Relaying a Fountain Code

Ramakrishna Gummadi and RS Sreenivas

Coordination Science Lab
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

IT School, PSU

06/03/08
Fountain Codes

Each client collects enough packets to decode

Coding doesn’t depend on the erasure probabilities - Rateless!

Most popular examples: LT, Raptor.
Luby Codes

- Average degree - $O(\log k) \Rightarrow \text{Logarithmic Per Symbol Complexity}$
- $k + O(\sqrt{k} \log^2 k)$ coded packets sufficient $\Rightarrow \text{Rate Optimal and very low overhead}$
- Decoding - Iterative BP decoding
Multiple hops

- Simply Forwarding - lose mincut capacity
- Decode and Reencode - Delay
- Random Linear Codes - $O(k)$ Complexity
- Chunked Codes [Harvey et. al. 2006] - $O(\log^2 k)$ complexity
- Trade off Schemes for Line networks [Pakzad et. al. 2005]
- Can we achieve the optimality of single hop?
Multiple hops

- Simply Forwarding - lose mincut capacity
- Decode and Reencode - Delay
- Random Linear Codes - $O(k)$ Complexity
- Chunked Codes [Harvey et. al. 2006] - $O(\log^2 k)$ complexity
- Trade off Schemes for Line networks [Pakzad et. al. 2005]
- Can we achieve the optimality of single hop?
Multiple hops

- Simply Forwarding - lose mincut capacity
- Decode and Reencode - Delay
- Random Linear Codes - $O(k)$ Complexity
- Chunked Codes [Harvey et. al. 2006] - $O(\log^2 k)$ complexity
- Trade off Schemes for Line networks [Pakzad et. al. 2005]
- Can we achieve the optimality of single hop?
Multiple hops

- Simply Forwarding - lose mincut capacity
- Decode and Reencode - Delay
- Random Linear Codes - $O(k)$ Complexity
- Chunked Codes [Harvey et. al. 2006] - $O(\log^2 k)$ complexity
- Trade off Schemes for Line networks [Pakzad et. al. 2005]
- Can we achieve the optimality of single hop?
Multiple hops

- Simply Forwarding - lose mincut capacity
- Decode and Reencode - Delay
- Random Linear Codes - $O(k)$ Complexity
- Chunked Codes [Harvey et. al. 2006] - $O(\log^2 k)$ complexity
- Trade off Schemes for Line networks [Pakzad et. al. 2005]
- Can we achieve the optimality of single hop?
Multiple hops

- Simply Forwarding - lose mincut capacity
- Decode and Reencode - Delay
- Random Linear Codes - $O(k)$ Complexity
- Chunked Codes [Harvey et al. 2006] - $O(\log^2 k)$ complexity
- Trade off Schemes for Line networks [Pakzad et al. 2005]
- Can we achieve the optimality of single hop?
Multiple hops

- Simply Forwarding - lose mincut capacity
- Decode and Reencode - Delay
- Random Linear Codes - $O(k)$ Complexity
- Chunked Codes [Harvey et. al. 2006] - $O(\log^2 k)$ complexity
- Trade off Schemes for Line networks [Pakzad et. al. 2005]
- Can we achieve the optimality of single hop?
Goals

• No Delay because of Decode and Re-encode
• Rateless
• Throughput rate to any node = Its min cut capacity from the source
• Complexity and Overhead similar to LT codes at all nodes
Goals

- No Delay because of Decode and Re-encode
- Rateless
  - Throughput rate to any node = Its min cut capacity from the source
  - Complexity and Overhead similar to LT codes at all nodes
Goals

- No Delay because of Decode and Re-encode
- Rateless
- Throughput rate to any node = Its min cut capacity from the source
- Complexity and Overhead similar to LT codes at all nodes
Goals

- No Delay because of Decode and Re-encode
- Rateless
- Throughput rate to any node = Its min cut capacity from the source
- Complexity and Overhead similar to LT codes at all nodes
Assumptions

- Tree Network
- Discrete Memoryless Erasure Channels
- Universal Upper bound on Erasure probabilities
Challenges

- Online Encoding
  - Intermediate Nodes can only access packets sequentially as received.

- Re-encoding the Coded packets
  - Intermediate nodes should be able to re-encode the coded packets without waiting to decode.
Challenges

- Online Encoding
  - Intermediate Nodes can only access packets sequentially as received.

- Re-encoding the Coded packets
  - Intermediate nodes should be able to re-encode the coded packets without waiting to decode.
A toy problem

\(i^{th}\) coded packet has to be a combination of the first \(i\) packets alone

- Generate a random set of \(k + o(k)\) symbols according to the LT encoding process.
- Run a mock decoder. Let \(\pi\) be the sequence in which we see decoded packets.
- Decodability \(\Rightarrow\) the coded packet used in the \(i^{th}\) step of decoding was a combination involving only the first \(i\) decoded packets.
- Do actual encoding *online* by assigning packet indices in the sequence defined by \(\pi\)!
A toy problem

\(i^{th}\) coded packet has to be a combination of the first \(i\) packets alone

- Generate a random set of \(k + o(k)\) symbols according to the LT encoding process.
- Run a mock decoder. Let \(\pi\) be the sequence in which we see decoded packets.
- Decodability \(\Rightarrow\) the coded packet used in the \(i^{th}\) step of decoding was a combination involving only the first \(i\) decoded packets.
- Do actual encoding \textit{online} by assigning packet indices in the sequence defined by \(\pi\)!
A toy problem

\(i^{th}\) coded packet has to be a combination of the first \(i\) packets alone

- Generate a random set of \(k + o(k)\) symbols according to the LT encoding process.
- Run a mock decoder. Let \(\pi\) be the sequence in which we see decoded packets.
- Decodability \(\Rightarrow\) the coded packet used in the \(i^{th}\) step of decoding was a combination involving only the first \(i\) decoded packets.
- Do actual encoding \textit{online} by assigning packet indices in the sequence defined by \(\pi\)!
A toy problem

\( i^{th} \) coded packet has to be a combination of the first \( i \) packets alone

- Generate a random set of \( k + o(k) \) symbols according to the LT encoding process.
- Run a mock decoder. Let \( \pi \) be the sequence in which we see decoded packets.
- Decodability \( \Rightarrow \) the coded packet used in the \( i^{th} \) step of decoding was a combination involving only the first \( i \) decoded packets.
- Do actual encoding online by assigning packet indices in the sequence defined by \( \pi \)!
A toy problem

The $i^{th}$ coded packet has to be a combination of the first $i$ packets alone

- Generate a random set of $k + o(k)$ symbols according to the LT encoding process.
- Run a mock decoder. Let $\pi$ be the sequence in which we see decoded packets.
- Decodability $\Rightarrow$ the coded packet used in the $i^{th}$ step of decoding was a combination involving only the first $i$ decoded packets.
- Do actual encoding *online* by assigning packet indices in the sequence defined by $\pi$!
Re-encoding coded packets

• Do concatenated coding - slap on successive layers of the same code at each hop
• Fix a sequence of block lengths, $k = k_0 \ldots k_n$ along the hops.
  • subject to: $k_i$ coded packets at node $i$ is enough to recover the $k_{i-1}$ packets that were recoded by node $i - 1$ w.h.p
  
  \[ k_i \leq k_0 (1 + \frac{\log^2 k_n}{\sqrt{k_i}}) \]
• Overhead doesn’t accumulate over hops, since $k_n = O(k_0)$ if we set $k_0 = \Omega(n^3)$.
• Complexity of encoding $\sim$ LT coding on a block length $k_i$
• Decoding - $i$ instances of LT decoding.
Re-encoding coded packets

- Do concatenated coding - slap on successive layers of the same code at each hop
- Fix a sequence of block lengths, $k = k_0 \ldots k_n$ along the hops.
  - subject to: $k_i$ coded packets at node $i$ is enough to recover the $k_{i-1}$ packets that were recoded by node $i-1$ w.h.p
  - $k_i \leq k_0 (1 + \frac{\log^2 kn}{\delta \sqrt{k}})^i$
- Overhead doesn’t accumulate over hops, since $k_n = O(k_0)$ if we set $k_0 = \Omega(n^3)$.
- Complexity of encoding $\sim$ LT coding on a block length $k_i$
- Decoding - $i$ instances of LT decoding.
Re-encoding coded packets

- Do concatenated coding - slap on successive layers of the same code at each hop
- Fix a sequence of block lengths, $k = k_0 \ldots k_n$ along the hops.
  - subject to: $k_i$ coded packets at node $i$ is enough to recover the $k_{i-1}$ packets that were recoded by node $i - 1$ w.h.p
  - $k_i \leq k_0 (1 + \frac{\log^2 \frac{kn}{\delta}}{\sqrt{k}})^i$
- Overhead doesn’t accumulate over hops, since $k_n = O(k_0)$ if we set $k_0 = \Omega(n^3)$.
- Complexity of encoding $\sim$ LT coding on a block length $k_i$
- Decoding - $i$ instances of LT decoding.
Re-encoding coded packets

- Do concatenated coding - slap on successive layers of the same code at each hop
- Fix a sequence of block lengths, $k = k_0 \ldots k_n$ along the hops.
  - subject to: $k_i$ coded packets at node $i$ is enough to recover the $k_{i-1}$ packets that were recoded by node $i-1$ w.h.p
- $k_i \leq k_0 \left(1 + \frac{\log_2 \frac{k n}{\delta}}{\sqrt{k}}\right)^i$
- Overhead doesn't accumulate over hops, since $k_n = O(k_0)$ if we set $k_0 = \Omega(n^3)$.
  - Complexity of encoding $\sim$ LT coding on a block length $k_i$
  - Decoding - $i$ instances of LT decoding.
Re-encoding coded packets

- Do concatenated coding - slap on successive layers of the same code at each hop
- Fix a sequence of block lengths, $k = k_0 \ldots k_n$ along the hops.
  - subject to: $k_i$ coded packets at node $i$ is enough to recover the $k_{i-1}$ packets that were recoded by node $i-1$ w.h.p
- $k_i \leq k_0 \left(1 + \frac{\log^2 \frac{kn}{\delta}}{\sqrt{k}}\right)^i$
- Overhead doesn’t accumulate over hops, since $k_n = O(k_0)$ if we set $k_0 = \Omega(n^3)$.
- Complexity of encoding $\sim$ LT coding on a block length $k_i$
- Decoding - $i$ instances of LT decoding.
Re-encoding coded packets

- Do concatenated coding - slap on successive layers of the same code at each hop
- Fix a sequence of block lengths, $k = k_0 \ldots k_n$ along the hops.
  - subject to: $k_i$ coded packets at node $i$ is enough to recover the $k_{i-1}$ packets that were recoded by node $i-1$ w.h.p
  - $k_i \leq k_0 (1 + \frac{\log^2 k_n}{\sqrt{k}})^i$
- Overhead doesn’t accumulate over hops, since $k_n = O(k_0)$ if we set $k_0 = \Omega(n^3)$.
- Complexity of encoding $\sim$ LT coding on a block length $k_i$
- Decoding - $i$ instances of LT decoding.
Terminology

- **Definition**
  
  An **Online Code Copy** is an ordered sequence of $k_{i+1}$ code symbols that can be generated in an "online fashion".

- **Definition**
  
  A **Code Matrix** is a $T(k) \times k_{i+1}$ random matrix of Code Symbols in which each row is an independent Online Code Copy.

- **Definition**
  
  The **Online phase** at node $i$ is defined to be the period until the time slot at which node $i$ collects a total of $k_{i+1}$ coded packets.

- **Definition**
  
  **State** is the number of packets successfully collected so far.
Terminology

• Definition

An **Online Code Copy** is an ordered sequence of \( k_{i+1} \) code symbols that can be generated in an “online fashion”.

• Definition

A **Code Matrix** is a \( T(k) \times k_{i+1} \) random matrix of Code Symbols in which each row is an independent Online Code Copy.

• Definition

The **Online phase** at node \( i \) is defined to be the period until the time slot at which node \( i \) collects a total of \( k_{i+1} \) coded packets.

• Definition

**State** is the number of packets successfully collected so far.
Terminology

- **Definition**
  
  An **Online Code Copy** is an ordered sequence of $k_{i+1}$ code symbols that can be generated in an “online fashion”.

- **Definition**
  
  A **Code Matrix** is a $T(k) \times k_{i+1}$ random matrix of **Code Symbols** in which each row is an independent **Online Code Copy**.

- **Definition**
  
  The **Online phase** at node $i$ is defined to be the period until the time slot at which node $i$ collects a total of $k_{i+1}$ coded packets.

- **Definition**
  
  **State** is the number of packets successfully collected so far.
Terminology

- **Definition**
  
  An **Online Code Copy** is an ordered sequence of \( k_{i+1} \) code symbols that can be generated in an "online fashion".

- **Definition**
  
  A **Code Matrix** is a \( T(k) \times k_{i+1} \) random matrix of **Code Symbols** in which each row is an independent **Online Code Copy**.

- **Definition**
  
  The **Online phase** at node \( i \) is defined to be the period until the time slot at which node \( i \) collects a total of \( k_{i+1} \) coded packets.

- **Definition**
  
  **State** is the number of packets successfully collected so far.
## Terminology

- **Definition**

  An **Online Code Copy** is an ordered sequence of \( k_{i+1} \) **code symbols** that can be generated in an “online fashion”.

- **Definition**

  A **Code Matrix** is a \( T(k) \times k_{i+1} \) random matrix of **Code Symbols** in which each row is an independent **Online Code Copy**.

- **Definition**

  The **Online phase** at node \( i \) is defined to be the period until the time slot at which node \( i \) collects a total of \( k_{i+1} \) coded packets.

- **Definition**

  **State** is the number of packets successfully collected so far.
**CodeMatrix**

<table>
<thead>
<tr>
<th>Code copy</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_{11}$</td>
<td>$C_{12}$</td>
<td>$C_{13}$</td>
</tr>
<tr>
<td></td>
<td>$C_{21}$</td>
<td>$C_{22}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$C_{31}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$C_{T1}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CODE BLOCK at node $i$**

Dimensions: $T \times k_{i+1}$

Number of indices used to generate the symbols: $k_i$
Algorithm

Procedure $LT - RELAY$ (node $i$)

1. **First generate:**
   
   (i) A random code matrix, $M_i = [c_{ij}]_{1 \leq i \leq T, 1 \leq j \leq k_i + 1}$
   
   (ii) An independent random online code copy, $R_i = \{\theta_j\}_{1 \leq j \leq k_i + 1}$

2. **Online phase (i.e. while in state $j$, $0 \leq j \leq k_i + 1$):**
   
   (i) In the first time slot of state $j$, use code symbol $\theta_j$ for coding.
   
   (ii) Remaining Slots: Pick $\hat{c}$ uniformly at random from the $j^{th}$ column of $M_i$. If not previously picked, send a packet coded according to $\hat{c}$. Else, it becomes an *idle slot*.

3. Beyond the online phase, generate independent coded packets at each time slot using the standard LT coding procedure.

**Theorem**

(w.h.p.) $N_i$, Number of idle slots at node $i$ satisfies $N_i \leq \log k$
Algorithm

Procedure \( LT - RELAY \) (node \( i \))

1. First generate:
   (i) A random code matrix, \( M_i = [c_{ij}]_{1 \leq i \leq T, 1 \leq j \leq k_{i+1}} \)
   (ii) An independent random online code copy, \( R_i = \{\theta_j\}_{1 \leq j \leq k_{i+1}} \)

2. Online phase (i.e. while in state \( j, 0 \leq j \leq k_{i+1} \)):
   (i) In the first time slot of state \( j \), use code symbol \( \theta_j \) for coding.
   (ii) Remaining Slots: Pick \( \hat{c} \) uniformly at random from the \( j^{th} \) column of \( M_i \). If not previously picked, send a packet coded according to \( \hat{c} \). Else, it becomes an idle slot.

3. Beyond the online phase, generate independent coded packets at each time slot using the standard LT coding procedure.

- Theorem

\( (w.h.p.) N_i, \ Number of idle slots at node \ i \ satisfies \ N_i \leq \log k \)
Algorithm

Procedure $LT - RELAY$ (node $i$)

1. First generate:
   (i) A random code matrix, $M_i = [c_{ij}]_{1 \leq i \leq T, 1 \leq j \leq k_i + 1}$
   (ii) An independent random online code copy, $\mathcal{R}_i = \{\theta_j\}_{1 \leq j \leq k_i + 1}$

2. Online phase (i.e. while in state $j$, $0 \leq j \leq k_i + 1$):
   (i) In the first time slot of state $j$, use code symbol $\theta_j$ for coding.
   (ii) Remaining Slots: Pick $\hat{c}$ uniformly at random from the $j^{th}$ column of $M_i$. If not previously picked, send a packet coded according to $\hat{c}$. Else, it becomes an idle slot.

3. Beyond the online phase, generate independent coded packets at each time slot using the standard LT coding procedure.

• Theorem

$(w.h.p.)N_i$, Number of idle slots at node $i$ satisfies $N_i \leq \log k$
Algorithm

Procedure $LT - RELAY$ (node $i$)

1. First generate:
   (i) A random code matrix, $\mathcal{M}_i = [c_{ij}]_{1 \leq i \leq T, 1 \leq j \leq k_{i+1}}$
   (ii) An independent random online code copy, $\mathcal{R}_i = \{\theta_j\}_{1 \leq j \leq k_{i+1}}$

2. **Online phase** (i.e. while in state $j$, $0 \leq j \leq k_{i+1}$):
   (i) In the first time slot of state $j$, use code symbol $\theta_j$ for coding.
   (ii) Remaining Slots: Pick $\hat{c}$ uniformly at random from the $j^{th}$ column of $\mathcal{M}_i$. If not previously picked, send a packet coded according to $\hat{c}$. Else, it becomes an *idle slot*.

3. Beyond the online phase, generate independent coded packets at each time slot using the standard LT coding procedure.

**Theorem**

(w.h.p.) $N_i$, Number of idle slots at node $i$ satisfies $N_i \leq \log k$
Algorithm

Procedure $LT - RELAY$ (node $i$)

1. First generate:
   
   (i) A random code matrix, $\mathcal{M}_i = [c_{ij}]_{1 \leq i \leq T, 1 \leq j \leq k_{i+1}}$
   
   (ii) An independent random online code copy, $\mathcal{R}_i = \{\theta_j\}_{1 \leq j \leq k_{i+1}}$

2. Online phase (i.e. while in state $j$, $0 \leq j \leq k_{i+1}$):
   
   (i) In the first time slot of state $j$, use code symbol $\theta_j$ for coding.
   
   (ii) Remaining Slots: Pick $\hat{c}$ uniformly at random from the $j^{th}$ column of $\mathcal{M}_i$. If not previously picked, send a packet coded according to $\hat{c}$. Else, it becomes an idle slot.

3. Beyond the online phase, generate independent coded packets at each time slot using the standard LT coding procedure.

• Theorem

$(w.h.p.) N_i$, Number of idle slots at node $i$ satisfies $N_i \leq \log k$
Uniformity in Code Symbol Selection

Theorem

Let $\chi = \{0, 1\}^{T(k) \times k}$ denote the ensemble of all possible realizations of the random matrix, $\Lambda$. For $\Psi = [\psi_{ij}] \in \chi$ and for any $S \subset \{1, \ldots, T(k)\} \times \{1, \ldots, k\}$, denote

$$W_S(\Psi) = \sum_{(i,j) \in S} \psi_{ij}$$

Take any $E \subset \{1, \ldots, T(k)\} \times \{1, \ldots, k\}$, with $|E| = r$, a constant represented as $E = \{e_1, \ldots, e_r\}$. For any $\phi = (\phi_1, \ldots, \phi_r) \in \{0, 1\}^r$ let $\Theta_\phi = \{\Psi \in \chi : \psi_{e_j} = \phi_j \text{ for } 1 \leq j \leq r\}$. Then, as $k \to \infty$, $P(\Theta_\phi)$ depends solely on $\sum_{i=0}^{r} \phi_i = W_E(\Psi) \forall \Psi \in \Theta_\phi$. 
Theorem

Given that (i) the subset of code symbols from $M_i$ used is uniformly random and (ii) $t$ is past the online phase, the set of all coded packets generated till time slot $t$ forms an LT code.

Proof.

Packets generated were the union of

1. $R_i$
2. An (almost) uniform random subset of code symbols from $M_i$
3. The independent LT coded packets generated past the online phase.
Min Cut Capacity

**Theorem**

Given that (i) the subset of code symbols from $M_i$ used is uniformly random and (ii) $t$ is past the online phase, the set of all coded packets generated till time slot $t$ forms an LT code.

**Proof.**

Packets generated were the union of

1. $R_i$
2. An (almost) uniform random subset of code symbols from $M_i$
3. The independent LT coded packets generated past the online phase.
Min Cut Capacity

- **Theorem**

  Given that (i) the subset of code symbols from $M_i$ used is uniformly random and (ii) $t$ is past the online phase, the set of all coded packets generated till time slot $t$ forms an LT code.

- **Proof.**

  Packets generated were the union of
  
  1. $R_i$
  2. An (almost) uniform random subset of code symbols from $M_i$
  3. The independent LT coded packets generated past the online phase.
Min Cut Capacity

Observation

\((w.h.p.)\) Assuming monotonically increasing erasure probabilities, the first \(k_i\) packets collected at node \(i\) can be decoded to recover the \((k_{i-1} - 1)\) packets that were recoded by node \(i - 1\).

Theorem

The code described is capacity achieving. That is, packets are transmitted from the source to the node \(i\) at a rate equal to 
\[
\min_{1 \leq j \leq i} (1 - \epsilon_j).
\]
Min Cut Capacity

• **Observation**

(w.h.p.) Assuming monotonically increasing erasure probabilities, the first $k_i$ packets collected at node $i$ can be decoded to recover the $k_{i-1}$ packets that were recoded by node $i - 1$.

• **Theorem**

The code described is capacity achieving. That is, packets are transmitted from the source to the node $i$ at a rate equal to $\min_{1 \leq j \leq i} (1 - \epsilon_j)$.
Min Cut Capacity

- **Observation**
  
  \((w.h.p.)\) Assuming monotonically increasing erasure probabilities, the first \(k_i\) packets collected at node \(i\) can be decoded to recover the \(k_{i-1}\) packets that were recoded by node \(i-1\).

- **Theorem**
  
  The code described is capacity achieving. That is, packets are transmitted from the source to the node \(i\) at a rate equal to:
  \[
  \min_{1 \leq j \leq i} (1 - \epsilon_j).
  \]
Min Cut Capacity

- **Observation**

  \( (\text{w.h.p.}) \) Assuming monotonically increasing erasure probabilities, the first \( k_i \) packets collected at node \( i \) can be decoded to recover the \( k_{i-1} \) packets that were recoded by node \( i - 1 \).

- **Theorem**

  The code described is capacity achieving. That is, packets are transmitted from the source to the node \( i \) at a rate equal to \( \min_{1 \leq j \leq i} (1 - \epsilon_j) \).
Conclusion

Have shown:

- Min Cut Capacity to every node.
- Ratelessness
- Order optimal delay
- Low overhead
- Low complexity.
- On arbitrarily large tree networks!
Thanks

Thank you!

For more details, please
(a) Talk to me, or
(b) Read a preprint from
http://decision.csl.uiuc.edu/~gummadi2/papers/fountain.pdf