NWCRG
Internet-Draft
Intended status: Informational
Expires: 8 July 2023

S. Yang
CUHK(SZ)
X. Huang
R. W. Yeung
CUHK
J. K. Zao
NCTU

4 January 2023

BATS Coding Scheme for Multi-hop Data Transport
draft-irtf-nwcrg-bats-07

Abstract

Linear network coding can in general improve the network communication performance in terms of throughput, latency and reliability. BATched Sparse (BATS) code is a class of efficient linear network coding scheme with a matrix generalization of fountain codes as the outer code, and batch-based linear network coding as the inner code. This document describes a baseline BATS coding scheme for communication through multi-hop networks, and discusses the related research issues towards a more sophisticated BATS coding scheme. This document is a product of the Coding for Efficient Network Communications Research Group (NWCRG).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document specifies a baseline BATched Sparse (BATS) code [Yang14] scheme for data delivery in multi-hop networks, and discusses the related research issues towards a more sophisticated scheme. The BATS code described here includes an outer code and an inner code. The outer code is a matrix generalization of fountain codes (see also the RapterQ code described in RFC 6330 [RFC6330]), which inherits the advantages of reliability and efficiency and possesses the extra desirable property of being network coding compatible. The inner code, also called recoding, is formed by linear network coding for combating packet loss, improving the multicast efficiency, etc. A detailed design and analysis of BATS codes are provided in the BATS monograph [Yang17].

A BATS coding scheme can be applied in multi-hop networks formed by wireless communication links, which are inherently unreliable due to interference, fading, multipath, etc. Existing transport protocols like TCP use end-to-end retransmission, while network protocols such as IP might enable store-and-forward at the relays, so that packet loss would accumulate along the way.

A BATS coding scheme can be used for various data delivery applications like file transmission, video streaming over wireless multi-hop networks, etc. Different from traditional forward error correcting (FEC) schemes that are applied either hop-by-hop or end-to-end, the BATS coding scheme combines the end-to-end coding (the outer code) with certain hop-by-hop coding (the inner code), and hence can potentially achieve better performance.

The baseline coding scheme described here considers a network with multiple communication flows. For each flow, the source node encodes the data for transmission separately. Inside the network, however, it is possible to mix the packets from different flows for recoding. In this document, we describe a simple case where recoding is performed within each flow. Note that the same encoding/decoding scheme described here can be used with different recoding schemes as long as they follow the principle as we illustrate in this document.

In this document, we focus on the case that each flow has only one destination node. Communicating the same data to multiple destination nodes is called multicast. Refer to Section 4 for the further discussion of multicast.

The purpose of the baseline BATS coding scheme is twofold. First, it provides researchers and engineers a starting point for developing network communication applications/protocols based on BATS codes. Second, it helps to make the research issues clearer towards a sophisticated BATS code based network protocol. Important research directions include the security issues, congestion control and routing algorithms for BATS codes, etc.

This document is a product of and represents the collaborative work and consensus of the Coding for Efficient Network Communications Research Group (NWCRG). It is not an IETF product and is not an IETF standard.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. A Use Case of BATS Coding Scheme

The BATS coding scheme described in this document can be used, for example, by a Data Delivery Protocol (DDP). Though this document is not about a DDP, we briefly illustrate in this section how a BATS coding scheme is employed by a DDP to make the role of the coding scheme clear. Some terms that will be used in this section are summarized here:

- DDP: data delivery protocol.

- DDP packet: the packet formed by a DDP employing a BATS coding scheme.

- source packet: the packet formed by the data for delivery.

- outer encoder: the outer code encoder of a BATS code.

- recoder: the inner code encoder of a BATS code.

- outer decoder: the outer code decoder of a BATS code.

- coded packet: the packet generated by the outer code encoder or a recoder.

- batch: a set of coded packets generated by a BATS coding scheme from the same subset of the source packets.

- recoded packet: a coded packet generated by a recoder.

- degree: the number of source packets used to generate a batch by the outer encoder. The degree can be different for different batch.

Other common terms can be found in RFC 8406 [RFC8406].

2.1. Introduction

We describe a data delivery process that involves one source node, one destination node, and multiple intermediate nodes in between. A BATS coding scheme includes an outer code encoder (also called outer encoder), an inner code encoder (also called recoder), and an outer decoder which decodes the outer code and the inner code jointly as illustrated in Figure 1. The functions of the outer encoder, recoder and outer decoder are described below:

```
               |
               |  {set of source packets}
               v
       +-+-+-+-+-+-+-+-+
       | outer encoder |
       |       v       | source node
       |    recoder    |
       +-+-+-+-+-+-+-+-+
               |
               |  {set of DDP packets}
               v
       +-+-+-+-+-+-+-+-+
       |               |
       |    recoder    | intermediate node
       |               |
       +-+-+-+-+-+-+-+-+
               |
               |  {set of DDP packets}
               v
              ...
               |
               |  {set of DDP packets}
               v
       +-+-+-+-+-+-+-+-+
       |               |
       | outer decoder | destination node
       |               |
       +-+-+-+-+-+-+-+-+
               |
               |  {set of source packets}
               v
```

Figure 1: A network model for data delivery.

At the source node, the DDP first processes the data to be delivered into a number of source packets each of the same number of bits (see details in Section 2.2.1), and then provides all the source packets to the outer encoder. The outer encoder will further generate a sequence of batches, each consisting of a fixed number of coded packets (see the description in Section 2.2.2).

Each batch generated at the source node is further processed by the recoder separately. The recoder may generate a number of new coded packets using the existing coded packets of the batch (see the description in Section 2.2.3). After processed by the recoder, the DDP forms and transmits the DDP packets using the coded packets, together with the corresponding batch information.

We assume that a DDP packet is either correctly received or completely erased during the communication. The DDP extracts the coded packets and the corresponding batch information from its received DDP packets. A recoder is employed at an intermediate node that does not need the data, and generates recoded packets for each batch (see the description in Section 2.2.3). The DDP forms and transmits DDP packets using the recoded packets and the corresponding batch information.

The outer decoder is employed at the destination node that needs the data. The DDP extracts the coded packets and the corresponding batch information from its received DDP packets. The outer decoder tries to recover the transmitted data using the received batches (see the description in Section 2.2.4). The DDP sends the decoded data to the application that needs the data.

2.2. Data Delivery Procedures

Suppose that the DDP has F octets of data for transmission. We describe the procedures of one BATS session for transmitting the F octets. There is a limit on F of a single BATS session. If the total data has more than the limit, the data needs to be transmitted using multiple BATS sessions. The limit on F of a single BATS session depends on the coding parameters to be discussed in this section, and will be calculated at the end of Section 2.4.2.

2.2.1. Source Node Data Partitioning and Padding

The DDP first determines the following parameters:

- Batch size (M): the number of coded packets in a batch generated by the outer encoder.

- Recoding field size (q): the number of elements in the finite field for recoding. q is 2 or 2^8

- BATS payload size (TO): the number of payload octets in a BATS packet, including the coded data and the coefficient vector.

Based on the above parameters, the parameters T, CO and K are calculated as follows:

- CO: the number of octets of a coefficient vector, calculated as CO = ceil(M*log2(q)/8), which is also called the coefficient vector overhead.

- T: the number of data octets of a coded packet, calculated as T = TO - CO.

- K: number of source packets, calculated as K = floor(F/T)+1.

The data MUST be padded to have T*K octets, which will be partitioned into K source packets b[0], ..., b[K-1], each of T octets. In our padding scheme, b[0], ..., b[K-2] are filled with data octets, and b[K-1] is filled with the remaining data octets and padding octets. Let P = K*T-F denote the number of padding octets. We use b[K-1, 0], ..., b[K-1, T-P-1] to denote the T-P source octets and b[K-1, T-P], ..., b[K-1, T-1] to denote the P padding octets in b[K-1], respectively. The padding insertion process is shown in Figure 2.

```
Z = T - P
j = 1
v = 1
Let bl be the last source packet b[K-1]
for i = Z, Z+1, ..., T-1 do
  bl[i] = j
  if i+1 >= v+Z do
    j += 1
    v += j
```

Figure 2: Data Padding Insertion Process

2.2.2. Source Node Outer Code Encoding Procedure

The DDP provides the BATS encoder with the following information:

- Batch size (M): the number of coded packets in a batch.

- Recoding field size (q): the number of elements in the finite field for recoding.

- Maximum degree (MAX_DEG): a positive integer that specifies the largest degree can be used.

- Degree distribution (DD): an unsigned integer array of size MAX_DEG+1. The i-th entry DD[i] is the probability that i is chosen as the degree, where i is between 0 and MAX_DEG.

- A sequence of batch IDs (BID) (j, j = 0, 1, ...).

- Number of source packets (K).

- Packet size (T): the number of octets in a source packet.

- Source packets (b[i], i = 0, 1, ..., K-1).

Using this information, the outer encoder generates M coded packets for each batch ID using the following steps to be described in details at Section 3.2:

- Obtain a degree d by sampling DD. Roughly, the value d is chosen with probability DD[d].

- Choose d source packets uniformly at random from all the K source packets. It is allowed that a source packet is used by mutiple batches.

- Generate M coded packets using the d source packets.

The DDP receives from the outer encoder a sequence of batches, where the batch with ID j has

- M coded packets (x[j,i], i =0, 1, ..., M-1), each containing TO octets.

The DDP will use the batches to form DDP packets to be transmitted to other network nodes towards the destination nodes. The DDP MUST deliver with each coded packet with its batch ID, which will be further used by both recoder and decoder.

The DDP MUST deliver some of the information used by the encoder to each recoder and the decoder, either embedded in the DDP packets or transmitted using separated protocol messages: For recoders, the DDP MUST deliver the following information to each recoder:

- M: batch size

- q: recoding field size.

For the decoder, the DDP MUST deliver the following information to the decoder:

* M: batch size

* q: recoding field size

* K: the number of source packets

* T: the number of octets in a source packet

* DD: the degree distribution.

The BID is used by both recoders and decoders. We will illustrate in Section 2.4 that how to embed BID, M, q, and K into DDP packets. The degree distribution DD does not need to be changed frequently. See Section 6 in [Yang17] about how to design a good degree distribution. Once designed, the degree distribution can be shared between the source node and the destination node by the DDP, which is not further discussed here.

2.2.3. Recoding Procedures

Both the source node and the intermediate nodes perform recoding on the batches before transmission. At the source node, the recoder receives the batches from the outer code encoding procedure. At an intermediate node, the DDP receives the DDP packets from the other network nodes. If the DDP choose not to recode, it just forwards the DDP packets it received. Otherwise, the DDP should be able to extract coded packets and the corresponding batch information from these packets.

For a received batch, the DDP determines a positive integer Mr, the number of recoded packets to be transmitted for the batch, and provides the recoder with the following information:

* the batch size M,

* the recoding field size q,

* a number of received coded packets of the same batch, each containing TO octets, and

* the number of recoded packets to be generated (Mr).

The recoder uses the information provided by the DDP to generate Mr recoded packets, each containing TO octets, to be described in Section 3.3. The DDP uses the Mr recoded packets to form the DDP packets for transmitting.

2.2.4. Destination Node Procedures

At the destination node, the DDP receives DDP packets from an intermediate network node, and should be able to extract coded packets and the corresponding batch information from these packets. The DDP then employs an outer decoder to recover the data transmitted by the source node.

The DDP provides the outer decoder (to be described in Section 3.4) with the following information:

- M: batch size,

- q: recoding field size,

- K: the number of source packets

- T: the number of octets of a source packet

- A sequence of batches, each of which is formed by a number of coded packets belonging to the same batch, with their corresponding BIDs.

The decoder uses this information to decode the outer code and the inner code jointly and recover the K source packets (see details in Section 3.4). If successful, the decoder returns the recovered K source packets to the DDP, which will use the K source packets to form the F octets data. The recommended padding deletion process is shown as follows:

```
// this procedure returns the number P of padding octets
// at the end of b[K-1]
Let bl be the last decoded source packet b[K-1]
PL = bl[T-1]
if PL == 1 do
    return P = 1
WI = T - 1
while bl[WI] == PL do
    WI = WI - 1
return P = (1 + bl[WI]) * bl[WI] + T - WI - 1
```

Figure 3: Data Padding Deletion Process

2.3. Recommendation for the Parameters

The recommendation for the parameters M and q is shown as follows:

* When q=2, M=16,32,64,128

* When q=256, M=4,8,16,32

It is RECOMMENDED that K is at least 128. The encoder/decoder SHALL support an arbitrary positive integer value less than 2^16. However, the BATS coding scheme to be described is not optimized for small K.

2.4. Coding Parameters in DDP Packets

Here we provide an example of embedding the aforementioned BATS coding parameters into the DDP packets which will be used for recoding and decoding. A DDP can form a DDP packet using a coded packet by adding necessary information that can help to deliver the DPP packet to the next node, e.g., the DDP protocol version, addresses and session identifiers. We will not go into the details of formatting these fields in a DDP packet, but focus on how to format the coding parameters and the coded packet in a DDP packet.

2.4.1. Coding Parameter Format

Here we provide an example of using 32 bits (4 octets) to embed the parameters K, M, q, and BID. The 32 bits are separated into three subfields, denoted as K, Mq and BID, respectively, as illustrated in Figure 4.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                K              | Mq  |         BID             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Coding parameter field format.

* K: 16-bit unsigned integer, specifying the number of source packets of the BATS session.

* Mq: 3-bit unsigned integer to specify the value of M and q as Table 1.

* BID: 13-bit unsigned integer, specifying the batch ID of the batch the packet belongs to.

Mq	M	q
000	16	2
010	32	2
100	64	2
110	128	2
001	4	256
011	8	256
101	16	256
111	32	256

Table 1: Values of Mq field

The choice of the coding parameters depends on the computation cost, the network conditions and the expected end-to-end coding performance. Usually, a larger batch size M will have a better coding performance, but higher computation cost for encoding, recoding and decoding. The field size q affects the coefficient vector overhead, and also the computation cost for recoding. Within a BATS session, the BID field should be different for all batches, and hence the maximum number of batches can be generated for the outer encoder is 2^13. For different BATS sessions, batches can use the same BID.

2.4.2. Coded Packet Format

```
             CO                            T
+-----------------------+-------------------------------+
|   coefficient vector  |          coded data           |
+-----------------------+-------------------------------+
```

Figure 5: Code packet format in a DDP packet.

A coded packet has TO=T+CO octets, where the first CO octets contain the coefficient vector and the remaining T octets contain the coded data.

- coefficient vector: CO = M*log2(q)/8 octets. For the values of M and q in Table 1, CO is at most 32 octets when q is 256 and 6 octets when q is 2.

- coded data: T octets. T should be chosen so that the whole DDP packet is at most PMTU.

Using the above formation, we can calculate the largest file size F for different parameters. For example, when q=2 and M=128, we have CO = 6 octets. Counting the 4 octets for embedding the coding parameters, we can choose T = PMTU-H-10, where H is the header length of a DDP packet. As K can be at most 2^16-1, F can be at most (PMTU-H-10)(2^16-1) octets. In this case, K/M is about 2^9 and the BID field allows transmitting 2^4*K/M batches.

3. BATS Code Specification

3.1. Common Parts

The T octets of a source packets are treated as a column vector of T elements in GF(256). The CO octets of coefficient vector are treated as a column vector of CO elements in GF(q), where q=2 or q=256. Linear algebra and matrix operations over finite fields are assumed in this section.

For the two elements of GF(2), their multiplication corresponds to a logical AND operation and their addition is a logical XOR operation. An element of the field GF(256) can be represented by a polynomial with binary coefficients and degree lower or equal to 7. The addition between two elements of GF(256) is defined as the addition of the two binary polynomials. The multiplication between two elements of GF(256) is the multiplication of the two binary polynomials modulo a certain irreducible polynomial of degree 8, called a primitive polynomial. One example of such a primitive polynomial for GF(256) is:

$$x^8 + x^4 + x^3 + x^2 + 1$$

A common primitive polynomial MUST be specified for all the finite field multiplications over GF(256). Note that a binary polynomial of degree less than 8 can be represented by a binary sequence of 8 bits, i.e., an octet.

Suppose that a pseudorandom number generator Rand() which generates an unsigned integer of 32 bits is shared by both encoding and decoding. The pseudorandom generator can be initialized by Rand_Init(S) with seed S. When S is not provided, the pseudorandom generator is initialized arbitrarily. One example of such a pseudorandom generator is defined in RFC 8682 [RFC8682].

A function called BatchSampler is used in both encoding and decoding. The function takes two integers j and d as input, and generates an array idx of d integers and a d x M matrix G. The function first initializes the pseudorandom generator with j, sample d distinct integers from 0 to K-1 as idx, and sample d*M integers from 0 to 255 as G. See the pseudocode in Figure 6.

```
function BatchSampler(j,d)
    // initialize the pseudorandom generator by seed j.
    Rand_Init(j)
    // sample d distinct integers between 0 and K-1.
    for k = 0, ..., d-1 do
        r = Rand() % K
        while r already exists in idx do
            r = Rand() % K
        idx[k] = r

    // sample d x M matrix
    for r = 0, ..., d-1 do
        for c = 0,...,M-1 do
            G[r,c] = Rand() % 256

    return idx, G
```

Figure 6: Batch Sampler Function

3.2. Outer Code Encoder

Define a function called DegreeSampler that returns an integer d using the degree distribution DD. We expect that the empirical distribution of the returning d converges to DD(d) when d < K. One design of DegreeSampler is illustrated in Figure 7. Note that when K < MAX_DEG, the degree value returned by DegreeSampler does not exactly follow the distribution DD, which however would not affect the practical decoding performance for the outer decoder to be described in Section 3.4.

```
function DegreeSampler(j, DD)
    Let CDF be an array
    CDF[0] = 0
    for i = 1, ..., MAX_DEG do
        CDF[i] = CDF[i-1] + DD[i]
    Rand_Init(j)
    r = Rand() % CDF[MAX_DEG]
    for d = 1, ..., MAX_DEG do
        if r >= CDF[d] do
            return min(d,K)
    return min(MAX_DEG,K)
```

Figure 7: Degree Sampler Function.

Let b[0], b[1], ..., b[K-1] be the K source packets. A batch with BID j is generated using the following steps.

- Obtain a degree d by calling DegreeSampler with input j.

- Obtain idx and G[j] by calling BatchSampler with input j and d.

- Let B[j] = (b[idx[0]], b[idx[1]], ..., b[idx[d-1]]). Form the batch X[j] = B[j]*G[j], whose dimension is T x M.

- Form the TO x M matrix Xr[j], where the first CO rows of Xr[j] form the M x M identity matrix I with entries in GF(q), and the last T rows of Xr[j] is X[j].

See the pseudocode of the batch generating process in Figure 8.

```
function GenBatch(j)
    d = DegreeSampler(j)
    (idx, G) = BatchSampler(j,d)
    B = (b[idx[0]], b[idx[i]], ..., b[idx[d-1]])
    X = B * G
    Xr = [I; X]
    return Xr
```

Figure 8: Batch Generation Function.

3.3. Inner Code Encoder (Recoder)

In general, the inner code of a BATS code comprises (random) linear network coding applied on the coded packets belonging to the same batch. The recoded packets have the same BID. Suppose that coded packets xr[i], i = 0, 1, ..., r-1, which have the same BID j, have been received at an intermediate node, and Mr recoded packets are supposed to be generated. Following traditional random linear network coding, a recoded packet can be generated by random linear combination: (randomly) choose a sequence of coefficients c[i], i = 0, 1, ..., r-1 from GF(q), and generate c[0]xr[0]+c[1]xr[1]+...+c[r-1]xr[r-1] as a recoded packet. This recoding approach, called random linear recoding, achieves good network coding performance for multicast when the batch size is sufficiently large.

For unicast communications in a single path as illustrated in Figure 1, it is not necessary to generate all the Mr recoded packets using random linear combination. Instead, xr[i], i = 0, 1, ..., r-1, are directly used as recoded packets, and max(Mr-r,0) recoded packets are generated using linear combinations. Compared with random linear recoding, this recoding approach, called systematic recoding, can reduce both the computation cost and also the recoding latency that accumulates linearly with the number of nodes. Note that the use of systematic recoding may not always achieve the optimal network coding performance as random linear recoding in more complicated communication scenarios that include multiple paths and multiple destination nodes.

3.4. Outer Decoder

The decoder receives a sequence of batches Yr[j], j = 0, 1, ..., n-1, each of which is a TO-row matrix over GF(256). Let Y[j] be the submatrix of the last T rows of Yr[j]. When q = 256, let H[j] be the first M rows of Yr[j]; when q = 2, let H[j] be the matrix over GF(256) formed by embedding each bit in the first M/8 rows of Yr[j] into GF(256). For successful decoding, we require that the total rank of all the batches is at least K.

The same degree distribution DD used for the outer encoder is supposed to be known by the outer decoder. By calling DegreeSampler and BatchSampler with input j, we obtain d[j], idx[j] and G[j]. According to the encoding and recoding processes described in Section 3.2 and Section 3.3, we have the system of linear equations Y[j] = B[j]G[j]H[j] for each received batch with ID j, where B[j] = (b[idx[j,0]], b[idx[j,1]], ..., b[idx[j,d-1]]) is unknown.

We first describe a belief propagation (BP) decoder that can efficiently solve the source packets when a sufficient number of batches have been received. A batch j is said to be decodable if rank(G[j]H[j]) = d[j] (i.e., the system of linear equations Y[j] = B[j]G[j]H[j] with B[j] as the variable matrix has a unique solution). The BP decoding algorithm has multiple iterations. Each iteration is formed by the following steps:

* Decoding step: Find a batch j that is decodable. Solve the corresponding system of linear equations Y[j] = B[j]G[j]H[j] and decode B[j].

* Substitution step: Substitute the decoded source packets into undecodable batches. Suppose that a decoded source packet b[k] is used in generating an undecodable Y[j]. The substitution involves 1) removing the entry in idx[j] corresponding to k, 2) removing the row in G[j] corresponding to b[k], and 3) reducing d[j] by 1.

The BP decoder repeats the above steps until no batches are decodable during the decoding step.

When the degree distribution DD in the outer code encoder (see Section 3.2) is properly designed, the BP decoder guarantees a high probability for the recovery of a given fraction of the source packets when K is large. To recover all the source packets, a precode can be applied to the source packets to generate a fraction of redundant packets before applying the outer code encoding. Moreover, when the BP decoder stops which may happen with a high probability when K is relatively small, it is possible to continue with inactivation decoding, where certain source packets are treated inactive so that a similar belief propagation process can be resumed. The reader is referred to RFC 6330 [RFC6330] for the design of a precode with a good inactivation decoding performance.

4. Research Issues

The baseline BATS coding scheme described in Section 2 and Section 3 needs various refinement and complement towards becoming a more sophisticated network communication application. Various related research issues are discussed in this section, but the security related issues are left to Section 6.

4.1. Coding Design Issues

When the number of batches is sufficiently large, the BATS code specification in Section 3 has nearly optimal performance in the sense that the decoding can be successful with a high probability when the total rank of all the batches used for decoding is just slightly larger than the number of source packet K. But when K is small, the degree sampler function in Figure 7 and the BatchSampler function in Figure 6 based on a pseudorandom generator may not sample all the source packets evenly, so that some of the source packets are not well protected. One approach to solve this issue is to generate a deterministic degree sequence when the number of batches is relatively small, and design a special pseudorandom generator that has a good sampling performance when K is small.

There are research issues related to recoding discussed in Section 3.3. One question is how many recoded packets to generate for each batch. Though it is asymptotically optimal when using the same number of recoded packets for all batches, it has been shown that transmitting a different number of recoded packets for different batches can improve the recoding efficiency. The intuition is that for a batch with a lower rank, a smaller number of recoded packets need to be transmitted. This kind of recoding scheme is called adaptive recoding [Yin19].

Packet loss in network communication is usually bursty, which may harm the recoding performance. One way to resolve this issue is to transmit the packets of different batches in a mixed order, which is also called batch interleaving [Yin20]. How to efficiently interleave batches without increasing end-to-end latency too much is a research issue.

Though we only focus on the BATS coding scheme with one source node and one destination node, a BATS coding scheme can be used for multiple source and destination nodes. To benefit from multiple source nodes, we would need different source nodes to generate statistically independent batches. It is well-known that linear network coding [Li03] achieves the multicast capacity. BATS codes can benefit from network coding due to its inner code. For multicast, each destination node performs independently the same operations as described in this document, but the inner code should be improved to taking multiple destination node into consideration. How to efficiently implement multicast needs further research.

4.2. Protocol Design Issues

The baseline scheme in this document focuses on reliable communication. There are other issues to be considered towards designing a fully functional DDP based on a BATS coding scheme. Here we discuss some network management issues that are closely related to a BATS coding scheme: routing, congestion control and media access control.

The outer code of a BATS code can be regarded as a channel code for the channel induced by the inner code, and hence the network management algorithms should try to maximize the capacity of the channel induced by the inner code. A network utility maximization problem [Dong20] for BATS coding can be applied to study routing, congestion control and media access control jointly. Compared with the network utility maximization for Internet, there are two major differences. First, the network flow rate is not measured by the rate of the raw packets. Instead, a rank based measurement induced by the inner code is applied for BATS coding schemes. Second, due to recoding, the raw packet rate may not be the same for different links of a flow, i.e., no flow conservation for BATS coding schemes. These differences affect both the objective and the constraints of the utility maximization problem.

Practical congestion control, routing and media access control algorithms for BATS coding schemes deserve more research efforts. The rate of transmitting batches can be controlled end-to-end. Due to the recoding operation, however, congestion control cannot be only performed end-to-end. The number of recoded packets generated for a batch must be controlled at the intermediate nodes, which introduces new research issues for congestion control. A BATS coding scheme is an extension of forward erasure correction coding. See RFC 9265 [RFC9265] for more discussion of forward erasure correction coding and congestion control.

For routing, the BATS coding scheme is flexible for implementing data transmission on multiple paths simultaneously. For unicast, it is optimal to transmit all the packets of a batch on the same path between the source node and the destination node, and for multicast, network coding gain can be obtained by transmitting packets of the same batch on different paths [Yang17]. Last, under the scenario of BATS coding schemes, media access control can have some different considerations: Retransmission is not necessary, and a reasonably high packet loss rate can be tolerated.

4.3. Usage Scenario Considerations

There are more research issues pertaining to various usage scenarios. The reliable communication technique provided by BATS codes can be used for a broad range of network communication scenarios. In general, a BATS coding scheme is suitable for data delivery in networks with multiple hops and unreliable links.

One class of typical usage scenario is wireless mesh and ad hoc networks [Toh02], including vehicular networks, wireless sensor networks, smart lamppost networks, etc. These networks are characterized by a large number of network devices connected wirelessly with each other without a centralized network infrastructure. A BATS coding scheme is suitable for high data load delivery in such networks without the requirement that the point-to-point/one-hop communication is highly reliable. Therefore, employing a BATS coding scheme can provide more freedom for media access control, including power control, and physical-layer design so that the overall network throughput can be improved.

Another typical usage scenario of BATS coding schemes is underwater acoustic networks [Sprea19], where the propagation delay of acoustic waves in underwater can be as long as several seconds. Due to the long delay, feedback based mechanisms become inefficient. Moreover, point-to-point/one-hop underwater acoustic communication (for both the forward and reverse directions) is highly unreliable. Due to these reasons, traditional networking techniques developed for radio and wireline networks cannot be directly applied to underwater networks. As a BATS coding scheme does not rely on the feedback for reliability communication and can tolerate highly unreliable links, it makes a good candidate for developing data delivery protocols for underwater acoustic networks.

Last but not least, due to its capability of performing multi-source multi-destination communications, a BATS coding scheme can be applied in various content distribution scenarios. For example, a BATS coding scheme can be a candidate for the erasure code used in the liquid data networking framework [Byers20] of CCN (content centric networking), and provides the extra benefit of network coding [Zhang16].

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

Subsuming both random linear network codes (RLNC) and fountain codes, BATS codes naturally inherit both their desirable security capability of preventing eavesdropping, as well as their vulnerability towards pollution attacks. In this section, we discuss some related research issues.

6.1. Preventing Eavesdropping

Suppose that an eavesdropper obtains a batch where the degree value d is strictly larger than the batch size M. Even if the eavesdropper has all the related encoding information, the system of linear equations related to this batch does not have a unique solution, and the probability that the eavesdropper can guess the d source packets used for encoding the batch correctly is 2^(-(d-M)T)<=2^(-T), where T is the number of octets of a source packet (see also [Bhattad07]). When inactivation decoding is applied, we can design the degree distribution DD so that the smallest degree is M+1, and hence prevent the eavesdropper from decoding source packets from individual batches.

If we allow the eavesdropper to collect multiple batches and use inactivation decoding, the same security holds if the total rank of all the batches collected by the eavesdropper is less than the number of source packet. Therefore, if the DDP can manage to restrict the eavesdropper from collecting a sufficient number of coded packets, the native security of BATS code is effective when T is sufficiently large. Here by native security, we mean the security protection provided by the BATS coding scheme without extra enhancement.

If the eavesdropper can collect a sufficient number of coded packets for correctly decoding, the native security of BATS code is ineffective. One solution in this case is to encrypt the whole data before using the BATS code scheme. Better schemes are desired towards reducing the computation cost of the whole data encryption solution. This is a research issue that depends on specific BATS code schemes, and will not be further discussed here.

The threat exists for eavesdropping on the initial encoding process, which takes place at the encoding nodes. In these nodes, the transported data are presented in plain text and can be read along their transfer paths. Hence, information isolation between the encoding process and all other user processes running on the source node MUST be assured.

In addition, the authenticity and trustworthiness of the encoding, recoding and decoding program running on all the nodes MUST be attested by a trusted authority. Such a measure is also necessary in countering pollution attacks.

6.2. Privacy-Preserving against Traffic Analysis

A security issue closely related to eavesdropping is traffic analysis. Even when eavesdropping is prevented, tracking the traffic flow patterns can help an attacker to know a certain information about the communication. Preventing traffic analysis is critical for communications that need to be anonymous. In [Fan09], a homomorphic encryption based approach is proposed for network coding to prevent traffic analysis. However, homomorphic encryption could be too computationally expensive for practical applications and cannot help with the traffic analysis by monitoring the frequency and timing of network traffic.

The network traffic using network coding does not necessarily satisfy the flow conservation property, and hence network coding can be used as a tool for defeating traffic analysis. For example, redundant network traffic can be generated by network coding to make it harder for an attacker to learn the true communication. Moreover, traffic analysis countermeasures can benefit from multipath communication

[Yang15], and network coding makes multiple-path communication more flexible and efficient. Therefore, using network coding brings new research issues for both traffic analysis and its countermeasure.

6.3. Countermeasures against Pollution Attacks

Like all network codes, BATS codes are vulnerable to pollution attacks. In these attacks, one or more compromised coding node(s) can pollute the coded messages by injecting forged packets into the network and thus prevent the receivers from recovering the transported data correctly. Moreover, a small number of polluted packets can infect a large number of packets by recoding and decoding [Zhao07].

The research community has long been investigating the use of homomorphic signatures to identify the forged packets and stall the attacks (see [Zhao07], [Yu08], [Agrawal09]). In these schemes, the source node attaches a signature to each packet to transmit, and the signature is allowed to be processed by network coding same as the payload. All the intermediate nodes and the destination node can verify the signature attached to a received packet. However, these countermeasures are regarded as being too computationally expensive to be employed in broadband communications.

A system-level approach based on trusted computing [TC-Wikipedia] can provide an alternative to protect BATS codes against pollution attacks. Trusted computing consists of software and hardware technologies so that a computer behaves as expected. Suppose that all the network nodes employ trusted computing. Two nodes will first gain trust with each other, and then negotiate an authentication method for exchanging the coded packets of the BATS coding scheme. A network node would not use any packets received from other nodes without trust to avoid the pollution attack.

7. References

7.1. Normative References

[RFC2119] Bradner, S. and RFC Publisher, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8406] Adamson, B., Adjih, C., Bilbao, J., Firoiu, V., Fitzek, F., Ghanem, S., Lochin, E., Masucci, A., Montpetit, M., Pedersen, M., Peralta, G., Roca, V., Ed., Saxena, P., Sivakumar, S., and RFC Publisher, "Taxonomy of Coding Techniques for Efficient Network Communications", RFC 8406, DOI 10.17487/RFC8406, June 2018, <https://www.rfc-editor.org/info/rfc8406>.

[RFC8682] Saito, M., Matsumoto, M., Roca, V., Ed., Baccelli, E., and RFC Publisher, "TinyMT32 Pseudorandom Number Generator (PRNG)", RFC 8682, DOI 10.17487/RFC8682, January 2020, <https://www.rfc-editor.org/info/rfc8682>.

7.2. Informative References

[Agrawal09] Agrawal, S. and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding", International Conference on Applied Cryptography and Network Security , 2009.

[Bhattad07] Bhattad, K. and K.R. Narayanan, "An efficient privacy-preserving scheme against traffic analysis attacks in network coding", ISIT , 2007.

[Byers20] Byers, J.W. and M. Luby, "Liquid Data Networking", ICN , 2020.

[Dong20] Dong, Y., Jin, S., Yang, S., and H.H.F. Yin, "Network Utility Maximization for BATS Code enabled Multihop Wireless Networks", ICC , 2020.

[Fan09] Yanfei, Y., Yixin, Y., Haojin, H., and X. Sherman, "Weakly Secure Network Coding", INFOCOM , 2009.

[Li03] Li, S.-Y.R., Yeung, R.W., and N. Cai, "Linear Network Coding", IEEE Transactions on Information Theory , 2003.

[RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., Minder, L., and RFC Publisher, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, DOI 10.17487/RFC6330, August 2011, <https://www.rfc-editor.org/info/rfc6330>.

[RFC9265] Kuhn, N., Lochin, E., Michel, F., Welzl, M., and RFC Publisher, "Forward Erasure Correction (FEC) Coding and Congestion Control in Transport", RFC 9265, DOI 10.17487/RFC9265, July 2022, <https://www.rfc-editor.org/info/rfc9265>.

[Sprea19] Sprea, N., Bashir, M., Truhachev, D., Srinivas, K.V., Schlegel, C., and C. Claudio Sacchi, "BATS Coding for Underwater Acoustic Communication Networks", OCEANS , 2019.

[TC-Wikipedia] "Trusted Computing", Wikipedia https://en.wikipedia.org/wiki/Trusted_Computing.

[Toh02] Toh, C.K., "Ad Hoc Mobile Wireless Networks", Prentice Hall Publishers , 2002.

[Yang14] Yang, S. and R.W. Yeung, "Batched Sparse Codes", IEEE Transactions on Information Theory 60(9), 5322-5346, 2014.

[Yang15] Yang, L. and F. Fengjun, "mTor: A multipath Tor routing beyond bandwidth throttling", NCS , 2015.

[Yang17] Yang, S. and R.W. Yeung, "BATS Codes: Theory and Practice", Morgan & Claypool Publishers , 2017.

[Yin19] Yin, H.H.F., Tang, B., Ng, K.H., Yang, S., Wang, X., and Q. Zhou, "A Unified Adaptive Recoding Framework for Batched Network Coding", ISIT , 2019.

[Yin20] Yin, H.H.F., Yeung, R.W., and S. Yang, "A Protocol Design Paradigm for Batched Sparse Codes", Entroy , 2020.

[Yu08] Yu, Z., Wei, Y., Ramkumar, B., and Y. Guan, "An Efficient Signature-Based Scheme for Securing Network Coding Against Pollution Attacks", INFOCOM , 2008.

[Zhang16] Zhang, G. and Z. Xu, "Combing CCN with network coding: An architectural perspective", Computer Networks , 2016.

[Zhao07] Zhao, F., Kalker, T., Medard, M., and K.J. Han, "Signatures for content distribution with network coding", ISIT , 2007.

Acknowledgments

The authors would like to thank the NWCRG chairs, Vincent Roca (our shepherd) and Marie-Jose Montpetit; and all those who provided comments -- namely (in alphabetical order), Emmanuel Lochin, David Oran, and Colin Perkins.

Authors' Addresses

Shenghao Yang
CUHK(SZ)
Shenzhen
Guangdong,
China
Phone: +86 755 8427 3827
Email: shyang@cuhk.edu.cn

Xuan Huang
CUHK
Hong Kong
Hong Kong SAR,
China
Phone: +852 3943 8375
Email: 1155136647@link.cuhk.edu.hk

Raymond W. Yeung
CUHK
Hong Kong
Hong Kong SAR,
China
Phone: +852 3943 8375
Email: whyeung@ie.cuhk.edu.hk

John K. Zao
NCTU
Hsinchu
Taiwan,
China
Email: jkzao@ieee.org