Function Computation over Linear Channels

Lorenzo Keller* Nikhil Karamchandani† Christina Fragouli*

*School of Computer and Communication Sciences
EPFL, Switzerland

†Dept. of Electrical and Computer Engineering
UCSD, USA

{Email : lorenzo.keller@epfl.ch, nikhil@ucsd.edu, christina.fragouli@epfl.ch}

***Abstract*—We consider multiple non-colocated sources communicating over a network to a common sink. We assume that the network operation is fixed, and its end result is to convey a fixed linear deterministic transformation of the source data to the sink. This linear transformation is known both at the sources and at the sink. We are interested in the problem of function computation over such networks. We design communication protocols that can perform computation without modifying the network operation, by appropriately selecting the codebook that the sources employ to map their measurements to the data they send over the network.**

I. Introduction

We consider multiple non-colocated sources communicating over a network to a common sink. We assume that the network operation is fixed, and its end result is to convey a fixed linear deterministic transformation of the source data to the sink. This linear transformation is known in advance both at the sources and at the sink. Examples of networks that fall in this category are sensor protocols that convey the average value of the sensor measurements at the sink by running a distributed consensus algorithm, see for example [1]. The sink effectively collects the sum of the values that the sensor nodes insert in the network. Another category is networks that perform fixed network coding operations. In this case, the sink observes the vector of the source data multiplied by the transfer matrix of the network.

We are interested in the problem of function computation over such networks. Our protocols perform computation without modifying the network operation, by appropriately selecting the codebook that the sources employ to map their measurements to the data they send over the network.

This problem is motivated from practical considerations, since it allows to decouple the function computation from the intermediate node operations, leading to systems which are easier to debug and more stable [2]. For example, we can imagine a sensor network, where for the majority of the time the sink collects the average value of the source measurements, but in some particular circumstances is interested in collecting all the node values, or in learning the maximum value. It can achieve this without altering all the sensor node operations, by simply informing the source nodes to use an appropriate codebook. In other words, the routing protocol remains oblivious to the particular function that is being computed.

Function computation over networks is a problem that has attracted significant interest in the sensor network community, both from a systems [3],[4] and a theoretical viewpoint [5],[6]. This problem has also been examined in the context of network coding: for example in [7],[8] the authors examine the problem of designing network codes that allow function computation at designated sinks. However, in all these cases, the designed protocols require that the operation of the intermediate network nodes are designed with the specific function computation task in mind. In contrast, in our work, the function computation is transparent to the intermediate network node operations, and is achieved by only selecting appropriate encodings at the sources. Function computation has also been examined using information theoretical tools, see for example [9],[10] and references therein. In our work, we are interested in combinatorial designs and bounds. In our parallel work in [11] we study function computation when the network transformation is unknown, and potentially changes over time. In this paper, we consider the case where this transformation is fixed and known to the sources and the sink.

We formulate several variations of this problem and introduce our notation in Section II. In the successive two sections, we study how to design codes when the network delivers the sum of the symbols transmitted by the sources. We focus on the sum channel, as calculating the average is a very popular sensor application, well studied and deployed. In Section III, we look at one timeslot computation, where we are allowed to send only one symbol from each source. In Section IV, we look at computation over multiple timeslots, where we use the network multiple times to convey the required information to the sink. In Section V, we show how to generalize the results found in the previous sections to a broader class of networks. Finally, we conclude the paper in Section VI.

II. Problem formulation and notation

Our network consists of a set of N sources $\sigma_1, \ldots, \sigma_N$, an arbitrary number of relay nodes, and a sink. We assume a synchronized operation of the network, where time is divided into timeslots and rounds. Each round contains T timeslots.

At the beginning of each round, every source σ_i observes a value $u_i \in \mathcal{A}$, where $\mathcal{A}$ is the source alphabet. Let u be the vector $(u_1, \ldots, u_N)$ and $K = |\mathcal{A}|$ be the cardinality of the source alphabet, then u can take K^N values. At each timeslot t, every source σ_i sends one symbol $x_i[t] \in \mathbb{F}_q$, where q is a prime number. Let $x[t] = (x_1[t] \ldots x_N[t]) \in \mathbb{F}_q^N$ be the $N \times 1$ vector containing the symbols injected by the N sources at timeslot t. At each timeslot t, the sink receives an $M \times 1$ output vector

$$y[t] = A \cdot x[t] \tag{1}$$

where the *channel matrix* A is an $M \times N$ matrix over $\mathbb{F}_q$[1]. This matrix is fixed and known in advance to both the sources and the sink. Let $(A)_{i,j}$ denote the entry in the i-th row and the j-th column of the matrix A. In every round, the sink collects T output vectors, and is interested in computing a function $f : \mathcal{A}^N \longrightarrow \mathcal{B}$ of the source observations, where $\mathcal{B}$ is a finite alphabet. To compute this function, the sink uses the T output vectors it received during the round.

In this work we assume that the channel matrix A has been designed for a specific purpose and is pre-specified, e.g. compute sum of values in the network. Some common channel matrices A are as follows:

- $A = (1 \dots 1)$: the sink receives the $(\mathrm{mod}\, q)$ sum of the input symbols. This channel matrix represents the case where the sink calculates the average value of the source measurements. We will call this the *sum channel*.
- $(A)_{i,j} \in \{0, 1\}$ for every i, j : the sink receives the sum of multiple subsets of sources. This type of channel matrix represents the case in which the sink collects averages of (possibly non-disjoint) subsets of the sources.
- Matrices such that for every $1 \leqslant j \leqslant N$, $|\{i|(A)_{i,j} \neq 0\}| = 1$, in other words matrices that have exactly one non-zero entry per column. In this case the sink receives the weighted sum of disjoint subset of the sources. This type of channel matrix represents the case in which the sources are partitioned in M disjoint sets and one sum is computed for every partition.

Note that although these matrices are not likely to be observed if A is drawn uniformly from all possible matrices, they are very popular in sensor network applications.

Given a network, an associated channel matrix A as in (1), and a specific function f of the source observations, our goal is to design a communication protocol that allows the sink to compute the function value at each round. Such a protocol consists of two parts. The first part is an encoding function that, given a set of observations u, determines the symbol $x_i[t]$ each source σ_i transmits during timeslot t of that round. The second part consists of a decoding function that maps the T channel output vectors collected at the sink to the value $f(u)$.

We distinguish between interactive and non-interactive communication protocols. In non-interactive protocols, the symbol that any source σ_i transmits in time slot t_0 depends only upon its own observation u_i during that round; in interactive protocols, what source σ_i transmits may additionally depend on all the information it has received from the network until timeslot t_0.

It is easy to see that we can compute all functions provided that either the field size q or the number of timeslots T is sufficiently large. However, for a given channel matrix A and number of timeslots T, it might not be possible to compute a specific function f. Example 1 provides such a situation. In particular, whether a specific function can be computed or not depends upon the field of operation $\mathbb{F}_q$, the channel matrix A, and the number of timeslots T.

Example 1 Assume that $N = 2$, $\mathcal{A} = \{0, 1\}$, $q = 2$, $T = 1$ and $A = (1 \dots 1)$. Let x_i^j denote the symbol sent by the source σ_i when $u_i = j \in \{0, 1\}$. Consider the function $f(u_1, u_2) = u_1$ AND u_2 that takes the value 1 if and only if $u_1 = u_2 = 1$. Clearly, each source needs to send a different symbol corresponding to observed values 0 and 1, i.e.,

$$x_1^0 \neq x_1^1 \text{ and } x_2^0 \neq x_2^1. \tag{2}$$

Moreover, we need that $(x_1^0 + x_2^0) \bmod 2 \neq (x_1^1 + x_2^1) \bmod 2$, and hence either $x_1^0 \neq x_2^0$ or $x_1^1 \neq x_2^1$ must hold, which contradicts (2). The same argument extends to any arbitrary number of sources N.

In this paper, we will study when it is possible[2] to compute specific functions f in three cases: (i) when $T = 1$, which we call one timeslot computation, (ii) when $T > 1$ and we have non-interactive communication and (iii) when $T > 1$ and we have interactive communication. In the last two cases, we will be interested in the minimum T that allows the function computation.

In the next two sections, we examine whether specific functions can be calculated over the sum channel. Then in Section V, we will show how to translate achievability results for the sum channel to a more general set of channels.

III. One timeslot ($T = 1$) computation

As we saw in Example 1, for a given channel A and field size q, it may not be possible to calculate all functions f. In this section, we first derive a simple necessary condition on whether a function is computable or not on a given channel A. We then design encoding and decoding schemes for specific functions over the sum channel, i.e., when A is the all ones vector.

A. A necessary condition

Proposition 1 provides a necessary condition that formalizes the following simple observation. Rank properties of the matrix A and the field size q limit the number of possible output values that the sink observes in each timeslot. If for a given function f, the number of outputs of the network is smaller than the possible output values for the function, then f cannot be computed in one timeslot.

Proposition 1 *Let $S \subseteq \{1, 2, \dots, N\}$ be a subset of the source indices, and denote these indices as $S = \{i_1, \dots, i_{|S|}\}$. Let $\hat{u}_{i_1}, \dots, \hat{u}_{i_{|S|}} \in \mathcal{A}$ be a set of values sent by the sources in S. Let A be a channel matrix over $\mathbb{F}_q$ and let $\bar{A}$ the matrix obtained by deleting columns $i_1, \dots, i_{|S|}$ from A. A function f can be computed only if*

$$|\{f(u)|u \in \mathcal{A}^N : u_i = \hat{u}_i \ \forall\ i \in S\}| \leqslant q^{\mathrm{rank}(\bar{A})}$$

[1]We underline that for simplicity we assume our network operates over a field $\mathbb{F}_q$, where q is a prime. However, for large enough q, such operations can serve as approximations of real operations.

[2]In this paper we study the feasibility of function computation for any pre-specified field size q. An alternate approach would be to minimize some cost metric by optimizing q, but this is not the objective of this work.

i.e., the number of possible outputs of the function when fixing the values of $\sigma_{i_1}, \ldots, \sigma_{i_{|S|}}$ is smaller than the possible different outputs of the channel when fixing $x_{i_1}, \ldots, x_{i_{|S|}}$.

Proof: Let $i_{|S|+1}, \ldots, i_N$ be the indices of the sources not in S. Let $\tilde{A}$ be the matrix obtained by keeping only the columns $i_1, \ldots, i_{|S|}$ of A. Let $\tilde{x} = (x_{i_1}, \ldots, x_{i_{|S|}})$ and $\bar{x} = (x_{i_{|S|+1}}, \ldots, x_{i_N})$. The network output can be rewritten as

$$y = \bar{A} \cdot \bar{x} + \tilde{A} \cdot \tilde{x}.$$

Notice that $\tilde{A} \cdot \tilde{x}$ is fixed. Since the column space of $\bar{A}$ has dimension $\mathrm{rank}(\bar{A})$ it contains $q^{\mathrm{rank}(\bar{A})}$ vectors. Since y is a possible output if and only if $y - \tilde{A} \cdot \tilde{x}$ is in the column space of $\bar{A}$, we have exactly $q^{\mathrm{rank}(\bar{A})}$ possible outputs. Clearly the number of possible network outputs has to be bigger than the number of possible outputs of the functions when fixing $u_{i_1}, \ldots, u_{i_{|S|}}$. ∎

Corollary 1 *Let A be a channel matrix over $\mathbb{F}_q$. A function f can be computed only if $q^{\mathrm{rank}(A)} \geqslant |\operatorname{range}(f)|$.*

In several of the functions we will examine in the following, the conditions in Proposition 1 are also sufficient. However, Example 1 provides a function where this is not the case.

B. m-state function

Each source observes a value in the set $\mathcal{A} = \{0, 1, \ldots, K-1\}$, and maps[3] it to one of m values, that we call "states". Let $\mathcal{S}$ be the set of possible states. The objective of the sink is to learn the state of each source. That is, evaluating the function f results in one of the m^N possible outputs. The m-state captures several interesting functions as special cases, that include

- *The membership function:* A membership criterion is specified, for example, whether the observed value belongs to a specific subset of the alphabet A. Each node determines whether it is a member, and sends a message from the set $\{\text{member}, \text{not member}\}$. In this case $m = 2$.
- *The identity function:* The sink wants to learn the message u_i that each source σ_i observes. In this case $m = K$.

The range of the m-state function is m^N and therefore from Corollary 1, for $A = (1 \ldots 1)$ it is necessary that $q \geqslant m^N$. This lower bound on q is also sufficient, and is achieved by the following scheme.

$$\begin{array}{ll} \text{Encoding} & x_i = m^{i-1} \cdot \mathcal{C}(u_i) \\ \text{Decoding} & f(u) = \left(\mathcal{C}^{-1}\left(\left\lfloor \frac{y}{m^0} \right\rfloor \bmod m\right), \ldots, \right. \\ & \left. \mathcal{C}^{-1}\left(\left\lfloor \frac{y}{m^{N-1}} \right\rfloor \bmod m\right)\right). \end{array}$$

where $\mathcal{C} : \mathcal{S} \to \{0, \ldots, m-1\} \in \mathbb{F}_q$ is a one-to-one mapping between the state values and elements in $\mathbb{F}_q$.

To understand why this scheme works, first note that if we express the symbol sent by every source σ_i in base m, it is a number of the form $\mathcal{C}(u_i)\ 0 \ldots 0$, i.e., a digit equal to value $\mathcal{C}(u_i)$ (which is less than m) followed by $i-1$ zeros. Now if we compute the sum of the input symbols in base m, note that no carry ever occurs and that only one input symbol is influencing the value of the i-th digit of the sum. This means that by expressing the network output in base m, we can reconstruct all the $C(u_i)$ that were sent by the sources. By inverting the mapping $\mathcal{C}$, we can therefore reconstruct the states in which the sources are.

Note that, for $m = K$, we get an upper bound on the required field size to calculate an arbitrary function f, since we can first calculate the identity function at the sink and then, having collected all the measurements, the function value.

C. Threshold function

Assume that $\mathcal{A} = \{0, 1\}$, the threshold function is defined as

$$f(u) = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} u_i \geqslant l \\ 0 & \text{otherwise.} \end{cases}$$

Note that the N-input OR and AND binary operations as well as the majority function are special cases of the threshold function. A scheme to compute the threshold function when $q > N$ is as follows:

$$\begin{array}{ll} \text{Encoding} & x_i = u_i \\ \text{Decoding} & f(u) = \mathbf{1}_{\{y \geqslant l\}}. \end{array}$$

We now show that any scheme to compute the threshold function over the $(1 \ldots 1)$ channel requires $q > N$. Let $x_i^j \in \mathbb{F}_q$ be the symbol sent by source σ_i when it observes $j \in \mathcal{A}$. Then for any collection of distinct indices $I = \{i_1, i_2, \ldots, i_k\}$ (with each $i_j \in \{1, 2, \ldots, N\}$), we have

$$\sum_{i \in I} x_i^0 \neq \sum_{i \in I} x_i^1. \tag{3}$$

To see why this is true, consider two distinct input vectors $u, v \in \mathcal{A}^N$. Vector u has 1 as input for all source indices in I. Additionally, it has 1 as input for a collection of $\max\{0, l - |I|\}$ other sources indices, say J, each of which is not in I. All the other inputs are set to 0. On the other hand, v has 1 as input for only the indices in J and all the other inputs are set to 0. Note that the number of inputs in J is always less than l. As a result we have $1 = f(u) \neq f(v) = 0$ and hence we require

$$\sum_{i \in I} x_i^1 + \sum_{i \in J} x_i^1 + \sum_{i \notin I \cup J} x_i^0 \neq \sum_{i \in I} x_i^0 + \sum_{i \in J} x_i^1 + \sum_{i \notin I \cup J} x_i^0$$
$$\implies \sum_{i \in I} x_i^1 \neq \sum_{i \in I} x_i^0.$$

From (3), the sink should receive a distinct symbol (in $\mathbb{F}_q$) from the network for each of the following input vectors:

$$(0\ 0 \ldots 0), (0\ 0 \ldots 1), (0\ 0 \ldots 1\ 1), \ldots, (1\ 1 \ldots 1).$$

Since the field size is q, this implies that $q > N$.

D. Histogram function

Let $\mathcal{A} = \{0, 1, \ldots, K-1\}$. The histogram function $f(u) : \mathcal{A}^N \to \{0, \ldots, N\}^K$ is defined as follows:

$$f(u) = v \quad \text{where } (v)_j = \sum_{i=1}^{N} \mathbf{1}_{\{u_i = j\}}.$$

[3]This map can potentially be different for every source node.

The histogram function is very useful, since it can be used to compute all symmetric functions of the node observations. For instance, all statistical functions fall in this category, such as the median, the mode and the maximum function.

When $q > N(N+1)^{K-2}$, this function can be computed with the following scheme:

$$\begin{aligned}\text{Encoding} \quad & x_i = \begin{cases} (N+1)^{u_i} & \text{if } u_i \leqslant K-2 \\ 0 & \text{otherwise} \end{cases} \\ \text{Decoding} \quad & f(u) = \Big(\left\lfloor \tfrac{y}{K^0} \right\rfloor \bmod (N+1), \ldots, \\ & \qquad \left\lfloor \tfrac{y}{K^{K-2}} \right\rfloor \bmod (N+1), \delta \Big), \\ & \text{where } \delta = N - \textstyle\sum_{i=0}^{K-2} \left\lfloor \tfrac{y}{K^i} \right\rfloor \bmod (N+1).\end{aligned}$$

The above scheme is similar to the scheme used for the identity function. In this case, every node sends a number of the form $10\ldots0$ in base $N+1$. If we compute the output of the network in base $N+1$, we will never incur any carry bits, since for each digit the number of 1's summed is at most N. This means that if we represent the output of the network in base $N+1$, the i-th digit is the number of sources that sent the message $10\ldots0$, a 1 followed by $i-1$ zeros. This gives the number of sources which observed the value i and thus the i-th coordinate of $f(u)$. This method can be used to compute the first $K-1$ coordinates of $f(u)$. The last coordinate can be easily computed by noting that the sum of all coordinates of $f(u)$ is always N.

Using the total number of possible histograms [12] and Corollary 1, for computing the histogram function we need

$$q \geqslant \binom{K+N-1}{K-1}.$$

When K is fixed and N grows, the above bound implies that $q = \Omega(N(N+1)^{K-2})$, which shows that the proposed scheme is within a constant factor from the minimum required q.

E. Maximum function

In this section, we are going to study the maximum function which can be derived from the histogram. We will show that despite the fact that this function has a much smaller range than the histogram, computing the maximum over the sum channel requires q to be as big (up to a constant factor) as that for the histogram. Let $\mathcal{A} = \{0, \ldots, K-1\}$. The maximum function is defined as

$$f(u) = \max_i u_i.$$

A scheme to compute the maximum function when $q \geqslant \frac{N^K - N}{N-1}$ is as follows:

$$\begin{aligned}\text{Encoding} \quad & x_i = c(u_i) = \textstyle\sum_{j=0}^{u_i - 1} N^j = \frac{N^{u_i}-1}{N-1} \\ \text{Decoding} \quad & f(u) = \textstyle\sum_{i=1}^{K} i \cdot \mathbf{1}_{\{c(k) \leqslant y \leqslant c(k+1)}.\end{aligned}$$

To understand why the above scheme works, first observe that

$$c(i+1) - N \cdot c(i) = \sum_{j=0}^{(i+1)-1} N^j - N \cdot \sum_{j=0}^{i-1} N^j = 1.$$

This implies that $N \cdot c(i) \leqslant c(i+1)$. Let $u_{\max} = \max_i u_i$ and $S = \{i | u_i = u_{\max}\}$. Then

$$y = \sum_{i=1}^{N} c(u_i) = |S| \cdot c(u_{\max}) + \sum_{i \notin S} c(u_i).$$

This implies that

$$\begin{aligned} y &\leqslant |S| \cdot c(u_{\max}) + (N - |S|) \cdot c(u_{\max}) \\ &\leqslant N \cdot c(u_{\max}) \leqslant c(u_{\max} + 1). \end{aligned}$$

Further,

$$y \geqslant |S| \cdot c(u_{\max}) \geqslant c(u_{\max}).$$

We will now prove a lower bound on the field size q required to compute the maximum function over the sum channel. Let x_i^j be the symbol sent by source σ_i when it observes $j \in \mathcal{A}$.

Let $S \subseteq \mathcal{A}^N$ be the collection of all vectors w such that

$$w_i \geqslant w_{i+1} \text{ for every } i \in \{1, 2, \ldots, N-1\}. \tag{4}$$

Let $u, v \in S$ be two distinct vectors. Consider the maximal set of indices $I \subseteq \{1, 2, \ldots, N\}$ such that for each $i \in I$, we have $u_i = v_i$. Let $\hat{u}$ ($\hat{v}$ respectively) be the input vector generated from u (v respectively) by setting all components in I to 0 and retaining all others.

From (4), there exists $j \geqslant 1$ such that

$$\begin{aligned} & u_i = v_i \text{ for every } i \in \{1, 2, \ldots, j-1\}, u_j \neq v_j, \\ & f(\hat{u}) = u_j, f(\hat{v}) = v_j \Longrightarrow f(\hat{u}) \neq f(\hat{v}). \end{aligned}$$

Thus the sink should receive different symbols for the input vectors $\hat{u}, \hat{v}$, and hence we have

$$\begin{aligned} & \sum_{i \in I} x_i^0 + \sum_{j \notin I} x_j^{u_j} \neq \sum_{i \in I} x_i^0 + \sum_{j \notin I} x_j^{v_j} \Longrightarrow \sum_{j \notin I} x_j^{u_j} \neq \sum_{j \notin I} x_j^{v_j} \\ & \Longrightarrow \sum_{i \in I} x_i^{u_i} + \sum_{j \notin I} x_j^{u_j} \neq \sum_{i \in I} x_i^{v_i} + \sum_{j \notin I} x_j^{v_j} \end{aligned}$$

where the last inequality follows since for each $i \in I$, $u_i = v_i$. Since $u, v \in S$ are arbitrary, this implies that the sink should receive a distinct symbol for every input vector in S.

The number of vectors in S is equal to the number of binary sequences of length $N+K-1$ with N zeroes and the rest all one. Thus we have $|S| = \binom{N+K-1}{K-1}$. Since the sink should receive a different symbol for every distinct input vector in S and the size of the field is q, to successfully compute the maximum function we need $q \geqslant \binom{N+K-1}{K-1}$ which is same as the lower bound for the histogram function.

IV. Computation over multiple timeslots

In the previous sections, we showed that some functions cannot be implemented with only one channel use if the field size q is not large enough. In this section, we will allow multiple uses of the network.

A. Non interactive computation

Here, we analyze codes that do not require any feedback from the sink to the sources. Our main observation is that computing a function using T timeslots in a network with underlying field $\mathbb{F}_q$ is equivalent to computing it in just one timeslot over a network that operates with symbols of an extension field $\mathbb{F}_{q^T}$. In the previous section, we devised schemes and proved bounds for computation over networks operating over a prime field. To study computation over multiple timeslots, we need to extend those results to general finite fields. We illustrate this via the example of threshold functions.

Example 2 Let $q = p^T$ for some prime p and let f be the threshold function. Then equation (3) in Section III-C provides a necessary condition for computation and implies that if we create a vector $v \in \mathbb{F}_q^N$ such that the i-th component $v_i = x_i^0 - x_i^1$, then we have that for any $I \subseteq \{1, 2, \ldots, N\}$,

$$\sum_{i \in I} v_i \neq 0 \tag{5}$$

where the sum is over the field $\mathbb{F}_q$. Since $q = p^T$, each element v_i can be viewed as a vector of length T over $\mathbb{F}_p$. Let $v_i = (v_{i1}, v_{i2}, \ldots, v_{iT})$. Given characteristic p, we want to find the minimum T such that (5) holds. It can be easily checked that $T = \lceil N/(p-1) \rceil$ suffices for any p. It remains to find if this is optimal. For any p, consider the following system of T polynomial equations in N variables, with each polynomial in $\mathbb{F}_p[x_1, x_2, \ldots, x_N]$:

$$\sum_{i=1}^{N} v_{ij} \cdot x_i^{p-1} = 0 \quad \forall\ j \in \{1, ..., T\}.$$

It can be verified that if there exists any non-trivial solution to the above system of polynomial equations, then it will imply that the vectors $v_1, v_2, \ldots, v_N$ violate (5). From the Chevalley-Warning theorem [13, Theorem 3, Pg. 5], there exists a non-trivial solution if $T < N/(p-1)$. Thus, the minimum T such that the threshold target function f can be computed in a network operating over the finite field of size $q = p^T$ is equal to $\lceil N/(p-1) \rceil$.

We now show that by appropriately using the sum channel multiple times, we can create (at least) two different types of equivalent channels, that are interesting (from a function computation viewpoint) in their own merit.

1) Linear map channel: Let $\mathcal{A} = \mathbb{F}_q$, and let A be the sum channel. We can implement the channel $y = B \cdot u$, where B is any desired $M_B \times N$ matrix, and y is the vector that the sink collects after $T = M_B$ timeslots, with the following scheme:

$$\begin{array}{ll} \text{Encoding} & x_i[t] = (B)_{t,i} \cdot u_i \\ \text{Decoding} & f(u) = (y[1] \ldots y[M_B]). \end{array}$$

For example, if we select B to be any $N \times N$ invertible matrix (i.e., $T = N$), we can calculate the identity function over $\mathbb{F}_q$. Moreover, if we have any information in advance about the sensor measurement structure, we can select an appropriate matrix B that allows to reconstruct the measurements u, potentially using $T \ll N$. This is the case in compressed sensing: if we know for example that the vector u is sparse, we can select a "good" matrix B, so that we can efficiently recover u [14]. Thus our formulation provides a novel application of compressed sensing methods.

2) Sum vector channel: Let $\mathcal{A} = \mathbb{F}_q^T$. Using the sum channel T times allows to also compute $y = b_1 \cdot u_1 + \ldots + b_N \cdot u_N$ where $b_i \in \mathbb{F}_q$, an arbitrary linear combinations of $T \times 1$ vectors produced at the sources.

To achieve this result, we use the following scheme:

$$\begin{array}{ll} \text{Encoding} & x_i[t] = b_i \cdot (u_i)_t \\ \text{Decoding} & f(u) = (y[1] \ldots y[T]). \end{array}$$

Since the sink receives a linear combination of the vectors inserted by the sources, we can use this as a building block to translate the coding schemes that we developed in [11] to this framework.

B. Interactive computation

In this section, we argue that using interactive communication may allow us to construct simpler codes. We will assume that the network output is not presented only to the sink, instead it is received by all sources as well. This assumption is realistic in many cases; for example, protocols used to compute averages and sums rely on flooding and the objective of their operation is to disseminate the sum (or the average) to all network nodes, see for example [1].

We will illustrate that interaction can lead to simpler protocols, by studying as an example the maximum function. We note that function computation under this model is very close to the work in [15], and thus the communication protocols and bounds developed there can be translated to our framework.

1) Maximum function: To illustrate the possibilities in code design when using feedback, we study the computation of the maximum function (as defined in section III-E). We show that if the source alphabet is $\mathcal{A} = \{0, 1, \ldots, K-1\}$ and $q > (K-1) \cdot N$, the maximum function can be computed in $K+1$ timeslots. Further, if the difference between the maximum ($u_{\max}$) and the minimum ($u_{\min}$) component of the input vector is at most u_δ, then the maximum function can be computed in $u_\delta + 2$ timeslots.

Let $d[t] = \lceil y[t-1]/N \rceil$, the scheme is as follows:

$$\begin{array}{ll} \text{Encoding} & x_i[t] = \begin{cases} u_i & \text{if } t = 1 \text{ or } u_i > d[t] \\ d[t] & \text{otherwise} \end{cases} \\ \text{Decoding} & f(u) = y[t_f]/N \text{ such that } y[t_f] = y[t_f + 1]. \end{array}$$

We now analyze the above scheme. Let $S[t] = \{i | u_i > d[t]\}$ be the set of sources that transmit their observed value as codeword at time t. Then we have

$$\begin{aligned} d[t+1] - d[t] &= \left\lceil \frac{\sum_{i \in S[t]} u_i + (N - |S[t]|) \cdot d[t]}{N} \right\rceil - d[t] \\ &= \left\lceil \frac{1}{N} \sum_{i \in S[t]} (u_i - d[t]) \right\rceil. \end{aligned}$$

If $d[t] < u_{\max}$, then there is at least one source in $S[t]$ and the value of this node is bigger than $d[t]$. Thus, $d[t+1] \geqslant d[t]+1$. On the other hand, when $d[t] = u_{\max}$ the set $S[t]$ is empty and therefore $d[t+1] = d[t]$. This implies that $d[t]$ will keep increasing in each timeslot until it equals $u_{\max}$. Since $d[t] \geqslant u_{\min}$ for all t, the scheme requires at most $u_{\max} - u_{\min} + 2$ timeslots to compute the maximum function. Thus, the convergence time is at most K and hence does not grow as the number of sources N increases.

V. Computation over linear channel matrices

In the previous sections we focused on schemes for the sum channel matrix $A = (1 \dots 1)$. Here we introduce an ordering of linear channels, such that if a function can be computed over a specific channel, it can also be computed over all channels with a higher ordering.

Consider a channel matrix A and observe that each source σ_i can multiply the symbol it sends at each timeslot by a constant. Moreover, the sink can multiply the vector it receives at each timeslot by an arbitrary matrix. These two observations lead to the following definition:

Definition 1 *If given two channel matrices A and A', we can find matrices B and D such that $A = B \cdot A' \cdot D$ where $D = \mathrm{diag}(d_1, \dots, d_N)$, then we write $A \preccurlyeq A'$.*

Proposition 2 *If a function f is computable over a channel A, then it is computable over any A' such that $A \preccurlyeq A'$.*

Proof: Since $A \preccurlyeq A'$, we know that there exists a matrix B and a matrix $D = \mathrm{diag}(d_1, \dots, d_N)$ such that $A = B \cdot A' \cdot D$. Each source maps the observed values to $\tilde{x}_i$ according to the code used to compute f over channel A and sends $x_i = d_i \cdot \tilde{x}_i$. Thus the vector sent over the channel is $x = D \cdot \tilde{x}$ where $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_N)$. The sink receives the vector $y = A' \cdot D \cdot \tilde{x}$ and creates the vector $\tilde{y} = B \cdot y = B \cdot A' \cdot D \cdot \tilde{x} = A \cdot \tilde{x}$. Since $\tilde{x}$ is a codeword that allows to reconstruct the value of f over the channel A, the sink can correctly compute the function. ■

The following proposition identifies channel matrices for which we can use the codes for the channel $(1 \dots 1)$.

Proposition 3 *Let A an $M \times N$ channel matrix such that there exists an $M \times 1$ vector b over $\mathbb{F}_q^M$ that is not orthogonal to each column of A, then $[1 \dots 1] \preccurlyeq A$.*

Proof: Since b is not orthogonal to any column of A each entry of $b^\top \cdot A$ is non-zero. This implies that

$$b^\top \cdot A \cdot \mathrm{diag}((b^\top \cdot A_1)^{-1}, \dots, (b^\top \cdot A_N)^{-1}) = [1 \dots 1]$$

where A_i is the i-th column of A. ■

Using the above propositions, we now provide some example channels where we can use codes for the $(1 \dots 1)$ channel.

Example 3 Let $q > 2^M$. Consider any matrix A over $\mathbb{F}_q$ such that $(A)_{\mathrm{ij}} \in \{0, 1\}$ and each column contains at least one non-zero entry. Then we have $[1 \dots 1] \preccurlyeq A$ since the vector $[2^0 2^1 \dots 2^{M-1}]$ is not orthogonal to every column of A.

Example 4 Any matrix A over $\mathbb{F}_q$ with exactly one non-zero entry per column is $[1 \dots .1] \preccurlyeq A$. Indeed the vector $[1 \dots 1]$ is not orthogonal to all columns of A.

Example 5 For any $1 \times N$ matrix A with non zero entries, we have $[1 \dots 1] \preccurlyeq A$. Indeed the vector $[1]$ is not orthogonal to each column of A.

VI. Conclusion

In this paper, we formulated the problem of function computation over linear deterministic channels. Such channels arise for example, in networks employing fixed network coding operations, or in networks that perform specific in-network processing to compute a linear transformation of the source measurements, such as the average value. We proposed several models for this problem, calculated upper and lower bounds on the required field size q and the number of channel uses T, and derived optimal code designs for a number of functions.

References

[1] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, Oct. 2003, pp. 482–491.

[2] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The tenet architecture for tiered sensor networks," *To Appear in the ACM Transactions on Sensor Networks (TOSN)*, 2010.

[3] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," in *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Dec. 2002, pp. 131–146.

[4] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2004, pp. 250–262.

[5] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communication*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[6] ——, "Toward a theory of in-network computation in wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98–107, Apr. 2006.

[7] A. Ramamoorthy, "Communicating the sum of sources over a network," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Jul. 2008, pp. 1646–1650.

[8] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing," in *Allerton Annual Conference on Communications, Control, and Computing*, Sep. 2008.

[9] V. Doshi, D. Shah, M. Medard, and S. Jaggi, "Distributed functional compression through graph coloring," in *Proceedings of the Data Compression Conference (DCC)*, Mar. 2007, pp. 93–102.

[10] N. Ma, P. Ishwar, and P. Gupta, "Information-theoretic bounds for multiround function computation in collocated networks," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Jun. 2009, pp. 2306–2310.

[11] N. Karamchandani, L. Keller, C. Fragouli, and M. Franceschetti, "Function computation via subspace coding," *EPFL Technical Report ARNI-REPORT-2010-001, http://infoscience.epfl.ch/record/143339*, Jan. 2010.

[12] R. P. Stanley, *Enumerative Combinatorics*. Cambridge University Press, 1997, vol. 1.

[13] J.-P. Serre, *A Course in Arithmetic*. Springer, 1973.

[14] S. Howard, A. Calderbank, and S. Searle, "A fast reconstruction algorithm for deterministic compressive sensing using second order reed-muller codes," in *Proceedings of the Conference of Information Sciences and Systems (CISS)*, Mar. 2008, pp. 11–15.

[15] A. K. Dhulipala, C. Fragouli, and A. Orlitsky, "Silence-based communication," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 350–366, Jan. 2010.