Physical Communication 6 (2013) 134–141

Contents lists available at SciVerse ScienceDirect

Physical Communication

journal homepage: www.elsevier.com/locate/phycom

Full length article

Function computation via subspace coding[☆]

Nikhil Karamchandani[b,c], Lorenzo Keller[a,*], Christina Fragouli[a], Massimo Franceschetti[b]

[a] École Polytechnique Fédérale de Lausanne (EPFL), CH-1015, Lausanne, Switzerland

[b] University of California, San Diego, La Jolla, CA 92093, United States

[c] University of California Los Angeles, Los Angeles, CA 90095-1594, United States

ARTICLE INFO

Article history:
Received 15 April 2011
Received in revised form 20 February 2012
Accepted 21 February 2012
Available online 3 March 2012

Keywords:
Function computation
Network coding
Subspace coding
Wireless sensor network

ABSTRACT

This paper considers function computation in a network where intermediate nodes perform randomized network coding, through appropriate choice of the subspace codebooks at the source nodes. Unlike traditional network coding for computing functions, that requires intermediate nodes to be aware of the function to be computed, our designs are transparent to the intermediate node operations.

1. Introduction

In sensor networks, the need for energy efficiency has stimulated research efforts towards in-network aggregation and function computation. Information-theoretic studies have focused on the design of *optimal* schemes for different target functions and network classes; see for example [1–3]. On the other hand, recent work [4,5] has pointed out the need to have *simple* coding schemes, since "systems are hard to develop and debug". They advocate a solution where most nodes in the network perform the same operations regardless of the function to be computed, and the onus of guaranteeing successful computation is on a few special nodes that are allowed to vary their operation.

Motivated by the above considerations, we consider the problem of computing functions in a network where multiple sources are connected to a single sink via relays. The sources may have several different possible codebooks, and can select which one to employ depending on the function to be computed. Given a certain target function, each source transmits a codeword corresponding to its observed message. The relay nodes, however, perform the same linear operations, for example randomized network coding (which is a practical and efficient way of transmitting data in a network [6]) irrespective of the target function, i.e., the vectors inserted by the sources are randomly combined and forwarded towards the sink, using linear coefficients that are unknown to both the sources and the sink. The sink then proceeds to evaluate the target function of the source messages.

Following [7–9], we use subspace coding for computing functions in our network model. Given a target function, we assume that each source uses a codebook consisting of subspaces. Each source message is mapped to a subspace in the codebook. When a source generates a message, it injects the basis vectors of the corresponding subspace into the network. The network operation is abstracted by assuming that the sink collects enough linear combinations of these vectors to learn the joint span of the injected subspaces. Given this information, the sink then attempts to compute the target function of the source messages. Our objective is to design codebooks which minimize the

☆ This work was partially funded by ERC Project NOWIRE (ERC-2009-StG-240317) and NSF CNS-0916778.

* Corresponding author.

E-mail addresses: nikhil@ee.ucla.edu (N. Karamchandani), lorenzo.keller@epfl.ch (L. Keller), christina.fragouli@epfl.ch (C. Fragouli), massimo@ece.ucsd.edu (M. Franceschetti).

1874-4907/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.phycom.2012.02.007

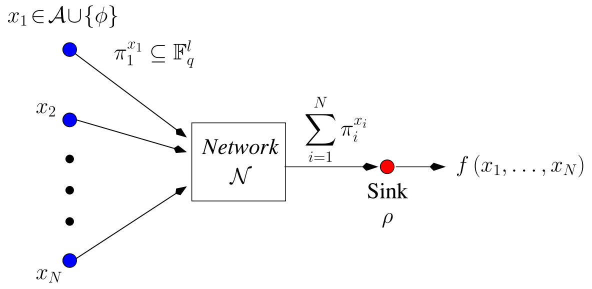
number of symbols each source needs to transmit, while guaranteeing successful function computation by the sink.

Thus, we envision a network architecture where intermediate network nodes always perform the same operations for information transfer, which leads to a simple implementation. At the same time, the sink has the flexibility to utilize the network to learn different functions of the source data by informing the source nodes to employ the corresponding codebooks. In this paper, we are interested in non-coherent communication; this allows for a low complexity asynchronous operation of the network [7]. Another approach would be to consider the coherent case as done in [10].

We note that a scheme which optimizes the intermediate node operations according to the function to be computed might need fewer transmissions. However, it would be more complex to implement, would require topology knowledge, and might be sensitive to the employed communication protocol. In contrast, our approach is transparent both to the topology and the employed communication protocol: the only requirement we impose is that we gather sufficient linearly independent combinations. As a result, our protocol would be very well suited to dynamically changing topologies, and could be applied without change on top of very different communication protocols.

The paper is organized as follows. Section 2 presents the problem formulation. In Section 3, we present various lower bounds on the number of symbols each source needs to transmit to evaluate an arbitrary function. In Section 4, we discuss various example target functions. In particular, we provide lower bounds as well as near-optimal coding schemes for the *identity*, *T-threshold*, *maximum* and *K-largest values* functions. Finally, in Section 3, we present a constructive scheme to evaluate arbitrary functions.

2. Problem formulation and notation


$x_1 \in \mathcal{A} \cup \{\phi\}$
$\pi_1^{x_1} \subseteq \mathbb{F}_q^l$
x_2
x_N
Network $\mathcal{N}$
$\sum_{i=1}^{N} \pi_i^{x_i}$
Sink ρ
$f(x_1, \ldots, x_N)$

We consider a set of N sources $\sigma_1, \sigma_2, \ldots, \sigma_N$ connected to a sink ρ via a network $\mathcal{N}$. Each source σ_i is either inactive or observes a message $x_i \in \mathcal{A}$, where $\mathcal{A}$ is a finite alphabet. For ease of notation, when a source σ_i is inactive we will set $x_i = \phi$. The sink needs to compute a *target function* f of the source messages, where f is of the form

$$f : (\mathcal{A} \cup \{\phi\})^N \longrightarrow \mathcal{B}.$$

Some example target functions are defined below.

Definition 2.1.
- The *identity* target function has $\mathcal{B} = (\mathcal{A} \cup \{\phi\})^N$ and is defined by
$$f(x_1, \ldots, x_N) = (x_1, \ldots, x_N).$$
- For $m \geq 1$, the *arithmetic sum* target function has $\mathcal{A} = \{1, \ldots, m\}$, $\mathcal{B} = \{0, 1, \ldots, mN\}$, and is defined by
$$f(x_1, \ldots, x_N) = x_1 + x_2 + \cdots + x_N$$
where '$+$' denotes ordinary integer summation. For any $a \in \mathcal{A} \cup \{\phi\}$, we set $a + \phi = a$.
- Let $\mathcal{A}$ be an ordered set. The *maximum* target function has $\mathcal{B} = \mathcal{A}$ and is defined by
$$f(x_1, \ldots, x_N) = \max\{x_1, \ldots, x_N\}.$$
For any $a \in \mathcal{A} \cup \{\phi\}$, we set $\max\{a, \phi\} = a$.
- The *parity* target function has $\mathcal{A} = \mathcal{B} = \{0, 1\}$, and is defined by
$$f(x_1, \ldots, x_N) = x_1 \oplus x_2 \oplus \cdots \oplus x_N$$
where $\oplus$ denotes mod-2 addition. Again, for any $a \in \mathcal{A} \cup \{\phi\}$ we set $a \oplus \phi = a$.
- The *majority* target function has $\mathcal{A} = \mathcal{B} = \{0, 1\}$, and is defined by
$$f(x_1, \ldots, x_N) = \begin{cases} 1 & \text{if } |\{i : x_i = 1\}| > |\{i : x_i = 0\}| \\ 0 & \text{otherwise.} \end{cases}$$

We consider operation using subspace coding. We denote a subspace by π and the union of two subspaces π_1, π_2 is defined as $\pi_1 + \pi_2 = \{\mathbf{x} + \mathbf{y} : \mathbf{x} \in \pi_1, \mathbf{y} \in \pi_2\}$. We write $\pi_1 + \pi_2 + \cdots + \pi_m$ as $\sum_{i=1}^{m} \pi_i$. The network operates as follows.

- At each source, every alphabet symbol is mapped to a subspace, which serves as the corresponding codeword. Thus, each source σ_i has an associated codebook $\mathcal{C}_i = \left\{\pi_i^j\right\}_{j \in \mathcal{A}}$ where π_i^j is a d-dimensional subspace[1] of an l-dimensional vector space $\mathbb{F}_q^l$ where $d, l \geq 1$ are design parameters. When the source σ_i is active and observes a message $x_i \in \mathcal{A}$, it injects into the network $\mathcal{N}$ a set of d vectors from $\mathbb{F}_q^l$ which span the subspace $\pi_i^{x_i}$. When the source is σ_i inactive, it does not make any transmissions and hence we set $\pi_i^{\phi} = \{0\}$.
- The sink ρ receives from the network $\mathcal{N}$ a set of vectors from $\mathbb{F}_q^l$ which span the union of the input subspaces[2] i.e., ρ observes $\sum_{i=1}^{N} \pi_i^{x_i}$.
- The sink uses the received information to compute the value of $f(x_1, x_2, \ldots, x_N)$.

A (d, l) *feasible code for computing* f is a collection of codebooks $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N\}$ such that each π_i^j in the codebooks is a d-dimensional subspace of $\mathbb{F}_q^l$ and the sink can compute the value of $f(x_1, x_2, \ldots, x_N)$ for any choice of input messages $x_1, x_2, \ldots, x_N$ where each $x_i \in \mathcal{A} \cup \{\phi\}$, i.e. for all $x_1, x_2, \ldots, x_N$ and $x_1', x_2', \ldots, x_N'$, if

$$f(x_1, x_2, \ldots, x_N) \neq f(x_1', x_2', \ldots, x_N'),$$

then $\sum_{i=1}^{N} \pi_i^{x_i} \neq \sum_{i=1}^{N} \pi_i^{x_i'}$.

[1] Although restricting our code design to subspaces of equal dimension may not always be optimal, it significantly simplifies the design, and is a standard approach in the literature [7,11].

[2] In practice, networks operate in rounds. The duration of a round can be chosen large enough to ensure that the sink receives enough linear independent combinations to span the union of the input subspaces.