

# Finding the $K$ Largest Metrics in a Noisy Broadcast Network

Chengzhi Li, Huaiyu Dai and Husheng Li

## Abstract

Distributed computing in an  $N$ -node noisy broadcast network is considered. Each node holds an  $n$ -bit integer as an input instead of a binary input assumed by most work in literature. The goal is for all the nodes to find out the  $K$  largest values with fault tolerance  $Q$ . We focus on protocol designs for the  $K = o(N^\beta)$  scenario, and investigate the communication complexity, i.e., the total number of broadcasts, with respect to various parameters:  $N, n, K$ . In particular, we show that the goal can be achieved by an oblivious protocol of complexity  $O(nN \log \frac{K}{Q})$ , admitting linear growth with  $n$  and  $N$  and sublinear growth with  $K$ .

## I. INTRODUCTION

How to perform reliable computation among distributed nodes through noisy communication is a critical issue for emerging wireless networks. In particular, noisy channels impose challenges to traditional designs of communication protocols. If nodes still follow the noiseless-channel protocols, they will lose consensus on execution due to discrepancy in individual interpretations of the transmission history, which almost surely leads to a failure.

While in certain applications such as continuous field monitoring [1], the classical block coding approach of information theory can be pursued, in many scenarios it is not directly applicable, either due to the fact that each node holds only finite bits [3], or the dynamic nature of interactive communication [9]. There is also an error propagation issue for the latter. In general increase in complexity is inevitable for noisy channels even if constant error tolerance is allowed. For a noiseless-channel protocol with communication complexity (in terms of the total number of communication bits) of  $m$ , one could repeat transmission of each bit  $O(\log(m/Q))$  times and take the majority of the results; this approach, termed standard amplification in literature, leads to a noisy-channel protocol of complexity  $O(m \log(m/Q))$  with error tolerance  $Q$ . Obtaining protocols with less increase in complexity, especially  $O(m)$ , is highly non-trivial, and in some cases impossible.

For noisy channels, it is also required that the protocol be *oblivious*, i.e., the transmission schedule of nodes is pre-determined, independent of initial inputs and the transmission history; this avoids contention and out-of-order execution.

In this paper, we will focus on a noisy broadcast network. Communication complexity of distributed computing in this setting was first considered by El Gamal in [2], where each of the  $N$  nodes, holding one binary input, can broadcast to all others through independent binary symmetric channels. Gallager presented a protocol of complexity  $O(N \log \log N)$  (suitable to compute any functions) [3], which was shown optimal for the identity function (i.e., the entire input) in [4]. In [5], Yao posed the question whether there exist nontrivial Boolean functions that can be computed with  $O(N)$  broadcasts; this is answered in the affirmative for the threshold functions in [6] with the independent and identically distributed (i.i.d.) random noise model, and in [8] for the OR function with the more realistic (and more general) adversarial model (where a "benign" adversary is allowed to arbitrarily reduce the error probability of each link at the

Chengzhi Li and Huaiyu Dai are with the department of electrical and computer engineering, North Carolina State University, Raleigh, NC, (email: cli3, hdai@ncsu.edu)

Husheng Li is with the department of electrical engineering and computer science, the University of Tennessee, Knoxville, TN, (email: husheng@ece.utk.edu)

This work was supported in part by the National Science Foundation under Grant CCF-0515164, CNS-0721825 and CCF-0830462.

beginning, or even dynamically cancel any errors on the fly, so as to prevent the protocols from exploiting the stochastic regularities of the previous i.i.d. model).

While most work in literature focuses on computing Boolean functions of binary variables, in this paper, we extend the study to find the  $K$  largest integer values ( $n$ -bit each), and devise a protocol of complexity  $O(nN \log(K/Q))$  with error tolerance  $Q$ . For constant  $K$ , this is another nontrivial function admitting linear complexity (while dealing with non-binary input and output).

Our study is motivated by distributed scheduling in wireless networks. A typical scheduling policy is based on the comparison of certain priority weights, e.g. the product of queue backlog and success probability in [11] or packet deadline in [12]. While in some circumstances, only the node holding the largest metric obtains the right of transmission, it is more general to consider the  $K$  largest metrics simultaneously. For example, there may not be a clear cut in metric comparison; if there are  $K > 1$  communication channels available, it may be more efficient to schedule multiple transmissions simultaneously. As another application, the scheduler may need to consider multiple factors to make a decision, including the queue length, packet deadline, link condition, etc. A possible scheduling policy is to find out a set of  $K_1$  nodes holding largest values in one factor and another set of  $K_2$  nodes holding largest values in another factor. The node(s) in the intersection of these sets are given the right to transmit.

The remainder of this paper is organized as follows. the system model and some preliminaries are given in Section II. The special case, computing the max function, is first treated in Section III. General protocols for finding the  $K$  largest integers in a noisy broadcast network are presented in Section IV. Some discussion about the computation of the mother function, the identity function, is given in Section V. The conclusion and future directions are provided in Section VI.

## II. SYSTEM MODEL AND PRELIMINARIES

### A. System Model

We assume a broadcast network of size  $N$ . Each node  $i \in \{1, \dots, N\}$  holds an  $n$ -bit integer  $x_i$  and the goal is for all nodes to find the  $K$  largest values (together with their holders), according to a protocol known to all. The network is synchronized and each node broadcasts one bit <sup>1</sup> in a predetermined schedule. Each broadcast bit is received and identified by all other nodes, through binary symmetric channels. We adopt the adversarial noise model, so the error probabilities on each link and at each slot need not be identical, but just independent and bounded above by a constant  $\epsilon$ .

Given an upper bound on the noise level  $\epsilon$ , and an error tolerance level  $Q$  for the protocol, we would like to design protocols that are efficient in communication complexity, i.e., the total number of broadcasts. As a common practice for studies in this area, we will focus on the scaling law of this metric with respect to various parameters:  $N$ ,  $n$ ,  $K$ . Clearly we prefer linear or sub-linear growth, rather than super-linear growth. Note that the exact values of  $\epsilon$  and  $Q$  are not important. By the standard amplification approach mentioned in the introduction, any smaller *constant* value only incurs a constant multiplicative factor in the final complexity result. In our following analysis, therefore,  $\epsilon$  can be assumed arbitrarily small (but constant) if needed.

### B. Overview of Results

Depending on different relations between the system parameters, simpler protocols exist in some scenarios, and sometimes more efficient protocols can be designed with some efforts. We start by considering different values of  $K$ .

- $K = o(N^\beta)$  ( $0 < \beta < 2/3$ ): This is our focus. We will start from  $K = 1$ , which admits simpler designs and yet provides some insights into the more general scenarios.

<sup>1</sup>This is a common assumption in the study of communication complexity, since the focus is on total number of communication bits.

- $K = 1$  (Section III): We differentiate two cases,  $n = \Omega(\log N)$  and  $n = o(\log N)$ , the former of which admits a simple  $O(nN)$  protocol  $\mathcal{P}_1$  that applies to a general  $K$  as well. For the latter, we present a broadcast protocol  $\mathcal{P}_2$  with complexity  $O(\max(n, \log \frac{1}{Q})N)$ .
- $K = \omega(1)$  (Section IV): We focus on the case  $n = o(\log N)$ . Obviously, we can repeat the protocol for the max function above with error tolerance  $Q/K$  for  $K$  times to obtain one with complexity of  $O(\max(n, \log \frac{K}{Q})KN)$ . We endeavor to develop a protocol  $\mathcal{P}_3$  of complexity  $O(nN \log \frac{K}{Q})$ , which is uniformly superior for all  $N, n$  and  $K = o(N^\beta)$ .
- $K = \Omega(N^\beta)^2$ : This is less interesting. Sorting algorithms in literature [15, 16] can be transformed into a noisy broadcast protocol following a similar approach given below, with complexity  $O(nN \log \frac{N}{Q})$ . From discussion of Section V, it is more efficient to directly evaluate the identity function in this scenario.

Note: As a common approach, we designate a head node in the network,  $H_0$ , to coordinate the protocol execution. This node takes (much) more responsibility *besides* its normal role as other nodes in the protocols. The final step of our protocols is to have  $H_0$  broadcast the results to the whole network (with the desired fidelity). This typically suffices the need; e.g., the nodes who hold the largest values get the right to transmit and others hold back. If the corresponding nodes' identities are also desired, we can allow extra  $O(K \log N)$  broadcasts either from the head node or the winner nodes, which are negligible compared to the main results above.

The following notations are used in the remainder of this paper:

- $C_N$ : the complexity of a protocol.
- $e_N$ : the probability of a protocol fails.

### C. Preliminaries

We first introduce two known results. The first is a generalization of the Chernoff bound.

#### 1) Hoeffding inequality [13]:

*Lemma 1:* Let  $X_i$  ( $1 \leq i \leq n$ ) be  $n$  i.i.d. random variables over the interval  $[a, b]$ . For the sum of these variables  $S = \sum_{i=1}^n X_i$ , we have the inequity

$$P(S - E(S) \geq n\delta) \leq e^{-\frac{2n\delta^2}{b-a}}, \quad \delta > 0.$$

#### 2) Constant-rate, Constant-fraction Minimum-distance Codes [4, 14]:

*Lemma 2:* For  $\gamma \in (0, \frac{1}{2})$  there is an integer  $C_1 = C_1(\gamma)$  such that for each positive integer  $n$ ,  $\forall C \geq C_1$ , there is a binary code  $\Gamma_n$  of size  $2^n$  and length  $Cn$ , such that for all  $v, w \in \Gamma_n$  with  $v \neq w$ , the hamming distance  $d(v, w) \geq \gamma Cn$ .

Based on the above facts, the following result can be derived, which will be extensively used in our protocol design.

*Lemma 3:* With  $O(n)$  broadcasts over a binary symmetric channel (with error probability bounded above by a constant  $\epsilon$ ), an  $n$ -bit integer can be correctly received by each node with probability of at least  $1 - 2^{-\alpha n}$ ,  $\forall \alpha > 0$ ; with  $O(\max(n, \log N))$  broadcasts it can be correctly received by each node with probability at least  $1 - \frac{1}{N^\alpha}$ ,  $\forall \alpha > 0$ .

*Proof:* The proof follows the lines of Observation 2.1 in [8]. According to Lemma 2, we encode the  $n$ -bit integer by a codeword  $v \in \Gamma_n$  with length  $Cn$ . A receiver gets a noisy copy  $v'$  of  $v$  and decodes it as  $w \in \Gamma_n$  which minimizes  $d(v', w)$ . The decoding error is upper bounded by  $P_r(d(v', v) \geq \frac{1}{2}\gamma Cn)$ ,

<sup>2</sup>We assume  $K < N/2$ ; the case  $K > N/2$  is equivalent to finding the  $N - K$  smallest values.

which, by Lemma 1, is bounded above by  $\exp(-2(\frac{1}{2}\gamma - \epsilon)^2 Cn)$ . Setting  $\gamma = 4\epsilon$ , this value is at most  $2^{-\alpha n}$  for  $C \geq \alpha \frac{1}{\epsilon^2}$ ,  $\forall \alpha > 0$ .

Further more, if  $n \geq \log N$ ,  $2^{-\alpha n} \leq \frac{1}{N^\alpha}$ . Therefore taking  $O(n)$  broadcasts, the integer can be received correctly with probability at least  $1 - \frac{1}{N^\alpha}$ . If  $n < \log N$ , we can pad it to the length  $\log N$ . ■

### III. PROTOCOL FOR COMPUTING THE MAX FUNCTION

In this section, two protocols  $\mathcal{P}_1$  and  $\mathcal{P}_2$  are proposed to evaluate the max ( $K = 1$ ) function with fault tolerance  $Q$ , one for  $n = \Omega(\log N)$  and the other for  $n = o(\log N)$ .

#### A. $n = \Omega(\log N)$

##### 1) Protocol $\mathcal{P}_1$ :

- (1) Each node encodes its value ( $n$ -bit) into a codeword of length  $O(n)$  and broadcasts the codeword once. The head node  $H_0$  decodes all the codewords and finds out the largest value.
- (2)  $H_0$  encodes the largest value into a codeword of length  $O(n)$  and then broadcasts the codeword.

##### 2) Analysis:

*Theorem 1:* The complexity of  $\mathcal{P}_1$   $C_N^{\mathcal{P}_1} = O(nN)$ .

*Proof:* It's easy to check

$$C_N^{\mathcal{P}_1} = O(nN) + O(n) = O(nN).$$

■

*Theorem 2:* The error probability of  $\mathcal{P}_1$   $e_N^{\mathcal{P}_1} \leq Q$ .

*Proof:* According to Lemma 3 and the independence of the communication links, at the end of phase 1 all the codewords are decoded correctly by  $H_0$  with probability at least  $1 - \frac{1}{N^\alpha}$  for some  $\alpha > 0$ . Similarly, at the end of phase 2 the probability that all the other nodes correctly decode the max value broadcasted from  $H_0$  is at least  $1 - \frac{1}{N^\alpha}$ . Thus,  $e_N^{\mathcal{P}_1} \leq \frac{2}{N^\alpha} < Q$  for suitable  $\alpha$ . ■

*Remark 1:* The protocol  $\mathcal{P}_1$  also applies to the general  $K(> 1)$  scenario after simple modifications. Namely, after decoding the  $N$  codewords,  $H_0$  finds out the  $K$  largest values and concatenates them into a word, which is then encoded into a codeword of length  $O(nK)$  and broadcasted. Following the same analysis, the complexity is still  $O(nN)$  and the error tolerance is bounded above by  $Q$ . The same conclusion holds even for the identity function.

#### B. $n = o(\log N)$

When  $n = o(\log N)$ , by the second part of Lemma 3, we can modify  $\mathcal{P}_1$  to obtain a protocol of complexity  $O(N \log N)^3$ . In the following, we present a protocol  $\mathcal{P}_2$  with error tolerance  $Q$  and complexity  $O(\max(n, \log \frac{1}{Q})N)$ .

1) *Protocol:* The intuition behind the protocol  $\mathcal{P}_2$  comes from the sports competition [8, 16]. Assume  $N$  teams participate the game. In the first round,  $N$  teams are partitioned into groups of equal size and each team competes with the other teams in the same group. The winners go into the next round and are grouped again. The competition continues until the champion stands out in the final. For simplicity, we assume  $N = 4^k$  for some integer  $k$ . The protocol  $\mathcal{P}_2$  is defined recursively (on the size of the network) as follows.

*Protocol  $\mathcal{P}_2$ :* Assume that  $N$  nodes are divided into  $N/4$  groups, each of size 4.

- (1) Each node encodes its integer into a codeword of length  $O(\max(n, \log \frac{1}{Q}))$  and broadcasts it.

<sup>3</sup>A naive protocol with complexity  $O(nN \log(\frac{nN}{Q}))$  would be to repeat each bit  $O(\log(\frac{nN}{Q}))$  times and decode it by taking the majority.

- (2) For each group,  $H_0$  compares the four values and determines the index of the maximum value. And then  $H_0$  concatenates the  $N/4$  indices (2 bits for each) into a word, which is encoded with an error correcting code of size  $O(N/4)$  and broadcasted.
- (3) Let  $J$  be the set of winning nodes in (2). The protocol proceeds by executing  $\mathcal{P}_2$  on  $J$  for 3 times. In each time, the head node  $H_0$  obtains a nominal max value (the result from one execution of the protocol on  $N/4$  nodes); it takes the majority as the result for the protocol on  $N$  nodes.

To complete the description of the protocol, consider the execution on a group of  $N = 4$  nodes. First, phase 1 above is executed. Then  $H_0$  decodes each codeword and determines the largest one.

$\mathcal{P}_2$  ends up with  $H_0$  knowing the maximal value. To share it with others,  $H_0$  employs a code of length  $O(\log N)$ . The protocol  $\mathcal{P}_2$  is non-oblivious. In Section IV-B, we'll discuss a general approach to translate a non-oblivious protocol to an oblivious one.

2) *Analysis:*

*Theorem 3:* The complexity of  $\mathcal{P}_2$   $C_N^{\mathcal{P}_2} = O(\max(n, \log \frac{1}{Q})N)$ .

*Proof:* Clearly, when  $N = 4$ ,  $C_4^{\mathcal{P}_2} = 4O(\max(n, \log \frac{1}{Q}))$ . Assume that  $C_m^{\mathcal{P}_2} \leq 4mO(\max(n, \log \frac{1}{Q}))$ , then<sup>4</sup>

$$\begin{aligned}
C_{4m}^{\mathcal{P}_2} &= 4mO(\max(n, \log \frac{1}{Q})) + 3C_m^{\mathcal{P}_2} \\
&\leq 16mO(\max(n, \log \frac{1}{Q})) \\
&= 4 \cdot 4mO(\max(n, \log \frac{1}{Q})).
\end{aligned}$$

Therefore we conclude  $C_N^{\mathcal{P}_2} = O(\max(n, \log \frac{1}{Q})N)$ . ■

*Theorem 4:* The error probability of  $\mathcal{P}_2$   $e_N^{\mathcal{P}_2} \leq Q$ .

*Proof:* By Lemma 3, it is easily proved that in a group of 4,  $H_0$  fails to get the correct max value with probability at most  $4Q^2$ , which is less than  $Q$  for sufficiently small  $Q$ . In particular, we have  $\epsilon_4^{\mathcal{P}_2} < Q$ . Assume  $\epsilon_{N/4}^{\mathcal{P}_2} < Q$ , then

$$\epsilon_N^{\mathcal{P}_2} \leq 4Q^2 + 3(\epsilon_{N/4}^{\mathcal{P}_2})^2 \leq 7Q^2 < Q.$$
■

*Remark 2:* While we ignore the broadcast by  $H_0$  in the above analysis, both its complexity  $O(N)$  and incurred error  $\frac{\log N}{N^\alpha}$  (for sufficiently large  $\alpha$ ) are negligible.

#### IV. PROTOCOLS FOR FINDING THE $K$ LARGEST VALUES

As we discussed in Section III-A,  $\mathcal{P}_1$  also applies to the general  $K$  scenario when  $n = \Omega(\log N)$ . Therefore we focus on  $n = o(\log N)$  in this section. A protocol  $\mathcal{P}_3$  with  $O(nN \log \frac{K}{Q})$  broadcasts is presented to find out the  $K$  ( $K = o(N^\beta)$ ) largest values with error tolerance of  $Q$ .

Intuitively,  $\mathcal{P}_3$  exploits the idea of "heapsort", which is considered as one of the most efficient sorting methods. The "heapsort" algorithm [15, 16] can be briefly described as follows:

- (1) build a binary tree on the given  $N$  numbers, with the values at the parent nodes no smaller than those at the children;
- (2) extract the root, which is the largest number;
- (3) rebuild the heap;
- (4) repeat (2), (3) until all the  $K$  largest numbers are found.

<sup>4</sup>Note that we induct on  $N$ .

## A. Protocol $\mathcal{P}_3$

1) *Protocol*: The protocol  $\mathcal{P}_3$  simulates the “heapsort” algorithm in a noisy network by three phases: build up an initial heap in phase 1, rebuild the heap in phase 2 and distribute the result in phase 3. In phase 1 we apply the truncation procedure to decrease the complexity, following the ideas in [16]. These three steps are described in detail below.

Initialization of *Protocol*  $\mathcal{P}_3$ <sup>5</sup>:  $H_0$  maintains a local binary tree, with each input element arbitrarily assigned to a node on the tree.  $H_0$  also keeps a counter for each element, which is initialized to 0. Define a constant  $q$ <sup>6</sup>. Each element encodes its integer into a codeword of length  $O(\max(n, \log \frac{3}{q}))$ <sup>7</sup>. Set  $S_0 = \{1, 2, \dots, N\}$ ,  $t = \frac{9}{v}(\log(\frac{K}{Q}) + 2)$  and  $d = \frac{108}{v} \log(\frac{N}{Q})$  for some  $v > \frac{1}{3}$ <sup>8</sup>.

Phase 1 of *Protocol*  $\mathcal{P}_3$ : Denote by  $S_r$  the set of elements remaining in the tree in round  $r$ .

- **Competition**:  $H_0$  partitions these elements into triplets according to their positions in the tree: if  $r$  is even (odd), each triplet consists of an element at an even (odd) level of the tree and its two children. Each element broadcasts its codeword *once*.  $H_0$  decodes these codewords and compares the three integers of each triplet<sup>9</sup>. If the element originally placed on the parent node wins in a triplet, its counter is increased by 1; if it loses but its counter is larger than 0, its counter is decreased by 1; if it loses and its counter is 0, it switches the place with the winner, whose counter is set to 0.
- **Truncation**: After  $r = jt (j = 1, 2, \dots, \lfloor \frac{\log N}{3t} \rfloor)$  rounds, elements at the leaves of the current tree are discarded.  $H_0$  labels each element of  $S_r$  as follows: the elements remaining in the tree are labeled with ‘1’ and the others with ‘0’. Then,  $H_0$  distributes these labels through  $O(|S_r|)$  broadcasts. Elements labeled with ‘1’ go into the next round.

Phase 1 ends after  $d$  rounds and then phase 2 starts.

Phase 2 of *Protocol*  $\mathcal{P}_3$ :

- $H_0$  extracts the value from the root and the root becomes a vacant node (called vacancy below).
- $H_0$  concatenates the indices of the vacancy’s children into a word, which is encoded into a codeword of length  $O(\log N)$  and broadcasted. The vacancy’s children encode their integers into codewords with size  $O(\log N)$  and broadcast them.  $H_0$  decodes the codewords and compares the two integers, moving up the winner to fill the vacancy in the tree.
- repeat (2) until the vacancy moves to the the bottom level of the tree, which takes  $\log N$  steps.
- repeat (1)-(3) for  $K - 1$  times.

After phase 2,  $H_0$  obtains  $K$  values, the nominal  $K$  largest values.

Phase 3 of *Protocol*  $\mathcal{P}_3$ :

$H_0$  concatenates the  $K$  values into a word and encodes it into a codeword of length  $O(\max(Kn, \log N))$ , then broadcasts it.

### 2) Analysis:

*Theorem 5*: The complexity of  $\mathcal{P}_3$   $C_N^{\mathcal{P}_3} = O(nN \log \frac{K}{Q})$ .

*Proof*: In the first  $\frac{\log N}{3}$  rounds of phase 1, due to the truncation, the number of broadcasts for distributing the codewords is at most

$$O(n)(N + (N + 1)/2 + (N + 1)/4 + \dots)t = O(nN \log \frac{K}{Q}).$$

<sup>5</sup>To avoid confusion between nodes of the network and nodes of the computing structure, the binary tree, nodes of the network are identified with the input elements they hold.

<sup>6</sup>interpreted as the desired probability that the largest element in a triplet wins

<sup>7</sup>This is to guarantee that the largest element in a triplet wins with probability  $q$ .

<sup>8</sup>interpreted as the average moving speed of an element on the tree

<sup>9</sup>As discussed above, the true largest element of a triplet wins with probability  $q$ .

And the number of broadcasts for  $H_0$  distributing the labels is  $O(\sum_{i=1}^{\lfloor \log \frac{\log N}{3^i} \rfloor} (|S_{it}|)) = O(N)$ . The number of elements left in the tree after  $\frac{\log N}{3}$  rounds is  $N/2^{\lfloor \frac{\log N}{3^i} \rfloor}$ , which is  $o(N)$  for constant  $Q$ . Therefore the complexity of phase 1 is  $O(nN \log(\frac{K}{Q}))$ . In phase 2, each re-heaping takes  $O(\log N \log N)$  broadcasts, so altogether  $O(K \log N \log N)$  broadcasts are needed, which is  $O(N)$  for  $K = o(N^\beta)$ . The complexity of phase 3 is  $O(\max(Kn, \log N))$ , which is  $O(N)$  according to the assumptions that  $K = o(N^\beta)$  and  $n = o(\log N)$ . Summing up the complexity of phase 1, 2 and 3, we conclude  $C_N^{\mathcal{P}_3} = O(nN \log \frac{K}{Q})$ .  $\blacksquare$

*Theorem 6:* The error probability of  $\mathcal{P}_3$   $e_N^{\mathcal{P}_3} \leq Q$ .

The protocol  $\mathcal{P}_3$  errs if

- (1) not all the  $K$  largest elements are at their correct positions in the heap<sup>10</sup> at the end of phase 1;
- (2) the heap is not rebuilt successfully for at least one time in phase 2;
- (3) the codeword from  $H_0$  is not decoded correctly by the whole network at the end of phase 3.

For the convenience of description we analyze the probability of the last two errors first and then discuss the first error. To rebuild the heap once in phase 2, all the nodes decode the codeword from  $H_0$  correctly in  $\log N$  steps with probability at least  $1 - \frac{\log N}{N^\alpha}$  for some  $\alpha > 0$  and  $H_0$  decodes the codewords from at most  $2 \log N$  elements correctly with probability at least  $1 - \frac{2 \log N}{N^\alpha}$  according to Lemma 3. Thus, the probability of the second error is at most  $K(\frac{\log N}{N^\alpha} + \frac{2 \log N}{N^\alpha})$ . Similarly, the probability of the last error is bounded by  $\frac{1}{N^\alpha}$ . Consequently, the probability of the last two errors is  $K(\frac{\log N}{N^\alpha} + \frac{2 \log N}{N^\alpha}) + \frac{1}{N^\alpha} < Q/4$  for suitable  $\alpha$ .

As for the first error, the  $K$  largest elements do not stay at their correct positions after  $d$  rounds due to two reasons, (A) either they are truncated during the truncation process or (B) they stay at somewhere else in the tree. We'll show the probability of the former one is bounded by  $\frac{Q}{2}$  (Lemma 5) and the probability of the latter is smaller than  $\frac{KQ}{N}$  (Lemma 6), which is less than  $\frac{Q}{4}$  for  $K = o(N^\beta)$ . Therefore, the total error probability of  $\mathcal{P}_3$   $e_N^{\mathcal{P}_3} \leq Q/4 + Q/2 + Q/4 = Q$ .

To analyze the probability of event A we consider the heap building process as random walks taken by different elements. We define the speed of one element, denoted by  $v$ , as the average levels it moves in each round from its initial position in the tree to its correct position in the heap. Take the maximal element for instance. In two successive rounds of phase 1, the maximal element goes up at least one level if it wins both rounds, at most goes down one level if it wins either of the two rounds and at most goes down two levels if it loses both. Therefore, on average it at least goes up  $q^2 - 2(1-q)q - 2(1-q)^2 = q^2 + 2q - 2$  levels every two rounds. Thus  $v \geq (q^2 + 2q - 2)/2$  for the maximal element.

The following lemma states that with high probability the random walks of the  $K$  largest elements are independent before the end of the truncation process.

*Lemma 4:* Before the end of the truncation process, the probability that any two of the  $K$  largest elements meet is negligible.

*Proof:* Denote by  $L$  the set of leaves in the tree after  $\frac{\log N}{3}$  rounds. During the first  $\frac{\log N}{3}$  rounds, at most  $\frac{\log N}{3}$  levels are truncated (at most one level is truncated in each round), which leads to  $|L| \geq \frac{1}{2}N^{\frac{2}{3}}$ . At the beginning of the protocol, more than one largest elements are located at the same subtree rooted at  $\forall l \in L$  with probability at most

$$\begin{aligned} p &< 1 - \frac{\frac{1}{2}N^{\frac{2}{3}}(\frac{1}{2}N^{\frac{2}{3}} - 1) \dots (\frac{1}{2}N^{\frac{2}{3}} - K + 1)}{(\frac{1}{2}N^{\frac{2}{3}})^K} \\ &< 1 - (1 - \frac{K-1}{\frac{1}{2}N^{\frac{2}{3}}})^K \\ &= o(1). \end{aligned}$$

<sup>10</sup>Every element should be at an appropriate position in the tree to make a heap. This position is called a correct position of that element in the heap.

The last equality is due to the fact that  $K = o(N^\beta)$  ( $0 < \beta < 2/3$ ). ■

Since with high probability the random walks of the  $K$  largest elements are independent during the truncation process, their speed  $v$  is the same. In the following proofs, we need  $v > 1/3$ , which can be satisfied by making  $q$  large enough since  $v$  is a function of  $q$ .

*Lemma 5:* The probability that at least one of the  $K$  largest elements is truncated from the tree is less than  $\frac{Q}{2}$ .

*Proof:* The proof bears the same spirit as Lemma 4.4 in [16]. Consider one such element  $M$ . During  $r = jt$  ( $j = 1, 2, \dots, \lfloor \frac{\log N}{3t} \rfloor$ ) rounds,  $M$  moves up at least  $vr$  levels. According to Hoeffding inequality in Lemma 1, the probability that  $M$  moves up less than  $j$  levels from the bottom of the tree is

$$\begin{aligned} p &= \exp\left(-\frac{r(2(vr - j)/r)^2}{1 - (-2)}\right) \\ &= \exp\left(-j \log\left(\frac{K}{Q} + 2\right) \frac{3}{v} \left[2v\left(1 - \frac{1}{9(\log \frac{K}{Q} + 2)}\right)\right]^2\right) \\ &< \exp\left(-j \log\left(\frac{K}{Q} + 2\right)\right) \quad \text{for } v > \frac{1}{3} \\ &< \frac{Q}{K2^{j+1}}, \end{aligned}$$

which indicates the probability of losing  $M$  due to the truncation is

$$\frac{1}{K} \left( \frac{Q}{4} + \frac{Q}{2^3} + \frac{Q}{2^4} + \dots \right) < \frac{Q}{2K}.$$

Therefore the  $K$  largest elements survive the truncation procedure with probability at least  $1 - \frac{Q}{2}$ . ■

The Lemma below states that any one of the  $K$  largest elements  $M$  arrives at its correct position in the heap after  $d = \frac{108}{v} \log(\frac{N}{Q})$  ( $\forall v > \frac{1}{3}$ ) rounds with probability at least  $1 - \frac{Q}{N}$ .

*Lemma 6:*  $\forall v > \frac{1}{3}$ , after  $d = \frac{108}{v} \log(\frac{N}{Q})$  rounds, any of the  $K$  largest elements  $M$  appears at its correct position with probability at least  $1 - \frac{Q}{N}$ .

*Proof:* The proof follows the lines of Lemma 4.3 in [16]. Denote by  $p(M)$  the correct position of  $M$  in the heap. In  $d$  rounds, the  $M$  moves  $vd$  levels averagely. Thus, for  $v > \frac{1}{3}$ , according to the Hoeffding inequality, the probability that  $M$  moves at least  $\frac{5}{6}vd$  levels is

$$\begin{aligned} p &= 1 - \exp\left(-\frac{d\left(\frac{2(vd - \frac{5}{6}vd)}{d}\right)^2}{1 - (-2)}\right) \\ &= 1 - \exp\left(-4 \log \frac{N}{Q} v\right) \\ &> 1 - \frac{Q}{N}. \end{aligned}$$

Assume some element, say,  $M'$ , stays at  $p(M)$  all the time until it is compared with  $M$  in the triplet including  $p(M)$ . The number of such comparisons is at most  $\frac{d}{2}$ . To keep staying at  $p(M)$ ,  $M'$  must win  $M$  in at least  $\frac{d}{4}$  rounds.

In the worst case,  $M$  needs at most  $\log N$  moves “walking” from its initial position to  $p(M)$ . The additional  $\frac{5}{6}vd - \log N$  levels are used to decrease the counter of  $M'$ . Since  $\frac{5}{6}vd - \log N > \frac{d}{4}$ , the element  $M'$  can not stay at  $p(M)$  after  $d$  rounds in this case (with probability at least  $1 - \frac{Q}{N}$ ). ■

## B. Translate $\mathcal{P}_3$ to an oblivious protocol

The protocol  $\mathcal{P}_3$  is non-oblivious since the set of elements to broadcast in each round depends on previous transmissions. As we mentioned before, non-obvious protocols may cause chaos in protocol execution. The “helper” idea of [8] can help solve this problem if the *number* of elements in each round is predetermined, which is briefly described below. We may describe the protocol  $\mathcal{P}_3$  by a sequence of nodes in the network  $\Omega = \{\omega_j\} = \{S_0, S_1, \dots, S_t\}$ , where  $t$  is the total number of rounds. Since both  $|S_i|$  and  $t$  are fixed, so is  $|\Omega|$ . We arbitrarily assign an element  $i$  in the network to  $\omega_j$ . If it's the turn of  $\omega_j$  to broadcast, the element  $i$  will broadcast the integer held by the element  $\omega_j$ . The question remains how the element  $i$  knows the integer of the element  $\omega_j$ . The answer is letting each element broadcast its integer though  $O(n)$  broadcasts<sup>11</sup> before the protocol starts so that each element contains noisy copies of integers owned by all the other elements.

## V. COMPUTING THE IDENTITY FUNCTION

Identity function is fundamental for computing, the complexity of which can serve as a baseline of comparison for other functions. In this section, we derive a protocol for the  $n$ -bit identity function, which is an extension of the work in [3, 4].

### A. Protocol $\mathcal{P}_4$

The nodes are partitioned into groups of size  $t = \log N$ . For a group  $g$ , assuming that its members are ordered from 1 to  $t$ , the protocol is executed in two phases:

- (1) Each node in the group  $g$  encodes its integer into a codeword with size  $O(\max(n, \log t))$  and broadcasts it.
- (2) After decoding the codewords from all other nodes in its group, each node concatenates all the results (including its own integer) into a word and encodes it into a codeword with size  $O(nt)$ , which is equally partitioned into  $t$  blocks. Then each node  $i \in g$  takes turn to broadcast the  $i$ th block of its codeword. The head node  $H_0$  collects all the blocks from this group and decodes it.

### B. Analysis

*Theorem 7:* The complexity of  $\mathcal{P}_4$   $C_N^{\mathcal{P}_4} = O(N \max(n, \log \log N))$ .

*Proof:* It's easy to check the communication complexity for each group is  $O(\max(n, \log t)t) + O(nt) = O(\max(n, \log t)t)$ . Therefore the total complexity is  $O(\max(n, \log t)t)N/t = O(\max(n, \log t)N) = O(\max(n, \log \log N)N)$ . ■

*Theorem 8:* The error probability of  $\mathcal{P}_4$   $e_N^{\mathcal{P}_4} = 2/(N \log N)$ .

Denote by  $U$  the set of nodes in a group which don't decode all the codewords in its group correctly, for which we have the following result.

*Lemma 7:*  $Pr[|U| \geq \epsilon t] \leq 4^{-t}$ .<sup>12</sup>

*Proof:* According to Lemma 2, assume that each  $n$ -bit integer is encoded by a codeword  $v \in \Gamma_n$  with length  $C_1 \max(n, \log t)$ . The decoding error of any  $n$ -bit integer at any node is bounded above by  $\exp(-2(\frac{1}{2}\gamma - \epsilon)^2 C_1 \max(n, \log t))$ . Summing over the  $t$  integers,

$$\begin{aligned} Pr[i \in U] &\leq t \exp(-2(\frac{1}{2}\gamma - \epsilon)^2 C_1 \max(n, \log t)) \\ &\leq t \exp(-2(\frac{1}{2}\gamma - \epsilon)^2 C_1 \log t). \end{aligned}$$

<sup>11</sup>The additional  $O(nN)$  broadcasts are needed for the oblivious protocol.

<sup>12</sup>Recall that  $\epsilon$  is the upper bound of the noise level, which can be assumed arbitrarily small.

Therefore

$$\Pr[|U| \geq \epsilon t] \leq w^{\epsilon t} 2^t < 4^{-t} \quad \text{for } w < \left(\frac{1}{8}\right)^{1/\epsilon},$$

where  $w = t \exp(-2(\frac{1}{2}\gamma - \epsilon)^2 C_1 \log t)$ . The last inequality can be achieved with sufficiently large  $C_1$ . ■

In phase 2, we encode the  $nt$ -bit word by a codeword  $v_2 \in \Gamma'_n$  with length  $C_2 nt$ . The decoding error at the head node is upper bounded by  $\Pr(d(v'_2, v_2) \geq \frac{1}{2}\gamma C_2 nt) = \Pr(d(v'_2, v_2) \geq 3\epsilon C_2 nt)$  for  $\gamma = 6\epsilon$ . The decoding error occurs as the blocks of bits from some node  $i \in U$  may be erroneous or bits may be corrupted by noise during transmission. Therefore, we have

$$\begin{aligned} & \Pr(d(v'_2, v_2) \geq 3\epsilon C_2 nt) \\ & \leq \Pr[C_2 n|U| \geq C_2 n\epsilon t] + \Pr[N_e \geq 2n C_2 \epsilon t] \\ & \leq 4^{-t} + \exp(-2n C_2 \epsilon^2 t) \\ & = 2 \cdot 4^{-t} \quad \text{for } C_2 \geq 1/\epsilon^2 \\ & = 2/N^2 \end{aligned}$$

where  $N_e$  is the number of bits corrupted by noise. By the union bound, the total error probability is  $2/(N \log N)$ , which completes the proof of Theorem 8.

*Remark 3:* The protocol  $\mathcal{P}_3$  admits the complexity  $O(nN \log(\frac{K}{Q}))$ , while the complexity of the protocol  $\mathcal{P}_4$  is  $O(\max(n, \log \log N)N)$ . When  $K = o(\log N)$  and  $n = o(\log \log N)$ , finding out the  $K$  largest integers directly in  $\mathcal{P}_3$  is more efficient than evaluating the identity function in  $\mathcal{P}_4$ .

## VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have studied the communication complexity of protocols to find out the  $K$  largest values ( $n$ -bit) in a noisy broadcast network of size  $N$  with error tolerance  $Q$ . We have considered various settings for  $K$ ,  $n$ ,  $N$ , and derived some efficient distributed protocols for them. Some interesting future directions are listed below:

- 1) Lower bound on complexity: A general non-trivial lower bound for the complexity of finding the  $K$ -largest integers is lacking. The same is true for the  $n$ -bit identity function. The lower bounding techniques in [4] depends much on the binary assumption (each node owns a binary bit), and is not readily extendable to the  $n$ -bit case.
- 2) General network setting: Some relevant work on the random planar network can be found in [17, 18]. Many problems remain open.

## REFERENCES

- [1] A. Giridhar and P. R. Kumar, "Computing and communication functions over sensor networks," *IEEE J. Selected Areas in Commun.*, vol. 23, no. 4, pp.755-764, April 2005.
- [2] A. El Gamal, Open problem presented in the 1984 Workshop on Specific Problems in Communication and Computation, sponsored by Bell Communication Research, 1984.
- [3] R. Gallager, "Finding Parity in a Simple Broadcast Network," *IEEE Trans. Info. Theory*, vol. 34, no. 2, pp. 176-80, Mar. 1988.
- [4] N. Goyal, G. Kindler and M. Saks, "Lower Bounds for the Noisy Broadcast Problem," *IEEE Symposium on Foundations of Computer Science*, 2005.
- [5] A. Yao, "On the complexity of communication under noise," invited talk in the 5th ISTCS Conference, 1997.
- [6] E. Kushilevitz and Y. Mansour. "Computation in Noisy Radio Networks," *ACM-SIAM Symp. Discrete Algorithms*, pp. 236-243, 1998.
- [7] U. Feige and J. Kilian, "Finding OR in a noisy broadcast network," *Information Processing Letters*, vol. 73, no. 1-2, pp. 69-75, 2000.
- [8] I. Newman, "Computing in fault tolerance broadcast networks," *IEEE Annual Conference on Computational Complexity (CCC)*, pp. 113-122, 2004.
- [9] L. J. Schulman, "Coding for interactive communication," *IEEE Trans. Info. Theory*, vol. 42, no. 6, Nov. 1996, pp. 1745-56.
- [10] S. Rajagopalan and L. Schulman, "A coding theorem for distributed computation," *Proc. 26th Annual ACM Symp. Theory of Comp.*, pp. 790-99, 1994.
- [11] L. Tassioulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Control*, vol. 37, pp.1936-1949, Dec. 1992.

- [12] A. Dua and N. Bambos, "Downlink wireless packet scheduling with deadlines," *IEEE Trans. Mobile Computing*, vol. 6, pp.1410–1425, Dec. 2007.
- [13] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Stat. Assoc.*, vol. 58, pp. 13-30, 1963.
- [14] J. H. van Lint, *Introduction to coding theory*, 3rd Edition, Springer-Verlog, 1999.
- [15] D. E. Knuth, *Sorting and Searching, The Art of Computer Programming*, vol. 3, Addison-Wesley, Reading, MA, 1973.
- [16] U. Feige, P. Raghavan, D. Peleg and E. Upfal, "Computing with Noisy information," *SIAM J. Comput.* vol. 23, no. 5, pp. 1001-1018, 1994.
- [17] Lei Ying, R. Srikant, and Geir E. Dullerud, "Distributed Symmetric Function Computation in Noisy Wireless Sensor Networks," *IEEE transaction on infor. Theory*, vol. 53, no. 12, Dec. 2007
- [18] Y. Kanoria and D. Manjunath, "On Distributed Computation in Noisy Random Planar Networks," IEEE International Symposium on Information Theory, ISIT 2007.