On connections between network coding and stochastic network theory -- an introduction

Bruce Hajek

Abstract: Randomly generated coded information blocks form the basis of novel coding techniques used in communication networks. The most studied case involves linear coding, in which coded blocks are generated as linear combinations of data blocks, with randomly chosen coefficients. Digital fountain codes, including Luby’s LT codes, and Shokrollahi ‘s Raptor codes, involve coding at the source, while network coding involves coding within the network. Recently Maneva and Shokrollahi found a connection between the analysis of digital fountain codes, and fluid and diffusion limit methods, such as in the work of Darling and Norris. A simplified description of the connection is presented, including fluid and diffusion approximations, with implications for code design.
Network coding is aimed at coding within a network.

This talk will be limited to the topic of coding at source nodes, which is much further towards real world use. (See www.digitalfountain.com)

Stochastic network theory can be helpful in learning how to apply such codes in networks (e.g. tradeoff between buffer size and erasure protection) and in designing such codes. This talk focuses on the later.
Outline

- Multicast, and linear codes for erasures
- Luby’s LT codes
- Markov performance analysis for Poisson model
- Fluid approximation
- Diffusion approximation and an application
- Tree approach
Multicast with lost packets

source message: k symbols: \( S_1, S_2, \ldots, S_k \)

(fixed length binary strings)

Symbols are repeatedly broadcast in random order to many receivers, but can be lost.
Each receiver tries to collect a complete set, for example:
Multicast with lost packets

source message: $k$ symbols: $S_1, S_2, \ldots, S_k$
(fixed length binary strings)

Symbols are repeatedly broadcast in random order
to many receivers, but can be lost.
Each receiver tries to collect a complete set, for example:

$S_1, \_\_\_, S_4, S_5, \_\_\_, S_{k-2}, \_\_\_, S_k$

$$P[\text{collection complete} | \text{receive } k \log k + kc \text{ symbols}]$$
$$\approx P[\text{collection complete} | \text{receive } \text{Poi}(k \log k + kc) \text{ symbols}]$$
$$= (1 - \frac{e^{-c}}{k})^k \approx \exp(-e^{-c})$$
Multicast with coding at source, and lost packets

Source message: $k$ symbols: 
(fixed length binary strings)

$S_1 \quad S_2 \quad S_k$

Source forms $m$ coded symbols:

$S_1 \quad S_2 \quad S_m$

Symbols are repeatedly broadcast in random order to many receivers, but can be lost.
Each receiver tries to collect enough distinct coded packets:

$S_1 \quad \_\_ \quad S_4 \quad S_5 \quad \_\_ \_\_ \quad S_{m-2} \quad \_\_ \quad S_m$

For a good code, only $k$ or a few more coded packets are enough. If $m >> k$, then problem of duplicates at receiver is reduced.
Linear coding and greedy decoding

Source symbols:

- **S1**
  - 0101

- **S2**
  - 0111

- **S3**
  - 1101

- **S4**
  - 0001

Received payloads:
Linear coding and greedy decoding

Source symbols:

- S1: 0101
- S2: 0111
- S3: 1101
- S4: 0001

Received payloads:

- 1111
- 0111
- 0110
- 1100
- 0110
Linear coding and greedy decoding

Source symbols:

Received payloads:

\[
\begin{align*}
S1 & \quad ? \\
S2 & \quad ? \\
S3 & \quad ? \\
S4 & \quad ? \\
\end{align*}
\]

\[
\begin{align*}
1111 & \quad 0111 & \quad 0110 & \quad 1100 & \quad 0110
\end{align*}
\]
Linear coding and greedy decoding

Source symbols:

Received payloads:

\begin{itemize}
\item S1
\item S2
\item S3
\item S4
\end{itemize}

\begin{itemize}
\item 1111
\item 0111
\item 0110
\item 1100
\item 0110
\end{itemize}
Linear coding and greedy decoding

Source symbols:

Received payloads:

```
+ 1000 0111 0001 1100 0001
S1 ? S2 0111 S3 ? S4 ?
```

INFORMS, November 6, 2006
Linear coding and greedy decoding

Source symbols:

Received payloads:

1000 0111 0001 1100 0001
Linear coding and greedy decoding

Source symbols:

Received payloads:

1000
0111
0001
1101
0001
Linear coding and greedy decoding

Source symbols:

- S1
  - ?

- S2
  - 0111

- S3
  - 1101

- S4
  - 0001

Received payloads:

- 1000
- 0111
- 0001
- 1101
- 0001
Linear coding and greedy decoding

Source symbols:

- S1
- S2
- S3
- S4

Received payloads:

- 0101
- 0111
- 0001
- 1101
- 0001
Linear coding and greedy decoding

Source symbols:

- S1: 0101
- S2: 0111
- S3: 1101
- S4: 0001

Received payloads:

- 0101
- 0111
- 0001
- 1101
- 0001
Greedy decoding can get stuck:

Source symbols:

Received payloads: 1000 0111 0001
Greedy decoding can get stuck:

Source symbols:

Received payloads:

Nevertheless, we will stick to using greedy decoding.
LT codes (M. Luby) random, rateless, linear codes

Given the number of source symbols $k$, and a probability distribution on \{1,2,\ldots,k\}, a coded symbol is generated as follows:

e.g., $k=8$
LT codes (M. Luby) random, rateless, linear codes

Given the number of source symbols $k$, and a probability distribution on \{1,2,\ldots, k\}, a coded symbol is generated as follows:

\[ \text{e.g., } k=8 \]

Step one: select a degree $d$ with the given probability distribution.
LT codes (M. Luby) random, rateless, linear codes

Given the number of source symbols $k$, and a probability distribution on $\{1, 2, \ldots, k\}$, a coded symbol is generated as follows:

e.g., $k=8$

Step one: select a degree $d$ with the given probability distribution.

e.g., $d=3$. 
LT codes (M. Luby) random, rateless, linear codes

Given the number of source symbols $k$, and a probability distribution on $\{1,2,\ldots,k\}$, a coded symbol is generated as follows:

Step one: select a degree $d$ with the given probability distribution.  

Step two: select a subset of $\{1,\ldots,k\}$ of size $d$.  

\text{e.g., } k=8  

\text{e.g., } d=3.
LT codes (M. Luby) random, rateless, linear codes

Given the number of source symbols $k$, and a probability distribution on $\{1,2,\ldots,k\}$, a coded symbol is generated as follows:

\[
\begin{align*}
\text{e.g., } k &= 8 \\
\text{Step one: select a degree } d \text{ with the given probability distribution. } \\
\text{Step two: select a subset of } \\
\{1, \ldots, k\} \text{ of size } d. \\
\text{e.g., } d &= 3. \\
\text{e.g., } \{3,5,6\} \text{ code vector 00101100}
\end{align*}
\]
LT codes (M. Luby) random, rateless, linear codes

Given the number of source symbols $k$, and a probability distribution on $\{1, 2, \ldots, k\}$, a coded symbol is generated as follows:

Step one: select a degree $d$ with the given probability distribution.  

Step two: select a subset of $\{1, \ldots, k\}$ of size $d$.  

Step three: form the sum of the source symbols with indices in the random set

\begin{itemize}
  \item e.g., $k=8$
  \item e.g., $d=3$.  
  \item e.g., $\{3,5,6\}$ code vector 00101100
\end{itemize}
LT codes (M. Luby) random, rateless, linear codes

Given the number of source symbols $k$, and a probability distribution on \{1,2, \ldots, k\}, a coded symbol is generated as follows:

Step one: select a degree $d$ with the given probability distribution. e.g., $d=3$.

Step two: select a subset of \{1, \ldots, k\} of size $d$. e.g., \{3,5,6\} code

Step three: form the sum of the source symbols with indices in the random set e.g., form $S_3+S_5+S_6$
LT codes (M. Luby) random, rateless, linear codes

Given the number of source symbols k, and a probability distribution on \{1,2, \ldots, k\}, a coded symbol is generated as follows:

\[ \text{e.g., } k=8 \]

Step one: select a degree d with the given probability distribution.

\[ \text{e.g., } d=3. \]

Step two: select a subset of \{1, \ldots, k\} of size d.

\[ \text{e.g., } \{3,5,6\} \text{ code vector } 00101100 \]

Step three: form the sum of the source symbols with indices in the random set.

\[ \text{e.g., form } S3+S5+S6 \]

The resulting coded symbol can be transmitted along with its code vector.
Ideal soliton distribution for degree (Luby)

<table>
<thead>
<tr>
<th>steps complete $\nu$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>$\cdots$</th>
<th>$k-2$</th>
<th>$k-1$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{6}$</td>
<td>$\frac{k}{12}$</td>
<td>$\cdots$</td>
<td>$\frac{k}{(k-2)(k-3)}$</td>
<td>$\frac{k}{(k-1)(k-2)}$</td>
<td>$\frac{k}{k(k-1)}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$\frac{k-1}{2}$</td>
<td>$\frac{k-1}{6}$</td>
<td>$\frac{k-1}{12}$</td>
<td>$\cdots$</td>
<td>$\frac{k-1}{(k-2)(k-3)}$</td>
<td>$\frac{k-1}{k(k-1)}$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$\frac{k-2}{2}$</td>
<td>$\frac{k-2}{6}$</td>
<td>$\frac{k-2}{12}$</td>
<td>$\cdots$</td>
<td>$\frac{k-2}{(k-2)(k-3)}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$k-2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k-1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ideal soliton distribution for degree (Luby)

Ideally we could recover all k source symbols from k coded symbols.

<table>
<thead>
<tr>
<th>steps complete $\nu$</th>
<th>$1$</th>
<th>$2$</th>
<th>$3$</th>
<th>$4$</th>
<th>$\cdots$</th>
<th>$k-2$</th>
<th>$k-1$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$1$</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{6}$</td>
<td>$\frac{k}{12}$</td>
<td>$\cdots$</td>
<td>$\frac{k}{(k-2)(k-3)}$</td>
<td>$\frac{k}{(k-1)(k-2)}$</td>
<td>$\frac{k}{k(k-1)}$</td>
</tr>
<tr>
<td>$1$</td>
<td>$1$</td>
<td>$\frac{k-1}{2}$</td>
<td>$\frac{k-1}{6}$</td>
<td>$\frac{k-1}{12}$</td>
<td>$\cdots$</td>
<td>$\frac{k-1}{(k-2)(k-3)}$</td>
<td>$\frac{k-1}{k(k-1)}$</td>
<td>$0$</td>
</tr>
<tr>
<td>$2$</td>
<td>$1$</td>
<td>$\frac{k-2}{2}$</td>
<td>$\frac{k-2}{6}$</td>
<td>$\frac{k-2}{12}$</td>
<td>$\cdots$</td>
<td>$\frac{k-2}{(k-2)(k-3)}$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
</tr>
<tr>
<td>$k-2$</td>
<td>$1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k-1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ideal soliton distribution for degree (Luby)

<table>
<thead>
<tr>
<th>steps complete $\nu$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
<th>$k - 2$</th>
<th>$k - 1$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{6}$</td>
<td>$\frac{k}{12}$</td>
<td>...</td>
<td>$\frac{k}{(k-2)(k-3)}$</td>
<td>$\frac{k}{(k-1)(k-2)}$</td>
<td>$\frac{k}{k(k-1)}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$\frac{k-1}{2}$</td>
<td>$\frac{k-1}{6}$</td>
<td>$\frac{k-1}{12}$</td>
<td>...</td>
<td>$\frac{k-1}{(k-2)(k-3)}$</td>
<td>$\frac{k-1}{k(k-1)}$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$\frac{k-2}{2}$</td>
<td>$\frac{k-2}{6}$</td>
<td>$\frac{k-2}{12}$</td>
<td>...</td>
<td>$\frac{k-2}{(k-2)(k-3)}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$k - 2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k - 1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ideally we could recover all $k$ source symbols from $k$ coded symbols.

--The ideal soliton distribution doesn’t work well due to stochastic fluctuations. Luby introduced a robust variation (see LT paper)
--Raptor codes use precoding plus LT coding to help at the end.
Analysis

A coded symbol with reduced degree one (which has not been processed yet) is said to be in the gross ripple. Let $X_v$ denote the number of symbols in the gross ripple after $v$ symbols have been decoded.

Decoding successful if and only if ripple is nonempty through step $k-1$. 
Analysis

A coded symbol with reduced degree one (which has not been processed yet) is said to be in the gross ripple. Let $X_v$ denote the number of symbols in the gross ripple after $v$ symbols have been decoded.

Decoding successful if and only if ripple is nonempty through step $k-1$.

Study Poisson case:

Let the number of coded symbols of each degree $j$ have the $Poi(k\beta_j)$ distribution, where $\beta_1, \ldots, \beta_k$ are nonnegative. The total number of symbols is thus $Poi((\sum \beta_j)k)$ distributed.
Examine arrival process for the gross ripple

next symbol decoded

specific symbol to be decoded later

Degree j coded packet
Examine arrival process for the gross ripple

next symbol decoded

specific symbol to be decoded later

Degree j coded packet

INFORMS, November 6, 2006
Examine arrival process for the gross ripple

next symbol
decoded

specific symbol to be decoded later

Degree j coded packet

\[
\lambda_v = \sum_{j=2}^{v+2} \frac{k \beta_j(j^{v-j-2})}{(k)}
\]

\[
\approx \sum_{j=2}^{\infty} k \beta_j j(j-1)t^{j-2}
\]

\[
= k \beta''(t)
\]

where \( t = \frac{v}{k} \) is scaled time, and \( \beta \) is the function \( \beta(t) = \sum_{j=2}^{\infty} \beta_j t^j \).
Evolution of gross ripple
Evolution of gross ripple

Serve one packet
Evolution of gross ripple

Serve one packet

Others defect w/p \( \frac{1}{k-v} \)
Evolution of gross ripple

Enter arrivals \( A_v \)

\[ Poi((k - v - 1)\lambda_v) \]

Serve one packet

Others defect w/p \( \frac{1}{k - v} \)
Evolution of gross ripple

\[ X_{v+1} = X_v - 1 - L_v + A_v \]

- \( X_0 \) is \( Poi(k; \beta_1) \)
- \( A_v \) is \( Poi((k - v - 1)\lambda_v) \)
- \( L_v \) given \( X_v = j \) is \( Binom(j - 1, \frac{1}{k-v}) \).
Evolution of gross ripple

\[ X_{v+1} = X_v - 1 - L_v + A_v \]

- \( X_0 \) is \( \text{Poi}(k; \beta_1) \)
- \( A_v \) is \( \text{Poi}((k - v - 1)\lambda_v) \)
- \( L_v \) given \( X_v = j \) is \( \text{Binom}(j - 1, \frac{1}{k-v}) \).

Fluid limit of \( \frac{X_{kt}}{k} \):

\[
\dot{x}_t = a_t - 1 - \frac{x_t}{1-t}; \quad x_0 = \beta_1 \quad \text{where} \quad a_t = (1-t)\beta''(t).
\]
Evolution of gross ripple

Enter arrivals

$A_v \sim \text{Poi}((k - v - 1)\lambda_v)$

Serve one packet

Others defect w/p $\frac{1}{k-v}$

$X_{v+1} = X_v - 1 - L_v + A_v$

- $X_0 \sim \text{Poi}(k; \beta_1)$
- $A_v \sim \text{Poi}((k - v - 1)\lambda_v)$
- $L_v$ given $X_v = j$ is $\text{Binom}(j - 1, \frac{1}{k-v})$.

Fluid limit of $\frac{X_{kt}}{k}$: $x_t = a_t - 1 - \frac{x_t}{1-t}$; $x_0 = \beta_1$ where $a_t = (1 - t)\beta''(t)$.

Solution: $x_t = (1 - t)(\beta'(t) + \ln(1 - t))$
Next: diffusion analysis
Next: diffusion analysis

Solution: \( x_t = (1 - t)(\beta'(t) + \ln(1 - t)) \)
Next: diffusion analysis

Solution: \( x_t = (1 - t)(\beta'(t) + \ln(1 - t)) \)
Diffusion limit of gross ripple

Recall:  
\[ X_{v+1} = X_v - 1 - L_v + A_v \]

\[ \dot{x}_t = a_t - 1 - \frac{x_t}{1-t}; \]

Let \( \tilde{x}_t = \text{weak } \lim_{k \to \infty} \frac{X_{tk} - kx_t}{\sqrt{k}} \)

- \( X_0 \) is \( \text{Poi}(k\beta_1) \)
- \( A_v \) is \( \text{Poi}((k - v - 1)\lambda_v) \)
- \( L_v \) given \( X_v = j \) is \( \text{Binom}(j - 1, \frac{1}{k-v}) \).
Diffusion limit of gross ripple

Recall: \[ X_{v+1} = X_v - 1 - L_v + A_v \]
\[ \dot{x}_t = a_t - 1 - \frac{x_t}{1-t}; \]

Let \( \tilde{x}_t = \text{weak lim}_{k \to \infty} \frac{X_{tk} - kx_t}{\sqrt{k}} \)

Then, for a Brownian motion \( \tilde{W} \),
\[ d\tilde{x}_t = -\frac{\tilde{x}_t}{1-t} dt + b_td\tilde{W}_t; \quad \tilde{x}_0 \sim N(0, \beta_1) \]

where \( b_t^2 = a_t + \frac{x}{1-t} \)

\( X_0 \) is \( Poi(k|\beta_1) \)
\( A_v \) is \( Poi((k - v - 1)\lambda_v) \)
\( L_v \) given \( X_v = j \) is \( Binom(j - 1, \frac{1}{k-v}) \).
Recall: \[ X_{v+1} = X_v - 1 - L_v + A_v \]
\[ \dot{x}_t = a_t - 1 - \frac{x_t}{1-t}; \]
\[ L_v \text{ given } X_v = j \text{ is Binom}(j - 1, \frac{1}{k-v}). \]

Let \[ \tilde{x}_t = \text{weak lim}_{k \to \infty} \frac{X_{tk} - kx_t}{\sqrt{k}} \]

Then, for a Brownian motion \( \widetilde{W} \),
\[ d\tilde{x}_t = -\frac{\tilde{x}_t}{1-t} dt + b_t d\widetilde{W}_t; \quad \tilde{x}_0 \sim N(0, \beta_1) \]

where \[ b_t^2 = a_t + \frac{x}{1-t} \]

Yields \[ \tilde{x}_t = (1-t)(\frac{\tilde{x}_t}{1-t}) = (1-t)W(\frac{\beta'(t)+\log(1-t)+t}{1-t}) \]
Recall:
\[ X_{v+1} = X_v - 1 - L_v + A_v \]
\[ \dot{x}_t = a_t - 1 - \frac{x_t}{1-t}; \]

- \( X_0 \) is \( \text{Poi}(k\beta_1) \)
- \( A_v \) is \( \text{Poi}((k-v-1)\lambda_v) \)
- \( L_v \) given \( X_v = j \) is \( \text{Binom}(j-1, \frac{1}{k-v}) \).

Let
\[ \tilde{x}_t = \text{weak lim}_{k \to \infty} \frac{X_{tk} - kx_t}{\sqrt{k}} \]

Then, for a Brownian motion \( \tilde{W} \),
\[ d\tilde{x}_t = -\frac{\tilde{x}_t}{1-t}dt + b_t d\tilde{W}_t; \quad \tilde{x}_0 \sim N(0, \beta_1) \]

where \( b_t^2 = a_t + \frac{x_t}{1-t} \)

Yields
\[ \tilde{x}_t = (1-t)(\frac{\tilde{x}_t}{1-t}) = (1-t)W\left(\frac{\beta'(t)+\log(1-t)+t}{1-t}\right) \]

In particular,
\[ \text{Var}(\tilde{x}_t) = (1-t)(\beta'(t) + \log(1-t) + t) = x_t + t(1-t) \]
Design based on diffusion limit

The diffusion limit result suggests the representation:

\[ X_v \overset{d.}{\approx} k x(t) + \sqrt{k \sigma_t} N(0, 1) \]

which in turn suggests guidelines for the degree distribution:

\[
\begin{align*}
    k x(t) & \geq c \sqrt{k \sigma_t} \quad \text{for } 0 \leq t \leq 1 - \delta \\
    \text{or } k(1-t)(\beta'(t) + \ln(1-t)) & \geq c \sqrt{k \sqrt{t(1-t) + x(t)}} \\
    \text{(drop } x(t)) \quad \text{or} \\
    \beta'(t) & \geq -\ln(1-t) + \frac{c}{\sqrt{k}} \sqrt{\frac{t}{1-t}} \\
    \text{for } 0 \leq t \leq 1 - \delta
\end{align*}
\]
Design based on diffusion limit

The diffusion limit result suggests the representation:

\[ X_v \overset{d.}{=} kx(t) + \sqrt{k \sigma_t} N(0, 1) \]

which in turn suggests guidelines for the degree distribution:

\[
\begin{align*}
  kx(t) & \geq c\sqrt{k \sigma_t} \quad \text{for } 0 \leq t \leq 1 - \delta \\
  \text{or} \quad k(1 - t)(\beta'(t) + \ln(1 - t)) & \geq c\sqrt{k} \sqrt{t(1 - t) + x(t)} \quad \text{(drop } x(t)) \\
  \beta'(t) & \geq -\ln(1 - t) + \frac{c}{\sqrt{k}} \sqrt{\frac{t}{1 - t}} \quad \text{for } 0 \leq t \leq 1 - \delta
\end{align*}
\]

Nearly same as suggested in [Sokrollahi 2006] based on Luby heuristic:

\[
\beta'(t) \geq -\ln \left( 1 - t - c\sqrt{\frac{1-t}{k}} \right)
\]
Alternative analysis approach--
“tree method” (as in branching processes)
Alternative analysis approach--“tree method” (as in branching processes)
Conclusions

• Coding at the source, network coding, and peer-to-peer networking (gossip type algorithms) provide a rich source of network design and analysis problems.
• Iterating design and analysis can lead to interesting tractable problems (as in LT code design)
• ODE and related diffusion and large deviation analysis appear better able to provide error vs. block size estimates than tree based methods.
• For error correction, order of iterations matters. There doesn’t seem to be an ODE analysis approach for errors.
Conclusions

• Coding at the source, network coding, and peer-to-peer networking (gossip type algorithms) provide a rich source of network design and analysis problems.
• Iterating design and analysis can lead to interesting tractable problems (as in LT code design)
• ODE and related diffusion and large deviation analysis appear better able to provide error vs. block size estimates than tree based methods.
• For error correction, order of iterations matters. There doesn’t seem to be an ODE analysis approach for errors.

Thanks!
Some key papers on codes for iterative decoding


[8] **M. Luby**, “LT Codes,” *Proc. IEEE Symposium on Theory of Computing*, 2002. (Presents LT codes, the first discovered digital fountain, a.k.a. rateless erasure recovery, codes. Similar to earlier erasure codes [4], but LT codes are nonsystematic, and the degree distribution of input symbols is Poisson, rather than designed. Soliton distribution is introduced, and robust soliton distribution is shown to yield complete recovery of all input symbols with high probability.)

Some key papers on related analysis and design


[13] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman, “Analysis of random processes via and-or trees,” in Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms, pp. 354-373, 1998. (Shows how a martingale concentration argument can be used to provide bounds associated with tree type analysis of iterative decoding algorithms.)

