## Short Communication

# Selection of the first $k$ largest processes in hypercubes

Jang-Ping SHEU
*Department of Electrical Engineering, National Central University, Chungli 32054, Taiwan, R.O.C.*

Chun-Lien WU
*Institute of Information Engineering, Tatung Institute of Technology, Taipei, Taiwan, R.O.C.*

Gen-Huey CHEN
*Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.*

Received October 1988
Revised January 1989

**Abstract.** In this paper, algorithms are proposed for finding the first $k$ largest numbered processes in hypercubes with and without faulty nodes.

**Keywords.** Election problem, $k$-selection problem, hypercubes.

## 1. Introduction

The election problem [2,4,5] in a network of processes is to select a leader process from this network. Each process has a unique positive integer number and the process with the largest number ought to be the leader. Rotem et al. [6] proposed algorithms to select the first $k$ largest numbered processes in a ring. Chandran and Rosenfeld [1] presented a parallel algorithm to find the highest $k$ elements out of $n$ elements in hypercubes. However, all of the highest $k$ elements do not know their ranks.

In this paper, we propose algorithms to determine the first $k$ largest numbered processes in hypercubes with or without faulty nodes. Our algorithms are based on the following assumptions:

(1) moving a fixed-sized data packet from one node to any one of its $n$ neighbors takes a unit communication time;

(2) the data communication is bidirectional;

(3) each node has memory of size $O(k)$.

0167-8191/89/$3.50 © 1989, Elsevier Science Publishers B.V. (North-Holland)

## 2. The $k$-selection algorithm without a faulty node

In this section, we present an algorithm for the $k$-selection problem in a hypercube without a faulty node. Each process running at a node is uniquely identified by an integer number. In order to find the first $k$ largest numbered processes we apply the alternate direction exchange method [7] with a slight modification. The alternate direction exchange method is that every node exchanges all its accumulated $2^{i-1}$ data packets with its opposite node in the $i$th direction, where these directions are in turn from the 1st bit to the $n$th bit. The objective of the $k$-selection problem is to select the first $k$ largest numbered processes. So, at each data exchange step only the first $k$ largest numbered processes so far would be the candidates available in the following steps. In other words, when the number of process numbers accumulated is greater than $k$, each node only holds the first $k$ largest ones and discards the others. In order to select the first $k$ largest numbered processes easily, at each step we must maintain the accumulated process numbers in a list and keep these numbers in the lists of every node in descending order all the time. The $k$-selection algorithm is as follows:

```
Algorithm FKSWNF: First k-Selection With NonFaulty Node.
    Initial: Each node is assigned a unique process number and this number is stored in an
    ordered data list.
    For $i = 1$ to $n$ do the following steps:
        Step 1:/* Data exchange (communication) phase */
            Each node exchange all its accumulated $\min\{k, 2^{i-1}\}$ process numbers with its
            opposite node in the $i$th direction.
        Step 2:/* Data merging (computation) phase */
            After data exchanging, each node orderly merges the $k$ process numbers newly
            received and those remaining in the previous step to obtain the new first $k$ largest
            process numbers.
```

This algorithm consists of $n$ steps of data communication and data computation phases. In the data computation phase, each node needs to merge two $k$-ordered data lists into one $k$-ordered data list. This operation can be performed with a two-way merge algorithm [3] but only needs $k$ data comparisons to obtain the new ordered data list with size $k$.

In the following, we shall analyze the time complexity of the proposed algorithm. The communication time is the time needed to send messages and each node sends process numbers simultaneously. Thus, the communication time $C(n)$ for an $n$-cube without a faulty node is derived as follows.

$$C(n) = (1 + 2 + 2^2 + \cdots + 2^i) + (n - i - 1)k \quad \text{where } i = \lfloor \log k \rfloor.$$

The computation time $E(n)$ for an $n$-cube is used for merging two ordered data lists to get the first $k$ largest numbers. By the two-way merge algorithm, it needs $O(k)$ time. Therefore, the computation time $E(n)$ is derived as follows:

$$E(n) = (2^1 + 2^2 + \cdots + 2^i) + (n - i)k \quad \text{where } i = \lfloor \log k \rfloor.$$

Without loss of generality, let $k = 2^i$, $C(n)$ is $k(\log N - \log k) + (k - 1)$ and $E(n)$ is $k(\log N - \log k) + 2(k - 1)$. When $k = 1$ and $N$, $C(n)$ is $\log N$ and $N - 1$, respectively. Likewise, $E(n)$ is $\log N$ and $2(N - 1)$. Obviously, our algorithm is optimal when $k$ is equal to 1 or $N$.

## 3. The $k$-selection algorithms with faulty nodes

In this section, we propose algorithms for the $k$-selection problem in hypercubes with the presence of one faulty node and two faulty nodes. We assume that the faulty nodes can be detected by the system and then their addresses would be known by all other nonfaulty nodes. Based on the recursive construction feature of the $n$-cube, we can divide an $n$-cube with one faulty node into two $(n-1)$-subcubes: one is nonfaulty and the other is faulty. In the same way, we can recursively subdivide the faulty $(n-1)$-subcube into one nonfaulty $(n-2)$-subcube and one faulty $(n-2)$-subcube, and so on. Finally, we can locate the faulty node in a faulty 1-subcube. The $k$-selection algorithm for an $n$-cube with one faulty node is based on the above recursive partitioning property of the $n$-cube.

When one faulty subcube and one nonfaulty subcube are exchange data and combined to get one larger dimensional faulty subcube, the opposite node of the faulty node, called delayed node, cannot receive any data directly from the faulty node. Therefore, one of the neighbors of the delayed node, called relay node, must send data received from the other nodes of the faulty subcube to the delayed node. We call this operation at each combination step an indirect transfer phase. At the indirect transfer phase of the $i$th direction, the address of the relay node is obtained by complementing that of the faulty node in the $i$th and $(i-1)$th bits and the address of the delayed node is obtained by complementing that of the relay node in the $(i-1)$th bit, where $2 \leqslant i \leqslant n$.

Let $C'(n)$ and $E'(n)$ denote the complexities of the communication time and computation time of the $k$-selection algorithm in an $n$-cube with one faulty node, respectively.

*Case* 1: $k \geqslant 2^{n-1}$.

$$C'(n) = 2N - \log N - 2, \qquad E'(n) = 2N - \log N - 4 + 2k.$$

*Case* 2: $k < 2^{n-1}$.

$$C'(n) = 2^{i+2} - i - 3 + 2(\log N - i - 1)k \quad \text{where } i = \lfloor \log k \rfloor,$$

$$E'(n) = 2^{i+2} - i - 5 + 2(\log N - i)k \quad \text{where } i = \lfloor \log k \rfloor.$$

Now, we propose an algorithm for the $k$-selection problem in an $n$-cube with two faulty nodes. The relative locations of two faulty nodes distributed in an $n$-cube has $n$ different cases depending on the Hamming distance of these two faulty nodes. Two faulty nodes in the $n$-cube may be or may not the neighboring. If two faulty nodes are neighboring, then they have one different bit; otherwise they have more different bits than one. However, we can always use the highest different bit between two faulty nodes, say the $x$th bit, to partition the $n$-cube with two faulty nodes into two $(n-1)$-subcubes such that each $(n-1)$-subcube has only one faulty node. After this partition, the first $k$ largest numbered processes for the two $(n-1)$-subcubes can be selected concurrently. Next, we can merge the first $k$ largest numbered processes in each of the two $(n-1)$-subcubes to obtain the first $k$ largest numbered processes in the $n$-cube with two faulty nodes.

Let $C''(n)$ and $E''(n)$ denote the complexities of the communication time and computation time of the $k$-selection algorithm in an $n$-cube with two faulty nodes, respectively.

*Case* 1: $k \geqslant 2^{n-1}$.

$$C''(n) = C'(n-1) + 2(2^{n-1} - 1), \qquad E''(n) = E'(n-1) + 2k.$$

*Case* 2: $k < 2^{n-1}$

$$C''(n) = C'(n-1) + 2k, \qquad E''(n) = E'(n-1) + 2k.$$

In conclusion, we proposed a $k$-selection algorithm which is optimal when $k$ is equal to 1 or $N$. Meanwhile, we also proposed algorithms for hypercubes with one faulty node and two faulty nodes. The function of our algorithms is that every node knows where the first $k$ largest numbered processes are and each of the first $k$ largest numbered processes can also know its rank.

## References

[1] S. Chandran and A. Rosenfeld, Order statistics on a hypercube, *Inform. Process. Lett.* **27** (1988) 129–132.

[2] D. Dolev, M. Klawe and M. Rodeh, An $O(n \log n)$ unidirectional distributed algorithm for extrema finding in a circle, *J. Algorithms* **3** (1982) 245–260.

[3] D.E. Knuth, *The Art of Computer Programming Vol. 3: Sorting and Searching* (Addison-Wesley, Reading, MA, 1974).

[4] E. Korach, S. Moran and S. Zaks, Tight lower and upper bounds for some distributed algorithms for a complete network of processors, in: *Proc. 3rd Annual ACM Symposium on Principles of Distributed Computing* (1984) 199–207.

[5] M.C. Loui, T.A. Matsushita and D.B. West, Election in a complete network with a sense of direction, *Inform. Process. Lett.* **22** (1986) 185–187.

[6] D. Rotem, E. Korach and N. Santoro, Analysis of a distributed algorithm for extrema finding in a ring, *J. Parallel Distrib. Comput.* **4** (1987) 575–591.

[7] Y. Saad and M.H. Schultz, Data communication in parallel architectures, *Parallel Comput.* **11** (1989) 131–150.