Mobile Agent Rendezvous When Tokens Fail

P. Flocchini
School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada
Email: flocchin@site.uottawa.ca
F. Luccio
Departimento di Scienza Matematiche, Trieste, Italy
Email: luccio@dsm.univ.trieste.it
E. Kranakis
School of Computer Science, Carleton University, Ottawa, Canada
kranakis@scs.carleton.ca
D. Krizanc
Wesleyan University, Middletown, CT, USA
Email: dkrizanc @wesleyan.edu
N. Santoro
School of Computer Science, Carleton University, Ottawa, Canada
Email: santoro@scs.carleton.ca
C. Sawchuk
School of Computer Science, Carleton University, Ottawa, Canada
Email: sawchuk@scs.carleton.ca

January 1, 2004

Abstract

The mobile agent rendezvous problem consists of $k \geq 2$ mobile agents trying to rendezvous or meet in a minimum amount of time on an n node network. Tokens and markers have been used successfully to achieve rendezvous when the problem is symmetric, e.g., the network is an anonymous ring and the mobile agents are identical and run the same deterministic algorithm. In this paper, we explore how token failure affects the time and memory requirements of mobile agent rendezvous under symmetric conditions.

1 Introduction

Tokens [3], [4] and markers [2] have been used successfully to achieve rendezvous when the mobile agent rendezvous problem is symmetric, e.g., the network is an anonymous ring and the mobile agents are identical and run the same deterministic algorithm. The tokens and markers were considered to be unfailing and thus always visible to any mobile agent on the same node as a token. In this paper, we explore how token failure affects the time and memory requirements of mobile agent rendezvous.

Each mobile agent has a single, identical, stationary token which consists of a single bit of memory. In the first step of a rendezvous algorithm, each mobile agent places its token on the node that it currently occupies. As mentioned, an unfailing token is always visible to any mobile agent on the same node as the token. If the token fails, however, it is no longer visible to any mobile agent and it remains in the failed state for the rest of the rendezvous algorithm.

1.1 The Network Model

The network model consists of $k \geq 2$ identical mobile agents in an anonymous, synchronous, n node ring. Each mobile agent, MA, owns a single, identical, stationary token that is comprised of one bit. A given node requires only enough memory to host a token and, at most, k mobile agents.

The MAs follow the same deterministic algorithm and begin execution at the same time. A MA releases its token in the first step of any rendezvous algorithm. Since the tokens are stationary, the original intertoken distances are maintained unless a token fails. A token fails when it is no longer visible to any MA on the same node. Tokens may fail upon release or may fail later. If a token has not failed, then it and any MA on a given node are visible to all MAs on the same node, but are not visible to any other MAs.

1.2 Outline of the Paper

Three cases of token failure are investigated. First, we assume tokens can fail only upon release. We prove that if the MAs have $O(k \log n)$ memory, $\gcd(k', n) = 1$ for all $k' \leq k$, and at most $k - 1$ tokens fail, then the mobile agent rendezvous problem can be solved in time $O(kn)$.

Second, we assume tokens can fail at anytime. We prove that if the MAs have $O(k \log n)$ memory, $\gcd(k', n) = 1$ for all $k' \leq k$, and at most $k - 1$ tokens fail, then the mobile agent rendezvous problem can be solved in time $O(k^2 n)$.

Finally, we assume tokens can fail at anytime, $\gcd(k', n) = 1$ for all $k' \leq k$, at most $k - 1$ tokens fail, and the MAs know n, the number of nodes in the ring. We prove that if the MAs have $O(\log n)$ memory, then the mobile agent rendezvous problem can be solved in time $O(kn)$.

We conclude by comparing the time and memory requirements of rendezvous when tokens can fail to those when tokens cannot fail.

2 Rendezvous When Tokens Fail Upon Release

First, we assume a token can fail only upon release, i.e., in the first step of a given algorithm. The MA that released the token is unaware that it failed. The MAs known k so, with adequate memory, they can walk around the ring and calculate the sequence $\mathcal{S} = d_1, \ldots, d_{3k}$, i.e., the sequence of the first $3k$ intertoken distances. Let $\mathcal{S}^{\mathcal{R}}$ denote the reverse of $\mathcal{S}$. Since the tokens fail only upon release, $\mathcal{S}^{\mathcal{R}}$ can be partitioned as follows:

$$\mathcal{S}^{\mathcal{R}} = Q^q + d_1, \ldots, d_\gamma \tag{1}$$

where Q^q is the concatenation of q copies of a unique aperiodic subsequence Q, $+$ is the concatenation operator, and $d_1, \ldots, d_\gamma$ is a subsequence such that $\gamma < |Q|$. Upon identifying the subsequence Q, the MAs can identify a unique node upon which to rendezvous.

Algorithm 1

```
1. Release the token at the starting node.
2. Choose a direction and start walking.
3. Compute the sequence of $3k$ intertoken distances i.e., $S = d_1, d_2, \ldots, d_{3k}$.
4. Let $S^R$ be the reverse of $S$.
5. Find the shortest aperiodic subsequence $Q$ that starts with the first element of
   $S^R$ and is repeated such that $S^R = Q^q + d_1, \ldots, d_\gamma$ where $\gamma < |Q|$.
6. Let $Q^R$ be the reverse of $Q$.
7. Let $lexi(someSequence)$ denote the lexicographically maximum rotation of
   $someSequence$.
8. Set $forward = lexi(Q)$ and $reverse = lexi(Q^R)$.
9. If $forward$ and $reverse$ differ, then determine which of these sequences is the
   lexicographic maximum and rendezvous at the node where this sequence begins.
10. Else let $MA_i$ and $MA_j$ denote the $MA$s at the beginning of $forward$ and
    $reverse$ respectively.
11. If $MA_i$ and $MA_j$ are the same $MA$, then rendezvous at the node where $MA_i$
    resides.
12. If $MA_i$ and $MA_j$ are distinct $MA$s, then look at the two paths between $MA_i$
    and $MA_j$ in the ring.
    i) If only one of the paths had an odd number of nodes,
    then rendezvous at the node in the midpoint of that path.
    ii) If both paths have an odd number of nodes, then
        a) if the paths differ in length, rendezvous at the
        midpoint of the shorter path,
        b) else compare the sequences of intertoken distances for
        the two paths and rendezvous at the node in the midpoint
        of the path that is the lexicographic maximum.
    iii) If both paths have an even number of nodes, then
    rendezvous at the node in the midpoint of the path that
    contains an odd number of $MA$s.
```

Theorem 1 *When the MAs have $O(k \log n)$ memory, $\gcd(k', n) = 1$ holds for all $k' \leq k$, $f \leq (k-1)$ tokens fail, and tokens can fail only upon release, then the mobile agent rendezvous problem can be solved in $O(kn)$ time.*

Proof of Theorem 1.

Let a be the number of tokens that do not fail, i.e., $a = k - f$. Let $A = d_1, \ldots, d_a$ be the sequence of intertoken distances that exist after the f tokens have failed such that $\sum_{i=1}^{a} d_i = n$. Let S be the sequence of $3k$ intertoken distances calculated by a given MA in step 3 of Algorithm 1. Let S^R be the reverse of S. With a renumbering of the intertoken distances in S,

$$S_R = A^\rho + d_1, \ldots, d_\gamma = (d_1, \ldots, d_a)^\rho + d_1, \ldots, d_\gamma. \qquad (2)$$

where A^ρ is the concatenation of ρ copies of the aperiodic subsequence A, $+$ is the concatenation operator, and $d_1, \ldots, d_\gamma$ is a subsequence such that $\gamma < a$. Thus there exists at least one aperiodic subsequence, namely A, that satisfies equation 1. If A is the shortest subsequence that satisfies equation 1, the MAs discover A in step 5 of Algorithm 1. Otherwise, the MAs discover a shorter aperiodic subsequence, Q, that satisfies equation 1.

The subsequence discovered in step 5 of Algorithm 1 is unique. If the shortest subsequence has z elements, these elements are the first z elements of S^R. Any other subsequence of the same length that satisfies equation 1 is also comprised of the first z elements of S^R and thus the subsequence discovered in step 5 is unique. This implies that all the MAs identify the same rendezvous node in the remaining steps of Algorithm 1 and rendezvous occurs.

Calculating S, the sequence of $3k$ intertoken distances requires $O(k \log n)$ memory and requires $O(kn)$ time. Identifying the appropriate subsequence in step 5, determining the rendezvous node, and walking to the rendezvous node can be done in $O(kn)$ time as well, so the overall time requirement is $O(kn)$. This completes the proof of Theorem 1. ■

It is interesting to note that when k is known and the tokens only fail upon release, Algorithm 1 also solves the mobile agent rendezvous problem when the ring is asynchronous.

3 Rendezvous When Tokens Fail After Release And the Ring Size is Unknown

Suppose token failures occur after release. In the following algorithm, if more than one but fewer than k MAs meet on a given node, then a partial rendezvous occurs, i.e., the MAs *merge* and act as one MA for the remainder of the algorithm.

Algorithm 2

```
1. Release token.
2. Set $r = 0$, where $r$ denotes a round of the algorithm.
3. Choose a direction and begin walking.
4. Upon meeting another $MA$, merge with that $MA$.
5. Calculate the first $k - r$ intertoken distances, i.e., $S = (d_1, \ldots, d_{k-r})$.
6. Estimate $n$ as $\hat{n} = \sum_{i=1}^{k-r} d_i$.
7. Calculate $S_{LMR}$, the lexicographically maximum rotation of $S$.
8. Set $h = 0$.
9. Walk to the node that starts $S_{LMR}$ and increment $h$ for each node travelled.
10. Wait $2\hat{n}$ - $h$ clock ticks.
11. If there are $k$ $MA$s or their merged equivalent on the current node, stop.
/* Rendezvous has occurred. */
12. Else if there are $1 < v < k$ $MA$s on the current node, then merge.
13. Set $r = r + 1$ and repeat from step 3.
```

The following three lemmata are used in the proof of Theorem 2. Lemma 1 demonstrates that the MAs are always less than a round apart. In fact, if MA_j is the first MA to complete step 5 of round r and does so at time τ, then all other MAs either merge with MA_j or complete round $r - 1$ by time τ. As a result, MA_j need only wait $3\hat{n}/2$ clock ticks for other MAs that have the same view. This lemma therefore provides the appropriate waiting time for step 10 of Algorithm 2.

Lemma 1 *Given a mobile agent MA^* and $0 \leq r \leq f - 1$, all other MAs will either finish round r or merge with MA^* by the time MA^* finishes step 5 of round $r + 1$ in Algorithm 2.*

Proof of Lemma 1.
Base case: $r = 0$.
Let MA_i be an arbitrary MA, other than MA^*, that finishes round $r = 0$ after MA^* does. Let $\hat{n}$ and $\tilde{n}$ denote the estimates for n calculated by MA^* and MA_i respectively in round $r = 0$. The two MAs will meet and merge if:

- *Case* 1: MA^*'s wait in step 10 of round $r = 0$ overlaps MA_i's walk in step 5 of round $r = 0$ by at least n steps, or
- *Case* 2: MA_i's wait in step 10 of round $r = 0$ overlaps MA^*'s walk in step 5 of round $r = 1$ by at least n steps.

If the two MAs do not meet and merge in round $r = 0$, then

- *Case* 3: MA_i must finish round $r = 0$ on or before the time that MA^* finishes step 5 of round $r = 1$.

In round $r = 0$ of Algorithm 2, MA^* begins waiting no later than $\frac{3}{2}\hat{n}$, thus *Case 1* requires that

$$\frac{3}{2}\hat{n} + n \leq \tilde{n}. \tag{3}$$

Suppose that *Case* 1 does not hold. *Case* 3 requires that

$$3\tilde{n} \leq 3\hat{n} + n^* \tag{4}$$

where n^* is the estimate for n calculated by MA^* in step 5 of round $r = 1$.
Suppose that *Case* 3 does not hold either. The final case, *Case* 2, requires that

$$\frac{3}{2}\tilde{n} \leq 3\hat{n}. \tag{5}$$

Since *Case* 3 does not hold, equation 4 implies that

$$3\hat{n} + n^* < 3\tilde{n} \tag{6}$$

and thus MA_i waits in round $r = 0$ while MA^* walks at least n steps in step 5 of round $r = 1$ and the two MAs subsequently meet and merge.
Suppose that, like the previous cases, *Case 2* does not hold. Since equation 5 does not hold, then

$$3\hat{n} < \frac{3}{2}\tilde{n}. \tag{7}$$

This implies, however, that $2\hat{n} < \tilde{n}$ and thus contradicts the fact that *Case 1* does not hold. Therefore one of the cases must hold and thus, on or before MA^* finishes step 5 of round $r = 1$, either MA^* and MA_i meet and merge or MA_i finishes round $r = 0$.
Inductive Hypothesis: The lemma holds for $r = q$.
$r = q + 1$:
The inductive hypothesis implies that in round $r = q$, the two MAs either meet and merge, or MA_i finishes round $r = q$ on or before MA^* finishes step 5 of round $r = q + 1$. MA_i only starts round $r = q + 1$ if the two MAs did not meet in round $r = q$, i.e., MA_i finished round $r = q$ on or before MA^* finished step 5 of round $r = q + 1$.

Let t_0 and t_1 denote the respective times that MA^* and MA_i begin round $r = q+1$ and let $\delta = t_0 - t_1$. The inductive hypothesis implies that $\delta > 0$ and that MA_i begins round $r = q+1$ no later than $t_0 + \hat{n}$, where $\hat{n}$ is the estimate for n calculated by MA^* in round $r = q+1$.

The two MAs will meet and merge in round $r = q+1$ if

- *Case* $1'$: MA^*'s wait in step 10 of round $r = q+1$ overlaps MA_i's walk in step 5 of round $r = q+1$ by at least n steps, or
- *Case* $2'$: MA_i's wait in step 10 of round $r = q+1$ overlaps MA^*'s walk in step 5 of round $r = q+2$ by at least n steps.

If the two MAs do not meet and merge in round $r = q+1$, then

- *Case* $3'$: MA_i must finish round $r = q+1$ on or before the time that MA^* finishes step 5 of round $r = q+2$.

Case $1'$ occurs if

$$t_0 + \frac{3}{2}\hat{n} + n \leq t_0 + \delta + \tilde{n} \tag{8}$$

where $\hat{n}$ and $\tilde{n}$ are MA^* and MA_i's respective estimates for n in round $r = q+1$.

Suppose that *Case* $1'$ does not hold. *Case* $3'$ occurs if MA_i finishes round $r = q+1$ on or before MA^* finishes step 5 of round $r = q+2$, i.e.,

$$t_0 + \delta + 3\tilde{n} \leq t_0 + 3\hat{n} + n^* \tag{9}$$

where n^* is MA^*'s estimate for n in round $r = q+2$.

Suppose that *Case* $3'$ does not hold. *Case* $2'$ occurs if

$$\delta + \frac{3}{2}\tilde{n} \leq 3\hat{n}. \tag{10}$$

However, since *Case* $3'$ does not hold, then

$$3\hat{n} < 3\hat{n} + n^* < \delta + 3\tilde{n}. \tag{11}$$

and thus MA_i waits in round $r = q+1$ while MA^* walks at least n nodes in step 5 of round $r = q+2$. This contradicts the fact that *Case* $1'$ did not occur. Thus one of three cases must occur when MA_i is in round $r = q+1$. This ends the proof of Lemma 1. ■

Lemma 2 proves that the MAs which see the same sequence of intertoken distances in a given round will rendezvous in that round.

Lemma 2 *The MAs that see the same sequence, up to a rotation, of intertoken distances S in a given round will rendezvous in that round.*

Proof of Lemma 2.
The MAs that see the same sequence of intertoken distances S, up to a rotation, will have the same estimate for $\hat{n}$, and will identify the same rendezvous node. Let t_0 denote the time that the first MA finishes calculating S. Lemma 1 implies that the remaining MAs that see rotations of S will start calculating those rotations no later than t_0. The first MA will wait at

the rendezvous node from no later than $t_0 + \frac{\hat{n}}{2}$ until exactly $t_0 + 2\hat{n}$. The remaining MAs that see rotations of S arrive at the rendezvous node no later than $t_0 + \frac{3\hat{n}}{2}$, and thus all MAs that saw rotations of S in a given round will rendezvous by the end of that round. This completes the proof of Lemma 2. ■

Lemma 3 ensures that no MA will *overshoot* and execute a round r where r exceeds the number of existing failures f.

Lemma 3 *A mobile agent MA_i will not execute round $r = f$ if fewer than f tokens have failed by the time that all MAs finishes step* 5 *of round* $f - 1$.

Proof of Lemma 3.
Base case: $f = 1$. If no tokens fail before all MAs, including MA_i, complete step 5 of round $r = 0$ then the MAs see, up to a rotation, the same sequence of intertoken distances S and thus identify the same rendezvous node. Since rendezvous occurs in round $r = 0$, round $r = 1$ is not executed.

Inductive hypothesis: The theorem is true for $f = q$.

Case $r = q + 1$:
If exactly q of $q + 1$ token failures have occurred, then the MAs are in rounds $r \leq q$. If all of the MAs do not merge or rendezvous in one of the first $q - 1$ rounds, then some MAs execute round q. If no more tokens fail before these MAs finish step 5 of round q, then all the MAs in round q calculate, up to a rotation, the same sequence of intertoken distances, S, and thus rendezvous occurs in round $r = q$. If an additional token fails before all the MAs finish step 5 of round $r = q$, then the MAs may calculate sequences of intertoken distances that differ by more than a rotation and thus rendezvous may not occur. Those MAs that continue to round $r = q + 1$, however, all calculate the same sequence, up to a rotation, of intertoken distances and thus rendezvous occurs in round $r = q + 1$. This completes the proof of Lemma 3. ■

Theorem 2 *When the MAs have $O(k \log n)$ memory,* $\gcd(k', n) = 1$ *holds for all $k' \leq k$, at most $(k - 1)$ tokens fail, and token failures occur after release, then the mobile agent rendezvous problem can be solved in $O(k^2 n)$ time.*

Proof of Theorem 2.
Let $f \leq k - 1$ be the number of tokens that actually fail. Lemma 3 implies that no MA will execute more than f rounds of Algorithm 2. Suppose that rendezvous has not occurred by the end of round $r = f - 1$. Let MA^* denote the first MA that begins round $r = f$ and let t_0 denote the time when MA^* starts round $r = f$ of Algorithm 2. Since f tokens have failed, MA^*s estimate for n will be correct, i.e., $\hat{n} = n$. The remaining MAs see the same sequence, up to a rotation, of intertoken distances as MA^*, and thus Lemma 2 implies that rendezvous occurs at the end of round f.

The number of failures, f, is at most $k - 1$ so at most $k - 1$ rounds of Algorithm 2 are executed. A round takes at most $k(n - 1)$ time, i.e., the product of the number of intertoken distances measured and the maximum intertoken distance possible. The resulting time required is $O(k^2 n)$. Because at most k intertoken distances are calculated and the maximum intertoken distance is $(n - 1)$, the resulting memory complexity is $O(k \log n)$. This completes the proof of Theorem 2. ■

4 Rendezvous When Tokens Fail After Release and the Ring Size is Known

The following algorithm is useful when the MAs know not only k, the number of MAs, but also know n, the number of nodes in the ring. With adequate memory, the MAs can walk around the ring and calculate the sequence $\mathcal{S} = d_1, \ldots, d_{3n}$ of $3n$ intertoken distances. If $\mathcal{S}^{\mathcal{R}}$, the reverse of $\mathcal{S}$, begins with an aperiodic subsequence A that is repeated at least twice, the MAs can identify a unique node upon which to rendezvous. Otherwise, the MAs restart the algorithm. A MA that identifies a node for rendezvous would wait at that node until all MAs that could have seen the same view have arrived at that node. If the equivalent of k MAs arrived at the node, rendezvous would occur. Otherwise, the MAs restart the algorithm.

Algorithm 3

```
1. Release the token at the starting node.
2. Choose a direction and start walking.
3. Compute the sequence of intertoken distances for $3n$ steps, i.e., $S = d_1, d_2, \ldots, d_x$
   such that $\sum_{i \in S} d_i = 3n$.
4. Let $S^R$ be the reverse of $S$.
5. Attempt to find an aperiodic subsequence $A$ that starts at the first element of
   $S^R$ and is repeated at least twice.
6. If no such subsequence exists, wait $n/2$ clock ticks and repeat from step 3.
7. Else let $A^R$ be the reverse of $A$.
8. Let $lexi(someSequence)$ denote the lexicographically maximum rotation of
   $someSequence$.
9. Set $forward = lexi(A)$ and $reverse = lexi(A^R)$.
10. Execute steps 9 through 12 of Algorithm 1.
11. Wait $n/2$ minus the distance travelled in step 10.
12. If $k$ $MA$s or their merged equivalent have arrives at the current node, ren-
    dezvous is complete.
13. Else repeat from step 3.
```

Theorem 3 *When the MAs have $O(\log n)$ memory, know k and n, $\gcd(k', n) = 1$ for all $k' \leq k$, at most $k-1$ failures occur, and tokens failures occur at anytime, then the mobile agent rendezvous problem can be solved in $O(kn)$ time.*

Proof of Theorem **??** Rendezvous fails when at least two MAs see different views during a given $3n$ node walk around the ring or when at least one MA does not see a repeated aperiodic sequence. Both of these situations arise when, in a given $3n$ step walk around the ring, at least one token fails. Since at most $k-1$ tokens can fail, then at most $k-1$ of the $3n$ step walks can be completed before all MAs see the same view. Thus rendezvous requires $O(kn)$ time. This completes the proof for Theorem 3. ■

5 The Cost of Token Failure

When tokens fail, the time and memory requirements of the mobile agent rendezvous problem increase. In Table 1, we compare the memory and time requirements for rendezvous with and without token failure.

$Knowledge$	$Memory$	$TokensFail$	$Time$
k	$O(k \log n)$	no	$O(n)$
	$O(k \log n)$	upon release	$O(kn)$
	$O(k \log n)$	anytime	$O(k^2 n)$
k	$O(\log n)$	no	$O(kn)$
k, n	$O(\log n)$	anytime	$O(kn)$

Table 1: The Cost of Token Failure

Kranakis et al [3] proved that when k is known and tokens cannot fail, the $k \geq 2$ mobile agent rendezvous problem can be solved with $O(k \log n)$ memory and $O(n)$ time. We proved that when tokens can fail upon release, however, the time required for rendezvous increases to $O(kn)$. If tokens can fail at anytime, the time required for rendezvous increases to $O(k^2 n)$.

Flocchini et al [4] also proved that when k is known and tokens cannot fail, the $k \geq 2$ mobile agent rendezvous problem can be solved with $O(\log n)$ memory and $O(kn)$ time. When tokens can fail at anytime, the memory and time required for rendezvous can be held to $O(\log n)$ and $O(kn)$ respectively but the mobile agents need to know n in addition to k.

6 Conclusion

The effect of token failure on the time and memory requirements of rendezvous suggests that it would be interesting to explore other sources of failure in the mobile agent rendezvous problem. For example, what are the implications for rendezvous when mobile agents fail? Mobile agent failure could be partial, such as not merging when appropriate, or absolute, such as not operating at all. It would also be interesting to determine the impact of network problems, such as heavy traffic, on mobile agent rendezvous.

References

[1] S. Alpern and S. Gal, The Theory of Search Games and Rendezvous, Kluwer Academic Publishers, Norwell, Massachusetts, 2003.

[2] V. Baston and S. Gal, Rendezvous Search When Marks are Left at the Staritng Points, Naval Research Logistics, 38, pp. 494-494, 1991.

[3] E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, Mobile Agent Rendezvous Problem in the Ring, International Conference on Distributed Computing System (ICDCS), pp. 592-599, 2003.

[4] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, Multiple Mobile Agent Rendezvous in the Ring, to appear, Proceedings of Latin America Theoretical Informatics (LATIN), 2004.

[5] Agent Rendezvous: A Dynamic Symmetry-Breaking Problem, Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP), LNCS 1099, pp. 610-621, 1996.