Network-Channel Separation

in

Adversarial Networks

Mayank Bakshi

Joint work with
Prof Michelle Effros and Prof Tracey Ho

Department of Electrical Engineering, Caltech

Mar 22, 2012
Q: Can Network Coding and Channel Coding be separated?

noisy channels + adversary
Q: Do the above two networks have same capacity regions?
Previous works: Separation holds when there is no adversary
- Separation based strategies simpler to design and analyze
- Noiseless links + adversary better understood than noisy channels + adversary
Network model:
- Directed edges (can be cyclic)
System Model

Network model:
- Directed edges
- Independent sources
System Model

Network model:
- Directed edges
- Independent sources
System Model

Network model:
- Directed edges
- Independent sources
- Independent, memoryless point-to-point channels

\[
P(z_i^{(e)} : e \in E, i = 1, \ldots, n) = \prod_{e \in E} \prod_{i=1}^{n} P_e(z_i^{(e)})
\]
System Model

Network model:
- Directed edges
- Independent sources
- Independent, memoryless point-to-point channels

\[
P(z_i^{(e)} : e \in E, i = 1, \ldots, n) = \prod_{e \in E} \prod_{i=1}^{n} P_e(z_i^{(e)})
\]

Channel capacity:
\[
C(e) \triangleq \max_{P_{X^{(e)}(\cdot)}} I(X^{(e)}; Y^{(e)})
\]
System Model

Noisy Channel

\[ Y(t) = C(e) X(t) + Z(t) \]

Network model:
- Directed edges
- Independent sources
- Independent, memoryless point-to-point channels

\[ P(z(e) : e \in E, i = 1, \ldots, n) = \prod_{e \in E} \prod_{i=1}^{n} P_e(z_{i}^{(e)}) \]

Channel capacity:

\[ C(e) \triangleq \max_{P_X(e)} I(X^{(e)}; Y^{(e)}) \]

Assume (for now):

\[ C(e) \in \mathbb{Z}^+ \ \forall \ e \in E \]
System Model

Network Code:
System Model

Network Code:

- Block network code
  - blocklength $n \in \mathbb{Z}^+$, rate $\mathbf{R} = (R_1, \ldots, R_k)$

$W^{(i)} \in \{1, 2, \ldots, 2^{nR_i}\}$

rate

$W^{(1)}$  
$W^{(2)}$  
$W^{(k)}$

$s_1$, $t_1$  
$s_2$, $t_2$  
$s_k$, $t_k$

$\hat{W}^{(1)}$, $\hat{W}^{(2)}$, $\hat{W}^{(k)}$
System Model

Network Code:

- Block network code
  - blocklength $n \in \mathbb{Z}^+$, rate $\mathbf{R} = (R_1, \ldots, R_k)$
  - mappings $(f_t(e) : e \in E, t = 1, 2, \ldots, n)$
    $\quad (g^{(i)} : i = 1, 2, \ldots, k)$
System Model

- Block network code

$W^{(i)} \in \{1, 2, \ldots, 2^{n R_i}\}$

Network Code:

- Block network code
- blocklength $n \in \mathbb{Z}^+$, rate $R = (R_1, \ldots, R_k)$
- mappings $(f_t^{(e)} : e \in E, t = 1, 2, \ldots, n)$
  $(g^{(i)} : i = 1, 2, \ldots, k)$

$X_t^{(e_1)} = f_t^{(e_1)} (W^{(1)}, Y_{1:t-1}^{(e_8)}), \ t = 1, 2, \ldots, n$
System Model

Network Code:

- Block network code
  - blocklength \( n \in \mathbb{Z}^+ \), rate \( \mathbf{R} = (R_1, \ldots, R_k) \)
  - mappings \((f_t^{(e)} : e \in E, t = 1, 2, \ldots, n)\)
    \((g^{(i)} : i = 1, 2, \ldots, k)\)

\[ X_t^{(e)} = f_t^{(e_1)}(Y_{1:t-1}^{(e_2)}, Y_{1:t-1}^{(e_3)}, Y_{1:t-1}^{(e_4)}, Y_{1:t-1}^{(e_8)}) \]
System Model

Network Code:

- Block network code
  - blocklength \( n \in \mathbb{Z}^+ \), rate \( \mathbf{R} = (R_1, \ldots, R_k) \)
  - mappings
    \[
    (f_t^{(e)} : e \in E, t = 1, 2, \ldots, n) \\
    (g_i^{(i)} : i = 1, 2, \ldots, k)
    \]

\[
\hat{W}^{(k)} = g^{(k)}(Y^{(e_5)}_{1:n}, Y^{(e_6)}_{1:n})
\]
System Model

Network Code:

- Block network code
  - blocklength $n \in \mathbb{Z}^+$, rate $R = (R_1, \ldots, R_k)$
  - mappings $(f_t^{(e)} : e \in E, t = 1, 2, \ldots, n)$
    $(g_i^{(i)} : i = 1, 2, \ldots, k)$

Byzantine Adversary:

- knows all messages and noise values
- can replace received vectors on any $K$ edges
Communication goal:

“Reliably” communicate $W^{(1)}, W^{(2)}, \ldots, W^{(k)}$ irrespective of adversary’s attack strategy
System Model

Communication goal:

“Reliably” communicate $W^{(1)}, W^{(2)}, \ldots, W^{(k)}$ \textit{irrespective of adversary’s attack strategy}

Rate vector:

$R = (R(1), R(2), \ldots, R(k))$

Capacity region $\mathcal{R}(\mathcal{N}, K)$

$R \in \mathcal{R}(\mathcal{N}, K)$ \iff there exists a network code for some block-length $n$ such that

$$\Pr(W^{(1)}, W^{(2)}, \ldots \neq \hat{W}^{(1)}, \hat{W}^{(2)}, \ldots) < \lambda$$

for every $\lambda > 0$ \textit{irrespective of adversary’s strategy.}
- $C(e) = 1$ for each $e \in E$

- Adversary can attack any one edge
- $C(e) = 1$ for each $e \in E$

- Adversary can attack any one edge

- Rate 1 is achievable
$W \in \{0, 1\}$

Blocklength = 1

- $C(e) = 1$ for each $e \in E$
- Adversary can attack any one edge
- Rate 1 is achievable
$W \in \{0, 1\}$

Blocklength = 1

- $C(e) = 1$ for each $e \in E$
- Adversary can attack any one edge
- Rate 1 is achievable
$W \in \{0, 1\}$

Blocklength =1

- $C(e) = 1$ for each $e \in E$

- Adversary can attack any one edge

- Rate 1 is achievable
Example

\[ W = 1 \]

\[ W \in \{0, 1\} \]

Blocklength = 1

- \( C(e) = 1 \) for each \( e \in E \)

- Adversary can attack any one edge

- Rate 1 is achievable
Some known results

Noiseless networks

- equal link capacities [Yeung et al 2006, Jaggi et al 2007]
  - Capacity = $\min\text{-cut} - 2K$ if each link has capacity 1
Some known results

Noiseless networks

- equal link capacities [Yeung et al 2006, Jaggi et al 2007]
  - Capacity = min-cut - 2K if each link has capacity 1

Previous example:

\[ \text{min-cut} = 3 \]
\[ K = 1 \]

\[ \Rightarrow \text{Capacity} = 1 \]
Some known results

Noiseless networks

- equal link capacities [Yeung et al 2006, Jaggi et al 2007]
  - Capacity = min-cut- $2K$ if each link has capacity 1

- partial results for unequal link capacities [Kim et al 2009]
Some known results

Noiseless networks

- equal link capacities [Yeung et al 2006, Jaggi et al 2007]
  - Capacity = min-cut - 2K if each link has capacity 1

- partial results for unequal link capacities [Kim et al 2009]

- node based adversaries [Kosut et al 2009]
Noisy networks

[Jaggi et al 2007]

- Noisy problem harder to deal with

- Separating channel coding and network coding gives achievable schemes!
Separating network and channel coding

Step 1: Channel code on each link

\[ W^{(1)} \rightarrow \hat{W}^{(1)} \]

\[ W^{(2)} \rightarrow \hat{W}^{(2)} \]

\[ W^{(k)} \rightarrow \hat{W}^{(k)} \]
Separating network and channel coding

Step 1: Channel code on each link

Step 2: Network code for noiseless network => network code for noisy network
Separating network and channel coding

Step 1: Channel code on each link

Step 2: Network code for noiseless network => network code for noisy network

=> Achievable rate vectors for noisy networks
- Separation holds between network coding and channel coding for **independent** channels

- Networks with links of same capacities are “**equivalent**”

[Song, Yeung, Cai ‘2006] for single source multicast
[ Koetter, Effros, Medard ‘2011] for general networks
Separation always optimal?

Parallel AWGNs [Koetter et al 2011]
Separation always optimal?

Parallel AWGNs [Koetter et al 2011]

- Separate coding on each link achieves at most $\log(1 + SNR)$
- Separate coding on each link achieves at most $\log(1 + SNR)$

- Receiver can “cancel out” noise by adding the two received vectors

- Capacity is infinite

Separation may be suboptimal when channels not independent!
Separating network and channel coding

Step 1: Channel code on each link

Step 2: Network code for noiseless network => network code for noisy network

=> Achievable rate vectors for noisy networks

Possible problem: Adversarial noise is not independent on different edges
Separation suboptimal

Networks with jointly distributed noise

\[ W \rightarrow Z \rightarrow W \]

Separation optimal

Networks of independent channels

\[ b_1^{(1)} b_2^{(1)} \ldots \]

\[ b_1^{(2)} b_2^{(2)} \ldots \]

\[ b_1^{(k)} b_2^{(k)} \ldots \]

\[ s_1 \]

\[ s_2 \]

\[ \vdots \]

\[ s_k \]

\[ t_1 \]

\[ t_2 \]

\[ \vdots \]

\[ t_k \]

\[ \hat{b}_1^{(1)} \hat{b}_2^{(1)} \ldots \]

\[ \hat{b}_1^{(2)} \hat{b}_2^{(2)} \ldots \]

\[ \hat{b}_1^{(k)} \hat{b}_2^{(k)} \ldots \]
Adversarial Networks?
- independent channels
- adversary may introduce "noise" that is not independent

Our work: Is separation optimal for adversarial networks?
Main Result

\[ N \]

Noisy network

\[ \hat{N} \]

Noiseless network

Theorem: \( \mathcal{R}(N, K) = \mathcal{R}(\hat{N}, K) \)
Key Ideas

$\mathcal{R}(N, K) \supseteq \mathcal{R}(\hat{N}, K)$: separate network coding and channel coding

Step 1: Channel code on each link
Key Ideas

$\mathcal{R}(N, K) \supseteq \mathcal{R}(\hat{N}, K)$: separate network coding and channel coding

Step 1: Channel code on each link

Step 2: Network code for noiseless network => network code for noisy network
Key Ideas

$\mathcal{R}(\mathcal{N}, K) \supseteq \mathcal{R}(\hat{\mathcal{N}}, K)$: separate network coding and channel coding

Step 1: Channel code on each link

Step 2: Network code for noiseless network $\Rightarrow$ network code for noisy network

$\Rightarrow R \in \mathcal{R}(\hat{\mathcal{N}}, K) \Rightarrow R \in \mathcal{R}(\mathcal{N}, K)$

(Same as Koetter et al’s proof)
Key Ideas

\( \mathcal{R}(N, K) \subseteq \mathcal{R}(\hat{N}, K) : \)

Step 1: “Emulate” channel noise on noiseless links

Channel coding:

\[
\begin{align*}
X_1^{(e)} & \xrightarrow{u} X_1^{(e)} \\
X_2^{(e)} & \xrightarrow{u} X_2^{(e)} \\
& \quad \cdots \\
X_n^{(e)} & \xrightarrow{u} X_n^{(e)} \\
\end{align*}
\]

Lossy source coding:

\[
\begin{align*}
X_1^{(e)} & \xrightarrow{b_1} X_1^{(e)} \\
X_2^{(e)} & \xrightarrow{b_2} X_2^{(e)} \\
& \quad \cdots \\
X_n^{(e)} & \xrightarrow{b_{nC(e)}} X_n^{(e)} \\
\end{align*}
\]

Given: \( p(y|x) \)
Design: \( p(x) \)

\[ C = \max_{p(x)} I(X^{(e)}; Y^{(e)}) \]

Given: \( p(x) \)
Design: \( p(y|x) \)

\[ R = \min_{p(y|x)} I(X^{(e)}; Y^{(e)}) \]
Key Ideas

\[ \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\mathcal{\hat{N}}, K) : \]

Step 1: “Emulate” channel noise on noiseless links

Channel coding:

\[ X_1^{(e)} \xrightarrow{u} c_e \xrightarrow{v} Y_1^{(e)} \]
\[ X_2^{(e)} \xrightarrow{} c_e \xrightarrow{} Y_2^{(e)} \]
\[ \vdots \]
\[ X_n^{(e)} \xrightarrow{} c_e \xrightarrow{} Y_n^{(e)} \]

Given: \( p(y|x) \)

Design: \( p(x) \)

\[ C = \max_{p(x)} I(X^{(e)}; Y^{(e)}) \]

Lossy source coding:

\[ X_1^{(e)} \xrightarrow{u} b_1 \xrightarrow{} Y_1^{(e)} \]
\[ X_2^{(e)} \xrightarrow{b_2} \xrightarrow{} Y_2^{(e)} \]
\[ \vdots \]
\[ X_n^{(e)} \xrightarrow{b_n c^{(e)}} \xrightarrow{} Y_n^{(e)} \]

Given: \( p(x) \)

Design: \( p(y|x) \)

\[ R = \min_{p(y|x)} I(X^{(e)}; Y^{(e)}) \]

For a fixed \( p(y|x) \), \( R = I(X^{(e)}; Y^{(e)}) \)
**Key Ideas**

\( \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\hat{\mathcal{N}}, K) : \)

Step 1: “Emulate” channel noise on noiseless links

**Channel coding:**

\[ X_1^{(e)} \xrightarrow{u} \mathcal{C}_e \xrightarrow{v} Y_1^{(e)} \]

\[ X_2^{(e)} \xrightarrow{v} \mathcal{C}_e \xrightarrow{v} Y_2^{(e)} \]

\[ \vdots \]

\[ X_n^{(e)} \xrightarrow{v} \mathcal{C}_e \xrightarrow{v} Y_n^{(e)} \]

Given: \( p(y|x) \)

Design: \( p(x) \)

\[ C = \max_{p(x)} I(X^{(e)}; Y^{(e)}) \]

**Lossy source coding:**

\[ X_1^{(e)} \xrightarrow{u} \text{Lossy Source Encoder} \xrightarrow{b_1} Y_1^{(e)} \]

\[ X_2^{(e)} \xrightarrow{b_2} \text{Lossy Source Encoder} \xrightarrow{b_3} Y_2^{(e)} \]

\[ \vdots \]

\[ X_n^{(e)} \xrightarrow{b_{nC(e)}} \text{Lossy Source Encoder} \xrightarrow{v} Y_n^{(e)} \]

Given: \( p(x) \)

Design: \( p(y|x) \)

\[ R = \min_{p(y|x)} I(X^{(e)}; Y^{(e)}) \]

For a fixed \( p(y|x) \), 
\[ R = I(X^{(e)}; Y^{(e)}) \leq C \]
Emulating channel noise

- mimic channel transition probability $p_e(y|x)$
- random codebook $p(y) = \sum_x p(x)p(y|x)$
- map input to a typical codeword from the Lossy Source Code according to joint distribution $p(x)p_e(y|x)$

$\mathcal{R}(N, K) \subseteq \mathcal{R}(\hat{N}, K)$
\( \mathcal{R}(N, K) \subseteq \mathcal{R}(\hat{N}, K) \)

**Emulating channel noise**

Koetter et al’s technique

- mimic channel transition probability \( p_e(y|x) \)

- random codebook \( p(y) = \sum_x p(x)p(y|x) \)

- map input to a typical codeword from the Lossy Source Code according to joint distribution \( p(x)p_e(y|x) \)

- Requires input empirical distribution to be close to \( p(x) \)

Adversary can control \( p(x) \)
Emulating channel noise

\[ \mathcal{R}(N, K) \subseteq \mathcal{R}(\hat{N}, K) \]

Our technique

\[
\begin{align*}
    X_1^{(e)} &\quad \overset{u}{\longrightarrow} \quad Y_1^{(e)} \\
    X_2^{(e)} &\quad \overset{u}{\longrightarrow} \quad Y_2^{(e)} \\
    \vdots &\quad \vdots \\
    X_n^{(e)} &\quad \overset{u}{\longrightarrow} \quad Y_n^{(e)} \\
\end{align*}
\]

Noisy channel

\[
\begin{align*}
    X_1^{(e)} &\quad \overset{u}{\longrightarrow} \quad Y_1^{(e)} \\
    X_2^{(e)} &\quad \overset{u}{\longrightarrow} \quad Y_2^{(e)} \\
    \vdots &\quad \vdots \\
    X_n^{(e)} &\quad \overset{u}{\longrightarrow} \quad Y_n^{(e)} \\
\end{align*}
\]

Noiseless channel

- Design different random codebook for each input type
Emulating channel noise

- Design different random codebook for each input type

Step 1: Describe empirically observed type
Emulating channel noise

\[ \mathcal{R}(N, K) \subseteq \mathcal{R}(\hat{N}, K) \]

- Design different random codebook for each input Type

\[ R(e) \subseteq R(\hat{e}) \]

Step 1: Describe empirically observed type

Step 2: Apply Lossy Source Code designed for the observed type

Noisy channel

Noiseless channel

Our technique
Emulating channel noise

\[ \mathcal{R}(N, K) \subseteq \mathcal{R}(\hat{N}, K) \]

- Design different random codebook for each input type

**Step 1:** Describe empirically observed type

- Works for all input distributions

**Step 2:** Apply Lossy Source Code designed for the observed type

- Works for all input distributions
Key Ideas

$\mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\hat{\mathcal{N}}, K)$:

Step 1: “Emulate” channel noise on noiseless links
Step 2: Operate a code for noisy network on the noiseless network by channel emulation
\[ \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\hat{\mathcal{N}}, K) : \]

Step 1: “Emulate” channel noise on noiseless links
Step 2: **Operate a code for noisy network on the noiseless network by channel emulation**

\[ \Pr(\text{error}, \hat{\mathcal{N}}) \leq \Pr(\text{error}, \mathcal{N}) \cdot 2^{n\epsilon} \]
Key Ideas

\( \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\hat{\mathcal{N}}, K) \):

Step 1: “Emulate” channel noise on noiseless links
Step 2: **Operate a code for noisy network on the noiseless network by channel emulation**

\[
\Pr(\text{error}, \hat{\mathcal{N}}) \leq \Pr(\text{error}, \mathcal{N}) \cdot 2^{n\epsilon}
\]

- need \( \Pr(\text{error}, \mathcal{N}) \leq 2^{-n\delta} \) for some \( \delta > \epsilon \)
\[ \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\hat{\mathcal{N}}, K) \]

\[
\begin{align*}
W^{(1)} & \rightarrow \text{Network Code} \rightarrow \hat{W}^{(1)} \\
W^{(2)} & \rightarrow \text{Network Code} \rightarrow \hat{W}^{(2)} \\
W^{(k)} & \rightarrow \text{Network Code} \rightarrow \hat{W}^{(k)} \\
\end{align*}
\]

\[ \Pr(\text{error}, \mathcal{N}) \leq 2^{-n\delta} \]

- Given network code of blocklength \( m \) s.t. \( W^{(i)} \in \{0, 1\}^{mR_i} \)
$\mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\mathcal{N}, K)$

- Given network code of blocklength $m$ s.t. $W^{(i)} \in \{0, 1\}^{mR_i}$
- Design code of blocklength $mN$ by concatenation

<table>
<thead>
<tr>
<th>$W^{(i)}(1)$</th>
<th>Network Code</th>
<th>$\hat{W}^{(i)}(1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W^{(i)}(2)$</td>
<td>Network Code</td>
<td>$\hat{W}^{(i)}(2)$</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>$W^{(i)}(N)$</td>
<td>Network Code</td>
<td>$\hat{W}^{(i)}(N)$</td>
</tr>
</tbody>
</table>

\[ \Pr(\text{error, } \mathcal{N}) \leq 2^{-n\delta} \]
\( \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\mathcal{\hat{N}}, K) \)

- Given network code of blocklength \( m \) s.t. \( W^{(i)} \in \{0, 1\}^{mR_i} \)
- Design code of blocklength \( mN \) by concatenation
- Add outer code, \( R = 1 - \alpha \)

Pr(error, \( \mathcal{N} \)) \( \leq 2^{-n\delta} \)
\[ \mathcal{R}(N, K) \subseteq \mathcal{R}(\hat{N}, K) \]

\[ \Pr(\text{error}, N) \leq 2^{-n\delta} \]

- Given network code of blocklength \( m \) s.t. \( W^{(i)} \in \{0, 1\}^{mR_i} \)
- Design code of blocklength \( mN \) by concatenation
- Add outer code, \( R = 1 - \alpha \)

Koetter et al: i.i.d. noise \( \Rightarrow \) Gallagher’s error exponents/Large Deviations \( \Rightarrow \) \( \Pr(\text{error}, N) \leq 2^{-n\delta} \)
\( \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\hat{\mathcal{N}}, K) \)

Our proof:
- Each \( W^{(i)}(j) = b_1^{(i)}(j), \ldots, b_{mR_i}^{(i)}(j) \)

\[
\Pr(\text{error}, \mathcal{N}) \leq 2^{-n\delta}
\]

\[
\begin{align*}
\tilde{b}_k^{(i)}(1) &\quad b^{(i)}_k(1) \quad \text{Network Code} \quad \hat{b}_k^{(i)}(1) \\
\tilde{b}_k^{(i)}(2) &\quad b^{(i)}_k(2) \quad \text{Network Code} \quad \hat{b}_k^{(i)}(2) \\
\vdots &\quad \vdots \quad \text{Decoder} \quad \vdots \\
\tilde{b}_k^{(i)}(NR) &\quad b^{(i)}_k(N) \quad \text{Network Code} \quad \hat{b}_k^{(i)}(NR)
\end{align*}
\]
$$\mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\hat{\mathcal{N}}, K)$$

Pr(error, \mathcal{N}) \leq 2^{-n\delta}

Our proof:
- Each $W^{(i)}(j) = b_1^{(i)}(j), \ldots, b_{mR_i}^{(i)}(j)$
- Minimum Hamming distance decoding
  - error probability is maximized when adversary acts independently at different time steps
\[ \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\mathcal{\hat{N}}, K) \]

Network Code

\[ \Pr(\text{error}, \mathcal{N}) \leq 2^{-n \delta} \]

Our proof:
- Each \( W^{(i)}(j) = b^{(i)}_1(j), \ldots, b^{(i)}_{mR_i}(j) \)
- Minimum Hamming distance decoding
  - error probability is maximized when adversary acts independently at different time steps
- Upper bound error on probability
  \[ P(W \neq \hat{W}) < 2^{-N\delta} \] in the worst case
Key Ideas

\[ \mathcal{R}(\mathcal{N}, K) \subseteq \mathcal{R}(\hat{\mathcal{N}}, K): \]

Step 1: “Emulate” channel noise on noiseless links
Step 2: Operate a code for noisy network on the noiseless network by channel emulation

\[ \mathbf{R} \in \mathcal{R}(\mathcal{N}, K) \implies \mathbf{R} \in \mathcal{R}(\hat{\mathcal{N}}, K) \]
Generalization

Network:
- Arbitrary demand model (e.g., some unicast, some multicast etc)
- Channels of arbitrary capacity

Byzantine Adversary:
- can replace received vectors on any edges in any subset $\sigma$, where $\sigma \in \Sigma$, the set of allowed attack sets.
  
  e.g. node-based attacks
**Implications**

- Code design: simplifies design of codes
  - channel code for each link
  - network code against adversary
• Code design: simplifies design of codes
  - channel code for each link
  - network code against adversary

Note: separation may perform poorly for small blocklength
Note: separation may perform poorly for small blocklength

\[ W \in \{0, 1\}^{nR} \]

With separation:

\[ \Pr(\text{error}) \sim 2^{-n\delta} \]
Small blocklength

Note: separation may perform poorly for small blocklength

\[ W \in \{0, 1\}^{nR} \]

With separation:

\[ \Pr(error) \sim 2^{-n\delta} \]

Without separation:

Channel code of length \( 2n \) can be used

\[ \Pr(error) \sim 2^{-2n\delta} \]
Implications

- Code design: simplifies design of codes if asymptotic rate is the criterion
  - channel code for each link
  - network code against adversary
Implications

- Code design: simplifies design of codes if asymptotic rate is the criterion
  - channel code for each link
  - network code against adversary

- Capacity calculation: reduces the class of problems
  Noiseless networks $\iff$ Noisy networks
  - equal link capacities [Yeung et al 2006, Jaggi et al 2007]
  - partial results for unequal link capacities [Kim et al 2009]
  - node based adversaries [Kosut et al 2009]
Implications

- Code design: simplifies design of codes if asymptotic rate is the criterion
  - channel code for each link
  - network code against adversary

- Capacity calculation: reduces the class of problems

  Noiseless networks <=> Noisy networks
  - equal link capacities [Yeung et al 2006, Jaggi et al 2007]
  - partial results for unequal link capacities [Kim et al 2009]
  - node based adversaries [Kosut et al 2009]
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems => reduces the class of problems
  - Network-Channel separation
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems => reduces the class of problems
  - Network-Channel separation
  - Network Decomposition
  - Feedback
  - Side Information
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems

- Network-Channel separation
- Network Decomposition
- Feedback
- Side Information
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems

- Network-Channel separation
- Network Decomposition
- Feedback
- Side Information
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems
  - Network-Channel separation
  - Network Decomposition
  - Feedback
  - Side Information

Our results:
  - Holds for several classes of demand [BEGK’07]
  - Does not hold in general
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems
  - Network-Channel separation
  - Network Decomposition
  - Feedback
  - Side Information

Observation:
- Feedback increases capacity [BE’09]
- Exact capacity region for the above network
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems

- Network-Channel separation
- Network Decomposition
- Feedback
- Side Information

Observation:

- Feedback increases capacity [BE’09]
- Exact capacity region for the above network

Bounds on capacities for a class of networks with feedback
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems
  - Network-Channel separation
  - Network Decomposition
  - Feedback
  - Side Information

- Multicast with side information only at sinks
  - Exact capacity region: cut-set bounds [BE’08]
General philosophy

- Bounds and techniques for one problem => Bounds and techniques for another problem
- Apply similar approach to other network problems
  - Network-Channel separation
  - Network Decomposition
  - Feedback
  - Side Information

Multicast with side information only at sinks

- Exact capacity region: cut-set bounds [BE’08]

Bounds when side information not at sink
Thank you!