ELSEVIER

Information Processing Letters 60 (1996) 81-89

Information Processing Letters

A fast maximum finding algorithm on broadcast communication [1]

Shyue-Horng Shiau, Chang-Biau Yang *

Department of Applied Mathematics, National Sun Yat-sen University, Kaohsiung, Taiwan 804, ROC

Received 1 April 1995; revised 1 August 1996
Communicated by S.G. Akl

Keywords: Parallel algorithms; Maximum finding; Broadcast communication

1. Introduction

One of the simplest parallel computation models is the *broadcast communication model* [1,2,4-9,11,12]. This model consists of some processors sharing one common channel for communications. Each processor in this model can communicate with others only through this shared channel. Whenever a processor broadcast messages, any other processor can hear the broadcast message via the shared channel. If more than one processor wants to broadcast messages simultaneously, a *broadcast conflict* occurs. When a conflict occurs, a conflict resolution scheme should be invoked to resolve the conflict. This resolution scheme will enable one of the broadcasting processors to broadcast successfully. The Ethernet, one of the famous local area networks, is an implementation of such a model. The study on this model is less than that on other parallel computer structures. However, Ethernet-like networks are very common and popular today. Thus, the distributed or parallel computation on this model is more realistic than that on other parallel computer structures.

A straightforward method for finding maximum under the broadcast communication model is to have all processors broadcast their elements one by one. Then the maximum element can be found after n broadcasts. The time required for this method is $O(n)$. Levitan and Foster [5,6] proposed a maximum finding algorithm, which is nondeterministic and requires $O(\log n)$ successful broadcasts in average. To produce a successful broadcast, we may have to resolve a broadcast conflict. The fastest nondeterministic conflict resolution scheme, which was proposed by Willard [10], requires $O(\log\log n)$ time in average to resolve a conflict. Thus, finding a maximum by Levitan and Foster's algorithm requires $O(\log n \log\log n)$ time in average.

Martel [7] showed this problem can be solved in $O(\log n)$ time in average by his algorithm. His basic idea is to estimate a proper broadcasting probability which is held by each data element. Each element decides individually whether it broadcasts in the current time slot or not. We can expect that one of the processors will succeed when about one data element in average wants to broadcast. Until now, Martel's algorithm is the

* Corresponding author. Email: cbyang@math.nsysu.edu.tw.

[1] This research work was partially supported by the National Science Council of the Republic of China under contract NSC 84-2121-M-110-005 MS.

PII S0020-0190(96)00143-3

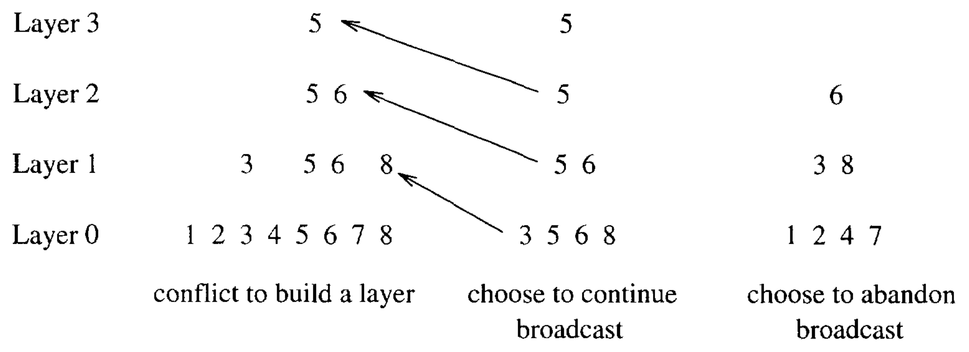

Layer 3
Layer 2
Layer 1
Layer 0
5
5 6
3 5 6 8
1 2 3 4 5 6 7 8
5
5
5 6
3 5 6 8
6
3 8
1 2 4 7
conflict to build a layer
choose to continue broadcast
choose to abandon broadcast

Fig. 1. An example for the layer concept in maximum finding.

fastest one foı solving the maximum finding problem under the broadcast communication model. However, the constants associated with the O(log n) bound proved by Martel [7] are not very tight.

The time required for an algorithm to solve a problem under the broadcast communication model includes three parts: (1) resolution time: spent to resolve conflicts, (2) transmission time: spent to transmit data, (3) computation time: spent to solve the problem. To minimize one of these three parts will improve the time complexity. For the maximum finding problem, it does not seem that transmission time and computation time can be reduced. Therefore, to minimize resolution time is the key point to improve the time complexity of maximum finding.

The layer concept proposed by Yang [11] is another idea which can reduce conflict resolution time. In the layer concept, when there is a broadcast conflict, only those processors who join this conflict have the right to choose to continue to broadcast in the next time slot and bring themselves up to the upper layer. That is, on upper layers, there are only few active processors. Thus, conflict resolution time can be reduced. In this paper, we shall apply the layer concept to solve the maximum finding problem under the broadcast communication model. Our algorithm is very simple. And we shall give a tight bound for the time complexity of our algorithm. The average number of time slots (including conflict slots, empty slots, and slots for successful broadcast) required is x, where $4\ln n - 4 < x < 5\ln n + \frac{5}{2}$. Here, it is assumed that each time slot requires constant time.

2. The algorithm for maximum finding

In our maximum finding algorithm, each processor holds one data element initially. The key point of our algorithm is to use broadcast conflicts to build broadcasting layers and then to distribute the data elements into those layers. For shortening the description in our algorithm, we use the term "data element" to represent "the processor storing the data element" and to represent the data element itself at different times if there is no ambiguity. When a broadcast conflict occurs, a new upper layer is built up. Each data element which joins the conflict flips a coin with equal probabilities. That is, the probability of getting a head is $\frac{1}{2}$. All which get heads continue to broadcast in the next time slot, and bring themselves up to the new upper layer. This new layer becomes the active layer. The others which get tails abandon broadcasting, and still stay on the current layer. At any time, only the processors which are alive and on the active layer may broadcast.

Fig. 1 shows an example for illustrating our algorithm. Initially, all data elements broadcast. Hence, in time slot 1, all of 1, 2, ..., 8 broadcast and a conflict occurs. Then layer 0 is built. Suppose that before time slot 2 coming, only 3, 5, 6, 8 get heads. Then in time slot 2, elements 3, 5, 6, 8 build layer 1 by conflict and bring themselves up to layer 1. At the same time, elements 1, 2, 4, 7 stay in layer 0. Before time slot 3 coming, all alive data elements 3, 5, 6, 8 on the active layer (layer 1) flip coins. This process will continue. Suppose that one day, in layer 3, only 5 get a head and broadcasts successfully. Now 5 is the first winner. Of course, 5 is not the last winner. The elements whose values are less than 5, such as 3 on layer 1 and 1, 2, 4 on layer 0, will drop out (become dead). The active layer is down to layer 2, which contains only one alive data element, that is 6. In fact, 6 is the next winner. The active layer is down to layer 1 which contains only one alive data element, that is 8. 8 is the next winner. And, 7 will drop out (become dead). Thus, 8 is the final winner, which is the maximum.

Our algorithm can represented as a recursive function: *Maxfind*(C), where C is a set of data elements (processors). Initially, each processor holds one alive data element and C is the set of all data elements.

Algorithm Maximum-Finding: *Maxfind*(C)
Step 1: Each processor sets an initial return value $r = -\infty$.
Step 2: Each alive data element in C broadcasts its value. Note that a data element is alive if its value is greater than r.
Step 3: There are three possible cases:
Case 3a: If a broadcast conflict occurs then
Step 3a.1: each alive data element in C flips a coin. All which get heads form C' and bring themselves to the upper layer.
Step 3a.2: $r =$ *Maxfind*(C').
Step 3a.3: go to Step 2.
Case 3b: If one data element successfully broadcast its value, each processor sets r to this value. Each element smaller than or equal to r drops out (becomes dead). Return r.
Case 3c: If no data element broadcasts, return r. (C is empty or all data elements of C are dead.)

Note that a new upper layer is built up when the recursive function *Maxfind* is called and the current layer is revisited after we return from *Maxfind*. In Case 3c, there are two possible situations. One is that no element goes up to the upper layer. The other is that no alive element remains on the current layer after the current layer is revisited because all elements go up to the upper layer, or some elements broadcast successfully on upper layers and they kill all elements on the current layer.

3. Analysis of the algorithm

In this section, we shall prove that the average time complexity of our maximum finding algorithm is $\Theta(\log n)$, where n is the number of input data elements.

Suppose that there are k data elements held by at least k processors in which each processor holds at most one data element. Let T_k denote the average number of time slots, including conflict slots, empty slots and slots for successful broadcasts, required when the algorithm is executed.

When there is zero or one input data element for the algorithm, one empty slot or one slot for successful broadcast is needed. Thus $T_0 = 1$ and $T_1 = 1$.

T_2 can be obtained by the following equation:

$$T_2 = 1 + \tfrac{1}{4}(T_2 + T_0) + \tfrac{1}{4}(T_1 + T_1) + \tfrac{1}{4}(T_1 + T_0) + \tfrac{1}{4}(T_0 + T_2). \tag{1}$$

We shall explain Eq. (1) term by term. The first term, is a conflict in the first time slot. Then, two alive data elements flipping coins can be divided into 4 cases, each having probability $\frac{1}{4}$. Here, it is assumed that the probability of each data element getting a head is $\frac{1}{2}$. The second term, $\frac{1}{4}(T_2 + T_0)$, is caused by that both elements get heads and bring themselves to the upper layer. The number of time slots required for the upper layer can be represented as recursive T_2. The empty current layer needs T_0 time slots. The third term, $\frac{1}{4}(T_1 + T_1)$, means that the smaller element gets a head and broadcasts successfully on the upper layer. This successful broadcast requires T_1 time slots. The other subterm, T_1, implies that the larger element stays in the current layer and is still alive. The fourth term, $\frac{1}{4}(T_1 + T_0)$, means that the larger element gets a head and broadcasts successfully. The subterm, T_0, implies that the smaller element stays in the current layer and becomes dead. Therefore, the current layer is empty. The final term, $\frac{1}{4}(T_0 + T_2)$, means that no element gets a head and brings itself to the upper layer. Thus, there are still two elements remaining in the current layer.

Substituting $T_0 = 1$ and $T_1 = 1$ into Eq. (1) and solving it, we get $T_2 = 5$.

When $k = 3$, there are eight cases as shown in Fig. 2. Therefore, extending T_2 to T_3, we can obtain T_3 as follows:

$$\begin{aligned} T_3 = 1 &+ \tfrac{1}{8}(T_3 + T_0) \\ &+ \tfrac{1}{8}(T_2 + T_1) + \tfrac{1}{8}(T_2 + T_0) + \tfrac{1}{8}(T_2 + T_0) \\ &+ \tfrac{1}{8}(T_1 + T_2) + \tfrac{1}{8}(T_1 + T_1) + \tfrac{1}{8}(T_1 + T_0) \\ &+ \tfrac{1}{8}(T_0 + T_3). \end{aligned} \tag{2}$$

The terms in Eq. (2) are one to one correspondence to the cases in Fig. 2. For example, the fourth term, $\frac{1}{8}(T_2 + T_0)$, means that the second smallest element stays in the current layer (not broadcasts). The fifth term, $\frac{1}{8}(T_2 + T_0)$, means that the smallest element does not get a head (not broadcasts).

Substituting $T_0 = 1$, $T_1 = 1$ and $T_2 = 5$ into Eq. (2) and solving it, we have $T_3 = \frac{19}{3}$.

The following lemma generalizes the above result.

Lemma 1.

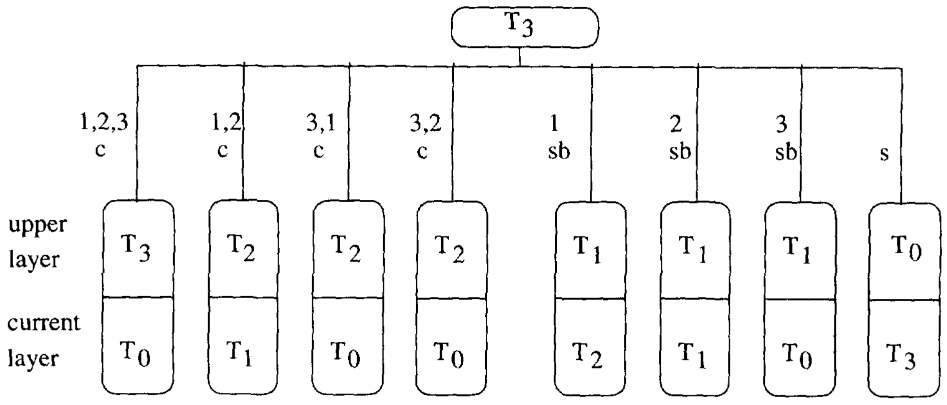
$$\begin{aligned} T_k = 1 + \frac{1}{2^k}\Big[& \qquad T_k && + T_0 \\ & + C^k_{k-1}T_{k-1} && + C^{k-2}_{k-2}T_1 + C^{k-1}_{k-2}T_0 \\ & + C^k_{k-2}T_{k-2} && + C^{k-3}_{k-3}T_2 + C^{k-2}_{k-3}T_1 + C^{k-1}_{k-3}T_0 \\ & + \quad \cdots && \\ & + \quad C^k_i T_i && + C^{i-1}_{i-1}T_{k-i} + \cdots + C^{k-3}_{i-1}T_2 + C^{k-2}_{i-1}T_1 + C^{k-1}_{i-1}T_0 \\ & + \quad \cdots && \\ & + \quad C^k_2 T_2 && + C^1_1 T_{k-2} + \cdots + C^{i-1}_1 T_{k-i} + \cdots + C^{k-3}_1 T_2 + C^{k-2}_1 T_1 + C^{k-1}_1 T_0 \\ & + \quad C^k_1 T_1 && + C^0_0 T_{k-1} + C^1_0 T_{k-2} + \cdots + C^{i-1}_0 T_{k-i} + \cdots + C^{k-3}_0 T_2 + C^{k-2}_0 T_1 + C^{k-1}_0 T_0 \\ & + \quad T_0 + T_k && \Big], \end{aligned}$$

for $k \geqslant 2$.

Proof. The terms in the lemma can be expressed as follows:

$$C^k_i T_i + \sum_{j=i-1}^{k-1} C^j_{i-1} T_{k-j-1}.$$

$C^k_i T_i$ means that i elements get heads and bring themselves into the upper layer. T_i time slots are required for finding the maximum of the elements on the upper layer. $C^j_{i-1} T_{k-j-1}$ means that the $(j+1)$th smallest element and other $(i-1)$ smaller elements get heads and bring themselves to the upper layer. Note that the kth smallest element is actually the largest one. After the maximum on the upper layer is found, there are still $k - j - 1$ alive elements on the current layer. Thus, T_{k-j-1} time slots are required for the alive elements on the current layer. □



Assume the data elements are 1, 2 and 3. s:silence c:conflict sb:successful broadcast.

Fig. 2. The eight cases when the number of input data elements is three.

Lemma 2.

$$\left(1-\frac{1}{2^{n-1}}\right)T_n-\left(1-\frac{1}{2^{n-2}}\right)T_{n-1}$$
$$=\frac{1}{2^n}\left[C_{n-2}^{n-1}(T_{n-1}-T_{n-2})+C_{n-3}^{n-1}(T_{n-2}-T_{n-3})+\cdots+C_2^{n-1}(T_3-T_2)+C_1^{n-1}(T_2-T_1)+2T_{n-1}\right],$$

for $n \geqslant 3$.

Proof. Applying $C_n^n+C_{n-1}^n+\cdots+C_1^n+C_0^n=2^n$ into the equation in Lemma 1, we obtain

$$\left(1-\frac{1}{2^{k-1}}\right)T_k=1+\frac{1}{2^k}\Big[C_{k-1}^kT_{k-1}+C_{k-2}^kT_{k-2}+\cdots+C_i^kT_i+\cdots+C_2^kT_2$$
$$+C_1^kT_1+2^0T_{k-1}+2^1T_{k-2}+\cdots+2^{k-3}T_2+2^{k-2}T_1+2^{k-1}T_0+T_0\Big]. \tag{3}$$

Substituting $k=n$ and $T_0=1$ into Eq. (3), we get

$$\left(1-\frac{1}{2^{n-1}}\right)T_n=1+\frac{1}{2^n}\Big[1+C_{n-1}^nT_{n-1}+C_{n-2}^nT_{n-2}+\cdots+C_3^nT_3+C_2^nT_2+C_1^nT_1$$
$$+2^0T_{n-1}+2^1T_{n-2}+\cdots+2^{n-3}T_2+2^{n-2}T_1+2^{n-1}T_0\Big]. \tag{4}$$

Substituting $k=n-1$ into Eq. (3), we get

$$\left(1-\frac{1}{2^{n-2}}\right)T_{n-1}=1+\frac{1}{2^n}\Big[2+2\cdot C_{n-2}^{n-1}T_{n-2}+\cdots+2\cdot C_3^{n-1}T_3+2\cdot C_2^{n-1}T_2+2\cdot C_1^{n-1}T_1$$
$$+2^1T_{n-2}+\cdots+2^{n-3}T_2+2^{n-2}T_1+2^{n-1}T_0\Big]. \tag{5}$$

Subtracting Eq. (5) from Eq. (4), we have

$$\left(1-\frac{1}{2^{n-1}}\right)T_n-\left(1-\frac{1}{2^{n-2}}\right)T_{n-1}$$
$$=\frac{1}{2^n}\Big[-1+C_{n-1}^nT_{n-1}-C_{n-1}^{n-1}T_{n-1}-C_{n-2}^{n-1}T_{n-2}+C_{n-2}^nT_{n-2}-C_{n-2}^{n-1}T_{n-2}-C_{n-3}^{n-1}T_{n-3}+\cdots$$
$$+C_3^nT_3-C_3^{n-1}T_3-C_2^{n-1}T_2+C_2^nT_2-C_2^{n-1}T_2-C_1^{n-1}T_1+C_1^nT_1-C_1^{n-1}T_1$$
$$+2^0T_{n-1}+C_{n-1}^{n-1}T_{n-1}\Big].$$

Substituting $C_i^n-C_i^{n-1}=C_{i-1}^{n-1}$ and $T_1=1$ into the above equality, we get the equation in the lemma. □

Lemma 3.

$$\frac{4}{n}\leqslant T_n-T_{n-1}\leqslant\frac{5}{n},\quad \textit{for } n\geqslant 3.$$

Proof. We shall prove this lemma by induction on n. The proof is divided into two parts. Now we prove the first part, $T_n-T_{n-1}\leqslant\frac{5}{n}$, for $n\geqslant 3$. When $n=3$, it is trivially true since $T_3-T_2=\frac{19}{3}-5=\frac{4}{3}\leqslant\frac{5}{3}$. By hypothesis, assume that $T_3-T_2\leqslant\frac{5}{3},T_4-T_3\leqslant\frac{5}{4},\ldots,T_{n-1}-T_{n-2}\leqslant\frac{5}{n-1}$ are all true. We shall verify that $T_n-T_{n-1}\leqslant\frac{5}{n}$ is also true.

Substituting

$$T_k-T_{k-1}\leqslant\frac{5}{k},\quad \text{for } 4\leqslant k\leqslant n-1,$$
$$T_3-T_2=\frac{19}{3}-5=\frac{4}{3},$$
$$T_2-T_1=5-1=4$$

into the equality of Lemma 2, we obtain the following inequality:

$$\left(1-\frac{1}{2^{n-1}}\right)T_n-\left(1-\frac{1}{2^{n-2}}\right)T_{n-1}$$
$$\leqslant\frac{1}{2^n}\left[C_{n-2}^{n-1}\cdot\frac{5}{n-1}+C_{n-3}^{n-1}\cdot\frac{5}{n-2}+\cdots+C_3^{n-1}\cdot\frac{5}{4}+C_2^{n-1}\cdot\frac{4}{3}+C_1^{n-1}\cdot 4+2T_{n-1}\right].\tag{6}$$

Since

$$C_{n-i-1}^{n-1}\cdot\frac{k}{n-i}=\frac{k}{n}\cdot C_{n-i}^n,\quad \text{for } 1\leqslant i\leqslant n-2,$$

the expression on the right-hand side of Eq. (6) can be derived as follows:

$$\frac{1}{2^n}\left[\frac{5}{n}C_{n-1}^n+\frac{5}{n}C_{n-2}^n+\cdots+\frac{5}{n}C_4^n+\frac{4}{n}C_3^n+\frac{8}{n}C_2^n+2T_{n-1}\right]$$
$$=\frac{1}{2^n}\left[\frac{5}{n}(2^n-C_n^n-C_1^n-C_0^n)-\frac{1}{n}C_3^n+\frac{3}{n}C_2^n+2T_{n-1}\right]$$
$$=\frac{1}{2^n}\left[\frac{5}{n}2^n-\frac{5}{n}(1+1)+2T_{n-1}+\left(-\frac{5}{n}\cdot n-\frac{1}{n}C_3^n+\frac{3}{n}C_2^n\right)\right].$$

In the above expression,

$$
\begin{aligned}
-\frac{5}{n}\cdot n-\frac{1}{n}C_3^n+\frac{3}{n}C_2^n &= -5-\frac{1}{n}\cdot\frac{n(n-1)(n-2)}{6}+\frac{3}{n}\cdot\frac{n(n-1)}{2}\\
&= -\frac{(n-6)^2+5}{6}\leqslant 0,\quad \text{for } n\geqslant 3.
\end{aligned}
$$

We obtain

$$
\Big(1-\frac{1}{2^{n-1}}\Big)T_n-\Big(1-\frac{1}{2^{n-2}}\Big)T_{n-1}\leqslant\frac{1}{2^n}\Big(\frac{5}{n}2^n-\frac{5}{n}\cdot 2+2T_{n-1}\Big). \tag{7}
$$

$$
\Big(1-\frac{1}{2^{n-1}}\Big)T_n-\Big(1-\frac{1}{2^{n-1}}\Big)T_{n-1}\leqslant\Big(1-\frac{1}{2^{n-1}}\Big)\cdot\frac{5}{n}.
$$

Dividing both sides by $1-\frac{1}{2^{n-1}}$, we have

$$
T_n-T_{n-1}\leqslant\frac{5}{n},\quad \text{for } n\geqslant 3.
$$

This finishes the proof of the first part.

In the following, we shall prove the second part, $T_n-T_{n-1}\geqslant\frac{4}{n}$, for $n\geqslant 3$. When $n=3$, it is trivially true since $T_3-T_2=\frac{19}{3}-5=\frac{4}{3}\geqslant\frac{4}{3}$.

By hypothesis, assume that $T_3-T_2\geqslant\frac{4}{3}, T_4-T_3\geqslant\frac{4}{4},\ldots,T_{n-1}-T_{n-2}\geqslant\frac{4}{n-1}$ are all true. We shall verify that $T_n-T_{n-1}\geqslant\frac{4}{n}$ is also true.

Substituting the assumption

$$
T_k-T_{k-1}\geqslant\frac{4}{k},\quad \text{for } 3\leqslant k\leqslant n-1,\quad \text{and}\quad T_2-T_1=4
$$

into the equality of Lemma 2, we obtain the following inequality:

$$
\begin{aligned}
&\Big(1-\frac{1}{2^{n-1}}\Big)T_n-\Big(1-\frac{1}{2^{n-2}}\Big)T_{n-1}\\
&\geqslant\frac{1}{2^n}\Big[C_{n-2}^{n-1}\cdot\frac{4}{n-1}+C_{n-3}^{n-1}\cdot\frac{4}{n-2}+\cdots+C_3^{n-1}\cdot\frac{4}{4}+C_2^{n-1}\cdot\frac{4}{3}+C_1^{n-1}\cdot 4+2T_{n-1}\Big]\\
&=\frac{1}{2^n}\Big[\frac{4}{n}\cdot 2^n-\frac{4}{n}(1+1)+2T_{n-1}+\Big(-\frac{4}{n}\cdot n+\frac{4}{n}C_2^n\Big)\Big].
\end{aligned}
$$

Since in the above expression, $-\frac{4}{n}\cdot n+\frac{4}{n}C_2^n=2n-6\geqslant 0$, for $n\geqslant 3$, we obtain

$$
\Big(1-\frac{1}{2^{n-1}}\Big)T_n-\Big(1-\frac{1}{2^{n-2}}\Big)T_n\geqslant\frac{1}{2^n}\Big(\frac{4}{n}2^n-\frac{4}{n}\cdot 2+2T_{n-1}\Big). \tag{8}
$$

Eq. (8) is quite similar to Eq. (7), therefore, we can also obtain

$$
T_n-T_{n-1}\geqslant\frac{4}{n},\quad \text{for } n\geqslant 3.
$$

This completes the proof. □

Lemma 4.

$$
4\ln n-4<T_n<5\ln n+\frac{5}{2},\quad \textit{for } n\geqslant 3.
$$

Proof. The proof is separated into two parts. We shall show the first part, $T_n<5\ln n$, for $n\geqslant 3$.

By Lemma 3, we have

$$\begin{aligned} T_n - T_{n-1} &\leqslant \frac{5}{n}, \quad \text{for } n \geqslant 3, \\ T_{n-1} - T_{n-2} &\leqslant \frac{5}{n-1}, \\ &\dots \\ T_3 - T_2 &\leqslant \frac{5}{3}. \end{aligned}$$

Summing the above inequalities, we obtain

$$\begin{aligned} T_n - T_2 &\leqslant \frac{5}{n} + \frac{5}{n-1} + \cdots + \frac{5}{3}, \\ T_n &\leqslant 5\left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3} + \frac{1}{2} + \frac{1}{1}\right) - 5 \cdot \frac{1}{2} - 5 \cdot \frac{1}{1} + T_2 \\ &= 5\left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3} + \frac{1}{2} + \frac{1}{1}\right) - \frac{5}{2}. \end{aligned} \tag{9}$$

Let

$$H(n) = \sum_{i=1}^{n} \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}.$$

$H(n)$ is called a harmonic number and it can be expressed as follows [3]:

$$H(n) = \ln n + r + \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \varepsilon,$$

where $0 < \varepsilon < \frac{1}{252n^6}$ and r is the Euler's constant, $r = 0.57721\ldots$.

Thus,

$$\ln n < H(n) < \ln n + 1, \quad \text{for } n \geqslant 3.$$

Applying the above inequality into the Eq. (9), we have

$$T_n < 5 \ln n + \frac{5}{2}.$$

The proof of

$$T_n > 4 \ln n - 4$$

is similar. □

Since the computation before each broadcast in each processor requires only constant time and each time slot needs constant time, by Lemma 4, we have the following theorem.

Theorem 5. *The average time complexity of Algorithm Maximum-Finding is* $\Theta(\log n)$.

4. Conclusion

The layer concept [11] can help us to reduce conflict resolution when an algorithm is not conflict-free under the broadcast communication model. In this paper, we apply the layer concept to solve the maximum finding problem and get good performance. The total number of time slots, including conflict slots, empty slots and slots for successful broadcasts, is x in average, where $4\ln n - 4 < x < 5\ln n + \frac{5}{2}$.

Although the bound of number of slots required is tight, one can move the upper bound and the lower bound closer if more initial values, such as T_4 and T_5, are given.

Acknowledgements

The authors thank the reviewers for their help to simplify the description of the algorithm.

References

[1] J.I. Capetanakis, Tree algorithms for packet broadcast channels, *IEEE Trans. Inform. Theory* **25** (5) (1979) 505-515.

[2] R. Dechter and L. Kleinrock, Broadcast communications and distributed algorithms, *IEEE Trans. Comput.* **35** (3) (1986) 210-219.

[3] D.E. Knuth, *The Art of Computer Programming: Fundamental Algorithms, Vol. 1* (Addison-Wesley, Reading, MA, 1968).

[4] J.H. Huang and L. Kleinrock, Distributed selectsort sorting algorithm on broadcast communication, *Parallel Comput.* **16** (1990) 183-190.

[5] S. Levitan, Algorithms for broadcast protocol multiprocessor, in: *Proc. 3rd Internat. Conf. on Distributed Computing Systems* (1982) 666-671.

[6] S.P. Levitan and C.C. Foster, Finding an extremum in a network, in: *Proc. 1982 Internat. Symp. on Computer Architecture* (1982) 321-325.

[7] C.U. Martel, Maximum finding on a multi access broadcast network, *Inform. Process. Lett.* **52** (1994) 7-13.

[8] W.M. Moh, C.U. Martel and T.S. Moh, A dynamic solution to prioritized conflict resolution on a multiple access broadcast channel, in: *Proc. 1993 Internat. Conf. on Parallel and Distributed Systems* (1993) 414-418.

[9] C.Y. Tang and M.J. Chiu, Distributed sorting on the serially connected local area networks, in: *Proc. 1989 Singapore Internat. Conf. on Networks* (1989) 458-462.

[10] D.E. Willard, Log-logarithmic protocols for resolving ethernet and semaphore conflicts, in: *Proc. 16th Ann. ACM Symp. on Theory of Computing* (1984) 512-521.

[11] C.B. Yang, Reducing conflict resolution time for solving graph problems in broadcast communications, *Inform. Process. Lett.* **40** (1991) 295-302.

[12] C.B. Yang, R.C.T. Lee and W.T. Chen, Parallel graph algorithms based upon broadcast communications, *IEEE Trans. Comput.* **39** (12) (1990) 1468-1472.