Lattice Coding Increases Multicast Rates for Gaussian Multiple-Access Networks

Bobak Nazer and Michael Gastpar

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

September 27, 2007
Multicasting over AWGN Networks

- Single sender

Multicasting problem only well understood for point-to-point channel networks.
Multicasting problem only well understood for point-to-point channel networks.
Multicasting over AWGN Networks

- Single sender
- $L$ receivers
- Gaussian channel network

Multicasting problem only well understood for point-to-point channel networks.
Multicasting over Gaussian Multiple-Access Networks

- Gaussian MACs
Multicasting over Gaussian Multiple-Access Networks

- Gaussian MACs
- Point-to-point AWGN channels
Multicasting over Gaussian Multiple-Access Networks

- Gaussian MACs
- Point-to-point AWGN channels

- Want a nice reduction to a point-to-point network.
Multicasting over Gaussian Multiple-Access Networks

- Gaussian MACs
- Point-to-point AWGN channels

Want a nice reduction to a point-to-point network.
Usual solution: replace MACs with bit pipes using a MAC code.
Multicasting over Gaussian Multiple-Access Networks
Multicasting over Gaussian Multiple-Access Networks

- Bit pipe network
- Multicast capacity known
- Reduction ignores linear functions performed by MACs.
Multicasting over Gaussian Multiple-Access Networks

- Bit pipe network
- Multicast capacity known

- Reduction ignores linear functions performed by MACs.
- MAC can do network coding with structured random codes.

UC Berkeley Wireless Foundations

Nazer and Gastpar
Overview

Main ideas for this talk:

$\rightarrow (Structured)$ random coding technique that exploits structural gain not beamforming gain.
Overview

Main ideas for this talk:

⇒ (Structured) random coding technique that exploits structural gain not beamforming gain.

⇒ New relaying strategy that allows relays in a network to reliably “compute-and-forward” functions of messages.
Overview

Main ideas for this talk:

$(Structured)$ random coding technique that exploits structural gain not beamforming gain.

New relaying strategy that allows relays in a network to reliably “compute-and-forward” functions of messages.

New (achievable) multicast rates for AWGN networks with multiple-access components.
Outline

1 Background: Random Coding Theorems

2 Motivating Example

3 Multicasting over AWGN Networks
   a. Problem Statement
   b. Coding Theorem
   c. Proof Ideas

4 Extensions
Point-to-Point Communication

- **Capacity** given by:

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

- Achievability proof: Draw \(2^{nR}\) codewords of length \(n\) i.i.d. with \(p(x)\). Expected performance good so there are good codes.
Point-to-Point Communication

- **Capacity** given by:

\[
C = \max_{p(x)} I(X; Y) = \frac{1}{2} \log_2 \left(1 + \frac{P}{N}\right)
\]

- Achievability proof: Draw \(2^{nR}\) codewords of length \(n\) i.i.d. with \(p(x)\). Expected performance good so there are good codes.
Multiple-Access Communication

- **Capacity region** is the convex closure of all rate pairs satisfying:

  \[
  R_1 < I(X_1; Y|X_2) \\
  R_2 < I(X_2; Y|X_1) \\
  R_1 + R_2 < I(X_1, X_2; Y)
  \]

  for some product distribution \( p(x_1)p(x_2) \).

- **Achievability proof**: Draw \( 2^{nR_1} \) codewords i.i.d. with \( p(x_1) \) and \( 2^{nR_2} \) codewords i.i.d. with \( p(x_2) \).
Multiple-Access Communication

\[ w_1 \rightarrow \text{ENC} \xrightarrow{X^n} Z^n \rightarrow \text{DEC} \rightarrow \hat{w}_1, \hat{w}_2 \]

\[ w_2 \rightarrow \text{ENC} \xrightarrow{X^n} Z^n \rightarrow \text{DEC} \rightarrow \hat{w}_1, \hat{w}_2 \]

- **Capacity region** is the convex closure of all rate pairs satisfying:

\[
R_1 < I(X_1; Y | X_2) = \frac{1}{2} \log_2 \left( 1 + \frac{P}{N} \right)
\]

\[
R_2 < I(X_2; Y | X_1) = \frac{1}{2} \log_2 \left( 1 + \frac{P}{N} \right)
\]

\[
R_1 + R_2 < I(X_1, X_2; Y) = \frac{1}{2} \log_2 \left( 1 + \frac{2P}{N} \right)
\]

for some product distribution \( p(x_1)p(x_2) \).

- **Achievability proof**: Draw \( 2^{nR_1} \) codewords i.i.d. with \( p(x_1) \) and \( 2^{nR_2} \) codewords i.i.d. with \( p(x_2) \).
Natural Extension to General Networks

- We expect capacity results in terms of mutual informations of some distributions. For example:

\[
\max_{p(x_1, x_3)p(x_2|x_3)p(\text{stuff})} \min \left\{ I(X_1, \text{stuff}; Y_1|X_3) + I(\text{stuff}; Y_3), \right.
\]

\[
I(X_1, X_2; Y_4), I(X_3; Y_1, \text{things}) \right\}
\]
Natural Extension to General Networks

- We expect capacity results in terms of mutual informations of some distributions. For example:

\[
\max_{p(x_1, x_3)p(x_2|x_3)p(\text{stuff})} \min \{I(X_1, \text{stuff}; Y_1|X_3) + I(\text{stuff}; Y_3),
\]

\[
I(X_1, X_2; Y_4), I(X_3; Y_1, \text{things})\}
\]

- Usual Achievability Proof: Draw codewords i.i.d. from desired distributions (as well as some very nice generalizations of this).
Natural Extension to General Networks

- We expect capacity results in terms of mutual informations of some distributions. For example:

\[
\max_{p(x_1, x_3)p(x_2|x_3)p(\text{stuff})} \min \{ I(X_1, \text{stuff}; Y_1|X_3) + I(\text{stuff}; Y_3), I(X_1, X_2; Y_4), I(X_3; Y_1, \text{things}) \}
\]

- Usual Achievability Proof: Draw codewords i.i.d. from desired distributions (as well as some very nice generalizations of this).

- Is focusing on the distributions enough? Do we just need better converses?
Natural Extension to General Networks

- We expect capacity results in terms of mutual informations of some distributions. For example:

$$\max_{p(x_1,x_3)p(x_2|x_3)p(\text{stuff})} \min \{ I(X_1, \text{stuff}; Y_1|X_3) + I(\text{stuff}; Y_3),$$

$$I(X_1, X_2; Y_4), I(X_3; Y_1, \text{things}) \}$$

- Usual Achievability Proof: Draw codewords i.i.d. from desired distributions (as well as some very nice generalizations of this).
- Is focusing on the distributions enough? Do we just need better converses?
- No! These techniques fail (in expectation) as they do not exploit structure.
Motivating Example: “AWGN Butterfly”

- Drawing codewords i.i.d. from specified distributions is insufficient to prove network capacity theorems in expectation.
Motivating Example: “AWGN Butterfly”

- Drawing codewords i.i.d. from specified distributions is insufficient to prove network capacity theorems in expectation.

- AWGN channels, equal SNRs, Gaussian MAC in the center.
Motivating Example: “AWGN Butterfly”

- Drawing codewords i.i.d. from specified distributions is **insufficient** to prove network capacity theorems in expectation.

- AWGN channels, equal SNRs, Gaussian MAC in the center.

- Really **just need sum** on center path.
Motivating Example: “AWGN Butterfly”

- Drawing codewords i.i.d. from specified distributions is insufficient to prove network capacity theorems in expectation.
- AWGN channels, equal SNRs, Gaussian MAC in the center.
- Really just need sum on center path.
- Want to benefit from MAC’s addition for structural gain.
Motivating Example: “AWGN Butterfly”

Standard relaying strategies:

- **Decode-and-forward**: Need to decode both messages then compute the sum.
Motivating Example: “AWGN Butterfly”

Standard relaying strategies:

- **Decode-and-forward**: Need to decode both messages then compute the sum.
- **Compress-and-forward**: Have to send messages and noise.
Motivating Example: “AWGN Butterfly”

Standard relaying strategies:

- **Decode-and-forward**: Need to decode both messages then compute the sum.
- **Compress-and-forward**: Have to send messages and noise.
- **Amplify-and-forward**: Noise builds up with each transmission.
Motivating Example: “AWGN Butterfly”

Standard relaying strategies:

- **Decode-and-forward**: Need to decode both messages then compute the sum.
- **Compress-and-forward**: Have to send messages and noise.
- **Amplify-and-forward**: Noise builds up with each transmission.

Really want to “compute-and-forward”!
Performance Comparison

For equal per user SNR:

- **Decode-and-forward** multicasting rate:

\[
R = \frac{1}{2} \log_2 \left( 1 + \frac{P}{N} \right) + \frac{1}{4} \log_2 \left( 1 + \frac{2P}{N} \right)
\]
Performance Comparison

For equal per user SNR:

- **Decode-and-forward** multicasting rate:

\[
R = \frac{1}{2} \log_2 \left( 1 + \frac{P}{N} \right) + \frac{1}{4} \log_2 \left( 1 + \frac{2P}{N} \right)
\]
Motivating Example

Performance Comparison

For equal per user SNR:

- **Decode-and-forward** multicasting rate:
  \[
  R = \frac{1}{2} \log_2 \left( 1 + \frac{P}{N} \right) + \frac{1}{4} \log_2 \left( 1 + \frac{2P}{N} \right)
  \]

- **Compute-and-forward** achieves:
  \[
  R = \frac{1}{2} \log_2 \left( 1 + \frac{P}{N} \right) + \frac{1}{2} \log_2 \left( \frac{1}{2} + \frac{P}{N} \right)
  \]

- **Compute-and-forward** relies on **lattices**.
Lattice Basics

- Lattice is a **linear** tiling of $\mathbb{R}^n$
Lattice Basics

- Lattice is a **linear** tiling of $\mathbb{R}^n$

- Channel coding:
  codewords are points in a **power constraint** ball
Lattice Basics

- Lattice is a \textit{linear} tiling of $\mathbb{R}^n$.
- Channel coding: codewords are points in a \textit{power constraint} ball.
Lattice Basics

- **Lattice is a** **linear** tiling of $\mathbb{R}^n$

- **Channel coding:**
  - codewords are points in a **power constraint** ball

- **Urbanke-Rimoldi '98:**
  - $\exists$ lattices that achieve AWGN capacity with ML decoding

Power Constraint $\mathbb{R}^2$
Random Coding vs. Lattice Coding
Random Coding vs. Lattice Coding
Random Coding vs. Lattice Coding
Random Coding vs. Lattice Coding

- Sum of codewords is **not** a codeword.
- Must decode individual messages.
Random Coding vs. Lattice Coding

- Sum of codewords is **not** a codeword.
- Must decode individual messages.
Random Coding vs. Lattice Coding

- Sum of codewords is **not** a codeword.
- Must decode individual messages.
Random Coding vs. Lattice Coding

- Sum of codewords is not a codeword.
- Must decode individual messages.

- Sum of codewords is a codeword.
- Can decode linear functions of messages.
Outline

1. Background: Random Coding Theorems

2. Motivating Example

3. Multicasting over AWGN Networks
   a. Problem Statement
   b. Coding Theorem
   c. Proof Ideas

4. Extensions
Problem Statement

- Single sender

$w \rightarrow \square$
Problem Statement

- Single sender
- \( L \) receivers
Problem Statement

- Single sender
- $L$ receivers
- Intermediate nodes

$w \rightarrow \hat{w}$

$\hat{w}$
Problem Statement

- Single sender
- $L$ receivers
- Intermediate nodes
- Point-to-point AWGN channels

$w \rightarrow \hat{w}$

UC Berkeley Wireless Foundations
Nazer and Gastpar
Problem Statement

- Single sender
- \( L \) receivers
- Intermediate nodes

- Point-to-point AWGN channels
- Gaussian multiple-access channels
Channel Model Details

- i.i.d. additive Gaussian noise: $Z \sim \mathcal{N}(0, N_j)$
- Same transmit power constraint:
  $$\frac{1}{n} \sum_{i=1}^{n} (x_j[i])^2 \leq P$$
- Channel quality controlled by noise variance.
- Scheme for different user transmit powers at the end of the talk...
Coding Theorem: New Achievable Rates

Any multicast rate achievable on the resulting network is achievable on the original network using a compute-and-forward scheme.
Coding Theorem: New Achievable Rates

- Reduction to bit pipe network
- MACs become nodes

**Theorem**

*Any multicast rate achievable on the resulting network is achievable on the original network using a compute-and-forward scheme.*
Coding Theorem: New Achievable Rates

Theorem

Any multicast rate achievable on the resulting network is achievable on the original network using a compute-and-forward scheme.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Multicast capacity found by analyzing flows to each receiver.

Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers interfere on MACs.
Random Coding: Interference Between Flows

- Multicast capacity found by analyzing flows to each receiver.
- Random coding: flows between receivers *interfere* on MACs.
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: \( R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \)
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: \( R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \)
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: \( R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \)
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: \( R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \)
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: $R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right)$
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: $R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right)$
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: \( R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \)
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: $R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right)$
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: \( R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \)
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: $R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right)$
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: \( R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \)
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: $R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right)$
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: $R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right)$
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: $R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right)$
Compute-and-Forward: No Interference Between Flows

- Calculate flows as if MACs are interference free!
- MAC constraints are only: \( R_j < \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \)
Proof Ideas

• *Convenient* to consider sending blocks of Gaussian sources at distortion targets and then treating these as supersymbols.

• Computing linear functions of Gaussians sources over Gaussian MACs and the associated *linear processing rate*.

• Map appropriate network code to our AWGN network with lattices.
From Gaussians to Bits

- Assume, for $k$ large enough, we can send a length-$k$ i.i.d. Gaussian source with variance $\sigma^2$ from our sender to every receiver with distortion $D$.

- Then we can multicast over the network at any rate less than $R(D) = \frac{1}{2} \log_2 \left( \frac{\sigma^2}{D} \right)$.

- Proof Sketch: Fix encoding and decoding functions for every interior node in the network. Take Gaussian vectors as supersymbols in a new block code.
**Linear Functions over a Gaussian MAC**

- **length-\(k\) Gaussian sources**

- **Want linear function**
  \[ U = \alpha_1 S_1 + \alpha_2 S_2 + \cdots + \alpha_M S_M \]
  at low distortion
  \[ D = E[(U - \hat{U})^2] \]

- **Erez-Litsyn-Zamir** *IT Trans.* 2005: \(\exists\) lattices good for both source and channel coding.

- Scale up each source and quantize onto the same lattice and transmit simultaneously. Receiver decodes the sum.

- Repeat \(\ell\) times with encoders sending quantization errors.
Linear Processing Rate

- **N. and Gastpar** IT Trans. Oct. 2007: Linear function received at distortion at most:

\[
D_\ell = \sigma_S^2 \max_j \alpha_j^2 \left( \frac{MN}{N + P} \right)^\ell
\]

- Linear processing rate is given by

\[
R_{LP} = \lim_{\ell \to \infty} \frac{1}{2\ell} \log_2 R(D_\ell)
\]

- Thus, structured random code can achieve at least

\[
R_{LP} = \frac{1}{2} \log_2 \left( \frac{1}{M} + \frac{P}{N} \right)
\]

- IID random code only achieves:

\[
R_{LP} = \frac{1}{2M} \log_2 \left( 1 + \frac{MP}{N} \right)
\]
Building a Network Code

- Reduction to point-to-point network:
  - Replacing all MACs with nodes with same connectivity
  - New node’s link capacities are given by the MAC’s linear processing rate

- Consider all channels in terms of equal capacity *chunks* and draw appropriate network code over prime-sized finite field.

- Network code equations also full rank over the reals.

- Refine Gaussian vectors across original network according to network code on the reals

- Receivers make LMMSE estimates.
Coding Theorem Restated

- Reduction to bit pipe network
- MACs become nodes

- New links capacities given by linear processing rate:

\[ R_{LP} = \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \]
Coding Theorem Restated

- Reduction to bit pipe network
- MACs become nodes

- New links capacities given by linear processing rate:

\[ R_{LP} = \frac{1}{2} \log_2 \left( \frac{1}{M_j} + \frac{P}{N_j} \right) \]
MAC with Unequal Users

- Different transmit power constraints:

\[ \frac{1}{n} \sum_{i=1}^{n} (x_j[i])^2 \leq P_j \]

\[ P_{AVG} = \frac{1}{M} \sum_{j=1}^{M} P_j \]

- Idea: Layer different linear functions on top of each other.

\[ R_{LP,1} = \frac{1}{2} \log_2 \left( \frac{1}{M} + \frac{\alpha P_{AVG}}{N} \right) \]

\[ R_{LP,2} = \frac{1}{2} \log_2 \left( \frac{1}{M} + \frac{(1 - \alpha) P_{AVG}}{N + \alpha M P_{AVG}} \right) \]
Beyond Network Coding...

The “Sum-Difference” Relay MAC:

- Two senders, one receiver
- Equal transmit powers and noise variances throughout
The “Sum-Difference” Relay MAC.

Look at symmetric rate point, $R = R_1 = R_2$:

- **Structured random code** allows one relay to decode the sum and the other the difference:

$$R = \frac{1}{2} \log_2 \left( \frac{1}{2} + \frac{P}{N} \right).$$

- **IID random code** results in decoding at the relays or compress-and-forward:

$$R_{DF} = \frac{1}{4} \log_2 \left( 1 + \frac{2P}{N} \right)$$

$$R_{CF} = \frac{1}{2} \log_2 \left( 1 + \frac{P}{N} \left( \frac{2P}{3P + N} \right) \right).$$
Concluding Remarks

Lattices can be extremely useful for solving AWGN communication problems.
This has been observed by others, too:

- **Philosof-Khisti-Erez-Zamir** ISIT 2007: Lattices help for MAC with two interferences, one known at each encoder
- **Krithivasan-Pradhan** arXiv July 2007: Lattices help for distributed source coding of difference of correlated Gaussians
- **Narayanan-Wilson-Sprintson** Allerton 2007: Lattices help for two-way relaying
- **Bresler-Parekh-Tse** Allerton 2007: Lattices help on a many-to-one Gaussian interference channel
**Concluding Remarks**

**Structured random codes** will be required to prove capacity (and rate-distortion) results for many networks to come...

Some previous work:

- **N. and Gastpar** IT Trans. October 2007: Computation over Multiple-Access Channels
- **N. and Gastpar** ITW 2007 Lake Tahoe: The Case for Structured Random Codes in Network Communication Theorems