BATS: Network Coding in Action

Shenghao Yang*, Raymond W. Yeung†‡, Jay H.F. Cheung† and Hoover H.F. Yin†
*Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
†Institute of Network Coding, The Chinese University of Hong Kong, Hong Kong, China

***Abstract*—BATS code is a low-complexity random linear network coding scheme that can achieve asymptotic bandwidth optimality for many types of networks with packet loss. In this paper, we propose a BATS code based network protocol and evaluate the performance by real-device experiments. Our results demonstrate significant ready-to-implement gain of network coding over forwarding in multi-hop network transmission with packet loss. We also propose an improved protocol to handle the practical issues observed in the experiments.**

I. Introduction

Packet loss in network communications is a general phenomenon. For example, due to noise and interference in wireless communication, the packets transmitted on the network links may not be correctly received. In moden wireline networks (e.g., the Internet backbone), packet loss is mainly due to congestion. The corrupted packets are detected and treated as lost packets. In this paper, we propose and conduct experiments on two network coding [1]–[3] based protocols for reliable transmission over networks with packet loss.

It is well known that for multicast networks, linear network coding [2], [3] in general has throughput gain over forwarding. For unicast networks, in the presence of packet loss, network coding can also improve the throughput. For example, in the line network illustrated in Fig. 1, wireless network links exist only between two neighboring nodes. Suppose the packets transmitted on a network link are erased independently with probability 0.1. If only forwarding is applied at the intermediate nodes, after L hops, the network throughput is upper bounded by $(1-0.1)^L$ packet per timeslot. In other words, while the capacity of the network is 0.9 packet per timeslot regardless of L (see discussion below), the network throughput decreases exponentially with L.

By means of *retransmission*, it is actually possible to achieve the capacity of the network in Fig. 1 in an ideal setting. One typical retransmission scheme is that the source node encodes its packet using a fountain code (e.g., LT codes [4], Raptor codes [5] and online codes [6]) and each node retransmits the packets that are not correctly received by the node at the next hop. However, the feedback required by retransmission costs bandwidth, which can be expensive, e.g., in wireless communication. In scenarios like satellite and deep space communication, feedback has long delay or may not even be available. Moreover, a retransmission scheme as described is not capacity achieving for networks with multicast links, e.g., the example in Fig. 2.

‡R. W. Yeung is also with the Department of Information Engineering, The Chinese University of Hong Kong, N.T., Hong Kong, the Shenzhen Key Laboratory of Network Coding Key Technology and Application, and Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China.

The following *baseline random linear network coding (RLNC) scheme* can achieve the capacity of networks with packet loss for a wide range of scenarios [7]–[13], including the networks in Figs. 1 and 2. The source node transmits random linear combinations of the input packets and an intermediate node transmits random linear combinations of the packets it has received. Note that no erasure coding is required for each link though packet loss is allowed. The network code itself plays the role of an end-to-end erasure code. A destination node can decode the input packets after it has received enough coded packets with linearly independent coding vectors.

However, complexity issues prevent practical implementation of the baseline RLNC scheme for real systems. These issues include

1) the computational cost of encoding and decoding at the source and sink nodes, respectively,
2) the storage and computational cost of network coding at the intermediate nodes, and
3) the coefficient vector overhead.

One effective strategy to resolve the above issues is to restrict the application of network coding to small subsets of the input packets [8]. In addition to disjoint subsets of the input packets [8], [14], this strategy can be applied on overlapped subsets of the input packets [15]–[19]. More sophisticated approaches apply network coding to small subsets of the coded packets generated from the input packets [20]–[24]. Among these approaches, *BATS codes* [20], [21] have the highest and close-to-optimal achievable rates [24]. Further, as a matrix generalization of fountain codes, BATS codes have the unique rateless property: A BATS code encoder can generate potentially unlimited number of *batches*, each of which consists of a set of coded packets. During the network transmission, network coding is restricted to the packets belonging to the same batch.

In this paper, we discuss how to design BATS code based network transmission protocols (BATS protocols), which have a different nature compared with traditional network protocols based on retransmission/forwarding. We focus on the major design issues of BATS protocols in handling both independent and burst packet loss. We propose two protocols, BATSpro-1 and BATSpro-2, both of which do not need any feedback for reliable transmission and can readily be used in line networks and its generalizations (e.g., the network in Fig. 2).

BATSpro-1 has a simple scheduling strategy and small storage requirement at the intermediate nodes: only one batch (e.g., 16 packets) is cached at an intermediate node. Our experiments show that in line networks, the throughput of

Fig. 1: A three-hop wireless network. Node R_0 is the source node, node R_3 is the destination node, and nodes R_1 and R_2 are the intermediate nodes that do not demand the input packets. Wireless network links exist only between two neighboring nodes. Each link can transmit one packet per timeslot.

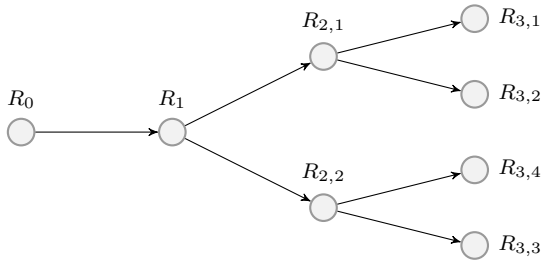

Fig. 2: A tree-type wireless network with multicast links. Node R_0 is the source node, nodes $R_{3,1}$, $R_{3,2}$, $R_{3,3}$ and $R_{3,4}$ are the destination nodes, and nodes R_1, $R_{2,1}$ and $R_{2,2}$ are the intermediate nodes that do not demand the input packets. The two outgoing links of $R_1/R_{2,1}/R_{2,2}$ share the same wireless channel.

BATSpro-1 decreases very slowly when the number of hops L increases, and it is much higher than that of using only forwarding at the intermediate network nodes. BATSpro-2 is an improvement of BATSpro-1 for networks with burst packet loss and dynamic link status, where more sophisticated scheduling and cache management are involved.

Line topology is a basic building block for general networks. As enabling techniques for line networks, our protocols can be extended for general networks. The BATS code protocols introduced here can be used with other wireless network technologies. For example, Huang et al. [26] have demonstrated the performance of BATS codes with the cross-next-hop network coding [27] and with multiple concurrent transmission sessions.

II. BATS Codes

We give an introduction to BATS codes in this section and refer readers to [21] for a more detailed discussion. In the sequel, a packet is regarded as a column vector over a finite field $\mathbb{F}$ of size q (e.g., $q = 2^8$). Fix integers K and T. We encode K input packets, each of which denoted by a column vector in $\mathbb{F}^T$. We equate a set of packets to a matrix formed by juxtaposing the packets in this set.

A. Encoding

A BATS code consists of an inner code and an outer code over the field $\mathbb{F}$. The outer code is a matrix generalization of a fountain code, and hence rateless. The outer code encodes the file to be transmitted into *batches*, each containing M packets. A batch is generated as follows:

1) Sample a degree distribution $\Psi = (\Psi_0, \Psi_1, \cdots, \Psi_D)$ and obtain a *degree* d with probability Ψ_d, where D is the maximum degree;
2) Uniformly at random choose d input packets and form a matrix $\mathbf{B}$ by juxtaposing the d packets;
3) The batch $\mathbf{X}$ is generated by
$$\mathbf{X} = \mathbf{BG}, \tag{1}$$
where $\mathbf{G}$ is a $d \times M$ matrix over $\mathbb{F}$, called the *generator matrix* of the batch.

All the batches are independently generated using the same three steps. The generator matrix $\mathbf{G}$ can be generated randomly or designed deterministically, and is known by the decoder. When $M = 1$ and the components of the generator matrices are all nonzero, the above batch encoding process becomes the encoding of LT codes.

We now turn to the inner code, which is formed by the linear transformations on the batches. The batches generated by the outer code are transmitted in a network employing linear network coding. We assume that linear network coding at the intermediate network nodes is only performed among packets belonging to the same batch so that the end-to-end transformation of each batch is a linear operation. Let $\mathbf{H}$ be the transfer matrix of a batch and $\mathbf{Y}$ be the output (received) packets of the batch. We have
$$\mathbf{Y} = \mathbf{XH} = \mathbf{BGH}. \tag{2}$$
The number of rows of $\mathbf{H}$ is M. The number of columns of $\mathbf{H}$ corresponds to the number of packets received for the i-th batch, which may vary for different batches and is finite. We assume that $\mathbf{H}$ is known by the decoder. In linear network coding, this knowledge can be obtained at the destination nodes through the coefficient vectors.

Suppose that the transfer matrices of the first n batches are $\mathbf{H}_1$, $\mathbf{H}_2$, …, $\mathbf{H}_n$. Denote by $\mathrm{rk}(\mathbf{H})$ the rank of a matrix $\mathbf{H}$. Let
$$h_r = \frac{|\{k \in \{1, 2, \ldots, n\} : \mathrm{rk}(\mathbf{H}_k) = r\}|}{n}. \tag{3}$$
The empirical rank distribution of the transfer matrices $h = (h_0, h_1, \ldots, h_M)$ is an important parameter for the design of BATS codes, which will be discussed further.

B. Decoding

The inner code preserves the degrees of the batches so that an efficient belief propagation (BP) decoding algorithm can be used to jointly decode the outer code and the inner code. A destination node tries to decode the input packets using $\mathbf{Y}$ and the knowledge of $\mathbf{G}$ and $\mathbf{H}$ for all the received batches. We say a batch is decodable if $\mathrm{rk}(\mathbf{GH})$ is equal to the degree d, i.e., the linear system in (2) with $\mathbf{B}$ as the variable has a unique solution. The BP decoding of BATS codes keeps looking for decodable batches. If a decodable batch is found, the input packets of this batch are recovered by solving the associated linear system (2) and the recovered input packets are substituted into the batches that have not been solved. If a decodable batch cannot be found, the BP decoding stops. Our goal is to decode a given fraction of the input packets before the BP decoding stops.

To guarantee the success of the BP decoding, a proper degree distribution is crucial. The asymptotic analysis of BP decoding in [21] induces a degree-distribution optimization problem, which maximizes the coding rate and has the rank distribution h as a parameter. It is demonstrated numerically

for general cases that the BATS code with BP decoding achieves rates very close to the average empirical rank $\sum_i ih_i$, the theoretical upper bound on the achievable rate of the code in packets per batch.

C. Practical Design

We have discussed how to recover a given fraction of the input packets. To reliably transmit all the input packets, we can use the precode technique first introduced for Raptor codes. That is, before applying the batch encoding process, the input packets are first encoded using a traditional erasure code (called a *precode*). The batch encoding process is applied to the precoded input packets generated by the precode. If the BP decoding of the BATS code can recover a given fraction of the precoded input packets, the precode is able to recover the original input packets in face of a fixed fraction of erasures. Due to similar requirements, the precode for Raptor codes can be applied to BATS without much modifications. Readers can find the detailed discussion of these techniques in [28], [29].

Though the degree distribution optimized asymptotically performs well when the number of input packets is very large (e.g., 100,000), when the number of input packets is small, BP decoding tends to stop before the desired fraction of input packets are decoded. When the BP decoding stops, one approach to continue the decoding process is *inactivation* [28], [30]–[32]. With inactivation, when there are no decodable batches, we instead pick an undecoded input packet b and mark it as *inactive*. We substitute the inactive packet b into the batches like a decoded packet, except that b is an indeterminate. The decoding process is repeated until all input packets are either decoded or inactive. The inactive input packets can be recovered by solving a linear system of equations using Gaussian elimination. In a nutshell, inactivation decoding trades computation cost (decoding inactive input symbols using Gaussian elimination) with coding overhead. BATS codes with inactivation decoding has demonstrated nearly optimal performance in simulation [21].

III. Basic BATS Protocol

Starting from this section, we discuss BATS protocols that use BATS codes in practical network transmission. A general BATS protocol has three kinds of modules in two layers as illustrated in Fig. 3: The higher layer includes the encoding and decoding modules of BATS codes and the lower layer includes multiple batch forwarding (BF) modules. In this paper, we only consider the sequential concatenation of these modules as shown in this figure.

We first introduce a basic BATS protocol for multihop wireless networks, called *BATSpro-1*. Though simple, it demonstrates the desired performance gain in some scenarios. We will discuss later how to improve the basic protocol to handle some practical issues.

A. Encoding Module

BATS encoding and decoding modules are employed only at the source and the destination nodes, respectively. BATS encoding takes a file as input and generates batches using the algorithm described in the last section. The output packets of the encoding module have the following structure. Consider

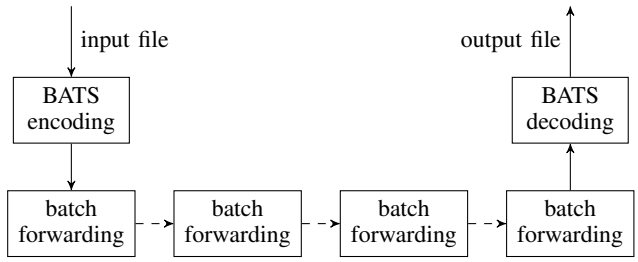

Fig. 3: Structure of a BATS code protocol in the line network in Fig. 1. Each batch forwarding module runs on a network node. The first node includes an encoding module and the last node includes a decoding module.

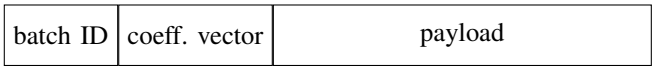

Fig. 4: Format of an output packet. A batch ID usually has 16 to 32 bits. Taking $M = 16$ and $q = 2^8$ as example, a coefficient vector has 16 bytes.

a batch $\mathbf{X}$ generated as (1). For the i-th packet in $\mathbf{X}$, we attach a coefficient vector e_i which is a (column) vector of M symbols in $\mathbb{F}$ known by the decoder. It is required that e_i, $i = 1, \ldots, M$ are linearly independent. Here we assume that all the components of e_i are zero except that the i-th component is one. The encoding module outputs M packets for a batch consecutively, where the i-th packet $p[i]$ includes three fields:

1) batch ID: a unique identifier for each batch,
2) coefficient vector e_i, and
3) the i-th column of $\mathbf{X}$ as the payload.

The format of an output packet is illustrated in Fig. 4.

A packet of the format in Fig. 4 is transmitted between the modules of the BATS code protocol and is called a *BATS packet*. For a BATS packet p, the three fields are referred to as p^I, p^C and p^B, respectively. We will simply refer to a BATS packet as a packet when the context makes it clear.

B. Batch Forwarding

The BF module is employed at all the nodes. Under the principle that only packets of the same batch can be network coded, we have a lot of freedom in designing batch forwarding, including how to manage the buffer content, and how to schedule the transmission of batches/packets. We first introduce the BF module of BATSpro-1.

The main procedure of the BF module of BATSpro-1 has the flow chart given in Fig. 5. The BF module repeats this procedure for each received packet. The design of this module is based on the assumption that the packets of the same batch are received consecutively. The batch of the received packet in the last run of the main procedure is called the *current batch*. A BF module has a variable to keep the batch ID of the current batch, which is initially null. The main procedure has two branches:

- If the batch ID of a received packet is the same as the current batch, the packet is just saved in the cache of the module.

- If the batch ID of a received packet is different from the current batch, in addition to saving the packet, the following operations are performed: First, the current batch is marked as a *readied batch*, and the variable of the current batch is changed to the batch ID of the received packet. Second, a recoding procedure (to be specified later) is applied on the readied batch to generate M recoded BATS packets, which are then transmitted to the next hop.

At the source node and the destination node, the operation of the BF module can be simplified to forwarding only.

Suppose the readied batch has $N \leq M$ packets, denoted by $p[i]$, $i = 1, \ldots, N$. The recoding procedure generates M recoded BATS packets $\tilde{p}[j], j = 1, \ldots, M$ as follows. First $\tilde{p}^I[j]$ is the batch ID of the readied batch. The other two fields of $\tilde{p}[j]$ are calculated by

$$\tilde{p}^C[j] = \sum_{i=1}^{N} \alpha[i,j] p^C[i], \quad \tilde{p}^B[j] = \sum_{i=1}^{N} \alpha[i,j] p^B[i],$$

where $\alpha[i,j], i = 1, \ldots, N$ are the linear combination coefficients chosen from $\mathbb{F}$. There are various approaches to choose $\alpha[i,j]$. One approach is to choose $\alpha[i,j]$ uniformly at random from the field $\mathbb{F}$. Another approach, called *systematic recoding*, chooses the coefficients of the first N recoded packets such that $\tilde{p}^B[j] = p^B[j], j \leq N$ and the coefficients of the remaining recoded packets randomly. Systematic recoding can reduce the computation cost of recoding significantly when $M - N$ is small.

Now let us check the storage cost of a BF module at an intermediate node. Suppose the BF modules output the BATS packets of the same batch ID consecutively and the network links preserve the order of the input packets. Then, the BATS packets of the same batch ID arrive a BF module consecutively. Thus, at most M BATS packets are cached before recoding is applied.

C. Decoding Module

Consider a batch $\mathbf{X}$ generated at the source node and a sequence of received BATS packets at the destination node $p[i], i = 1, \ldots, N$ with the batch ID of $\mathbf{X}$. It can be verified inductively that

$$p^B[i] = \mathbf{X} p^C[i], \quad i = 1, \ldots, N.$$

Writing the above equations in matrix form, we obtain (2). The decoding module can apply the algorithm introduced in the last section to decode input packets.

As an option, the decoding module sends a feedback to the encoding module when the decoding succeeds.

IV. Experiments on BATSpro-1

In this section, we discuss the implementation and experiments of BATSpro-1.

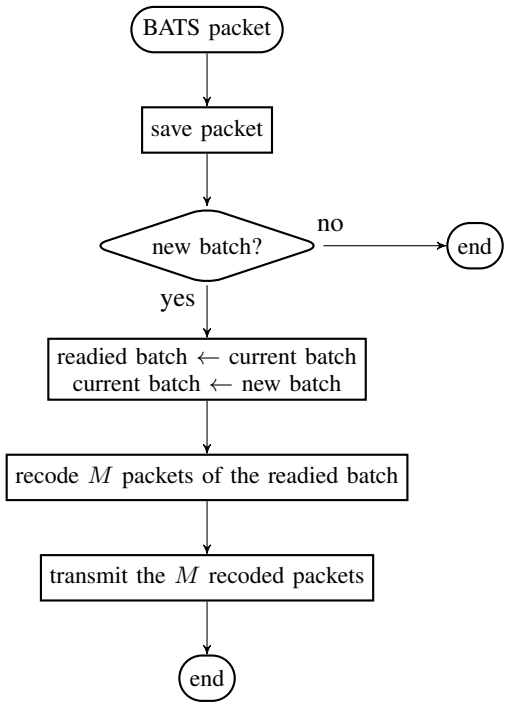

Fig. 5: Flow chart of the batch forwarding module of BATSpro-1 in an intermediate node.

A. Implementation in Real Devices

In the existing network protocol stack, one possible place to implement BATSpro-1 is the MAC layer or between the IP layer and the MAC layer. Here we choose to implement BATSpro-1 at the application layer based on the UDP protocol. The IP protocol is almost transparent in our experiments so it has little effect on the performance. BATSpro-1 is implemented using C/C++ in linux operating systems (Ubuntu and OpenWRT) and portable to other operating systems.

We use two laptop computers as the source and destination nodes, respectively. Label the source node by R_0 and the destination node by R_L. The wireless interfaces of R_0 and R_L are set to the station mode and the access point (AP) mode, respectively. The intermediate nodes are WiFi routers with dual radios that operate on different WiFi channels. These routers are labeled by $R_1, R_2, \ldots, R_{L-1}$, respectively. For each WiFi router, one of the interfaces is set to the station mode and the other is set to the AP mode. We connect these devices to form a line network:

- The source node is connected to the AP of R_1.
- The interface of R_i in the station mode is connected to the AP of R_{i+1}, for $i = 1, \ldots, L-2$.
- The interface of R_{L-1} in the station mode is connected to the destination node.

We choose the wireless channels carefully to avoid cross channel interference. Further, retransmission in WiFi is suppressed since network coding has the function of erasure correction.

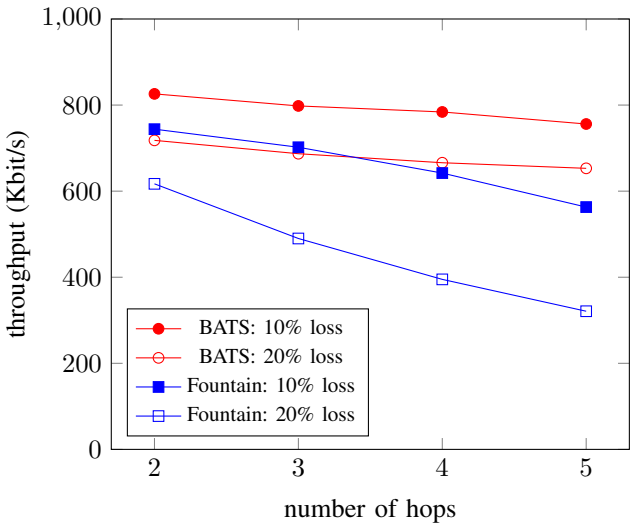

Fig. 6: Compare BATSpro-1 and fountain code protocol when the link loss rate is 10 and 20 percent. For both protocols: $K = 5120$, $q = 2^8$ $T = 1024$, and the source node transmission rate is 1000 Kbit/s. For BATSpro-1, $M = 16$.

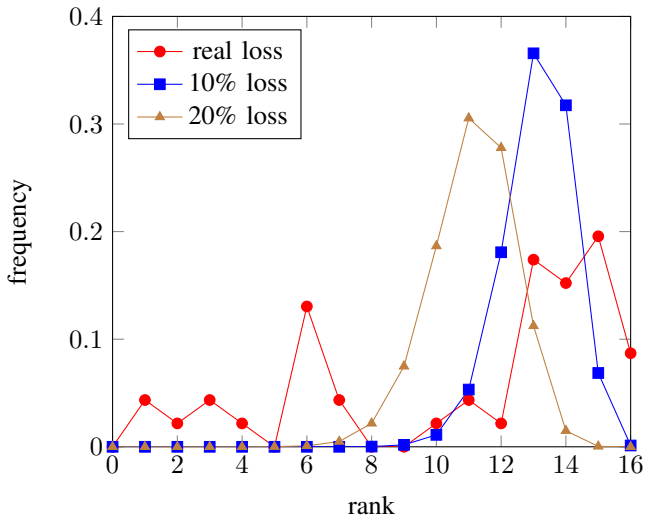

Fig. 7: Empirical rank distribution example in a four-hop line network with real-world packet loss. The two curves marked by 10% and 20% loss are the expected rank distributions of the four-hop line network with 10% and 20% percent independent loss rates, respectively. The average ranks of the three cases are 11.07, 13.13 and 11.12, respectively.

TABLE I
COMPARE BATSPRO-1 AND FOUNTAIN CODES IN A FOUR-HOP LINE NETWORK WITH REAL-WORLD PACKET LOSS. FOR BOTH PROTOCOLS: $K = 512$, $q = 2^8$, $T = 1024$, AND THE SOURCE NODE TRANSMISSION RATE IS 500 KBIT/S. FOR BATSPRO-1, $M = 16$.

	Fountain	BATS
trials	279.45	291.45
	241.42	314.10
	315.36	319.78
	237.53	362.61
	260.61	296.41
mean	266.87	316.87

B. Compare with Fountain Codes

We compare BATSpro-1 with a fountain code protocol where the encoding/decoding module is similar to BATS encoding/decoding module with $M = 1$, but the intermediate nodes only forward packets. So in the fountain code protocol, the coefficient vector field is not required. Packet loss in WiFi occurs naturally due to noise and interference in air. Since the loss pattern fluctuates quickly, it is hard to conduct fair comparison. We instead artificial delete a packet with certain probability at the WiFi interface independently. The devices in this case are put close enough so that the real-world loss is neglectable.

The experiment results are given in Fig. 6. For both BATS codes and Fountain codes, we use $K = 5120$, $q = 2^8$ and $T = 1024$. For the BATS codes, we use batch size 16. The source node transmission rate is 1000 Kbit/s for both BATSpro-1 and the fountain code protocol.

We also conduct the experiments with real-world packet loss (not the artificially generated packet loss). In Table I, we give the experiment results for five trials of BATSpro-1 and five trials of fountain codes in a network with four hops. In this case, $K = 512$ and the source node transmission rate is 500 Kbit/s for both BATSpro-1 and the fountain code protocol.

V. IMPROVED BATS PROTOCOL

In this section, we first discuss a couple of issues of BATSpro-1, and then propose a new protocol, BATSpro-2, to resolve these issues.

A. Burst Loss

Though BATSpro-1 works very well when the link loss is independent as we demonstrated in the last section, the advantage of network coding in BATSpro-1 can be reduced when the link loss is bursty. Burst loss tends to occur in wireless transmission due to interference and fading.

Let us use an example to illustrate the issue of burst loss for BATSpro-1. Suppose in a line network, 10 percent of the packets transmitted on a link are erased, but the losses on each link are not independent: the packets belonging to a batch are either all erased or all correctly received. We can check that in BATSpro-1, an intermediate node always receives M packets for a batch. For such a loss pattern, the performance of BATSpro-1 is reduced to $(1-0.1)^L$ packet per timeslot for L hops, the same as that of the fountain code calculated in the introduction.

In a real-world scenario, the burst loss is not as extreme as the above example. We sample an empirical rank distribution in our real-world experiments of BATSpro-1 given in Fig. 7, where the expected rank distributions of independent link loss are also plotted for comparison.

One approach to resolve the above issue of burst loss is to interleave the packets of different batches for transmission. This technique is used in the physical layer of wireless transmission to resolve the issue of burst error. When using interleaving, since the packets of the same batch are not transmitted consecutively, more packets are required to be cached at an intermediate node so that effective recoding can be applied.

B. Adaptive Recoding

We say a set of packets belonging to the same batch are linearly independent if the coefficient vectors of these packets are linearly independent. Define the *rank of a batch* at a network node as the maximum number of linearly independent received packets of the batch. Note that the rank of a batch at the destination node is exactly the rank of the transfer matrix of the batch (see (2)). The theory of BATS codes tells us that the inner code should maximize the average empirical rank $\sum_i ih_i$. Since the rank of a batch tends to decrease for each hop of transmission, the recoding scheme should try to maximize the total expected ranks of all batches at the next hop node.

In BATSpro-1, the recoding generates the same number of coded packets for all batches, no matter how many packets are received in the batch. This is not optimal for maximizing the total expected ranks. Intuitively, we should transmit more packets for a batch with a higher rank because compared with a batch with a lower rank, the former contains more useful information for decoding. This intuition can be justified in Appendix A.

Further, due to interleaving, an intermediate node may not receive the packets of the same batch consecutively. Therefore, an adaptive recoding approach must be applied based on the currently received packets of a batch. For each batch b in the cache of a node, we maintain two variables:

1) r_b: the rank of the packets received, and
2) t_b: the number of packets transmitted.

We define a *priority function* F which takes these two variables as inputs. Hence, the *priority* of a batch b is defined as $F(b) = F(r_b, t_b)$. Whenever there is an opportunity for transmission, the node transmits a packet of the batch b with the largest value of $F(b)$. One example of the priority function is

$$F(b) = r_b - t_b.$$

When two batches b and b' have $r_b - t_b = r_{b'} - t_{b'}$ and $r_b > r_{b'}$, we may opt to transmit a packet of batch b. Accordingly, we may modify the above definition as

$$F(b) = r_b - t_b + \frac{r_b}{M}.$$

The use of the above priority function will be justified in Appendix B.

To simplify the computation at the intermediate nodes, we may not want to check the linear independence of the packets of a batch. Instead, we may just count how many packets of a batch are received, and redefine r_b as the number of packets received for batch b. The above adaptive recoding approach can be similarly applied.

C. Batch Forwarding in BATSpro-2

BATSpro-2 shares the same encoding and decoding modules as BATSpro-1. There are two main procedures in the BF module of BATSpro-2, named the receiving procedure and the transmitting procedure. The flow charts of the procedures are given in Fig. 8 and Fig. 9 respectively.

The receiving procedure repeats for each received packet. There is no assumption on the ordering of the received packets.

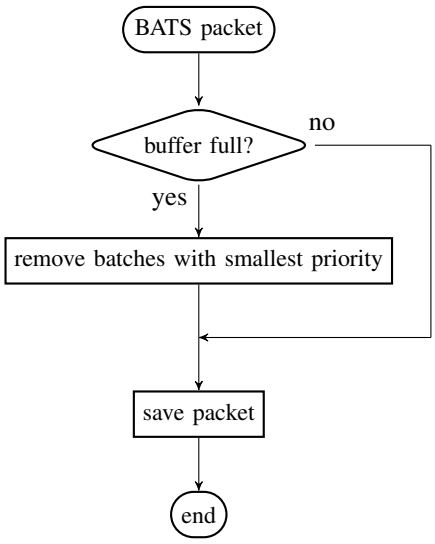

Fig. 8: Flow chart of the receiving procedure in the BF of BATSpro-2 at an intermediate node.

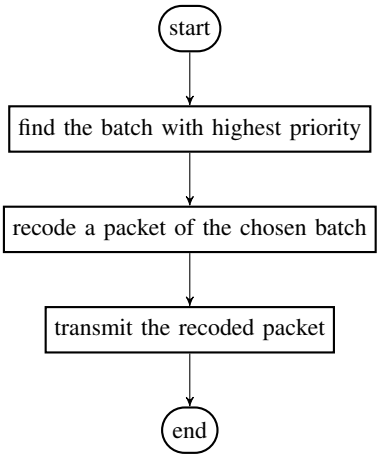

Fig. 9: Flow chart of the transmitting procedure in the BF of BATSpro-2 at an intermediate node.

The buffer size is a tunable parameter according to the memory limitation. If the buffer is full, batches with the smallest priority are removed. To ensure that there are enough slots in the buffer for the batch of the receiving packet, the removal process continues until there are at least M empty slots left.

The transmitting procedure is repeated automatically. The batch with the highest priority is selected, and a packet for the selected batch is generated for transmission. The way to generate a new packet is exactly the same as the way to generate a recoded packet of the readied batch in BATSpro-1. Further, the priority-based batch selection also plays the role of an interleaver: we deliberately decrease the priority of a batch if it was just selected.

At the destination node, the BF module of BATSpro-2 simply forwards packets to the decoding module.

VI. Concluding Remarks

In this paper, we discussed in details the design and the implementation of BATS code enabled network transmission protocols for multihop wireless networks. Our experiments show that using BATS code can achieve much higher rates than using fountain codes in multihop wireless networks. There are more refinements to be considered in the future. For example, when feedback is available, the protocols can make use of this additional information for recoding and rate control. We will conduct more experiments in various real-world scenarios to evaluate the refined protocols.

Acknowledgement

This work was partially funded by a grant from the University Grants Committee of the Hong Kong Special Administrative Region (Project No. AoE/E-02/08) and a grant from the Shenzhen Key Laboratory of Network Coding Key Technology and Application, Shenzhen, China (ZSDY20120619151314964). We thank P2 Mobile Technologies Limited (P2MT) for supplying the wireless routers in our experiments, and thank Qiaoqiao Zhou for optmizing the degree distributions of BATS codes.

Appendix
Justifications of Adaptive Recoding

Consider a batch with rank r at an intermediate node, and t recoded packets are generated uniformly at random as we specified in Section III-B. The t recoded packets are transmitted to the node at the next hop and each of the transmitted packets is independently erased with probability p. We are interested in two quantities in this probability model. The first is the expected rank of the batch at the next hop:

$$E_q(r,t) = \sum_{i=0}^{t} \binom{t}{i}(1-p)^i p^{t-i} \sum_{j=0}^{\min\{i,r\}} j\zeta_j^{i,r}$$

where $\zeta_j^{i,r}$ is the probability that any i already received packets have rank j. The second is the probability $f_q(r,t)$ that a new recoded packet u generated uniformly at random, if correctly transmitted, is linearly independent with the already received packets of this batch. We have

$$f_q(r,t) = \sum_{i=0}^{t} \binom{t}{i}(1-p)^i p^{t-i} \sum_{j=0}^{\min\{r,i\}} \zeta_j^{i,r}(1-q^{-r+j}).$$

For convenience, we write $B_p(t,i) = \binom{t}{i}(1-p)^i p^{t-i}$.

A. Number of Packets to Recode

See a characterization of $\zeta_j^{i,r}$ in [21]. We know that

$$\lim_{q\to\infty} \zeta_j^{i,r} = \begin{cases} 1 & j = \min\{r,i\}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, when q is large (e.g., $q = 2^8$), we can approximate $E_q(r,t)$ by

$$E(r,t) = \sum_{i=0}^{t} B_p(t,i)\min\{r,i\}.$$

It is straightforward to check that

$$E(r,t+1) = \begin{cases} E(r,t) + (1-p) & t < r, \\ E(r,t) + (1-p)\sum_{i=0}^{r-1} B_p(t,i) & t \geq r. \end{cases}$$

Now we consider two batches with rank r_1 and r_2 at a node, where $r_1 > r_2$. Suppose that we transmit $M+s$ packets for the batch with rank r_1 and $M-s$ packets for the batch with rank r_2. We want to show that $s = 0$ as in BATSpro-1 is not the optimal choice. Let $\beta(s) = E(r_1, M+s) + E(r_2, M-s)$. We can check that

$$\begin{aligned}
&\beta(1) - \beta(0) \\
&= [E(r_1, M+1) - E(r_1, M)] - [E(r_2, M) - E(r_2, M-1)] \\
&= (1-p)\sum_{i=0}^{r_1-1} B_p(M,i) - (1-p)\sum_{i=0}^{r_2-1} B_p(M-1,i) \\
&\geq (1-p)\sum_{i=0}^{r_2} B_p(M,i) - (1-p)\sum_{i=0}^{r_2-1} B_p(M-1,i) \\
&\geq 0,
\end{aligned}$$

where first inequality follows from $r_1 - 1 \geq r_2$ and the last inequality is verified in the following lemma.

Lemma 1. *When $t > r$, $\sum_{i=0}^{r+1} B_p(t+1,i) \geq \sum_{i=0}^{r} B_p(t,i)$, where the equality holds only when $p = 0$ or 1.*

Proof: By using the formula

$$\binom{t+1}{i} = \binom{t}{i-1} + \binom{t}{i}, i = 1, 2, \ldots, t,$$

we have

$$\begin{aligned}
&\sum_{i=0}^{r+1} B_p(t+1,i) = \sum_{i=0}^{r+1} \binom{t+1}{i}(1-p)^i p^{t+1-i} \\
&= p^{t+1} + p\sum_{i=1}^{r+1} \binom{t}{i}(1-p)^i p^{t-i} + \sum_{i=0}^{r} \binom{t}{i}(1-p)^{i+1} p^{t-i} \\
&= \sum_{i=0}^{r} B_p(t,i) + \binom{t}{r+1}(1-p)^{r+1} p^{t-r} \geq \sum_{i=0}^{r} B_p(t,i),
\end{aligned}$$

where the equality holds only when $p = 0$ or 1. ∎

B. Priority Function

Similar to Section A in this Appendix, when q is large, we can approximate $f_q(r,t)$ by

$$\begin{aligned}
f(r,t) &= \sum_{i=0}^{\min\{t,r-1\}} \binom{t}{i}(1-p)^i p^{t-i} \\
&= \begin{cases} 1 & t < r, \\ \sum_{i=0}^{r-1} \binom{t}{i}(1-p)^i p^{t-i} & t \geq r. \end{cases}
\end{aligned}$$

The function $f(r,t)$ is the ideal priority function to be used for the independent erasure model defined at the beginning of this appendix. However, the priority function $F(r,t)$ given in Section V-B is computationally much simpler and hence more favored in real implementations. In the following, we check that the priority functions $F(r,t)$ and $f(r,t)$ are actually consistent in most cases.

It is easy to see that

1) $f(r+1,t) \geq f(r,t)$ for $t, r \geq 0$;
2) $f(r+1,t+1) = f(r,t) = 1$ for $t < r$;
3) $f(r+1,t+1) \geq f(r,t)$ for $t \geq r$ due to Lemma 1, where equality holds only when $p = 0$ or 1; and
4) $f(r,t+1) \leq f(r,t)$ for $t < r$.

On the other hand, we can check that $f(r,t+1) \leq f(r,t)$ for $t \geq r$ as

$$\begin{aligned} f(r,t+1) &= \sum_{i=0}^{r-1} \binom{t+1}{i} (1-p)^i p^{t+1-i} \\ &= p^{t+1} + p\sum_{i=1}^{r-1} \binom{t}{i} (1-p)^i p^{t-i} + \sum_{i=0}^{r-2} \binom{t}{i} (1-p)^{i+1} p^{t-i} \\ &= f(r,t) - \binom{t}{r-1} (1-p)^r p^{t+1-r} \leq f(r,t) \end{aligned}$$

where equality holds only when $p = 0$ or 1.

Consider two batches b and b' with $r_b = r$, $t_b = t$, $r_{b'} = r + \alpha$ and $t_{b'} = t + \beta$ for $\alpha \geq \beta$. We discuss the following three cases where $0 < p < 1$.

Case I: $\alpha \geq \beta \geq 0$. If $r_{b'} - t_{b'} = r - t + \alpha - \beta > r_b - t_b$, which implies $\alpha > \beta$, then $f(r+\alpha, t+\beta) \geq f(r,t)$ by applying the above properties of f inductively. If $r_{b'} - t_{b'} = r_b - t_b$, which implies $\alpha = \beta$, then $f(r_{b'}, t_{b'}) \geq f(r_b, t_b)$. Therefore, $f(r,t)$ and $F(r,t)$ are consistent.

Case II: $\alpha \geq 0 > \beta$. Now $r_{b'} - t_{b'} > r_b - t_b$ and $f(r + \alpha, t+\beta) \geq f(r+\alpha, t) \geq f(r,t)$. Therefore, $f(r,t)$ and $F(r,t)$ are consistent.

Case III: $0 > \alpha > \beta$. Now $r_{b'} - t_{b'} > r_b - t_b$. When $t+\beta < r+\alpha$, we have $f(r+\alpha, t+\beta) = 1 \geq f(r,t)$. However, $f(r,t)$ may be inconsistent when $t+\beta \geq r+\alpha$ since we may not have $f(r+\alpha, t+\beta) \geq f(r,t)$. Numerical results show that which one of $f(r+\alpha, t+\beta)$ and $f(r,t)$ is larger depends on the value of p. Since $f(r+\alpha, t+\beta) \geq f(r+\beta+1, t+\beta) \geq f(1, t-r)$, we have

$$\begin{aligned} & f(r+\alpha, t+\beta) - f(r,t) \\ \geq\ & p^{t-r} - \sum_{i=0}^{r-1} \binom{t}{i} (1-p)^i p^{t-i} \\ \geq\ & p^{t-r} - [1 - t(1-p)^{t-1} p - (1-p)^t], \end{aligned}$$

which is positive when p is sufficiently small. Therefore, the priority function $F(r,t)$ is also consistant with $f(r,t)$ when p is sufficiently small.

REFERENCES

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

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

[3] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[4] M. Luby, "LT codes," in *Proc. IEEE FOCS '02*, Nov. 2002, pp. 271–282.

[5] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[6] P. Maymounkov, "Online codes," NYU, Tech. Rep., Nov. 2002.

[7] T. Ho, B. Leong, M. Medard, R. Koetter, Y. Chang, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE ISIT '03*, Jun. 2003.

[8] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Allerton Conf. Comm., Control, and Computing*, Oct. 2003.

[9] S. Jaggi, P. A. Chou, and K. Jain, "Low complexity optimal algebraic multicast codes," in *Proc. IEEE ISIT '03*, Jun. 2003.

[10] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Proc. ACM SPAA '03*, New York, NY, USA, 2003, pp. 286–294.

[11] 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.

[12] Y. Wu, "A trellis connectivity analysis of random linear network coding with buffering," in *Proc. IEEE ISIT '06*, Seattle, USA, Jul. 2006.

[13] A. F. Dana, R. Gowaikar, R. Palanki, B. Hassibi, and M. Effros, "Capacity of wireless erasure networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 3, pp. 789–804, 2006.

[14] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE INFOCOM '05*, 2005.

[15] D. Silva, W. Zeng, and F. R. Kschischang, "Sparse network coding with overlapping classes," in *Proc. NetCod '09*, 2009, pp. 74–79.

[16] A. Heidarzadeh and A. H. Banihashemi, "Overlapped chunked network coding," in *Proc. ITW '10*, 2010, pp. 1–5.

[17] Y. Li, E. Soljanin, and P. Spasojevic, "Effects of the generation size and overlap on throughput and complexity in randomized linear network coding," *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 1111–1123, Feb. 2011.

[18] B. Tang, S. Yang, Y. Yin, B. Ye, and S. Lu, "Expander graph based overlapped chunked codes," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, Cambridge, MA, USA, July 1-6 2012, pp. 2451–2455.

[19] ——, "Expander graph based overlapped chunked codes," 2013, submitted for journal publication.

[20] S. Yang and R. W. Yeung, "Coding for a network coded fountain," in *Proc. IEEE ISIT '11*, Saint Petersburg, Russia, Aug. 2011.

[21] ——, "Batched sparse codes," 2014, accepted by IEEE Transactions on Information Theory. [Online]. Available: http://arxiv.org/abs/1206.5365

[22] K. Mahdaviani, M. Ardakani, H. Bagheri, and C. Tellambura, "Gamma codes: a low-overhead linear-complexity network coding solution," in *Proc. NetCod '12*, 2012, pp. 125–130.

[23] K. Mahdaviani, R. Yazdani, and M. Ardakani, "Linear-complexity overhead-optimized random linear network codes." [Online]. Available: http://arxiv.org/abs/1311.2123

[24] S. Yang and B. Tang, "From LDPC to chunked network codes," accepted by Information Theory Workshop (ITW), 2014 IEEE.

[25] S. Yang and R. W. Yeung, "Subset coding for communication systems," US Patent US 8,693,501, 2014.

[26] Q. Huang, K. Sun, X. Li, and D. Wu, "Just fun: A joint fountain coding and network coding approach to loss-tolerant information spreading," in *Proceedings of ACM MobiHoc*, Philadelphia, PA, USA, Aug. 2014.

[27] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *Proc. SIGCOMM '06*, New York, NY, USA, 2006.

[28] A. Shokrollahi and M. Luby, *Raptor Codes*, ser. Foundations and Trends in Communications and Information Theory. now, 2011, vol. 6.

[29] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ forward error correction scheme for object delivery – rfc 6330," 2011. [Online]. Available: http://datatracker.ietf.org/doc/rfc6330/

[30] A. Shokrollahi, S. Lassen, and R. Karp, "Systems and processes for decoding chain reaction codes through inactivation," U.S. Patent 6,856,263, Feb. 15, 2005.

[31] T. C. Ng and S. Yang, "Finite length analysis of BATS codes," in *Proc. IEEE Inte. Symp. on Network Coding NetCod '13*, Calgary, Canada, Jun. 7-9 2013.

[32] ——, "Finite-length analysis of BATS codes," 2013, submitted for journal publication. [Online]. Available: http://arxiv.org/abs/1312.4811