

# A Fast Initialization Algorithm for Single-Hop Wireless Networks\*

Shyue-Horng Shiau                      Chang-Biau Yang<sup>†</sup>  
Department of Computer Science and Engineering  
National Sun Yat-sen University  
Kaohsiung, Taiwan 804, R.O.C.  
shiaush@cse.nsysu.edu.tw      cbyang@cse.nsysu.edu.tw

## ABSTRACT

Given a set of  $n$  stations, the *initialization* problem is to assign each station a unique identification number, from 1 to  $n$ . In the single-hop wireless Networks with collision detection, Nakano and Olariu proposed an algorithm to build a partition tree and solve the problem. In this paper, we shall classify the partition tree into four parts. By reviewing the classifications, we find that three ideas can improve the algorithm. We show that it needs  $2.88n$  time slots for solving the problem containing  $n$  stations. After applying our three ideas, the number of time slots will be improved to  $2.46n$ .

**Keywords:** parallel algorithm, initialization, broadcast communication, wireless network, conflict

## 1 INTRODUCTION

In recent years, due to the affluent development of wireless personal communication, the research in wireless networks (WN) [1, 2, 8, 11–13] has become more interesting and attractive. 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.



Figure 1: An example for 8 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 [9] classified the single-hop WN models by the capability with *collision detection* (CD). In the single-hop WN model with CD (WNCD), exactly one of three states on a radio channel can be decided by each station. The three states are as follows:

- NULL: No station wants to perform a transmission.
- SINGLE: Exactly one station wants to perform a transmission.
- COLLISION: Two or more stations want to perform a transmission simultaneously.

The WNCD model is one of the simplest parallel computation models and has been studied for more than two decades. Some researchers regarded the model as the broadcast communication model [3–5, 7, 12, 15–19, 21, 22, 25, 26]. In the WNCD model some important and essential components have been investigated such as sorting algorithms [3, 4, 6, 14, 19, 26], maximum finding algorithm [5, 15] and graph algorithms [22, 23, 25]. The *initialization* problem [8, 9] was discussed recently. The problem is to assign each of  $n$  stations a unique identification number, from 1 to  $n$ .

In the WNCD model, to solve a problem by 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.

Reducing conflict resolution time can be achieved by applying two concepts. The first concept is to dynamically estimate a proper broadcasting probability [5, 8]. The second concept to reduce conflict resolution time is the layer concept proposed by Yang [22]. 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 *initialization* problem is to retain competitions among remaining stations repeatedly. Each competition will produce a winner to win its unique identification number. The time required for this straightforward method is  $O(n \log n)$ . Nakano and Olariu [9] presented an *initialization* algorithm and improved the time complexity from  $O(n \log n)$  to  $O(n)$ . The method of their algorithm is to construct a strictly binary tree in which each internal node has exactly two children. The tree is called the *partition tree*. The method and the layer concept have a common strategy, which is divide-and-conquer. Obviously the strategy will bring benefit for solving problems.

In this paper, we shall reconstruct and analyze Nakano and Olariu's *initialization* algorithm based on the layer concept. We show that it needs  $2.88n$  time slots for solving the initialization problem containing  $n$  stations. It means that the partition tree contains  $2.88n$  nodes in average. We examine the nodes of the tree and find that some nodes may be reduced. Thus we separate the nodes of the tree into four classifications for more detailed examination.

\*This research work was partially supported by the National Science Council of the Republic of China under NSC-90-2213-E-110-030.

<sup>†</sup>To whom all correspondence should be sent.

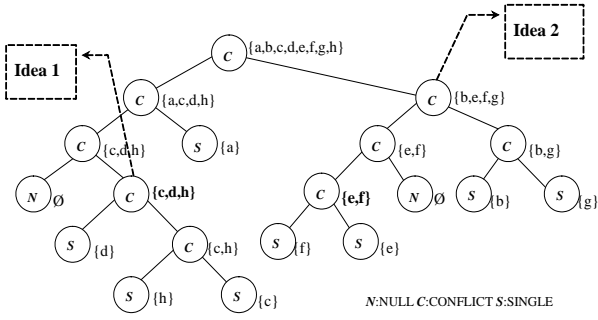


Figure 2: An example for a possible partition tree containing 8 stations.

After reviewing the classifications on the partition tree, we find that three ideas can improve the algorithm. Our first idea is to reduce redundant conflicts. The second idea is to reduce some conflicts that may be redundant with very high probability. After applying the two ideas, the partition tree will become unbalanced. Thus, the third idea is to adapt the optimal probability  $\frac{1}{2}$  to 0.418 and make the partition tree rebalanced. After applying the three ideas, we reduce the required time slots from  $2.88n$  to  $2.46n$ . In this paper, we will omit the proofs of the lemmas and theorems, which can be found in our technique report [20].

## 2 THE INITIALIZATION PROBLEM

Given a set of  $n$  stations, the initialization problem is to assign each station a unique identification number, from 1 to  $n$ . The problem is fundamental to network design and multiprocessor systems. In the WNCD model, the problem was solved by Nakano and Olariu [9]. Note that the channel in the model is single with CD and the number of  $n$  stations is *unknown*. In the next two sections, we shall represent Nakano and Olariu's algorithm in the view of the layer concept and analyze its average time complexity.

## 3 NAKANO AND OLARIU'S ALGORITHM WITH THE LAYER CONCEPT

The idea of Nakano and Olariu's algorithm [9] is to build a partition tree, which is a strictly binary tree. Each internal node represents two or more stations. Within each internal node, each of the associated stations decides to put itself into the left or right node by flipping a fair coin. An example of a possible partition tree containing 8 stations,  $\{a, b, c, d, e, f, g, h\}$ , is shown in Figure 2.

For two reasons, we translate Figure 2 into Figure 3 by our layer concept. The first reason is that after translation, to trace their algorithm and to understand our layer concept become easier. The second reason is to present the intuition where the algorithm can be improved efficiently. The key point of our layer concept is to use broadcast conflicts to build broadcasting layers and then to distribute the stations into those layers.

In Figure 3, initially, all stations can broadcast. Hence, in time slot 1, all of  $\{a, b, c, d, e, f, g, h\}$  broadcast and a CONFLICT occurs. Then layer 1 is built. Suppose that in time slot 2, only  $\{a, c, d, h\}$  decide to broadcast again. In fact, the decision is made randomly. Then a CONFLICT occurs and layer 2 is built. And  $\{a, c, d, h\}$  bring themselves up to layer 2. At the same time,  $\{b, e, f, g\}$  still stay in layer 1. In time slot 3, the stations on the active layer, which have the right to decide if they continue to broadcast or not, are  $a, c, d$  and  $h$ . Suppose in time slot 3, stations  $c, d$  and  $h$  want to broadcast. Then the channel state is CONFLICT and layer 3 is built. Since the rest progress

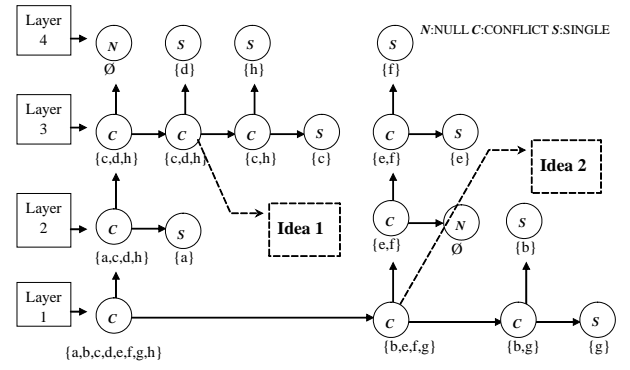


Figure 3: An example for the layer concept in the initialization problem containing 8 stations.

is a recursive way, the description of the further trace is omitted.

## 4 ANALYSIS OF NAKANO AND OLARIU'S ALGORITHM WITH THE LAYER CONCEPT

In this section, we shall analyze the average time complexity of Nakano and Olariu's algorithm [9] with a different method from their original analysis. The method of analysis has been used in our previous papers [15]. By the method, the analyzes of the two algorithms form a platform for comparing with each other on the average time complexity.

Suppose that there are  $k$  stations. 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 station 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:

$$T_2 = 1 + \frac{1}{4} (T_2 + T_0) + \frac{1}{4} (T_1 + T_1) + \frac{1}{4} (T_1 + T_1) + \frac{1}{4} (T_0 + T_2). \quad (1)$$

Generalizing Eq. (1) and rearranging it, for  $k \geq 3$ , we obtain

$$(1 - \frac{1}{2^{k-1}})T_k = 1 + \frac{1}{2^{k-1}} \left[ 1 + C_{k-1}^k T_{k-1} + C_{k-2}^k T_{k-2} + \dots + C_2^k T_2 + C_1^k T_1 \right].$$

### Lemma 1

$$2.8 \leq T_n - T_{n-1} \leq 3, \text{ for } n \geq 3.$$

### Theorem 2

$$2.8n - 0.6 < T_n < 3n - 1, \text{ for } n \geq 3.$$

In Section 8, we perform some simulations as shown in Figure 6. It shows that Theorem 2 can be expressed as follows:

$$\frac{T_n}{n} \cong 2.88. \quad (2)$$

## 5 CLASSIFYING THE NODES IN THE PARTITION TREE

In the previous two sections, We presented Nakano and Olariu's algorithm [9] and analyzed its average time complexity. Their algorithm is based on the partition tree. For learning more information about the partition tree, in this

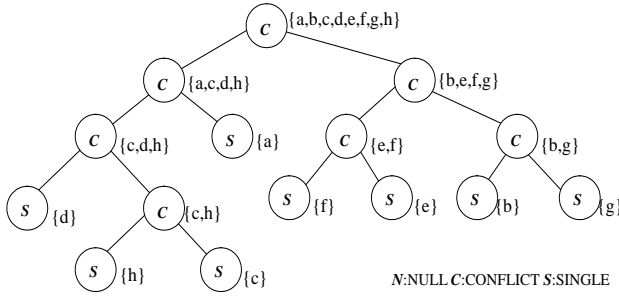


Figure 4: An example for a pure partition tree containing 8 stations.

section, we shall present an analysis of the partition tree. By the analysis, not only we can learn about the classification of the nodes in the partition tree, but also we know that the classification information provides an important basement. By learning the tree, we can find out which node can not be or may be reduced.

Figure 2 shows an example of a partition tree containing 8 stations. It is easy to find that each of the NULL nodes accompanies with exactly one CONFLICT node at either left or right side. By cutting off the NULL nodes and their companion CONFLICT nodes in Figure 2, we transform the partition tree into the pure partition tree shown in Figure 4.

In Figure 4, the pure partition tree contains 8 SINGLE nodes, each of which corresponds to one station, and it contains 7 CONFLICT nodes. In general, if a pure partition tree contains  $n$  stations, then it contains  $n$  SINGLE nodes (leaf nodes) and  $n - 1$  CONFLICT nodes (internal nodes) [2, 12]. In other words, the pure partition tree contains  $2n - 1$  nodes.

The gap between the number nodes of pure partition and partition tree,  $2n$  and  $2.88n$ , is  $0.88n$ . It means that the average number of NULL nodes and their companion CONFLICT nodes is  $0.88n$ . In other words, the average number of NULL nodes is  $0.88n/2 = 0.44n$ . The partition tree with probability  $\frac{1}{2}$  implies that the tree is symmetric, thus we can predict that the average number of left or right NULL nodes is  $0.44n/2 = 0.22n$ .

This prediction can be verified by the similar analysis used in the previous section. Thus we only show the generalized equation as follows. Let  $U_k$  denote the average number of left NULL nodes.

$$\begin{aligned} & (1 - \frac{1}{2^{k-1}})U_k \\ &= \frac{1}{2^k}[U_0 + C_{k-1}^k(U_{k-1} + U_1) + C_{k-2}^k(U_{k-2} + U_2) \\ &+ \dots + C_1^k(U_1 + U_{k-1})], \text{ for } k \geq 2. \end{aligned}$$

**Lemma 3**

$$0.2 \leq U_n - U_{n-1} \leq 0.24, \text{ for } n \geq 3.$$

**Theorem 4**

$$0.2n - 0.6 < U_n < 0.24n - 1, \text{ for } n \geq 3.$$

By the simulation presented Section 8, Theorem 4 can be expressed as follows:

$$\frac{U_n}{n} \cong 0.22. \quad (3)$$

It means that if a partition tree contains  $n$  stations, then in average, it will have approximately  $0.22n$  left NULL nodes. We can also conclude that there are approximately

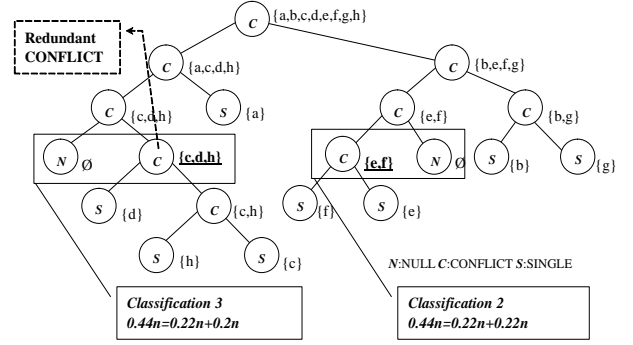


Figure 5: An example for the four classifications of a partition tree containing 8 stations.

$0.22n$  right NULL nodes. The result equal to the gap of the first and second upper line in Figure 6.

By the above analysis of a partition tree with  $n$  stations, we can separate the nodes into four classifications as follows. (see Figure 5.)

- Classification 1: The total number of SINGLE nodes is  $n$ . Obviously, it can not be reduced.
- Classification 2: The right NULL nodes and their companion CONFLICT nodes are grouped into a classification. For example, in Figure 5, the right NULL node,  $\emptyset$ , and its companion CONFLICT nodes  $\{e, f\}$  belong to this classification. The average number of right NULL nodes included in this classification is  $0.22n$ . This classification is hard to be reduced.
- Classification 3: The left NULL nodes and their companion CONFLICT nodes are grouped into a classification. For example, in Figure 5, the left NULL node,  $\emptyset$ , and its companion CONFLICT node pointed by "Redundant CONFLICT",  $\{c, d, h\}$ , belong to this classification. The average number of the left NULL nodes included in this classification is  $0.22n$ . This classification can be reduced. The more detailed explanation will be given in the Section 8.
- Classification 4: Excluding the above three classifications from the partition tree, the remaining nodes are CONFLICT nodes, which belong to this classification. Note that the number of the remaining CONFLICT nodes is  $n - 1$ . By combining classifications 1 and 4, we can build a pure partition as shown in Figure 4.

## 6 OUR CRBP ALGORITHM BASED ON CONFLICT REDUCTION

After carefully reviewing the four classifications of the partition tree in previous section, we find that the nodes of the tree can be reduced by three ideas. In this section, we shall present a fast algorithm under the first two ideas.

The first idea can be illustrated by the first node containing three stations,  $c$ ,  $d$  and  $h$ , in Figure 3. By Nakano and Olariu's algorithm, each of the three stations transmits on the channel and a broadcast conflict occurs in time slot 3. Each of the three stations in this layer flips a fair coin. If the three stations flip "tail", the state will be NULL in time slot 4. And all the three stations stay in the current layer. After that, each of the three stations trasmits in time slot 5, and then there will be a conflict occurs. After the conflict occurs, in time slot 6, only station  $d$  flips "head" to move up the upper layer and the other stations stay current layer.

By examining the process, we find that the CONFLICT node in time slot 5 is redundant. The CONFLICT node is pointed by "Redundant CONFLICT" in Figure 3. Because the conflict occurs in time slot 3, we can confirm that the current layer contains more than one station. If the three stations decide to stay in the current layer by flipping a coin, the next state will be NULL in time slot 4. After that, it is not necessary for the three stations to confirm that the current layer contains more than one station again by a conflict in time slot 5, since it has been confirmed by a conflict in time slot 3. Thus the CONFLICT node is indeed redundant.

The second idea is to reduce some broadcast conflicts which are *probably* redundant. The probably redundant conflict can be verified by a probability thought. Assume the stations in the current layer have received eight SINGLE from the stations in the upper layers. Then each of the stations in the current layer can guess that there will be more than one station in the layer. If these stations ignore the information and try to broadcast. A broadcast conflict will occur with a very high probability. If these stations in the same layer learn the information and immediately decide to go up the upper layer or stay in the current layer, it will save a conflict to confirm the layer contain more than one station. With a very low probability, it will get a penalty since the guess may be wrong (The layer really contains one or no station.). We shall get some benefit on average.

An example was shown in Figure. 3. In layer 1, stations b, e, f and g receive four SINGLE in time slots 6, 8, 9 and 10. They learn that the upper layers contain four stations. Each of stations in layer 1 guesses that there are more than one station in same layer. And immediately they decide to go up the upper layer or stay in the current layer. It will save a CONFLICT node in time slot 11. The CONFLICT node is pointed by "Idea 2" in Figure. 3.

Our simulation shows that if the upper layers contain more than four stations, we can get best benefit that the stations in the current layer decide to up the upper layer or stay in the current layer.

Combining the two ideas, we modify Nakano and Olariu's algorithm [9] into a fast algorithm named as CRBP and show it as follows.

#### Algorithm 1

```

ID-Initializing: CRBP(C, n)
Each station in C broadcasts its message.
// There are three possible cases:
If a broadcast conflict occurs then
  each of stations sets  $n' = n$ 
  // Based first idea, without conflict to confirm the
  // layer again, each of stations in C flips a coin
  // until the upper layer is not empty.
  do while  $n' = n$ 
    each of stations in C flips a coin.
    All which get heads form U and bring
    themselves to the upper layer.
     $n' = CRBP(U, n)$ .
  enddo
   $S = C - C'$  (Now  $C - C'$  is formed by which get
  tails and stay in the current layer.)
  If  $n' - n < 4$  then
     $m = CRBP(S, n')$ .
  else
    // Based on second idea, the upper layers
    // contain more than four stations.
     $m = GUESS(S, n')$ .
  endif
  Return m.
endif
If one station successfully broadcast its message, then
  the unique station sets  $ID \leftarrow n$  and

```

```

  leaves the algorithm.
  all stations set  $n \leftarrow n + 1$ .
  Return n.

```

```

endif
if no station broadcasts, then
  n is unchanged by each station. (C is empty.).
  Return n.
endif
end

```

#### Algorithm 2

```

ID-Initializing: GUESS(C, n)
Each of stations in C flips a coin.
All which get heads form U and bring themselves to the
upper layer.
 $n' = CRBP(U, n)$ .
 $S = C - C'$  (Now  $C - C'$  is formed by which get tails
and stay in the current layer.)
If  $n' - n < 4$  then
   $m = CRBP(S, n')$ .
else
  // Based on second idea, the upper layers contain
  // more than four stations.
   $m = GUESS(S, n')$ .
endif
Return m.
end

```

## 7 ANALYSIS OF OUR CRBP ALGORITHM

Suppose that there are  $k$  stations. Let  $F_k$  denote the average number of time slots, including conflict slots, empty slots and slots for successful broadcasts, required when the algorithm is executed.  $F_k$ , for  $k \geq 3$ , can be formulated as follows.

$$\begin{aligned}
F_n = 1 + \frac{1}{2^n} [ & C_n^n (F_n + (F_0 + F_0)) + C_{n-1}^n (F_{n-1} + (F_1 + F_0)) + \\
& C_{n-2}^n (F_{n-2} + (F_2 - 1 + \frac{1}{2^2})) + \\
& C_{n-3}^n (F_{n-3} + (F_3 - 1 + \frac{1}{2^3})) + \dots + \\
& C_5^n (F_5 + (F_{n-5} - 1 + \frac{1}{2^{n-5}})) + \\
& C_4^n (F_4 + (F_{n-4} - 1 + \frac{1}{2^{n-4}})) + \\
& C_3^n (F_3 + (F_{n-3} - 1 + \frac{1}{2^{n-3}})) + C_2^n (F_2 + F_{n-2}) + \\
& C_1^n (F_1 + F_{n-1}) + C_0^n (F_0 + F_{n-1}) ]
\end{aligned}$$

#### Lemma 5

$$2.4 \leq F_n - F_{n-1} \leq 2.6, \text{ for } n \geq 6.$$

#### Theorem 6

$$2.4n + \frac{143}{210} < F_n < 2.6n - \frac{67}{210}, \text{ for } n \geq 6.$$

By our simulation in Figure 6, Theorem 6 can be expressed as follows:

$$\frac{F_n}{n} \cong 2.5. \quad (4)$$

It means that if a partition tree contained  $n$  stations, then in average it will have approximate  $2.5n$  nodes, including CONFLICT, NULL and SINGLE nodes. In other words, it needs  $2.5n$  time slots for solving the initialization problem containing  $n$  stations. By comparing Eq. (4) with Eq. (2), we can see that an improvement can be obtained by our fast algorithm.

Intuitively the partition tree is optimal under a balanced probability,  $p = \frac{1}{2}$ . However, after applying our first idea, the partition tree will become unbalanced. Thus the third idea is to adjust the optimal probability  $\frac{1}{2}$  to 0.418 and make the partition tree rebalanced. Since the proof of the intuition and rebalance is tedious, we shall omit it. We only show a table for  $2 \leq n \leq 9$  as follows. In the table,  $n$  is the

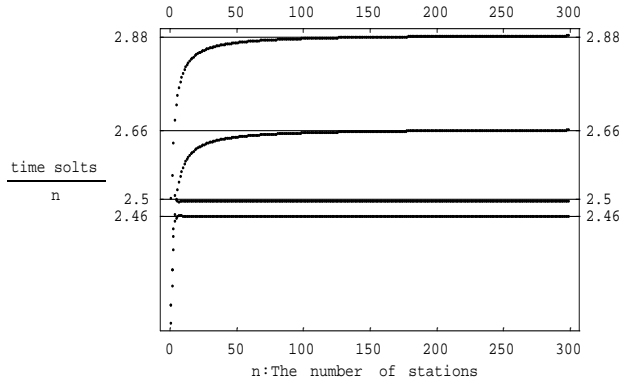


Figure 6: A simulation for CRBP algorithms with three improvement ideas.

number of stations and  $p$  is optimal probability for each  $n$  individually.

2	3	4	5	6	7	8	9
0.414	0.427	0.420	0.417	0.417	0.417	0.417	0.418

The average probability in the table is 0.418. We take it as the bias probability in our CRBP algorithm.

## 8 SIMULATIONS OF OUR CRBP ALGORITHMS

In Figure 6, we show the simulation of our first two ideas two algorithms. Our CRBP algorithm with  $p = 0.418$  is exhibited at the lowest line. Nakano and Olariu's algorithm with  $p = \frac{1}{2}$  is displayed at the highest line. The remaining two lines illustrate our first and second ideas. The simulation is executed with two tools, the Mathematica and C language. And the two results are same.

In Figure 6, our CRBP algorithm with  $p = 0.418$  is faster than Nakano and Olariu's algorithm [9]. The Figure shows that if a partition tree contains  $n$  stations, then in average the tree includes  $2.88n$  nodes. With our CRBP algorithm, the result will be improved to  $2.46n$  nodes. In this paper, we assume that the number of stations is unknown under WNCD. Thus we don't know what the optimal probability is. We take  $p = 0.418$  as a basement for our CRBP algorithm. If we take  $p$  near 0.418, the performance will have only a very tiny differentiation.

## 9 CONCLUSION

The layer concept and the partition tree method can improve conflict resolution. By applying them, many important essential problems under WNCD can be solved efficiently. Maximum finding[24] is one of those problems. We use the layer concept to obtain a fast algorithm for solving the problem. Nakano and Olariu [9] applied the partition tree method to solve the initialization problem. In this paper, we apply the layer concept to analyze the partition tree. The tree includes  $2.88n$  nodes in average. After classifying the nodes of the tree into four classifications, we find that two of them can be reduced. And by our third idea on the bias probability, our CRBP algorithm can improve the result from  $2.88n$  to  $2.46n$ .

Another way to reduce conflict resolution is to dynamically estimate a proper broadcasting probability. By using the dynamic probability concept, Martel [5] proposed an efficient maximum finding algorithm, which improve 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.

Micic and Stojmenovic [8] efficiently solved the *initialization* problem by using the same concept. Their algorithm is a hybrid randomized initialization protocol which combines two algorithms. The first algorithm is to reduce the nodes in Classification 2 of the partition tree. The second one is to reduce the nodes in Classifications 3 and 4 of the partition tree. Our classification not only implies that their algorithm may be more efficient than our CRBP, but also predicts the bound of their algorithm. The implication and the prediction need more examination. And our classification implies that the *initialization* problem should have a native bound. It also needs more detailed verification.

Nakano and Olariu [10] showed a history view on the leader selection problem. Nakano, Olariu and Zomaya [10] leaded a breadth-first view into the energy-efficient routing problem. By these concepts, our future work is to find some mining informations in the partition tree and make more improvement.

## 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 Transactions on Fundamentals*, No. 2, pp. 447–454, Feb. 2002.
- [2] J. Carle and J. F. Myoupo, "Collision detection based-deterministic protocol for dynamic initialization of radio networks," *Proc. ISCA 13th Int. Conf. on Parallel and Distributed Computing Systems (PDCS-2000)*, Las Vegas, USA, pp. 159–164, 2000.
- [3] J. H. Huang and L. Kleinrock, "Distributed selectsort sorting algorithm on broadcast communication," *Parallel Computing*, Vol. 16, pp. 183–190, 1990.
- [4] S. Levitan, "Algorithms for broadcast protocol multiprocessor," *Proc. of 3rd International Conference on Distributed Computing Systems*, pp. 666–671, 1982.
- [5] C. U. Martel, "Maximum finding on a multi access broadcast network," *Information Processing Letters*, Vol. 52, pp. 7–13, 1994.
- [6] C. U. Martel and M. Moh, "Optimal prioritized conflict resolution on a multiple access channel," *IEEE Transactions on Computers*, Vol. 40, No. 10, pp. 1102–1108, Oct. 1991.
- [7] C. U. Martel, W. M. Moh, and T. S. Moh, "Dynamic prioritized conflict resolution on multiple access broadcast networks," *IEEE Transactions on Computers*, Vol. 45, No. 9, pp. 1074–1079, 1996.
- [8] A. Micic and I. Stojmenovic, "A hybrid randomized initialization protocol for tdma in single-hop wireless networks," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS.02)*, pp. 147–154, 2002.
- [9] K. Nakano and S. Olariu, "Randomized initialization protocols for ad-hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 7, pp. 749–759, July 2000.
- [10] K. Nakano and S. Olariu, "A survey on leader election protocols for radio networks," *Proc. International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pp. 71–76, 2002.

- [11] K. Nakano and S. Olariu, "Uniform leader election protocols in radio networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 5, pp. 516–526, May 2002.
- [12] K. Nakano, S. Olariu, and A. Zomaya, "Energy-efficient routing in the broadcast communication model," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 2, pp. 1201–1210, Dec. 2002.
- [13] K. Nakano, S. Olariu, and A. Y. Zomaya, "Energy-efficient permutation routing protocols in radio networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 6, pp. 544–557, June 2001.
- [14] K. V. S. Ramarao, "Distributed sorting on local area network," *IEEE Transactions on Computers*, 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," *Information Processing Letters*, Vol. 60, pp. 81–96, 1996.
- [16] S. H. Shiau and C. B. Yang, "The layer concept and conflicts on broadcast communication," *Journal of 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 on broadcast communications," *Proc. of Second International Conference on Parallel Computing Systems (PCS99)*, Ensenada, Baja California, Mexico, pp. 87–92, 1999.
- [18] S. H. Shiau and C. B. Yang, "A fast sorting algorithm and its generalization on broadcast communications," *Proc. of Sixth Annual International Computing and Combinatorics Conference (COCOON'2000)*, Bondi Beach, Sydney, Australia, 2000.
- [19] S. H. Shiau and C. B. Yang, "Sorting algorithms for broadcast communications: Algorithms and fundamental analysis," Technical Report, Department of Computer Science and Engineering, National Sun Yat-sen University, April 2000, also see <http://par.cse.nsysu.edu.tw/~cbyang/person/publish/d00sortubcm.ps>.
- [20] S. H. Shiau and C. B. Yang, "A fast initialization algorithm for single-hop wireless networks," Technical Report, Department of Computer Science and Engineering, National Sun Yat-sen University, March 2004, also see <http://par.cse.nsysu.edu.tw/~cbyang/person/publish/d04initubcm.ps>.
- [21] D. E. Willard, "Log-logarithmic protocols for resolving ethernet and semaphore conflicts," *Proc. of 16th Annual ACM Symposium on Theory of Computing*, pp. 512–521, 1984.
- [22] C. B. Yang, "Reducing conflict resolution time for solving graph problems in broadcast communications," *Information Processing Letters*, Vol. 40, pp. 295–302, 1991.
- [23] C. B. Yang, "Computational geometry on the broadcast communication model," *Journal of Information Science and Engineering*, Vol. 15, pp. 383–395, May 1999.
- [24] C. B. Yang, R. C. T. Lee, and W. T. Chen, "Finding minimum spanning trees based upon single-channel broadcast communications," *Proc. of International Computer Symposium 1988, Taipei, Taiwan*, pp. 1451–1456, 1988.
- [25] C. B. Yang, R. C. T. Lee, and W. T. Chen, "Parallel graph algorithms based upon broadcast communications," *IEEE Transactions on Computers*, Vol. 39, No. 12, pp. 1468–1472, Dec. 1990.
- [26] 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. of International Conference on Computing and Information*, pp. 350–359, 1991.