Forty-Seventh Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
September 30 - October 2, 2009

Nonlinear Network Coding is Necessary to Combat General Byzantine Attacks

Oliver Kosut, Lang Tong, and David Tse

***Abstract*— We consider the problem of achieving capacity through network coding when some of the nodes act covertly as Byzantine adversaries. For several case-study networks, we investigate rates of reliable communication through network coding and upper bounds on capacity. We show that linear codes are inadequate in general, and a slight augmentation of the class of linear codes can increase throughput. Furthermore, we show that even this nonlinear augmentation may not be enough to achieve capacity. We introduce a new class of codes known as bounded-linear that make use of distributions defined over bounded sets of integers subject to linear constraints using real arithmetic.**

I. Introduction

Network coding allows routers in a network to execute possibly complex codes in addition to mere forwarding; it has been shown that allowing them to do so can increase throughput [1]. However, taking advantage of this use of coding at internal nodes means that the sources and destinations must rely on other nodes—nodes they may not have complete control over—to reliably perform certain functions. If these internal nodes do not perform the function correctly, or, worse, maliciously attempt to subvert the goals of the users, launching a so-called Byzantine attack [2], [3], standard network coding techniques have no method to maintain performance.

Suppose an omniscient adversary controls an unknown portion of the network, and may arbitrarily corrupt the values sent on certain links. We wish to determine how the size of the adversarial part of the network influences the maximum achievable throughput. If the adversary may control any t unit-capacity edges in the network, then it has been shown that, for the multicast problem (one source and many destinations), the capacity reduces by $2t$ compared to the non-Byzantine problem [4], [5], [6]. To achieve this rate, only linear network coding is needed. Furthermore, if there is just one source and one destination, only routing is needed at internal nodes.

The above model assumes that any set of t edges may be adversarial, which may be overly pessimistic depending on the situation. If the adversary cuts a certain number of transmission lines in a network, this would be a reasonable model. If, on the other hand, the adversary seizes a single router, it will control the values on all links connected to that router, which may vary in number depending on which router is attacked. For example, consider the network shown in Figure 1, which we call the Cockroach Network, and suppose one node (other than the source or destination) is a traitor; i.e., it is controlled by the adversary. Since any node has at most 2 output edges, it would be tempting to use a code that can handle any 2 malicious edges. However, this assumes that, for instance, the edges $(4, D)$ and $(5, D)$ may be simultaneously controlled by the adversary, which is not the case if it can only control one node. In fact, since the min-cut of this network is 4, the best achievable rate assuming 2 malicious edges is 0. As we will show in Section IV, the capacity of this network is in fact 2, so the aforementioned edge result can be overly pessimistic for this more specialized problem.

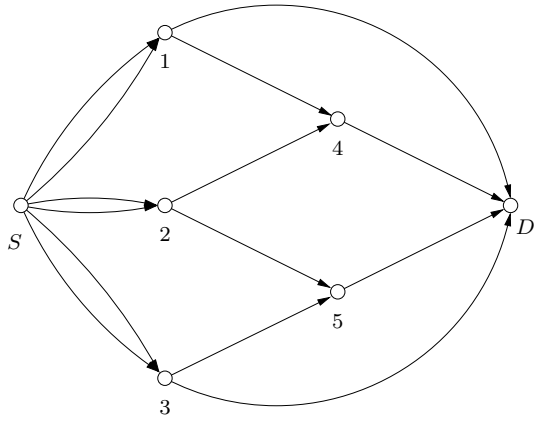

Fig. 1. The Cockroach Network. All edges have capacity 1. We consider the case that one internal node is a traitor.

In this paper we focus mainly on the single-source single-destination problem when one node may be adversarial. We make two main contributions. First, we show that linear coding is insufficient to solve this problem. We consider a class of nonlinear network codes that we denote *linear-plus*, in which each edge may carry, in addition to a linear function of the message, a nonlinear function occupying an arbitrarily small amount of link capacity. We show that even this small amount of nonlinearity is enough to increase the achievable rate for some networks. For example, capacity can be achieved in the aforementioned Cockroach Network with a linear-plus code, whereas it cannot with a linear code. A similar sort of nonlinearity was shown to help [7], which studied the problem of detecting misbehaving nodes in wireless networks. Our second contribution is to introduce

O. Kosut and L. Tong are with Cornell University, Ithaca, NY {oek2,lt35}@cornell.edu

D. Tse is with the University of California, Berkeley, CA dtse@eecs.berkeley.edu

This work is supported in part by the National Science Foundation under Award CCF-0635070 and the the Army Research Office under Grant ARO-W911NF-06-1-0346.

a very different class of nonlinear codes, called *bounded-linear*, which can outperform even linear-plus. Bounded-linear codes take advantage of special properties of probability distributions over bounded sets of integers subject to linear constraints under real arithmetic, as opposed to finite-field arithmetic.

The paper is organized as follows. Section II formally defines the problem. Section III gives the cut-set upper bound, a relatively simple upper bound for the Byzantine network coding problem with node attacks. The proceeding four sections consider four case-study networks. Section IV studies the aforementioned Cockroach network, and shows that linear codes do not achieve capacity whereas linear-plus codes do. Section V studies a network we call the Caterpillar Network, a network for which even linear-plus codes are insufficient, but bounded-linear codes achieve capacity. Section VI studies the Super Cockroach Network, a larger version of the Cockroach Network which is more difficult to solve, but its capacity can be achieved using bounded-linear codes. Section VII studies the Beetle Network, a network with a zero-capacity edge, the presence of which affects its capacity. This is a property that, as far as we know, has not been observed in non-Byzantine network coding. Finally, we conclude in Section VIII.

II. Problem Formulation

Let (V, E) be an directed acyclic graph. For each edge $e \in E$, there is an edge capacity c_e. One node in V is denoted S, the source, and one is denoted D, the destination. We wish to determine the maximum achievable throughput from S to D when any single node in $V \setminus \{S, D\}$ is the *traitor*; i.e. it is controlled by the adversary. Given a rate R and a block-length n, the message W is chosen at random from the set $\{1, \ldots, 2^{nR}\}$. Each edge e holds a value $X_e \in \{1, \ldots, 2^{nc_e}\}$. A code is be made up of three components:

1) an encoding function at the source, which produces values to place on all the output edges given the message,
2) a coding function at each internal node $v \in V \setminus \{S, D\}$, which produces values to place on all output edges from v given the values on all input edges to v,
3) and a decoding function at the destination, which produces an estimate $\hat{W}$ of the message given the values on all input edges.

Suppose $i \in V \setminus \{S, D\}$ is the traitor. It may subvert the coding function at node i by placing arbitrary values on all the output edges from this nodes. Given i, we may consider the destination's estimate $\hat{W}$ a deterministic function of the message W and the values on all edges out of node i. In particular, we define the function $\hat{W}_i$ so that

$$\hat{W} = \hat{W}_i(W, X_i) \tag{1}$$

where X_i is the set of all values on edges out of node i.

We say that a rate R is *achievable* if there exists a code operating at that rate with some block-length n such that

$$w = \hat{W}_i(w, x_i) \tag{2}$$

for all messages w, all possible traitors i, and all values x_i. That is, the destination always decodes correctly no matter what the traitor does. Let the *capacity* C be the supremum over all achievable rates.

III. Cut-Set Upper Bound

Theorem 1: Consider a cut $A \subseteq V$ with $S \in A$ and $D \notin A$. Let E_A be the set of edges that cross the cut. For two nodes $i, j \in A$, let E_i and E_j be the set of edges that originate at i and j respectively, and cross the cut. Let E'_i and E'_j be subsets of E_i and E_j respectively containing the edges e for which there is no path that flows through e followed by any edge in $E_A \setminus E_i \setminus E_j$. The following upper bounds hold on the capacity of the network:

$$C \leq \sum_{e \in E_A \setminus E'_i \setminus E'_j} c_e, \tag{3}$$

$$C \leq \sum_{e \in E_A \setminus E_i} c_e. \tag{4}$$

Note that considering (4) over all cuts gives

$$C \leq \min_{i \in V} \text{mincut}_{V \setminus \{i\}}(S; D) \tag{5}$$

where $\text{mincut}_{V \setminus \{i\}}$ is the min-cut after node i and all edges connected to it are removed from the network. However, (3) does not lead to

$$C \leq \min_{i,j \in V} \text{mincut}_{V \setminus \{i,j\}}(S; D). \tag{6}$$

Section VII provides an example for which (6) does not hold.

Proof: First we prove (3). Suppose it were not true. Then there would exist a code achieving a rate R such that

$$R > \sum_{e \in E_A \setminus E'_i \setminus E'_j} c_e. \tag{7}$$

For some subset of edges F, let X_F be the values on those edges under this code assuming all nodes are honest. We will consider two possibilities, one when node i is the traitor and it alters the values on E'_i, and one when node j is the traitor and it alters the values on E'_j. In either case, $X_{E_A \setminus E_i \setminus E_j}$ is a direct function of the message and independent of the traitor's actions, because there is no path through an edge E'_i or E'_j followed by an edge in $E_A \setminus E_i \setminus E_j$. Because of (7), there are two messages w_a and w_b for which the values on $X_{E_A \setminus E'_i \setminus E'_j}$ are the same.

Suppose that there is no path through an edge in E'_i followed by node j or through an edge in E'_j followed by node i. Hence, if one of i or j is the traitor and it changes the values on E'_i or E'_j, the output of the other node is unaffected. One possibility is that w_a is the message, node i is the traitor, and it sends $X_{E'_i}(w_b)$ along E'_i. Another possibility is that w_b is the message, node j is the traitor, and it sends $X_{E'_j}(w_a)$ along E'_j. Because $X_{E_A \setminus E'_i \setminus E'_j}(w_a) = X_{E_A \setminus E'_i \setminus E'_j}(w_b)$, in both of these possibilities, exactly the same values are on all edges in E_A. Therefore, the decoder must make an error on either w_a or w_b.

Now consider the case that either there is a path through E'_i followed by j or a path through E'_j followed by i. Both

these cannot be true simultaneously, because we assume the graph is acyclic. Assume without loss of generality that it is the former. It must be that $E'_j = E_j$, because if there were a path through an edge in E_j followed by an edge in $E_A \setminus E_i \setminus E_j$, this would mean there would be a path through E'_i followed by $E_A \setminus E_i \setminus E_j$, which contradicts the definition of E'_i. Again we consider two possibilities. If w_a is the message and i is the traitor, it may place $X_{E'_i}(w_b)$ on E'_i. Node j is honest and it may be influenced by this change, so we denote the value on E_j as $X_{E_j}(w_a, X_{E'_i}(w_b))$. The second possibility is that w_b is the message, node j is the traitor, and it places $X_{E_j}(w_a, X_{E'_i}(w_b))$ on E_j. Recall that there is no path from E_j to i, so this has no influence on any edges coming out of node i. Therefore, in both these possibilities, the same values are on E_A, so there is an error. This proves (3).

Now we prove (4). Suppose node i is the traitor and it fixes the value of X_{E_i}. Then $X_{E_A \setminus E_i}$, even if it depends on X_{E_i} in general, is a direct function of the message. If (4) does not hold, then there are two messages which produce the same value of $X_{E_A \setminus E_i}$, meaning the same value on all of E_A, which causes an error. ■

IV. The Cockroach Network

Consider again the Cockroach Network of Figure 1. We wish to determine the capacity of this network when there is at most one traitor node. It is easy to see by Theorem 1 that the capacity is no more than 2. In Section IV-A, we show that the linear capacity is $4/3$. In Section IV-B we propose a linear-plus code that achieves rate 2. Therefore the capacity is 2, and achieving capacity is impossible with a linear code.

A. Linear Capacity

A rate of $4/3$ can be achieved using just routing at internal nodes. Figure 2 shows a scheme that achieves this. Each link is used at most three times, and the message is split into four parts, denoted a, b, c, d. Observe that each of the four parts is routed through three node-independent paths. Since there is at most one traitor, at most one of the three copies of each part received at the destination will be corrupted. Hence, if the destination takes the majority vote of its three copies of each part, it is guaranteed to recover the complete message exactly.

We now show that no rates higher than $4/3$ can be achieved with linear codes. Consider any linear code. For any link (i, j), let $X_{i,j}$ be the value placed on this link. For every node i, let X_i be the set of messages on all links out of node i, and Y_i be the set of messages on all links into node i. Let $G_{X_i \to Y_j}$ be the linear transformation from X_i to Y_j, assuming all nodes behave honestly. Observe that

$$Y_D = G_{X_S \to Y_D} X_S(w) + \sum_i G_{X_i \to Y_D} e_i \qquad (8)$$

where e_i represents the difference between what a traitor places on its outgoing links and what it would have placed on those links if it were honest. Only one node is a traitor, so at most one of the e_i is nonzero. Note also that the output

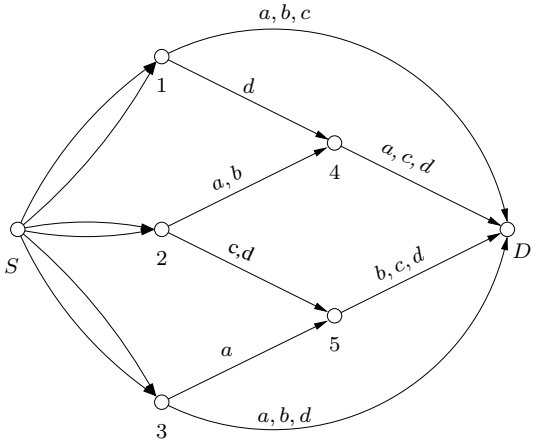

Fig. 2. A code achieving rate $4/3$ for the Cockroach Network using just routing.

values of the source X_S is a function of the message w. We claim that for any achievable rate R,

$$R \leq \frac{1}{n}\left[\text{rank}(G_{X_S \to Y_D}) - \max_{i,j} \text{rank}(G_{X_i X_j \to Y_D})\right] \qquad (9)$$

where n is the block length used by this code. To show this, first note that for any pair of nodes i, j there exist K, H_1, H_2 such that

$$G_{X_S \to Y_D} = K + G_{X_i \to Y_D} H_1 + G_{X_j \to Y_D} H_2 \qquad (10)$$

and where

$$\text{rank}(K) = \text{rank}(G_{X_S \to Y_D}) - \text{rank}(G_{X_i X_j \to Y_D}). \qquad (11)$$

That is, the first term on the right hand side of (10) represents the part of the transformation from X_S to Y_D that cannot be influenced by X_i or X_j. Consider the case that $\text{rank}(K) < R$. Then there must be two messages w_1, w_2 such that $KX_S(w_1) = KX_S(w_2)$. If the message is w_1, node i may be the traitor and set

$$e_i = H_1(X_S(w_2) - X_S(w_1)). \qquad (12)$$

Alternatively, if the message is w_2, node j may be the traitor and set

$$e_j = H_2(X_S(w_1) - X_S(w_2)). \qquad (13)$$

In either case, the value received at the destination is

$$Y_D = KX_S(w_1) + G_{X_i \to Y_D} H_1 X_S(w_2) + G_{X_j \to Y_D} H_2 X_S(w_1).$$

Therefore, these two cases are indistinguishable to the destination, so it must make an error for at least one of them. This proves (9).

Now we return to the specific case of the Cockroach Network. Observe that the $X_{4,D}$ is a linear combination of $X_{1,4}$ and $X_{2,4}$. Let k_1 be the number of dimensions of $X_{4,D}$ that depend only on $X_{1,4}$ and are independent of $X_{2,4}$. Let k_2 be the number of dimensions of $X_{4,D}$ that depend only on $X_{2,4}$, and let k_3 be the number of dimensions that depend on both $X_{1,4}$ and $X_{2,4}$. Certainly $k_1 + k_2 + k_3 \leq n$. Similarly, let

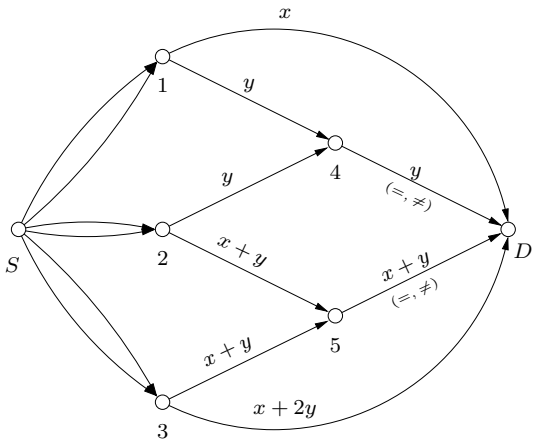

Fig. 3. A nonlinear code for the Cockroach Network achieving the capacity of 2.

l_1, l_2, l_3 be the number of dimensions of $X_{5,D}$ that depend only on $X_{2,5}$, that depend only on $X_{3,5}$, and that depend on both respectively. Finally, let m_1 and m_2 be the number of dimensions of $X_{1,D}$ and $X_{3,D}$ respectively.

We may write the following:

$$\begin{aligned}
\text{rank}(G_{X_S \to Y_4}) - \text{rank}(G_{X_2, X_3 \to Y_4}) &\leq m_1 + k_1, \\
\text{rank}(G_{X_S \to Y_4}) - \text{rank}(G_{X_1, X_3 \to Y_4}) &\leq k_3 + l_1, \\
\text{rank}(G_{X_S \to Y_4}) - \text{rank}(G_{X_1, X_2 \to Y_4}) &\leq l_3 + m_2.
\end{aligned}$$

Therefore, using (9), any achievable rate R is bounded by

$$R \leq \frac{1}{n} \min\{m_1 + k_1, k_3 + l_1, l_3 + m_2\} \tag{14}$$

subject to

$$k_1 + k_2 + k_3 \leq n, \tag{15}$$
$$l_1 + l_2 + l_3 \leq n, \tag{16}$$
$$m_1 \leq n, \tag{17}$$
$$m_2 \leq n. \tag{18}$$

It is not hard to show that this implies $R \leq 4/3$.

B. Capacity-Achieving Linear-Plus Code

We now introduce a linear-plus code to achieve the capacity of 2. We work in the finite field of p elements. Let the message w be a $2k$-length vector split into two k-length vectors x and y. We will use a block length large enough to place one of $2p^k$ values on each link. In particular, enough to place on a link some linear combination of x and y plus one additional bit. For large enough k, this extra bit becomes insignificant, so we still achieve a rate of 2.

The scheme is shown in Figure 3. Node 4 receives the vector y from both 1 and 2. It forwards one of these copies to D (it does not matter which). In addition, it forwards one additional bit comprised of one of the special symbols $=$ or $\neq$. If the two received values of y agree, it forwards $=$, otherwise it sends $\neq$. The link $(4, D)$ can accommodate this, since it may have up to $2p^k$ messages placed on it. Node 5 does the same with its two copies of the vector $x + y$.

The destination's decoding strategy depends on which of the two messages sent from nodes 4 and 5 are $=$ or $\neq$, as follows:

- If the message from node 4 is $\neq$ but the message from 5 is $=$, then the traitor must be either node 1, 2, or 4. In any case, the vector $x + 2y$ received from node 3 is certainly trustworthy. However, $x + y$ is trustworthy as well, because even if node 2 is the traitor, its transmission must have matched whatever was sent by node 3, because if not node 5 would have transmitted $\neq$. Since it did not, we can trust both $x+y$ and $x+2y$, from which the destination can decode the message $w = (x, y)$.
- If the message from 5 is $\neq$ but the message from 4 is $=$, then we are in the symmetric situation and can reliably decode w from x and y.
- If both the messages from 4 and 5 are $\neq$, then we know the traitor is node 2, in which case we can trust x and $x + 2y$, and hence decode w.
- If both messages are $=$, then the destination cannot eliminate any node as a possible traitor. However, at most one of $x, y, x+y, x+2y$ can have been corrupted by the traitor, because no node controls more than one of the vectors received at the destination. For instance, if node 1 is the traitor, it may choose whatever it wants for x, and the destination would never know. However, node 1 cannot impact the value of y without inducing a $\neq$, because its transmission to node 4 is verified against that from node 2. Similarly, node 3 controls $x + 2y$ but not $x + y$. Nodes 4 and 5 control only y and $x + y$ respectively. Node 2 controls nothing, because both y and $x + y$ are checked against other transmissions. Therefore, if the destination can find three of $x, y, x+y, x+2y$ that all agree on the message w, then this message must be the truth because only one of them could be corrupted, and w can be decoded from the other two. Conversely, there must be a group of three of $x, y, x + y, x + 2y$ that agree, because at most one has been corrupted. Hence, the destination can always decode w.

V. The Caterpillar Network

The Caterpillar Network is shown in Figure 4. We consider a slightly different version of the node-based Byzantine attack on this network: at most one node may be a traitor, but only nodes 1–4. Theorem 1 gives an upper bound of 2 for this network. We omit the proof, but it can be shown that no linear-plus code can achieve a higher rate than $1\frac{5}{6}$ for this network. In Section V-A, we illustrate the class of bounded-linear codes with a code that achieves the capacity of 2. In Section V-B, we prove Lemma 1, which is stated in Section V-A and provides a key property of the distributions we use in bounded-linear codes.

A. The Bounded-Linear Code

Fix an integer k and let the random variables $X, Y, Z, W \in \{-k, \ldots, k\}$ have the joint distribution $p(xyzw)$ uniform

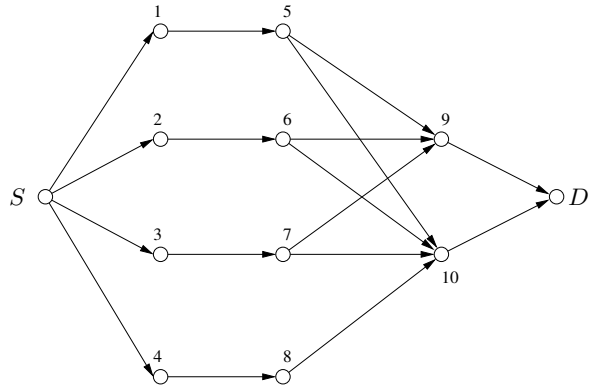

Fig. 4. The Caterpillar Network. All edges have unit capacity. One node may be a traitor, but only nodes 1–4.

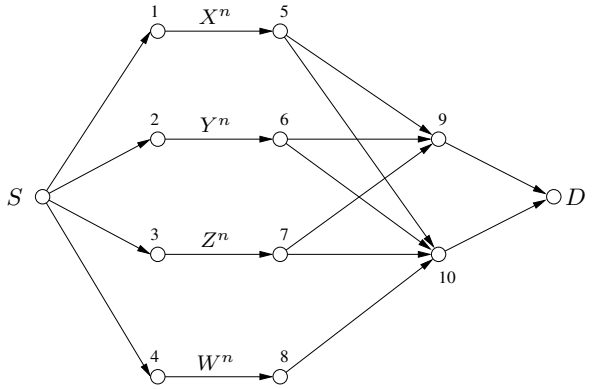

Fig. 5. A bounded-linear code for the Caterpillar Network achieve the capacity of 2.

over all X, Y, Z, W satisfying the two linear constraints

$$X + Y + Z = 0, \tag{19}$$

$$X + 2Y + 3W = 0. \tag{20}$$

The following lemma gives a key property of distributions of this type.

Lemma 1: Let $X_1, \ldots, X_m \in \{-k, \ldots, k\}$ have distribution $p(x_1 \ldots x_m)$ such that

$$\sum_{i=1}^{m} c_i X_i = 0$$

with probability 1 for integers c_i. For some other distribution $q(x_1 \cdots x_m)$, if

$$q(x_1 x_i) = p(x_1 x_i) \text{ for } i = 2, \ldots, m, \tag{21}$$

$$q(x_2 \cdots x_m) = p(x_2 \cdots x_m) \tag{22}$$

then

$$q(x_1 \cdots x_m) = p(x_1 \cdots x_m).$$

For $p(xyzw)$, the relevant property from Lemma 1 that we will use to prove correctness of our code for the Caterpillar Network will be that for some $q(xyz)$,

$$\text{if } q(xy) = p(xy), q(xz) = p(xz), q(yz) = p(yz), \quad \text{then} q(xyz) = p(xyz). \tag{23}$$

Observe that given any two of X, Y, Z, W, the other two are fixed. The values of the pair (x, y) for which $p(x, y) > 0$ are constrained by the fact that Y and Z must be in $\{-k, \ldots, k\}$. In particular, the set of (x, y) with positive probability is given by

$$\{(x, y) : |x + 2y| \leq k\}. \tag{24}$$

The size of this set is $\mathcal{O}(k^2)$. Therefore, since p is uniform over all allowable values,

$$H(XYZW) = H(XY) = \log(\mathcal{O}(k^2)). \tag{25}$$

Let n be a multiple of the denominator used in p. Let $T^n(XYZW) \in \{-k, \ldots, k\}^4$ be the set of sequences (x^n, y^n, z^n, w^n) with joint type exactly equal to p. Because of our choice of n, this set is nonempty, and

$$|T^n(XYZW)| \geq \frac{1}{(n+1)^{(2k+1)^4}} 2^{nH(XYZW)}. \tag{26}$$

We will design a code in which the number of messages is $|T^n(XYZW)|$, and each edge in the network carries one n-length sequence of one of the variables X, Y, Z, W. Therefore the achieved rate will be given by

$$R^{(k,n)} = \frac{\log |T^n(XYZW)|}{\log(2k+1)^n} \tag{27}$$

$$\geq \frac{nH(XYZW) - (2k+1)^4 \log(n+1)}{n \log(2k+1)} \tag{28}$$

$$= \frac{\log(\mathcal{O}(k^2)) - (2k+1)^4 \frac{\log(n+1)}{n}}{\log(2k+1)}. \tag{29}$$

Observe that

$$\lim_{k \to \infty} \lim_{n \to \infty} R^{(k,n)} = 2. \tag{30}$$

Therefore, our code will achieve rate 2.

The code we propose for the Caterpillar Network is shown in Figure 5. Each message is associated with an element of $T^n(XYZW)$. Given the message, the source transmits the corresponding sequences to nodes 1–4, which, if honest, forward those sequences to nodes 5–8. Nodes 5–8 forward their received sequences to nodes 9 and 10. Let $q(xyz)$ be the observed joint type of (X^n, Y^n, Z^n) at nodes 9 and 10. Observe that q may not equal p, because one of nodes 1–3 may be the traitor. Furthermore, since nodes 5–8 cannot be traitors, nodes 9 and 10 observe the same value of q. If $q(xyz) = p(xyz)$, then node 9 forwards X^n to D, and node 10 forwards Y^n to D. If $q(xyz) \neq p(xyz)$, then by (23), one of $p(xy) \neq q(xy)$, $p(xz) \neq q(xz)$, or $p(yz) \neq q(yz)$ must hold. Node 9 forwards to D whichever of X^n, Y^n, Z^n was not involved in the mismatched joint type, and node 10 forwards W^n to D. In either case, the destination receives two of X^n, Y^n, Z^n, W^n, from which it can reconstruct the identity of the message.

We now show that nodes 9 and 10 never forward an incorrect sequence to D. Only one of X^n, Y^n, Z^n can be

corrupted by the traitor, and each of X, Y, Z is a deterministic function of the other two, so if $q(xyz) = p(xyz)$, then the values of X^n and Y^n observed by nodes 9 and 10 must be the true ones. On the other hand, if one of the three pairwise marginal types of $q(xyz)$ do not match, then the traitor must be one of the two nodes corresponding to the pair of variables. Therefore the non-disagreeing variable is trustworthy, as is W^n.

B. Proof of Lemma 1

We will use the notation x_i^j to denote the vector $[x_i, x_{i+1}, \ldots, x_j]^T$, and x^m to denote x_1^m. Also we refer to the vector $[c_1 \ldots, c_m]$ by c.

Observe that if $cx^m \neq 0$, then

$$q(x^m) \leq q(x_2^m) \tag{31}$$
$$= p(x_2^m) \tag{32}$$
$$= \sum_{x_1'} p(x_1' x_2^m) \tag{33}$$
$$= p(x^m) \tag{34}$$

where (32) holds by (22) and (34) holds because for all $x_1' \neq x_1$, the equality condition does not hold, so $p(x_1' x_2^m) = 0$. Therefore, if we let

$$v(x^m) = \begin{cases} q(x^m) & \text{if } cx^m \neq 0 \\ p(x^m) - q(x^m) & \text{if } cx^m = 0 \end{cases}$$

then $v(x^m) \geq 0$, and it will be enough to show that

$$v(x^m) = 0 \tag{35}$$

for all x^m.

Fix x_1, x_i for some $i \in \{2, \ldots, m\}$. We may write

$$q(x_1 x_i) = \sum_{x_2^{i-1} x_{i+1}^m} q(x^m) \tag{36}$$
$$= \sum_{x_2^{i-1} x_{i+1}^m : cx^m \neq 0} q(x^m) + \sum_{x_2^{i-1} x_{i+1}^m : cx^m = 0} q(x^m). \tag{37}$$

Furthermore,

$$p(x_1 x_i) = \sum_{x_2^{i-1} x_{i+1}^m : cx^m = 0} p(x^m). \tag{38}$$

Therefore by (21)

$$\sum_{x_2^{i-1} x_{i+1}^m : cx^m \neq 0} v(x^m) = \sum_{x_2^{i-1} x_{i+1}^m : cx^m = 0} v(x^m). \tag{39}$$

By a similar argument using (22), if $cx^m = 0$, then

$$v(x^m) = \sum_{x_1' \neq x_1} v(x_1' x_2^m). \tag{40}$$

We proceed to prove (35) by induction on x_1. Suppose that $v(x_1' x_2^m) = 0$ for all $x_1' > x_1$ and all x_2^m. If we adopt the convention that $v(x_1' x_2^m) = 0$ if $x_1' > k$, then this certainly holds for $x_1 = k$, so it will be enough to show that $v(x_1 x_2^m) = 0$ for all x_2^m. Assume without loss of generality that $c_1 > 0$. If x_2^m is such that $c[x_1, x_2^m] < 0$, then the value of x_1' for which $c[x_1', x_2^m] = 0$ satisfies $x_1' > x_1$. Hence

$$0 = v(x_1' x_2^m) = \sum_{x_1'' \neq x_1} v(x_1'' x_2^m) \tag{41}$$

where we have used (40). Therefore, $v(x_1 x_2^m) = 0$ if $c[x_1\ x_2^m] < 0$, so (39) becomes

$$\sum_{x_2^{i-1} x_{i+1}^m : cx^m > 0} v(x^m) = \sum_{x_2^{i-1} x_{i+1}^m : cx^m = 0} v(x^m). \tag{42}$$

Since this holds for all x_i, we may write

$$\sum_{x_2^m : cx^m > 0} (c_i x_i + \tfrac{c_1}{m-1} x_1) v(x^m)$$
$$= \sum_{x_2^{i-1} x_{i+1}^m : cx^m = 0} (c_i x_i + \tfrac{c_1}{m-1} x_1) v(x^m). \tag{43}$$

Summing (43) over $2 \leq i \leq m$ gives

$$\sum_{x^m : cx^m > 0} cx^m v(x^m) = \sum_{x^m : cx^m = 0} cx^m v(x^m) = 0. \tag{44}$$

Since $v(x^m) \geq 0$ and $cx^m > 0$ for all terms on the left hand side of (44), $v(x_1 x_2^m) = 0$ if $c[x_1, x_2^m] < 0$. Therefore the left hand side of (39) for any i is 0, so $v(x_1 x_2^m) = 0$ if $c[x_1, x_2^m] = 0$. This completes the argument that $v(x_1 x_2^m) = 0$ for all x_2^m.

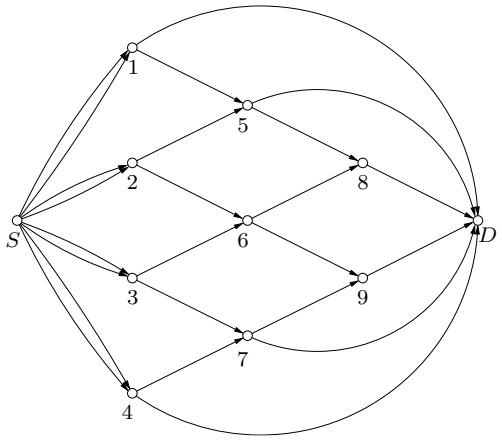

Fig. 6. The Super Cockroach Network. All edges have capacity 1.

VI. The Super Cockroach Network

Figure 6 shows the Super Cockroach Network. It has the same basic structure as the Cockroach Network, but with one additional stage. It is easy to see that the cut-set bound is 4. We omit the proof, but a bounded-linear code exists that achieves rate 4. It makes use of many of the same ideas as in Section V, allowing internal nodes to meaningfully compare pairs of variables even when they are nearly statistically independent.

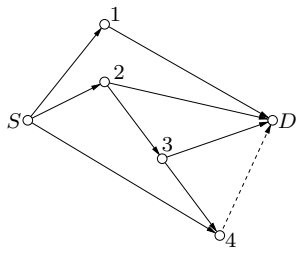

Fig. 7. The Beetle Network. All edges have capacity 1 except the dashed edge, which has zero capacity.

VII. The Beetle Network

The Beetle Network, shown in Figure 7, under the presence of a single traitor node, has two interesting properties. First, the value of the cut-set bound given in (4), which is 1, does not match the value of the expression in (6), which is 0. Second, it has a zero capacity edge, the presence of which has a positive effect on the capacity. That is, the capacity of this network, as we will demonstrate, is 1, but if the zero-capacity edge $(4, D)$ were removed, the capacity would be 0, as can easily be verified by Theorem 1. As we will show, the explanation for this is similar to the reason linear-plus codes can outperform linear codes: that a small addition to a code, even if it takes up an arbitrarily small amount of link capacity, can strictly increase the achieved rate.

We now present a simple linear-plus code for the Beetle Network which achieves rate 1. Each unit-capacity edge carries a copy of the message w. That is, the source sends w along all three of its output links, and nodes 1, 2, and 3 each receive one copy of w and forward it along all of their output links. Node 4 receives a copy of w from the source and and one from node 3. It compares them and sends to the destination one of the symbols $=$ or $\neq$ depending on whether the two copies agreed or not. Because w may be a vector of arbitrary length, sending this single bit along edge $(4, D)$ takes zero rate, so we do not exceed the edge capacity.

We now give the decoding procedure. Let w_1, w_2, and w_3 be the values of w received at the destination from nodes 1, 2, and 3 respectively. If either $w_2 \neq w_3$ or the destination receives $\neq$ from node 4, then certainly the traitor must by one of nodes 2, 3, or 4, so w_1 is trustworthy and the destination decodes from it. Now consider the case that $w_2 = w_3$ and the destination receives $=$ from node 4. The destination decodes from w_2 or w_3. Certainly if the traitor is either node 1 or 4, then $w_2 = w_3 = w$. If the traitor is node 3, then $w_2 = w$, so we still decode correctly. If the traitor is node 2, then it must send the same value of w to both the destination and node 3, because node 3 simply forwards its copy to the destination, and we know $w_2 = w_3$. Furthermore, this value of w must be the true one, because otherwise node 4 would observe that the copy sent along edge $(3, 4)$ is different from that sent from the source, so it would transmit $\neq$ to the destination. Since it did not, node 2 cannot have altered any of its output values. Therefore the destination always decodes correctly.

VIII. Conclusion

With the four examples we studied, we saw that generalizing the types of Byzantine attacks on network coding can substantially change the nature of the problem. As opposed to edge-based Byzantine attacks, node-based attacks require the use of nonlinear codes, and sometimes even nonlinear codes that hardly resemble standard linear codes, such as bounded-linear codes.

We looked primarily at the case that one node in the network is a traitor, but this can be generalized further. Certainly one could study the problem with t node traitors, or with t node traitors taken from only a subset of nodes in the network, such as in the Caterpillar Network example. Most generally, one could specify a list of sets of edges, from which any single set may be selected by the adversary to control. Whether this very general problem is still harder than the node-based problem is unknown.

In addition, in this paper we considered only one very specific adversary model, that being the omniscient, or worst case, adversary. In [6], two weaker traitor models were considered for the edge problem, both resulting in higher achievable rates. It would be interesting to see whether a similar nonlinearity is required with those traitor models for general Byzantine attacks as well.

References

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204–1216, 2000.

[2] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382–401, 1982.

[3] D. Dolev, "The Byzantine generals strike again," *Journal of Algorithms*, vol. 3, no. 1, pp. 14–30, 1982.

[4] N. Cai and R. W. Yeung, "Network error correction, part I: Basic concepts and upper bounds," *Comm. in Inf. and Syst.*, vol. 6, no. 1, pp 19–36, 2006.

[5] N. Cai and R. W. Yeung, "Network error correction, part II: Lower bounds," *Comm. in Inf. and Syst.*, vol. 6, no. 1, pp 37–54, 2006.

[6] S. Jaggi, et al, "Resilient Network Coding in the Presence of Byzantine Adversaries," in *Proc. INFOCOM*, pp. 616–624, 2007.

[7] G. Liang and N. H. Vaidya, "When WatchDog Meets Coding," *Technical Report*, May 2009.