Compute-and-Forward: A Novel Strategy for Cooperative Networks

Bobak Nazer and Michael Gastpar

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

Asilomar 2008

October 27, 2008
Wireless Network

- Want to send messages to other users in a wireless network.
Wireless Network

- Want to send messages to other users in a wireless network.
Wireless Network

- Want to send messages to other users in a wireless network.
Wireless Network

- Want to send messages to other users in a wireless network.
Wireless Network

- Want to send messages to other users in a wireless network.
Wireless Network

- Want to send messages to other users in a wireless network.
Wireless Network

- Want to send messages to other users in a wireless network.
Want to send messages to other users in a wireless network.
Wireless Network

- Want to send messages to other users in a wireless network.
- Usually try to convert physical layer into a graph of bit pipes.
Can now send messages over **reliable bit pipes using routing**.

More generally, could use network coding over bit pipes.
Bit Pipe Network

- Can now send messages over reliable bit pipes using routing.
- More generally, could use network coding over bit pipes.
Bit Pipe Network

- Can now send messages over **reliable bit pipes using routing**.
- More generally, could use network coding over bit pipes.
Can now send messages over reliable bit pipes using routing.

More generally, could use network coding over bit pipes.
Bit Pipe Network

- Can now send messages over **reliable bit pipes using routing**.
- More generally, could use network coding over bit pipes.
Features of a Wireless Network

But the wireless medium is not a set of bit pipes! Have to deal with:

- **Interfering** transmissions.
- **Broadcast** constraints.
- (Possibly unknown) **fading** coefficients.
- Additive **noise**.

Typical received signal:

\[ Y_i = \sum_j h_{ij} X_j + Z_i \]
Cooperative Strategies

Lots of ways to exploit the wireless medium:

- Distributed MIMO
- Wireless Network Coding
- Distributed Beamforming
- Cooperative Diversity

Many of these strategies make use of the noisy linear combinations of the physical layer.

Cross-layer design: give higher layers direct access to the channel to implement cooperative strategies. Can we do anything else?
Compute-and-Forward Abstraction

- Users reliably decode linear combinations of messages according to fading coefficients.
- Collect equations and solve for desired messages.
- Users **reliably** decode **linear combinations** of messages according to **fading** coefficients.
- Collect equations and solve for desired messages.
Compute-and-Forward Abstraction

\[ w_1 \rightarrow w \quad a_1 w_1 + a_2 w_2 \quad \hat{w}_2 \]

\[ b_1 w_1 + b_2 w_2 \quad \hat{w}_1 \quad c_1 w_1 + c_2 w_2 \]

\[ d_1 w_1 + d_2 w_2 \]

- Users **reliably** decode linear combinations of messages according to fading coefficients.
- Collect equations and solve for desired messages.
Users reliably decode linear combinations of messages according to fading coefficients.

Collect equations and solve for desired messages.
Compute-and-Forward Abstraction

Receivers have to decide:

• which messages to decode individually,
• which messages to decode as a linear equation, and
• which messages to ignore as noise

Key questions about Compute-and-Forward:

• Can equations (sometimes) be decoded at higher rates than individual messages?
• Can we still decode individual messages at the same rates as before?

If so, we can implement many cooperative strategies in a modular fashion. (Added bonus: noise does not build up across the network.)
Computation Coding

Classical Multiple-Access

- Separate messages using scheduling, coding, frequency division, etc.
Computation Coding

Classical Multiple-Access

- Separate messages using scheduling, coding, frequency division, etc.
- Receiver must decode either $x_1$, $x_2$, or both.
**Computation Coding**

**Classical Multiple-Access**

- Separate messages using scheduling, coding, frequency division, etc.
- Receiver **must** decode either $x_1$, $x_2$, or both.
**Computation Coding**

### Classical Multiple-Access

- Separate messages using scheduling, coding, frequency division, etc.
- Receiver must decode either $x_1$, $x_2$, or both.

### Computation Coding

- Messages collide on the channel and transmitters use the same (linear) codebook.

---

UC Berkeley Wireless Foundations

Nazer and Gastpar
Computation Coding

Classical Multiple-Access

- Separate messages using scheduling, coding, frequency division, etc.
- Receiver must decode either $x_1$, $x_2$, or both.

Computation Coding

- Messages collide on the channel and transmitters use the *same* (linear) codebook.
- Receiver can decode just $x_1 + x_2$. 
Gaussian Channel Model

- **Broadcast**: each user $k$ has a complex-valued channel input $X_k$ with the usual transmit power constraint:

\[ \frac{1}{n} \sum_{i=1}^{n} |X_k[i]|^2 \leq \text{SNR} \]

- **Multiple-access**: each user $k$ sees a noisy linear combination from a subset $B_k$ of other relays:

\[ Y_k[i] = \sum_{j \in B_k} h_{jk} X_j[i] + Z_k[i] \]

- Circularly symmetric **Gaussian noise**, $Z_i \sim \mathcal{CN}(0, 1)$.
- Transmitters **do not know** fading coefficients but receivers do.
How many bits can I send?

- Classical strategy: use a combination of multiple-access and superposition codes to establish bit pipes between users.
- Achievable rates for these coding techniques are well understood.
How many equations can I send?

- Transmitters use codes with **linear structure**. Receivers decode one (or more) equations.
- Need to characterize achievable rates.
Compute-and-Forward with Lattices

First, pick a good lattice $\Lambda$ (using Erez-Litsyn-Zamir ’05):
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 **linear** tiling of $\mathbb{R}^n$

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

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

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

- **Erez-Litsyn-Zamir '05**: $\exists$ lattices that are good source and channel codes
Compute-and-Forward with Lattices

First, pick a good lattice $\Lambda$ (using Erez-Litsyn-Zamir '05):
Compute-and-Forward with Lattices

First, pick a good lattice \( \Lambda \) (using Erez-Litsyn-Zamir ‘05):

1. Each encoder transmits a point from the lattice \( \Lambda \).
Compute-and-Forward with Lattices

First, pick a good lattice \( \Lambda \) (using Erez-Litsyn-Zamir ’05):

1. Each encoder transmits a point from the lattice \( \Lambda \).
2. Each relay decodes a linear function with integer coefficients, \( U_k \), of the transmitted codewords. Integer coefficients approximate channel coefficients.

\[
Y_k = \sum_{j=1}^{\infty} h_{jk}X_j + Z_k
\]

\[
U_k = \sum_{j=1}^{\infty} a_{jk}X_j \quad \text{where } a_{jk} \in \mathbb{Z}
\]
Compute-and-Forward with Lattices

First, pick a good lattice $\Lambda$ (using Erez-Litsyn-Zamir '05):

1. Each encoder transmits a point from the lattice $\Lambda$.

2. Each relay decodes a linear function with integer coefficients, $U_k$, of the transmitted codewords. Integer coefficients approximate channel coefficients.

   $$Y_k = \sum_{j=1}^{h_j} h_{jk} X_j + Z_k$$

   $$U_k = \sum_{j=1}^{a_{jk}} a_{jk} X_j \text{ where } a_{jk} \in \mathbb{Z}$$

3. Decoder collects equations of codewords and tries to solve for desired messages.
Random Coding vs. Lattice Coding
Random Coding vs. Lattice Coding

- Random Coding: Circles
- Lattice Coding: Hexagons
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 integer combinations of messages.
Random Coding vs. Lattice Coding

- Sum of codewords is \textbf{not} a codeword.
- Must decode individual messages.

- Sum of codewords is a codeword.
- Can decode integer combinations of 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 integer combinations of messages.
Achievable Rates

- Channel to user given by \( Y = h^T X + Z \).
- User decodes integer equation \( U = a^T X \).

**Theorem**

*The user can decode the equation at rate:*

\[
R = \log \left( \frac{\text{SNR}}{1 + \text{SNR}\|h - a\|^2} \right)
\]

- \( \|h - a\|^2 \) is a **mismatch penalty** (or approximation error).
- Actually, we can do better!
Noise-Approximation Tradeoff

- Channel to user given by $Y = h^T X + Z$.
- $h$ may not be close to integer vector (large approximation error $\|h - a\|^2$).
- Idea: User scales observed channel output by $\lambda \in \mathbb{C}$ before decoding:

$$\lambda Y = \lambda h^T X + \lambda Z$$

$$\tilde{Y}_k = \tilde{h}^T X + \tilde{Z}$$

- New approximation error $\min_a \|\lambda h - a\|$ may be smaller than original $\min_a \|h - a\|$.
- Noise variance goes from 1 to $|\lambda|^2$. 
Noise-Approximation Tradeoff

\[ \lambda = 1 \]

\[ h = (1, \frac{5}{4}) \]

\[ \lambda h = (1, \frac{5}{4}) \]

Noise Variance = 1

Approximation Error = \[ \left( \frac{1}{4} \right)^2 = \frac{1}{16} \]
Noise-Approximation Tradeoff

\[ \lambda = 2 \]
\[ h = \left(1, \frac{5}{4}\right), \quad \lambda h = \left(2, \frac{10}{4}\right) \]

Noise Variance = 4

Approximation Error = \( \left(\frac{1}{2}\right)^2 = \frac{1}{4} \)
Noise-Approximation Tradeoff

\[ \lambda = 3 \]

\[ h = (1, \frac{5}{4}), \lambda h = (3, \frac{15}{4}) \]

Noise Variance = 9

Approximation Error = \( \left(\frac{1}{4}\right)^2 = \frac{1}{16} \)
Noise-Approximation Tradeoff

\[ \lambda = 4, \quad h = \left(1, \frac{5}{4}\right), \quad \lambda h = (4, 5) \]

Noise Variance = 16

Approximation Error = 0
Achievable Rates

Theorem

The user can decode the equation $U = a^T X$ at rate:

$$R = \max_{\lambda \in \mathbb{C}} \log \left( \frac{\text{SNR}}{|\lambda|^2 + \text{SNR} \|\lambda h - a\|^2} \right)$$
Achievable Rates

Theorem

The user can decode the equation $U = a^T X$ at rate:

$$R = \max_{\lambda \in \mathbb{C}} \log \left( \frac{\text{SNR}}{|\lambda|^2 + \text{SNR}\|\lambda h - a\|^2} \right)$$

$$= \log \left( \frac{1}{\|a\|^2 - \lambda_{\text{MMSE}} < a, h >} \right)$$

- The optimal choice of $\lambda$ is always given by the MMSE coefficient:

$$\lambda_{\text{MMSE}} = \frac{\text{SNR} < h, a >}{1 + \text{SNR}\|h\|^2}$$
Maximizing Rates

- User should choose equation coefficients to maximize the rate. Only need to evaluate $O(||h||^2 \text{SNR})$ possible coefficient vectors. The rest trivially give 0 rate.

- Equation decoded successfully if all messages in the equation are below this rate.

- If a set of equations can be solved for a particular message, can recover that message at the minimum of the equation rates.
Large Gains are Possible

- In many cases, can decode equations at higher rates than any individual message.

- Example: no fading Gaussian MAC given by $Y = \sum_{m=1}^{M} X_m + Z$.

- Bit pipe solution: $M$ individual messages can be recovered each at rate:

  $$R = \frac{1}{M} \log (1 + M \text{SNR})$$

- Compute-and-forward: Sum of $M$ messages can be recovered at rate:

  $$R = \log \left( \frac{1}{M} + \text{SNR} \right)$$

- We may only need the sum of messages in some cases (e.g. network coding).
Layering Equations

We may want to send \textbf{more than one} equation at a time:

- Two lattices: one at power $\text{SNR}_1$, other at $\text{SNR}_2$.
- User first decodes integer equation $U_1 = a^T X$ (from first lattice) then $U_2 = b^T X$ (from second lattice).
- Rates given by:

  \[
  R_1 = \max_{\lambda \in \mathbb{C}} \log \left( \frac{\text{SNR}_1}{|\lambda|^2(1 + \text{SNR}_2\|h\|^2) + \text{SNR}_1\|\lambda h - a\|^2} \right)
  \]

  \[
  R_2 = \max_{\lambda \in \mathbb{C}} \log \left( \frac{\text{SNR}_2}{|\lambda|^2 + \text{SNR}_1\|\lambda h - a\|^2 + \text{SNR}_2\|\lambda h - b\|^2} \right)
  \]
Bit Pipe Solution is a Special Case

- If we restrict our choice of coefficients to unit vectors, then compute-and-forward simply becomes the bit pipe solution.

- Single equation example: for any channel \( h \), if \( a = [1 0 0 \cdots 0] \), (meaning try to decode just \( x_1 \)) then:

\[
R = \log \left( 1 + \frac{|h_1|^2 \text{SNR}}{1 + \text{SNR} \sum_{m=2}^{M} |h_m|^2} \right)
\]

- This is just the corner point of the usual multiple-access region.
Simple Example: “Wireless Butterfly”

Considered relaying strategies:

- **Compute-and-forward**: Decode linear equation of packets. *Approximation error* for non-integer channel coefficients.

- **Bit pipe**: Need to decode *both messages* then compute the sum.

- **Analog network coding**: Have to send messages and *noise*.

For the sake of comparison, we explicitly forbid compute-and-forward from decoding unit vectors.
No Fading Case

- No fading in the network.
- Compute-and-forward is asymptotically optimal!
- Bit pipe is interference limited.
- Analog network coding is noise limited.
Fading Case

- Rayleigh fading known only at receivers.
- Outage probability is 0.25
- Compute-and-forward good at moderate SNR.
- Bit pipe good at low SNR.
- Analog network coding good at high SNR.

![Graph showing the relationship between SNR in dB and multicast rate for different coding methods.](image)
Conclusions

• **Compute-and-Forward** is a physical layer scheme that extends the bit pipe solution to decoding reliable equations.

• Significant gains are possible since it exploits the linear combinations of the wireless channel.

• Can be used in a modular fashion for many cooperative schemes including:
  - Distributed MIMO (**Nazer-Gastpar ISIT ’08**)
  - Wireless Network Coding (**Nazer-Gastpar Allerton ’07, WiNC ’08**)
  - Distributed Estimation in Sensor Networks (**Sarwate-Nazer-Gastpar SSP ’07, Nazer-Gastpar IZS ’08**)
  - Neighborhood Gossip (**Nazer-Dimakis-Gastpar Allerton ’08**)

UC Berkeley Wireless Foundations

Nazer and Gastpar
Related Work

Lattice (and linear) codes are useful in many other multi-user problems!

- Distributed Source Coding (Krithivasan-Pradhan ’08)
- Distributed Function Compression (Körner-Marton ’79, Krithivasan-Pradhan ’07, Wagner ’08)
- Two-Way Relay Channel (Wilson-Narayanan-Pfister-Sprintson ’07, ’08, Nam-Chung-Lee ’08)
- Interference Cancellation (Philosof-Khisti-Erez-Zamir ’07, Bresler-Parekh-Tse ’07, Sanderovich-Peleg-Shamai ’08, Sridharan-Jafarian-Vishwanath-Jafar-Shamai ’08)
- Secrecy (He-Yener ’08)