Mobile Agents Rendezvous When Tokens Fail

Paola Flocchini* Evangelos Kranakis† Danny Krizanc‡ Flaminia L. Luccio§
Nicola Santoro† Cindy Sawchuk†

February 27, 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 ring 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 required for mobile agent rendezvous under symmetric conditions with different types of knowledge. Our results suggest that knowledge of n is better than knowledge of k in terms of achieving rendezvous as quickly as possible in the faulty token setting.

*School of Information Technology and Engineering, University of Ottawa, <flocchin@uottawa.ca>
†School of Computer Science, Carleton University, <{kranakis,santoro,sawchuk}@scs.carleton.ca>
‡Department of Mathematics, Wesleyan University, <dkrizanc@wesleyan.edu>
§Dipartimento di Scienze Matematiche, Universita di Trieste, <luccio@dsm.univ.trieste.it>

1 Introduction

1.1 The Problem

The general *rendezvous search* problem includes many situations that arise in the real world, e.g., searching for or regrouping animals, people, equipment, and vehicles. It has been extensively studied (see [1] for a detailed review), with operations research specialists producing the bulk of this research. In these investigations, the search domain is usually assumed to be *continuous*, and *symmetry* occurs largely because locations in the search domain are indistinguishable. The searchers break the symmetry by running the *same randomized algorithm* or by running *different deterministic algorithms*. In the latter case, for example, one searcher remains stationary while another searches the domain. Even when the searchers in a general rendezvous problem marked their starting points, as prescribed by Baston and Gal [3], the case where searchers ran the *same deterministic algorithm* was largely unexplored.

Computer scientists have been interested in the rendezvous search problem in the context of mobile agents and networks. After being dispersed in a network, mobile agents may need to gather or rendezvous in order to share information, receive new instructions, or download new programs that extend the mobile agents' capabilities. The *mobile agent rendezvous* problem differs from the general rendezvous problem in significant ways. Network topologies are *discrete* and include highly symmetric structures like rings, tori, and hypercubes. In addition, memory and time requirements are important in computer science problems but in the general rendezvous problem research the *memory* requirements are usually ignored.

In the mobile agent rendezvous problem, *symmetry* can occur at several levels (topological structure, nodes, agents, algorithm), each playing a key role in difficulty of the problem and of its resolution. The standard symmetric situation is indeed a *ring* network of n identical nodes in which k identical agents running the same deterministic protocol must rendezvous.

If no other computational power is given to the agents, rendezvous becomes impossible to achieve by deterministic means (eg. see [9]), and thus the problem is in general deterministically *unsolvable*.

The research has thus focused on what additional powers the agents need to perform the task. The aim is to identify the "weakest" possible condition. Some investigators have considered empowering the agents with unique *identifiers*; this is for example the case of of Yu and Yung [9], and of Dessmark, Fraigniaud, and Pelc [4] who also assume unbounded memory; note that having different identities allows each agent to execute different algorithms. Other researchers have empowered the agents with the ability to leave notes in each node they travel, i.e., *read/write* capacity; this is the model of Barrière *et al* [2], and of Dobrev *et al* [5].

In the third approach, originally developed by Baston and Gal [3] for the general rendezvous problem, the agents are given something much less powerful than distinct identities or than the ability to write in every node. Specifically, each agent has available a marker, called *token*, placed in its home base, visible to any agent passing by that node; the tokens are however all *identical*. Since the tokens are stationary, the original intertoken distances are preserved while the agents move and, with enough memory, could be used to break the symmetry of the setting. This is the approach taken by Kranakis *et al* [6] and Flocchini *et al* [7], and is the one we follow in this paper.

In the case of $k = 2$ mobile agents, Kranakis *et al* [6] proved that solving the rendezvous problem in an n node anonymous ring requires $\theta(\log \log n)$ memory; several algorithms were presented to demonstrate that when more memory is available, rendezvous time can be reduced.

Flocchini *et al* [7] studied the use of tokens to break symmetry when $k \geq 2$ and proved that when the agents run the same deterministic algorithm, rendezvous is impossible if the sequence of

TokensFail	*Knowledge*	*Time*	*Algorithm*
never	n or k	$O(n)$	[7]
upon release	n	$O(n)$	$Meet1$
	k	$O(k\ n)$	$Meet2$
anytime	n	$O(k\ n)$	$Meet3$
	k	$O(k^2\ n)$	$Meet4$

Table 1: The Cost of Token Failure

the original intertoken distances is periodic. They also presented several algorithms, with various memory and time requirements, that solve the mobile agent rendezvous problem for two or more mobile agents. Finally, they proved that a solution to the leader election in the $k \geq 2$ mobile agent case implies a solution to the mobile agent rendezvous problem, but the reverse is not true.

The tokens used in the research of Kranakis *et al* [6] and Flocchini *et al* [7] were not subject to failure. Once placed on a node, a token was always visible to any mobile agent on the same node.

In this paper, we are concerned with *fault tolerance*; we investigate how the mobile agent rendezvous problem changes when *tokens can fail*. Such failures may be the result of hostile nodes or malicious mobile agents and may occur either as soon as the token is placed by the mobile agent or at any time later.

1.2 Our Contribution

In this paper, we study the rendezvous of identical agents in an anonymous ring when tokens may fail. We are concerned with the failure of token which will make it no longer visible by the

agents, and investigate how time required to rendezvous changes under symmetric conditions with different knowledge available to the agents, in particular, knowledge of the number of agents and the size of the ring.

We consider two types of failures. We consider first the case when tokens can fail and thus disappear *upon their release*. We then consider the more complex situations arising when the tokens can fail *anytime*.

Some facts can be determined quite easily. If both the number k of agents and the size n of the ring are unknown, the problem is unsolvable [7]. If all tokens fail, clearly no rendezvous is possible. In this paper we assume that either n or k is known to all agents, that the number f of failed token is $f < k$. Furthermore, we assume that $\gcd(k', n) = 1$ for all $k' \leq k$. Under these conditions we show that, rendezvous is possible regardless of the number of failures and of when they occur. It follows from [7] that if this condition does not hold, there will always be initial conditions under which rendezvous is not possible.

The solutions and their time complexity are clearly different depending on the types of faults, as well as on whether the agents know the size n of the ring or their number k. The time required by our solution protocols is summarized in Table 1. When tokens are reliable and never fail, the existing algorithm [7] ensures rendezvous in $O(n)$ time when either k or n is known. We note that all of these algorithms require $O(k \log n)$ memory.

If the token fail upon release, knowledge of n allows the agents to overcome all the failures; in fact, rendezvous takes place within $O(n)$ time units. If the agents know k, we propose an algorithm that uses $O(kn)$ time units to rendezvous. Interestingly, this bound holds also if the ring is *asynchronous*.

If tokens can fail anytime, but the mobile agents know n, we show that the time required for rendezvous only increases by a factor of k over that required when no tokens fail. On the other hand, if the agents know only k, we describe an algorithm that uses $O(k^2 n)$ time units to rendezvous. Both of these algorithms work in the synchronous setting.

For both types of failures, our results suggest that knowledge of n yields a time saving over knowledge of just k. We note that in the fault-free environment this does not hold, in that it is always possible to compute n from k, or vice versa, in $O(n)$ time.

2 Definitions and Basic Properties

We consider a collection of $k \geq 2$ identical mobile *agents* scattered on a synchronous ring of n identical nodes. The ring is *unoriented*; that, is there is no globally consistent indication of "left", and the agents have no globally consistent notion of "clockwise". Each mobile agent 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.

Each agent is initially in a different node (its *home base*) whose location is unknown to the others. The agents follow the same deterministic algorithm and begin execution at the same time. The task for the agents is to *rendezvous*, i.e., to meet in the same node in finite time; the goal is to do so in as little time as possible. The problem is to design an algorithm, the same for all agents, that will allow them to perform the task. An agent releases its token in the first step of any rendezvous algorithm; when an agent transits on a node with a token, it can see it. Moreover, agents can see each other when they are in the same node.

We call *intertoken distances* the sequence of distances between the fault-free tokens, once they have been released. Since the tokens are stationary, the original intertoken distances are maintained unless a token fails.

When a token fails it is no longer visible to any agent passing by the node. We will consider two types of failures: when tokens can fail and thus disappear *upon their release*, and when the tokens can fail *anytime*. Let f be the number of token that fail.

Some limiting facts can be determined quite easily.

Fact 1 *The problem is unsolvable:*
1. If both the number k of agents and the size n of the ring are unknown.
2. If all tokens fail; i.e., $f = k$.

Proof Part (1) is due to [7]. Part (2) is well-known. ■

We assume that the agents of knowledge of either k or n and that the number f of failed token is less than k. We also assume that that $\gcd(k', n) = 1$ for all $k' \leq k$. It follows from [7] that if this condition does not hold, there will exist initial configurations under which rendezvous will not be possible.

3 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 agent that released the token is unaware that it failed. If neither n, the size of the ring, nor k, the number of agents, are known to the agent the problem is unsolvable [7]; thus, in the following we will consider two cases: when only n is known and when only k is known.

3.1 n known, k unknown

The algorithm is rather simple. Each agent relases its token, and moves along the ring computing the distances between the successive tokens it encounters, and returns to its homebase. Let $s_1, \ldots, s_h$ be the sequence it computed, where s_1 and s_h are the distances between the homebase and the first encountered token, and between the last token and the homebase, respectively. If its own token did not fail, $h = k - f$ and this sequence is exactly the (circular) sequence $\mathcal{S} = d_1, \ldots, d_h$ of the intertoken distances of the tokens that did not fail. If however the agent finds that its token has failed, then $h = k - f + 1$ and the agent must adjust the sequence to determine the actual (circular) sequence of intertoken distances: $\mathcal{S} = d_1, \ldots, d_{h-1} = s_1 + s_h, s_2, \ldots, s_{h-1}$.

Since the failures happen before the agents start to move around, the circular sequence computed in this way is the same for all agents within a rotation (due to the fact that the ring is not oriented).

The agents will then agree on a meeting node by exploiting the asymmetry of the sequence (which cannot be periodic since $\gcd(k', n) \neq 1$ for all $k' \leq k$). More precisely, let $\mathcal{S}^{\mathcal{R}}$ denote the reverse of $\mathcal{S}$; each agent will compute the lexicographically maximum string in both $\mathcal{S}$ and $\mathcal{S}^{\mathcal{R}}$. If both strings start at the same node, that will become the meeting point; otherwise, the meeting point will be the midpoint in the odd path between the starting nodes of the two strings.

The routine to identify the meeting point, which will be used in all our algorithms, is described in Table 2, where $lexi(someSequence)$ denotes the lexicographically maximum rotation of $someSequence$. The overall algorithm is described in Table 3. The proof of the following theorem is straightforward.

Meetingpoint(string)
1. Compute $forward = lexi(string)$ and $reverse = lexi(string^R)$.
2. Let x and y denote the nodes at the beginning of *forward* and *reverse* respectively.
3. If $x = y$, then x is the rendezvous point.
4. If $x \neq y$, then the rendezvous point is the midpoint of the odd path between x and y (such a path must exist because n is odd).

Table 2: Calculation of the meeting point.

Let $m = k - f$ be the number of remaining tokens.
Algorithm MEET1
1. Release the token at the starting node.
2. Choose a direction and walk once around the ring.
3. Compute the sequence of intertoken distances $s_1, \ldots, s_h$.
   If a token is present at this node: $\mathcal{S} = d_1, \ldots, d_m = s_1, \ldots, s_h$.
   If no token is present at this node: $\mathcal{S} = d_1, \ldots, d_m = s_1 + s_h, s_2, \ldots, s_{h-1}$.
4. Move to $Meetingpoint(S)$.

Table 3: Failure at the beginning, n known.

Theorem 1 *When the agents have $O(k \log n)$ memory,n is known, $\gcd(k', n) = 1$ 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 less than $2n$ time units.*

Notice that, for this algorithm to work, the system does not need to be synchronous; in fact, each agent could independently make its calculations and move to the rendezvous point even if the ring is *asynchronous*.

3.2 k known, n unknown

More interesting is the case when n is unknown. As in the previous case, the agents will walk around the ring long enough to be able to calculate a sequence of intertoken distances; this sequence must be such that it covers the entire ring and contains an asymmetry that can be exploited by the agents to agree on a meeting point.

Unlike the previous case, however, since n is not known, the circular sequences computed by the agents using solely knowledge of k, are not necessarily identical. In fact, each agent, to construct its sequence, will fully traverse the ring the same (unknown) number of times but, depending on the starting point and on the number of failures, it will also traverse a portion of the ring which is in general different for different agents.

Although the sequences are not the same, we will show that the agents can still identify an unique point where to meet by calculating $3k$ intertoken distances. The Algorithm Meet2 is described in Table 4.

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:

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

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

Algorithm MEET2

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 $S^R = Q^q + d_\gamma, \ldots, d_1$ where $\gamma < |Q|$.
6. Move to $Meetingpoint(Q)$

Table 4: Failure at the beginning, k known.

Theorem 2 *When the agents have $O(k \log n)$ memory, k is known, $\gcd(k', n) = 1$ 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 2.

Let $m = k - f$ be the number of tokens that do not fail. Let $A = \delta_1, \ldots, \delta_m$ be the sequence of the m intertoken distances that exist after the f tokens have failed; clearly $\sum_{i=1}^{m} \delta_i = n$. Let $S(a)$ be the sequence of $3k$ intertoken distances calculated by a given agent a in step 3 of Algorithm MEET2. Let $S^R(a)$ be the reverse of $S(a)$. For all agents, the $3k$ intertoken distances are of the form

$$S^R(a) = A^\rho + d_\gamma, \ldots, d_1 = (\delta_1, \ldots, \delta_m)^\rho + d_\gamma, \ldots, d_1. \tag{2}$$

where A^ρ is the concatenation of ρ copies of the aperiodic subsequence A, $+$ is the concatenation operator, and $d_\gamma, \ldots, d_1$ is a subsequence such that $\gamma < m$. Thus, there exists at least one aperiodic subsequence, namely A, that satisfies equation 1. Note that A is aperiodic as it has k or fewer intertoken distances, and by assumption $\gcd(k', n) = 1$ for all $k' \leq k$.

If A is the shortest subsequence that satisfies equation 1, the agents discover A in step 5 of Algorithm MEET2. Otherwise, the agents discover a shorter aperiodic subsequence, Q, that satisfies equation 1.

The subsequence discovered in step 5 of Algorithm MEET2 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 agents identify the same rendezvous node in the remaining steps of Algorithm MEET2 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 takes $O(kn)$ time, so the overall time requirement is $O(kn)$. This completes the proof of Theorem 2. ■

When the tokens only fail upon release, Algorithm MEET2 also solves the mobile agent rendezvous problem when the ring is *asynchronous*. Once an agent has calculated S^R, it can identify the smallest aperiodic sequence that satisfies equation 2. Since this sequence is unique, the agent can then identify the unique rendezvous node, walk there, and wait until all k agent arrive. The algorithm does not depend on timing but rather on the agents' ability to count and then identify the rendezvous node.

4 Rendezvous When Tokens Can Fail At Any Time

We now consider the situation when token failures can occur at any time. In this case, for the algorithms to work the system must be synchronous.

The idea is still to make use of the intertoken distances to agree on a meeting point. In this case, however, the problem is made much more complicated by the fact that these distances vary unpredictably during the algorithm. To cope with that, our algorithm works in rounds, in each round, the agents compute some intertoken distances and try to meet in a node. In a round, however, the rendezvous may fail because the agents might have computed different intertoken distances. In such a case, only groups of agents (the ones that have computed the same intertoken distances) might meet (in a "false" rendezvous point).

To make the algorithms work, the rounds must be synchronized so that the agents start them within some bounded time interval (if not simultaneously). In other words, an agent arriving at its meeting point will have to know how long to wait before declaring that point a false rendezvous

point and correctly start the next round, or before realizing that such a point is a true meeting point.

4.1 n known k unknown

First of all notice that since k is unknown, a group of agents finding themselves on the same node cannot determine whether rendezvous has been accomplished simply by counting how many they are. A different strategy will have to be used.

A round is composed of three distinct steps. In *step*1 an agent travels around the ring to compute the intertoken distances; it then identifies the lexicographical maximum string. In *step*2 the agent moves to the computed meeting point (let us assume that it takes t time units to go there), and waits $n - t$ units to synchronize with the others for the next step. Notice that, in general, the agent cannot understand at this point if rendezvous is accomplished just by counting the other agents on the meeting node. As we will prove, if this is the third time that the same string has been calculated, the agent can be sure that this is a true meeting point. In this case, in fact, the agent knows that everybody else has seen this string at least in the previous and in the current rounds and is now in its own meeting point. *Step*3 is used for notifying all agents that indeed the meeting point is the correct one. The agents who know, go around the ring. The agents who do not know, wait for n units: if nothing happens, they go to the next round; if a notifying agent arrives, they terminate the algorithm.

The Algorithm is described in Table 5.

Algorithm MEET3
1. Release token.
2. Set $S_1 = S_2 = S_3 = \emptyset$
3. Set $r = 0$, choose a direction and begin walking.
4. Travel for n time units computing the intertoken distances $S = (s_1, \ldots, s_h)$.
   If a token is present at this node: $\mathcal{S} = d_1, \ldots, d_m = s_1, \ldots, s_h$.
   If no token is present at this node: $\mathcal{S} = d_1, \ldots, d_m = s_1 + s_h, \ldots, s_{h-1}$.
5. Set $t = 0$.
6. Walk to $Meetingpoint(S)$ and increment t for each node traveled.
7. Wait $n - t$ clock ticks.
8. Set $S_1 = S_2$; $S_2 = S_3$; $S_3 = S$.
9. If $S_1 = S_2 = S_3$
10. become(notifying), go around the ring, and then terminate
11. Else wait n units
12. If a notifying agents comes, terminate /* Rendezvous has occurred. */
13. Else set $r = r + 1$ and repeat from step 3.

Table 5: Failure at any time, n known.

Lemma 1 *When an agent sees the same string for the third time, all the other agents have seen it at least twice (in the previous and in the current round).*

Sketch of Proof of Lemma 1.
Let a be an agent that sees the same string three times. The first time (round i) it goes to the

meeting point after going around the ring, the second time (round $i+1$) it is already in the meeting point after going around the ring and it does not move. Agent a sees again the same string in round $i+2$, this means that during the previous round nothing has changed, regardless of the starting point, everybody has seen the same string and has met (without knowing it) in the same meeting point. As a consequence, everybody has started round $i+2$ from the same starting point and, thus, has computed again the same string. ■

Theorem 3 *When the agents have $O(k \log n)$ memory, know n, $\gcd(k', n) = 1$ holds for all $k' \leq k$, $f \leq (k-1)$ tokens fail, and tokens can fail upon release or later, then the mobile agent rendezvous problem can be solved in $O(kn)$ time.*

Proof of Theorem 3.
By lemma 1 we know that when an agent sees the same string for the third time the meeting point was the true rendezvous point. All the agents met at that point and after n time units everybody will terminate. In each round, every agent has spent n time units to compute the intertoken distances, n additional time units to move to the meeting point and wait for the restart, n time units for the termination check. Thus, all agents are perfectly synchronized and start each round simultaneously. After, at most $k+2$ rounds rendezvous is accomplished and the total time is then $O(kn)$. ■

4.2 k known n unknown

Lack of knowledge of n makes the situation more complicated; the main problem is to achieve synchronization.

In the following algorithm, if more than one but fewer than k agents meet on a given node, they *merge* and act as one agent for the remainder of the algorithm. Merged agents will be perceived by other agents with their multiplicity; thus, if an agent meets a group of merged agents, it will "see" how many they are. In this algorithm, before starting the next round, every group that has met in a false meeting point, merge and behave like one agent.

Since the agents, not knowing n, cannot travel around the ring, they will travel trying to guess the ring size at each round. Initially they compute k intertoken distances and they guess n to be the sum of those distances (they use this guessed value for synchronization purposes); at each subsequent round they compute one less intertoken distance and they change their guess of n.

The algorithm is designed in such a way that 1) it is guaranteed that agents are always in at most one round apart, 2) after at most $k-1$ rounds rendezvous is accomplished. The Algorithm is described in Table 6.

The following three lemmata are used in the proof of Theorem 4. Lemma 2 proves that the agents are always less than a round apart and thus an agent need only wait $\frac{3\hat{n}}{2}$ clock ticks for any agents that saw the same view.

Lemma 2 *Let a be the first agent to finish estimating n in round r, where $0 \leq r \leq k-1$, and let τ denote the time at which agent a finishes the estimation. All other agents will either finish round $r-1$ or merge with a on or before time τ.*

Proof of Lemma 2.

Base case: $r = 1$.
Let τ_1 be the time that a finishes estimating n in round $r = 1$ and let b be an arbitrarily chosen

Algorithm MEET4
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 agent, merge with it.
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. If S is periodic, wait $2\hat{n}$ steps, set $r = r + 1$, and repeat from step 3.
8. Calculate S_{LMR}, the lexicographically maximum rotation of S.
9. Set $t = 0$.
10. Walk to the node that starts S_{LMR} and increment t for each node traveled.
11. Wait $2\hat{n}$ - t clock ticks.
12. If there are k agents or their merged equivalent on the current node, stop.
/* Rendezvous has occurred. */
13. Else if there are $1 < v < k$ agents on the current node, then merge.
14. Set $r = r + 1$ and repeat from step 3.

Table 6: Failure at any time, k known.

mobile agent other than a. If b does not finish round $r = 0$ by time τ_1, then it is either walking around the ring or waiting at a given node. If b is still walking at time τ_1, then it walked at least n nodes while a was waiting in round $r = 0$, and thus merged with a. Suppose that b was waiting at time τ_1. If b began waiting at time $t \leq \tau_1 - n$, then a met and merged with b as the former estimated n in round $r = 1$. If b began waiting at time $t > \tau_1 - n$, however, then b met and merged with a while the latter waited in round $r = 0$.

Inductive Hypothesis: $r = m$.

$r = m + 1$.

Let τ_{m+1} be the time that a finishes estimating n in round $r = m + 1$.

Case 1: Suppose that a was also the first agent to finish estimating n in the previous round $r = m$. Let τ_m denote the time at which a finished that estimation. The inductive hypothesis implies that all other agents finished round $r = m - 1$ no later than τ_m and thus start round $r = m$ no later than $\tau_m + 1$.

Let b be an arbitrarily chosen mobile agent other than a. Consider the case where b did not finish round $r = m$ by time τ_{m+1}. If b was still walking, then it walked at least n nodes

while a waited in round $r = m$ and thus had met and merged with a. If b was waiting at time τ_{m+1}, then two cases arise. If b started waiting at time $t \leq \tau_{m+1} - n$, then a met b as the former was estimating n in round $r = m + 1$.

Otherwise, b started waiting at time $t > \tau_{m+1} - n$ and thus b met and merged with a while the latter waited in round $r =m$.

Case 2: Suppose that a was not the first agent to finish estimating n in the previous round $r = m$. Let b denote the agent that was first to finish estimating n in round $r = m$. If b has not finished round $r = m$ by time τ_{m+1}, then it is still waiting in round $r = m$. Since b started waiting before a started round $r = m + 1$, a met b while the former estimated n in round $r = m + 1$. This completes the proof of Lemma 2. ■

Lemma 3 *Mobile agents that see the same sequence S, up to a rotation, of intertoken distances are in the same round. If S is aperiodic, then these agents will rendezvous at the end the given round. However, if S is periodic, then the agents cannot rendezvous during this round and must proceed to the next round.*

Proof of Lemma 3.
Let $\mathcal{MA}(\mathcal{S})$ denote the set of agents that see the same sequence S, up to a rotation, of intertoken distances. Since $r = k$ - $|S|$,

all members of $\mathcal{MA}(\mathcal{S})$ are in the same round.

If S is aperiodic, the agents in $\mathcal{MA}(\mathcal{S})$ can identify the lexicographically maximum rotation of S and thus identify the same rendezvous node. Let t_0 denote the time that the first agent in $\mathcal{MA}(\mathcal{S})$ finishes calculating S. Lemma 2 implies that the remaining agents that see the same view will begin calculating S no later than t_0. The first agent will wait at the rendezvous node from no later than $t_0 + \frac{\hat{n}}{2}$ until exactly $t_0 + 2\hat{n}$, where $\hat{n} = \sum_{i=1}^{|S|} d_i$.

The remaining agents that see rotations of S arrive at the rendezvous node no later than $t_0 + \frac{3\hat{n}}{2}$, and thus all agents that saw rotations of S will rendezvous by the end of the round.

If S is periodic, however, then the agents in $\mathcal{MA}(\mathcal{S})$ cannot identify the lexicographically maximum rotation of S and thus cannot identify a unique rendezvous node. When the agents in $\mathcal{MA}(\mathcal{S})$ calculated $\hat{n}$ in round r, they were assuming that r tokens had failed.

If r tokens had failed and thus the estimate for n was correct, S would have been aperiodic, since $\gcd(k', n) = 1$ for all $k' \leq k$. Thus agents that see the same periodic view are executing a round where r is strictly less than the number of tokens that have failed. The agents must proceed to the next round. This completes the proof of Lemma 3. ■

Lemma 4 *If at most f tokens have failed, then the agents will not execute round $r = f + 1$ in Algorithm MEET4.*

Proof of Lemma 4.
The proof proceeds by induction on the number of failed tokens.

$f = 1$:
If the token fails after the agents have estimated n in round $r = 0$, then rendezvous occurs in round $r = 0$. If the token fails while the agents are estimating n in round $r = 0$ such that at least two agents have different estimates for n, then rendezvous does not occur in round $r = 0$. In the next round, $r = 1$, no other tokens fail and thus rendezvous occurs in round $r = 1$. The agents do not execute round $r = m + 1 = 2$.

Inductive Hypothesis: $f = m$.

$f = m + 1$:
Suppose that the agents execute round $r = m + 1$. The inductive hypothesis implies that more than m tokens failed. If the $(m + 1)$th token failed before the agents start estimating n in round $r = m+1$, then the agents have the same estimate for n and rendezvous occurs. If the $m+1$th token failed while the agents are estimating n such that at least two agents have different estimates for n, then rendezvous does not occur in round $r = m+1$. In the next round, $r = m+2$, no other tokens fail and thus the agents rendezvous. The agents do not execute round $r = m + 2$. This completes the proof of Lemma 4. ■

Theorem 4 *When the agents have $O(k \log n)$ memory, know k $\gcd(k', n) = 1$ holds for all $k' \leq k$, $f \leq (k - 1)$ tokens fail, and tokens can fail upon release or later, then the mobile agent rendezvous problem can be solved in $O(k^2 n)$ time.*

Proof of Theorem 4.
Let $f \leq k-1$ be the number of tokens that actually fail. Lemma 4 implies that no agent will execute more than $f+1$ rounds of Algorithm MEET4. Suppose that rendezvous has not occurred by the end of round $r = f$. Let a^* denote the first agent that begins round $r = f+1$ and let t_0 denote the time that a^* starts round $r = f+1$. Since f tokens have failed, a^*'s estimate for n in round $r = f+1$ will be correct, i.e., $\hat{n} = n$. The remaining agents see the same aperiodic sequence, up to a rotation, of intertoken distances as a^* and thus Lemma 3 implies that rendezvous occurs at the end of round f.

The number of token that fail, f, is at most $k-1$ so, as mentioned above, at most k rounds of Algorithm MEET4 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. As a result, the time required is $O(k^2 n)$. Because at most k intertoken distances are calculated and the maximum intertoken distance is $n-1$, the memory required is $O(k \log n)$. This completes the proof of Theorem 4. ■

5 Conclusion

The effect of token failure on the time required to 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.

Acknowledgements

This research is supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) and MITACS (Mathematics of Information Technology and Complex Systems) grants, and by an OGS (Ontario Graduate Scholarship).

References

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

[2] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. "Election and rendezvous in fully anonymous systems with sense of direction". In *10th Symposium on Structural Information and Communication Complexity* (SIROCCO '03), 2003.

[3] V. Baston and S. Gal. "Rendezvous search when marks are left at the starting points". *Naval Research Logistics*, 38, pp. 469-494, 1991.

[4] A. Dessmark, P. Fraigniaud, and A. Pelc. "Deterministic rendezvous in graphs". In *European Symposium on Algorithms* (ESA '03), pp. 184-195, 2003.

[5] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. "Multiple agents rendezvous in a ring in spite of a black hole". In *Symposium on Principles of Distributed Systems* (OPODIS '03), LNCS, 2003.

[6] E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. "Mobile agent rendezvous problem in the ring". In *International Conference on Distributed Computing Systems* (ICDCS '03), pp. 592-599, 2003.

[7] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. "Multiple mobile agent rendezvous in the ring". In *Latin American Conference of Theoretical Informatics* (LATIN '04), 2004.

[8] C. Sawchuk. *Mobile Agent Rendezvous in the Ring.* Ph.D Thesis, Carleton University, January 2004.

[9] X. Yu and M. Yung. "Agent rendezvous: A dynamic symmetry-breaking problem". In *International Colloquium on Automata, Languages, and Programming* (ICALP '96), LNCS 1099, pp. 610-621, 1996.