PAPER

A Fast Initialization Algorithm for Single-Hop Wireless Networks*

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

SUMMARY 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 classification, 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 $2.46n$.

key words: *parallel algorithm, initialization, broadcast, communication, wireless network, conflict*

1. Introduction

In recent years, due to the rapid adoption of wireless personal communication, the research into wireless networks (WN) [1]–[5] 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 receive 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 [6] 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 statuses on a radio channel can be decided by each station. The three statuses are as follows:

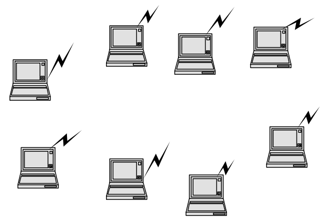

Fig. 1 An example for 8 stations.

- 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 [5], [7]–[17]. In the WNCD model some important and essential components have been investigated such as sorting algorithms [7], [15], [17]–[19], maximum finding algorithm [12], [13] and graph algorithms [9], [10], [20]. The *initialization* problem [6] was discussed recently. The problem is to assign each of n stations a unique identification number, from 1 to n.

Reducing conflict resolution time can be achieved by applying two concepts. The first concept is to dynamically estimate a proper broadcasting probability [12]. The other concept is the layer concept proposed by Yang [10]. To apply the layer concept for finding the maximum among a set of n numbers [13], we can improve the time complexity from $\Theta(\log^2 n)$ to $\Theta(\log n)$ [14]. 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 [6] presented an *initialization* algorithm and improved the time complexity from $O(n \log n)$ to $O(n)$. 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 algorithm and the layer concept share a common strategy, which is divide-and-conquer. Obviously the strat-

Manuscript received January 4, 2005.
Manuscript revised May 4, 2005.

†The author is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 804 Taiwan. He is also with the Department of Computer Aided Media Design, Chung Jung University, Tainan, Taiwan.
††The author is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 804 Taiwan.

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

a) E-mail: shiaush@mail.cju.edu.tw
b) E-mail: cbyang@cse.nsysu.edu.tw

DOI: 10.1093/ietcom/e88–b.11.4285

egy 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 in average 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 removed. Thus we divide the nodes of the tree into four classes for more detailed examination.

After reviewing the classifications on the partition tree, we propose three ideas to 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 adjust 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 average. In this paper, we will omit the proofs of the lemmas and theorems, which can be found in our technique report [21].

This paper is organized as follows. Section 2 includes two subsections. In Sect. 2.1, we shall present Nakano and Olariu's algorithm based on the partition tree [6] and reconstruct the algorithm with the layer concept. In Sect. 2.2 we shall analyze the reconstructed algorithm. In Sect. 3, the partition tree will be classificated into four parts. In Sects. 4 and 5, we shall present and analyze our initialization algorithm based on our three ideas. The simulation will be given in Sect. 6. Finally, Sect. 7 concludes the paper.

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 [6]. Note that the channel in the model is single with CD and the number of stations involved in the model is assumed to be *unknown*. In the next two subsections, we shall represent Nakano and Olariu's algorithm in the view of the layer concept and analyze its average time complexity.

2.1 Nakano and Olariu's Algorithm with the Layer Concept

The idea of Nakano and Olariu's algorithm [6] 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 Fig. 2.

For two reasons, we translate Fig. 2 into Fig. 3 by our layer concept. One reason is that after translation, to trace their algorithm and to understand our layer concept become easier. The other reason is to present the intuition where

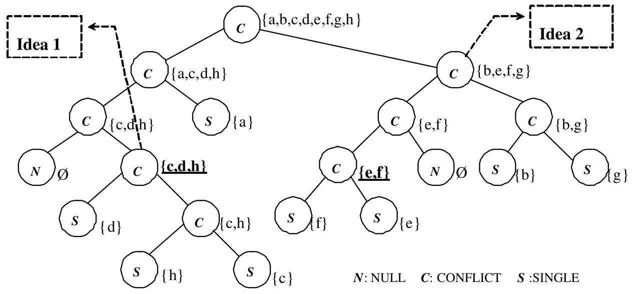

Fig. 2 An example for a possible partition tree containing 8 stations.

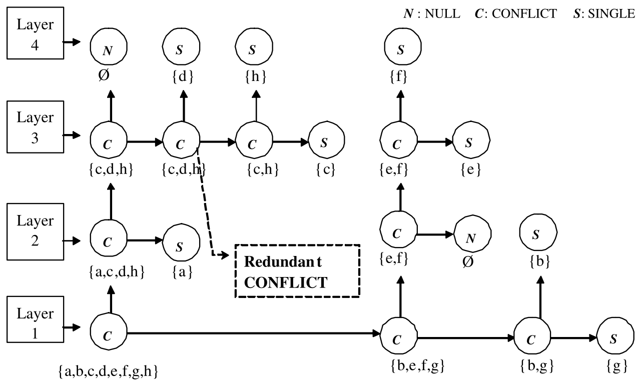

Fig. 3 An example for the layer concept in the initialization problem containing 8 stations.

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 Fig. 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. Since the rest progress is a recursive way, the description of the further trace is omitted.

Before ending this subsection, we shall translate their algorithm into the view of layer concept as follows. Their algorithm can represented as a recursive function: *Initial*(C, n), where C is a set of stations. Initially, Each station sets an initial counting value $n = 1$.

2.2 Analysis of Nakano and Olariu's Algorithm with the Layer Concept

In this subsection, we shall analyze the average time complexity of Nakano and Olariu's algorithm [6] with a different method from their original analysis. The analysis technique is similar to that used in our previous paper [13].

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 1 ID_Initializing_with_Layer_Concept: *Initial* (C, n)

```
Each station in $C$ broadcasts its message.
{There are three possible cases:}
if a broadcast conflict occurs then
    each station in $C$ flips a coin. All which get heads form $C'$ and bring themselves to the upper layer.
    $n' = Initial(C', n)$.
    $m = Initial(C - C', n')$. {Now $C - C'$ is formed by which get tails.}
    Return $m$.
end if
if exactly one station successfully broadcasts its message then
    the unique station sets $ID \leftarrow n$ and leaves the algorithm.
    And all stations set $n \leftarrow n + 1$.
    Return $n$.
end if
if no station broadcasts then
    $n$ is unchanged by each station. {$C$ is empty.}
    Return $n$.
end if
```

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:

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

We shall only explain the first and second terms in Eq. (1). The first term, 1, is a conflict in the first time slot. Then, two stations flipping coins can be divided into 4 cases, each having probability $\frac{1}{4}$. Here, it is assumed that the probability of each station getting a head is $\frac{1}{2}$. The second term, $(T_2 + T_0)$, is caused by that both stations get heads and bring themselves to the upper layer. The number of time slots required for the upper layer can be represented as recursive T_2. The empty current layer needs T_0 time slots. The explanation of the remaining terms is skipped. Substituting $T_0 = 1$ and $T_1 = 1$ into Eq. (1) and solving it, we get $T_2 = 5$. When $k = 3$, there are 2^3 cases. Therefore, extending T_2 to T_3, we obtain T_3 as follows:

$$T_3 = 1 + \frac{1}{2^3}[C_3^3(T_3 + T_0) + C_2^3(T_2 + T_1) + C_1^3(T_1 + T_2) + C_0^3(T_0 + T_3)]. \quad (2)$$

Substituting $T_0 = 1$, $T_1 = 1$ and $T_2 = 5$ into Eq. (2) and solving it, we have $T_3 = \frac{23}{3}$. Generalizing Eq. (2) and rearranging it, for $k \geq 3$, we obtain

$$\left(1 - \frac{1}{2^{k-1}}\right)T_k = 1 + \frac{1}{2^{k-1}}[1 + C_{k-1}^k T_{k-1} + C_{k-2}^k T_{k-2} + \cdots + C_1^k T_1]. \quad (3)$$

Lemma 1:

$$\left(1 - \frac{1}{2^{n-1}}\right)T_n - \left(1 - \frac{1}{2^{n-2}}\right)T_{n-1} = \frac{1}{2^{n-1}}[C_{n-2}^{n-1}(T_{n-1} - T_{n-2}) + C_{n-3}^{n-1}(T_{n-2} - T_{n-3}) + \cdots + C_1^{n-1}(T_2 - T_1) + T_{n-1}], \quad \text{for} \quad n \geq 3.$$

Lemma 2:

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

Theorem 3:

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

In Sect. 6, we perform some simulations as shown in Fig. 6. It shows that Theorem 3 can be expressed as follows:

$$\frac{T_n}{n} \cong 2.88. \quad (4)$$

It means that if a partition tree containing n stations, then in average it will have approximately $2.88n$ nodes, including CONFLICT nodes, NULL nodes and SINGLE nodes. In other words, it needs about $2.88n$ time slots for solving the *initialization* problem with n stations.

3. The Classification of the Nodes in the Partition Tree

For learning more information about the partition tree of Nakano and Olariu's algorithm, in this section, we shall present an analysis of the partition tree. With 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 removed.

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 Fig. 2, we transform the partition tree into the pure partition tree shown in Fig. 4.

In Fig. 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) [1], [5]. Thus the pure partition tree contains $2n - 1$ nodes.

The gap between the number nodes of pure partition and partition tree, $2n-1$ and $2.88n$, is about $0.88n$. It means that the average number of NULL nodes and their companion CONFLICT nodes is $0.88n$. In other words, the average

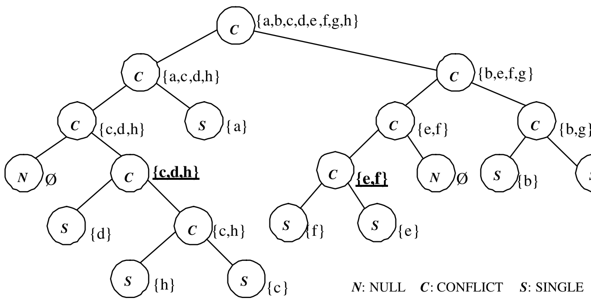

Fig. 4 An example for a pure partition tree containing 8 stations.

number of NULL nodes is $\frac{0.88n}{2} = 0.44n$. 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 $\frac{0.44n}{2} = 0.22n$. This prediction can be verified by the similar analysis used in Sect. 2.2. Thus we only show the result as follows. Let U_k denote the average number of left NULL nodes.

Theorem 4:

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

By the simulation presented in Sect. 6, Theorem 4 can be expressed as follows:

$$\frac{U_n}{n} \cong 0.22. \tag{5}$$

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

By the above analysis of a partition tree with n stations, we can divide the nodes into four classes as follows. (See Fig. 5.)

- Class A contains all SINGLE nodes, whose size is n. Obviously, this class can not be removed.
- Class B contains the right NULL nodes and their companion CONFLICT nodes. For example, in Fig. 5, the right NULL node, ∅, and its companion CONFLICT nodes $\{\mathbf{e}, \mathbf{f}\}$ belong to this class. The average number of right NULL nodes included in this class is $0.22n$. This class is difficult to be reduced.
- Class C contains the left NULL nodes and their companion CONFLICT nodes are grouped into this classification. For example, in Fig. 5, the left NULL node, ∅, and its companion CONFLICT node pointed by "**Redundant CONFLICT**," $\{\mathbf{c}, \mathbf{d}, \mathbf{h}\}$, belong to this class. The average number of the left NULL nodes included in this class is $0.22n$. This class may be reduced. The more detailed explanation will be given in Sect. 4.

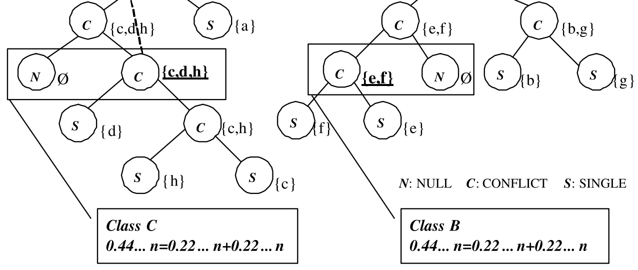

Fig. 5 An example for the four classes of a partition tree containing 8 stations.

- Class D Excluding the above three classes, the remaining nodes are CONFLICT nodes, and they belong to this class, whose size is $n - 1$. By removing classes B and C, we can build a pure partition tree as shown in Fig. 4.

4. Our CRBP Algorithm Based on Conflict Reduction

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

The first idea can be illustrated by the first node containing three stations, c, d and h, in Fig. 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 all the three stations flip "tail," the status will be NULL in time slot 4. And all the three stations stay in the current layer. After that, a conflict occurs when each of the three stations transmits in time slot 5. After the conflict occurs, in time slot 6, only station d flips "head" to move up to the upper layer and the other two 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 Fig. 5. Because the conflict occurs in time slot 3, we are sure 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 status 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 SINGLEs from the stations in the upper layers. Then each of the stations in the current layer can guess that there should be more than one station in the layer. If these stations ignore the information and try to broadcast, a broadcast conflict may 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 that the layer contains 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). Thus, we can get some benefit on average.

See Fig. 3. In layer 1, stations b, e, f and g receive four SINGLEs in time slots 6, 8, 9 and 10. They learn that the upper layers contain four stations. Each station in layer 1 guesses that there should be more than one station in same layer. And by a random choice, immediately each station decides to go up the upper layer or to stay in the current layer. It will save a CONFLICT node in time slot 11. The

Algorithm 2 ID-Initializing: *CRBP*(C, n)

```
Each station in $C$ broadcasts its message.
{There are three possible cases:}
if a broadcast conflict occurs then
  each station sets $n' = n$
  {Based on our first idea, without conflict to confirm the layer again,
  each station in $C$ flips a coin until the upper layer is not empty.}
  while $n' = n$ do
    each station in $C$ flips a coin.
    All which get heads form $C'$ and bring themselves to the upper layer.
    $n' = CRBP(C', n)$.
  end while
  $S = C - C'$. {Now $C - C'$ is formed by all which get tails and stay in
  the current layer.}
  if $n' - n < 4$ then
    $m = CRBP(S, n')$.
  else
    {Based on our second idea, the upper layers contain more than four
    stations.}
    $m = GUESS(S, n')$.
  end if
  Return $m$.
end if
if exactly one station successfully broadcasts its message, then
  the unique station sets $ID \leftarrow n$ and leaves the algorithm.
  all stations set $n \leftarrow n + 1$.
  Return $n$.
end if
if no station broadcasts, then
  $n$ is unchanged by each station. {$C$ is empty.}
  Return $n$.
end if
```

Algorithm 3 ID-Initializing: $GUESS(C, n)$

```
Each station in $C$ flips a coin.
All which get heads form $C'$ and bring themselves to the upper layer.
$n' = CRBP(C', n)$.
$S = C - C'$. {Now $C - C'$ is formed by all which get tails and stay in the
current layer.}
if $n' - n < 4$ then
  $m = CRBP(S, n')$.
else
  {Based on our second idea, the upper layers contain more than four
  stations.}
  $m = GUESS(S, n')$.
end if
Return $m$.
```

CONFLICT node is pointed by "**Idea 2**" in Fig. 5.

Our simulation shows that if the upper layers contain more than four stations, we can get the best benefit on guessing that there are more than one station in the current layer.

Combining the above two ideas and a bias probability, $p = 0.418$, we modify Nakano and Olariu's algorithm [6] to become a faster algorithm named as conflict reduction and bias probability algorithm (CRBP for short) as follows.

5. 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. When there is zero or one station for the algorithm, one empty slot or one slot for successful broadcast is needed. Thus $F_0 = 1$ and $F_1 = 1$.

F_2 and F_3 can be obtained by the following equations:

$$F_2 = 1 + \frac{1}{4}[(F_2 + F_0) + (F_1 + F_1) + (F_1 + F_1) + (F_0 + F_2 - 1)] \tag{6}$$

$$F_3 = 1 + \frac{1}{2^3}[C^3_3(F_3 + F_0) + C^3_2(F_2 + F_1) + C^3_1(F_1 + F_2) + C^3_0(F_0 + F_3 - 1)]. \tag{7}$$

Since the terms of Eq. (6) and Eq. (7) are quite similar to Eq. (1) and Eq. (2), we shall not explain their meanings. Solving the two equations, we get $F_2 = \frac{9}{2}$ and $F_3 = 7$.

When $k = 5$, there are 2^5 cases. By our algorithm, extending F_3 to F_5, we can obtain F_5 as follows:

$$F_5 = 1 + \frac{1}{2^5}[C^5_5(F_5 + G_0) + C^5_4(F_4 + G_1) + C^5_3(F_3 + F_2) + C^5_2(F_2 + F_3) + C^5_1(F_1 + F_4) + C^5_0(F_0 + F_5 - 1)]. \tag{8}$$

Since most terms of Eq. (8) are same as the above equations, we shall only explain the first and second term, $C^5_5(F_5 + G_0)$ and $C^5_4(F_4 + G_1)$.

The first term, $C^5_5(F_5 + G_0)$, means that all five stations get a head and bring themselves to the upper active layer, and no station stays in the current layer. In other words, the current layer becomes empty. After the four stations get their ID numbers in some upper layers and the active layer is down to the current layer, the $GUESS$ algorithm, G_0, is invoked. It takes two time slots to identify if the layer is empty. Thus, the subterm, G_0, is given as follows:

$$G_0 = F_0 + F_0 = 2. \tag{9}$$

Nakano and Olariu's algorithm takes only one time slot to identify an empty layer. Comparing with it, our algorithm needs one more time slot. The extra one time slot is a penalty. It is seldom that the upper layer contains five stations and the current layer contains no station. In other words, the penalty occurs with very low probability and in most case it will save a conflict time slot. Thus, it can be recovered and finally it gets a benefit by our second idea.

The second term, $C^5_4(F_4 + G_1)$, means that four stations get a head and bring themselves to the upper active layer, and only one station stays in the current layer. Since there are four stations in the upper layer, thus the station in the current layer would guess that it may contain more than one station in the current layer. Unfortunately, its guess is wrong and it will get a penalty. But fortunately, the probability of getting a penalty is very low. The only one station in the current layer may decide either to move to upper layer or to stay in the current layer. Thus, the subterm, G_1, is given as follows:

$$G_1 = \frac{1}{2}(F_1 + F_0) + \frac{1}{2}(F_0 + F_1) = F_1 + F_0 = 2. \tag{10}$$

Equation (10) means that it takes two time slots to identify if the layer contains only one station.

When $k = 10$, there are 2^{10} cases. By our algorithm, extending F_5 to F_{10}, we can obtain F_{10} as follows:

$$\begin{aligned} F_{10} = {} & 1 + \frac{1}{2^{10}}[C_{10}^{10}(F_{10}+G_0) + C_9^{10}(F_9+G_1) \\ & + \cdots + C_4^{10}(F_4+G_6) \\ & + C_3^{10}(F_3+F_7) + C_2^{10}(F_2+F_8) \\ & + C_1^{10}(F_1+F_9) + C_0^{10}(F_0+F_{10}-1)]. \end{aligned} \tag{11}$$

Take as an example, the sixth term, $C_5^{10}(F_5+G_5)$, means that five stations get a head and bring themselves to the upper active layer, and the other five stations stay in the current layer. The $GUESS$ algorithm, G_5, is then invoked By $GUESS$ algorithm, there are 32 cases. Thus, the subterm, G_5, is given as follows:

$$\begin{aligned} G_5 = {} & \frac{1}{2^5}[C_5^5(F_5+G_0) + C_4^5(F_4+G_1) \\ & + C_3^5(F_3+F_2) + C_2^5(F_2+F_3) \\ & + C_1^5(F_1+F_4) + C_0^5(F_0+F_5)]. \end{aligned} \tag{12}$$

Comparing with Eq. (8), we get

$$G_5 = F_5 - 1 + \frac{1}{2^5}. \tag{13}$$

Substituting Eq. (9), Eq. (10) and Eq. (13) into Eq. (11), we obtain F_{10} as follows:

$$\begin{aligned} F_{10} = {} & 1 + \frac{1}{2^{10}}[C_{10}^{10}(F_{10}+(F_0+F_0)) \\ & + C_9^{10}(F_9+(F_1+F_0)) \\ & + C_8^{10}\left(F_8+\left(F_2-1+\frac{1}{2^2}\right)\right) \\ & + C_7^{10}\left(F_7+\left(F_3-1+\frac{1}{2^3}\right)\right) \\ & + \cdots + C_4^{10}\left(F_4+\left(F_6-1+\frac{1}{2^6}\right)\right) \\ & + C_3^{10}(F_3+F_7) + C_2^{10}(F_2+F_8) \\ & + C_1^{10}(F_1+F_9) + C_0^{10}(F_0+F_{10}-1)]. \end{aligned}$$

Generalizing the above result, for $k \geq 3$, we obtain

$$\begin{aligned} F_n = {} & 1 + \frac{1}{2^n}[C_n^n(F_n+(F_0+F_0)) \\ & + C_{n-1}^n(F_{n-1}+(F_1+F_0)) \\ & + C_{n-2}^n\left(F_{n-2}+\left(F_2-1+\frac{1}{2^2}\right)\right) \\ & + C_{n-3}^n\left(F_{n-3}+\left(F_3-1+\frac{1}{2^3}\right)\right) \\ & + \cdots + C_3^n\left(F_3+\left(F_{n-3}-1+\frac{1}{2^{n-3}}\right)\right) \\ & + C_2^n(F_2+F_{n-2}) \\ & + C_1^n(F_1+F_{n-1}) + C_0^n(F_0+F_n-1)]. \end{aligned}$$

Table 1 Optimal probalility for n stations, $2 \leq n \leq 9$.

n	2	3	4	5
Probability	0.4142	0.4266	0.4197	0.4174
n	6	7	8	9
Probability	0.4167	0.4169	0.4172	0.4175

Lemma 5:

$$\begin{aligned} & \left(1-\frac{1}{2^{n-1}}\right)F_n - \left(1-\frac{1}{2^{n-2}}\right)F_{n-1} \\ & = \frac{1}{2^{n-1}}[C_{n-2}^{n-1}(F_{n-1}-F_{n-2}) + C_{n-3}^{n-1}(F_{n-2}-F_{n-3}) \\ & \quad + \cdots + C_1^{n-1}(F_2-F_1) - 1 + F_{n-1}] \\ & \quad + \frac{1}{2^n}\left[-\frac{1}{2}n + 2 - C_3^{n-1}\right. \\ & \quad \left. - \frac{2^3C_2^{n-1} + 2^2C_1^{n-1} + 2^1C_0^{n-1} + 1}{2^n}\right]. \end{aligned}$$

Lemma 6:

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

Theorem 7:

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

By our simulation in Fig. 6, Theorem 7 can be expressed as follows:

$$\frac{F_n}{n} \cong 2.5. \tag{14}$$

It means that if a partition tree contains 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. (14) with Eq. (4), it is clear that an improvement has been obtained by our 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 to make the partition tree rebalanced. Since the proof of the intuition and rebalance is tedious, we omit it here. We show the optimal probability for each n, $2 \leq n \leq 9$, in Table 1.

The average probability in Table 1 is 0.418. We take it as the bias probability in our CRBP algorithm.

6. Simulations of Our CRBP Algorithm

In Fig. 6, we show the simulation of our first two ideas. 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, respectively the The simulation is executed with two tools, the Mathematica and C language. And the two results are the same.

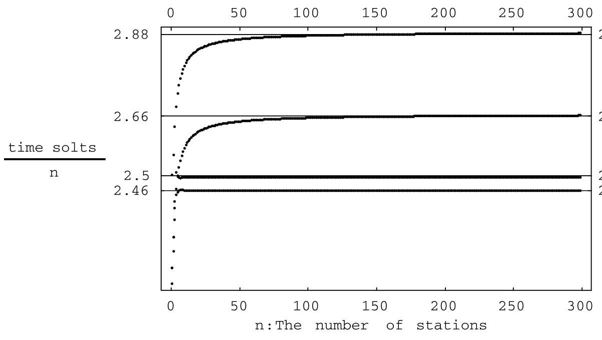

Fig. 6 A simulation for CRBP algorithms with three improvement ideas.

In Fig. 6, our CRBP algorithm with $p = 0.418$ is faster than Nakano and Olariu's algorithm [6]. 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 average. 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 difference.

7. 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[22] is one of those problems. We use the layer concept to obtain a fast algorithm for solving the problem. Nakano and Olariu [6] 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 classes, 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 $2.46n$.

Nakano and Olariu [23] showed a history view on the leader selection problem. Nakano, Olariu and Zomaya [23] leaded a breadth-first view into the energy-efficient routing problem. By these concepts and the method [13], our future work is to find some mining information in the partition tree and to obtain more improvement.

References

[1] 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), pp.159–164, Las Vegas, USA, 2000.
[2] K. Nakano, S. Olariu, and A.Y. Zomaya, "Energy-efficient permutation routing protocols in radio networks," IEEE Trans. Parallel Distrib. Syst., vol.12, no.6, pp.544–557, June 2001.
[3] 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.
[4] 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.
[5] 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.
[6] 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.
[7] S. Levitan, "Algorithms for broadcast protocol multiprocessor," Proc. 3rd International Conference on Distributed Computing Systems, pp.666–671, 1982.
[8] 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.
[9] 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.
[10] C.B. Yang, "Reducing conflict resolution time for solving graph problems in broadcast communications," Inf. Process. Lett., vol.40, pp.295–302, 1991.
[11] 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.
[12] C.U. Martel, "Maximum finding on a multi access broadcast network," Inf. Process. Lett., vol.52, pp.7–13, 1994.
[13] S.H. Shiau and C.B. Yang, "A fast maximum finding algorithm on broadcast communication," Inf. Process. Lett., vol.60, pp.81–96, 1996.
[14] 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.
[15] 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.
[16] 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, 2000.
[17] 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 Univesity, April 2000, also see http://par.cse.nsysu.edu.tw/˜cbyang/person/publish/d00sortubcm.pdf
[18] 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.
[19] K.V.S. Ramarao, "Distributed sorting on local area network," IEEE Trans. Comput., vol.37, no.2, pp.239–243, Feb. 1988.
[20] C.B. Yang, "Computational geometry on the broadcast communication model," J. Information Science and Engineering, vol.15, pp.383–395, May 1999.
[21] 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 Univesity, March 2004, also see http://par.cse.nsysu.edu.tw/˜cbyang/person/publish/d04initwireless.pdf
[22] C.B. Yang, R.C.T. Lee, and W.T. Chen, "Finding minimum spanning trees based upon single-channel broadcast communications," Proc. International Computer Symposium 1988, pp.1451–1456, Taipei, Taiwan, 1988.
[23] 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.

Shyue-Horng Shiau received the B.S. degree from the Department of Engineering Science at National Cheng Kung University, Tainan, Taiwan, in 1984, and the M.S. degree from the Department of Applied Mathematics at National Sun Yat-sen University, Kaohsiung, Taiwan, in 1994. He is currently a Ph.D. 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 B.S. degree in electronic engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1982, and the M.S. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 1984. Then, he received the Ph.D. 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.