A Fast Sorting Algorithm on Broadcast Communications

Shyue-Horng Shiau Chang-Biau Yang
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan 804, Republic of China
shiaush@cse.nsysu.edu.tw cbyang@cse.nsysu.edu.tw

Abstract

In this paper, we shall propose a fast algorithm to solve the sorting problem under the broadcast communication model, such as an Ethernet. The key point of our algorithm is to use successful broadcasts to build broadcasting layers logically and then to distribute the data elements into those logic layers properly. Thus, the number of broadcast conflicts is reduced. Suppose that there are n input data elements and n processors are available. We show that the average time complexity of our algorithm is $\Theta(n)$. Here, it is assumed that each time slot requires constant time.

1. Introduction

One of the simplest parallel computation models is the *broadcast communication model* [1, 2, 4–8, 14–16] . 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 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. For the sorting 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.

The probability concept, proposed by Martel [7], is to to estimate a proper broadcasting probability, which can reduce conflict resolution time. Another idea which can reduce conflict resolution time is the layer concept proposed by Yang [15]. To apply the layer concept for finding the maximum among a set of numbers [12], we showed the time complexity can be improved from $\Theta(\log^2 n)$ to $\Theta(\log n)$ [13].

Some researchers [2, 4, 5, 10, 17] have used the broadcast communication model to solve the sorting algorithm. The selection sort can be simulated as a straightforward method for sorting under the broadcast communication model. The method is to have all processors broadcast their elements and find the maximum element repeatedly. In other words, the maximum elements which are found sequentially form the sorting sequence. Levitan and Foster [5, 6] proposed a maximum finding algorithm, which is nondeterministic and requires $O(\log\ n)$ successful broadcasts in average. Martel [7] and Yang [12] also proposed maximum finding algorithms, which require $O(\log\ n)$ and $\Theta(\log n)$ time slots in average. Thus, the straightforward sorting method based on finding the maximum repeatedly requires $O(n \log\ n)$ successful broadcasts time in average.

Some researchers [2, 4, 5, 10, 17] have used the broadcast communication model to solve the sorting algorithm. However, their algorithms requires $O(n)$ successful broadcasts. Thus, combined with conflict resolution schemes, the required time is about $O(n \log n)$ or $(n \log\log n)$.

In this paper, we make use of loser selection method [9] to resolve conflicts (also see [11]). The method is to have each processor which gets head flips a coin repeatedly. All which get heads continue to broadcast in the next time slot, until exactly one processor gets head and broadcasts its value successfully.

In the layer concept, when there is a successful broadcast, each processor whose element is greater than the successful one brings itself up to the upper layer. Therefore those processors are divided into two layers. After several successful broadcasts occur, there are only few processors on the active layers. Thus, conflict resolution time can be

reduced. Our sorting algorithm is based on the layer concept. And we shall give the time complexity analysis of our algorithm. The average number of time slots (including conflict slots, empty slots, and slots for successful broadcast) required is $\Theta(n)$. Here, it is assumed that each time slot requires constant time.

2. The algorithm for loser selection

We shall first briefly review the loser selection algorithm, which is proposed and analyzed by H. Prodinger [9] (also see [11]). In the loser selection problem, it is assumed that a party of n persons want to select one of the members (the loser) to pay a beer for everybody. The procedure is as follows: Everybody is flipping a coin (with outputing head and tail, each with probability $\frac{1}{2}$); then, recursively, the head-party is selecting one of their members. However, there is one exception: If all persons have thrown "tails", then all of them have to repeat the procedure. 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. The loser selection algorithm is as follows:

Algorithm Loser-Selection: $SelectLoser(C)$

Step 1: Each data element in C flips a coin(The probability of getting a head is $\frac{1}{2}$).

Step 2: All which get heads form C' and each data element in C' broadcasts its value.

Step 3: There are three possible cases:

- Case 3a: If a broadcast conflict occurs then $Cr = SelectLoser(C')$.
- Case 3b: If exactly one data element successfully broadcasts its value then return Cr.
- Case 3c: If no data element broadcasts, go to step 1.(No data element in C got head.)

This whole procedure is called a round. When the conflict has been resolved and the loser has been selected, we say that "the round is over ". Let S_n denote the average number of coin flips for selecting a loser from n persons.

Pordinger used Rice's method to show that S_n has an asymptotic logarithmic bound. And he also pointed out that even though the loser selection problem seems to be very simple, the analysis is not totally trivial.

Proposition 1 *[9].*
$S_0 = S_1 = 0$*, and for* $n \geq 2$*,*

$$S_n \left(1 - 2^{-n}\right) = 1 + \sum_{k=0}^{n} 2^{-n} C_k^n S_k \, .$$

Therefore we find that $S_2 = 2, S_3 = \frac{7}{3}, S_4 = \frac{8}{3}$.

Lemma 2 *[9].*
The average depth S_n *of the tree built by* n *persons who are selecting a loser by a coin flipping process is*

$$S_n \sim \log_2 n + \frac{1}{2} - \delta(\log_2 n)$$

with the periodic function (of period 1 *and very small amplitude)*

$$\delta(x) = \frac{1}{L} \sum_{k \neq 0} \zeta(1 - \chi_k) \Gamma(1 - \chi_k) e^{2k\pi i x}$$

where $\frac{1}{L} = \log 2$ *and* $\chi_k = \frac{2k\pi i}{L}, k \in Z$ *.* $\zeta(s)$ *denotes Riemann's* ζ*-function;* $\Gamma(s)$ *denotes the* Γ*-function.*

3. The algorithm for sorting

In our sorting algorithm, each processor holds one data element initially. The key point of our algorithm is to use the successful broadcast(the loser) to build broadcasting layers and then to distribute the data elements into those layers properly.

When a successful broadcast(the loser) occurs, a new upper layer is built up. All processors which participated the conflict are divided into two layers. Each data element, except the successful one, which is greater than or equal to the successful one(the loser) will bring itself up to the new upper layer. This new layer now becomes the active layer. And others which are smaller than the successful one will stay on the current layer. After each data element on the upper layer has broadcast its value, the current layer will become the active layer again. At any time, only the processors which are on the active layer may broadcast.

Our algorithm can be represented as a recursive function: $Sorting(M)$, where M is a set of data elements (processors). Initially, each processor holds one data element and M is the set of all data elements. Our sorting algorithm is as follows:

Algorithm Sorting: $Sorting(M)$

Step 1: Each data element in M broadcasts its value.

Step 2: There are three possible cases:

- Case 2a: If exactly one data element successfully broadcasts its value, then return the only one element as the sorted sequence.
- Case 2b: If no data element broadcasts then return NULL sequence.

Case 2c: If a broadcast conflict occurs then $Cr = SelectLoser(M)$. And return $Sorting(U')$, Cr, $Sorting(L')$ as the sorted sequence, where U' contains all whose values greater than or equal to Cr (other than Cr itself), and L' contains all whose values smaller than Cr.

Figure 1 shows an example for illustrating our sorting algorithm. Initially, all data elements broadcast. Hence, in time slot 1, all of 1,2,...,8 broadcast and a conflict occurs. Then we apply the loser selection algorithm to resolve the conflict. After the first loser 4 is selected(4 is randomly selected), the layer structure is built and the party are divided into two parts. The first round is over. The larger numbers 5,6,7,8 are on the upper layer. On the other hand, the smaller numbers 1,2,3 are on the lower layer. And the upper layer becomes the active layer now. The rounds for dividing the active layer into two layers will be performed recursively. Until the active layer contains only one data element or no data element, the active layer will turn down to the lower layer.

round 1	2	3	4	5	6	7	8	9	10	11
s	s	s	n	s	n	x	n	s	x	x
$5,7,6,\hat{8}$ $3,1,2$	$7,6,8$ $\hat{5}$ $\emptyset$	$\emptyset$ $\hat{8}$ $7,6$	$\emptyset$	$\emptyset$ $\hat{7}$ 6	$\emptyset$	6	$\emptyset$	3 $\hat{2}$ 1	3	1

Figure 1. An example for the sorting algorithm. Assume the data elements are 5, 7, 3, 4, 1, 6, 2, 8. A number with a hat,such as $\hat{5}$, means that it is selected as a loser in the round. $\emptyset$ denotes an empty set, s:select a loser, n:no data element broadcasts,x: exactly one data element broadcasts.

4. Analysis of the algorithm

In this section, we shall prove that the average time complexity of our sorting algorithm is $\Theta(n)$, 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 $T_1 = 1$. We have the following recursive formula:

$$\begin{aligned} T_n = 1 + S_n + \tfrac{1}{n}[& \quad (T_0 + T_{n-1}) \\ & + (T_1 + T_{n-2}) \\ & + \cdots \\ & + (T_{n-1} + T_0) \quad]. \end{aligned} \tag{1}$$

We shall explain Eq.(1) term by term. The first term, 1, is a conflict cost that each data element in M broadcasts its value in step 1 of $Sorting(M)$. The second term, S_n, is the average number of time slots for loser selection. Then, the selected loser might be one of the n elements randomly, each case having probability $\frac{1}{n}$. The third term, $(T_0 + T_{n-1})$, is caused by that the selected loser is the largest element. Thus, the upper layer is empty and needs T_0 time slots. The lower layer contains $n-1$ data elements which are smaller than the selected loser. The number of time slots required for this lower layer is T_{n-1}. The fourth term, $(T_1 + T_{n-2})$, means that the second largest element is selected to be a loser. The final term, $(T_{n-1} + T_0)$, means that the smallest element gets a head (the loser). All other subterms have similar meanings.

Rearranging Eq. (1), we have

$$T_n = 1 + S_n + \frac{2}{n}\sum_{k=0}^{n-1} T_k \tag{2}$$

which includes two recursive functions, T_n and S_n.

Sedgewick et al. [11, p.251] specifically defined an *additive parameter* to be any parameter whose cost function satisfies the linear recursive schema. It is possible to develop a fully general treatment of the average-case analysis of any additive parameter for both of the Catalan tree model and the binary search tree model.

For binary search trees, let c_N and e_N denote the expected values over random binary search trees of size N, respectively. Then,

$c_N = e_N + \frac{2}{N}\sum_{k=1}^{N} c_{k-1}$, for $N \geq 1$ with $c_0 = e_0$, where e_N satisfies the linear recursive schema.

Although Eq. (2) is quite similar to the above equality, its additive parameter S_n isn't a linear recursive schema (Pordinger [9, theorem 14] used Rice's method to show that S_n has an asymptotic logarithmic bound). Therefore, we can not apply the result of additive parameters directly. Fortunately, Eq. (2) can be solved with basic mathematic method as follows.

Lemma 3

$$\begin{aligned} T_n = n + (\tfrac{2}{3}S_2 - \tfrac{1}{6}S_1)(n+1) - S_n \\ +2(n+1)\sum_{k=3}^{n} \tfrac{1}{k+1}(S_k - S_{k-1}) \end{aligned}$$

Proof: Rearranging Eq. (2), we have

$$T_n = 1 + S_n + \frac{2}{n}\left(\sum_{k=0}^{n-2} T_k + T_{n-1}\right) \tag{3}$$

Substituting n with $n-1$, we get

$$T_{n-1} = 1 + S_{n-1} + \frac{2}{n-1}\sum_{k=0}^{n-2} T_k$$

Rearranging the above equality, we have

$$\sum_{k=0}^{n-2} T_k = \frac{n-1}{2}\left(T_{n-1} - 1 - S_{n-1}\right)$$

Substituting the above equality into Eq. (3), we get

$$\begin{aligned} &T_n \\ &= 1 + S_n + \tfrac{2}{n}\left[\tfrac{n-1}{2}\left(T_{n-1} - 1 - S_{n-1}\right) + T_{n-1}\right] \\ &= 1 + S_n + \tfrac{n-1}{n}T_{n-1} - \tfrac{n-1}{n} - \tfrac{n-1}{n}S_{n-1} + \tfrac{2}{n}T_{n-1} \end{aligned}$$

$$T_n - \tfrac{n+1}{n}T_{n-1} = \tfrac{1}{n} + S_n - \tfrac{n-1}{n}S_{n-1}$$

Dividing both sides by $n+1$, we have

$$\begin{aligned} &\tfrac{1}{n+1}T_n - \tfrac{1}{n}T_{n-1} \\ &= \tfrac{1}{n+1}\tfrac{1}{n} + \tfrac{1}{n+1}S_n - \tfrac{1}{n+1}\tfrac{n-1}{n}S_{n-1} \\ &= (\tfrac{1}{n} - \tfrac{1}{n+1}) + \tfrac{1}{n+1}S_n - \tfrac{1}{n+1}\tfrac{n-1}{n}S_{n-1} \end{aligned}$$

By the above equality, we have

$$\begin{aligned} &\tfrac{1}{n+1}T_n - \tfrac{1}{n}T_{n-1} \\ &\qquad = (\tfrac{1}{n} - \tfrac{1}{n+1}) + \tfrac{1}{n+1}S_n - \tfrac{1}{n}\tfrac{n-1}{n+1}S_{n-1} \\ &\tfrac{1}{n}T_{n-1} - \tfrac{1}{n-1}T_{n-2} \\ &\qquad = (\tfrac{1}{n-1} - \tfrac{1}{n}) + \tfrac{1}{n}S_{n-1} - \tfrac{1}{n-1}\tfrac{n-2}{n}S_{n-2} \end{aligned}$$

$\cdots$

$$\begin{aligned} &\tfrac{1}{4}T_3 - \tfrac{1}{3}T_2 \\ &\qquad = (\tfrac{1}{3} - \tfrac{1}{4}) + \tfrac{1}{4}S_3 - \tfrac{1}{3}\tfrac{2}{4}S_2 \\ &\tfrac{1}{3}T_2 - \tfrac{1}{2}T_1 \\ &\qquad = (\tfrac{1}{2} - \tfrac{1}{3}) + \tfrac{1}{3}S_2 - \tfrac{1}{2}\tfrac{1}{3}S_1 \end{aligned}$$

Summing the above equalities, we obtain

$$\begin{aligned} &\tfrac{1}{n+1}T_n - \tfrac{1}{2}T_1 \\ &= \left(\tfrac{1}{2} - \tfrac{1}{n+1}\right) + \tfrac{1}{n+1}S_n \\ &\quad + \sum_{k=3}^{n} \tfrac{1}{k}\left(1 - \tfrac{k-1}{k+1}\right)S_{k-1} - \tfrac{1}{2}\tfrac{1}{3}S_1 \end{aligned}$$

Since $T_1 = 1$, we have

$$\begin{aligned} &\tfrac{1}{n+1}T_n \\ &= \tfrac{1}{2} + \left(\tfrac{1}{2} - \tfrac{1}{n+1}\right) + \tfrac{1}{n+1}S_n \\ &\quad + \sum_{k=3}^{n} \tfrac{1}{k}\left(1 - \tfrac{k-1}{k+1}\right)S_{k-1} - \tfrac{1}{2}\tfrac{1}{3}S_1 \\ &= \tfrac{n}{n+1} + \tfrac{1}{n+1}S_n + 2(\tfrac{1}{3}S_2 - \tfrac{1}{n+1}S_{n-1}) \\ &\quad + 2\sum_{k=3}^{n} \tfrac{1}{k+1}(S_k - S_{k-1}) - \tfrac{1}{6}S_1 \\ &= \tfrac{n}{n+1} + \tfrac{2}{3}S_2 - \tfrac{1}{6}S_1 - \tfrac{1}{n+1}S_n \\ &\quad + 2\sum_{k=3}^{n} \tfrac{1}{k+1}(S_k - S_{k-1}) \end{aligned}$$

Multiplying both sides by $n+1$, we have

$$\begin{aligned} T_n = n + (\tfrac{2}{3}S_2 - \tfrac{1}{6}S_1)(n+1) - S_n \\ + 2(n+1)\sum_{k=3}^{n} \tfrac{1}{k+1}(S_k - S_{k-1}) \end{aligned}$$

This completes the proof. □

The above lemma will be divided into two part to get the bounds of T_n. First, we will focus on solving S_n. And then we will show that $\sum_{k=3}^{n} \frac{1}{k+1}(S_k - S_{k-1})$ can be bounded.

In Lemma 2, fluctuations with a similar pattern, such as $\delta(x)$, surface in a great many areas of the analysis of algorithms. Their amplitude is usually $\ll 10^{-5}$ [3]. However, it is hard to handle that this fluctuation is applied into $\sum_{k=3}^{n} \frac{1}{k+1}(S_k - S_{k-1})$ in Lemma 3 directly. Thus, we shall apply the basic mathematic method [12], in the following, to obtain that $S_n - S_{n-1}$can be bounded within a tight range.

Lemma 4

$$\frac{1}{n} \le S_n - S_{n-1} \le \frac{2}{n}, \textit{for } n \ge 3$$

Proof:

$$\begin{aligned} S_n = 1 + \tfrac{1}{2^n}[\quad & C_n^n S_n \\ & + C_{n-1}^n S_{n-1} \\ & + \cdots \\ & + C_1^n S_1 \\ & + C_0^n S_n \quad] \end{aligned} \tag{4}$$

And,

$$\begin{aligned} S_{n-1} = 1 + \tfrac{1}{2^n}[\quad & 2C_{n-1}^{n-1} S_{n-1} \\ & + 2C_{n-2}^{n-1} S_{n-2} \\ & + \cdots \\ & + 2C_1^{n-1} S_1 \\ & + 2C_0^{n-1} S_{n-1} \quad] \end{aligned} \tag{5}$$

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

$$\begin{aligned}
&S_n - S_{n-1} \\
&= \tfrac{1}{2^n}[\quad C_n^n S_n - C_{n-1}^{n-1} S_{n-1} \\
&\quad + \quad C_{n-1}^n S_{n-1} - C_{n-1}^{n-1} S_{n-1} - C_{n-2}^{n-1} S_{n-2} \\
&\quad + \quad C_{n-2}^n S_{n-2} - C_{n-2}^{n-1} S_{n-2} - C_{n-3}^{n-1} S_{n-3} \\
&\quad + \quad \cdots \\
&\quad + \quad C_3^n S_3 - C_3^{n-1} S_3 - C_2^{n-1} S_2 \\
&\quad + \quad C_2^n S_2 - C_2^{n-1} S_2 - C_1^{n-1} S_1 \\
&\quad + \quad C_1^n S_1 - C_1^{n-1} S_1 \\
&\quad + \quad C_0^n S_n - C_0^{n-1} S_{n-1} - C_0^{n-1} S_{n-1} \quad].
\end{aligned}$$

Substituting $C_i^n - C_i^{n-1} = C_{i-1}^{n-1}$ and $S_1 = 0$ into the above equality, we get

$$\begin{aligned}
&(1 - \tfrac{1}{2^{n-1}})\,(S_n - S_{n-1}) \\
&= \tfrac{1}{2^n}[\quad C_{n-2}^{n-1}(S_{n-1} - S_{n-2}) \\
&\qquad + C_{n-3}^{n-1}(S_{n-2} - S_{n-3}) \\
&\qquad + \cdots \\
&\qquad + C_2^{n-1}(S_3 - S_2) \\
&\qquad + C_1^{n-1}(S_2 - S_1) \\
&\qquad - S_{n-1} \quad].
\end{aligned} \tag{6}$$

We shall prove this lemma by induction on n. The proof is divided into two parts. Now we prove the first part, $S_n - S_{n-1} \le \frac{2}{n}$, for $n \ge 3$. When $n = 3$, it is trivially true since $S_3 - S_2 = \frac{7}{3} - 2 = \frac{1}{3} \le \frac{2}{3}$. By hypothesis, assume that $S_3 - S_2 \le \frac{2}{3}, S_4 - S_3 \le \frac{2}{4}, \cdots, S_{n-1} - S_{n-2} \le \frac{2}{n-1}$ are all true. We shall verify that $S_n - S_{n-1} \le \frac{2}{n}$ is also true.

Substituting

$$\begin{aligned}
&S_k - S_{k-1} \le \tfrac{2}{k}, \text{ for } 4 \le k \le n-1, \\
&S_3 - S_2 = \tfrac{7}{3} - 2 = \tfrac{1}{3}, \\
&S_2 - S_1 = 2 - 0 = 2
\end{aligned}$$

into Eq. (6), we obtain the following inequality

$$\begin{aligned}
&(1 - \tfrac{1}{2^{n-1}})\,(S_n - S_{n-1}) \\
&\le \tfrac{1}{2^n}[\quad \tfrac{2}{n-1} C_{n-2}^{n-1} \\
&\qquad + \tfrac{2}{n-2} C_{n-3}^{n-1} \\
&\qquad + \cdots \\
&\qquad + \tfrac{2}{4} C_3^{n-1} \\
&\qquad + \tfrac{1}{3} C_2^{n-1} \\
&\qquad + 2C_1^{n-1} - S_{n-1}]
\end{aligned} \tag{7}$$

Since

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

the expression of the right hand of Eq. (7) can be derived as follows:

$$\begin{aligned}
&\tfrac{1}{2^n}[\quad \tfrac{2}{n-1} C_{n-2}^{n-1} \\
&\qquad + \tfrac{2}{n-2} C_{n-3}^{n-1} \\
&\qquad + \cdots \\
&\qquad + \tfrac{2}{4} C_3^{n-1} \\
&\qquad + (\tfrac{2}{3} - \tfrac{1}{3}) C_2^{n-1} \\
&\qquad + (\tfrac{2}{2} + \tfrac{2}{2}) C_1^{n-1} - S_{n-1}] \\
&= \tfrac{1}{2^n}[\quad \tfrac{2}{n} C_{n-1}^n \\
&\qquad + \tfrac{2}{n} C_{n-2}^n \\
&\qquad + \cdots \\
&\qquad + \tfrac{2}{4} C_3^n \\
&\qquad + \tfrac{2}{n} C_2^n \\
&\qquad - \tfrac{1}{3} C_2^{n-1} + \tfrac{2}{2} C_1^{n-1} - S_{n-1}] \\
&= \tfrac{1}{2^n}[\quad \tfrac{2}{n}(2^n - 2) - \tfrac{2}{n} C_1^n - \tfrac{1}{3} C_2^{n-1} + \tfrac{2}{2} C_1^{n-1} - S_{n-1}]
\end{aligned}$$

In the above expression, it can be easily verified that

$$-\tfrac{2}{n} C_1^n - \tfrac{1}{3} C_2^{n-1} + \tfrac{2}{2} C_1^{n-1} \le 0$$

,for $n \ge 3$ and n is an integer.

We obtain

$$\begin{aligned}
&(1 - \tfrac{1}{2^{n-1}})(S_n - S_{n-1}) \\
&\le \tfrac{1}{2^n}\left[\tfrac{2}{n}(2^n - 2)\right] \\
&= (1 - \tfrac{1}{2^{n-1}}) \cdot \tfrac{2}{n}
\end{aligned}$$

Dividing both sides by $1 - \frac{1}{2^{n-1}}$, we have $S_n - S_{n-1} \le \frac{2}{n}$,for $n \ge 3$. This finished the proof of the first part.

Since the second part, $S_n - S_{n-1} \ge \frac{1}{n}$, for $n \ge 3$ is quite similar to the first part, the proof of the second part is omitted. □

Lemma 5

$$\frac{1}{3} - \frac{1}{n+1} \le \sum_{k=3}^{n} \frac{1}{k+1}(S_k - S_{k-1}) \le 2(\frac{1}{3} - \frac{1}{n+1})$$

Proof: By Lemma 4, we get

$$\begin{aligned}
&\sum_{k=3}^{n} \tfrac{1}{k+1}(S_k - S_{k-1}) \\
&= \tfrac{1}{n+1}(S_n - S_{n-1}) + \tfrac{1}{n}(S_{n-1} - S_{n-2}) \\
&\quad + \cdots + \tfrac{1}{5}(S_4 - S_3) + \tfrac{1}{4}(S_3 - S_2) \\
&\le \tfrac{1}{n+1}\tfrac{2}{n} + \tfrac{1}{n}\tfrac{2}{n-1} + \cdots + \tfrac{1}{5}\tfrac{2}{4} + \tfrac{1}{4}\tfrac{2}{3} \\
&= 2(\tfrac{1}{3} - \tfrac{1}{n+1})
\end{aligned}$$

The proof of $\frac{1}{3} - \frac{1}{n+1} \le \sum_{k=3}^{n} \frac{1}{k+1}(S_k - S_{k-1})$ is similar. □

Lemma 6

$$3n - S_n \le T_n \le 3\frac{2}{3}n - S_n - \frac{4}{3}$$

Proof: Combining Lemma 3 and Lemma 5, we have

$$n + \frac{4}{3}(n+1) - S_n + 2(n+1)(\frac{1}{3} - \frac{1}{n+1}) \leq T_n$$
$$\text{and, } T_n \leq n + \frac{4}{3}(n+1) - S_n + 4(n+1)(\frac{1}{3} - \frac{1}{n+1})$$

Thus, $3n - S_n \leq T_n \leq 3\frac{2}{3}n - S_n - \frac{4}{3}$. □

Since the computation before each broadcast in each processor requires only constant time and each time slot needs constant time, by Lemma 6 and Lemma 2, we have the following theorem.

Theorem 7

The average time complexity of Algorithm Sorting is $\Theta(n)$.

5. Conclusion

The layer concept [15] 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 sorting problem and get good performance. The total number of time slots, including conflict slots, empty slots and slots for successful broadcasts, is $\Theta(n)$.

An *additive parameter* [11, p.251] specifically defined an *additive parameter* to be any parameter whose cost function satisfies the linear recursive schema. In Eq. (2), its additive parameter S_n isn't a linear recursive schema. Therefore, we can not apply the result of additive parameters directly. Although we bounded it within a tight range using basic mathematic method, we hope we can show its asymptotic bound in the near future. And to generalize our sorting algorithm, such as to select k largest numbers, is also our future work.

References

[1] J. I. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25(5):505–515, May 1979.

[2] R. Dechter and L. Kleinrock. Broadcast communications and distributed algorithms. *IEEE Transactions on Computers*, 35(3):210–219, Mar. 1986.

[3] P. Flajolet and R. Sedgewick. Mellin transforms and asymptotics: Finite differences and rice's integrals. *Theoretical Computer Science*, 144:101–124, 1995.

[4] J. H. Huang and L. Kleinrock. Distributed selectsort sorting algorithm on broadcast communication. *Parallel Computing*, 16:183–190, 1990.

[5] S. Levitan. Algorithms for broadcast protocol multiprocessor. In *Proc. of 3rd International Conference on Distributed Computing Systems*, pages 666–671, 1982.

[6] S. P. Levitan and C. C. Foster. Finding an extremum in a network. In *Proc. of 1982 International Symposium on Computer Architechure*, pages 321–325, 1982.

[7] C. U. Martel. Maximum finding on a multi access broadcast network. *Information Processing Letters*, 52:7–13, 1994.

[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. of 1993 International Conference on Parallel and Distributed Systems*, pages 414–418, 1993.

[9] H. Prodinger. How to select a loser. *Discrete Mathematics*, 120:149–159, 1993.

[10] K. V. S. Ramarao. Distributed sorting on local area network. *IEEE Transactions on Computers*, C-37(2):239–243, Feb. 1988.

[11] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Company, USA, 1996.

[12] S. H. Shiau and C. B. Yang. A fast maximum finding algorithm on broadcast communication. *Information Processing Letters*, 60:81–96, 1996.

[13] S. H. Shiau and C. B. Yang. The layer concept and conflicts on broadcast communication. *Journal of Chang Jung Christian University*, 2(1):37–46, June 1998.

[14] C. Y. Tang and M. J. Chiu. Distributed sorting on the serially connected local area networks. In *Proc. of 1989 Singapore International Conference on Networks*, pages 458–462, 1989.

[15] C. B. Yang. Reducing conflict resolution time for solving graph problems in broadcast communications. *Information Processing Letters*, 40:295–302, 1991.

[16] C. B. Yang, R. C. T. Lee, and W. T. Chen. Parallel graph algorithms based upon broadcast communications. *IEEE Transactions on Computers*, 39(12):1468–1472, Dec. 1990.

[17] C. B. Yang, R. C. T. Lee, and W. T. Chen. Conflict-free sorting algorithm broadcast under single-channel and multichannel broadcast communication models. In *Proc. of International Conference on Computing and Information*, pages 350–359, 1991.