DISTRIBUTED COMPUTING ON ANONYMOUS HYPERCUBES WITH FAULTY COMPONENTS*
(Extended Abstract)

Evangelos Kranakis
(kranakis@scs.carleton.ca)
Nicola Santoro
(santoro@scs.carleton.ca)
School of Computer Science
Carleton University
K1S 5B6 Ottawa, Ontario, Canada

Abstract. We give efficient algorithms for distributed computation on anonymous, labeled, asynchronous hypercubes with possible faulty components (i.e. processors and links). The processors are deterministic and execute identical protocols given identical data. Initially, they know only the size of the network (in this instance, a power of 2) and that they are inter-connected in a hypercube network. Faults may occur only before the start of the computation (and that despite this the hypercube remains a connected network). However the processors do not know where these faults are located. As a measure of complexity we use the total number of bits transmitted during the execution of the algorithm and we concentrate on giving algorithms that will minimize this number of bits. The main result of this paper is an algorithm for computing boolean functions on anonymous hypercubes with at most \( \gamma \) faulty components, \( \gamma \geq 1 \), with bit complexity \( O(N\delta_n(\gamma)^2\lambda^2 \log \log N) \), where \( \gamma \) is the number of faulty components, of which \( \lambda \) is the number of faulty links, and \( \delta_n(\gamma) \) is the diameter of the hypercube.

1980 Mathematics Subject Classification: 68Q99
CR Categories: C.2.1
Key Words and Phrases: Anonymous labeled hypercube, boolean function, diameter, faulty component, group of automorphisms.

* Research supported in part by National Science and Engineering Research Council grants.
1 Introduction

In this paper we consider algorithms which are appropriate for distributed computation on anonymous, labeled, asynchronous, n-dimensional hypercubes $Q_n$ with faulty components (i.e. processors and links). The processors occupy the nodes of a hypercube and want to compute a given boolean function $f$ on $\leq N = 2^n$ variables. Initially each non-faulty processor $p$ has an input bit $b_p$. When the computation terminates all processors must output the same value $f(< b_p : p \text{ non-faulty }>).^2$

The problem arising is to determine the bit complexity (i.e. total number of bits transmitted) of computing boolean functions on faulty hypercubes. In the present paper we give efficient algorithms for computing boolean functions on such networks.

1.1 Assumptions and Related Literature

The network we consider is the anonymous, asynchronous hypercube with possible faulty components. The number of faulty components may be arbitrary as long as the hypercube remains connected. If a processor is faulty then all the links adjacent to it are also interpreted as faulty. Faults may occur only before the start of the computation. We assume that the network links are FIFO, and that the processors have a sense of direction. By this we mean that the hypercube is canonically labeled (the label of link $xy$ is $i$ if and only if $x, y$ differ at exactly the $i$th bit) and that these labels are known to the processors concerned. In addition we assume that the following assumptions hold:

- the processors know the network topology (in this instance hypercube), and the size of the network, but they do not necessarily know where the faulty links may be,
- the processors are anonymous (i.e., they do not know either the identities of themselves or of the other processors), they are deterministic (i.e. they all run deterministic algorithms), and they all run the same algorithm given the same data.

The assumptions listed above are meant to take "maximum" advantage of network distributivity. For a discussion regarding the necessity of some of the above assumptions see [2]. Routing algorithms on hypercubes have been studied in [6]. Faulty hypercube networks have been examined in several papers under the much stronger assumption of synchronous and/or non-identical processors. In such networks it is possible to apply reconfiguring techniques [7] (nodes of an $n-1$-dimensional hypercube are mapped into non-faulty nodes of an $n$-dimensional hypercube with $O(1)$ dilation) or even non-faulty subcube techniques [5] (for a

---

2 Our notation $b_p$ for the bit associated with processor $p$ does not mean that we assign names to processors. In addition the input $< b_p : p \in \text{ non-faulty } >$ represents the assignment of bits to all the non-faulty processors of the network, and it will be computed by all the processors via an "input collection" algorithm.