Processing Delays Do Not Degrade Network Error-Correction Capacity in Directed Networks

Zhiqing Xiao, Yunzhou Li, Xin Su, and Jing Wang

Abstract—**This letter considers the network error-correction capacity in the presence of Byzantine attacks. Unlike prior papers, which assume no delay in any part of networks, we consider the effects of processing delays in the nodes. We prove that these processing delays have no impact on the network error-correction capacity region for any directed network.**

Index Terms—**Network error-correction, processing delay, capacity region.**

I. Introduction

NETWORK error-correction uses redundant link resource to combat potential Byzantine adversaries in communication networks. In recent years, there have been many results on the network error-correction capacity (or the supremum of the achievable rates of network error-correction codes) for specific networks [1]–[6]. However, the network error-correction capacity in general networks is still an open problem.

In the aforementioned works [1]–[6], the encoding, recoding, and decoding operations, as well as other potential operations such as modulation, are assumed to be executed instantaneously, and there are no delays in any part of the networks. However, for those capacity-attaining codes, the coding always contain complex operations, which are likely to introduce processing delays in the nodes.

The impact of delay in memoryless networks were considered in [7], [8]. Both papers considered the memoryless channels that are specified by conditional distributions. They proved that the existence of delays in arbitrary parts of the networks do not change the capacity region. In their papers, two concepts, *delay profile* and *capacity region*, are introduced. Specifically, the delay profile denotes the delay in the input of every node, while the capacity region is the closure of the set of the achievable rate tuples with delays into consideration. Afterward, they used reductions to show the equivalence between the capacity region with and without delays: If a rate tuple is achievable in the network without delay, then the same tuple is also (asymptotically) achievable in the network with an arbitrary finite delay. Therefore, delays will not degrade the capacity region.

In this paper, we consider the impact of delay on noiseless networks in the presence of Byzantine attacks. The model in this paper follows the conventional model in the previous researches of network error-correction codes, where the adversaries are able to falsify the signals transmitted at the network channels. What's worse, the adversaries have unlimited computing power, and are smart enough to attack in a way that is least favorable to the defenders. Therefore, the adversarial errors in different uses of a channel are highly correlated, and the channel is not memoryless any more. Since [7], [8] only work for memoryless networks, their results are inapplicable for our Byzantine adversary model.

The main contribution of this paper is to show that the processing delays in the nodes have no impact on the network error-correction capacity region in a directed network with Byzantine attacks. To be specific, for any network topology, link capacities, and erroneous link patterns, the processing delays of the network nodes do not affect the network error-correction capacity region for all connections, including the unicast connections and the multicast connections.

In our analysis, we will adopt the concepts of *delay profile* and *capacity region* in [7], [8], and use the idea of reduction to prove the equivalence between the capacity region with and without delay.

The rest of the paper is organized as follows. Section II-A formulates the system model of network error-correction. Section II-B presents the main result on the impact of the delays. The proof of the main result is presented in Section III. Section IV discusses some variants of the main result. The paper concludes in Section V.

II. System Model and Main Result

A. Model Set-Up

In this paper, a network, denoted by $\mathcal{N} = (\mathcal{V}, \mathcal{E})$, is a directed graph. The graph can be either cyclic or acyclic. Every direct edge $e \in \mathcal{E}$ represents a noiseless channel leading from a node (denoted as $\mathrm{Tail}(e)$) to another node (denoted as $\mathrm{Head}(e)$). The set of incoming edges and outgoing edges of a node $v \in \mathcal{V}$ are denoted as $\mathrm{In}(v)$ and $\mathrm{Out}(v)$ respectively.

Each edge $e \in \mathcal{E}$ has a weight c_e to denote its capacity. For an edge e with capacity c_e, it transmits a value within its alphabet

Manuscript received December 26, 2014; accepted April 17, 2015. Date of publication April 23, 2015; date of current version December 8, 2015. This work was supported by the National Basic Research Program of China (973 Program) under Grant 2013CB329002, the National High Technology Research and Development Program of China (863 Program) under Grant 2015AA01A706, the National S&T Major Project under Grant 2014ZX03003002-002, the Program for NCET in University under Grant NCET-13-0321, the International Science and Technology Cooperation Program under Grant 2012DFG12010, and the Tsinghua Research Funding under Grant 2010THZ03-2. The associate editor coordinating the review of this paper and approving it for publication was B. Bellalta.

Z. Xiao is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: xzq.xiaozhiqing@gmail.com).

Y. Li, X. Su, and J. Wang are with the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China (email: liyunzhou@tsinghua.edu.cn; suxin@tsinghua.edu.cn; wangj@tsinghua.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2015.2425872

1558-2558 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

$\{1,\ldots,2^{c_e}\}$ in every channel use. The input signals and the output signals of the edge e are denoted as $\{X_i(e), i=1,2,\ldots\}$ and $\{Y_i(e), i=1,2,\ldots\}$ respectively. Let $X^j(e)=\{X_i(e, 1\leq i\leq j\}$. For $E\subseteq\mathcal{E}$, define $X_i(E)=\{X_i(e): e\in E\}$.

Each node $v\in\mathcal{V}$ uses its incoming signals to determine its output signals. Specifically, for $i\in\{1,\ldots,n\}$, if no messages originate at node v, $X_i(\text{Out}(v))$ is merely determined by $Y^{i-1}(\text{In}(v))$. If a message M originates at node v, $X_i(\text{Out}(v))$ is determined by both $Y^i(\text{In}(v))$ and the message M. Additionally, if node v needs to recover a message, the recovered message $\hat{M}$ is determined by both $Y^n(\text{In}(v))$ and the messages that originate at node v. All these operations are henceforth collectively called the "code." n is called the block-length of the code.

As mentioned, for every node in the network, there exists a delay from the arrival of its incoming signals to the transmission of its outgoing signals. Hereby, we define the delay d_e for every edge e according to the delay of the node $\text{Tail}(e)$. For a particular node v, the delay of its outgoing edges may be different. Accordingly, we further define the delay profile (denoted as $\mathbf{d}=(d_e: e\in E)$) as a vector of non-negative real numbers, which indicate the delay of edges in the network. The network with its associated delay profile can be denoted by $\mathcal{N}(\mathbf{d})$.

Remark 1: In this paper, we use boldface symbols to represent delay profiles. $\mathbf{0}$ is the delay profile where all entries are zero. $\mathbf{d}_e=(0,\ldots,0,1,0,\ldots,0)$ is the delay profile where only the delay of edge e is 1 and others are all zero. Define the ceiling of a delay profile as $\lceil\mathbf{d}\rceil=(\lceil d_e\rceil: e\in\mathcal{E})$. For two profiles $\mathbf{d}^{(1)}$ and $\mathbf{d}^{(2)}$, we say $\mathbf{d}^{(1)}\leq\mathbf{d}^{(2)}$ when $d_e^{(1)}\leq d_e^{(2)}$ holds for every edge e. By definition, for any delay profile $\mathbf{d}$, we have $\mathbf{d}\geq\mathbf{0}$.

The adversarial model in this paper is exactly the same with prior papers [3], [4]. Assumptions include: (1) The adversary controls a set of edges or nodes in the network, and assigns their outputs as whatever it wants. The combination of erroneous edges and nodes is fixed for all time. In other words, for a particular edge e, adversary can directly determine its input signal $\{X_i(e), i=1,2,\ldots\}$ if it controls the node $\text{Tail}(e)$, and it can directly determine its output signals $\{Y_i(e), i=1,2,\ldots\}$ if it controls the edge e. (2) The adversary knows both the network error-correction code and the messages *a prior*. (3) The adversary has infinite computation power, and can arbitrarily determine the outputs of the erroneous edges and nodes.

Remark 2: This adversary model is usually known as omniscient Byzantine adversary [3], [4]. The adversary is very intelligent and knowledgable, and its only limitation is the nodes and edges it controls. This assumption focuses on the combinatorial behavior of the adversary: the combination of erroneous nodes and edges should belong to a set of all possible combinations. Without this limitation, the adversary would fully control the network, which makes the network impossible for any reliable communications. Additionally, ignoring the adversary's other limitations (i.e., we do not restrict adversary's computing power or its knowledge of messages) may probably exaggerate the threatens of the adversary, but it is sensible when we need a conservative estimation of a system.

There are multiple unicast/multicast connections in the network, each of which has its own source node and destination node(s). The set of connections is denoted as $\mathcal{K}$. All connections in the network communicate simultaneously.

Nodes use a network error-correction code to enforce these connections. Inter-connection coding is allowed. For a set of combinations of edges and nodes (denoted as $\mathcal{A}$) and a network error-correction code, if the code can correctly recover the messages no matter which combination in $\mathcal{A}$ is controlled by the adversary, we say the code has network error-correction capability $\mathcal{A}$. For example, if a code is able to correctly recover the message when at most one edge in the network is controlled by the adversary, the code has error-correction capability $\{\{e\}: e\in\mathcal{E}\}$. (Please refer to [3], [4] for details about the error-correction capability.) We henceforth fix the error-correction capability $\mathcal{A}$, and no longer explicitly mention it when there is no ambiguity.

A rate tuple $(R_k: k\in\mathcal{K})$ is achievable if there exists a $(n,\{2^{nR_k}: k\in\mathcal{K}\})$ network error-correction code such that:

- the block-length of the code is n;
- the message set for connection $k\in\mathcal{K}$ is $\{1,\ldots,2^{nR_k}\}$;
- the code has the desired error-correction capability $\mathcal{A}$.

The network error-correction capacity region (denoted as $\mathscr{R}(\mathcal{N}(\mathbf{d}))$) is defined as the closure of the set of all achievable rate tuples.

Remark 3: The network error-correction capacity region is defined as such to be consistent to the definition of capacity region in [7], [8]. Here, (i) $\mathscr{R}(\mathcal{N}(\mathbf{d}))$ is not only determined by $\mathcal{N}(\mathbf{d})$, but also determined by $\mathcal{K}$ and $\mathcal{A}$ implicitly; (ii) The set $\mathscr{R}(\mathcal{N}(\mathbf{d}))$ is a closed set, and the rate tuples therein are asymptotically achievable rather than directly achievable. (Please refer to [9, Definition 2], for its details.) This definition is fully compatible with the definition of network error-correction capacity in [1]–[4].

B. Main Result

Theorem 1: For a given network $\mathcal{N}$, a set of connections $\mathcal{K}$, and a delay profile $\mathbf{d}$,

$$\mathscr{R}\left(\mathcal{N}(\mathbf{d})\right)=\mathscr{R}\left(\mathcal{N}(\mathbf{0})\right).$$

The proof of this theorem is provided in Section III.

This theorem transfers the similar result from the cases of memoryless probabilistic channels to the network error-correction cases, and shows that the processing delays do not change the network error-correction capacity region.

Remarkably, the model here (as well as those in [1]–[4] does not depend on the existence of the probabilistic distributions of any variables. Specifically, we do not assume that the message follows any probability distribution (say, the uniform distribution). Accordingly, the concept of "error probability" is not defined, either. Although this seems slightly non-standard compared to traditional information theory, it reveals the combinatorial nature of the network error-correction problem. Please also notice that this approach is fully compatible with the standard statistical assumptions, which will be discussed in Section IV.

TABLE I
THE CODE IN THE NETWORK WITH DELAY $\mathbf{d}$

time	$X(e)$	$X(\mathcal{E}\setminus\{e\})$	$Y(e)$	$Y(\mathcal{E}\setminus\{e\})$
1	$X_1(e)$	$X_1(\mathcal{E}\setminus\{e\})$	$Y_1(e)$	$Y_1(\mathcal{E}\setminus\{e\})$
2	$X_2(e)$	$X_2(\mathcal{E}\setminus\{e\})$	$Y_2(e)$	$Y_2(\mathcal{E}\setminus\{e\})$
3	$X_3(e)$	$X_3(\mathcal{E}\setminus\{e\})$	$Y_3(e)$	$Y_3(\mathcal{E}\setminus\{e\})$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
n	$X_n(e)$	$X_n(\mathcal{E}\setminus\{e\})$	$Y_n(e)$	$Y_n(\mathcal{E}\setminus\{e\})$

III. PROOF OF THE MAIN RESULT

Lemma 1: Given a network $\mathcal{N}$, connections $\mathcal{K}$, and two delay profiles $\mathbf{d}^{(1)}$ and $\mathbf{d}^{(2)}$ such that $\mathbf{d}^{(1)} \geq \mathbf{d}^{(2)}$,

$$\mathscr{R}\left(\mathcal{N}\left(\mathbf{d}^{(1)}\right)\right) \subseteq \mathscr{R}\left(\mathcal{N}\left(\mathbf{d}^{(2)}\right)\right).$$

Proof: Assume that a rate tuple $(R_k : k \in \mathcal{K})$ is achievable when the delay profile is $\mathbf{d}^{(1)}$. When the delay profile is $\mathbf{d}^{(2)}$, the same rate tuple is also achievable since the nodes can deliberately postpone their output by $\mathbf{d}^{(1)} - \mathbf{d}^{(2)}$, which reduces to the case of $\mathbf{d}^{(1)}$. Therefore, any rate tuple that is achievable when the delay profile is $\mathbf{d}^{(1)}$ is also achievable when the delay profile is $\mathbf{d}^{(2)}$. Then the lemma can be proved by taking the closure of the achievable rate tuples. ∎

Lemma 2: Given a network $\mathcal{N}$, connections $\mathcal{K}$, a delay profile $\mathbf{d}$, and an edge e in the network,

$$\mathscr{R}(\mathcal{N}(\mathbf{d})) \subseteq \mathscr{R}(\mathcal{N}(\mathbf{d}+\mathbf{d}_e)).$$

Proof: Now we show that, for a rate tuple $(R_k : k \in \mathcal{K})$ that can be attained when the delay profile is $\mathbf{d}$, the same rate tuple is also asymptotically achievable when the delay profile is $\mathbf{d}+\mathbf{d}_e$.

Assume that Table I is the network error-correction code that attains an achievable rate tuple $(R_k : k \in \mathcal{K})$ when the delay profile is $\mathbf{d}$. The block length of the code is n. In time $i \in \{1, \ldots, n\}$, the input signals and the output signals of edges $\mathcal{E}$ are denoted as $X_i(\mathcal{E})$ and $Y_i(\mathcal{E})$ respectively. If there are no errors, the input of edge e at time 1, $X_1(e)$, is only determined by messages that originate at node Tail(e); the input of the edge e at time 2, $X_2(e)$, is determined by both the aforementioned messages and all incoming signals of Tail(e) at time 1; the output signal of edge e at time 1, $Y_1(e)$, is identical to $X_1(e)$; and so forth. This table is actually an instance of signals in the whole network. Unlike previous researches, where the network error-correction codes were specified by encoders and decoders explicitly, here we show a snapshot of effects of the code. In this snapshot, the appearance of $X_i(\mathcal{E})$, the transmitted signal in time i, is after the appearance of $Y^{i-1}(\mathcal{E})$. This relationship is used to imply the casuality of coders in latter contents.

Now we consider the effect of the adversary. As mentioned before, the Byzantine adversary controls a combination of nodes and edges. If the adversary controls a node, say the node Tail(e), it can alter all signals in column $X(e)$ of Table I. Similarly, if the adversary controls an edge, say the edge e, it can alter all signals in column $Y(e)$ of Table I. Such alterations can propagate to signals in other columns. For example, errors in $X(e)$ may affect $Y(e)$ since $Y(e)$ is determined by $X(e)$, while errors in $Y(e)$ may indirectly affect $X(e)$ since $X(e)$ is partly determined by all input signals of the node Tail(e), which are in turn determined by $Y(e)$.

TABLE II
A CODE IN THE NETWORK WITH DELAY $\mathbf{d}+\mathbf{d}_e$

time	$X(e)$	$X(\mathcal{E}\setminus\{e\})$	$Y(e)$	$Y(\mathcal{E}\setminus\{e\})$
1		$X_1(\mathcal{E}\setminus\{e\})$		$Y_1(\mathcal{E}\setminus\{e\})$
2	$X_1(e)$		$Y_1(e)$	
3		$X_2(\mathcal{E}\setminus\{e\})$		$Y_2(\mathcal{E}\setminus\{e\})$
4	$X_2(e)$		$Y_2(e)$	
5		$X_3(\mathcal{E}\setminus\{e\})$		$Y_3(\mathcal{E}\setminus\{e\})$
6	$X_3(e)$		$Y_3(e)$	
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$2n-1$		$X_n(\mathcal{E}\setminus\{e\})$		$Y_n(\mathcal{E}\setminus\{e\})$
$2n$	$X_n(e)$		$Y_n(e)$	

TABLE III
THE CODE IN THE NETWORK WITH DELAY $\mathbf{d}+\mathbf{d}_e$

time	$X(e)$	$X(\mathcal{E}\setminus\{e\})$	$Y(e)$	$Y(\mathcal{E}\setminus\{e\})$
1		$X_1(\mathcal{E}\setminus\{e\})$		$Y_1(\mathcal{E}\setminus\{e\})$
2	$X_1(e)$	$\tilde{X}_1(\mathcal{E}\setminus\{e\})$	$Y_1(e)$	$\tilde{Y}_1(\mathcal{E}\setminus\{e\})$
3	$\tilde{X}_1(e)$	$X_2(\mathcal{E}\setminus\{e\})$	$\tilde{Y}_1(e)$	$Y_2(\mathcal{E}\setminus\{e\})$
4	$X_2(e)$	$\tilde{X}_2(\mathcal{E}\setminus\{e\})$	$Y_2(e)$	$\tilde{Y}_2(\mathcal{E}\setminus\{e\})$
5	$\tilde{X}_2(e)$	$X_3(\mathcal{E}\setminus\{e\})$	$\tilde{Y}_2(e)$	$Y_3(\mathcal{E}\setminus\{e\})$
6	$X_3(e)$	$\tilde{X}_3(\mathcal{E}\setminus\{e\})$	$Y_3(e)$	$\tilde{Y}_3(\mathcal{E}\setminus\{e\})$
7	$\tilde{X}_3(e)$		$\tilde{Y}_3(e)$	
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$2n-1$		$X_n(\mathcal{E}\setminus\{e\})$		$Y_n(\mathcal{E}\setminus\{e\})$
$2n$	$X_n(e)$	$\tilde{X}_n(\mathcal{E}\setminus\{e\})$	$Y_n(e)$	$\tilde{Y}_n(\mathcal{E}\setminus\{e\})$
$2n+1$	$\tilde{X}_n(e)$		$\tilde{Y}_n(e)$	

Since the rate tuple is achievable, there is a network error-correction code such that it can combat the adversarial errors no matter how errors are generated and propagated. That is, the recovered messages of this code are always correct. From then on, we call this code "the original code."

Now we use the original code to construct another network error-correction code to attain the same tuple $(R_k : k \in \mathcal{K})$ when the delay profile is $\mathbf{d}+\mathbf{d}_e$.

When there exists an additional delay on the edge e, $X_1(e)$ and $Y_1(e)$ appear one time-unit after other transmitted and received signals. That is, the additional delay $\mathbf{d}_e$ makes $X_1(e)$ and $Y_1(e)$ appear in time 2. Due to the casuality of coders, the transmitted signals $X_i(\mathcal{E})$ is available only after the appearance of $Y_1(\mathcal{E}), \cdots, Y_{i-1}(\mathcal{E})$ in the original code. Therefore, if we want to use the original code, the subsequent transmitted signal $X_2(\mathcal{E}\setminus\{e\})$ is only available at time 3. What's worse, since there is one unit delay in edge e, $X_2(e)$ is only available at time 4. See Table II for the occurrence time of $X_2(\mathcal{E})$.

Repetitively using the original code will complete Table II. The code transmits message only at the original rate tuple $(R_k : k \in \mathcal{K})$ in $2n$ times. However, half of the cells in the table remain blank, which means that the transmitter and receiver are far from being fully used. In the sequel, we will make full use of these blank cells to deploy the original code again.

In Table III, two original codes are employed in the network with delay $\mathbf{d}+\mathbf{d}_e$. The first code, whose inputs and outputs are denoted by $X_i(\mathcal{E})$ and $Y_i(\mathcal{E})$ respectively, is identical to the employment of the code in Table II. Besides, the second code, whose inputs and outputs are denoted by $\tilde{X}_i(\mathcal{E})$ and $\tilde{Y}_i(\mathcal{E})$ respectively, is the one-unit delay version of the code in Table II. In this way, two codes together transmit two copies of messages in $(2n+1)$ channel uses. Considering that $\frac{2n}{2n+1} \to 1$

when n goes to infinity, the original rate tuple can be achievable asymptotically.

In this way, we construct a network error-correction code that asymptotically attains the rate $(R_k : k \in \mathcal{K})$ with delay $\mathbf{d} + \mathbf{d}_e$. Then the lemma is easily proved by taking the closure of all achievable rate tuples. ■

Proof of Theorem: On the one hand, Lemma 1 shows that $\mathscr{R}(\mathcal{N}(\mathbf{d})) \subseteq \mathscr{R}(\mathcal{N}(\mathbf{0}))$. On the other hand, we will show that $\mathscr{R}(\mathcal{N}(\mathbf{0})) \subseteq \mathscr{R}(\mathcal{N}(\mathbf{d}))$ in the sequel. Apply Lemma 2 $\prod_{e\in\mathcal{E}} \lceil d_e \rceil$ times, which results in

$$\mathscr{R}\left(\mathcal{N}(\mathbf{0})\right) \subseteq \mathscr{R}\left(\mathcal{N}\left(\lceil \mathbf{d} \rceil\right)\right).$$

Additionally, Lemma 1 leads to

$$\mathscr{R}\left(\mathcal{N}\left(\lceil \mathbf{d} \rceil\right)\right) \subseteq \mathscr{R}\left(\mathcal{N}(\mathbf{d})\right).$$

Therefore, $\mathscr{R}(\mathcal{N}(\mathbf{0})) \subseteq \mathscr{R}(\mathcal{N}(\mathbf{d}))$. Consequently, the proof is complete. ■

IV. Discussions of Variants

In this section, we discuss the impact of delays in the statistical variants of our model. Two assumptions in Section II-A are changed here:

1) Vanishing Error Probability: Section II-A requires network error-correction code to inerrably recover the message. By introducing the definition of error probability of a code, this assumption can be slightly relaxed by allowing a small value of error probability.

Definition of Error Probability: Consider a $(n, \{2^{nR_k} : k \in \mathcal{K}\})$ network error-correction code. For each connection $k \in \mathcal{K}$, the message M_k is independently and uniformly distributed in the message set $\{1, \cdots, 2^{nR_k}\}$. For any $a \in \mathcal{A}$, define Υ_a as the number of different of messages $(m_k : k \in \mathcal{K}) \in \prod_{k\in\mathcal{K}} \{1, \cdots, 2^{nR_k}\}$ such that when the adversary controls the combination a, the recovered messages are not always $(m_k : k \in \mathcal{K})$ when the source messages are $(m_k : k \in \mathcal{K})$. For $a \in \mathcal{A}$, define $P_e(a)$ as the error probability of a code when the adversary controls a:

$$P_e(a) = \frac{\Upsilon_a}{2^{n\sum_{k\in\mathcal{K}} R_k}}.$$

Then the error probability of a code is defined as

$$P_e = \max_{a\in\mathcal{A}} P_e(a).$$

With the definition of error probability, we can modify the definition of "achievable rate tuple": A rate tuple $(R_k : k \in \mathcal{K})$ is achievable if for any positive real number $\epsilon > 0$, there exists a $(n, \{2^{nR_k} : k \in \mathcal{K}\})$ network error-correction code such that its error probability $P_e < \epsilon$.

2) Noisy Edge Transmissions: Section II-A requires $Y_i(e = X_i(e)$ when edge e is not controlled by the adversary. This assumption can be relaxed, too. Consider the following edge transmission model: if an edge e is not controlled by the adversary, $Y_i(e)$ and $X_i(e)$ follow a memoryless conditional probability distribution $p_e(y|x)$; otherwise, $Y_i(e)$ is arbitrarily determined by the adversary.

In fact, the aforementioned two modifications do not affect the proof in Section III. Reprising the proof will lead to similar results in these variants, which is omitted here.

V. Conclusion

This paper investigates the effect of processing delays on network error-correction model. The result shows that the processing delays in the nodes do not affect the network error-correction capacity region at all.

Acknowledgment

The authors would like to thank Shenghao Yang, who made a significant contribution to this paper.

References

[1] N. Cai and R. Yeung, "Network error correction, Part II: lower bounds," *Commun. Inf. Syst.*, vol. 6, no. 1, pp. 37–54, 2006.
[2] S. Yang, C. Ngai, and R. Yeung, "Construction of linear network codes that achieve a refined singleton bound," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2007, pp. 1576–1580.
[3] S. Kim, T. Ho, M. Effros, and A. Avestimehr, "Network error correction with unequal link capacities," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 1144–1164, Feb. 2011.
[4] O. Kosut, L. Tong, and D. Tse, "Polytope codes against adversaries in networks," *IEEE Trans. Inf. Theory*, vol. 60, no. 6, pp. 3308–3344, Jun. 2014.
[5] Z. Xiao, Y. Li, M. Zhao, X. Xu, and J. Wang, "Allocation of network error correction flow to combat Byzantine attacks," *IEEE Trans. Commun.*, vol. 63, no. 7, pp. 2605–2618, Jul. 2015.
[6] Z. Xiao, J. Chen, Y. Li, and J. Wang, "Distributed multilevel diversity coding," *IEEE Trans. Inf. Theory*, vol. 61, no. 11, pp. 6368–6384, Nov. 2015.
[7] M. Effros, "On dependence and delay: Capacity bounds for wireless networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2012, pp. 550–554.
[8] R. Koetter, M. Effros, and M. Medard, "A theory of network equivalence—Part II: Multiterminal channels," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 3709–3732, Jul. 2014.
[9] R. Koetter, M. Effros, and M. Medard, "A theory of network equivalence—Part I: Point-to-point channels," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 972–995, Feb. 2011.