SigSag: Iterative Detection through Soft Message-Passing

Arash Saber Tehrani, Alexandros G. Dimakis, and Michael J. Neely
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089, USA
email: {saberteh, dimakis, mjneely}@usc.edu

***Abstract*—The multiple-access framework of ZigZag decoding (Gollakota and Katabi 2008) is a useful technique for combating interference via multiple repeated transmissions, and is known to be compatible with distributed random access protocols. However, in the presence of noise this type of decoding can magnify errors, particularly when packet sizes are large. We show that ZigZag decoding can be seen as an instance of belief propagation in the high-SNR limit. Building on this observation, we present a simple soft-decoding version, called SigSag, that improves performance. We show that for two users, collisions result in a cycle-free factor graph that can be optimally decoded via belief propagation. For collisions between more than two users, we show that if a simple bit-permutation is used then the graph is locally tree-like with high probability, and hence belief propagation is near-optimal. Further, we introduce the joint channel-collision decoding which decodes the collided packets while the packets are coded by an LDPC code. Through simulations we show that our scheme performs better than coordinated collision-free time division multiple access (TDMA) and the ZigZag decoder. Furthermore, we investigate the performance of the joint channel-collision decoder in different scenarios and show that it performs better than TDMA and ZigZag decoder accompanied by sum-product decoding.**

EDICS: SPC-DETC, WIN-CONT.

I. Introduction

Despite the substantial amount of theoretical work in multiuser detection and interference cancellation, most implementations rely on carrier sense (CSMA) to limit collisions while CDMA receivers decode each user by treating interference as noise. One important step towards practical systems that decode interfering users was ZigZag decoding by Gollakota and Katabi [2] that relies on the 802.11 MAC. This restricts the design space into a very simple repetition of packets when there are collisions and decoding failures. We build on the same assumptions as the original ZigZag framework. However, we develop a soft-decoding version, called SigSag, that is just as simple to implement but results in significantly improved reception rates.

Specifically, we assume there are N users, each having a packet of B bits, trying to communicate with an access point (AP) (See Fig. 1). Each user relies on carrier sensing to detect if other users are transmitting. If this fails (the case of a *hidden terminal*) there is interference at the AP, modeled by a simple linear superposition of the symbols plus noise. In this paper we only consider the worst case, where carrier sensing constantly fails and there are always packet collisions. The improved decoding probabilities we obtain in this paper can be used to improve performance of higher layer multi-access protocols that incorporate ZigZag-like collision frames, such as those in [24]. Under the 802.11 protocol, each sender retransmits its packet until it receives an acknowledgment that the AP successfully decoded it. In our model, each of the N users transmits its packet N times and the AP receives linear equations involving the sum of the collided symbols plus noise. Similar to prior work, we model the random delay offsets (jitter) of the 802.11 protocol as follows: As packets from two or more users collide, each user chooses a random bit-offset uniform over $[0, W]$, where W is a positive integer, and transmits its packet again. The maximum likelihood (ML) detection (also referred to as a multiuser demapping) problem consists of finding the most likely user symbols given the noise statistics. In the high-SNR case (when the noise is negligible) this simply reduces to solving linear equations, while for the noisy case it becomes a statistical inference problem which is well known to be computationally intractable.

Our contributions: We show that the ZigZag algorithm can be seen as an instance of belief propagation in the high-SNR limit, essentially attempting to solve linear equations by back-substitution only. We introduce a new algorithm called SigSag that *exploits* and *maintains* soft information about each symbol. If ZigZag is interpreted as hard-decision belief propagation, our algorithm is a natural generalization that maintains likelihoods and runs in a loopy manner on the factor graph created by the linear equations formed by collided packets.

We show that for $N = 2$ users, our iterative soft message-passing algorithm is optimal with high probability. Specifically, we establish that the corresponding factor graph is cycle-free with probability $1-O(1/B)$. Further, for $N \geq 3$ users, we show that the factor graph is locally a tree by establishing that the local neighborhood is cycle-free with high probability. Our results establish that while the maximum-likelihood inference problem is NP-hard in general, the random jitter bit-offsets and bit permutations in packets create easy instances with high probability. For these instances the maximum likelihood

This work appeared in part at the IEEE INFOCOM conference, Shanghai, China, April 2011 [1]. This work was supported in part by one or more of the following: the DARPA IT-MANET grant W911NF07-0028, the NSF grant CCF-0964479.

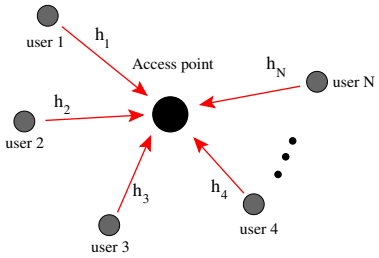

Fig. 1. All nodes send data to the access point (AP) together.

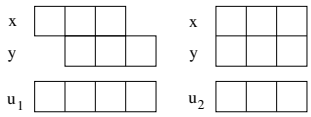

Fig. 2. Two consecutive collisions of two packets $\mathbf{x}$ and $\mathbf{y}$ of $B = 3$ symbols sent by $N = 2$ users. Both packets are transmitted twice and the AP receives $\overline{\mathbf{u}}_1$ and $\overline{\mathbf{u}}_2$.

solution is found by our soft message-passing detector. The last part of our theoretical analysis shows that random jitter and fading can critically influence performance even in the high-SNR regime.

We further introduce the *joint channel-collision decoding* for the transmitted packets coded by LDPC codes. The joint channel-collision decoding is an instance of belief propagation over the joint factor graph formed by combining the Tanner graph of the LDPC codes and the factor graph resulting from the collision of the packets (collision graph).

Our preliminary experimental analysis shows the substantial practical benefits of using SigSag soft message-passing compared to ZigZag and Time Division Multiple Access (TDMA), also often referred to as collision free scheduling. Furthermore, we investigate the effect of different scheduling schemes in performance of the joint channel-collision decoding.

A. System Model

As shown in Fig. 1, we have N users trying to communicate to an access point (AP). Each user re-transmits N times and the packets collide forming linear equations at the AP. We assume that the users can permute the bits of their packets before each transmission, as discussed in [12]. We further assume a block fading channel model where data sent by user i on the c^{th} transmission are attenuated by coefficients $h_i^{(c)}$ assumed known at the receiver. Note that for estimating the fading coefficients $h_i^{(c)}$ at the receiver, the 802.11 adds a known preamble sequence to each packet. Further, for this estimation to be exact, we require fading to be extremely slow. Under this model, the AP receives the signal

$$\overline{\mathbf{u}}_c = \mathbf{u}_c + \nu^{(c)} = \sum_{i=1}^{N} h_i^{(c)} \mathbf{T}_{w(i,c)}(\mathbf{x}_i) + \nu^{(c)}, \qquad (1)$$

where $c \in \{1, 2, \ldots, N\}$ is the collision round, $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,B}]$ is the packet (assuming BPSK, $x_{i,j} = \pm 1$) sent by the i^{th} user, $\nu^{(c)}$ is the channel noise vector of the c^{th} collision, assumed to be independent and identically distributed Gaussian for simplicity. Further, $\overline{\mathbf{u}}_c$ is the noisy data received at the access point on the c^{th} collision. Finally, $\mathbf{T}_{w(i,c)}(\mathbf{x}_i)$ is an operator that takes the B dimensional vector $\mathbf{x}_i$ and creates a $B + W$ dimensional vector by padding zeros to the beginning and end, where the number of padded zeros at the beginning is determined by the time delay (jitter) user i chooses randomly before the c^{th} transmission. We assume that the access point knows the jitter. This can be achieved as explained in [2]. As a result the whole system can be modeled as $A\overline{\mathbf{x}} + \nu = \overline{\mathbf{u}}$ (with $A\overline{\mathbf{x}} = \mathbf{u}$), where A is the *collision matrix*, $\overline{\mathbf{x}} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, and $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]^T$.

As an example, consider the case shown in Fig. 2 which depicts two consecutive collisions of the packets sent by $N = 2$ users. Notice that each collision results in at least B linear equations. As explained, the whole system of linear equations can be rephrased in the form of $A\overline{\mathbf{x}} = \mathbf{u}$, where here $\overline{\mathbf{x}} = [\mathbf{x}, \mathbf{y}]^T$ and $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2]^T$.[1] The set of linear equations corresponding to the collision patterns shown in Fig. 2 are

$$\begin{aligned}
\overline{u}_{1,1} &= h_1^{(1)} x_1 + \nu_1^{(1)} \\
\overline{u}_{1,2} &= h_1^{(1)} x_2 + h_2^{(1)} y_1 + \nu_2^{(1)} \\
\overline{u}_{1,3} &= h_1^{(1)} x_3 + h_2^{(1)} y_2 + \nu_3^{(1)} \\
\overline{u}_{1,4} &= h_2^{(1)} y_3 + \nu_4^{(1)} \\
\overline{u}_{2,1} &= h_1^{(2)} x_1 + h_2^{(2)} y_1 + \nu_1^{(2)} \\
\overline{u}_{2,2} &= h_1^{(2)} x_2 + h_2^{(2)} y_2 + \nu_2^{(2)} \\
\overline{u}_{2,3} &= h_1^{(2)} x_3 + h_2^{(2)} y_3 + \nu_3^{(2)}
\end{aligned}$$

and the corresponding collision matrix A is

$$A = \begin{bmatrix}
h_1^{(1)} & & & & & & \\
& h_1^{(1)} & & h_2^{(1)} & & & \\
& & h_1^{(1)} & & h_2^{(1)} & & \\
& & & & & & h_2^{(1)} \\
h_1^{(2)} & & & h_2^{(2)} & & & \\
& h_1^{(2)} & & & h_2^{(2)} & & \\
& & h_1^{(2)} & & & & h_2^{(2)}
\end{bmatrix}.$$

Notice that if there was no noise ($\nu^{(c)} = \mathbf{0}$), the optimal decoder would simply have to solve these linear equations. In the presence of noise, optimal decoding would correspond to finding which vectors $\mathbf{x} \in \{\pm 1\}^B$, $\mathbf{y} \in \{\pm 1\}^B$ have the highest likelihood under the noise statistics, which, for general N, reduces to a computationally intractable integer least-square problem.

II. Related Work

The amount of related work in communications and information theory on multiuser joint detection and decoding is overwhelming and without trying to be comprehensive we will point the interested reader to [3], [5], [9]–[11], [13], and references therein. A comprehensive overview on the recent advances in interference cancellation can be found in [15].

ZigZag decoding [2] is the closest to our work. The approach taken in [2] is to restrict the design space to packet

[1] For more clarity, here we call the two packets by $\mathbf{x}$ and $\mathbf{y}$ instead of $\mathbf{x}_1$ and $\mathbf{x}_2$.

repetitions and hence maintain compatibility with 802.11. One important advantage of this approach is that, unlike *joint decoding* methods, senders do not need to reduce their rate in order for the AP to be able to decode the collisions.

By exploiting the small variations in the transmission times, ZigZag attempts to decode the packets of each user by solving the linear equations $A\bar{\mathbf{x}} = \mathbf{u}$ by back-substitution. This version of ZigZag (the Forward ZigZag) is identical to the belief propagation decoder for the BEC (see e.g. [6], [8]), also known as the leaf-stripping decoder which looks for a degree one variable, makes a hard decision (decides either ± 1) about it and removes it from the factor graph. This algorithm is feasible because the random jitter allows some symbols to appear without interference (i.e. have degree 1) with high probability. An improved version of ZigZag runs this algorithm from different starting points (Forward-Backward [2]) and then combines partial soft-information from the two executions to decode the variables. Our algorithm is essentially an optimized version of this idea that keeps track of how this soft information evolves and defers the rounding decisions until the end. Remap [12] is an extension of ZigZag where senders permute their data after the first transmission and Chorus [14] resolves collisions over a *multi-hop* network to improve latency and transmission diversity of the network.

Our work is closely related to wireless network coding (see e.g. [17], [19], [22]) and cross-layer designs that exploit coding [18]. In our case, it is the interference that creates the code, similarly to [21]. Our technical analysis relies on iterative message passing algorithms and the theory developed for sparse-graph codes [6], [8], [16], [20].

A. Shortcomings of ZigZag

In this paper we extend ZigZag decoding [2], addressing two main drawbacks:

1) ZigZag can fail to decode when back-substitution is not possible.
2) Noise is accumulated as the ZigZag decoder advances through the packet.

1) Back-substitution Failure: While it is often possible to decode even when back-substitution cannot be initiated (see examples in Section IV-D and IV-E), ZigZag requires back-substitution and fails if this cannot be done. This causes a threshold on the bit error probability that ZigZag can achieve. This threshold depends on the number of collided packets N and the maximum waiting time before each transmission W, but not on the SNR.

Consider the case of $N = 2$ users where each user waits for a random time uniformly chosen from $\{0, 1, \ldots, W\}$ before each transmission. Assuming the users have similar SNR's[2], the ZigZag decoder fails to retrieve the packets if the two collision patterns are identical. In other words, if we define $w_i^{(c)}$ for $i \in \{1, 2\}$ to be the waiting time of user i before its c^{th} transmission, then the patterns will be identical if

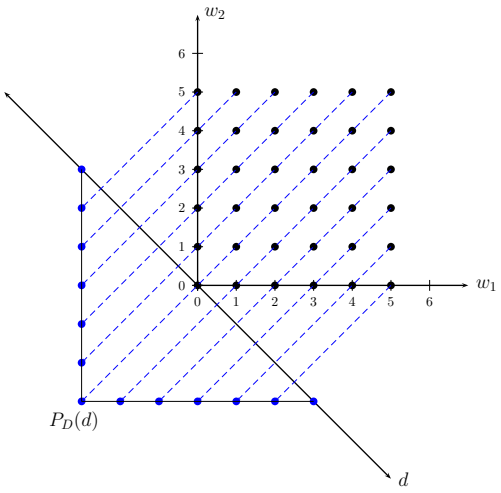

Fig. 3. The probability mass function of $d = w_1 - w_2$ when w_i's are discrete random variables uniformly chosen from $\{0, 1, \ldots, W\}$ (here $W = 5$).

$D^{(1)} \triangleq w_1^{(1)} - w_2^{(1)}$ is equal to $D^{(2)} \triangleq w_1^{(2)} - w_2^{(2)}$, i.e. the waiting time difference of two users for both transmissions are equal.[3]

As shown in Fig. 3 the probability mass function of $D \triangleq w_1 - w_2$, where w_1 and w_2 are independent and uniformly distributed over $\{0, \ldots, W\}$, is

$$P(D = k) = \begin{cases} \frac{(W+1)-|k|}{(W+1)^2} & \text{if } |k| \leq W \\ 0 & \text{otherwise} \end{cases}$$

and the probability of having identical collision patterns in both transmissions is

$$\sum_{k=-W}^{W} (P(D = k))^2 = \frac{1}{3} \frac{2W^2 + 4W + 3}{(W + 1)^3}.$$

When the above happens, the ZigZag is unable to decode the packets since it cannot initiate the back-substitution.

For $N = 3$ users, the error happens if either two out of three collision patterns are identical, or two packets collide in the same way in all three collisions. As we are only looking for a lower bound, we consider only the latter case and we have

$$P\,[\text{failure}] > \textstyle\sum_{k=-W}^{W} (P(D = k))^3$$

where $P(D = k)^3$ is the probability that users 1 and 2 have a delay difference of k on all 3 collision rounds (which is the same probability if computed between users 1 and 3 or users 2 and 3).

Fig. 4 shows the bound on the error probability of the ZigZag decoder for $N = 2, 3$ users as a function of the maximum waiting time W. Again note that the above bound holds, regardless of the SNR.

2) Error Accumulation: The second shortcoming of ZigZag is error accumulation which can be described as follows. ZigZag decoding relies on continuously repeating two steps: i) finding free chunks and decoding them, and ii) removing the

[2]If the SNR's are significantly different, ZigZag can decode the packets using the capture effect [2].

[3]Note that a collision (with bit-offset D) can be detected at the receiver by adding a known preamble to each packet, as explained in [2].

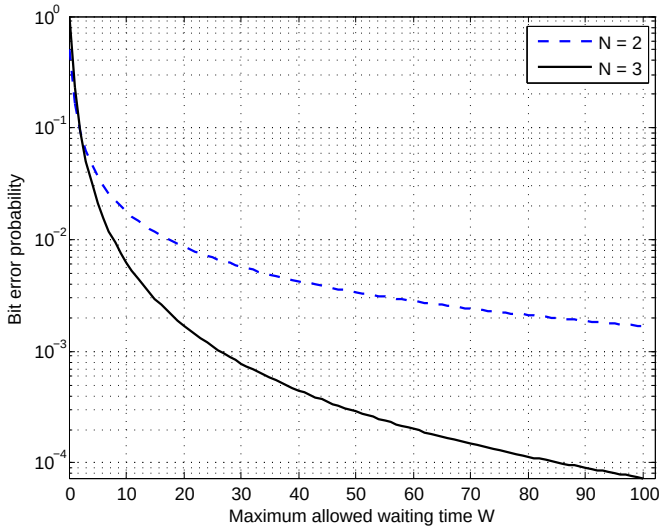

Fig. 4. The BER threshold of the ZigZag decoder for $N = 2$ and $N = 3$ user.

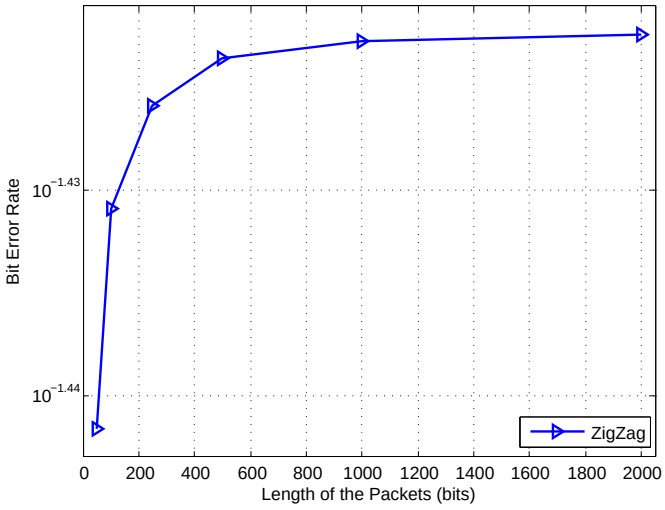

Fig. 5. ZigZag error accumulation for $N = 2$ users at $SNR = 5dB$. A very small degradation in performance is illustrated.

decoded bits from other collisions to produce new free chunks (back-substitution). At each decoding step, ZigZag makes a hard decision on the value of the bit which causes ZigZag to aggregate noise as it decodes through the packet. This aggregation becomes more severe as the length of the collided packets grow and worsens the performance of ZigZag. Fig. 5 shows the bit error probability of ZigZag for $N = 2$ users at $SNR = 5dB$, as the length of the packets B grows.

On the other hand, as it is shown in Fig. 6, the performance of our proposed SigSag algorithm improves as the length of the packets grow. This is due to the fact that the factor graph on which we run the message passing for SigSag becomes more tree-like as the length of the packets grow. This is proved in section IV-B.

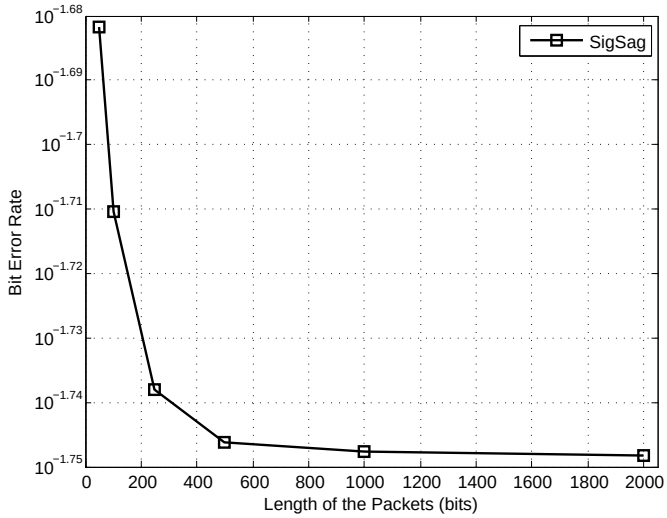

Fig. 6. Performance of SigSag as the length of the packets grow for $N = 2$ users at $SNR = 5dB$.

III. SigSag Algorithm

In this section we introduce SigSag, our soft iterative message-passing decoder, which attempts to find a good approximation decoding for the users' packets. There are two variations of SigSag with slightly different objectives: The first uses iterative sum-product to compute the marginals for each bit and hence minimize the bit-error rate. The second uses max-product to compute the jointly most likely configuration, attempting to minimize the block-error rate. When executed on factor graphs with cycles these algorithms can be suboptimal and in practice we observe that max-product has slightly better performance for the cases tested.

The algorithm is based on message passing over the *factor graph* which has *variable nodes* (or "bit nodes") on the left, *factor nodes* in the middle, and *observation nodes* (or received values) on the right, as shown in Fig. 7. We derive the messages for the first few steps of the max-product algorithm for the factor node f_{11}, as explained in [7], [8], on the factor graph shown in Fig. 7 which corresponds to the collision pattern shown in Fig. 2. Here, $\mu_{a \longrightarrow b}(x)$ represents the probability distribution function of x estimated at node a and sent as a message to node b. This distribution (sent over link $a \longrightarrow b$) is estimated from the previous messages that were sent on all incoming links to node a (other than the link connecting it to b). The vector $\overline{\mathbf{u}}_c$ is the noisy observation at the receiver on the c^{th} collision. Also let $\mathbf{u}_c$ be the received signal if the channel was noise-free, i.e. $\overline{\mathbf{u}}_c = \mathbf{u}_c + \nu$ where ν is the noise vector.

step 1:

$$\mu_{\overline{u}_{11} \longrightarrow f_{11}}(u_{11}) = \begin{cases} P(u_{11} = h_1^{(1)} | \overline{u}_{11}) \\ P(u_{11} = -h_1^{(1)} | \overline{u}_{11}) \end{cases}$$

step 2:

$$\mu_{f_{11} \longrightarrow x_1}(x_1) = \begin{cases} P(x_1 = 1) = P(u_{11} = h_1^{(1)} | \overline{u}_{11}) \\ P(x_1 = -1) = P(u_{11} = -h_1^{(1)} | \overline{u}_{11}) \end{cases}$$

step 3:

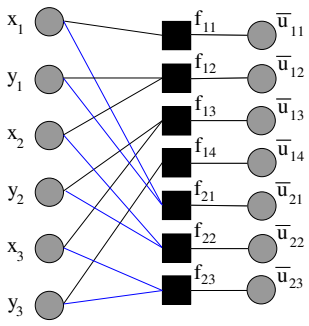

Fig. 7. Two consecutive collisions of two senders and the factor graph representation of the pattern.

$$\mu_{x_1 \longrightarrow f_{21}}(x_1) = \begin{cases} P(x_1 = 1) \\ P(x_1 = -1) \end{cases}$$

Step 4 is quite similar to step 1, only the degree of the nodes are different:

$$\mu_{\overline{u}_{21} \longrightarrow f_{21}}(u_{21}) = \begin{cases} P(u_{21} = h_1^{(2)} + h_2^{(2)} | \overline{u}_{21}) \\ P(u_{21} = h_1^{(2)} - h_2^{(2)} | \overline{u}_{21}) \\ P(u_{21} = -h_1^{(2)} + h_2^{(2)} | \overline{u}_{21}) \\ P(u_{21} = -h_1^{(2)} - h_2^{(2)} | \overline{u}_{21}) \end{cases}$$

In step 5, the algorithm applies the max operation to find the most probable configuration for each mass point $\{1, -1\}$:

$$\mu_{f_{21} \to y_1}(y_1) = \begin{cases} P(y_1 = 1) = \\ \frac{1}{z} \max \Big(P(u_{21} = h_1^{(2)} + h_2^{(2)} | \overline{u}_{21}) P(x_1 = 1), \\ \qquad P(u_{21} = -h_1^{(2)} + h_2^{(2)} | \overline{u}_{21}) P(x_1 = -1) \Big) \\ P(y_1 = -1) = \\ \frac{1}{z} \max \Big(P(u_{21} = h_1^{(2)} - h_2^{(2)} | \overline{u}_{21}) P(x_1 = 1), \\ \qquad P(u_{21} = -h_1^{(2)} - h_2^{(2)} | \overline{u}_{21}) P(x_1 = -1) \Big) \end{cases}$$

where z is a normalization factor. By changing the *max* operation to *sum* the algorithm becomes the *sum-product algorithm*. Further, instead of propagating the probabilities, one can transmit log-likelihood ratios

$$m_{f \longrightarrow x}(x) = \log \frac{P(x = 1)}{P(x = -1)}$$
$$m_{x \longrightarrow f}(x) = \log \frac{P(x = 1)}{P(x = -1)}$$

for factor node f and variable node x, where $P(x = 1)$, and $P(x = -1)$ are derived as mentioned earlier. For instance, for the sum-product algorithm, $m_{f_{21} \longrightarrow y_1}(y_1)$ from the above example can be computed to be

$$m_{f_{21} \longrightarrow y_1}(y_1) = \log \frac{P(y_1=1)}{P(y_1=-1)}$$
$$m_{f_{21} \longrightarrow y_1}(y_1) = \log \left(\frac{P(u_{21}=h_1^{(2)}+h_2^{(2)}|\overline{u}_{21}) e^{m_{x_1 \longrightarrow f_{21}}(x_1)}}{P(u_{21}=h_1^{(2)}-h_2^{(2)}|\overline{u}_{21}) e^{m_{x_1 \longrightarrow f_{21}}(x_1)}} \right.$$
$$\left. \frac{+P(u_{21}=-h_1^{(2)}+h_2^{(2)}|\overline{u}_{21})}{+P(u_{21}=-h_1^{(2)}-h_2^{(2)}|\overline{u}_{21})} \right)$$

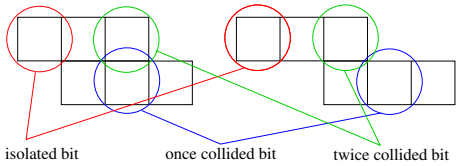

Fig. 8. Demonstration of isolated, once collided and twice collided bits for the two user scenario.

Similar to ZigZag, SigSag can be extended to support systems of higher alphabets such as 4-QAM and 16-QAM. Note that for higher alphabets the computation of messages is more complex, i.e. the computation complexity of SigSag grows exponentially in the size of the alphabet.

IV. Theoretical Analysis

In this section, we analyze the performance of SigSag theoretically, and prove that it is maximum likelihood (ML) for two users and close to ML for more than two users. Then we consider the high SNR regime and discuss the importance of jitter and fading.

A. Factor graph is cycle-free for two users

We show that for the case of two users the factor graph of the received signal is cycle-free if the bit-offsets of the two consecutive collisions are different. This result does not require permutations of bits with every retransmission. We start by observing that there are three different kinds of bits

1) Isolated or degree zero bits: These are received in free symbols, i.e. without colliding with any other bit in both collision rounds.
2) Once collided or degree one bits: These collide with a bit from the other packet in one collision and are received unharmed on the next collision.
3) Twice collided or degree two bits: These collide with different bits of the other packet in each collision.

The above node classes are shown in Fig. 8.

Theorem 1: In the case of two users transmitting without permuting their bits, the resulting factor graph is cycle-free if the bit-offsets of the two consecutive collisions are different.

Proof: See appendix for the proof. ■

We proved that if the bit-offsets of the two collisions are different the factor graph is cycle-free. Since message passing on cycle-free factor graphs is optimal [7], our soft receiver is performing ML detection for two users.

B. Tree-like factor graph for multiple users

We now consider any number of users $N \geq 2$, but assume that the users permute the bits of the packets randomly before each transmission. We define the directed neighborhood of depth ℓ of an edge $e = (f, v)$, from a factor node f to a variable node v, denoted by $\mathbf{N}_e^\ell$, to be the induced subgraph containing all edges and nodes on all paths $e_1, \ldots, e_\ell$ such that $e_1 \neq e$ and e_1 is adjacent to v, as shown in Fig. 16. A *tree-like* factor graph is one for which every edge has a *directed neighborhood* (up to a certain depth) that is a tree. Here, it is shown that the factor graph formed from the collision pattern

is probabilistically tree-like as the length of the packets B grows, and thus, the belief propagation result is close to ML.

Theorem 2: Consider the collision of N packets of length B, sent by N users that use independent psuedo-random permutations of their symbols (uniformly distributed over all permutations). The resulting left-regular bipartite factor graph G is locally tree-like with high probability. Specifically,

$$P(\mathbf{N}_e^{2\ell^*} \text{ is not a tree}) \leq \frac{s}{B} \tag{2}$$

where $e = (f, v)$ is a randomly chosen edge of the graph, $2\ell^*$ is a fixed depth, and s is a suitable constant that depends on ℓ^*, N, but not on B.

Proof: See appendix. ∎

C. Importance of Jitter

The purpose of this section is to point out the importance of jitter in our model. To begin, consider a high SNR system with flat fading $h_i^c = h_i$ for all $i, c \in \{1, \ldots, N\}$ where h_i is a positive constant. That is, the fading coefficients are constant for each user and do not change over time. An indoor wireless LAN is a good example of such a system where the fading coefficients are constant for each user over time.

We highlight the importance of jitter by showing that for the above system, when the maximum delay $W = 0$, the linear equations are not full rank even if the users permute their packets before each round of transmission. Observing the above while the users are not permuting their bits is trivial, since the fading coefficients are constant, and as a result the AP receives B equations each copied N times which obviously makes matrix A singular.

Theorem 3: For the above system, if the users transmit packets consecutively without any delay ($W = 0$), the matrix A is rank deficient regardless of the particular permutations used.

Proof: See appendix. ∎

We emphasize that rank deficiency of matrix A does not mean the received patterns are not decodable (this is illustrated with an example in Section IV-E). However, in the high SNR regime, full rank of the matrix A is quite useful, as it allows decoding to be done by solving the linear system $A\overline{\mathbf{x}} = \mathbf{u}$ via Gaussian elimination. Full rank is also a requirement for ZigZag decoding to work (in fact, ZigZag requires stronger assumptions because it only performs the back-substitution steps of Gaussian elimination). Zero noise examples illustrate these ideas in the next two subsections.

D. Random Delay of one Symbol Allows Full Rank

Consider the equations $A\overline{\mathbf{x}} = \mathbf{u}$ that result from N packet collisions from N users, where all packets are size B bits. Assume users do not permute their bits. Further, assume for simplicity that there is no noise, and that for each user i, the channel gain h_i is non-zero and is the same on each of the N transmissions of its packet. Observe that if we can decode the first bits of all packets, then we can decode the system. The reason is that as the first bits of all packets are decodable and known, then by removing the first bits from the equations we will again have the same pattern with packets of length $B-1$, since there are no permutations.

We first show a simple coordinated arrangement that uses a maximum delay of one symbol and guarantees all users can decode: At the i^{th} transmission, where $i \in \{1, 2, \ldots, N\}$, user i transmits its packet with zero delay, while the rest of the users transmit with one time-slot delay. Then clearly the first bits of all packets can be decoded (just look at the i^{th} transmission to find the first bit of packet i). Thus, all bits of all packets can be decoded.

Now consider an uncoordinated model where users randomly and uniformly choose a delay of 0 or 1, independently across users and across transmissions. Thus, the maximum delay is $W = 1$. We show that if the number of users N is large, then all packets can be decoded with high probability. To see this, form the $N \times N$ "jitter matrix" $\mathbf{J}$ which consists of the coefficients of the first equations resulting from the first symbol of each collision, i.e., $\mathbf{J} \cdot [x_{11}, \ldots, x_{N1}]^T = [u_{11}, u_{21}, \ldots, u_{N1}]^T$. Note that a row i of $\mathbf{J}$ might be identically zero if all packets choose to transmit with a delay of 1 time-slot in transmission i (so the first received symbol of collision i is null). However, if $\mathbf{J}$ is nonsingular, then the first bits of all packets can be determined by simple inversion of $\mathbf{J}$, and so all bits of all packets can be determined. As an example, the linear equations and their resulting matrix $\mathbf{J}$ for the pattern shown in Fig. 2 are

$$\mathbf{J} = \begin{bmatrix} h_1 & 0 \\ h_1 & h_2 \end{bmatrix}, \quad \begin{cases} h_1 x_1 = u_{11} \\ h_1 x_1 + h_2 y_1 = u_{21} \end{cases}$$

Note that each column j of $\mathbf{J}$ is composed of entries that are either 0 or h_j. We can multiply each column j by $1/h_j$ to form a new matrix $\overline{\mathbf{J}}$ that is a 0/1 matrix, and that has the same rank as $\mathbf{J}$. The matrix $\overline{\mathbf{J}}$ is an $N \times N$ matrix with elements independently taking the values $(0, 1)$ with probability $1/2$. A classic result by Komlós [4] establishes that such a matrix $\overline{\mathbf{J}}$ is nonsingular with probability going to 1 as $N \to \infty$.

E. Importance of Fading

We note that our SigSag algorithm exploits fading in a way that pure back-substitution or Gaussian elimination cannot. To illustrate the importance of fading, we consider a simple example with 3 users and zero noise. Assume the delay offset is 0 in all three collision rounds. Thus, the resulting equations are redundant and the matrix A is obviously rank deficient. ZigZag would thus fail in this scenario. However, our algorithm would perfectly decode each user with probability 1. For intuition on this, we simply show that the bits are perfectly decodable with probability 1 just by looking at the first collision: Suppose the channel coefficients on the first round are $h = [h_1, h_2, h_3]$. The first bit transmitted by each user is either $+1$ or -1. Thus, there are only 8 possible

resulting collision values for the first received symbol:

$$\begin{aligned} h_1 + h_2 + h_3 \\ h_1 + h_2 - h_3 \\ h_1 - h_2 + h_3 \\ h_1 - h_2 - h_3 \\ -h_1 + h_2 + h_3 \\ -h_1 + h_2 - h_3 \\ -h_1 - h_2 + h_3 \\ -h_1 - h_2 - h_3 \end{aligned}$$

If all 8 possible values are distinct, then we can completely decode the first bit of all 3 users. Clearly this holds not just for the first bit, but for all bits, and so in the noiseless case with different fading coefficients, all three packets can be completely decoded after only one transmission round. It is clear that if the distribution on the joint fading coefficients have a continuous joint density, then the probability that all 8 collision values are distinct is equal to 1. For example, the probability $P\left[h_1 + h_2 + h_3 = h_1 + h_2 - h_3\right]$ is the probability that the 3-dimensional vector (h_1, h_2, h_3) lies on the 2-dimensional plane $h_3 = 0$, and hence this probability is equal to 0. While this example is for the noiseless case, it shows that fading can be exploited to achieve very high decoding probability in the high SNR regime.

V. Joint Channel-Collision Decoding

In this section, we change our setup and move beyond the 802.11 standard, i.e. we consider the case where the packets $\mathbf{x}_i$ for $i \in \{1, 2, \dots N\}$ are coded by LDPC codes. Different users may use different codes. Thus the variable nodes (transmitted bits) are connected to both factor nodes of the collisions and check nodes of the code. Combining Tanner graph of the code and the collision factor graph, we form a bigger joint factor graph as shown in Fig. 9 which we refer to as the *joint channel-collision graph*. We call the factor nodes representing the collision between bits *collision factor nodes* and the factor nodes representing the relation between bits of a coded packet *parity check nodes*.

We introduce the joint channel-collision decoding which is done by running belief propagation over the joint channel-collision graph. For the joint system, the message from collision factor nodes to variable nodes is computed the same as described before, and the message from parity check nodes to variable nodes is computed as described in coding literatures [7], [23]. However, the computation of the message from the variable nodes to factor nodes changes. The message from a variable node to a factor node (either a check node or a collision node) is computed as follows

$$m_{x \longrightarrow g}(x) = \sum_{c \in \mathcal{N}(x) \setminus \{g\}} m_{c \longrightarrow x}(x) + \sum_{f \in \mathcal{N}(x) \setminus \{g\}} m_{f \longrightarrow x}(x)$$

where m is the incoming log-likelihood ratio messages from neighboring factor nodes, c represents the check nodes, and f represents the collision nodes. We can decode and extract the data for all collided packets simultaneously by running belief propagation over the joint factor graph. Note that belief propagation on the joint graph is specific to this problem in the sense that beliefs on collision factor nodes are computed over the real field while beliefs over the parity check nodes are computed over the binary field.

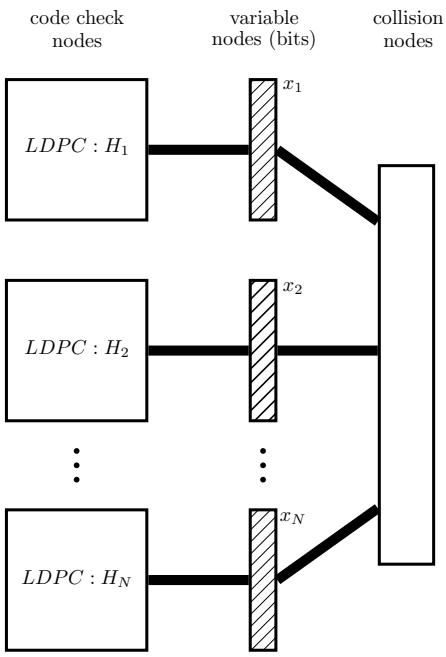

Fig. 9. The block diagram of the joint channel-collision decoding where variable nodes (bits of packets $\mathbf{x}_1, \dots, \mathbf{x}_N$) are connected to both check nodes and collision nodes.

One might wonder how well the joint channel-collision decoding performs. Recall that on section IV-B we showed that the collision factor graph is asymptotically tree-like in B. This is also true for the other component of the joint channel-collision graph, i.e. the Tanner graph of the code which is shown to be asymptotically tree-like in [8]. However, to find a lower bound on locally tree-like probability of the joint channel-collision graph, we need to consider the *joint cycles* formed by merging the two graphs as explained in the next subsection.

A. Joint Cycles

As explained, by combining the collision factor graph and N Tanner graphs as shown in Fig. 9, we get the *joint channel-collision graph* which we refer to (in short) as the joint graph. Unfortunately, even if the Tanner and collision graphs are tree-like, the joint factor graph may not be. Specifically, the joint graph may contain short cycles with the length as short as 6 in the worst case. We refer to these unwanted cycles as joint cycles since both check nodes and collision nodes contribute in forming them. Let us clarify this with an example. Consider the case where two users send coded packets $\mathbf{x}, \mathbf{y}$ twice. Further, assume that the bit x_1 collides with y_1 in the first collision, and with y_2 in the second collision. Furthermore, let the bits y_1 and y_2 in the packet $\mathbf{y}$ be connected to the same parity check in the code. As a result, the joint graph has a cycle of length 6 as shown in Fig. 10.

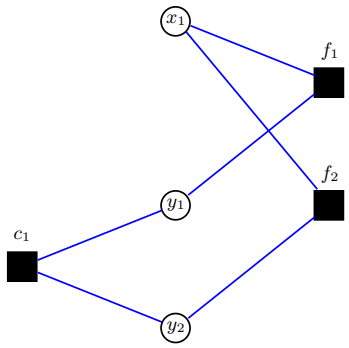

Fig. 10. The joint cycle formed by collision of x_1 with y_1 and y_2, and adjacency of y_1 and y_2 to the same check c_1 node in the Tanner graph of the code.

Similar to Theorem 2, we can show that the joint graph is asymptotically locally tree-like by modifying the value s in (2). Note that a collision node is adjacent to bits from different packets and at most one from each, while a check node is only connected to the bits of the same packet. One easy way to show that the joint graph is locally tree-like with high probability is to map it to an already proven asymptotically locally tree-like graph which has higher probability of forming cycles than the joint graph. We use this to prove that the joint graph is locally tree-like in the following theorem.

Theorem 4: Consider the case where N users transmit their packets N times and the packets are coded with a (n_v, n_c)-regular LDPC code. Call the joint graph formed by this setup G. Then, as N grows, G is locally a tree with high probability.

Proof: We construct a graph with lower probability of being tree-like than G and for the new graph, we prove that the neighborhood of each edge is a tree. Consider a graph $\tilde{G}$ where $d_v = N+n_cn_v$ and $d_f = N+1$, where each factor node besides its root in a higher level (recall Fig. 16) is connected to N other bits one from each packet. Note that in graph G each variable node (bit) is connected to $n_v(n_c-1)$ bits from the packet it belongs to through check nodes and to N bits from each of other packets through the collision nodes. In graph $\tilde{G}$, on the other hand, each bit belonging to packet $\mathbf{x}$ is connected to $d_v = N+n_vn_c$ factor nodes while through these d_v factor nodes, it gets connected to $N+n_cn_v$ bits from its own packet, namely $\mathbf{x}$, and $N+n_cn_v$ bits from each of other packets. As a result, it is apparent that $P(\tilde{G}$ is tree-like$) \leq P(G$ is tree-like$)$. Further, It is easy to check that the same derivation as in Theorem 2 with some modifications can be used to show that the graph $\tilde{G}$ is tree-like. For example, the values V_i and F_i in the derivation of the proof of 2, change to $V_i = (N+n_cn_v)^iN^i$ and $F_i = (N+n_cn_v)^{i+1}N^i$. ■

Unfortunately, for short LDPC codes, joint cycles may worsen the performance of message passing. We investigate different scheduling for message passing in section VI. That is, we run belief propagation on the collision graph for n_{col} iterations, and insert the obtained belief as an input to another message passing scheme over the Tanner graph for $n_{\mathcal{C}}$ iterations. We repeat this process for n_{tot} times and represent it by the tuple $(n_{col}, n_{\mathcal{C}}, n_{tot})$.

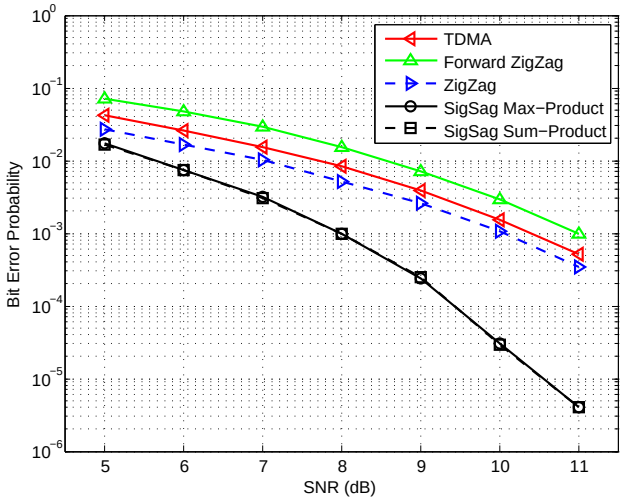

Fig. 11. Bit error rate comparison of TDMA, forward ZigZag, forward-backward ZigZag, SigSag max-product, and SigSag sum-product over different SNR's for $N = 2$ users transmitting packets of length $B = 100$.

VI. Experimental Results

In this section we present our preliminary experimental evaluation of SigSag for $N = 2, 3$ users, and compare against a ZigZag decoder and a coordinated system that uses an interference-free round-robin TDMA schedule. Surprisingly, we find that the performance of our system greatly surpasses that of the coordinated TDMA. The reason is that each bit influences multiple observations, *acting as a code that our near-optimal decoding algorithm can decode with high probability*. Since any message passing algorithm can be used for SigSag to compute likelihoods of bits, we consider both max-product and sum-product. We call the former SigSag max-product, and the latter SigSag sum-product.

For our simulation we consider an additive white Gaussian noise channel with fading modeled as a uniform random variable chosen between $[0.8, 1.2]$. The block fading is assumed, i.e. fading is identical for all the bits of a packet in each transmission; but it can vary across users and in different transmissions. In our simulations, Users transmit packets of length $B = 100$, with the maximum waiting time of $W = 10$. Here, we consider the case where packets always collide with each other, and ZigZag never fails to decode. Specifically, cases where the back-substitution for ZigZag cannot be initiated are simply thrown out, so that our results make ZigZag look even better (recall that SigSag does not require back-substitution to be initiated). Further, for simulating the TDMA, we transmit packets from $\{0,1\}^B$ over the channel and the receiver decodes by making hard decisions over the received real values.

Our first experiment compares the performance of SigSag, without any permutation, versus forward ZigZag, ZigZag (forward-backward), and TDMA. For SigSag, message passing is run for 20 iterations (after which it converged). As shown in Fig. 11, for $N = 2$ users, SigSag max-product and SigSag sum-product perform quite similar to each other while both surpass ZigZag and TDMA with a great margin.

For the second experiment, we consider the previous scenario, only this time with $N = 3$ users. For simulating ZigZag,

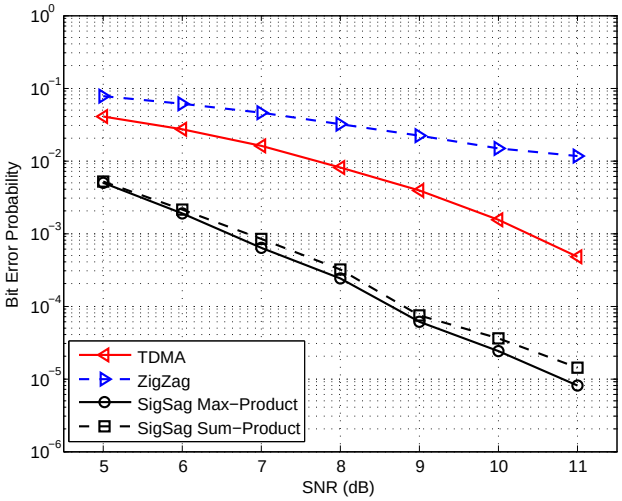

Fig. 12. Bit error rate comparison of TDMA, ZigZag, SigSag max-product, SigSag sum-product over different SNR's for $N = 3$ users transmitting packets of length $B = 100$.

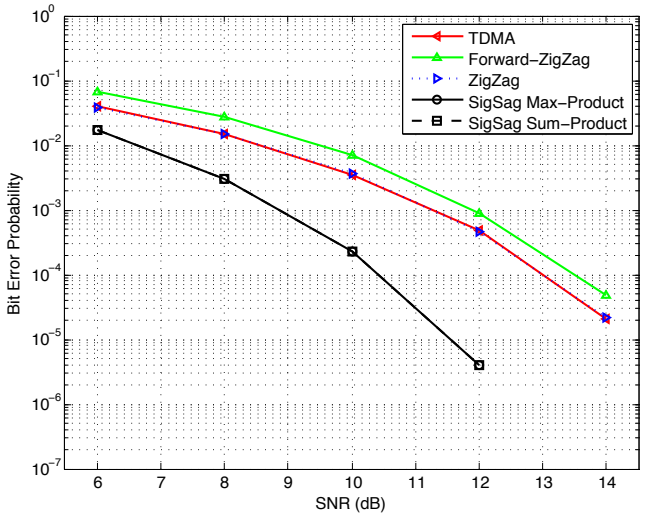

Fig. 13. Bit error rate comparison of TDMA, ZigZag, SigSag max-product, SigSag sum-product over different SNR's for $N = 2$ users transmitting 4-QAM symbol streams of length $B = 200$.

we used its extension for more than two users from [2]. As shown in Fig. 12, the performance of ZigZag degrades as number of users increase and it becomes worse than TDMA. Again, as before, SigSag max-product and SigSag sum-product surpass other methods while this time SigSag max-product performs even better than SigSag sum-product.

Notice that our proposed algorithms are computationally efficient, requiring a linear number of messages per iteration and typically a number of iterations that scales logarithmically in the packet sizes and number of users [8], [23]. There is slightly more complexity compared to ZigZag since soft information needs to be communicated with each message, but the performance benefits would probably justify these additional requirements for most applications.

A. Higher Alphabets

Here, we consider the case where the transmitted symbols are modulated over the system of higher alphabets. Specifically, we assume the packets are modulated with 4-QAM and 16-QAM. For the experiment, we consider the same setup as the previous sections. Further, we assume two users transmit streams of 4-QAM and 16-QAM symbols of length 200. As shown in Fig. 13 and 14 our SigSag algorithms perform better than TDMA and ZigZag for any alphabet size. Further, we observe that as the size of the alphabet grows the performance of all algorithms deteriorates, however, the gap between SigSag and other decoders increases. As before, SigSag max-product and SigSag sum-product have quite similar performance. Furthermore, we observe that ZigZag which has a better performance than TDMA over BPSK, performs similar to TDMA over 4-QAM, and worse than TDMA over 16-QAM.

B. Joint Channel-Collision Decoding

Here, we examine the performance of joint channel-collision decoding. We consider two cases where the packets are coded by the Tanner code of length 155. We compare our algorithm

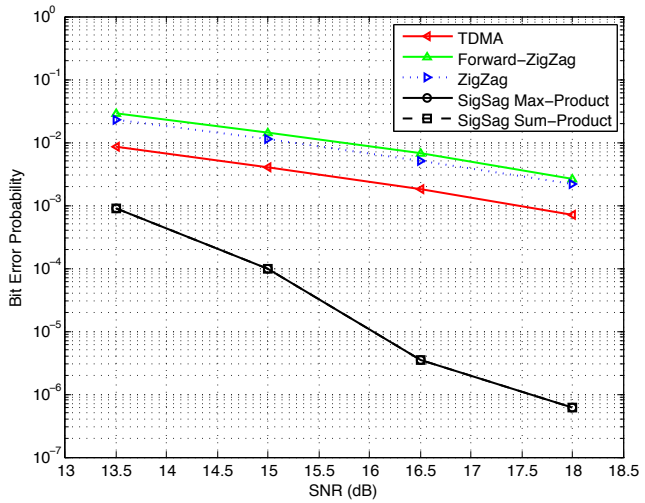

Fig. 14. Bit error rate comparison of TDMA, ZigZag, SigSag max-product, SigSag sum-product over different SNR's for $N = 2$ users transmitting 16-QAM symbol streams of length $B = 200$.

with other decoders where the transmitted packets are first received by either ZigZag or a collision-free system and then are decoded by sum-product decoding. For our algorithm, we only consider the sum-product algorithm. Note that unlike before, we draw the plots with respect to the frame error rate (FER) of the algorithms.

We compare the performance of different scheduling schemes in the joint channel-collision decoding. As shown in Fig. 15, the joint channel-collision decoder performs better than other receivers, namely ZigZag and collision-free receiver, which was not surprising as the experiments in the previous section show that SigSag performs better than ZigZag and TDMA as a receiver.

Interestingly, we observe that among different scheduling schemes of the joint channel-collision decoder, the $(20, 40, 1)$ scheme performs better. That is, as the communication between the belief propagation over the Tanner graph and collision graph increases, the performance of the joint channel-collision decoder worsens. We conjecture that this is due to

effect of the joint cycles, i.e., this is due to the joint factor graph structure that makes the optimized 155 Tanner code perform worse.

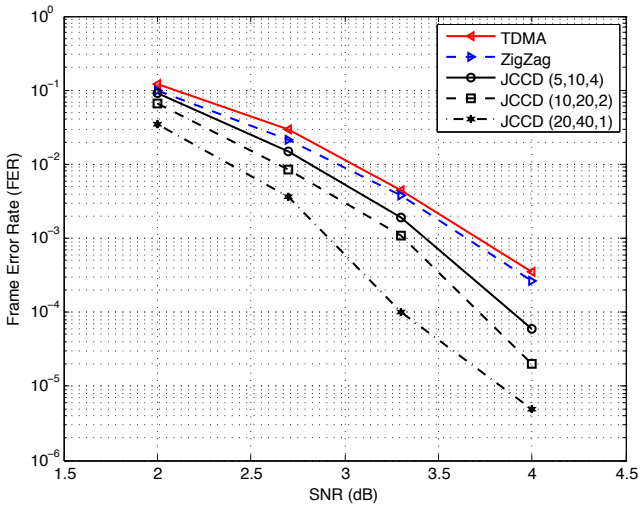

Fig. 15. Frame error rate comparison of joint channel-collision decoding with scheduling (5,10,4), (10,20,2), and (20,40,1) with the ZigZag and collision-free system followed by the sum-product decoder over the packets coded by the 155-Tanner code.

VII. Conclusion

In this paper we introduced a novel decoding algorithm that is compatible with the 802.11 framework and allows the decoding of interfering users. As it is well known in the information and coding theory community, repetition codes are highly suboptimal. In this paper we show that *repetition with the addition of small random jitter and bit permutation forms good codes* that can be efficiently decoded and can outperform TDMA. We further introduce the joint channel-collision decoder which acts as a receiver for LDPC coded packets. There are several issues that need to be further investigated. On the practical side, how to enable bit permutations and access to soft information with current infrastructure would be important for exploiting interference at the physical layer [12], [21]. Further, our fading assumptions would need to be tested in real deployments and the performance of our algorithm investigated under different fading and retransmission models. On the theoretical side, one future direction is to use density evolution [8] to predict the exact performance of the proposed decoders and further optimize the design of the jitter distribution. More generally, investigating the factor graphs created by uncoordinated users interfering and exploiting the message-passing inference machinery for such problems is promising direction for wireless communications.

VIII. Appendix: Proofs of the Theorems

Here we present the proofs of Theorems 1, 2, 3.

Proof of Theorem 1: All possible cycles consist only of degree two variable nodes and degree two factor nodes, and have even length. Further, observe that there can be no cycle of length two since a bit cannot collide with itself. Here we use the same notation introduced in II-A1, where $D^{(1)}$ and $D^{(2)}$ are the time difference between user transmissions

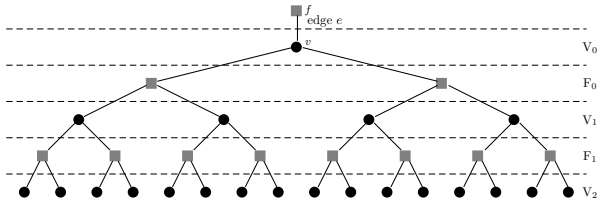

Fig. 16. Section of the the directed neighborhood of depth 3 of an edge.

in first and second collisions. Arrange the variable nodes according to their order of appearance in the packets. In other words, arrange them in two columns $x_1, x_2, \ldots, x_B$ and $y_1, y_2, \ldots, y_B$. Beginning from an arbitrary variable node, we try to build a cycle by connecting the variable nodes through arbitrary factor nodes.

Because all factor nodes of a cycle have degree 2 and connect to one "x" node and one "y" node, we can view the cycle by omitting the factor nodes and considering only the variable nodes. Assume that we intend to make a cycle which begins and ends at x_i, where x_i is a degree two node (twice collided). First we connect x_i to y_j, and then we connect y_j to x_k. Notice that these two edges have already determined the pattern of both collisions, since $D^{(1)} = i-j$ and $D^{(2)} = k-j$. For example, we can deduce that the node x_i collides with y_j on the first collision and with $y_{i-D^{(2)}} = y_{i-k+j}$ on the second one. Note that $k \neq i$, since we assumed that collision patterns are different, i.e., $D^{(1)} \neq D^{(2)}$. This also shows that there is no cycle of length four in our factor graph. As a result, we consider two cases, either $k > i$ or $k < i$. Consider the former $k > i$. As mentioned before, we want to continue connecting the bits from y_j and get back to x_i. The path is $\{x_i, y_j, x_k, y_{k-D^{(1)}}, x_{k-D^{(1)}+D^{(2)}}, \ldots\}$. We see that on every return to an "x" node, the index increases by exactly $k-i$ (note that $(k-D^{(1)}+D^{(2)})-k = k-i$). Thus the chain must increase away from x_i until it reaches a degree-1 node that terminates and shows this is not a cycle. The same argument holds for the case $k < i$, only this time the chain moves toward the start of the packets and terminates there. ■

Proof of Theorem 2: This proof uses a technique similar to the local tree-like proofs for random sparse graphs developed for Low Density Parity Check (LDPC) codes [8]. The ensemble of random graphs we are dealing with, formed by the random jitter and permutations, are different from the LDPC ensembles but still the same approach works.

Note that all the variable nodes v have degree $d_v = N$ and the degree of the factor nodes d_f is less than or equal to N ($d_f \leq N$). In other words, each bit appears in exactly N equations and each equation contains at most N variables. Since increasing the degree of the factor nodes only increases the probability of existence of cycles, without loss of generality we assume $d_f = N$ as shown in Fig. 16. For a given integer ℓ, we want to show that if $\mathbf{N}_e^{2\ell}$, then $\mathbf{N}_e^{2\ell+1}$ and $\mathbf{N}_e^{2\ell+2}$ are trees with a certain (high) probability. To this end, note that at each level i there are $V_i = (d_v-1)^i(d_f-1)^i = (N-1)^{2i}$ variable nodes and $F_i = (d_v-1)^{i+1}(d_f-1)^i = (N-1)^{2i+1}$ factor nodes. As a result, assuming that $\mathbf{N}_e^{2\ell}$ is a tree, there are

$$\begin{aligned}\overline{V}_\ell &= \sum_{i=0}^{\ell} V_i \\ &= \sum_{i=0}^{\ell} (d_v-1)^i (d_f-1)^i \\ &= \sum_{i=0}^{\ell} (N-1)^{2i} \\ &= \frac{(N-1)^{2\ell+2}-1}{(N-1)^2-1} = \frac{(N-1)^{2\ell+2}-1}{N(N-2)}\end{aligned}$$

variable nodes and

$$\begin{aligned}\overline{F}_\ell &= 1+\sum_{i=0}^{\ell-1} F_i \\ &= 1+(d_v-1)\sum_{i=0}^{\ell-1}(d_v-1)^i(d_f-1)^i \\ &= 1+(N-1)\sum_{i=0}^{\ell-1}(N-1)^{2i} \\ &= 1+(N-1)\frac{(N-1)^{2\ell}-1}{(N-1)^2-1}\end{aligned}$$

factor nodes in it. Recall that the collision matrix A is $M \times NB$, where $NB \le M \le N(B+W)$. We want to compute the probability that the $\mathbf{N}_e^{2\ell+1}$ is a tree. For that, we consider a variable node in the $2\ell^{th}$ depth. The probability that its $(k+1)^{th}$ edge does not create a loop, assuming that its previous k edges have not, is

$$P \ge \frac{(M-\overline{F}_\ell-k)d_f}{Md_f-\overline{F}_\ell-k} \quad \ge 1-\frac{\overline{F}_{\ell^*}}{M},$$

for $\ell \le \ell^*$. This is due to the fact that there are in total M factor nodes, that $\overline{F_\ell}+k$ of them are already connected through *one edge* to variable nodes in the directed neighborhood. As a result, $Md_c-\overline{F_\ell}-k$ edges are left in the graph, from which the $(k+1)^{th}$ edge of the variable node can choose one to connect to a factor node. However, for this edge not to form a loop, it should choose from one of the $(M-\overline{F_\ell}-k)d_c$ edges belonging to the factor nodes that has not appeared in the directed neighborhood. Thus the probability that $\mathbf{N}_e^{2\ell+1}$ is a tree, given that $\mathbf{N}_e^{2\ell}$ is a tree, is lower bounded by $(1-\frac{\overline{F}_{\ell^*}}{M})^{F_{\ell+1}}$.

We now bound the probability that $\mathbf{N}_e^{2\ell+2}$ is a tree, given that $\mathbf{N}_e^{2\ell+1}$ is a tree. Notice that each factor node is allowed to have at most one neighbor from each packet. So consider the variable node at level zero (V_0) to belong to the i^{th} packet. We use R_ℓ for number of bits from the i^{th} packet at level ℓ, and (since number of bits from each of other packets are equal at each level) K_ℓ for number of bits from each one of the other packets at level ℓ. Apparently, at level $\ell=0$, we have $R_0=1$, and $K_0=0$. It is easy to check that $R_\ell=(N-1)K_{\ell-1}$ and $K_\ell=\frac{V_\ell-R_\ell}{N-1}$ for all ℓ. Unlike the previous part, the number of bits at each level are different for different users. However, we can bound the total number of bits from each user at level ℓ by $c\frac{\overline{V}_\ell}{N}$, i.e $c\frac{\overline{V}_\ell}{N} \ge \sum_{\ell=0}^{\ell^*} R_\ell$ and $c\frac{\overline{V}_\ell}{N} \ge \sum_{\ell=0}^{\ell^*} K_\ell$ where c is a positive scalar (for example $c=2$). As a result, the same as above, we can show that the outgoing edges of the factor nodes at depth $2\ell+1$ will not make a loop with probability

$$Q \ge \frac{(B-c\frac{\overline{V}_\ell}{N})d_v}{Bd_v-\frac{\overline{V}_\ell}{N}} \quad \ge 1-c\frac{\overline{V}_{\ell^*}}{NB}.$$

So, the probability that $\mathbf{N}_e^{2\ell+2}$ is a tree, given that $\mathbf{N}_e^{2\ell+1}$ is a tree, is lower-bounded by $(1-c\frac{\overline{V}_{\ell^*}}{NB})^{V_{\ell+1}}$. Thus,

$$\begin{aligned}P(\mathbf{N}_e^{2\ell^*} \text{ is a tree}) &\ge (1-c\frac{\overline{V}_{\ell^*}}{NB})^{\overline{V}_{\ell^*}}(1-\frac{\overline{F}_{\ell^*}}{M})^{\overline{F}_{\ell^*}} \\ &\ge (1-c\frac{\overline{V}_{\ell^*}}{NB})^{\overline{V}_{\ell^*}}(1-\frac{\overline{F}_{\ell^*}}{N(B+W)})^{\overline{F}_{\ell^*}}.\end{aligned}$$

For large enough n,

$$\begin{aligned}P(\mathbf{N}_e^{2\ell^*} \text{ is not a tree}) &\le \frac{c^2\frac{\overline{V}_{\ell^*}^2}{N}+\frac{\overline{F}_{\ell^*}^2}{N}}{B} \\ &= \frac{c^2\overline{V}_{\ell^*}^2+\overline{F}_{\ell^*}^2}{NB}.\end{aligned}$$

■

Proof of Theorem 3: Define $(\mathbf{bit}_{n,b})$ as the matrix of bits for each user $n \in \{1,\dots,N\}$ and each bit $b \in \{1,\dots,B\}$. Define $\mathbf{x}_{nb}$ as the matrix of "*faded-bits*" where each row n is multiplied by the fading coefficient h_n (so that $\mathbf{x}_{nb}=h_n\mathbf{bit}_{nb}$) for all n and b). If $W=0$, each faded-bit received on each collision round is a sum of contributions from each of the N users. Assume each user permutes its bits, and $\left(\tilde{\mathbf{x}}_{nb}^{(c)}\right)_{b=1}^{B}$ represents the vector of permuted bits for user $n \in \{1,\dots,N\}$ on collision round $c \in \{1,\dots,N\}$. It is assumed that the receiver knows the permutation orders used on each transmission. Let $u_{c,b}$ represent the b^{th} bit received on the c^{th} collision round, for $b \in \{1,\dots,B\}$ and $c \in \{1,\dots,N\}$:

$$u_{c,b}=\textstyle\sum_{n=1}^{N}\tilde{x}_{nb}^{(c)}$$

These values $u_{c,b}$ represent NB linear equations, involving NB unknowns (x_{nb}). Thus, matrix A will be rank deficient if at least one of the equations is redundant.

Let S be the sum of all elements in the $N \times B$ matrix $(\mathbf{x}_{nb})$. On each collision round $c \in \{1,\dots,N\}$, each row of matrix $(\tilde{\mathbf{x}}_{nb}^{(c)})$ is just a permutation of the corresponding row of matrix $(\mathbf{x}_{nb})$, and so the sum of all its elements is also equal to S, i.e.

$$\textstyle\sum_{b=1}^{N}\sum_{n=1}^{N}\tilde{x}_{nb}^{(c)}=S, \quad \text{for all } c \in \{1,\dots,N\} \quad .$$

Let us now sum the B received bits $u_{1,b}$ from the first collision round

$$S=\textstyle\sum_{b=1}^{B}u_{1,b}=\sum_{b=1}^{B}\sum_{n=1}^{N}\tilde{x}_{nb}^{(1)}. \tag{3}$$

Now sum the first $B-1$ received values $u_{2,b}$ on the second round of collision, i.e.

$$\begin{aligned}\textstyle\sum_{b=1}^{B-1}u_{2,b} &= \textstyle\sum_{b=1}^{B-1}\sum_{n=1}^{N}\tilde{x}_{nb}^{(2)} \\ &= \textstyle\sum_{b=1}^{B}\sum_{n=1}^{N}\tilde{x}_{nb}^{(2)}-\sum_{n=1}^{N}\tilde{x}_{nB}^{(2)} \\ &= S-u_{2,B}\end{aligned}$$

Subtracting this from equation (3) yields:

$$\textstyle\sum_{b=1}^{B}u_{1,b}-\sum_{b=1}^{B-1}u_{2,b}=u_{2,B} \quad .$$

Therefore, we can infer the value of the $(2B)^{th}$ received value $\mathbf{u}_{2B} = u_{2,B}$ only from observing the first $2B-1$ bits, and hence the equation associated with this bit is redundant. ■

ACKNOWLEDGMENT

The authors would like to thank Prof. Giuseppe Caire for several stimulating discussions.

REFERENCES

[1] A. Saber Tehrani, A. G. Dimakis, and M. J. Neely, *SigSag: Iterative Detection through Soft Message-Passing*, IEEE INFOCOM, Shanghai, China, April 2011.

[2] S. Gollakota and D. Katabi, *ZigZag Decoding: Combating Hidden Terminals in Wireless Networks*, ACM SIGCOMM, vol. 38, no. 4, pp. 159–170, Oct. 2008.

[3] J. Boutros and G. Caire, *Iterative multiuser joint decoding: unified framework and asymptotic analysis*, IEEE Trans. on Info. Theory, July 2002, pp. 1772–1793.

[4] J. Komlós, *On the determinant of* $(0,1)$ *matrices*, Studia Scientiarum Mathematicarum Hungarica, pp. 7–22, 1967.

[5] S. Verdu, *Multiuser Detection*, Cambridge, U.K.: Cambridge Univ.Press, 1998.

[6] M. Luby, *LT Codes*, The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.

[7] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, *Factor graphs and the sum-product algorithm*, IEEE Trans. Info. Theory, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[8] T. Richardson and R. Urbanke, *The capacity of low-density parity check codes under message-passing decoding.* IEEE Trans. Info. Theory, vol. 47, pp. 599–618, Feb. 2001.

[9] D. Reynolds, X. Wang, and H. V. Poor, *Blind adaptive space-time multiuser detection with multiple transmitter and receiver antennas*, IEEE Trans. on Signal Processing, vol. 50, no. 6, pp. 1261–1276, June 2002.

[10] H. El Gamal and E. Geraniotis, *Iterative multiuser detection for coded CDMA signals in AWGN and fading channels*, IEEE Journal on Selected Areas in Communications, vol. 18, pp. 30–41, Jan. 2000.

[11] H. V. Poor, *Iterative multiuser detection*, IEEE Signal Processing Magazine, Vol. 21, pp. 81–88, Jan. 2004.

[12] L. Erran, J. Liu, Kun Tan, H. Viswanathan, and R. Yang, *Retransmission Repeat: Simple Retransmission Permutation Can Resolve Overlapping Channel Collisions*, Eighth ACM Workshop on HotNets, Oct. 2009.

[13] B. Nazer and M. Gastpar, *Compute-and-forward: Harnessing interference with structured codes*, *submitted to IEEE Trans. Info. Theory*. Available under: http://arxiv.org/abs/0908.2119

[14] X. Zhang, K. G. Shin, *Chorus: Collision Resolution for Efficient Wireless Broadcast*, Proceedings of IEEE INFOCOM, 2010.

[15] J. G. Andrews, *Interference Cancellation for Cellular Systems: A Contemporary Overview*, IEEE Wireless Communications Magazine, Apr. 2005.

[16] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein, *Growth Codes: Maximizing Sensor Network Data Persistence*, Proceedings of ACM SIGCOMM, Pisa, Italy, Sep. 2006.

[17] F. Soldo, A. Markopoulou, and A. Lopez-Toledo, *A Simple Optimization Model for Wireless Opportunistic Routing with Intra-session Network Coding*, in Proc. of NetCod 2010, Toronto, Canada, June 2010.

[18] A. Khreishah, C. C. Wang, and N. B. Shroff, Cross-layer Optimization *for Wireless Multihop Networks with Pairwise Intersession Network Coding*, IEEE Journal on Selected Areas in Communications, vol. 27, no. 5, pp. 606–621, June 2009.

[19] A. Eryilmaz, A. Ozdaglar, and M. Medard, *On the Delay and Throughput Gains of Coding in Unreliable Networks*, IEEE Trans. on Info. Theory, vol. 54, no. 12, pp. 5511–5524, 2008.

[20] H. Pishro-Nik, N. Rahnavard, and F. Fekri, *Non-uniform error correction using low-density parity check codes*, IEEE Trans. on Info. Theory, vol. 51, no. 7, pp. 2702–2714, July 2005.

[21] S. Katti, S. Gollakota, and D. Katabi, *Embracing Wireless Interference: Analog Network Coding*, ACM SIGCOMM, 2007.

[22] S. Y. El Rouayheb, A. Sprintson, and C. Georghiades, *On the Relation Between the Index Coding and the Network Coding Problems*, in proceedings of ISIT 2008, Toronto, Canada.

[23] T. Richardson and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.

[24] J. Paek and M. J. Neely, *Mathematical Analysis of Throughput Bounds in Random Access with ZigZag Decoding*, Proc. of 7th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), June 2009.