that is, known to a node not in s'_g).

The set of reports R_B made in network B is exactly equal to the set of reports R_A made in network A , so networks A and B are indistinguishable to the sender. Furthermore, every element of s_g is a jammer, but every element of s'_g is a normal user, and each set is indistinguishable from every other set. Therefore, the transmitter cannot exclude any set s_g or s'_g . To ensure availability, the transmitter must send on a different code for each s_g , plus send on one additional code for all the normal users, for a total of $\lfloor \frac{j}{k} \rfloor + 1$ codes. ■

We require availability for every single non-attacking node, thus $k = 1$, and the minimum number of codes that is required for j attackers is $j + 1$. The requirement for termination means that R_A and R_B are finite and therefore the jammers can force the use of $j + 1$ codes in finite time. This shows that our scheme, which uses $2j + 1$ codes in the worst case, is within a factor of two of optimal in power usage. In particular, given equal jammer power and noise level, a network with an optimal algorithm and some transmitter power level p cannot perform better than a network using our algorithm and power level $2p$. Our evaluation in Section VI also shows that in a practical setting, the performance of our scheme approaches the performance of an optimal scheme.

V. MITIGATING THE IMPACT OF FALSE ALARMS

Due to the probabilistic nature of packet reception, a signal may be lost even when there is no jamming. When a jammer raises the interference level, the probability of loss increases further. Such losses raise the prospect of *false alarms*: because our scheme detects jamming when the detectable code is lost and the test code is received, the loss of an unjammed detectable code would result in a false alarm. Such losses result in a larger-than-necessary cover, since $J \not\subseteq \kappa$, non-leaf codes known to jammer. Due to the physical effects of simultaneous transmissions, the presence of excess codes in the cover results in reduced power efficiency.

One option for coping with the effects of false alarms while achieving higher power efficiency is to periodically empty J . Emptying J resets the state of our system, allowing jammers to deny service again. In particular, we reset the loss counter in Section III, so that rather than bounded losses across the entire lifetime of the system, we only limit the number of losses over a fixed period of time. In the rest of this section, we explore the trade-off between false alarms and missed detection, and

show that parameters exist for which a low false alarm and a reasonable detection rate are achievable, which increases the amount of time between when J needs to be emptied.

We define the false alarm rate as the probability of detecting jamming on a code when the jammer does not jam that particular code, and we define the detection rate as the probability of detecting jamming on a code when the jammer jams that code.

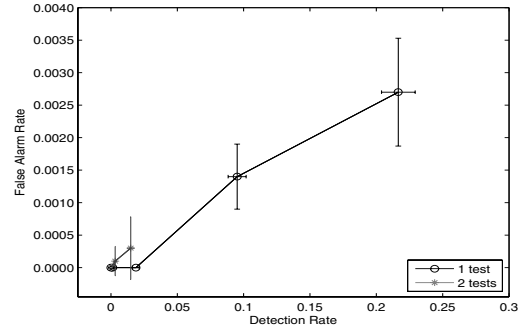
To determine the false alarm rate, we consider the worse of two scenarios: first, when the jammer does not emit any power, or second, when the jammer jams on a random code. In the latter case, the jammer effectively raises the noise floor, increasing the probability of loss on the detectable code, which in turn increases the probability of false alarm. As a result, we only evaluate false alarms in the worst case; that is, when the jammer is jamming another code.

We propose a general testing algorithm, of which our previous testing method is a special case, then we analyze the possibility of having a reasonable detection rate while achieving a negligible false alarm rate. Instead of assigning each user only one leaf code, we now allow them to hold k leaf codes. We modify the detection scheme such that a user reports jamming when a message is lost on cover but is successfully received on at least p out of k leaf codes. Our previous testing scheme is a special case where $k = p = 1$.

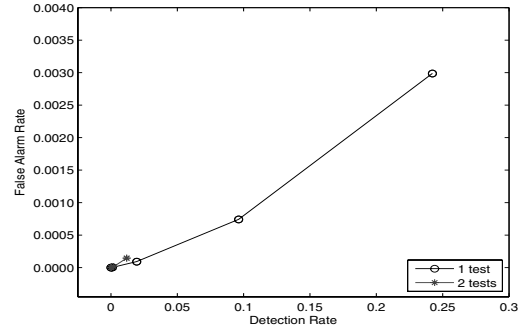
It is well known that a jamming signal in a DS-CDMA system, after despreading, can be approximated as a random variable with Gaussian probability distribution [9]. Noise is generally modeled as Additive White Gaussian Noise (AWGN), and the desired signal is a single fixed value, so the sum of the signal, the noise, and the jamming signal is a normally distributed random variable. Let the length of codes be η , and the signal to interference and noise ratio be $SNIR$; then the message bit, normalized with respect to the signal power, is distributed $\approx N(\eta, \eta (\frac{1}{SNIR})^2)$. Whenever the random variable is negative, the bit is decoded incorrectly. Thus the bit error probability is $P_e^{bit} \simeq Q(\frac{1}{SNIR}\sqrt{\eta})$. If a message is ℓ -bits long, and all ℓ bits must be successfully received, and if we allocate a fraction r of the transmission power to the cover and divide the remaining $1 - r$ evenly amongst the test codes, then the false alarm rate is given by P_F :

$$\begin{aligned}
 P_d^{cover} &= \left(1 - Q\left(r \frac{1}{SNIR} \sqrt{\eta}\right)\right)^\ell = \Phi^\ell\left(r \frac{1}{SNIR} \sqrt{\eta}\right) \\
 P_d^{test} &= \sum_{i=p}^k \binom{k}{i} \left(\Phi^\ell\left(\frac{1-r}{k} \frac{1}{SNIR} \sqrt{\eta}\right)\right)^i \\
 &\quad \left(1 - \Phi^\ell\left(\frac{1-r}{k} \frac{1}{SNIR} \sqrt{\eta}\right)\right)^{k-i} \\
 P_F &= (1 - P_d^{cover})(P_d^{test})
 \end{aligned}$$

where P_d^{cover} is the probability that the cover message is received, and P_d^{test} is the probability that at least p out of k test messages were received. To calculate detection probability, we assume P_d^{cover} is negligible, (that is, the probability of



(a) Simulated false alarm rate vs detection rate



(b) Theoretical false alarm rate vs detection rate

Fig. 4. False Alarm Rate vs Detection Rate. For clarity, the x- and y-scales are different.

receiving a cover message correctly when the jammer knows the code is negligible; this assumption matches our simulation results). The detection rate is approximately $P_D \simeq P_d^{test}$. We performed a Monte Carlo simulation for the case where $p = 1, k = 1, 2, \dots, 10, SNIR = \frac{1}{15}, \eta = 256, l = 64, r = 0.75, 0.8, 0.85, \dots, 1.00$. The results were consistent with those predicted by above analysis and are shown in Figure 4. Because the probability of detection and probability of false alarm are negligibly small for $k \geq 3$, we show the results for $k < 3$. Each point represents 10 runs of 1000 messages, each 64-bits long; we plot the mean and 95% confidence intervals. Because each data point represents two variables (probability of detection and probability of false alarm), each point includes two error bars that indicate the confidence interval of each variable. For clarity, the x- and y-scales of this figure are different. We use one data point to show that the false alarm rate can in fact be much smaller than the detection rate. At $k = 1, r = 0.75$, the false alarm rate is around 0.3% while detection rate is around 25%.

VI. EVALUATION

In this section, we describe the results of two simulation-based evaluations. In the high-level simulation, we assumed the codes were completely orthogonal, and precisely modeled the received signal and noise levels after processing gain, using the resulting signal-to-noise ratio to select a bit error rate. In the low-level simulation, we modeled the use of random codes

TABLE II
SUMMARY OF SIMULATION PARAMETERS

Simulator	<i>ns-2</i>
Mobility	Random Waypoint
Pause Time	0
Maximum Speed	20 m/s
Traffic Pattern	20 CBR sources, 4 packets/s
Packet Size	512 bytes of data per packet
MAC Protocol	None
Modulation Scheme	QPSK
Processing Gain (η)	16, 32, 64, 128, 256, 512, 1024
Simulated Time	2000 s per run
Runs	10 per parameter set

and tree remerging in order to ensure that the nearly orthogonal random codes provided sufficient interference cancellation, and that the tree remerging scheme does not substantially reduce the effectiveness of jammer detection.

A. High-Level Simulation Methodology

Our high-level simulation was based on the *ns-2* discrete-event network simulator [1]. We implemented the following features into *ns-2*: simultaneous sending of a single packet on multiple codes, simultaneous reception of multiple packets on multiple codes, and our test code selection algorithm [3] (starting at a depth 4 in a tree of depth 6). We conducted our simulations on a 1500 m \times 300 m area with 50 users.

For each incoming packet, we computed the maximum interference power experienced during the reception of the packet, including power contributed by any outgoing packets, the jammers, other incoming packets, and ambient noise (conservatively computed as half of the IEEE 802.11b carrier sense threshold from *ns-2*). We then computed the signal-to-noise ratio, including processing gain from the original packet, the energy cost of simultaneously sending on multiple codes, and any processing gain that the jammers receive as a result of transmitting on the same code. Based on this signal-to-noise ratio and the modulation scheme in use, we determine the resulting symbol error rate p . We then computed the probability that the packet would be lost, which is given by $1 - (1 - p)^{\frac{s}{b}}$, where s is the length of the packet in bits and b is the number of bits per symbol, and accepted the packet with this probability.

We chose processing gains from 16 to 1024 to demonstrate the trade-offs in choice of processing gain. Our modulation scheme was QPSK; we also considered 16-QAM and 64-QAM in preliminary experiments, since those modulation schemes are used for higher data rates (those above 18 Mbps) in IEEE 802.11a [5], [6]. However, preliminary experiments showed that decreasing the processing gain and bits-per-symbol proportionally so as to achieve the same data rate would result in better performance, so we used QPSK for all of our experiments.

We determined our bandwidth based on a 80MHz-wide channel, equivalent to four 802.11a channels. This should be achievable with modern hardware, since 802.11a cards using two channels simultaneously have been on the market for several years. In addition, the 5.8GHz ISM band used by

802.11a provides for 12 non-overlapping channels. A modulation scheme that gives us b bits per symbol combined with processing gain η gives us a bit rate of $\frac{b \cdot 40 \times 10^6}{\eta}$. In QPSK, $b = 2$.

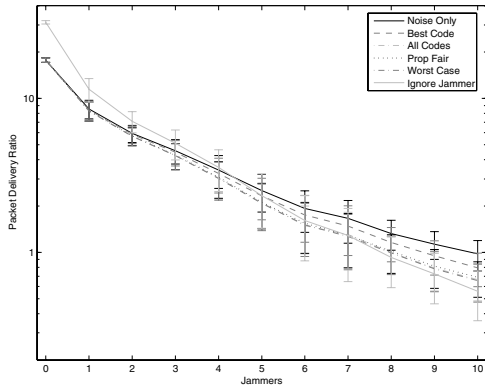
Our communications pattern was 20 broadcast streams, each sending 4 packets per second containing 512 bytes of data. This traffic rate could be used for highly compressed audio or for navigation data. We selected our malicious users from users that were not transmitting in our communications pattern, thereby avoiding the possibility that a jammer might jam its own packets. We performed experiments for a number of jammers between 0 and 10.

We considered five jammer strategies in which jammers *collude* and share all codes amongst themselves. Each strategy is executed on a per-packet basis. In each strategy, the attacker determines the subset of its codes on which the transmitter sent that packet. The jammer then allocates its power between that subset of codes². In the first strategy, the jammer jams on a code that is orthogonal to all transmitted codes. This represents a best-case scenario, which cannot be improved upon by any code selection algorithm. We call this the “Noise Only,” and any jammer that has compromised any user should be able to perform better. The second strategy represents a jammer that jams on a single code per transmission, and chooses a code to jam that covers the most potential victims (that is, one that is closest to the root of the tree). In the event that multiple codes are close to the root, it chooses only one such code. We call this the “Best Code” jammer, because it allocates all of its power to the single best code. The third strategy is to jam all codes in that subset with equal amounts of power, which we call the “All Codes” strategy. The fourth strategy is to proportionally share the jamming power among all codes in the subset, where each code gets an amount of power proportional to the number of users potentially affected. The fifth strategy is the impossible (and worst-case) strategy where the jammer provides full jamming power to *each* code in the subset, which we call the “Worst Case” strategy. We also consider the “Worst Case” strategy when the sender uses no probe codes, which we call “Ignore Jammer.”

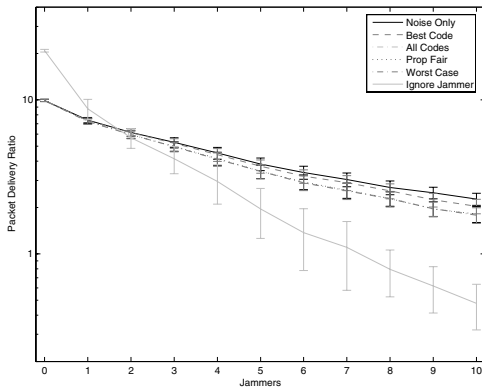
We did not use the MAC protocol from 802.11 because 802.11 defers transmission when it senses another transmission. A jammer within the carrier sense range of any device, then, would be able to prevent that device from transmitting, effectively preventing communication. We chose to use a simple (null) MAC protocol which passes each packet without modification, buffering, or delay between the network layer and the physical layer. All computations to determine collision and transmission delay are performed in the physical layer of the simulator.

For simplicity of evaluation, in our simulation, each jammer transmits at a power level equal to the power level of a legitimate transmitter. However, our results also show what

²We consider a particularly strong attacker model where the attacker knows all codes currently in use; in practice, a jammer may not be able to hear each packet that it attempts to jam, and thus cannot determine the set of codes in use.



(a) Packet Delivery Ratio, $\eta = 32$



(b) Packet Delivery Ratio, $\eta = 512$

Fig. 5. Simulation Results.

happens at increased jammer power. In particular, if jammers transmit with twice as much power, the performance will be no worse than if there were twice as many jammers, because the aggregate jamming power level will be the same, but in the latter case, the jammers will know more of the code tree.

For each processing gain, jamming strategy, and number of jammers, we performed 10 simulation runs of the described scheme, each of which represented 2000 simulated seconds.

B. High-Level Simulation Results

Figure 5 shows the results of our simulation runs. Each graph shows the *packet delivery ratio* (PDR), which is the number of packets received divided by the number of packets sent. Each PDR result is averaged over 10 simulator runs; the error bars represent the 95% confidence interval of this average. Because our communications pattern is broadcast, a single transmitted packet may be received by several different users. In particular, out of 50 users, any user other than the transmitter and the jammers could potentially receive the packet. However, because not all users are within wireless transmission range, the packet delivery ratio cannot reach 49, even when there are no jammers.

Figure 5(a) shows the performance of our scheme when $\eta = 32$. At the left side of the graph, where there are few jammers, ignoring the jammer substantially outperforms the other

four schemes; this is because when ignoring the jammer, no probe codes are sent, doubling the effective transmit power and thereby increasing range. As the number of jammers increases, the strategy of ignoring the jammer becomes less advantageous; however, the difference between ignoring the jammer and avoiding codes used by the jammer remains slight, even at 10 jammers. This is because avoiding jammers requires spreading one’s transmission power across multiple codes. At $\eta = 32$, the improved jamming rejection (a factor of 32) does not substantially overcome the reduced effective transmission power. The effect of small η on jamming rejection is even more pronounced when $\eta = 16$; though we do not present the results in detail due to space limitations, in those scenarios, ignoring the jammer outperforms all other approaches, even with 10 jammers.

Figure 5(b) shows the performance of our scheme when $\eta = 512$. As when $\eta = 32$, the “Ignore Jammer” approach provides substantial benefits when there are no jammers. However, as the number of jammers increases, our scheme provides substantial advantages regardless of jammer strategy. In fact, when $\eta = 512$, our scheme provides *almost all of the performance improvement possible*, because any scheme choosing CDMA codes must necessarily perform below the “Noise Only” line, and any reasonable scheme will perform above the “Ignore Jammer” line for a sufficiently large η and number of jammers. (When Forward Error Correction is used, higher delivery rates can be achieved, but the use of FEC is orthogonal to the approach described here, and is therefore applicable to all schemes, including ours and “Noise Only”).

Our simulation environments show performance results in a wideband setting (a 80 MHz-wide channel at 2.4 GHz). Ultra-Wideband [10] (UWB) provides for extremely high symbol rates and uses extremely high processing gain. For example, Intel’s Wireless USB [4] uses 528 MHz-wide channels and substantially greater processing gain than we used in our simulations. When applied to an UWB, our approach provides additional ability to reject jammer interference, which would bring the network environment closer to the ideal environment assumed in Theorem III.1.

C. Low-Level Simulation Methodology

In order to determine the effectiveness of using random codes and the tree remerging scheme, we implemented our scheme including these two elements (which are missing from the high-level simulation above) in MATLAB. The simulation scenario consists of one base station, 50 normal users, and 0 to 10 jammers. The total jamming power at each receiver is equal to the number of jammers times the total received base station power; as before, jammers that emit more power can be modeled by increasing the number of jammers. To make decoding more challenging, we assumed a noise level that is 15dB higher than base station power. We simulated FH-CDMA with 127 channels and 63 hops per bit across those channels. In this simulation, jammers did not collude (the high-level simulation considers collusion); rather, each jammer simply jams on its hopping pattern in the cover. For each number of jammers, we performed 10 tests in which 10,000 6-bit packets are transmitted by the base station.

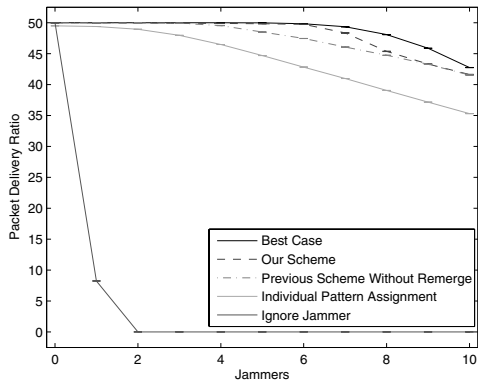


Fig. 6. Simulation Result

D. Low-Level Simulation Results

Figure 6 shows the results of our low-level simulation. Again, we computed the packet delivery ratio by dividing the number of packets received by the number of packets sent. For each jamming strategy and number of jammers, we plotted the average and 95% confidence interval of the packet delivery ratio. Because we had 50 normal users in each scenario (in addition to the transmitter and jammers), and because all normal users are within wireless transmission range of the transmitter, the best possible result is a packet delivery ratio of 50. This graph shows that for any number of jammers, our scheme's performance is very close to the optimal performance where the system transmits on $j + 1$ hopping patterns, and is nearly perfect when there are fewer than 6 jammers. Our previously proposed scheme allows us to operate at 20% higher PDR than when each user is assigned an individual hopping pattern. Our remerging scheme provides additional benefits over our previously proposed scheme. The effect of tree remerging is greatest for a medium number of jammers; when the number of jammers is small, the increased code usage does not yet substantially affect the ability of receivers to decode the packet; when the number of jammers is large, both systems tend to use many leaf codes, substantially increasing the size of the cover.

The performance results are consistent with the expected result. The best case line reflects the use of $j + 1$ codes; our remerging scheme uses $2j + 1$ codes, our old scheme uses $j \lceil \log_2 n \rceil$ codes, and the scheme that allocates each user a different pattern uses n codes. Not only do we provide statistically significant improvements with just 50 nodes, the improvements would grow even larger if the network had more users.

These results are somewhat better than those of the high-level simulation for five reasons: first, every receiver is within the transmission range of the transmitter, thus the performance is better even without any jammers; second, jammer and transmitter powers are well-balanced at each receiver (that is, there are no near-far problems which overwhelm certain victims with excessive jamming power); third, jammers do not collude, which can reduce the effectiveness of jamming; fourth, this simulation has a higher per-code transmission power due to tree remerging; finally, this simulation uses much shorter packets,

reducing the packet error rate at the same symbol error rate.

VII. CONCLUSIONS

Our paper contains five main contributions. First, we showed that in a theoretical environment, jammers can cause only a limited number of losses. Second, we developed a dynamic tree-remerging scheme that scales to an arbitrary number of receivers without increasing the number of codes in use, and uses fewer codes than previous schemes, even in relatively small networks. Third, we showed that our remerging scheme is within a factor of two of optimal, since no system can escape jamming without using at least $j + 1$ codes. Fourth, we analyzed false alarms using both experimental and theoretical techniques and demonstrated that false alarms can be reduced to negligible levels while retaining a reasonable level of detection. Finally, we performed an extensive analysis of our system using both a chip-accurate MATLAB simulation and an bit-accurate event-driven simulation in the *ns-2* network simulator; both simulations show that our scheme approaches the best possible performance.

Our results showed that asymmetry can be imposed even in the symmetric environment of broadcast transmission and reception. We also showed that jamming can be efficiently and effectively detected and mitigated in a wireless broadcast network. These techniques will be even more promising in the future, as the use of Ultra-Wideband expands.

REFERENCES

- [1] *ns* notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. Available from <http://www-mash.cs.berkeley.edu/ns/>.
- [2] Timothy X Brown, Jesse James, and Amita Sethi. Jamming and sensing of encrypted wireless ad hoc networks. Technical Report CU-CS-1005-06, University of Colorado at Boulder, 2006.
- [3] Jerry T. Chiang and Yih-Chun Hu. Cross-layer jamming detection and mitigation in wireless broadcast networks. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 346–349, New York, NY, USA, 2007. ACM.
- [4] Intel Corporation. Wireless USB: The first high-speed personal wireless interconnect. Available at <http://www.intel.com/technology/comms/wusb/download/wirelessUSB.pdf>, 2005.
- [5] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.
- [6] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, High-speed Physical Layer in the 5 GHz Band*, IEEE Std 802.11a-1999. The Institute of Electrical and Electronics Engineers, New York, New York, 1999.
- [7] J. S. Lee. Overview of the technical basis of qualcomm's cdma cellular telephone system design: a view of north american tia/eia is-95. In *Proc ICCS '94*, volume 2, pages 353–358, November 1994.
- [8] Bluetooth SIG. Bluetooth specification documents. Available at <https://www.bluetooth.org/spec/>, 2003-2006.
- [9] A. J. Viterbi. *CDMA Principles of Spread Spectrum Communication*. Addison-Wesley, 1995.
- [10] M. Z. Win and R. A. Scholtz. Ultra-wide bandwidth time-hopping spread-spectrum impulse radio for wireless multiple-access communications. *IEEE Transactions on Communications*, 48(4):679–689, April 2000.
- [11] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of The 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2005)*, pages 46–57, May 2005.