單向一致環自我穩定 *k*-值諧齊時計

Self-stabilizing *k*-value unison clocks for unidirectional uniform rings

江振瑞

Jenh-Ruey Jiang

玄奘人文社會學院
資訊管理系

Hsuan Chuang College

Information Management Department

<<摘要>>

在此篇論文中，我們提出一個自我穩定的隨機演算法以製作一個單向一致性環上之 k-值($k\geq 2$)諧齊時計。我們所提出的演算法具有兩個性質：諧齊性質及自我穩定性質。諧齊性質使環上每個節點的時計值在等值之後會以一致的速率增加，而自我穩定特性則使得系統具有容錯特性；也就是說，即使環上節點的時計值在起始狀態下並不一致，所有的時計值最終也會收斂於一個單一值。在此篇論文中，我們證明了所提演算法的正確性，並將所提的演算法與許多相關的方法做了比較。

關鍵字：自我穩定，諧齊時計，容錯性，隨機演算法，單向一致環

Self-stabilizing *k*-value unison clocks for unidirectional uniform rings

Jenh-Ruey Jiang

Hsuan Chuang College

Information Management Department

Abstract

In this paper, we propose a self-stabilizing randomized algorithm that maintains *k*-value unison clock, $k \geq 2$, for uniform rings. The proposed algorithm has two properties: unison property and self-stabilization properties. The unison property means that once all clocks are of a same value, they will increase at the same rate henceforth. And the self-stabilization property makes the system fault-tolerant; namely, even clocks are of different values initially, the clocks will converge to a same value eventually. We have proved the correctness of the algorithm and compare the algorithm with related ones.

Keywords: self-stabilization, unison clock, fault-tolerance, randomized algorithm, unidirectional uniform ring.

1. Introduction

In this paper, we propose a self-stabilizing randomized algorithm that maintains k-value unison clocks, $k \geq 2$, for uniform rings. We say that a ring is uniform if all nodes (processors) on the ring execute a same algorithm. The nodes execute the algorithm phase by phase. At each phase, all nodes simultaneously read the clocks of their left neighbors and then modify their own clocks. We assume the ring unidirectional since each node in the ring reads the left neighbor's clock only. The proposed algorithm has the property of unison; i. e., once all the clocks are of a same value, they will all increase by one in every following phase (and hence all the clocks are of the same value again and again). The proposed algorithm also has the property of self-stabilization; i. e., all clocks will converge to a same value even though they are of different values initially.

The concept of self-stabilization is first introduced by Dijkstra in 1974 [Dijk74], and the study of self-stabilizing systems has become a growing branch of fault-tolerance. A self-stabilizing system can, starting from any possible state, converge to a legitimate state in a finite time. Such a system has the fault-tolerant property: after any unexpected perturbation, the system can recover and return to a legitimate state without outside intervention. A good survey of self-stabilization can be found in the paper of [Sch93].

Several self-stabilizing unison clock algorithms have been proposed in the literature. In 1990, Gouda and Herman proposed a unison clock algorithm for general node topology [GH90]. The drawback of this algorithm is that the clock value is unbounded. Arora, Dolev and Gouda implemented O(n)-value unison clock algorithm for general node topology [ADG91], where n is the number of nodes. Herman and Ghosh [HG95] proposed two 3-value unison clock algorithms. One is for the tree

topology and is deterministic. And the other one is for general node topology and is randomized. Note that all the algorithms mentioned above utilize no variable but the clock variable; thus, the number of node states is equal to the number of clock values. However, in addition to clock variables, the algorithms mentioned below will utilize other supplemental variables. Thus, the number of node states is more than the number of clock values. It goes without saying that it is preferable not to use supplemental variables.

Lin and Simon [LS95] implemented 56-state, binary-value unison clock algorithm for rings of odd or $2q^r$ size, where q is prime and $r \leq 2$. Huang and Liu proposed 4-state, binary-value unison clock algorithm for unidirectional, odd-size, uniform rings [HL98a]. Hunag and Liu also proposed 2^{m+1}-state, 2^m-value unison clock algorithm for unidirectional, odd-sized, uniform rings, where $m \geq 1$ [HL98b]. Huang and Liu further proposed a k^2-state k-value unison clock algorithm, $k \geq 2$, for unidirectional, uniform rings [HL99].

Unison clocks have several applications; below we give two examples. First, we can utilize them to implement a token circulation. In a k-node ring with k-value unison clocks, we just regard the node as the owner of the token if the node's identifier equals to the clock value. Second, if we have a set of nodes whose computation are divided into repeating, serial phases numbered 0, 1, …, $k-1$, then regardless of any initial state, we can utilize unison clock to guide all the nodes to execute the-same-numbered phase.

The rest of this paper is organized as follows. In section 2, we introduce the proposed algorithm. The correctness of the proposed algorithm is given in section 3. In section 4, we compare the proposed algorithm with related ones. At last, we draw a conclusion in section 5.

2. The Algorithm

Consider a ring constituted by n nodes, which are labeled with 0, 1, 2, …, n–1. Since the ring is assumed to be uniform, the labels are only for presenting the algorithm and have nothing to do with algorithm's execution logic. Every node maintains a k-value ($k \geq 2$) clock variable x, and we use $x.i$ to denote "clock of node i." Let node j be the left neighbor of node i, then $j=i-1$ for $i=1, 2, \ldots, n-1$ and $j=n-1$ for $i=0$. The algorithm will be described by a set of rules of the following format:

Condition $\rightarrow$ *Action*;

Each node executes the algorithm (rules) phase by phase. With respect to node i, *Condition* part is a predicate depending on $x.i$ and $x.j$, and *Action* part is an assignment statement that updates $x.i$ referencing $x.i$ or $x.j$. *Action* part is performed while the *Condition* is justified. We say that a node *applies* a rule if the rule's Condition part is justified and the rule's Action part is performed by the node.

The proposed algorithm is represented by the following two rules:

[R1]: $x.i = x.j \quad \rightarrow \quad x.i = x.i \oplus 1$;

[R2]: $x.i \neq x.j \quad \rightarrow \quad x.i = Random(x.i \oplus 1, x.j \oplus 1, \quad p)$;

Note that $\oplus$ represents the modulo-k addition and $Random(a, b, p)$ is a function that will return a with probability p or return b with probability $1-p$.

In the next section, we will prove that the proposed algorithm has the following

two properties:

(1) Unison property: Once all the clocks are of a same value, they will increase by one under modulo-*k* arithmetic in every following phase.
(2) Self-stabilization property: Starting from any possible state where clocks may be of different values, all clocks will be of a same value eventually.

3. Correctness

In this section, we prove that the proposed algorithm satisfies the unison and the self-stabilization properties.

Theorem 1. (unison property) Once all the clocks are of a same value, they will increase by one under modulo-*k* arithmetic in every following phase.

Proof:

When all the clocks are of a same value, every node will apply rule [R1], which makes all clocks increase by one under modulo-*k* arithmetic. □

To prove the self-stabilization property, we first give the following definitions: segment, head and length.

A *segment* is the maximal sequence of consecutive nodes in the ring that have the same clock value. And the left-most node of the segment is called the *head* node. The number of nodes of the segment is called the *length* of the segment. Note that a head node of a segment may join the segment of its left neighbor if it applies rule [R2].

Lemma 1. The number of segments never increases.

Proof:

Consider a certain segment S, in which the nodes have clock value x. Let the head node of S be h, and let y be the clock value of h's left neighbor. After one phase, all nodes of S except h will apply rule [R1] and have clock value $x \oplus 1$. Node h will apply rule [R2] and have clock value $x \oplus 1$ or $y \oplus 1$. If h's clock value is $x \oplus 1$ then h remains in S. Otherwise, if h's clock value is $y \oplus 1$, then h joins the segment of h's left neighbor. In either case, the number of segments does not increase. □

Lemma 2. The number of segments decreases with nonzero probability.

Proof:

Consider three consecutive segments S_1, S_2 and S_3, where the length of S_2 is l. With respect to the length of S_2, there are three cases to consider:

(1) the length of S_2 increases by one:

This case occurs when the head node of S_2 joins S_1 and the head node of S_3 remains in S_3

(2) the length of S_2 decreases by one:

This case occurs when the head node of S_3 joins S_2 and the head node of S_2 remains in S_2

(3) the length of S_2 remains unchanged:

If neither case (1) nor case (2) occurs, the length of S_2 remains unchanged.

If we can find a sequence of phases of algorithm execution such that the occurrences of case (2) outnumber the occurrences of case (1) by l, then segment S_2 disappear and the number of segments decreases. This testifies the lemma. □

Theorem 2. (self-stabilization property) Starting from any possible state where clocks may be of different values, all clocks will be of a same value eventually.

Proof:

To prove the theorem, we can just show that the number of segments will be one eventually. This is the direct consequence of Lemma 1 and Lemma 2. □

4. Comparison with related algorithms

In this section, we compare our algorithm with related ones in terms of the number of node states, the number of clock values, the applicable topology and whether the algorithm is randomized or deterministic. The comparison is illustrated in Table 1.

Algorithm	**Number of node states**	**Number of Clock values**	**Applicable Topology**	**Randomized or Deterministic**
Gouda and Herman's [GH90]	Unbounded	Unbounded	General	Deterministic
Arora , Dolev and Gouda's [ADG91]	O(n)	O(n)	General	Deterministic
Herman and Ghosh's (1) [HG95]	3	3	Trees	Deterministic
Herman and Ghosh's (2) [HG95]	3	3	General	Randomized
Lin and Simon's [LS95]	56	2	Rings of odd or $2q^r$ size, q is prime and $r \leq 2$.	Deterministic
Huang and Liu's (1) [HL98a]	4	2	Unidirectional uniform rings of odd size.	Deterministic
Huang and Liu's (2) [HL98b]	2^{m+1}, $m \geq 1$	2^m , $m \geq 1$	Unidirectional uniform rings of odd size.	Deterministic
Huang and Liu's (3) [HL99]	k^2, $k \geq 2$	k, $k \geq 2$	Unidirectional uniform rings	Hybrid
The proposed algorithm	k, $k \geq 2$	k, $k \geq 2$	Unidirectional uniform rings	Randomized

Table 1. The comparison of the proposed algorithms with related ones.

By Table 1, we know that the proposed algorithm can accommodate any clock values that is larger than 2, and that the proposed algorithm uses no variable but the clock variable. The applicable topology of the proposed algorithm is unidirectional uniform ring with no restriction on the ring size. To sum up, the proposed algorithm is comparable with related ones.

5. Conclusion

In this paper, we have proposed a *k*-value unison clock algorithm for uniform rings, where $k \geq 2$. We have proved the correctness of the algorithm and compared the algorithm with related ones in terms of the number of node states, the number of clock values, the applicable topology and whether the algorithm is randomized or deterministic. The comparison shows that the proposed algorithm is comparable with related ones.

References

[ADG91] Arora A, Dolev S and Gouda M, "Maintaining digital clocks in steps," *Parallel Processing Letters*, vol. 1, pp.11-18, 1991.

[Dijk74] Dijkstra EW, "Self-stabilizing systems in spite of distributed control," *Communication ACM*, vol. 17, no. 11, pp. 643-644, 1974.

[GH90] Gouda MG and Herman T, "Stabilizing unison," *Information Processing Letters*, vol. 35, pp. 171-175, 1990.

[HG95] Herman T and Ghosh S, "Stabilizing phase-clocks," *Information Processing Letters*, vol. 54, pp. 259-265, 1995.

[HL98a] Huang ST and Liu TJ, "Four-state stabilizing phase clock for unidirectional rings of odd size," *Information Processing Letters*, vol. 65,

pp. 325-329, 1998.

[HL98b] Huang ST and Liu TJ, “Self-stabilizing 2^m-clock for unidirectional rings of odd size,” *Technical Report*, Tsing Hua University, 1998.

[HL99] Huang ST and Liu TJ, “Self-stabilizing *k*-clock for unidirectional rings,” *Technical Report*, Tsing Hua University, 1999.

[LS95] Lin C and Simon J, ”Possibility and Impossibility results for self-stabilizing binary clocks on synchronous ring,” *Proceedings of the second workshop on self-stabilizing systems*, pp. 10.1-10.15, 1995.

[Sch93] Schneider M, “Self-stabilization,” *ACM Computing Survey*, vol. 25, pp. 45-67, 1993.