PAPER

Generalization of Sorting in Single Hop Wireless Networks*

Shyue-Horng SHIAU†,†† ***and*** **Chang-Biau YANG**†a), ***Nonmembers***

SUMMARY The generalized sorting problem is to find the first k largest elements among n input elements and to report them in a sorted order. In this paper, we propose a fast generalized sorting algorithm under the single hop wireless networks model with collision detection (WNCD). The algorithm is based on the maximum finding algorithm and the sorting algorithm. 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. We prove that the average time complexity required for our generalized sorting algorithm is $\Theta(k + \log(n - k))$. When $k = 1$, our generalized sorting algorithm does the work of finding maximum, and when $k = n$, it does the work of sorting. Thus, the analysis of our algorithm builds a connection between the two extremely special cases which are maximum finding and sorting.

key words: *parallel algorithm, wireless, sorting, broadcast communication, conflict, generalized sorting*

1. Introduction

In recent years, due to the rapid adoption of wireless personal communication, the research on wireless networks (WN) [1], [2], [10]–[13] has become more active. The single hop WN model consists of n stations sharing a common radio frequency channel for communicating with each other. Figure 1 illustrates an example of eight stations.

Each station in this model can communicate with others only through the common channel. Whenever a station broadcasts messages, any other station can hear the broadcast messages via the common channel. If more than one station wants to broadcast messages simultaneously, a *broadcast conflict* occurs. In the model, it is assumed that each station has the capability to detect collision. When a conflict occurs and is detected, a conflict resolution scheme should be invoked to resolve the conflict. This resolution scheme will enable one of the broadcasting stations to broadcast successfully.

Nakano and Olariu [11] distinguished the single hop WN models from the capability with *collision detection* (CD). In the single hop WN model with CD (WNCD), exactly one of three statuses on a radio channel can be detected by each station. The three statuses are as follows:

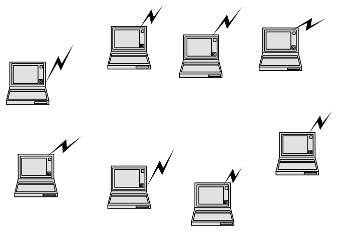
Fig. 1 An example for 8 stations.

- NULL: No station makes a transmission.
- SINGLE: Exactly one station makes a transmission.
- COLLISION: Two or more stations make transmissions simultaneously.

The WNCD model is one of the simplest parallel computation models which have been studied for more than two decades. Some researchers regarded the model as the broadcast communication model [4]–[7], [9], [13], [15]–[17], [19]–[21], [23], [24]. In the WNCD model some important and essential components were investigated such as the sorting problem [3]–[5], [8], [14], [24], maximum finding problem [7], [15] and graph algorithm [21]–[23]. The initialization problem [10], [11] was also discussed. In this paper, we shall focus on a generalization for the sorting problem in the WNCD model. The generalization was discussed by Martel and Moh [8].

In the WNCD model, to solve a problem with an algorithm, the required time 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 solving a problem under the WNCD model, it does not seem that transmission time and computation time can be reduced. Therefore, to minimize resolution time is the key issue to improve the time complexity. In this paper, we assume that the number of stations is unknown and each station does not have a unique identification.

Reducing conflict resolution time can be achieved by applying two concepts. The first concept is to dynamically estimate a proper broadcasting probability. Using the dynamic probability concept, Martel [7] proposed an efficient maximum finding algorithm, which improves the time complexity from $O(\log^2 n)$ to $O(\log n)$, where n is the number of data elements and there are n stations available. By using the

Manuscript received January 11, 2005.
Manuscript revised September 19, 2005.
†The authors are with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan 804.
††The author is with the Department of Computer Aided Media Design, Chung Jung University, Tainan, Taiwan.
*This research work was partially supported by the National Science Council of the Republic of China under NSC89-2213-E110-005.
a) E-mail: cbyang@cse.nsysu.edu.tw
DOI: 10.1093/ietisy/e89–d.4.1432

same concept, Micic and Stojmenovic [10] efficiently solved the initialization problem. The improvement is from en to $2.15n$, where n is the number of stations. The second concept to reduce conflict resolution time is the layer concept proposed by Yang [21]. To apply the layer concept for finding the maximum among a set of n numbers [15], we can improve the time complexity from $\Theta(\log^2 n)$ to $\Theta(\log n)$ [16]. Thus applying the two concepts, we can improve the time complexity for solving other problems in the WNCD model.

A straightforward method for solving the sorting problem under the WNCD model is to simulate the selection sort directly. The method is to have all stations broadcast their elements and to find maxima repeatedly. In other words, the maximum elements which are found sequentially form the sorted sequence. Thus, the time required for this straightforward sorting method is $O(n \log n)$.

A modified version of selection sort was proposed by Martel and Moh [8]. The main difference from the straightforward method is that each of the maxima comes out from the result of the previous iteration. It can also be viewed as a modified quicksort. They showed that their sorting algorithm requires $O(n)$ time in average. They also proved that the expected time for finding the first k largest elements is $O(k + \log n)$. Though their algorithm is fast under the WNCD model, the constant bounds associated with $O(n)$ and $O(k + \log n)$ are not very tight.

In this paper, we apply the layer concept to the sorting algorithm. In the sorting algorithm, we make use of the maximum finding method [15] to build some logic layers and reduce conflicts. After the process of maximum finding terminates, we can use these successful broadcasts to build broadcasting layers logically and then to distribute the data elements into those logic layers properly. Thus, conflict resolution time can be reduced. The average number of time slots (including conflict slots, empty slots, and slots for successful broadcast) required for sorting n elements is T_n, where $\frac{10}{3}n - \frac{2}{3} \leq T_n \leq \frac{11}{3}n - \frac{4}{3}$. Here, it is assumed that each time slot requires constant time. As we can see that the bound of our time complexity is very tight.

We also extend the sorting algorithm to the generalized sorting, which is to find the first k largest elements among n stations. The average number of time slots required for our generalized sorting algorithm is T_k^n, where $\frac{10}{3}k + 4\ln(n-(k-2)) - \left(\frac{2}{3} + 4\ln 2\right) \leq T_k^n \leq \frac{11}{3}k + 5\ln(n-(k-1)) - \frac{4}{3}$. The bound of the time complexity is also very tight. In this paper, we will omit the proofs of the lemmas and theorems, which can be found in our technique report [18].

On the high level view, our sorting algorithm and generalization algorithm have a similar approach as that proposed by Martel and Moh [8]. On the low level view, our algorithms are based on the layer concept, thus they can be analyzed easily and a tighter bound can be obtained.

In addition, by the analysis of our generalized sorting, we figure out the whole process of the sorting. The whole process not only shows the final result of the sorting, but also includes each step of finding each element (from the maximum element to the minimum element). In other words, when one needs sort only the first k largest elements, not all elements, our analysis can predict what time complexity will be and which method will be better, if other methods also provide the generalized analysis.

Finally, we make two simulations. The first simulation shows that our generalized sorting algorithm fits the theoretical bound from $k = 1$ to $k = n$. The second simulation shows that our algorithm outperforms Martel and Moh's algorithm [8] from $k = 1$ to $k = n$. If we slightly modify their original algorithm, the simulation shows that their algorithm has a little bit better performance than our algorithm when $k \simeq n$. But at other values of k, our algorithm still outperforms the modified algorithm.

This paper is organized as follows. In Sect. 2, we shall give the notations used in this paper and review the previous results for maximum finding [15]. In Sects. 3 and 4, we shall present our sorting algorithm and give its analysis. In Sects. 5 and 6, we shall present our generalized sorting algorithm and give the analysis of the generalization. The two simulations will be given in Sect. 7. Finally, Sect. 8 concludes the paper.

2. Notations and the Previous Result for Maximum Finding

We assume that there are n distinct input elements and n stations are available in the single hop WNCD model. Note that the value of n is unknown. And each station holds exactly one data element. The *generalized sorting* problem here is to find the first k largest element in the nonincreasing order.

In the WNCD model, the timing is slotted. When exactly one station broadcasts message in one time slot, it is a *successful slot*. When no station broadcasts in one time slot, it is an *empty slot*. If two or more stations broadcast in one time slot, it is a *conflict slot*. In other words, every time slot can have one of the three statuses. It is assumed that each station can realize the status of every time slot.

We shall use the notation $\langle x_0, \cdots, x_t \rangle$ to denote a linked list, where x_t is the head of the list. Suppose $L_1 = \langle x_0, \cdots, x_t \rangle$ and $L_2 = \langle y_0, \cdots, y_{t'} \rangle$, then $\langle L_1, L_2 \rangle$ represents the concatenation of L_1 and L_2, which is $\langle x_0, \cdots, x_t, y_0, \cdots, y_{t'} \rangle$. And for shortening the description in our algorithm, we use the term "data element" to represent "the station storing the data element" and to represent the data element itself in different situation if there is no ambiguity.

In the maximum finding algorithm of our previous result [15], the key point is to use broadcast conflicts to build broadcasting layers and then to distribute the data elements into those layers.

When a broadcast conflict occurs, 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

Algorithm 1 Algorithm Modified Maximum-Finding: $Maxfind(C, L_m)$

```
Step 1: Each alive data element in $C$ broadcasts its value. Note that
    a data element is alive if its value is greater than the head of $L_m$.
    Each element smaller than or equal to the head drops out (becomes
    dead).
Step 2: There are three possible cases:
    Case 2a: If a broadcast conflict occurs, then do the following.
        Step 2a.1: Each alive data element in $C$ flips a coin. All
            which get heads form $C'$ and bring themselves to the
            upper layer, and the others form $C''$ and stay on the
            current layer.
        Step 2a.2: Perform $L_m = Maxfind(C', L_m)$.
        Step 2a.3: Perform $L_m = Maxfind(C'', L_m)$.
        Step 2a.4: Return $L_m$.
    Case 2b: If exactly one data element successfully broadcasts its
        value $y$, then return $\langle L_m, y\rangle$.
    Case 2c: No data element broadcasts. This case results from that
        either no alive data element chooses to broadcast or each
        data element in the active layer is dead. There are two sub-
        cases for the situation of the last time slot which is not silent
        as follows:
        Subcase A: It is a conflict. Before the next time slot com-
            ing, each alive data element which causes the conflict
            will randomly, with probability $\frac{1}{2}$, chooses to con-
            tinue or not again. On the current layer, each data ele-
            ment which chooses to continue broadcasts its value.
            Then go to step 2.
        Subcase B: It is a successful broadcast. Change the active
            layer to the lower layer. If the active layer is not below
            the bottom, then go to step 2; otherwise Return $L_m$.
```

themselves up to a new upper layer, then the new upper layer is built up. 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 stations which are alive and on the active layer may broadcast.

The algorithm can be represented as a recursive function $Maxfind(C, L_m)$, where C represents a set of data elements (stations). Initially, each station holds one alive data element and sets an initial linked list $L_m = \langle x_0\rangle$. After the end of the algorithm, we can get a sequence of successful broadcasts, which is represented by the linked list $L_m = \langle x_0, x_1, x_2, \cdots, x_{t-1}, x_t\rangle$, where $x_{i-1} < x_i$, $1 \le i \le t$ and $x_0 = -\infty$. Note that L_m is held by each station and the head x_t is the maximum. We slightly revise the algorithm [15] for maximum finding to $Maxfind(C, L_m)$ as follows:

Using Shiau and Yang's [15] proof method, we can easily prove that the average number of time slots, including conflict slots, empty slots and successful slots, M_n, for $Maxfind(C, L_m)$ is bounded as follows:

$$4\ln n - \frac{3}{2} < M_n < 5\ln n + 2. \tag{1}$$

Note that Eq. (1) is slightly different from the original result [15] since the algorithm is slightly modified. And the differential of M_n is as follows:

$$\frac{4}{n} \le M_n - M_{n-1} \le \frac{5}{n}. \tag{2}$$

Algorithm 2 Algorithm Sorting: $Sorting(S)$

```
Step 1: Each data element in $S$ broadcasts its value.
Step 2: There are three possible cases:
    Case 2a: If exactly one data element successfully broadcasts its
        value $y$, then return $\langle y\rangle$.
    Case 2b: If no data element broadcasts, then return $NULL$.
    Case 2c: If a broadcast conflict occurs, then perform $L_m =$
        $Maxfind(S, \langle x_0\rangle)$.
        Step 2c.1: Set $L_s = NULL$.
        Step 2c.2: For $i = t$ down to 1
                Set $L_s = \langle x_i, L_s\rangle$.
                do $L_s = \langle Sorting(S_i), L_s\rangle$.
        Step 2c.3: Return $L_s$.
```

In this paper, we shall use the differential, $M_n - M_{n-1}$, and

$$M_2 = \frac{9}{2}. \tag{3}$$

3. Our Sorting Algorithm

Our sorting algorithm is based on the maximum finding algorithm. By using the maximum finding algorithm, we can get serial elements in the sequence of successful broadcasts. The serial elements are regarded as separators to build broadcasting layers and to distribute the remaining data elements into those layers properly. Then each layer performs the sorting algorithm recursively until the layer is empty or contains exactly one data element.

The algorithm can be represented as a recursive function: $Sorting(S)$, where S represents a set of data elements (stations). Each station holds one data element initially and maintains a linked list L_s. In the end, we will get a sorted linked list L_s. After $L_m = Maxfind(S, \langle x_0\rangle)$ is performed, a series of successful broadcasts, $\langle x_0, x_1, x_2, \cdots, x_{t-1}, x_t\rangle$, will appear from x_1 to x_t, where $x_{i-1} < x_i$, $1 \le i \le t$ and $x_0 = -\infty$. And $S_i = \{x | x_{i-1} < x < x_i\}$ represents a set of data elements which are bounded by x_{i-1} and x_i. Our sorting algorithm is as follows:

In the above algorithm, x_1 is the element of the first successful broadcast, and x_t is the largest element in S. For example, suppose that S has 2 data elements $\{1, 2\}$. We can get a series of successful broadcasts L_m after $Maxfind(S, \langle x_0\rangle)$ is performed. The first successful broadcast may be one of the two elements randomly, each case having probability $\frac{1}{2}$. If the first successful broadcast is 2, then $L_m = \langle x_0, 2\rangle$, and the final sorted linked list is $\langle x_0, Sorting(S_1), 2\rangle$, where $S_1 = \{x | x_0 < x < x_1\} = \{x | -\infty < x < 2\} = \{1\}$. If the first successful broadcast is 1, then $L_m = \langle x_0, 1, 2\rangle$, and the final sorted linked list is $\langle x_0, Sorting(S_1), 1, Sorting(S_2), 2\rangle$, where $S_1 = \{x | x_0 < x < x_1\} = \{x | -\infty < x < 1\} = NULL$ and $S_2 = \{x | x_1 < x < x_2\} = \{x | 1 < x < 2\} = NULL$.

Martel *et al.* [8] showed the sorting problem can be solved in $O(n)$ time. They also proved that the expected time for finding the first k largest numbers is $O(k + \log n)$.

However, the constant bounds associated with $O(n)$ and $O(k + \log n)$ proved by Martel *et al.* [8] are not very tight.

4. Analysis of the Sorting 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 n data elements held by at least n processors in which each processor holds at most one data element. Let T_n denote the average number of required time slots, including conflict slots, empty slots and slots for successful broadcasts, 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 $T_1 = 1$.

If $n = 2$, we have the following recursive formula:

$$T_2 = M_2 + R_2, \text{ where } R_2 = \left\{ \begin{array}{lll} & \frac{1}{2}[& T_1] \\ + & \frac{1}{2}[T_0 & +T_0] \end{array} \right\}. \quad (4)$$

In the first equation, the first term M_2 is the average number of time slots required for the maximum finding algorithm while $n = 2$. The second term R_2 is the average number of time slots required for finishing the sorting after the maximum finding terminates.

In the equation for R_2, the first successful broadcast may be either one of the 2 elements randomly, each case having probability $\frac{1}{2}$. The subterm of first term, T_1, arises when the first successful broadcast is the largest element. Thus, the layer between $x_0 = -\infty$ and the largest element contains one element, which is the second largest element, and T_1 time slots is needed.

In second term, the subterm, $[T_0 + T_0]$, arises when the first successful broadcast is the second largest element and the second successful broadcast is the largest element. Thus, the layers between $x_0 = -\infty$, the second and the largest element are both empty and T_0 time slots are needed in both layers.

By Eq. (3), we have

$$T_2 = \frac{9}{2} + \left\{ \frac{1}{2}[1] + \frac{1}{2}[1 + 1] \right\} = 6.$$

When $n = 3$, we also have the following recursive formula:

$$T_3 = M_3 + R_3, \text{ where } R_3 = \left\{ \begin{array}{lll} & \frac{1}{3}[& T_2] \\ + & \frac{1}{3}[T_0 & +T_1] \\ + & \frac{1}{3}[R_2 & +T_0] \end{array} \right\}. \quad (5)$$

In the equation for R_3, the first successful broadcast may be any one of the 3 elements randomly, each case having probability $\frac{1}{3}$. The subterm of the first term, T_2, arises when the first successful broadcast is the largest element. Thus, the layer between $x_0 = -\infty$ and the largest element contains two elements, which is the second and third largest element, and it needs T_2 time slots for sorting.

In the second term, the subterm, $[T_0 + T_1]$, arises when the first successful broadcast is the second largest element and the second successful broadcast is the largest element. Thus, the layer between the second and the largest element is empty and it needs T_0 time slots. And the layer between $x_0 = -\infty$ and the second largest element contains one element and it needs T_1 time slots.

In third term, the subterm, $[R_2 + T_0]$, arises when the first successful broadcast is the third largest element. Then, two elements (largest and second largest) remains and they may broadcast in the next time slot. Thus, this situation is the same as that of R_2 in Eq. (4). After the situation represented in R_2 is resolved, finally, the layer between $x_0 = -\infty$ and the third largest element is empty and T_0 time slots are required.

Generalizing Eq. (5), we have

$$T_n = M_n + R_n, \text{ where}$$

$$R_n = \left\{ \begin{array}{lll} & \frac{1}{n}[& T_{n-1}] \\ + & \frac{1}{n}[T_0 & +T_{n-2}] \\ + & \frac{1}{n}[R_2 & +T_{n-3}] \\ + & \cdots & \\ + & \frac{1}{n}[R_{k-1} & +T_{n-k}] \\ + & \cdots & \\ + & \frac{1}{n}[\,R_{n-1} & +T_0] \end{array} \right\}. \quad (6)$$

In the above equations, M_n represents the average number of time slots required for finding the maximum among n data elements in $Maxfind(C, L_m)$, and R_n represents the average number of time slots required for finishing sorting after the maximum is found.

In the equation for R_n, the first successful broadcast may be any one of the n elements randomly, each case having probability $\frac{1}{n}$. The subterm T_{n-1} of the first term arises when the first successful broadcast is the largest element. Thus, the layer between $x_0 = -\infty$ and the largest element contains $n-1$ elements and it needs T_{n-1} time slots for sorting.

In the second term, the subterm, $T_0 + T_{n-2}$, arises when the first successful broadcast is the second largest element. Thus, the layer between the second and the largest element is empty and it needs T_0 time slots. And the layer between $x_0 = -\infty$ and the second largest element contains $n - 2$ elements and it needs T_{n-2} time slots for sorting.

In the kth term, the subterm, $R_{k-1} + T_{n-k}$, arises when the first successful broadcast is the kth largest element. Thus, R_{k-1} is the number of time slots required for finishing the sorting on the $k - 1$ elements after the maximum is

Table 1 An example of the generalized sorting algorithm with 10 elements.

S			k	L_m	t	i	L_g	c	*Return*
{5,1,7,0,9,2,8,3,6,4}			6	{2,6,8,9}			NULL	0	
	{}		5		4	1	⟨9⟩	1	
									(NULL,0)
							⟨9⟩	1	
	{7}		4		4	2	⟨8,9⟩	2	
									(⟨7⟩,1)
							⟨7,8,9⟩	3	
	{5,3,4}		2	{3,5}	4	3	⟨6,7,8,9⟩	4	
		{4}	1		2	1	⟨5⟩	1	
									(⟨4⟩,1)
							⟨4,5⟩	2	
		{}	2		2	2	⟨4,5⟩	2	(⟨4,5⟩,2)
					4	3	⟨4,5,6,7,8,9⟩	6	
	{1,0}		6		4	4	⟨4,5,6,7,8,9⟩	6	(⟨4,5,6,7,8,9⟩,6)

found. And the layer between $x_0 = -\infty$ and the first successful broadcast element contains $n-k$ data elements and it needs T_{n-k} time slots for sorting $n-k$ data elements.

Substituting $R_k = T_k - M_k$, $2 \le k \le n-1$ into Eq. (6), we have

$$T_n = M_n + \left\{ \begin{array}{lll} & \frac{1}{n}[& T_{n-1}] \\ + & \frac{1}{n}[T_0 & +T_{n-2}] \\ + & \frac{1}{n}[(T_2 - M_2) & +T_{n-3}] \\ + & \cdots & \\ + & \frac{1}{n}[(T_{k-1} - M_{k-1}) & +T_{n-k}] \\ + & \cdots & \\ + & \frac{1}{n}[(T_{n-1} - M_{n-1}) & +T_0] \end{array} \right\}. \quad (7)$$

Regrouping Eq. (7), we obtain

$$T_n = M_n + \frac{1}{n}\left[T_0 + \sum_{2\le k\le n-1}(T_k - M_k) + \sum_{0\le k\le n-1} T_k\right]. \quad (8)$$

Lemma 1:

$$\frac{1}{n+1}T_n - \frac{1}{3}T_2 = \sum_{3\le k\le n} \frac{1}{k+1}(M_k - M_{k-1}).$$

Lemma 2:

$$4\left(\frac{1}{3} - \frac{1}{n+1}\right) \le \sum_{3\le k\le n} \frac{1}{k+1}(M_k - M_{k-1})$$
$$\le 5\left(\frac{1}{3} - \frac{1}{n+1}\right).$$

Theorem 3:

$$\frac{10}{3}n - \frac{2}{3} \le T_n \le \frac{11}{3}n - \frac{4}{3}.$$

Martel and Moh [8] also showed their sorting algorithm requires $O(n)$ time. As we can see in Theorem 3, it has a tighter bound. It is clear that in our sorting algorithm, each station needs $O(n)$ space to store and to maintain the sorted linked list L_s.

Algorithm 3 Algorithm Generalized Sorting: $(L_g, c) = GSorting(S, k)$

```
Step 1: Each data element in $S$ broadcasts its value.
Step 2: There are three possible cases:
    Case 2a: If exactly one data element successfully broadcasts its
        value $y$, then return $(\langle y \rangle, 1)$.
    Case 2b: If no data element broadcasts, then return $(NULL, 0)$.
    Case 2c: If a broadcast conflict occurs, then perform $L_m =$
        $Maxfind(S, \langle x_0 \rangle)$.
        Step 2c.1: Set $L_g = NULL$ and $c = 0$.
        Step 2c.2: For $i = t$ down to 1
            If $k \le c$, then goto Step 2c.3.
            Set $L_g = \langle x_i, L_g \rangle$ and $c = c + 1$.
            If $k > c$, then do
            $(L'_g, c') = GSorting(S_i, k - c)$.
            Set $L_g = \langle L'_g, L_g \rangle$ and $c = c' + c$.
        Step 2c.3: Return $(L_g, c)$.
```

5. The Generalized Sorting Algorithm

For given n distinct elements, the generalization of sorting is to find the first k, $k \ge 1$, largest elements in the nonicressing order. Our generalized sorting algorithm can be represented as a recursive function: $(L_g, c) = GSorting(S, k)$, where L_g is the linked list storing the sorted sequence, c is the number of elements of L_g, and S represents a set of unsorted data elements (stations). In the algorithm, each station holds exactly one data element and maintains a linked list L_g. Our algorithm is as follows:

For example, the input file contains 10 elements, which is $\{5, 1, 7, 0, 9, 2, 8, 3, 6, 4\}$. Suppose we want to sort the first 6 largest elements. The whole process is shown in Table 1.

After the maximum element 9 is found, all elements are separated into four parts by $2, 6, 8, 9$ properly which are $\{\}, \{7\}, \{5, 3, 4\}$ and $\{1, 0\}$. And the algorithm is recursively called with each of these four subsets.

On the high level view, both our sorting and generalization algorithms have a similar approach as that proposed by Martel and Moh [8]. However, on the low level view, probability plays a kernel role in their algorithm, and our al-

Table 2 The process from maximum finding to sorting.

k	T^n_k	Results	Reference
$k = 1$	$T^n_{k=1} = M_n$	$4\ln n - \frac{3}{2} < M_n < 5\ln n + 2$	Eq. (1)
$2 \le k \le n$	$T^n_{2\le k\le n}$	$\frac{10}{3}k + 4\ln(n-(k-2)) - \left(\frac{2}{3} + 4\ln 2\right) \le T^n_k \le \frac{11}{3}k + 5\ln(n-(k-1)) - \frac{4}{3}$	Theorem 6
$k = n$	$T^n_{k=n} = T_n$	$\frac{10}{3}n - \frac{2}{3} \le T_n \le \frac{11}{3}n - \frac{4}{3}$	Theorem 3

n: number of nodes k: the first k largest numbers

gorithms are based on the layer concept. This is the main difference between Martel's and our algorithms. Besides, our algorithms can be analyzed easily. And a tighter bound will be obtained in the next section.

6. Analysis of the Generalized Sorting

Let T^n_k denote the average number of time slots, including conflict slots, empty slots and successful slots, required for the generalized sorting algorithm $GSorting(S,k)$. T_n in Eq. (7) can be regarded as T^n_n. Therefore, generalizing Eq. (7), we obtain

$T^n_k = M_n$, if $k = 1$,

if $2 \le k \le n$,

$$T^n_k = M_n + \frac{1}{n}\left\{\begin{array}{ll} [& T^{n-1}_{k-1}] \\ +[T^0_0 & +T^{n-2}_{k-2}] \\ +[(T^2_2 - M_2) & +T^{n-3}_{k-3}] \\ +\cdots+ & \\ +[(T^{k-3}_{k-3} - M_{k-3}) & +T^{n-(k-2)}_2] \\ +[(T^{k-2}_{k-2} - M_{k-2}) & +T^{n-(k-1)}_1] \\ +[(T^{k-1}_{k-1} - M_{k-1}) & +0] \\ +[(T^k_k - M_k) & +0] \\ +[(T^{k+1}_k - M_{k+1}) & +0] \\ +\cdots+ & \\ +[(T^{n-2}_k - M_{n-2}) & +0] \\ +[(T^{n-1}_k - M_{n-1}) & +0] \end{array}\right\}. \quad (9)$$

The first term, M_n, denotes the average number of time slots required for the maximum finding algorithm on n data elements [15].

The second term, $\frac{1}{n}$, represents the case that one of elements in M_n successfully broadcasts its value in Step 1 of $GSorting(S,k)$. Then, the first successful broadcast may be any one of the n elements randomly, each case having probability $\frac{1}{n}$.

The third term, $\left[T^{n-1}_{k-1}\right]$, denotes that the first successful broadcast is the largest data element in the progress of the maximum finding algorithm. Since we have the largest data element, we need to find the $k-1$ largest data elements among the remaining $n-1$ data elements, which is represented by T^{n-1}_{k-1}.

The fourth term, $\left[T^0_0 + T^{n-2}_{k-2}\right]$, means that the first successful broadcast is the second largest data element. Obviously, the second successful broadcast is the largest data element. T^0_0 represents a $NULL$ layer because it is empty between the largest and second largest data elements. And the subterm, T^{n-2}_{k-2}, represents that we need to find the $k-2$ largest data elements among the $n-2$ remaining data elements after the largest and second largest broadcasting.

The $(k+3)$th term, $\left[(T^{k-1}_{k-1} - M_{k-1}) + 0\right]$, represents that the first successful broadcast is the kth largest data element. Thus, $(T^{k-1}_{k-1} - M_{k-1})$ is the number of time slots required for finishing the sorting on the $k-1$ elements after the maximum is found. And since the first k, $k \ge 1$, largest elements can be found in the layers upper than the first successful broadcast element, we need not sort data elements in the lower layer, which stores the $n-k$ smaller elements. We omit the explanation of the rest of the equation, since it is quite similar to the above cases.

Lemma 4: Let $L_n = M_n - M_{n-1}$, then for $2 \le k \le n$

$$T^n_k - T^{n-1}_k = L_n + \frac{1}{n}L_{n-1} + \frac{1}{n-1}L_{n-2} + \cdots + \frac{1}{n-(k-2)}L_{n-(k-1)}.$$

Lemma 5: For $2 \le k \le n$,

$$T^k_k + 4\left[\ln(n-(k-2)) - \ln 2\right] \le T^n_k \le T^k_k + 5\ln(n-(k-1)).$$

Theorem 6: For $2 \le k \le n$,

$$\frac{10}{3}k + 4\ln(n-(k-2)) - \left(\frac{2}{3} + 4\ln 2\right) \le T^n_k \le \frac{11}{3}k + 5\ln(n-(k-1)) - \frac{4}{3}.$$

Martel and Moh [8] showed that the expected time for finding the first k largest numbers is $O(k + \log n)$. Comparing it with Theorem 6, our result has a tighter bound. In addition, we point out the relationship between the two special cases: maximum finding and sorting. When $k = 1$, $T^n_{k=1} = M_n$, it represents the case of maximum finding. When $k = n$, $T^n_{k=n} = T_n$, it represents the case of sorting, which has been proved in Theorem 3.

By Theorem 6, we prove that the average time complexity of the generalized sorting is $\Theta(k + \log(n-k))$. From the analysis, we can see the whole process of the generalized sorting. The process begins from maximum finding and ends at the completion of sorting.

Table 2 summarizes the results that include maximum finding, sorting and the process between them.

7. Simulations of Our Generalized Sorting Algorithm

To evaluate the accuracy of our theoretical analysis, we perform a simulation of our generalized sorting algorithm. The

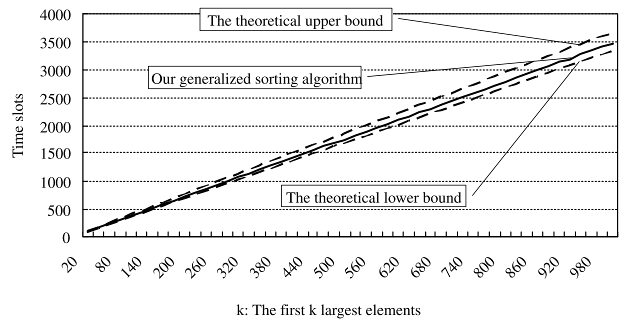

Fig. 2 A simulation for our generalized sorting algorithm with 1000 elements.

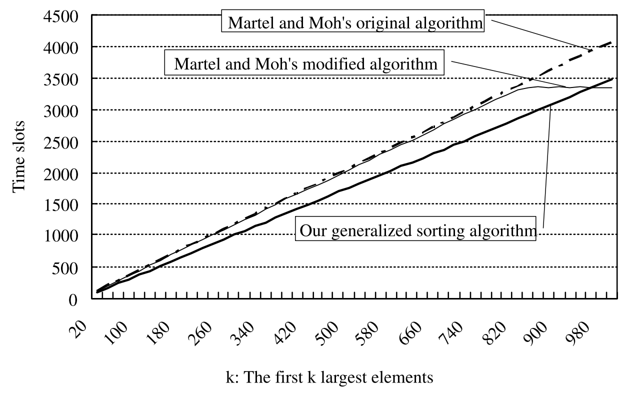

Fig. 3 A performance simulation for our generalized sorting algorithm and Martel and Moh's algorithm with 1000 elements.

simulation result is shown in Fig. 2. As expected, it shows that our algorithm fits the theoretical bound.

In Fig. 3, the performance simulation shows that our algorithm outperforms Martel and Moh's algorithm [8] from $k = 1$ to $k = n$. The termination condition of their original algorithm is to check whether the remaining set is empty. We can slightly modify their termination condition to check whether the remaining set is NULL or SINGLE. The simulation shows that their modified algorithm has a little bit better performance than our algorithm when $0.96n \leq k \leq n$. But our algorithm performs better at other values of k. The simulation shows when one needs sort only the first k largest elements, not all elements, our algorithm could be a good choice.

8. Conclusion

The layer concept [21] can help us to reduce the number of conflicts when an algorithm is not conflict-free under the WNCD model. In this paper, we apply the layer concept to solve the generalized sorting problem. We prove that the total average number of time slots, including conflict slots, empty slots and successful slots, is $\Theta(k + \log(n - k))$. When $k = 1$, it becomes maximum finding, and when $k = n$, it is a sorting process. Thus, in our analysis, we also built up the connection between the two special cases, maximum finding and sorting.

A good sorting algorithm will win in the end of the algorithm, but it may not always lead up in the whole process. To sort all elements may not be always necessary. Under the circumstances that we need to sort the first k largest elements, not all elements, our analysis can figure out the whole process of sorting.

Our analysis can be regarded as a base for the work which has the same high level approach. If there is another maximum finding algorithm under the WNCD model whose bound can be analyzed, not only it can be applied to our sorting and generalization approaches, but also the approaches can be analyzed immediately by the methodology of our analysis in this paper.

We are wondering whether there exists a maximum finding algorithm which can be applied into our sorting approach and always leads up in the whole process of sorting. This is one of our future works. And another future work is to find the native bound of the sorting approach under the WNCD model.

References

[1] R.S. Bhuvaneswaran, J.L. Bordim, J. Cui, N. Ishii, and K. Nakano, "An energy-efficient initialization protocol for wireless sensor networks," IEICE Trans. Fundamentals, vol.E85-A, no.2, pp.447–454, Feb. 2002.
[2] J.L. Bordim, J. Cui, N. Ishii, and K. Nakano, "Doubly-logarithmic energy-efficient initializaiton protocol for single-hop radio network," IEICE Trans. Fundamentals, vol.E85-A, no.5, pp.967–976, May 2002.
[3] R. Dechter and L. Kleinrock, "Broadcast communications and distributed algorithms," IEEE Trans. Comput., vol.35, no.3, pp.210–219, March 1986.
[4] J.H. Huang and L. Kleinrock, "Distributed selectsort sorting algorithm on broadcast communication," Parallel Comput., vol.16, pp.183–190, 1990.
[5] S. Levitan, "Algorithms for broadcast protocol multiprocessor," Proc. 3rd International Conference on Distributed Computing Systems, pp.666–671, 1982.
[6] S.P. Levitan and C.C. Foster, "Finding an extremum in a network," Proc. 1982 International Symposium on Computer Architechure, pp.321–325, 1982.
[7] C.U. Martel, "Maximum finding on a multi access broadcast network," Inf. Process. Lett., vol.52, pp.7–13, 1994.
[8] C.U. Martel and M. Moh, "Optimal prioritized conflict resolution on a multiple access channel," IEEE Trans. Comput., vol.40, no.10, pp.1102–1108, Oct. 1991.
[9] C.U. Martel, W.M. Moh, and T.S. Moh, "Dynamic prioritized conflict resolution on multiple access broadcast networks," IEEE Trans. Comput., vol.45, no.9, pp.1074–1079, 1996.
[10] A. Micic and I. Stojmenovic, "A hybrid randomized initialization protocol for tdma in single-hop wireless networks," Proc. International Parallel Distributed Processing Symposium (IPDPS'02), pp.147–154, 2002.
[11] K. Nakano and S. Olariu, "Randomized initialization protocols for ad-hoc networks," IEEE Trans. Parallel Distrib. Syst., vol.11, no.7, pp.749–759, July 2000.
[12] K. Nakano and S. Olariu, "Uniform leader election protocols in radio networks," IEEE Trans. Parallel Distrib. Syst., vol.13, no.5, pp.516–526, May 2002.
[13] K. Nakano, S. Olariu, and A. Zomaya, "Energy-efficient routing in the broadcast communication model," IEEE Trans. Parallel Distrib. Syst., vol.13, no.2, pp.1201–1210, Dec. 2002.

[14] K.V.S. Ramarao, "Distributed sorting on local area network," IEEE Trans. Comput., vol.C-37, no.2, pp.239–243, Feb. 1988.
[15] S.H. Shiau and C.B. Yang, "A fast maximum finding algorithm on broadcast communication," Inf. Process. Lett., vol.60, pp.81–96, 1996.
[16] S.H. Shiau and C.B. Yang, "The layer concept and conflicts on broadcast communication," J. Chang Jung Christian University, vol.2, no.1, pp.37–46, June 1998.
[17] S.H. Shiau and C.B. Yang, "A fast sorting algorithm and its generalization on broadcast communications," Proc. Sixth Annual International Computing and Combinatorics Conference (COCOON'2000), Bondi Beach, Sydney, Australia, pp.252–261, 2000.
[18] S.H. Shiau and C.B. Yang, "Generalization of sorting in single hop wireless networks," Technical Report, Department of Computer Science and Engineering, National Sun Yat-sen Univesity, March 2004, also see http://par.cse.nsysu.edu.tw/˜cbyang/person/publish/d04sortkwireless.pdf
[19] D.E. Willard, "Log-logarithmic protocols for resolving ethernet and semaphore conflicts," Proc. 16th Annual ACM Symposium on Theory of Computing, pp.512–521, 1984.
[20] D.E. Willard, "Log-logarithmic selection resolution protocols in a multiple access channel," SIAM J. Comput., vol.15, pp.468–477, 1986.
[21] C.B. Yang, "Reducing conflict resolution time for solving graph problems in broadcast communications," Inf. Process. Lett., vol.40, pp.295–302, 1991.
[22] C.B. Yang, "Computational geometry on the broadcast communication model," J. Inf. Sci. Eng., vol.15, pp.383–395, May 1999.
[23] C.B. Yang, R.C.T. Lee, and W.T. Chen, "Parallel graph algorithms based upon broadcast communications," IEEE Trans. Comput., vol.39, no.12, pp.1468–1472, Dec. 1990.
[24] C.B. Yang, R.C.T. Lee, and W.T. Chen, "Conflict-free sorting algorithm broadcast under single-channel and multi-channel broadcast communication models," Proc. International Conference on Computing and Information, pp.350–359, 1991.

Shyue-Horng Shiau received the BS degree from the Department of Engineering Science at National Cheng Kung University, Tainan, Taiwan, in 1984, and the MS degree from the Department of Applied Mathematics at National Sun Yat-sen University, Kaohsiung, Taiwan, in 1994. He is currently a PhD candidate in the Department of Computer Science and Engineering at National Sun Yat-sen University. He joined the faculty of the Department of Computer Aided Media Design, Chung Jung University, Tainan, Taiwan, as a lecturer in 1999. His research interests include algorithms and parallel processing.

Chang-Biau Yang received the BS degree in electronic engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1982, and the MS degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 1984. Then, he received the PhD degree in computer science from National Tsing Hua University in 1988. He is currently a professor in the Department of Computer Science and Engineering, National Sun Yat-sen University. His research interests include computer algorithms, interconnection networks, and bioinformatics.