Just FUN: A Joint Fountain Coding and Network Coding Approach to Loss-Tolerant Information Spreading

Qiuyuan Huang, Kairan Sun, Xin Li, Dapeng Wu
Department of Electrical & Computer Engineering, University of Florida, USA

***Abstract*—To address the problem of information spreading over lossy communication channels, this paper proposes a joint FoUntain coding and Network coding (FUN) approach. Different from the Transmission Control Protocol (TCP), our FUN approach is a mechanism of Forward Error Correction (FEC), which does not use retransmission for recovery of lost packets. The novelty of our FUN approach lies in combining the best features of fountain coding, intra-session network coding, and cross-next-hop network coding. As such, our FUN approach is capable of achieving unprecedented high throughput over lossy channels. Experimental results demonstrate that our FUN approach achieves higher throughput than the existing schemes for multihop wireless networks.**

***Index Terms*—Fountain code, network coding, erasure channel, multihop, multiple sources, loss tolerant, information spreading**

I. Introduction

Information spreading plays a central role in human society. In the information age, how to efficiently and reliably spread information with low delay is critical for numerous activities involving humans and machines, e.g., the spread of tweets in Twitter, the dissemination of data collected by wireless sensor networks, and delivery of Internet TV. This paper is concerned with the problem of information spreading over lossy communication channels (a.k.a., erasure channels) where packets may be lost/discarded due to bit errors or buffer overflow. For wired networks over optical fiber channels, the bit error rate can be as low as 10^{-12}; hence packet loss is mainly due to overflow of the buffers at routers rather than bit errors. For wireless channels, packet loss is mainly due to uncorrectable bit errors[1], which are caused by fading, shadowing, interference, path loss, noise, etc. [1]

To address the packet loss problem, this paper proposes a joint FoUntain coding and Network coding (FUN) approach, which belongs to the category of FEC. Under our FUN coding approach, each source node uses a *fountain code* to encode information packets (which we call as native packets); each intermediate node (or a relay node) uses *intra-session network coding* to re-encode the packets in the same batch of the same session[2] received from the upstream node, and, if possible, uses *cross-next-hop network coding* to re-encode packets destined to different next-hop nodes; a sink node decodes the coded packets on the fly, and is able to reconstruct all the native packets as long as it receives sufficient number of coded packets.

The main contributions of this paper are:

1) We propose an FUN approach, which consists of three coding components, namely, fountain coding, intra-session network coding, and cross-next-hop network coding. The novelty of FUN lies in combining the best features of fountain coding, intra-session network coding, and cross-next-hop network coding. As such, our approach is capable of achieving unprecedented high throughput while allowing uncoordinated multiple source transmission of the same file to the same destination. Hence, FUN is well suited for peer-to-peer content delivery network (CDN), file transfer from distributed storage networks, social networks, social TV, and mobile TV.
2) We develop practical protocols for the proposed FUN approach, which are backward compatible with the existing protocols. Therefore, our protocols can seamlessly work with existing Medium Access Control (MAC) protocols, IP, TCP, UDP, HTTP, etc. In addition, the proposed FUN architecture provides a unified framework for existing network coding schemes, our proposed FUN codes, and existing protocols.

To evaluate the performance of FUN, we conduct extensive experiments. Experimental results demonstrate that our approach achieves higher throughput than existing schemes for multihop wireless networks.

The rest of the paper is organized as follows. Section II reviews the related work. In Section III, we present the overview of the FUN architecture. Section IV describes the FUN coding schemes in detail. In Section V, we show the experimental results. Section VI concludes the paper and points out the future directions.

[1]At the transmitter, most physical layer schemes encode messages by both an error-detecting code such as cyclic redundancy check (CRC) and an error-correction code. At the receiver, a received packet is first decoded by the error-correction decoder. If the resulting packet has uncorrectable bit errors, it will not pass the check of the error-detecting module. Most physical layer designs, if not all, will drop those packets that have uncorrectable bit errors.

[2]In this paper, a unicast session is identified by a unique source/destination IP address pair while a multicast session is identified by a tuple of the source IP address and all the multicast receiver IP addresses. In our future work, we will consider a session identified by a unique tuple of source/destination IP addresses, source/destination Layer-4 port numbers.

II. Related Work

In this section, we review various FEC mechanisms for erasure channels, i.e., erasure codes, network coding, and joint erasure coding and intra-session network coding.

A. Erasure Codes

Erasure codes can be used to recover native packets without feedback and retransmission. Erasure codes include Reed-Solomon codes, LDPC codes, and fountain codes [2]. Compared to erasure codes (including fountain codes), our proposed FUN approach can achieve much higher throughput for communication over multihop wireless networks. The lower bound on the end-to-end packet loss rate under FUN is $\max_{i\in\{1,2,\cdots,N_h\}} p_i$ (where p_i is the packet loss rate of Link i and N_h is the number of hops from the source to the destination) while the end-to-end packet loss rate under an erasure code is $1-\prod_{i=1}^{N_h}(1-p_i)$ [3], which is much larger than $\max_{i\in\{1,2,\cdots,N_h\}} p_i$ for large N_h. For example, for $N_h = 2$ and $p_i = 0.1$ ($i = 1, 2$), the end-to-end packet loss rate under FUN is 0.1 while the end-to-end packet loss rate under an erasure code is 0.19; for $N_h = 10$ and $p_i = 0.1$ ($\forall i$), the end-to-end packet loss rate under FUN is still 0.1 while the end-to-end packet loss rate under an erasure code is 0.65, which is 6.5 times as much as 0.1. The much lower end-to-end packet loss rate achieved by FUN translates to much higher throughput (date rate), compared to erasure codes. In our simulations, we observe that the throughput under FUN is 35 times as large as that under a fountain code (RQ code), for $N_h = 10$ and $p_i = 0.1$ ($\forall i$).

The reason why FUN achieves higher throughput over multihop lossy networks than erasure codes, is because under FUN, each relay node performs network coding and hence coded packets that are lost at each hop are regenerated/re-coded for the next hop. To illustrate how FUN works, here is an analogy: a person carries a leaky tank of water from the source node to the destination; in each hop, the tank leaks p percent of water; at each relay node, the tank gets refilled to its full capacity; finally, the tank only lost p percent of water from the source node to the destination since only the lost water in the last hop is not refilled. In contrast, erasure codes (including fountain codes) are like no refill at any relay node since network coding is not used at any relay node; hence the tank loses $(1-(1-p/100)^{N_h})\times 100$ percent of water from the source node to the destination. Here, $p/100$ is the percentage, e.g., $p = 10$ gives $p/100 = 10\%$.

B. Network Coding

Simply forwarding packets is not an optimal operation at a router from the perspective of maximizing throughput. Network coding was proposed to achieve maximum throughput for multicast communication [4]. Network coding techniques can be classified into two categories: intra-session (where coding is restricted to the same multicast or unicast session) [4]–[6] and inter-session (where coding is applied to packets of different sessions) [7]–[9]. For wireless communication, cross-next-hop network coding [8], [9] and intra-session network coding [3], [10], [11] are usually used. Under cross-next-hop network coding, a relay node applies coding to packets destined to different next-hop nodes. Cross-next-hop network coding is a special type of inter-session network coding.

Cross-next-hop network coding has been heavily studied in the wireless networking area. The major works include [8], [9]. In [8], Katti et al. proposed an opportunistic network coding scheme for unicast flows, called COPE, which can achieve throughput gains from a few percent to several folds depending on the traffic pattern, congestion level, and transport protocol. In [9], Rayanchu el al. developed a loss-aware network coding technique for unicast flows, called CLONE, which improves reliability of network coding by transmitting multiple copies of the same packet, similar to repetition coding [12].

Different from COPE [8], which does not add redundancy to the coded packets, our FUN approach adds redundancy to the received packets at a relay node; specifically, under FUN, a relay node re-codes the received packets, using random linear code [3], [11]. Different from CLONE [9], which uses repetition coding, our FUN approach uses random linear coding, which is more efficient than repetition coding. Our experimental results show that FUN achieves much higher throughput over lossy channels, compared to COPE.

C. Joint Erasure Coding and Intra-Session Network Coding

The following Joint Erasure coding and intra-session Network coding (JEN) approach has been used for unicast/multicast communication in [10]: The source node uses *random linear erasure coding* (RLEC) to encode the native packets and add a global encoding vector to the header of each coded packet. A relay node uses *random linear network coding* (RLNC) to re-code the packets it has received, i.e., the relay node generates a coded packet by randomly linearly combining the packets that it has received and stored in its buffer; the relay node also computes the global encoding vector of the re-coded packet, and add the global encoding vector to the header of the re-coded packet. A destination node can decode and recover K native packets as long as it receives enough coded packets that contain K linearly independent global encoding vectors. In practice, under JEN, the data to be transmitted is partitioned into multiple segments [13], or generations [14], or blocks [15], or batches [16], and coding is restricted within the same segment/generation/block/batch. In doing so, the encoding vector is small enough to be put into the header of a coded packet. To combine the best features of JEN and fountain codes, Yang and Yeung proposed BATched Sparse (BATS) codes [3], [11]. Our FUN approach combines the best features of fountain coding, intra-session network coding, and cross-next-hop network coding. Different from BATS codes, FUN utilizes cross-next-hop network coding and hence achieves a higher rate.

III. FUN Overview

In this section, we introduce FUN, a new forwarding architecture for wireless multihop networks. Since a wireless channel is a shared medium, it can be regarded as a broadcast channel, i.e., a transmitted packet can be overheard by all the nodes within the transmission range of the sender of the packet.

We consider a pair of nodes, say Node A and Node B. Assume that there are two unicast flows between the two nodes, i.e., a forward flow from Node A to Node B and a backward flow from Node B to Node A. We propose two coding schemes, i.e., FUN-1 and FUN-2:

- FUN-1 basically combines BATS coding [3] with COPE [8] for two flows. But FUN-1 is not a simple combination of BATS and COPE; a relay node needs to recover BATS-encoded packets of the forward flow before recovering packets of the backward flow.
- FUN-2 combines BATS coding with RLNC for two flows; each relay node needs to add a new encoding vector to the header of a re-coded packet; only the destination node performs decoding.

Under FUN-1, two sub-layers, i.e., Layer 2.1 and Layer 2.2, are inserted between Layer 2 (MAC) and Layer 3 (IP). Layer 2.1 is for cross-next-hop network coding, similar to the functionality of COPE [8]. Layer 2.2 is for BATS coding [3]. At a source node, Layer 2.2 uses a fountain code to encode all native packets from upper layers (similar to the outer code in a BATS code); there is no Layer 2.1 at a source node. At a relay node, Layer 2.1 is used for cross-next-hop network coding and Layer 2.2 is used for intra-session network coding (similar to the inner code in a BATS code); for Layer 2.2, the relay node runs a procedure called *FUN-1-2.2-Proc*, which performs RLNC within the same batch. At a destination node, Layer 2.2 decodes the coded packets received; there is no Layer 2.1 at a destination node.

Under FUN-2, only one sub-layer, i.e., Layer 2.2, is inserted between Layer 2 (MAC) and Layer 3 (IP). At a source node, Layer 2.2 uses a fountain code to encode all native packets from upper layers (similar to the outer code in a BATS code). At a relay node, if Layer 2.2 receives a packet with FUN-2 switch enabled, it will run a procedure called *FUN-2-2.2-Proc* for mixing packets from two flows; otherwise, it will run the procedure *FUN-1-2.2-Proc*, which does not mix packets from two different flows. Note that different from a BATS code, *FUN-2-2.2-Proc* performs re-coding of batches from two different flows. At a destination node, Layer 2.2 decodes the coded packets received.

Different from COPE, FUN-2 is an end-to-end solution, i.e., a re-coded packet is never decoded at a relay node. In contrast, under COPE, a next-hop node, say, Node A, needs to recover the native packet, whose next-hop node is Node A; the recovery process is like decoding; the relay node may not recover the native packet if the relay node does not have enough known packets that are mixed in the XOR-ed packet.

In the current version, both FUN-1 and FUN-2 are restricted to two flows, i.e., forward flow and backward flow between two nodes. The advantage is that there is no need for co-ordination while a higher coding gain can be achieved. The limitation is that it restricts its use to two flows between two nodes.

Both FUN-1 packet and FUN-2 packet have two headers as shown in Fig. 1. If a re-coded packet is mixed from two flows (i.e., forward and backward flows), it will have a non-empty Header 2; otherwise, there will be no Header 2.

Header 1 and Header 2 have the same structure for FUN-1 and FUN-2. Fig. 2 shows the structure for FUN-1 Header 1 and Header 2. Fig. 3 shows the structure for FUN-2 Header 1 and Header 2. In Figs. 2 and 3, the NC switch consists of two bits and indicates one of the following four schemes is used: 1) FUN-1, 2) FUN-2, 3) RLNC, 4) no network coding. COPE is a special case of FUN-1, where there is no encoding vector in FUN Headers; in other words, if the NC switch equals 00 (in binary format) and there is no encoding vector in FUN Headers, then the packet is a COPE packet. BATS is a special case of FUN-2, where there is no FUN Header 2. The fountain code corresponds to the no-network-coding case with the NC switch equal to 11 (in binary format) and no encoding vectors in FUN header and no Header 2.

In fact, our FUN architecture is extensible to accommodate more than two flows and more than two FUN headers.

IV. Description of FUN coding

A. FUN-1

At a source node, Layer 2.2 uses a RaptorQ (RQ) code [17] to encode all native packets from Layer 3. The RQ code is the most advanced fountain code that is available commercially.

Assume Node A will transmit K native packets to Node B, and Node B will transmit K native packets to Node A[3]. Each packet has T symbols in a finite field $\mathbb{F}_q$, where q is the size of the field. Denote a packet by a column vector in $\mathbb{F}_q^T$. In the rest of the paper, we denote the set of K native packets by the following matrix

$$\mathbf{B} = [b_1, b_2, \cdots, b_K], \tag{1}$$

where b_i is the i-th native packet. With an abuse of notation, when treating packets as elements of a set, we write $b_i \in \mathbf{B}$, $\mathbf{B}' \in \mathbf{B}$, etc.

Next we describe the outer code, inner code, XOR coding, and precoding of FUN-1.

1) Outer Code of FUN-1: The outer code of FUN-1 is the same as the outer code of a BATS code. The outer coding of FUN-1 is performed at a source node at Layer 2.2.

At a source node, each coded batch has M coded packets. The i-th batch $\mathbf{X}_i$ is generated from a subset $\mathbf{B}_i \subset \mathbf{B}$ ($\mathbf{B} \in \mathbb{F}_q^{T\times K}$) by the following operation

$$\mathbf{X}_i = \mathbf{B}_i \mathbf{G}_i \tag{2}$$

where $\mathbf{G}_i \in \mathbb{F}_q^{d_i \times M}$ is called the generator matrix of the i-th batch; $\mathbf{B}_i \in \mathbb{F}_q^{T\times d_i}$; $\mathbf{X}_i \in \mathbb{F}_q^{T\times M}$. Similar to a fountain code, matrix $\mathbf{B}_i$ is randomly formed by two steps: 1) sample a given degree distribution $\Psi = (\Psi_0, \Psi_1, \cdots, \Psi_K)$ and obtain a degree d_i with probability Ψ_{d_i}; 2) uniformly randomly choose d_i packets from $\mathbf{B}$ to form $\mathbf{B}_i$. Matrix $\mathbf{G}_i$ is randomly

[3]For simplicity of notation, we let the two flows have the same number K of packets to transmit. Actually, this is not required for the FUN architecture.

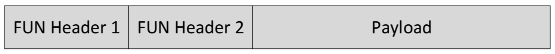

Fig. 1. The structure of a FUN-1/FUN-2 packet

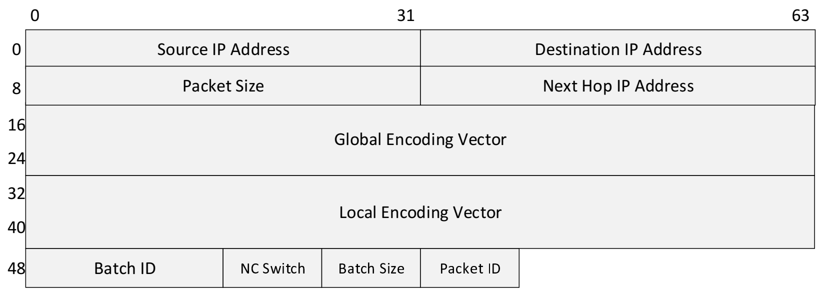

Fig. 2. The header structure of a FUN-1 packet

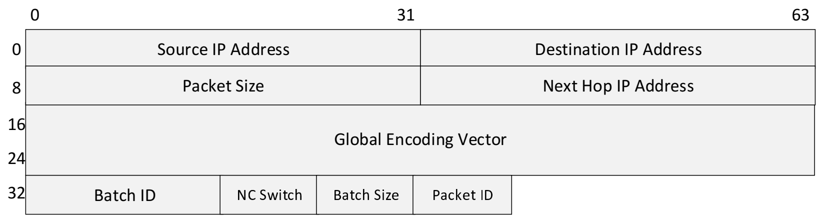

Fig. 3. The header structure of a FUN-2 packet

generated; specifically, all the entries in $\mathbf{G}_i$ are independent, identically distributed with a uniform distribution in $\mathbb{F}_q$. In our implementation, $\mathbf{G}_i$ is generated by a pseudorandom number generator and can be recovered at the destinations using the same pseudorandom number generator with the same seed.

2) Inner Code of FUN-1: At a relay node, Layer 2.2 performs inner coding of FUN-1, which is the same as that for a BATS code. We first consider the first down-stream relay node, say, Node R_1. Assume $\mathbf{X}'_{i,1}$ are the set of packets of the i-th batch correctly received by Node R_1, transmitted by the source. Since there may be lost packets from the source to Node R_1, we have $\mathbf{X}'_{i,1} \subseteq \mathbf{X}_i$. We write

$$\mathbf{X}'_{i,1} = \mathbf{X}_i \mathbf{E}_{i,1} \tag{3}$$

where $\mathbf{E}_{i,1}$ is an erasure matrix, representing the erasure channel between the source and Node R_1. $\mathbf{E}_{i,1}$ is an $M \times M$ diagonal matrix whose entry is one if the corresponding packet in $\mathbf{X}_i$ is correctly received by Node R_1, and is zero otherwise. Hence, matrix $\mathbf{X}'_{i,1} \in \mathbb{F}_q^{T\times M}$ has the same dimensions as $\mathbf{X}_i$. Here, with an abuse of the notation $\mathbf{X}'_{i,1}$, we replace each lost packet in $\mathbf{X}_i$ by a column vector whose entries are all zero, resulting in matrix $\mathbf{X}'_{i,1}$.

At Node R_1, the inner coding of FUN-1 is performed by

$$\mathbf{Y}_{i,1} = \mathbf{X}'_{i,1}\mathbf{H}_{i,1} = \mathbf{X}_i\mathbf{E}_{i,1}\mathbf{H}_{i,1} = \mathbf{B}_i\mathbf{G}_i\mathbf{E}_{i,1}\mathbf{H}_{i,1}, \tag{4}$$

where $\mathbf{H}_{i,1} \in \mathbb{F}^{M\times M}$ is the transfer matrix of an RLNC for the i-th batch at Node R_1. After inner-encoding, each column of the product matrix $\mathbf{E}_{i,1}\mathbf{H}_{i,1}$ is added to the header of the corresponding coded packet as a global encoding vector, which is needed by the destination node for decoding.

At the relay node of the j-th hop, denoted as Node R_j, the following re-coding is performed

$$\begin{aligned} \mathbf{Y}_{i,j} &= \mathbf{X}'_{i,j}\mathbf{H}_{i,j} = \mathbf{Y}_{i,j-1}\mathbf{E}_{i,j}\mathbf{H}_{i,j} \\ &= \mathbf{B}_i\mathbf{G}_i\mathbf{E}_{i,1}\mathbf{H}_{i,1}\cdots\mathbf{E}_{i,j}\mathbf{H}_{i,j}, \end{aligned} \tag{5}$$

where $\mathbf{E}_{i,j}$ is an erasure matrix of the i-th batch for the erasure channel from Node R_{j-1} to Node R_j; $\mathbf{H}_{i,j} \in \mathbb{F}_q^{M\times M}$ is the transfer matrix of an RLNC for the i-th batch at Node R_j. After inner-encoding, each column of the product matrix $\mathbf{E}_{i,1}\mathbf{H}_{i,1}\cdots\mathbf{E}_{i,j}\mathbf{H}_{i,j}$ is used to update the global encoding vector of the corresponding coded packet.

The above inner-coding procedure is implemented in software module *FUN-1-2.2-Proc* mentioned in Section III.

3) XOR Coding of FUN-1: The XOR coding of FUN-1 is similar to COPE [8]. At Node R_i w.r.t. the forward flow from Node A to Node B, the XOR encoding procedure is shown in Algorithm 1. The XOR decoding procedure is shown in Algorithm 2.

4) Precoding of FUN-1: At a source node, precoding is performed, similar to Raptor codes. The precoding can be achieved by a traditional erasure code such as LDPC and Reed-Solomon code. The precoding of FUN-1 is performed at a source node at Layer 2.2. After precoding, the resulting packets is further encoded by the outer encoder described in Section IV-A1.

B. FUN-2

FUN-2 consists of outer code, inner code, and precoding, which are described as below.

Algorithm 1 XOR encoding of FUN-1

```
1: while at least one flow (forward or backward flow) is alive do
2:   if Layer 2.2 output queues for the forward flow and the backward flow both have at least one batch of $M$ re-coded packets, say, the $i$-th batch for the forward flow and the $j$-th batch for the backward flow then
3:     for $m = 1, \cdots, M$ do
4:       Pick Packet $y_{i,m}$ at the head of Layer 2.2 output queue for the forward flow;
5:       Pick packet $\bar{y}_{j,m}$ at the head of Layer 2.2 output queue for the backward flow;
6:       $p_m = y_{i,m} \oplus \bar{y}_{j,m}$;
7:       Put the following in the header of packet $p_m$: 1) packet ID $m$, 2) the MAC address of the next-hop node of Packet $y_{i,m}$, 3) batch ID $i$ of Packet $y_{i,m}$, 4) the MAC address of the next-hop node of packet $\bar{y}_{j,m}$, 5) batch ID $j$ of packet $\bar{y}_{j,m}$, 6) local encoding vectors of packets $y_{i,m}$ and $\bar{y}_{j,m}$;
8:       Enable the bit $FUN\_XOR$ in the header of packet $p_m$, i.e., $FUN\_XOR = 1$;
9:       Place packet $p_m$ in Layer 2.1 output queue;
10:    end for
11:  else
12:    if Layer 2.2 output queue of one flow (forward or backward) has at least two batch of $M$ re-coded packets, say, the $i$-th batch being the head-of-line batch then
13:      for $m = 1, \cdots, M$ do
14:        Pick Packet $y_{i,m}$ at the head of Layer 2.2 output queue of the flow;
15:        Disable the bit $FUN\_XOR$ in the header of packet $y_{i,m}$, i.e., $FUN\_XOR = 0$;
16:        Place packet $y_{i,m}$ in Layer 2.1 output queue;
17:      end for
18:    end if
19:  end if
20: end while
```

1) Outer Code of FUN-2: The outer code of FUN-2 is the same as the outer code of FUN-1, except the decoding process. In the decoding process, the destination node of the forward flow is also a source node of the backward flow. So this destination node can use its known packets of the backward flow to decode the coded packets of the forward flow. To limit the size of the encoding vector in the packet header, at a relay node, FUN-2 only allows the mixing of two batches from two flows once; i.e., if a packet is already a mixture of two packets from two flows, it will not be re-coded again at a relay node. The FUN-2 outer decoding procedure is shown in Algorithm 3. For simplicity, we denote the two nodes as Node 0 and 1. The equation in Step 17 can be proved as below. Since the inner coding is a mixture of two flows according to Eq. (8), we have

$$\mathbf{Y}_{i,j} = [\mathbf{B}_{i,j}, \mathbf{B}_{k,1-j}][\mathbf{H}_{i,j}, \mathbf{H}_{k,1-j}]^T \quad (6)$$

$$= \mathbf{B}_{i,j}\mathbf{H}_{i,j} + \mathbf{B}_{k,1-j}\mathbf{H}_{k,1-j} \quad (7)$$

Hence, $\mathbf{B}_{i,j}\mathbf{H}_{i,j} = \mathbf{Y}_{i,j} - \mathbf{B}_{k,1-j}\mathbf{H}_{k,1-j}$. Since $\mathbf{B}_{i,j}\mathbf{H}_{i,j}$ are the coded packets of the i-th batch and Destination j, we assign $\mathbf{B}_{i,j}\mathbf{H}_{i,j}$ to $\mathbf{Y}_{i,j}$. This proves the equation in Step 17.

2) Inner Code Encoding of FUN-2: The inner code of FUN-2 is similar to the inner code of FUN-1 in the sense that both of them use RLNC. The difference is that FUN-2 mixes the packets of two flows while FUN-1 does not. Specifically, under FUN-2, at the relay node of the j-th hop, denoted as Node R_j, the following re-coding is applied to two juxtaposed matrices of received packets:

$$\mathbf{Z}_{i,j} = [\mathbf{Z}_{i,j-1}\mathbf{E}_{i,j}, \mathbf{Z}_{k,j+1}\overline{\mathbf{E}}_{k,j}]\mathbf{H}_{i,j}, \quad (8)$$

where $\mathbf{Z}_{i,j} \in \mathbb{F}_q^{T\times M}$ contains M re-coded packets of the i-th batch, generated by Node R_j; $\overline{\mathbf{E}}_{k,j} \in \mathbb{F}_2^{M\times M}$ is an erasure matrix of the k-th batch for the erasure channel from Node R_{j+1} to Node R_j; $\mathbf{H}_{i,j} \in \mathbb{F}_q^{2M\times M}$ is the transfer matrix of an RLNC for the i-th batch of the forward flow and the k-th batch of the backward flow at Node R_j. The FUN-2 inner-coding procedure is shown in Algorithm 4, where Destination 0 and 1 denote the destination of the forward and backward flow, respectively; Packet $y_{i,m}$ has batch ID i and its position in the batch is m; a buffer being complete means that a buffer is full with M packets OR a newly arriving packet has a batch ID, which is larger than that of the packets in the buffer. After inner-encoding, each column of the matrix $\mathbf{H}_{i,j}$ is added to the global encoding vector of the corresponding coded packet. The inner-coding procedure is implemented in software module *FUN-2-2.2-Proc* mentioned in Section III.

3) Precoding of FUN-2: The precoding of FUN-2 is the same as the precoding of FUN-1.

V. Experimental Results

A. Development of Simulator

We implement our proposed FUN-1 and FUN-2 on QualNet [18]. For comparison, we also implement a BATS code [3], a fountain code (specifically, the RQ code [17]), RLNC [13], and COPE [8] in QualNet. For COPE, we only implement the XOR operation for mixing two flows; and Layer 4 in the COPE scheme is TCP; the reason why we use TCP for COPE is because each scheme needs to achieve perfect recovery of lost packets to make a fair comparison. In the future, we will

Algorithm 2 XOR decoding of FUN-1

```
1: while at least one flow (forward or backward flow) is alive do
2:   if the head-of-line packet in Layer 2.1 input queue has $FUN\_XOR = 0$ then
3:     Move this head-of-line packet to Layer 2.2 input queue;
4:   else
5:     Move this head-of-line packet to its corresponding queue in the buffer of Layer 2.1 (each queue is uniquely identified by the same set of receiver MAC addresses in the packet header);
6:     if a queue in the buffer of Layer 2.1 has two batches of received packets (assume the head-of-line XOR-ed batch has batch ID $i$ for the forward flow and batch ID $j$ for the backward flow) then
7:       while the head-of-line packet, say $p_m$, has its forward-flow batch ID equal $i$ do
8:         Recover the packet of the forward flow by $y_{i,m} = p_m \oplus \bar{y}_{j,m}$, where $\bar{y}_{j,m}$ is a packet in $\overline{\mathbf{Y}}_{j,l-1} = \overline{\mathbf{Y}}_{j,l}\overline{\mathbf{E}}_{j,l-1}\overline{\mathbf{H}}_{j,l-1}$, where $l$ is the index of the current node $R_l$, $l-1$ is the index of the next-hop node $R_{l-1}$ of the backward flow, $\overline{\mathbf{Y}}_{j,l-1}$ is the matrix whose columns are coded packets, re-coded by Node $R_{l-1}$ for the backward flow, $\overline{\mathbf{E}}_{j,l-1}$ is an erasure matrix for the erasure channel from Node $R_l$ to Node $R_{l-1}$, $\overline{\mathbf{E}}_{j,l-1}$ can be obtained by Node $R_l$ via the global encoding vector in FUN-1 packet header, $\overline{\mathbf{H}}_{j,l-1}$ is the transfer matrix of an RLNC for the $j$-th batch at Node $R_{l-1}$ for the backward flow, $\overline{\mathbf{H}}_{j,l-1}$ can be obtained by the local encoding vectors in FUN-1 packet header;
9:         Recover the packet of the backward flow by $\bar{y}_{i,m} = p_m \oplus y_{j,m}$, where $y_{j,m}$ is a packet in $\mathbf{Y}_{j,l+1} = \mathbf{Y}_{j,l}\mathbf{E}_{j,l+1}\mathbf{H}_{j,l+1}$, where $l$ is the index of the current node $R_l$, $l+1$ is the index of the next-hop node $R_{l+1}$ of the forward flow, $\mathbf{Y}_{j,l+1}$ is the matrix whose columns are coded packets, re-coded by Node $R_{l+1}$ for the forward flow, $\mathbf{E}_{j,l+1}$ is an erasure matrix for the erasure channel from Node $R_l$ to Node $R_{l+1}$, $\mathbf{E}_{j,l+1}$ can be obtained by Node $R_l$ via the global encoding vector in FUN-1 packet header, $\mathbf{H}_{j,l+1}$ is the transfer matrix of an RLNC for the $j$-th batch at Node $R_{l+1}$ for the forward flow, $\mathbf{H}_{j,l+1}$ can be obtained by the local encoding vectors in FUN-1 packet header;
10:      end while
11:    end if
12:  end if
13: end while
```

implement COPE with UDP and hop-by-hop retransmission for packet recovery as in Ref. [8].

For RLNC, a file is segmented into batches, each of which consists of M native packets. Each batch is transmitted independently as if it is a single file; there is no coding across two batches. A source node keeps transmitting coded packets of a batch until the source node receives an ACK message from the destination node. A relay node has a buffer of M packets; when a relay node receives a packet from its upstream node, it places the packet in the buffer; if the buffer is full, the newly arriving packet will push out the oldest packet; then the relay node takes all the packets in the buffer as input and generates one RLNC-coded packet, which is then sent out to its downstream node. When a destination node decodes all the native packets in a batch, the destination node transmits an ACK message toward the source node. Upon receiving the ACK message, the source node stops transmitting the coded packets of the current batch, and starts to transmit the coded packets of the next batch.

We use IEEE 802.11b for the physical layer and MAC layer of each wireless node, and use the Ad hoc On-Demand Distance Vector (AODV) protocol for routing. For COPE, we use TCP as the Layer 4 protocol; for FUN-1, FUN-2, BATS, fountain code, and RLNC, we use UDP as the Layer 4 protocol. All the experiments have the following setting: the packet size $T = 1024$ bytes; the batch size $M = 16$ packets.

B. Performance Evaluation

We conduct experiments for the following four cases: 1) two hops with no node mobility (fixed topology) under various packet loss rate per hop, 2) various number of hops with no node mobility (fixed topology) under fixed packet loss rate per hop, 3) a large number nodes with node mobility (dynamic topology). There are two flows (forward and backward flows) between each source/destination pair. For each case, we compare the performance of seven schemes, i.e., FUN-1, FUN-2, a BATS code, a fountain code, RLNC, COPE, and TCP.

As we know, the lower the packet sending rate of UDP, the lower throughput. But too high packet sending rate of UDP will incur congestion and packet loss. Hence, in the experiments of FUN-1, FUN-2, BATS, fountain code, and RLNC, which use UDP as their Layer 4 protocol, we tune the packet sending rate of UDP until we find the maximum throughput for each of these five schemes. At the optimal packet sending rate of UDP, we conduct ten experiments for each of these five schemes, and take the average throughput of the ten experiments as the performance measure of each scheme.

For COPE and TCP, we conduct ten experiments for each of these two schemes, and take the average throughput of the ten experiments as the performance measure of COPE and TCP.

Algorithm 3 Outer decoding of FUN-2 at Node j ($j \in \{0, 1\}$)

```
1:  while not all native packets destined to Node $j$ are decoded do
2:    if the receiving queue of Layer 2.2 is complete then
3:      Pick packets of the same batch from the head of the receiving queue of Layer 2.2;
4:      if FUN-2 Header 2 is not empty then
5:        if the destination IP address in FUN-2 Header 1 is that of Node $j$ then
6:          Let $i$ equal the batch ID in FUN-2 Header 1 of any picked packet;
7:          Let $k$ equal the batch ID in FUN-2 Header 2 of any picked packet;
8:          Let $\mathbf{H}_{k,1-j}$ be a matrix whose columns are the global encoding vectors in FUN-2 Header 2 of all the picked
            packets;
9:          Let $\mathbf{B}_{k,1-j}$ be a matrix whose columns are native packets of the $k$-th batch, sent from Node $j$ to Node $1-j$;
10:       else
11:         Let $i$ equal the batch ID in FUN-2 Header 2 of any picked packet;
12:         Let $k$ equal the batch ID in FUN-2 Header 1 of any picked packet;
13:         Let $\mathbf{H}_{k,1-j}$ be a matrix whose columns are the global encoding vectors in FUN-2 Header 1 of all the picked
            packets;
14:         Let $\mathbf{B}_{k,1-j}$ be a matrix whose columns are native packets of the $k$-th batch, sent from Node $j$ to Node $1-j$;
15:       end if
16:       Let $\mathbf{Y}_{i,j}$ be a matrix whose columns are the payloads of all the picked packets;
17:       Compute $\mathbf{Y}_{i,j} = \mathbf{Y}_{i,j} - \mathbf{B}_{k,1-j}\mathbf{H}_{k,1-j}$;
18:     else
19:       Let $i$ equal the batch ID in FUN-2 Header 1 of any picked packet;
20:       Let $\mathbf{Y}_{i,j}$ be a matrix whose columns are the payloads of all the picked packets;
21:     end if
22:     Do outer decoding in the same way as BATS outer decoding with input packets $\mathbf{Y}_{i,j}$;
23:   end if
24: end while
```

1) Case 1: Two Hops with No Node Mobility: The setup of this set of experiments is the following. There are three nodes in the network: a source node, a destination node, and one relay node. The communication path from the source node to the destination node has two hops. All the three nodes are immobile; hence the network topology is fixed.

Denote K the number of native packets to be transmitted by the source. In the experiments, we measure the throughput in Mbits/s under different values of K and different packet loss rate (PLR). The PLR is the same for all the links/hops in the network. Here, the PLR does not include packet loss due to the thermal noise in the physical layer and packet collision, which is out of our direct control; here the PLR is achieved by randomly dropping a correctly received packet at Layer 2 with a probability equal to the given PLR.

Table I shows the total throughput of the two flows (i.e., forward/backward flows) of the seven schemes under Case 1. We have the following observations:

- For both the lossless and lossy situations, FUN-1 and FUN-2 achieve the highest throughput. This is because FUN-1 and FUN-2 combine the best features of fountain coding, intra-session network coding, and cross-next-hop network coding.
- The throughput of FUN-2 is higher than or equal to that of FUN-1. This is because FUN-2 uses RLNC at a relay node and RLNC has a higher coding gain than the XOR operation used in FUN-1.
- For both the lossless and lossy situations, the fountain code achieves a higher throughput than the BATS code. This is because a BATS code only achieves higher throughput than a fountain code when there are more than two hops or the PLR is high (say, 20%).
- For the lossless situation, COPE achieves a higher throughput than the BATS and the fountain code for $K = 1600$ but achieves a lower throughput than the BATS and the fountain code for $K = 6400$ and 16000. This may be due to the fact that BATS codes and fountain codes are erasure channel coding (while COPE is not) and hence, the more native packets to transmit, the higher coding gain.
- For both the lossless and lossy situations, RLNC achieves a lower throughput than the fountain code and the BATS code. This is because a coded packet in RLNC is restricted to one batch of M native packets while a coded packet in the BATS code and the fountain code is a random mixture of all the K native packets; hence each native packet has a less chance of being recovered in RLNC for the same number of coded packets, compared to the BATS code and the fountain code.
- RLNC achieves a lower throughput than COPE in the lossless situation, but achieves a higher throughput than COPE in the lossy situation. This is because RLNC is

TABLE I
THROUGHPUT UNDER CASE 1

K	Scheme	Throughput (Mbits/s)	
		PLR = 0	PLR = 10%
1600	FUN-1	0.697	0.652
	FUN-2	0.697	0.668
	BATS	0.484	0.488
	Fountain	0.498	0.508
	RLNC	0.460	0.340
	COPE	0.520	N/A
	TCP	0.375	N/A
6400	FUN-1	0.720	0.665
	FUN-2	0.727	0.669
	BATS	0.517	0.502
	Fountain	0.533	0.513
	RLNC	0.460	0.340
	COPE	0.500	N/A
	TCP	0.378	N/A
16000	FUN-1	0.714	0.637
	FUN-2	0.714	0.655
	BATS	0.521	0.487
	Fountain	0.533	0.493
	RLNC	0.460	0.340
	COPE	0.504	N/A
	TCP	0.379	N/A

erasure channel coding: when there is no packet loss, the redundancy induced by RLNC reduces the throughput; when there is packet loss, the reliability induced by RLNC make it achieve a higher throughput.

- TCP achieves the least throughput for the lossless situation. This is because TCP has a slow start and a congestion avoidance phase, which reduces throughput. In contrast, COPE has network coding gain and FUN-1, FUN-2, the BATS code, the fountain code, and RLNC use UDP with optimal packet sending rate.
- For PLR=10%, COPE and TCP time out and could not receive all K number of packets due to high packet loss rate. This is consistent with the fact that TCP performs poorly under environments of high packet loss rates [19].

2) Case 2: Various Number of Hops with No Node Mobility: The setup of this set of experiments is the following. The network consists of a source node, a destination node, and 1 or 2 or 4 relay nodes. All the nodes in the network form a chain topology from the source node to the destination node. The communication path from the source node to the destination node has 2 or 3 or 5 hops. All the nodes are immobile; hence the network topology is fixed. For all the experiments in Case 2, we set PLR=10% for each hop/link. Again, the PLR does not include packet loss due to the thermal noise in the physical layer and packet collision.

Table II shows the throughput of seven schemes under Case 2. We have the following observations:

- For both the lossless and lossy situations, FUN-1 and FUN-2 achieve similar throughput and their throughput is the highest among the seven schemes.
- When the number of hops is two, the throughput of FUN-2 is not less than than FUN-1. This is because FUN-2 uses RLNC at a relay node and RLNC is better than the XOR operation used in FUN-1.
- When the number of hops is more than two, FUN-1 may achieve a higher throughput than FUN-2. This is because FUN-2 only allows mixing two flows once but FUN-1 allows mixing two flows unlimited number of times. Hence, FUN-1 potentially has a higher coding gain than FUN-2 due to more coding opportunities. However, this is not always true. Since FUN-1 always uses broadcast and FUN-2 has to use unicast when FUN-2 packet has already been mixed from two flows once, FUN-2 unicast packets are more reliably received than FUN-1 broadcast packets under 802.11 MAC. It is known that in the 802.11 unicast mode, packets are immediately acknowledged by their intended next-hop nodes; if no ACK message is received, the 802.11 MAC layer will retransmit the packet a number of times (with exponential backoff) until an ACK message is receiver or a time-out event happens. But a broadcast packet will not be acknowledged and retransmitted under 802.11.
- When the number of hops is two, the fountain code achieves a higher throughput than the BATS code; when the number of hops is more than two, the BATS code achieves a higher throughput than the fountain code.
- COPE and TCP time out and could not receive all K number of packets due to high packet loss rate. Hence, COPE and TCP achieve the least throughput.
- For all the situations in Case 2, RLNC achieves a lower throughput than the BATS code. This is because a coded packet in RLNC is restricted to one batch of M native packets while a coded packet in the BATS code is a random mixture of all the K native packets.
- When the number of hops is 2 and 3, RLNC achieves a lower throughput than the fountain code. This is because a coded packet in RLNC is restricted to one batch of M native packets while a coded packet in the fountain code is a random mixture of all the K native packets. But when the number of hops is 5, RLNC achieves a higher throughput than the fountain code. This is because the more relay nodes, the more opportunities for network coding in RLNC while the fountain code does not have such a benefit.
- For the case of $K = 6400$ and five hops, the fountain code does not receive all the K native packets within 6000 seconds, which we call “timeout”. The timeout is because the end-to-end packet loss is too high for five hops.

3) Case 3: A Large Number of Nodes with Node Mobility: The setup of this set of experiments is the following. There are 400 nodes in the network. All the nodes move under the random waypoint mobility model, i.e., each node selects a random position, moves towards it in a straight line at a constant speed that is randomly selected from a range, and pauses at that destination; each node repeats this process throughout the experiment. Due to node mobility, the communication routes

TABLE II
THROUGHPUT UNDER CASE 2

K	Scheme	Throughput (Mbits/s)		
		2 hops	3 hops	5 hops
1600	FUN-1	0.652	0.413	0.045
	FUN-2	0.652	0.364	0.042
	BATS	0.488	0.317	0.036
	Fountain	0.508	0.271	0.005
	RLNC	0.340	0.202	0.024
	COPE	N/A	N/A	N/A
	TCP	N/A	N/A	N/A
6400	FUN-1	0.665	0.376	0.026
	FUN-2	0.669	0.357	0.033
	BATS	0.502	0.327	0.025
	Fountain	0.513	0.220	N/A
	RLNC	0.340	0.202	0.024
	COPE	N/A	N/A	N/A
	TCP	N/A	N/A	N/A

TABLE III
THROUGHPUT UNDER CASE 3

Scheme	Throughput (Mbits/s)
FUN-1	0.669
FUN-2	0.691
BATS	0.330
Fountain	0.385
RLNC	0.291
COPE	0.493
TCP	0.451

change over time. Hence, the network topology is dynamic.

In this set of experiments, the range of the nodes' speed is from 5 meters/s to 10 meters/s. All the nodes move in a square area of 3000 meters by 3000 meters. We measure the throughput between a specific pair of source/destination nodes. This pair of source/destination nodes do not move and are not in the transmission range of each other. Hence, they need relay nodes to forward packets to the destination. The relay nodes are moving around. The number of hops between the intended source and the intended destination is varying since the relay nodes are moving around.

To make the experiments more realistic, we also generate some background traffic. The background traffic is generated in the following manner: we randomly select 100 pairs of nodes out of the 400 nodes; generate a Constant Bit Rate (CBR) flows between each pair of nodes. Each CBR flow lasts for 30 seconds with a random start time. Since the data rate needs to be constant for CBR, the source generates a packet every T_c second ($T_c \in (0, 1]$); the packet size is 1024 bytes. For example, for $T_c = 1$ second, the data rate is 1024 bytes/s. The number of hops from the source node to the destination node is random, depending on the positions of all the nodes. Since all the nodes are mobile, the network topology is dynamic.

In this set of experiments, the number of native packets to be transmitted by the source under study is 1600 packets, i.e., $K = 1600$. Table III shows the throughput of seven schemes under Case 3. We have the following observations:

- FUN-2 achieves the highest throughput and FUN-1 achieves the second highest throughput. This is because the number of hops in Case 3 is usually small (mostly two hops) and hence FUN-2 performs better than FUN-1 as in Case 1.
- COPE achieves the third highest throughput and TCP achieves the fourth highest throughput. Their throughput is higher than the BATS code, the fountain code, and RLNC; this may be because congestion and MAC contention are serious performance-limiting problems in multihop wireless networks and COPE and TCP both have congestion control to avoid congestion/contention with the background traffic while the BATS code, the fountain code, and RLNC do not. We will study how congestion control (e.g., back-pressure) affects the throughput in our future work.
- The fountain code achieves a higher throughput than the BATS code. This is because when a relay node moves out of the transmission range of the source node, the BATS code suffers more than the fountain code. This can be illustrated by an example for the BATS code: a relay node may hold $M/2$ packets of the same batch when it moves out of the transmission range of the source; if this relay node does not come back within the transmission range of the source, the $M/2$ packets will never be forwarded to the destination since they are waiting for the batch to be complete to be allowed to conduct network coding. This waste of $M/2$ packets reduces the throughput of the BATS code. In comparison, the fountain code does not have a batch structure; the relay node always forwards any packet in its buffer without waiting for future packets.
- RLNC achieves a lower throughput than the BATS code as in Case 3.

In summary, all the experimental results demonstrate that our FUN approach achieves higher throughput than BATS code, the best fountain code (RQ code), RLNC, COPE, and TCP for multihop wireless networks.

VI. CONCLUSION

This paper is concerned with the problem of information spreading over lossy communication channels. To address this problem, a joint FoUntain coding and Network coding (FUN) approach has been proposed. The proposed FUN architecture provides a unified framework for FUN-1, FUN-2, BATS codes, fountain codes, RLNC, COPE, and existing protocols. The novelty of our FUN approach lies in combining the best features of fountain coding, intra-session network coding, and cross-next-hop network coding. As such, our FUN approach is capable of achieving unprecedented high throughput over lossy channels. Experimental results demonstrate that our FUN approach achieves higher throughput than existing schemes for multihop wireless networks.

It is worth mentioned that the computational complexity of FUN-1 and FUN-2 (in terms of number of additions and multiplications per coded packet) is linear w.r.t. the batch size M. Since M is usually small, the delay incurred by FUN-1 and FUN-2 at a terminal node or a relay node is small.

Our future work includes extending intra-session network coding to general intra-session network coding, which applies to both unicast and multicast.

ACKNOWLEDGMENT

This work was supported in part by NSF CNS-1116970. The authors would like to thank Drs. Shenghao Yang and Raymond Yeung for sharing the computer programs of BATS codes and insightful comments. Thanks also go to Dr. Muriel Médard for constructive suggestions and comments.

REFERENCES

[1] T. S. Rappaport, *Wireless communications: principles and practice*. Prentice-Hall: Upper Saddle River, NJ, 1996.

[2] J. Qureshi, C. H. Fohy, and J. Cai, "Primer and recent developments on fountain codes," *arXiv preprint arXiv:1305.0918*, 2013.

[3] S. Yang and R. W. Yeung, "Batched sparse codes," *submitted to IEEE Transactions on Information Theory*.

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[5] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[6] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, 2003.

[7] Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," Microsoft Corporation, Redmond, WA, Tech. Rep., 2004.

[8] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, 2006, pp. 243–254.

[9] S. Rayanchu, S. Sen, J. Wu, S. Banerjee, and S. Sengupta, "Loss-aware network coding for unicast wireless sessions: design, implementation, and performance evaluation," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1, 2008, pp. 85–96.

[10] D. S. Lun, M. Médard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3–20, 2008.

[11] S. Yang and R. W. Yeung, "Coding for a network coded fountain," in *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, 2011, pp. 2647–2651.

[12] S. Lin and D. J. Costello, *Error control coding*, 2nd ed. Pearson Prentice-Hall: Upper Saddle River, NJ, 2004.

[13] M. Wang and B. Li, "Lava: A reality check of network coding in peer-to-peer live streaming," in *IEEE INFOCOM 2007*, pp. 1082–1090.

[14] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[15] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Medard, "Codecast: a network-coding-based ad hoc multicast protocol," *IEEE Wireless Communications*, vol. 13, no. 5, pp. 76–81, 2006.

[16] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proceedings of ACM SIGCOMM*, 2007.

[17] A. Shokrollahi and M. Luby, "Raptor codes," *Foundations and Trends in Communications and Information Theory*, vol. 6, no. 3–4, pp. 213–322, 2011. [Online]. Available: http://dx.doi.org/10.1561/0100000060

[18] "Qualnet web site," http://web.scalable-networks.com/content/qualnet.

[19] G. Holland and N. Vaidya, "Analysis of tcp performance over mobile ad hoc networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 275–288, 2002.

Algorithm 4 Inner encoding of FUN-2

```
1:  **for** j=0 **to** 1 **do**
2:    Initialize two buffers for Destination $j$, which are denoted by $F_{j,new}$ and $F_{j,old}$;
3:  **end for**
4:  **while** at least one flow (forward or backward flow) is alive **do**
5:    **if** Layer 3 output queue is not empty **then**
6:      Pick Packet $y_{i,m}$ at the head of Layer 3 output queue; assume the destination of Packet $y_{i,m}$ is $j$;
7:      **switch** (State of Buffers $F_{j,new}$, $F_{j,old}$, $F_{1-j,old}$)
8:      **case** Buffer $F_{j,new}$ is not complete**:**
9:        Insert $y_{i,m}$ to the $m$-th position of Buffer $F_{j,new}$;
10:       **if** Buffer $F_{j,new}$ is complete **then**
11:         Goto Step 7;
12:       **end if**
13:     **case** Buffer $F_{j,new}$ is complete **AND** Buffer $F_{j,old}$ is empty **AND** Buffer $F_{1-j,old}$ is empty**:**
14:       Move all the packets in $F_{j,new}$ to $F_{j,old}$;
15:       **if** the packets in $F_{j,old}$ were mixed before **then**
16:         Apply RLNC to all the packets in $F_{j,old}$ and generate $M$ re-coded packets;
17:         Move the re-coded packets to the unicast output queue of Layer 2;
18:       **end if**
19:     **case** Buffer $F_{j,new}$ is complete **AND** Buffer $F_{j,old}$ is empty **AND** Buffer $F_{1-j,old}$ is complete**:**
20:       Move all the packets in $F_{j,new}$ to $F_{j,old}$;
21:       **if** the packets in $F_{j,old}$ were mixed before **then**
22:         Apply RLNC to all the packets in $F_{j,old}$ and generate $M$ re-coded packets;
23:         Move the re-coded packets to the unicast output queue of Layer 2;
24:       **else**
25:         Apply RLNC to all the packets in $F_{j,old}$ and $F_{1-j,old}$ and generate $M$ re-coded packets;
26:         Move the re-coded packets to the broadcast output queue of Layer 2;
27:       **end if**
28:     **case** Buffer $F_{j,new}$ is complete **AND** Buffer $F_{j,old}$ is complete **AND** Buffer $F_{1-j,old}$ is empty**:**
29:       Apply RLNC to all the packets in $F_{j,old}$ and generate $M$ re-coded packets;
30:       Move the re-coded packets to the unicast output queue of Layer 2;
31:       Move all the packets in $F_{j,new}$ to $F_{j,old}$;
32:       **if** the packets in $F_{j,old}$ were mixed before **then**
33:         Apply RLNC to all the packets in $F_{j,old}$ and generate $M$ re-coded packets;
34:         Move the re-coded packets to the unicast output queue of Layer 2;
35:       **end if**
36:     **end switch**
37:     **if** $y_{i,m}$ has not been inserted into $F_{j,new}$ **then**
38:       Insert $y_{i,m}$ to the $i$-th position of $F_{j,new}$;
39:     **end if**
40:   **end if**
41: **end while**
```