Space Efficient Processor Identity Protocols

Shang-Hua Teng

March 1989
CMU-CS-89-123

Abstract

In this paper, a space efficient probabilistic protocol is presented for the *Processor Identity Problem*–an essential problem in distributed computation and asynchronous parallel computation. Our protocol uses only $O(n \log^2 n)$ bits. The new protocol improves the previous known protocol, due to Lipton and Park, which uses $O(n^2)$ bits. Our protocol is very simple, *fully distributed* and *symmetric*. This provides a very practical and important primitive for distributed systems and asynchronous parallel systems.

This work was supported in part by National Science Foundation Grant CCR-87-13489.

The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the US Government.

1 Introduction

The space complexity of a fundamental problem in distributed and asynchronously parallel computation, the *Processor Identity Problem*, is studied. The Processor Identity Problem assigns unique identifiers to processors in an asynchronous distributed system. More specifically, a solution to the processor identity problem is a protocol run by n asynchronous processors which communicate via a shared memory to produce a unique assignment of processors to elements of the set $\{1, 2, \cdots, n\}$. This problem was first introduced in [6] where Lipton and Park gave a $O(n^2)$ bits probabilistic protocol to the problem.

Theorem 1.1 (Lipton and Park) *The Processor Identity Problem can be solved in Ln^2 bits with probability at least $1 - c^L$ for some constant $c > 1$.*

It seems, from the construction in [6], that the $\Omega(n^2)$ bits are required for the Processor Identity Problem; and it is not clear how to reduce the failure probability to $\frac{1}{n^c}$, for any constant $c > 0$, using no more than $O(n^2)$ bits.

In this paper, a space efficient probabilistic protocol is presented for the *Processor Identity Problem*. Our protocol uses only $O(n \log^2 n)$ bits. The new protocol improves the previous known protocol by simultaneously reducing the number of bits required and the failure probability.

Theorem 1.2 (Main Result)

1. *The Processor Identity Problem can be solved in $O(n \log^2 n)$ bits with probability at least $1 - \frac{1}{n^c}$, for any constant $c >$.*

2. *If there are n^2 bits in the shared memory, then there exists a Processor Identity Protocol with failure probability bounded by $\frac{1}{2^{\sqrt{n}}}$.*

Like the protocol in [6], our protocol is very simple, *fully distributed* and *symmetric* [5,6]. Our solution also makes no assumption about the initial contents of the elements in the shared memory.

As observed by Lipton and Park [6], a great number of multiprocessor coordination problems, distributed computing problems, and asynchronous parallel computation problems, such as Choice Coordination problem [7], Mutual Exclusion Problem [2,3,4], Drinking Philosopher Problem [1], many Consensus Problems, assume that processors initially have unique identifiers. Hence, our new protocol can be used as the first step to solve these problems efficiently. The solution to the processor identity problem provides a very important primitive for distributed computation and asynchronous parallel computation [6].

Therefore, our solution provides a very practical and important primitive for distributed systems and asynchronously parallel systems.

2 Definitions

The computation model used in the Processor Identity Problem is called *Asynchronous CRCW PRAM* which is an asynchronous distributed systems of n processors that communicate among each other via a common shared memory. Each processor has its local memory and can perform asynchronous read and write operations to the elements in the shared memory and some basic logic and arithmetic operations. Concurrent reads to an element in the shared memory is allowed and if more than one processor tries to write to a single element in the shared memory, it is assumed that an arbitrary processor succeeds. *The only way processors can communicate is through the common memory.* Each processor has a random number generator.

There is neither a central clock nor a central arbiter in the system. There is no assumption about the speed of processors in the system except that each operation performed by a processor takes finite amount of time.

Definition 2.1 (Processor Identity Problem) *The Processor Identity Problem is to design a protocol which is run on each processor to produce a one-to-one assignment of the n processors to elements of the set $\{1, 2, \cdots, n\}$.*

The following is a theorem proved by Lipton and Park which says that there is no deterministic protocol for the Processor Identity Problem.

Theorem 2.1 (Lipton and Park) *Assume that all processors start in identical states, for any fixed time t, no protocol exists which always solves the Processor Identity Problem within time t.*

3 A Simple and Space Efficient Protocol

In this paper, the *Processor Identity Protocol* has three parameters (K, L, M^2). Such a protocol is denoted by $\mathcal{PIP}(K, L, M^2)$, which uses KLM^2 bits. In each $\mathcal{PIP}(K, L, M^2)$ protocol, the KLM^2 bits in the common shared memory are partitioned into K $L \times M^2$ bit-arrays, $\mathcal{B}_1, \ldots, \mathcal{B}_K$ (see Figure 1).

To simplify the specification, the following set of notations is used.

- $\mathcal{B}_i[j, *]$: the j^{th} row of $\mathcal{B}_i$;
- $|V|$: the number of 1's in a 0–1 vector V, called the *size* of V;
- $|row(\mathcal{B})|$: the maximum size over rows of a 0–1 array $\mathcal{B}$, called the *row-size* of $\mathcal{B}$;
- $I(\mathcal{B})$: the index of the row with maximum size. If there is a tie, $I(\mathcal{B})$ denotes any one of them.
- $|\mathcal{B}|$: the number of 1's in a 0–1 array $\mathcal{B}$, called the *size* of $\mathcal{B}$;

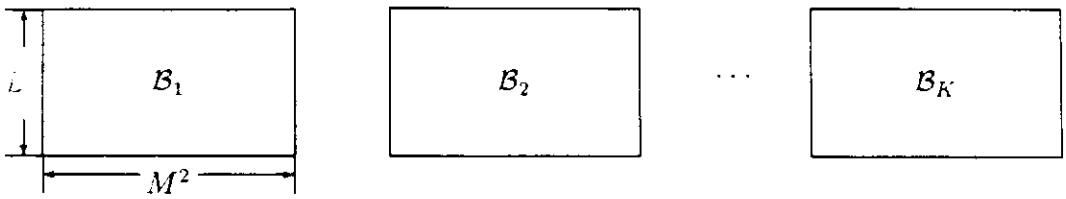

Figure 1: The Partition of the Memory

3.1 The Protocol

Our Processor Identity Protocol is *completely distributed* and *symmetric* [5,6]. Even though the protocol is probabilistic, it is *safe* in the sense that

1. if one processor terminates, then all processor terminates;
2. if the protocol terminates, than it generates a valid identity for each processor.

Moreover, our protocol terminates with very high probability. The following probabilistic algorithm is run on each processor;

Identity Protocol:

1. **select** b from $\{1, 2, \cdots, K\}$, randomly;;
2. **select** $i_1, \ldots, i_L$ from $\{1, 2, \cdots, M^2\}$, randomly;;
3. **initialize** all array elements in each array to 0;;
4. **repeat**
   (a) for $k = 1, 2, \cdots, L$
      i. $\mathcal{B}_b[k, i_k] = 1$;;
      ii. for $l = 1, 2, \cdots, K$
         - if $\sum_{l=1}^{K} |row(\mathcal{B}_l)| = n$, **exit** with identifier $(b, I(\mathcal{B}_b), i_{I(\mathcal{B}_b)})$;;

The basic idea of the protocol is that at the first step, each processor randomly selects the first $logK$ bits of its name. It will be shown that with very high probability, processors

are approximately evenly partitioned into K groups. Hence, the problem size is reduced from n to $\frac{n}{K}$. Then the method of Lipton and Park can be used on the smaller-sized problem. However, the protocol of Lipton and Park can not be used directly to the smaller-sized problem, because it is not known *a priori* the number of processors in each group. Hence, each processor can not determine whether the subprotocol on its group is successful. A critical observation to circumvent this problem is that each processor can check whether all subprotocols succeed by checking whether $\sum_{l=1}^{K} |row(\mathcal{B}_l)| = n$.

3.2 The Correctness

We have to prove that our protocol is safe. In other words, we have to show that

Lemma 3.1 (Correctness)

1. *if one processor terminates, then all processor terminates;*
2. *if the protocol terminates, than no two processors exits with the same identifier.*

[**PROOF**]: The Proof is similar to that of Lipton and Park [6]. We first show that our protocol satisfies the condition (2).

Since all array elements of $\mathcal{B}_1, \ldots, \mathcal{B}_K$ are set to 0 at the first step, and each processor can only write one 1 in each row of its corresponding array. Hence, if a processor exits with an identity $(b, I(\mathcal{B}_b), i_{I(\mathcal{B}_b)})$, and n_b processors choose $\mathcal{B}_b$, then the $\mathcal{I}(\mathcal{B}_b)^{th}$ row of $\mathcal{B}_b$ must contain n_b 1's each of which is written by a different processor. So, it is impossible that another processor exits with the same identity $(b, I(\mathcal{B}_b), i_{I(\mathcal{B}_b)})$, because in that case, there is a row in $\mathcal{B}_b$ containing n_b 1's; while $\mathcal{I}(\mathcal{B}_b)$ contains at most $n_b - 1$ 1's. This contradicts the definition of $\mathcal{I}(\mathcal{B}_b)$.

Since each processor can only write one 1 in each row of its corresponding array. Hence, no processor can exit before others finish setting all array elements to 0. Now, suppose one processor exits, then it must be the case that $\sum_{l=1}^{K} |row(\mathcal{B}_l)| = n$. Since, the arrays does not change during the **repeat loop**, hence, each processor will eventually detects $\sum_{l=1}^{K} |row(\mathcal{B}_l)| = n$ and exits. □

As proven in Lemma 3.1, upon termination, each processor obtains an unique identifier of the form (i, j, k), where $1 \leq i \leq K$, $1 \leq j \leq L$, and $1 \leq k \leq M^2$. The following protocol transforms this set of unique identifiers to a one-to-one assignment of the n processors to elements of the set $\{1, 2, \cdots, n\}$.

Let $\mathcal{ID} = \{(i, j, k) | 1 \leq i \leq K, 1 \leq j \leq K, 1 \leq k \leq M^2\}$. For each pair (i_1, j_1, k_1) and (i_2, j_2, k_2) from $\mathcal{ID}$, $(i_1, j_1, k_1) < (i_2, j_2, k_2)$ if (i) $i_1 < i_2$, or (ii) $i_1 = i_2$ and $j_1 < j_2$, or (iii) $i_1 = i_2$, $j_1 = j_2$, and $k_1 < k_2$. Let $(l_1, \ldots, l_n) \in \mathcal{ID}^n$. The *rank* of l_i in $(l_1, \ldots, l_n)$ is the number of elements in $(l_1, \ldots, l_n)$ which are less than or equal to l_i.

It follows from Lemma 3.1 that a successful execution of the Identity Protocol generates n elements $(l_1, \ldots, l_i)$ from $\mathcal{ID}$. The following Ranking Protocol computes the rank of

each elements in $(l_1, \ldots, l_i)$ using KLM^2 bits. Assume another KLM^2 bits in the shared memory are partitioned into K $L \times M^2$ bit-arrays, $\mathcal{C}_1, \ldots, \mathcal{C}_K$. Assume a processor obtained an identifier $(i, j, k) \in \mathcal{ID}$ from the Identifier Protocol.

Ranking Protocol:

1. **initialize** all array elements in each array to 0;;
2. **repeat**
   (a) $\mathcal{C}_i[j, k] = 1$;;
   (b) for $l = 1, 2, \cdots, K$
   - if $\sum_{l=1}^{K} |\mathcal{C}_l| = n$, **exit** with the rank of (i, j, k);;

The correctness of the Ranking Protocol can be proven similarly as that of the Identity Protocol (see Lemma 3.1).

3.3 Failure Analysis

It follows from the protocol that the protocol terminates iff

$$\sum_{l=1}^{K} |row(\mathcal{B}_l)| = n. \tag{1}$$

An execution of the protocol is *feasible* if (1) is satisfied; It is *regular* if there is no $1 \leq b \leq K$ such that no more than M processors choose the same array $\mathcal{B}_b$; It is *b–resolvable* if there are n_b processors choose $\mathcal{B}_b$ and $|row(\mathcal{B}_b)| = n_b$. Clearly, an execution of the protocol is feasible iff for all $1 \leq b \leq K$, it is *b–resolvable.*

Let $Pr(f)$ be the probability that an execution of the protocol is feasible and $Pr(r)$ be the probability that an execution of the protocol is regular.

Lemma 3.2 *With probability at least* $1 - K(\frac{en}{KM})^M$, *an execution of the protocol is regular.*

[**PROOF**]: Let

$$PATTERN = \{(n_1, \ldots, n_K) \mid \sum_{l=1}^{K} n_l = n \ \& \ n_l \geq 0\}$$

$$BAD(M) = \{(n_1, \ldots, n_K) \in PATTERN \mid \exists l, n_l > M\}$$

Clearly,

$$Pr(r) = 1 - \frac{|BAD(M)|}{|PATTERN|}$$

Let $Pr(rb)$ be the probability that there are more than M processors choose $\mathcal{B}_b$, clearly, for all $1 \leq i, j \leq K$,

$$Pr(ri) = Pr(rj) \tag{2}$$

$$\frac{|BAD(M)|}{|PATTERN|} \leq K \cdot Pr(r1) \tag{3}$$

$$Pr(r1) \leq \binom{n}{M} \left(\frac{1}{K}\right)^M \leq \left(\frac{e \cdot n}{K \cdot M}\right)^M \tag{4}$$

Therefore $Pr(r) \geq 1 - K(\frac{en}{KM})^M$. □

Lemma 3.3 *If $n_b \leq M$, then with probability at least $1 - \frac{1}{e^L}$, an execution of the protocol is b–resolvable.*

[**PROOF**]: See Lipton and Park [6]. □

Theorem 3.1 *The Processor Identity Problem can be solved in $O(n \log^2 n)$ bits with probability at least $1 - \frac{1}{n^c}$, for some constant $c > 0$.*

[**PROOF**]: Let $K = n/\log n$, $L = c_1 \log n$, and $M = c_2 \log n$, where $c_1 = \frac{c+1}{\log e}$ and $c_2 = c+1$. It follows from Lemma 3.2 that the an execution of the protocol is regular with probability at least

$$1 - K\left(\frac{en}{KM}\right)^M = 1 - \frac{n}{\log n}\left(\frac{en \log n}{nc_2 \log n}\right)^{c_2 \log n} \geq 1 - \left(\frac{1}{2}\right)^{c_2 \log n - \log n} \geq 1 - \frac{1}{n^c}$$

It follows from Lemma 3.3 that an execution of the protocol is feasible under the condition that it is regular is at least

$$1 - K\frac{1}{e^L} = 1 - \frac{n}{\log n}\frac{1}{e^{c_1 \log n}} \geq 1 - \frac{1}{2^{c_1 \log e \log n - \log n}} \geq 1 - \frac{1}{n^c}$$

Hence, an execution of the protocol terminates with probability at least $1 - O(\frac{1}{n^c})$. Since the protocol is safe (Lemma 3.1), the theorem follows. □

3.4 Failure–Space Trade–off

A processor identity protocol $\mathcal{PIP}(K, L, M^2)$ is an *ϵ–protocol* if an execution of $\mathcal{PIP}(K, L, M^2)$ terminates with probability at least $1 - \epsilon$. A *space–minimal* ϵ–protocol tries to minimize the number of bits required.

Lemma 3.2 and Lemma 3.3 provide a formula of the trade-off between the space requirement and the probability of failure. It follows from Lemma 3.2 and 3.3 that the failure probability of a processor identity protocol $\mathcal{PIP}(K, L, M^2)$ is at most $\max\{K(\frac{en}{KM})^M, K\frac{1}{2^L}\}$.

Hence, in any ϵ–protocol, $\epsilon < 1$,

1. $L \geq \log K - \log \epsilon$; and

2. $K(\frac{en}{KM})^M \leq \epsilon$.

This implies that $KM > en$. Since the number of bits required in $\mathcal{PIP}(K, L, M^2)$ is KM. Hence, in space-minimal protocol, $KM = c_1 n$, with $c > e$ and M is minimized under the condition of (2). It follows that $L = \Theta(\log n - \log \epsilon)$ and $M = \Theta(\log n - \log \epsilon)$. So, use the processor identity protocol given in this paper, the number of bits required in an ϵ-protocol is $\Theta(n(\log n - \log \epsilon)^2)$.

Theorem 3.2 *If there are $n^{1+2\epsilon}$, $\epsilon > 0$, bits in the common memory, then there is a processor identity protocol with failure probability bounded by $\frac{1}{2^{n^\epsilon}}$.*

Corollary 3.1 *If there are n^2 bits in the common memory, then there is a processor identity protocol with failure probability bounded by $\frac{1}{2^{\sqrt{n}}}$.*

Therefore, if n^2 bits, the same number of bits in Lipton and Park's protocol, are available in the common memory, then the failure probability is reduced from $\frac{1}{2^L}$ to $\frac{1}{2^{\sqrt{n}}}$, where L is some constant.

4 Open Question

In this paper, a new $O(n \log^2 n)$ bit processor identity protocol is presented. This improves the previous protocols by simultaneously reducing the number of bits required and the failure probability.

The following question is still open.

- Is there a processor identity protocol which uses $o(n \log^2 n)$ bits with failure probability bounded by $\frac{1}{2}$?

Conjecture 4.1 *Any processor identity protocol with failure probability bounded by $\frac{1}{2}$ requires $\Omega(n \log n)$ bits.*

Acknowledge We would like to thank Alan Frieze, Gary Miller, and Arvin Park for valuable discussion. We also thank Manpreet Khaira for proofreading the paper and helpful comments.

References

[1] K. M. Chandy and J. Misra. The drinking philosophers problems. *ACM Transactions on Programming Languages and Systems.*, 6:632–646, 1984.

[2] E. W. Dijkstra. Solution of a problem in concurrent programming control. *CACM.*, 8:569–578, 1965.

[3] D. Knuth. Additional comments on a problem in concurrent control. *CACM.*, 9:321–322, 1966.

[4] L. Lamport. The mutual exclusion problem: part i and ii. *JACM.*, 33:313–348, 1986.

[5] D. Lehmann and M. O. Rabin. On advantages of free choice: a symmetric and fully distributed solution to the dining philosophers problem. In *Procedings of 8th Annual ACM Symposium on Principle of Programming Languages*, pages 133–138, ACM, January 1981.

[6] Richard J. Lipton and Arvin Park. The processor identity problem. manuscript, 1988.

[7] M. O. Rabin. The choice coordination problem. *Acta Informatica.*, 17:121–134, 1982.