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

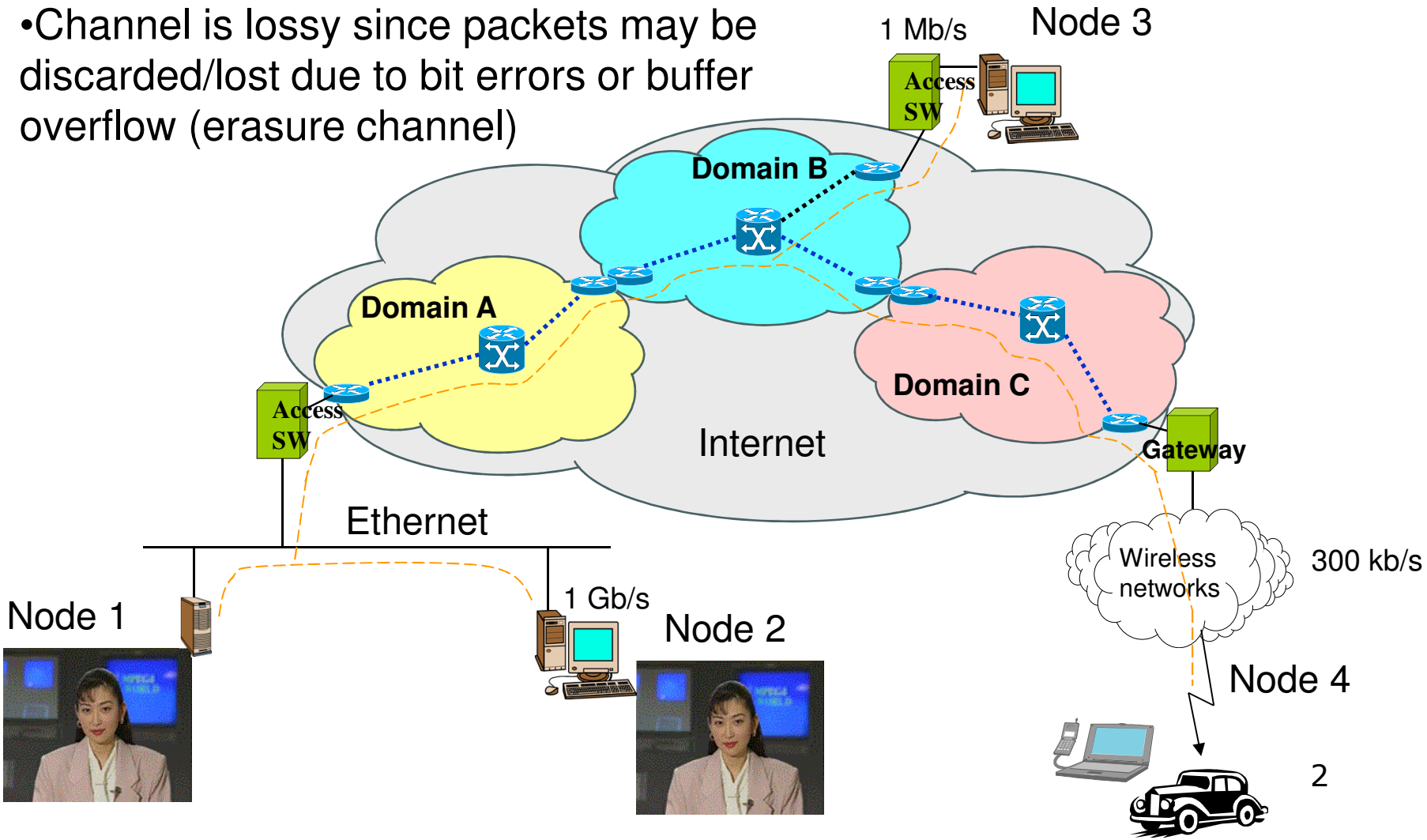
Dapeng Oliver Wu

Electrical & Computer Engineering

University of Florida

Loss-Tolerant Information Spreading

- Spread of microblog or TV to many people
- Channel is lossy since packets may be discarded/lost due to bit errors or buffer overflow (erasure channel)

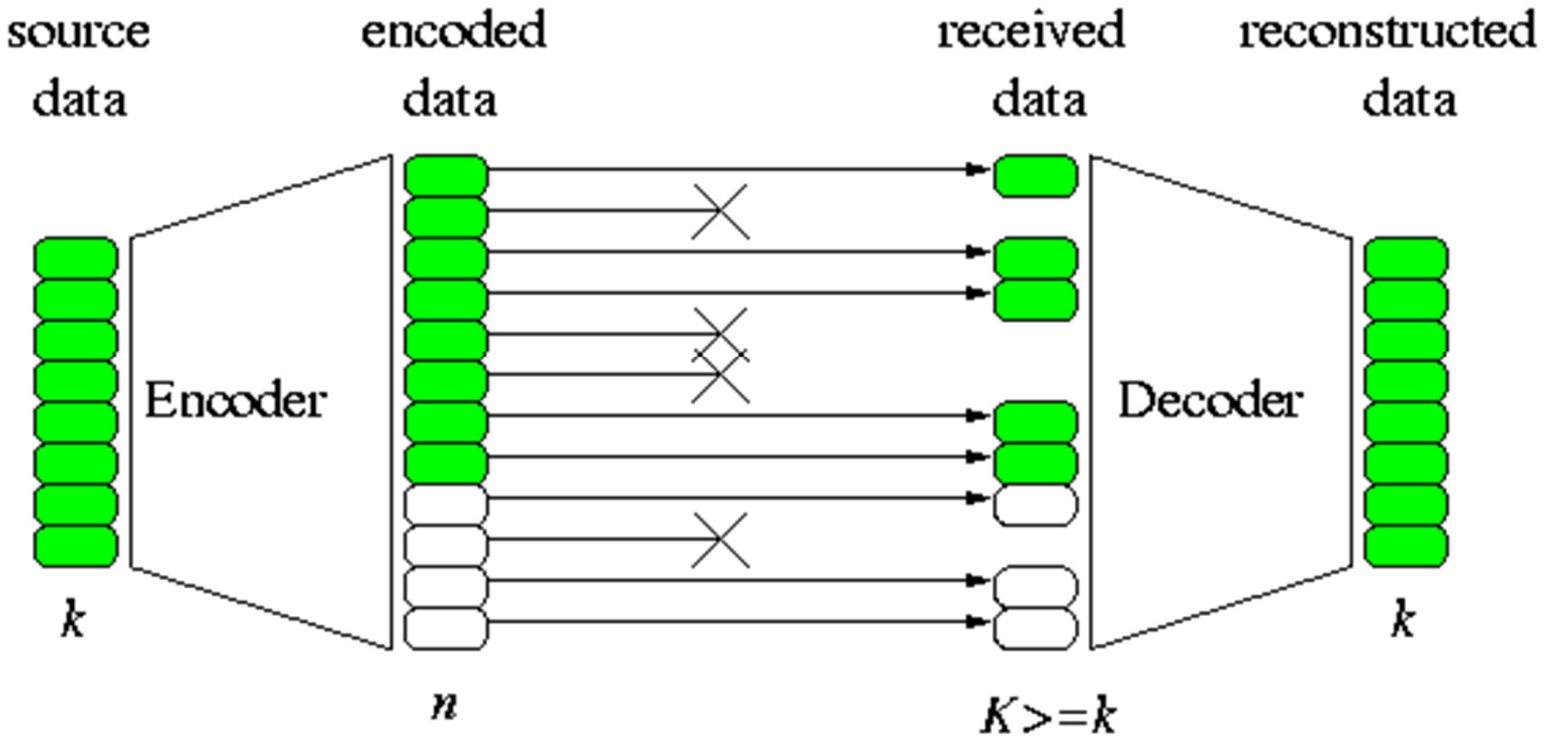


Outline

- Review of existing approaches for erasure channels
- FUN architecture and overview
- Description of FUN coding
- Experimental results
- Conclusion

Erasure Channel and Erasure Channel Coding

- An erasure channel only induces packet loss but no packet error.



Erasure Codes

- Linear erasure codes
  - The coded packets are generated by linearly combining the native packets with coefficients from a finite field
- Nonlinear erasure codes
  - A nonlinear erasure code can reduce the computational complexity by using only binary addition and shift operations instead of more complicated finite field multiplications as in a linear erasure code.
  - Hence, nonlinear erasure codes are particularly suited for mobile phone applications, which require low computational complexity and low power consumption.

Erasure Codes

- Under an erasure code, a receiver can recover K native packets from n coded packets (received by the receiver), where $n = (1 + \varepsilon)K$ and ε can be very small, e.g., ε can be as small 10^{-6} for RQ codes.
- To recover the K native packets, it does not matter which packets the receiver has received; as long as it has received any K linearly independent packets, the receiver is able to decode the K native packets.
- Erasure codes include Reed-Solomon codes, LDPC codes, and fountain codes.

Fountain Codes

- An erasure code can be classified as a fountain code if it has the following properties:
  - (Ratelessness) The number of coded packets that can be generated from a given set of native packets should be sufficiently large. The reason why this code is called fountain code is because the encoder generates essentially unlimited supply of codewords, in analogy to a water fountain, which produces unlimited drops of water.
  - (Efficiency and flexibility) Irrespective of which packets that the receiver has received, the receiver should be able to decode K native packets using any K linearly independent received coded packets.
  - (Linear complexity) The encoding and decoding computation cost should be a linear function of the number of native packets K.

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.
- Network coding techniques can be classified into two categories:
  - intra-session (where coding is restricted to the same multicast or unicast session)
  - inter-session (where coding is applied to packets of different sessions)

Cross-next-hop Network Coding (1)

- For wireless communication, cross-next-hop network coding and intra-session network coding 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 (2)

- Cross-next-hop network coding uses per-next-hop queueing at each relay node while inter-session network coding may use per-flow queueing at each relay node or add a very large global encoding vector to the header of each coded packet.
- Hence, cross-next-hop network coding is more scalable than a general inter-session network coding.
- As such, for core routers, it is desirable to use cross-next-hop network coding instead of a general inter-session network coding.

Cross-next-hop Network Coding (3)

- Cross-next-hop network coding has been heavily studied in the wireless networking area.
- Major works include
  - COPE: 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.
  - CLONE: 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.

Joint Erasure Coding and Intra-Session Network Coding (JEN)

- JEN works as below:
  - 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.

Practical JEN Approach

- In practice, under JEN, the data to be transmitted is partitioned into multiple segments, or generations, or blocks, or batches, 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.
- Silva et al. proposed a network coding technique with overlapping segments to improve the performance of JEN with non-overlapping segments. This technique intends to combine network coding with a fountain code.

BATched Sparse (BATS) Codes

- To combine the best features of JEN and fountain codes and strike a balance between the two approaches, Yang and Yeung proposed BATS codes.
- A BATS code consists of an inner code and an outer code.
  - The outer code is a matrix generalization of a fountain code. At a source node, the outer code encoder encodes native packets into batches, each of which contains M packets. When the batch size M is equal to 1, the outer code reduces to a fountain code.
  - The inner code is an RLNC performed at each relay node. At each relay node, RLNC is applied only to the packets within the same batch of the same flow; hence the structure of the outer code is preserved.

Outline

- Review of existing approaches for erasure channels
- FUN architecture and overview
- Description of FUN coding
- Experimental results
- Conclusion

FUN Architecture

- We propose 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 & FUN-2 Schemes

- FUN-1 basically combines BATS coding with COPE for two flows.
  - But FUN-1 is not a simple combination of BATS and COPE; a relay node needs local encoding vectors 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.

FUN-1

- 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.
  - Layer 2.2 is for BATS coding.
- 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.

FUN-1 (cont'd)

- 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.

FUN-2

- 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.

Restriction on Number of Flows for Joint Coding

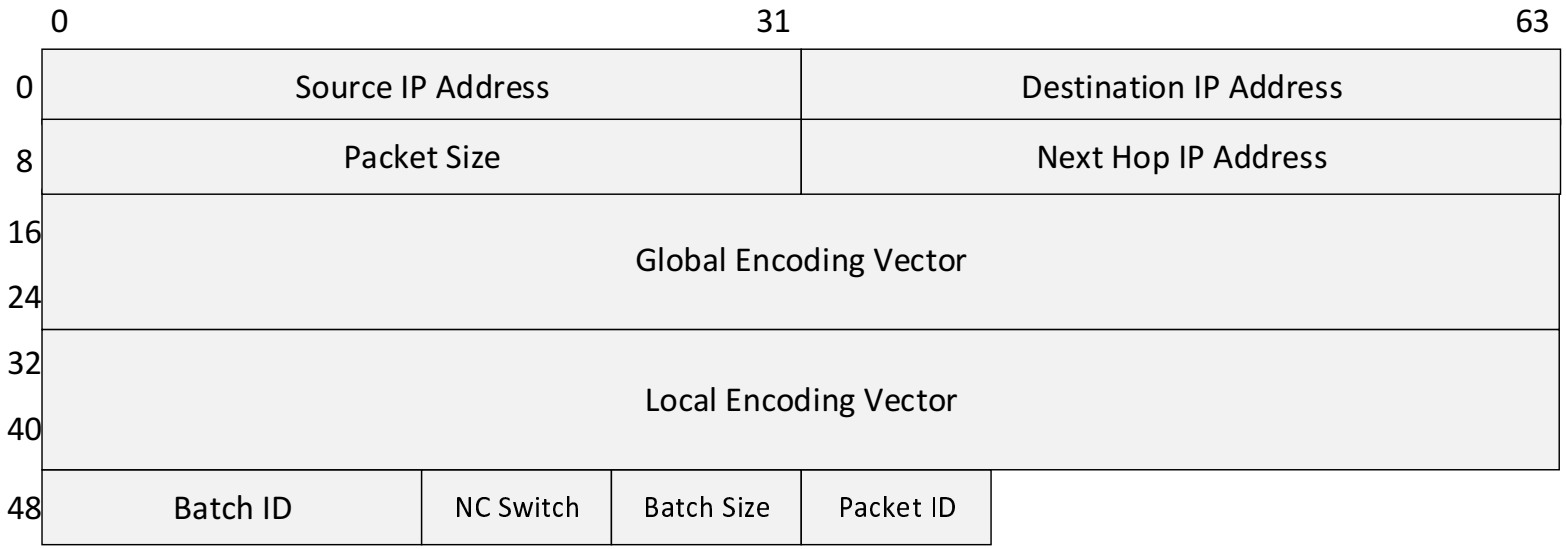
- In the current version, both FUN-1 and FUN-2 are restricted to two flows, i.e., forward flow and backward flow between two nodes.
- The advantage is that there is no need for co-ordination while a higher coding gain can be achieved.
- The limitation is that it restricts its use to two flows between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

Structure of a FUN-1/FUN-2 Packet

- Both FUN-1 packet and FUN-2 packet have two headers as shown below.

- 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.

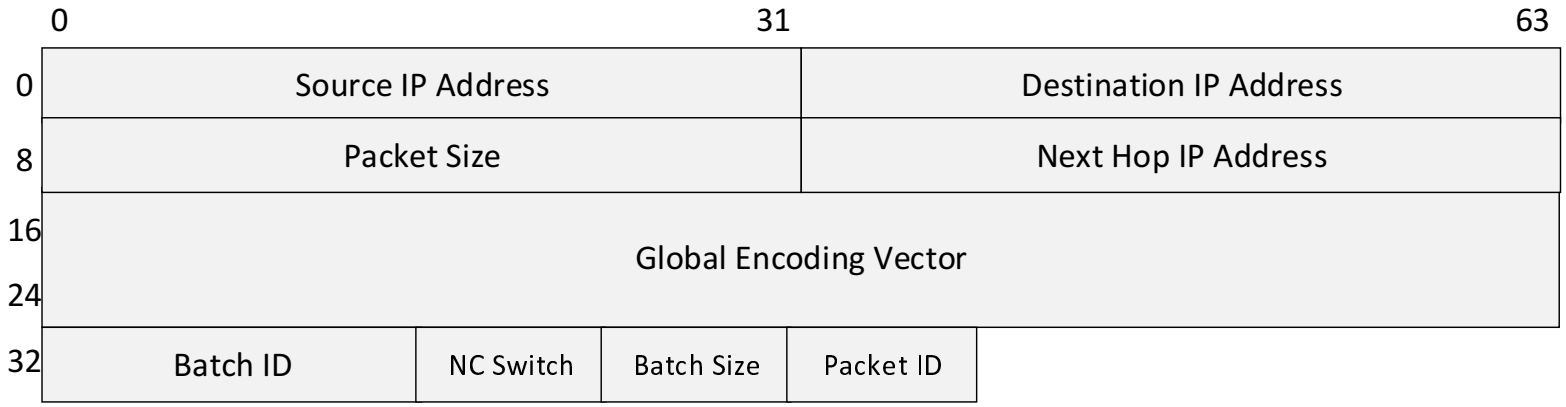
Header Structure of a FUN-1 Packet



NC Switch

- The NC switch consists of two bits and indicates one of the following four schemes is used:
  - FUN-1
    - 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.
  - FUN-2
    - BATS is a special case of FUN-2, where there is no FUN Header 2.
  - RLNC
  - no network coding
    - 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.

Header Structure of a FUN-2 Packet



Outline

- Review of existing approaches for erasure channels
- FUN architecture and overview
- Description of FUN coding
- Experimental results
- Conclusion

FUN-1

- FUN-1 consists of outer code, inner code, XOR coding, and precoding.
- Assume Node A will transmit K native packets to Node B, and Node B will transmit K native packets to Node A. 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 $\mathbb{F}_q^T$. Denote the set of K native packets by the following matrix

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

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.

Outer Code of FUN-1

- 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

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

where $\mathbf{G}_i \in \mathbb{F}_q^{d_i \times M}$ is called the generator matrix of the i-th batch; $\mathbf{B}_i \in \mathbb{F}_q^{T \times d_i}$; $\mathbf{X}_i \in \mathbb{F}_q^{T \times M}$.

Outer Code of FUN-1 (cont'd)

- 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 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.

Inner Code of FUN-1 (1)

- 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

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

where $\mathbf{E}_{i,1}$ is an erasure matrix, representing the erasure channel between the source and Node R_1.

Inner Code of FUN-1 (2)

- $\mathbf{E}_{i,1}$ 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 $\mathbf{X}'_{i,1}$.

Inner Code of FUN-1 (3)

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

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

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.

Inner Code of FUN-1 (4)

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

$$\mathbf{Y}_{i,j} = \mathbf{X}'_{i,j}\mathbf{H}_{i,j} = \mathbf{Y}_{i,j-1}\mathbf{E}_{i,j}\mathbf{H}_{i,j}$$
$$= \mathbf{B}_i\mathbf{G}_i\mathbf{E}_{i,1}\mathbf{H}_{i,1}\cdots\mathbf{E}_{i,j}\mathbf{H}_{i,j},$$

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.

XOR Encoding of FUN-1

Algorithm 1 XOR encoding of FUN-1

while at least one flow (forward or backward flow) is alive **do**
  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**
    for $m = 1, \cdots, M$ **do**
      Pick Packet $y_{i,m}$ at the head of Layer 2.2 output queue for the forward flow;
      Pick packet $\bar{y}_{j,m}$ at the head of Layer 2.2 output queue for the backward flow;
      $p_m = y_{i,m} \oplus \bar{y}_{j,m}$;
      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 $\bar{y}_{j,m}$;
      Enable the bit FUN_XOR in the header of packet p_m, i.e., $FUN\_XOR = 1$;
      Place packet p_m in Layer 2.1 output queue;
    end for
  else
    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**
      for $m = 1, \cdots, M$ **do**
        Pick Packet $y_{i,m}$ at the head of Layer 2.2 output queue of the flow;
        Disable the bit FUN_XOR in the header of packet $y_{i,m}$, i.e., $FUN\_XOR = 0$;
        Place packet $y_{i,m}$ in Layer 2.1 output queue;
      end for
    end if
  end if
end while

XOR Decoding of FUN-1

Algorithm 2 XOR decoding of FUN-1

while at least one flow (forward or backward flow) is alive **do**
  if the head-of-line packet in Layer 2.1 input queue has $FUN\_XOR = 0$ **then**
    Move this head-of-line packet to Layer 2.2 input queue;
  else
    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);
    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**
      while the head-of-line packet, say p_m, has its forward-flow batch ID equal i **do**
        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;
        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;
      end while
    end if
  end if
end while

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 of FUN-1.

FUN-2

- FUN-2 consists of
  - outer code,
  - inner code, and
  - precoding.

Outer Code of FUN-2

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

while not all native packets destined to Node j are decoded **do**
  if the receiving queue of Layer 2.2 is complete **then**
    Pick packets of the same batch from the head of the receiving queue of Layer 2.2;
    if FUN-2 Header 2 is not empty **then**
      if the destination IP address in FUN-2 Header 1 is that of Node j **then**
        Let i equal the batch ID in FUN-2 Header 1 of any picked packet;
        Let k equal the batch ID in FUN-2 Header 2 of any picked packet;
        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;
        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 $1-j$;
      else
        Let i equal the batch ID in FUN-2 Header 2 of any picked packet;
        Let k equal the batch ID in FUN-2 Header 1 of any picked packet;
        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;
        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 $1-j$;
      end if
      Let $\mathbf{Y}_{i,j}$ be a matrix whose columns are the payloads of all the picked packets;
      Compute $\mathbf{Y}_{i,j} = \mathbf{Y}_{i,j} - \mathbf{B}_{k,1-j}\mathbf{H}_{k,1-j}$;
    else
      Let i equal the batch ID in FUN-2 Header 1 of any picked packet;
      Let $\mathbf{Y}_{i,j}$ be a matrix whose columns are the payloads of all the picked packets;
    end if
    Do outer decoding in the same way as BATS outer decoding with input packets $\mathbf{Y}_{i,j}$;
  end if
end while

Inner Code Encoding of FUN-2

Algorithm 4 Inner encoding of FUN-2

for j=0 **to** 1 **do**
  Initialize two buffers for Destination j, which are denoted by $F_{j,new}$ and $F_{j,old}$;
end for
while at least one flow (forward or backward flow) is alive **do**
  if Layer 3 output queue is not empty **then**
    Pick Packet $y_{i,m}$ at the head of Layer 3 output queue; assume the destination of Packet $y_{i,m}$ is j;
    switch (State of Buffers $F_{j,new}$, $F_{j,old}$, $F_{1-j,old}$)
    case Buffer $F_{j,new}$ is not complete:
      Insert $y_{i,m}$ to the m-th position of Buffer $F_{j,new}$;
      if Buffer $F_{j,new}$ is complete **then**
        Goto Step 7;
      end if
    case Buffer $F_{j,new}$ is complete **AND** Buffer $F_{j,old}$ is empty **AND** Buffer $F_{1-j,old}$ is empty:
      Move all the packets in $F_{j,new}$ to $F_{j,old}$;
      if the packets in $F_{j,old}$ were mixed before **then**
        Apply RLNC to all the packets in $F_{j,old}$ and generate M re-coded packets;
        Move the re-coded packets to the unicast output queue of Layer 2;
      end if
    case Buffer $F_{j,new}$ is complete **AND** Buffer $F_{j,old}$ is empty **AND** Buffer $F_{1-j,old}$ is complete:
      Move all the packets in $F_{j,new}$ to $F_{j,old}$;
      if the packets in $F_{j,old}$ were mixed before **then**
        Apply RLNC to all the packets in $F_{j,old}$ and generate M re-coded packets;
        Move the re-coded packets to the unicast output queue of Layer 2;
      else
        Apply RLNC to all the packets in $F_{j,old}$ and $F_{1-j,old}$ and generate M re-coded packets;
        Move the re-coded packets to the broadcast output queue of Layer 2;
      end if
    case Buffer $F_{j,new}$ is complete **AND** Buffer $F_{j,old}$ is complete **AND** Buffer $F_{1-j,old}$ is empty:
      Apply RLNC to all the packets in $F_{j,old}$ and generate M re-coded packets;
      Move the re-coded packets to the unicast output queue of Layer 2;
      Move all the packets in $F_{j,new}$ to $F_{j,old}$;
      if the packets in $F_{j,old}$ were mixed before **then**
        Apply RLNC to all the packets in $F_{j,old}$ and generate M re-coded packets;
        Move the re-coded packets to the unicast output queue of Layer 2;
      end if
    end switch
    if $y_{i,m}$ has not been inserted into $F_{j,new}$ **then**
      Insert $y_{i,m}$ to the i-th position of $F_{j,new}$;
    end if
  end if
end while

Outline

- Review of existing approaches for erasure channels
- FUN architecture and overview
- Description of FUN coding
- Experimental results
- Conclusion

Experimental Setup (1)

- We implement our proposed FUN-1 and FUN-2 on QualNet. For comparison, we also implement a BATS code, a fountain code (specifically, the RQ code), RLNC , and COPE 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.

Experimental Setup (2)

- 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.

Experiments of Three Cases

- We conduct experiments for the following three 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.

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

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

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

Conclusion

- This work 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 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 high throughput.

Conclusion (cont'd)

- FUN provides a unified framework for fountain coding and network coding.
- 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.
- Experimental results demonstrate that our FUN approach achieves higher throughput than existing schemes for multihop wireless networks.
- Our future work includes extending intra-session network coding to general intra-session network coding, which applies to both unicast and multicast.

Thank you!