Probabilistic Solitude Detection on Rings of Known Size

Karl Abrahamson*
Andrew Adler†
Lisa Higham*
David Kirkpatrick*

Technical Report 86-26
December, 1986

*Department of Computer Science
†Department of Mathematics

Abstract

Upper and lower bounds that match to within a constant factor are found for the expected bit complexity of a problem on asynchronous unidirectional rings of known size n, for algorithms that must reach a correct conclusion with probability at least $1 - \epsilon$ for some small preassigned $\epsilon \geq 0$. The problem is for a nonempty set of *contenders* to determine whether there is precisely one contender. If distributive termination is required, the expected bit complexity is $\Theta(n \min(\log \nu(n) + \sqrt{\log\log(1/\epsilon)}, \sqrt{\log n}, \log\log(1/\epsilon)))$, where $\nu(n)$ is the least nondivisor of n. For nondistributive termination, $\sqrt{\log\log(1/\epsilon)}$ and $\sqrt{\log n}$ are replaced by $\log\log\log(1/\epsilon)$ and $\log\log n$ respectively. The lower bounds hold even for probabilistic algorithms that exhibit some nondeterministic features.

1 Introduction

Our primary objective is to gain a deeper understanding of the nature of distributed computation. Our choice of an asynchronous ring as a network topology to study is motivated not just by the simplicity of this configuration but by the fact that it exhibits interesting features of distributed computation that can be expected to show up in other more complex topologies as well.

Typical of issues that need to be addressed in solving problems in any distributed setting are the following:

- **Knowledge:** What do individual processors know about the global properties (size, organization) of the network? Are processors distinguishable? To what extent can this knowledge help in solving a specific problem?
- **Type of Algorithms:** Is the desired algorithm deterministic, randomized or probabilistic? How does the type of algorithm affect the complexity of the solution? We use the terms *randomized* and *error-free* to describe an algorithm which may rely on coin tosses, but which never produces an incorrect result. The terms *probabilistic* and *error-tolerant* describe an algorithm which gives incorrect results with low but positive probability.
- **Type of termination:** Must algorithms terminate distributively, or is nondistributive termination acceptable? (An algorithm terminates distributively if each processor, after reaching a conclusion, will not revoke its conclusion upon the receipt of subsequent messages. This is the usual requirement of an algorithm. A process executing a nondistributively terminating algorithm can never know that the algorithm has terminated.) What price do we pay for insisting on distributive termination?
- **Measure of complexity:** What are appropriate things to measure in analysing the complexity of a specific problem? Messages, communication bits, synchronous time?

One of a few fundamental problems that have been well studied in a distributed computation setting is that of electing a leader — that is, causing a unique processor among a specified set of contending processors to enter a distinguished (leadership) state. Leader election on an asynchronous ring can be viewed (cf. [1,4]) as the composition of two even more fundamental problems: attrition and solitude detection. The attrition problem is that of reducing a set of contenders to exactly one contender. Solitude detection is the problem of confirming that attrition is complete. In fact, both attrition and solitude detection deterministically reduce to leader election in simultaneous $O(n)$ bits and time, on rings of size n, which further justifies the view of leader election as attrition plus solitude detection.

This paper is concerned with the solitude detection problem on a unidirectional asynchronous ring. Let a nonempty set of processes on a ring be distinguished as *contenders*. The solitude detection problem is for every process to determine whether or not there is only one contender. A solitude detection algorithm is initiated simultaneously by all of the contenders.

Deterministic [3,6,7], randomized [1,4] and probabilistic [2,5] solutions to problems related to solitude detection, including leader election and maximum finding, have been considered earlier. Of particular relevance to the present paper are earlier results on the bit complexity of solitude verification, a subproblem of solitude detection.

Solitude verification requires a contender to conclude that it is alone precisely when it is alone. There are no other requirements. For example, a solitude verification algorithm can fail to terminate when there are two or more contenders. The bit complexity of solitude verification is defined to be its complexity when there is exactly one contender. The relation between solitude verification, solitude detection and leader election is discussed in the concluding section of the paper.

Prior results for solitude verification concern the case of processors which do not have exact knowledge of the ring size n. In the cases summarized below, as in the cases considered in this paper, processors do not necessarily have distinct identities. All of the stated upper bounds for solitude verification in fact apply to the stronger problem of solitude detection.

The results below involve a parameter ϵ. Throughout this paper, ϵ represents a real number satisfying $0 < \epsilon \leq 1/4$.

First suppose that processors have no knowledge of n. Then solitude verification is impossible with any distributively terminating probabilistic algorithm that is correct with probability bounded away from zero. However, there exists a nondistributively terminating probabilistic algorithm that solves solitude detection with probability of error at most ϵ using $O(n \log(1/\epsilon))$ expected bits of communication. Furthermore, any such algorithm requires $\Omega(n \log(1/\epsilon))$ expected bits of communication [2].

Now suppose that all processors know two integers N and Δ such that $N - \Delta \leq n \leq N$. If $\Delta \geq N/2$ then solitude verification is impossible for any randomized (i.e. error-free) algorithm even with nondistributive termination. However, there exist distributively terminating probabilistic algorithms that solve solitude detection with probability of error at most ϵ using $O(n\sqrt{\log(N/n)} + n \log(1/\epsilon))$ expected bits of communication. Furthermore, when $n \leq N/2$ any such algorithm requires

$\Omega(n\sqrt{\log(N/n)} + n\log(1/\epsilon))$ expected bits [2].

If $\Delta < N/2$ then there exist distributively terminating deterministic solitude detection algorithms that communicate $O(n \log n)$ bits in the worst case. Also, any even nondistributively terminating, nondeterministic solitude verification algorithm (i.e. a certification of solitude) must use $\Omega(n \log \Delta)$ bits in the best case [1].

If $\Delta \leq N/k$ where $k > 2$ then there exist distributively terminating probabilistic algorithms that solve solitude detection with probability of error at most ϵ using $O(n \log\log(1/\epsilon))$ expected bits. Furthermore, all such algorithms require $\Omega(\min(n \log \Delta,\ n \log\log(1/\epsilon)))$ expected bits [2].

None of the results above adequately address the (realistic) case where Δ is very small, specifically $\Delta = 0$ (i.e. n is known exactly). It is to this case that we devote our attention in the remainder of this paper. The main results can be summarized briefly as follows. As above, all complexity bounds apply to the case of exactly one contender. In all cases, the bit complexity is $\Theta(n)$ when there are two or more contenders.

The bit complexity of detecting solitude without error using a distributively terminating algorithm is $\Theta(n\sqrt{\log n})$. When nondistributive termination is permitted, the bit complexity of error-free solitude detection drops to $\Theta(n \log\log n)$.

The expected bit complexity of verifying solitude with probability of error at most ϵ is $\Theta(n \min(\log \nu(n) + \sqrt{\log\log(1/\epsilon)}, \sqrt{\log n}, \log\log(1/\epsilon)))$ if distributive termination is desired, and $\Theta(n \min(\log \nu(n) + \log\log\log(1/\epsilon), \log\log n, \log\log(1/\epsilon)))$ if nondistributive termination is acceptable, where $\nu(n)$ denotes the smallest positive nondivisor of n.

Thus the asymptotic complexity of solitude detection for both distributively and nondistributively terminating algorithms is known, to within constant factors, for all values of the relevant parameters n and ϵ. In fact, our lower bound results are proved on a model of computation that permits much more powerful algorithms than those used to achieve the corresponding upper bound results. A more detailed discussion contrasting our upper and lower bounds is presented in the concluding section of the paper.

2 Solitude Verification Algorithms

This section describes solitude detection algorithms for four conditions, depending on whether a randomized (error-free) or probabilistic (error-tolerant) algorithm is desired, and whether the algorithm must terminate distributively or not. In the probabilistic case, the algorithm errs (with low probability) only when there are two or more contenders. Since all of the algorithms are similar, they are all presented as a single parameterized algorithm, consisting of five stages. Not all stages are executed in all conditions. Stage 4 is only executed if distributive termination is required. Stage 5 is only executed by probabilistic algorithms.

The algorithm has an integer parameter $l \geq 4$, which is adjusted according to type of algorithm desired. Let $\nu(n)$ be the smallest positive nondivisor of n. Then $\nu(n)$ is a prime power; say, $\nu(n) = p^s$. Let t be the smallest integer such that $p^t \geq l$. Let $m = p^{s+t}$. Notice that m does not divide n and $m > l$.

The algorithm is described for a contender. Non-contenders cooperate, as described. If a contender receives evidence that it is not alone before the algorithm is finished then it sends one of two kinds of alarm. A loud alarm is sent if the evidence is conclusive. Having sent a loud alarm, a contender aborts the algorithm, and concludes that it is not alone. A soft alarm is sent during a probabilistic algorithm when a contender has received strong but not conclusive evidence that it is not alone. After sending a soft alarm, a contender waits to receive an alarm, then proceeds directly to stage 5.

Alarms are forwarded by non-contenders. A contender which receives an alarm while expecting some other type of message aborts what it is doing, and sends a loud alarm. Each contender sends at most one alarm of each kind. A contender which has finished the algorithm without sending or receiving a loud alarm concludes that it is alone.

Stage 1: (The purpose of this stage is to keep the complexity low when there are many contenders.) Toss an unbiased coin $K = \lceil 2 \log m \rceil$ times, and send the outcomes, one at a time, to the right. After sending each coin toss, receive a toss from the left. If the toss received does not match what was just sent, send a loud alarm.

Stage 2: (This stage will generate an alarm if there are i contenders, where $2 \leq i \leq l$.) Send a counter, initially 1, to the right. The counter is incremented mod m^2 by each non-contender, and propagates to the next contender. Receive a count from the left. If the count is not congruent to $n \pmod{m^2}$, send a loud alarm.

Stage 3: (This stage generates an alarm within every sequence of l distinct contenders.) Inform the contender to the right whether the distance separating it from yourself is greater than n/l.

(a) For an error-free algorithm, send a counter, initially one, to the right. Each non-contender increments the counter, until the counter reaches a value greater than n/l. At that point, the message "long" is propagated to the next contender. Receive a message from the left. If the message is not "long," send a loud alarm.

(b) For an error-tolerant algorithm with error probability at most ϵ, let $\lambda = 4\lceil \log(4l/\epsilon) \rceil$. If $n \le \min(200 \log(1/\epsilon),\ 11\lambda l + l))$, then use a deterministic counter, as in (a). Otherwise, start a counter, initially zero. Before forwarding the counter to the right, the contender and each non-contender increments the counter with probability $\lambda l/n$. When the counter reaches a value greater than 2λ, the message "long" is propagated to the next contender. Receive a message from the left. If the message is a counter, not "long," let c be the value of the counter, send a soft alarm, wait for a soft alarm to arrive, then go to stage 5.

Stage 4: (This stage is only executed if distributive termination is desired. It serves to flush alarms.) Alternately send and receive l "ok" messages. Of course, if an alarm arrives, forward a loud alarm.

Stage 5: (This stage is only executed by processors which sent a soft alarm. It eliminates the possibility of error when there is a single contender.) Let c be the count received in stage 3, and let $\hat{g} = cn/(\lambda l)$ be an estimate of the distance to the nearest contender to the left. Let $K = \left\lceil \log(1/\epsilon) + \sqrt{\log(n/\hat{g})}\, \right\rceil + 2$, and alternately send and receive up to K coin tosses, as in stage 1. Send a loud alarm as soon as the toss received does not match that just sent.

Correctness

Error-free case. Alarms are sent only when a contender has conclusive evidence that it is not alone. Hence, when there is a single contender, the algorithm answers correctly.

Suppose there are $i \ge 2$ contenders. Shortly we will show that, if an alarm is sent by any contender, then all contenders conclude "not alone". So suppose no alarm is sent. Let $g_1, \ldots, g_i$ be the lengths of the gaps separating the contenders. Then $g_1 + \cdots + g_i = n$. Since no alarms are sent at stage 2, it must be the case that $g_j \equiv n \pmod{m^2}$ for $j = 1, \ldots, i$. Let r be the remainder when n is divided by m^2. Then $ir \equiv r \pmod{m^2}$, from which it follows that $m^2 \mid (i-1)r$. But $m \nmid n$, so $m \nmid r$, and, since m is a prime power, $m \mid (i-1)$. Hence, $i > m > l$, and one of the gaps g_j must be less than n/l. So some processor will detect a short gap at stage 3, and will send an alarm.

If nondistributive termination is sufficient, then it is clearly sufficient for an alarm to be sent. For distributive termination, each contender should receive an alarm before it reaches a conclusion. But notice that, for any l consecutive contenders (assuming more than l contenders), one of the gaps to the left of one of those contenders is shorter than n/l. So each contender will surely receive an alarm by the time it reaches the end of stage 4.

Error-tolerant case. Again, a loud alarm is only sent when a processor receives conclusive evidence that it is not alone, so it suffices to consider the case where there are $i \geq 2$ contenders. There are two ways to err: either some processor reaches the end of stage 4 without having sent any kind of alarm, or every processor sends an alarm, and some processor fails to send a loud alarm at stage 5. We show that the probability of each kind of error occurring is at most $\epsilon/2$.

Consider the first kind of error. Let $k = \lfloor n/l \rfloor > 11\lambda$. The mean value of the stage 3 count at distance k from the contender which started the count is at most λ. The probability that the count reaches $2\lambda+1$ before the k^{th} coin toss is well into the tail of the binomial distribution, and is less than $\binom{k}{2\lambda}\left(\frac{\lambda}{k}\right)^{2\lambda}\left(1-\frac{\lambda}{k}\right)^{k-2\lambda}$. But $\binom{k}{2\lambda} < k^{2\lambda}(2\lambda)^{-2\lambda}e^{2\lambda}$, so the probability that a gap of length at most k appears to be "long" is less than $e^{2\lambda}2^{-2\lambda}e^{-\lambda}\left(\frac{k}{k-\lambda}\right)^{2\lambda} < .83^{\lambda} < \epsilon/(4l)$.

Say that a gap is short if its length is less than n/l. It suffices to estimate the probability that *some* short gap is counted as long. That probability is clearly maximum when there are fewer than $2l$ contenders, since combining two very short gaps to produce one short gap can only increase the probability of many increments in some short gap. So the probability that some short gap is counted as long is less than $2l(\epsilon/4l) = \epsilon/2$.

Now consider stage 5. Loud alarms can only help, so suppose each contender has sent a soft alarm, and executes stage 5. Stage 5 is a version of an algorithm described in [2]. That algorithm assumes that the j^{th} contender, for $j = 1, \ldots, i$, has an estimate $\hat{g}_j$ of the gap g_j between it and the nearest contender to its left, and that $E(\hat{g}_j) \leq g_j$. Suppose we use, for $\hat{g}_j$, the value of $\hat{g}$ obtained by the j^{th} contender at stage 5. Assuming the j^{th} contender reaches stage 5, $E(\hat{g}_j) = g_j$. Theorem 3 of [2] guarantees that, when there are $i > 1$ contenders, then with probability at least $1-\epsilon/2$ every contender will send an alarm.

Complexity

When there is one contender. The bit complexity of stages one and two together is $O(n \log m) = O(n \log \nu(n) + n \log l)$.

In the error-free case, stage 3 costs $O((n/l)\log n)$ bits. For a nondistributively terminating algorithm, a choice of $l = \max(4, \lceil \log n \rceil)$ yields a total complexity of $O(n \log \nu(n) + n \log\log n) = O(n \log\log n)$.

For the distributively terminating error-free algorithm, stage 4 costs an additional $O(nl)$ bits. Choosing $l = \max(4, \lceil \sqrt{\log n}\, \rceil)$ gives a total complexity of $O(n\sqrt{\log n})$ bits.

Now consider the error-tolerant version. We can assume that $n > 200\log(1/\epsilon)$, since otherwise an error-free algorithm is used, and has the desired complexity. When there is just one contender, the mean number of times its stage 3 counter is incremented is $\lambda l \geq 4\lambda$. The probability that a single contender sends a soft alarm at stage 3 (i.e. there are at most 2λ increments) is less than $2\binom{n}{2\lambda}\left(\frac{4\lambda}{n}\right)^{2\lambda} \times \left(1-\frac{4\lambda}{n}\right)^{n-2\lambda} < 2(0.66)^{\lambda} < \epsilon$. Moreover, given that a single contender does send a soft alarm, its estimate $\hat{g}$ of the gap to its left (which is just n) is almost surely very close to $2n/l$. So $E(\sqrt{\log(n/\hat{g})}) < \sqrt{\log l}$. Given the choices of l which will be made below, the total expected number of bits sent in stage 5 when there is a single contender is $O(\epsilon n \log(1/\epsilon)) = O(n)$.

The expected cost of stage 3 is $O((n/l)\log\lambda) = O((n/l)\log\log l + (n/l)\log\log(1/\epsilon))$. Choosing $l = \max(4, \lceil\log\log(1/\epsilon)\rceil)$ gives a total expected cost for stages 1, 2, 3 and 5 of $O(n\log\nu(n) + n\log\log\log(1/\epsilon))$ bits, so that is the cost of achieving nondistributive termination. When stage 4 is included, choose $l = \max(4, \left\lceil\sqrt{\log\log(1/\epsilon)}\right\rceil)$. The complexity of achieving distributive termination is then $O\left(n\log\nu(n) + n\sqrt{\log\log(1/\epsilon)}\right)$ expected bits.

When there are two or more contenders. The following analysis applies to both the error-free and error-tolerant versions. Let k be 2, 3 or 4. The probability that a given contender executes stage k is at most $1/m^2$, due to the coin tosses in stage 1. The probability that a given non-contender participates in stage k is also at most $1/m^2$, since a non-contender participates only if the nearest contender to its left does. So the total expected cost of stages 2, 3 and 4 is $O\left((n/m^2)(\log m + \log\gamma + l)\right)$, where γ is either λ or n/l. But in all cases, that is $O(n)$. Stages 1 and 5 require $O(n)$ expected bits, since each contender sends $O(1)$ expected bits before it sends an alarm, and each non-contender sends as many bits as the nearest contender to its left. So the total cost is $O(n)$ expected bits, when there are two or more contenders.

When ϵ is very small, our error-tolerant algorithm can be more expensive than our error-free algorithm. Obviously, the cheaper algorithm should be used. In fact, there is a third algorithm, given in [2], which in some circumstances can be more efficient than either the error-free or error-tolerant algorithm given here. That algorithm does not require exact knowledge of n, only that n is known to within a factor of $c < 2$, and uses $O(n\log\log(1/\epsilon))$ expected bits. By choosing the best of the three algorithms, we find that the expected bit complexity of solitude detection with confidence $1-\epsilon$, when there is one initiator, is $\Theta(n\min(\log\nu(n) + \sqrt{\log\log(1/\epsilon)}, \sqrt{\log n}, \log\log(1/\epsilon)))$ for distributive termination and $\Theta(n\min(\log\nu(n) + \log\log\log(1/\epsilon), \log\log n, \log\log(1/\epsilon)))$ for nondistributive termination.

The $O(n\log\log(1/\epsilon))$ bit algorithm of [2] has poor complexity when there are two or more contenders. But it can be modified along the lines of the algorithm given here to use only $O(n)$ bits when there are two or more contenders. Thus, regardless of which of the three algorithms is chosen, the bit complexity is $O(n)$ when there are two or more initiators.

Given that our upper bound is a combination of three rather different algorithms, it would not be surprising, in the absence of further results, to find that a fourth algorithm performed better than all three of ours, at least sometimes. We now turn to lower bounds, and show that our upper bound is in fact optimal to within a constant factor. In some sense, there are just three essentially different algorithms, or three ideas to be exploited, each, for distributive and nondistributive termination.

3 A Model for Proving Lower Bounds

Our objective is to study the inherent bit complexity of distributed probabilistic algorithms that verify solitude on unidirectional rings with a high probability of correctness. A distributed algorithm is an assignment of processes to processors. So it suffices to model computations induced by a cyclic sequence of processes. Our model is based on the following non-restrictive assumptions:

i) messages are self-delimiting;

ii) only contenders may initiate; and

iii) communication is message driven, and only one message is sent in response to receipt of a message.

The algorithms of section 2 are consistent with these assumptions. Furthermore, in the algorithms every contender (respectively, non-contender) executes the same probabilistic process. In our model we relax this assumption and allow the possibility that individual contenders (respectively, non-contenders) may execute different processes. It is assumed, however, that the correctness of the distributed algorithms does not rely on either the distinctness or the distribution of individual processes. For a further discussion of the differences between the models assumed for the algorithms and for the lower bounds, see section 5.

A *message* is an element of $M = \{0,1\}^* \cdot \Box$. The symbol $\Box$ is called the end of message marker. If m is a message we denote by $\|m\|$ the length of m, including the end marker. A *communication event* is an element of $M \cup \{\triangle\}$. The null event $\triangle$ denotes the absence of an input or an output message and should be distinguished from the empty message '$\Box$'.

Any even length sequence of communication events, $C = (e_1, e_2, \ldots, e_{2t})$ describes a *communication history*. The subsequence $(e_1, e_3, \ldots e_{2t-1})$ is called the *input history* of C and subsequence $(e_2, e_4, \ldots, e_{2t})$ is called the *output history.* Communication history C is said to be *reduced* if C does not begin or end with a pair of null events.

If $h \in (M \cup \{\triangle\})^*$ is any history we denote by $|h|$ the length of h and $\|h\|$ the *cost* of h, that is, the sum of the lengths of all messages in h.

A (*probabilistic*) *process* is modelled by an assignment of probabilities to communication histories. Let π be a process and C be a reduced communication history with input history h and output history h' (where, necessarily, $|h| = |h'|$). We denote by $h\,\{\pi\}\,h'$ the event that process π produces output history h', given that it has input history h. An event $h\,\{\pi\}\,h'$ with nonzero probability is called a *computation* of π with input history h. The possibility that a process can produce more or fewer messages than it reads is accounted for by padding one or both of h and h' with null events.

We will only rarely have occasion to deal with unreduced communication histories. Identical probabilities are assigned to each of the events $h\,\{\pi\}\,h'$, $\triangle h\,\{\pi\}\,\triangle h'$ and $h\triangle\,\{\pi\}\,h'\triangle$.

Since we have restricted our attention to message driven processes, we can assume that input histories are elements of $\triangle^*\, M^*$. In fact, our main focus will be on what we call *normal* communication histories, in which the entire communication history is an element of $\triangle^*\, M^*\, \triangle^*$.

If h is a history let $h_{(i)}$ denote the length i prefix of h. By the sequential nature of communication, we have,

Property 3.1: For all histories h and h' and all $i \geq 1$, $\Pr(h\,\{\pi\}\,h') \leq \Pr(h_{(i)}\,\{\pi\}\,h'_{(i)})$.

We say that process π is an *initiator* if $\sum_{m \in M} \Pr(\triangle\,\{\pi\}\,m) > 0$. All other processes are *non-initiators.* An initiator is just a process which has nonzero probability of sending a message before it receives one. We presume that the contenders in any ring are just the initiators.

If π_1 and π_2 are processes then their composition, denoted $\pi_1\,\pi_2$, is the process π satisfying

$$\Pr(h\,\{\pi\}\,h') = \sum_{h''} \Pr(h\,\{\pi_1\}\,h'') \cdot \Pr(h''\,\{\pi_2\}\,h').$$

Thus, $\pi_1\,\pi_2$ is obtained by identifying the output of π_1 with the input of π_2. If at most one of π_1 and π_2 are initiators and π_1 and π_2 realize only normal communication histories then so does their composition.

The notion of process composition suffices to allow us to study the behaviour of a sequence of processes *on a line* as a function of the behaviours of the individual processes. Our objective, however, is to study the behaviour of processes on a ring. Informally, process π is on a ring if its output is fed back into its input. Fortunately, the essential properties of process rings are reflected by properties of associated process lines. The relationship between rings and lines is given by the following:

Property 3.2: Let π and π' be any processes.

i) If $\Pr(\triangle h \{\pi\} h \triangle) = p$ then, with probability p, computations of π on a ring result in π producing output history $h\triangle$.

ii) If $\Pr(\triangle^t \{\pi\} h) = p$ then, with probability at least p computations of $\pi' \pi$ on a ring result in π producing as output a (possibly infinite) string of messages with prefix h.

The placement of nulls in property 3.2 is important. For example, it is quite possible that, when given input history h, process π produces output history h with positive probability, that is, $\Pr(h \{\pi\} h) > 0$. But when π's output is fed back into its input, π produces no messages; π is deadlocked, waiting for itself.

If $\pi_1, \ldots, \pi_t$ is a sequence of processes, let $\pi_{i,j}$ denote the composition $\pi_i \cdots \pi_j$, for $1 \leq i \leq j \leq t$. It is convenient to view a sequences of processes $\pi_1, \ldots, \pi_t$ as a single process, namely the composition $\pi_{1,t}$. By abuse of notation, a sequence is frequently identified with its composition. The real difference between the two is that, although $\pi_{1,t}$ is a single process, with communication cost assigned only to its input and its output, the sequence $\pi_1, \ldots, \pi_t$ has communication cost assigned to each link from π_i to π_{i+1}, for $1 \leq i < t$.

The notion of a computation can be extended to sequences of processes. A sequence $h_0, \ldots, h_t$ of histories describes a computation of the process line $\pi_{1,t}$ which is equivalent to the conjunction of the independent events $h_i \{\pi_{i+1}\} h_{i+1}$, for $0 \leq i < t$, in the appropriate product space. The *cost* of such a computation is given by $\sum_{i=1}^{t} \|h_i\|$. Note that h_0 does not contribute to the cost. If $\pi_1, \ldots, \pi_t$ are processes (or sequences of processes), let $h_0 \{\pi_1\} h_1 \cdots \{\pi_t\} h_t$ denote the event described by sequence $h_0, \ldots, h_t$.

We distinguish a subset $M_a \subseteq M$ called *accepting messages*, and a subset $M_r \subseteq M$ called *rejecting* messages. A history is an *accepting history* (respectively, *rejecting history*) iff its last message is an accepting message (respectively, rejecting message). A computation described by $h_0, \ldots, h_t$ *asserts solitude* if each of the histories h_i is an accepting history, and *asserts non-solitude* if each of the h_i is a rejecting history. A process π is said to *terminate distributively* if π never outputs another message after having output an accepting message.

A sequence $\pi_{1,t}$ of processes is said to *assert solitude* (respectively, *non-solitude*) on a ring with probability p if computations of $\pi_{1,t}$ on a ring assert solitude (respectively, non-solitude) with probability p.

The following lemma shows that if the expected cost of computations of a one-initiator sequence $\pi_{1,t}$ on a ring is bounded then inexpensive computations of the form $\triangle h\,\{\rho_{1,t}\}\,h\triangle$ occur with reasonably high probability, where $\rho_{1,t}$ is some cyclic permutation of $\pi_{1,t}$ and h is some fixed element of M^*.

Lemma 3.3: Let $\pi_1, \ldots, \pi_t$ be any sequence of processes with exactly one initiator. Suppose that $\pi_{1,t}$ asserts solitude on a ring with probability at least $1-\epsilon$. Suppose also that the expected cost of computations of $\pi_{1,t}$ that assert solitude is at most μt. Then there exists an integer i, where $1 \leq i \leq t$, and an accepting history h with $\|h\| \leq 4\mu$ such that

$$\Pr(\triangle h\,\{\pi_{i+1,t}\,\pi_{1,i}\}\,h\triangle \;\&\; A) \geq (1-\epsilon)2^{-(4\mu+3)}$$

where A denotes the event that a computation of the process line $\pi_{i+1,t}\,\pi_{1,i}$ with input history $\triangle h$ has total cost at most $2\mu t$.

Proof: We know that with probability at least $1-\epsilon$ each of the processes π_1, $\ldots$, π_t outputs an accepting history. Since the expected cost of accepting computations is at most μt, the probability that an arbitrary computation of $\pi_{1,t}$ accepts and communicates fewer than $2\mu t$ bits is at least $(1-\epsilon)/2$.

Let e_i denote the expected number of bits in the output history of process π_i, over all accepting computations of $\pi_{1,t}$ with costs of at most $2\mu t$ bits. For some i, $e_i \leq 2\mu$, and hence with probability at least $(1-\epsilon)/4$, π_i has an accepting output history with no more than 4μ bits and the entire computation has cost at most $2\mu t$. But there are fewer than $2^{4\mu+1}$ distinct histories with at most 4μ bits and hence, with probability at least $(1-\epsilon)2^{-(4\mu+3)}$, π_i outputs some fixed accepting history h, where $\|h\| \leq 4\mu$ and the entire (accepting) computation has cost at most $2\mu t$. Thus $\Pr(\triangle h\,\{\pi_{i+1,t}\,\pi_{1,i}\}\,h\triangle \;\&\; A) \geq (1-\epsilon)2^{-(4\mu+3)}$. ∎

Let χ be a number (which will be given a specific value whenever necessary) called the *cheapness threshold.* A computation of an arbitrary sequence of processes is said to be *cheap* if it has total cost at most χ. Let $h\,\langle\pi_{1,t}\rangle\,h'$ denote the event $h\,\{\pi_{1,t}\}\,h'$ & A, where A is the event that the computation of processor sequence $\pi_{1,t}$ with input history h is cheap. Similarly, $h_0\,\langle\pi_1\rangle\,h_1 \cdots \langle\pi_t\rangle\,h_t$ denotes the event $h_0\,\{\pi_1\}\,h_1 \cdots \{\pi_t\}\,h_t$ & A.

At the heart of our lower bound proofs is the observation that a sequence of histories of sufficiently small total cost must contain the same history twice. The following lemma refines that observation to a probabilistic setting, and provides information about the separation between the repetitions.

Lemma 3.4: Let $\pi_{1,t}$ be any sequence of processes. Let σ and τ be positive integers satisfying

(a) $\tau \geq 36^2$,

(b) $24\chi < t \log \tau$, and

(c) $t > \tau\sigma$.

Let h^0 and h^1 be any histories. Then there exist integers i, j and m, where $1 < i < j \leq t$ and $1 \leq m < \tau$, and a history h^* such that

i) $j - i = m\sigma$ and

ii) $\Pr(h^0 \langle \pi_{1,i-1} \rangle h^* \langle \pi_{i,j-1} \rangle h^* \langle \pi_{j,t} \rangle h^1) \geq \tau^{-1} \Pr(h^0 \langle \pi_{1,t} \rangle h^1)$.

Proof: Suppose without loss of generality that the event $C = h^0 \langle \pi_{1,t} \rangle h^1$ has nonzero probability. For $1 \leq i < t$, let e_i be the expected cost of the output history of π_i, conditional on C. That is, $e_i = \sum_{h_i} \|h_i\| \cdot \Pr(h^0 \{\pi_{1,i}\} h_i \{\pi_{i+1,t}\} h^1 \mid C)$. Let $\delta = \log \tau$.

If $e_i < \delta/8$, say that link i is *cheap*. If $\|h\| < \delta/4$, say that history h is *short*. Suppose that link i is cheap and let h_i^* be the short history which maximizes $\Pr(h^0 \{\pi_{1,i}\} h_i^* \{\pi_{i+1,t}\} h^1 \mid C)$. Since there are fewer than $2^{\delta/4+1} = 2\tau^{1/4}$ short histories, it follows that $\Pr(h^0 \{\pi_{1,i}\} h_i^* \{\pi_{i+1,t}\} h^1 \mid C) \geq \frac{1}{4\tau^{1/4}}$, since otherwise $e_i > (\delta/4)(1 - \frac{2\tau^{1/4}}{4\tau^{1/4}}) = \delta/8$, contradicting the cheapness of link i.

For $1 \leq j < t - (\tau - 1)\sigma$, let $B_j = \{j + k\sigma \ : \ 0 \leq k < \tau\}$. Choose a j such that at least 1/3 of the τ members of B_j are cheap links. Such a j must exist, since otherwise at least 2/3 of at least $\tau\sigma\lfloor (t-1)/\tau\sigma \rfloor \geq t/2$ links are not cheap, contradicting the assumption that $\sum_{i=1}^{t} e_i \leq \chi < t\delta/24$.

Again, because there are at most $2\tau^{1/4}$ short histories, at least $w = \lceil \tau^{3/4}/6 \rceil$ of the cheap members k of B_j have identical h_k^*. Let $i_1, \ldots, i_w$ be w such members, and let h^* denote the common history. Let D_s denote the event $h^0 \{\pi_{1,i_s-1}\} h^* \{\pi_{i_s,t}\} h^1$, for $1 \leq s \leq w$. By the inclusion-exclusion principle,

$$\sum_{r<s} \Pr(D_r \,\&\, D_s \mid C) \geq \left(\sum_s \Pr(D_s \mid C) \right) - 1.$$

Since $Pr(D_s \mid C) \geq \frac{1}{4\tau^{1/4}}$, there must exist r and s such that

$$\begin{aligned} \Pr(D_r \,\&\, D_s \mid C) &\geq \frac{\frac{w}{4\tau^{1/4}} - 1}{\binom{w}{2}} \\ &\geq \frac{1}{\tau}, \qquad \text{for } \tau \geq 36^2. \end{aligned}$$

Thus $\Pr(D_r \,\&\, D_s \,\&\, C) \geq \tau^{-1}\Pr(C)$. So, it suffices to choose $i = i_r$ and $j = i_s$. ■

Lemma 3.4 only locates repeated histories. The following property and lemma use repeated histories to relate lines of different sizes.

Property 3.5: Let $\pi_1, \ldots, \pi_t$ be a sequence of processes and let $1 < i < j \leq t$. Let h^0, h^1 and h^* be histories. Let $p = \Pr(h^0 \langle \pi_{1,i-1} \rangle h^* \langle \pi_{i,j-1} \rangle h^* \langle \pi_{j,t} \rangle h^1)$. Then $\Pr(h^0 \langle \pi_{1,i-1} \rangle h^* \langle \pi_{j,t} \rangle h^1) \geq p$ and $\Pr(h^* \langle \pi_{i,j-1} \rangle h^*) \geq p$.

Proof: It suffices to observe that any computation satisfying $h^0 \langle \pi_{1,i-1} \rangle h^* \langle \pi_{i,j-1} \rangle h^* \langle \pi_{j,t} \rangle h^1$ includes disjoint subcomputations satisfying $h^0 \langle \pi_{1,i-1} \rangle h^*$, $h^* \langle \pi_{i,j-1} \rangle h^*$ and $h^* \langle \pi_{j,t} \rangle h^1$. ■

Lemma 3.6: Let $\pi_1, \ldots, \pi_t$ be any sequence of processes with one distinguished process, say π_i. Let σ, τ and t_0 be positive integers satisfying

(a) $\tau \geq 36^2$,

(b) $t_0 \leq t$,

(c) $48\chi \leq t_0 \log \tau$, and

(d) $t_0 \geq 2\tau\sigma$.

Let h^0 and h^1 be any histories. Then there exists a length r (non-contiguous) subsequence $\pi'_{1,r} = \pi_{i_1} \cdots \pi_{i_r}$ of $\pi_{1,t}$, including π_i, such that

i) $t_0 - \tau\sigma < r < t_0$,

ii) $r \equiv t \pmod{\sigma}$, and

iii) $\Pr(h^0 \langle \pi'_{1,r} \rangle h^1) \geq \tau^{-1-2\tau \ln \frac{t-t_0}{\sigma}} \Pr(h^0 \langle \pi_{1,t} \rangle h^1)$.

Proof: Suppose that conditions a) through d) are satisfied. The idea is to apply Lemma 3.4 and property 3.5 repeatedly, each time eliminating some processes between repeated histories, being careful not to eliminate π_i. At a given step, the sequence remaining has some length t', where $t_0 \leq t' \leq t$, and $t' \equiv t \pmod{\sigma}$. Let $x = \pi_{j_1} \cdots \pi_{j_{t'}}$ be the current remaining sequence, which is a (non-contiguous) subsequence of $\pi_{1,t}$ including π_i. Let σ^* be the largest multiple of σ which does not exceed $\sigma + \frac{t'-t_0}{2\tau}$.

Sequence x has a contiguous subsequence of length at least $\lfloor t'/2 \rfloor$ not containing π_i. Assume without loss of generality that $x = uv$, where π_i occurs in sequence u, and v has length at least $\lfloor t'/2 \rfloor$. Treating u as a single process, sequence uv has length at least $\lceil (t'+1)/2 \rceil$. By Lemma 3.4, with σ^* playing the role of σ and

$\lceil (t'+1)/2 \rceil$ playing the role of t, and by property 3.5, there exists a length s (non-contiguous) subsequence $y = \pi_{k_1} \cdots \pi_{k_s}$ of x, including π_i, such that $s = t' - m\sigma^*$, $1 \leq m < \tau$, and $\Pr(h^0 \langle y \rangle h^1) \geq \tau^{-1} \Pr(h^0 \langle x \rangle h^1)$. We call the act of constructing a subsequence y of x satisfying the above properties *shrinking* the sequence x. By starting with $\pi_{1,t}$ and shrinking some number g times in this fashion, we eventually construct a sequence $\pi'_{1,r}$ where

i) $t_0 - \tau\sigma < r < t_0$,

ii) $r \equiv t \pmod{\sigma}$, and

iii) $\Pr(h^0 \langle \pi'_{1,r} \rangle h^1) \geq \tau^{-g} \Pr(h^0 \langle \pi_{1,t} \rangle h^1)$

Each time we shrink, except for the final shrink, the value of $t' - t_0$ decreases by a factor of $1 - 1/(2\tau)$. Since the last shrink is by at least σ processes, it follows that $g \leq 1 + \hat{g}$ where $\hat{g}$ is the smallest integer such that $(t - t_0)\left(1 - \frac{1}{2\tau}\right)^{\hat{g}} < \sigma$. Taking logarithms to the base e, and using the fact that $\ln\left(1 - \frac{1}{2\tau}\right) < -\frac{1}{2\tau}$, we get that $\hat{g} \leq 2\tau \ln \frac{t-t_0}{\sigma}$. ∎

4 Lower Bounds

Let $\pi_1, \ldots, \pi_n$ be a sequence of processes with exactly one initiating process π_1. Say that $\pi_{1,n}$ *requires k bits for confidence* $1 - \epsilon$ if at least one of the following is true.

i) the expected cost of computations of $\pi_{1,n}$ on a ring that assert solitude is at least k.

ii) computations of $\pi_{1,n}$ on a ring fail to assert solitude with probability greater than ϵ.

iii) there is a sequence of processes $\pi'_{1,n}$, including two or more initiators, such that a) for every j there is an i such that $\pi'_j = \pi_i$, and b) terminating computations of $\pi'_{1,n}$ on a ring fail to assert non-solitude with probability greater than ϵ. (Thus, with probability greater than ϵ, some processor erroneously concludes that there is only one contender.) We call $\pi'_{1,n}$ a fooling sequence for $\pi_{1,n}$.

The following theorems, together with our upper bounds, completely characterize the bit complexity of solitude verification to within a constant factor in all of the cases we have considered. In the interest of ease of presentation, we make little effort to establish strong constant factors.

Theorem 4.1: There is a constant $c > 0$ such that if $\pi_{1,n}$ is any sequence of distributively terminating processes with exactly one initiating process π_1, then $\pi_{1,n}$ requires $cn \min\left(\sqrt{\log n}, \sqrt{\log\log(1/\epsilon)}\right)$ bits for confidence $1 - \epsilon$.

Proof: The theorem is trivially true when n or $1/\epsilon$ are of moderate size, even for reasonable values of c, so assume that n is very large and ϵ is very small. Suppose that computations of $\pi_{1,n}$ on a ring assert solitude with probability at least $1-\epsilon$. Furthermore, assume that the expected cost of computations of $\pi_{1,n}$ on a ring that assert solitude is μn where $1 \leq \mu < (1/21)\min\left(\sqrt{\log n}, \sqrt{\log\log(1/\epsilon)}\right)$. We construct a fooling sequence $\pi'_{1,n}$ for $\pi_{1,n}$.

By Lemma 3.3, there exists an integer i, $1 \leq i \leq n$, a cyclic permutation $\rho_{1,n} = \pi_{i+1,n}\pi_{1,i}$ of $\pi_{1,n}$ and an accepting history h with $\|h\| \leq 4\mu$ such that $\Pr(\triangle h\,\langle\rho_{1,n}\rangle\, h\triangle) \geq (1-\epsilon)2^{-4\mu-3}$, where the cheapness threshold is $\chi = 2\mu n$. Since $\mu < \sqrt{\log n}/21$, we have $n > 8\mu\tau$. Apply Lemma 3.6, choosing $t_0 = \lceil n/(4\mu)\rceil$, $\tau = 2^{384\mu^2}$ and $\sigma = \lfloor n/(8\mu\tau)\rfloor$. The conditions of the lemma are easily checked. It follows that there exists subsequence $z = \pi_{i_1}\cdots\pi_{i_r}$ of $\rho_{1,n}$, including π_1, such that $r < t_0$ and

$$\begin{aligned}
\Pr(\triangle h\,\langle z\rangle\, h\triangle) &\geq \tau^{-1-2\tau\ln(16\mu\tau)}\Pr(\triangle h\,\langle\rho_{1,n}\rangle\, h\triangle)\\
&\geq (1-\epsilon)\tau^{-1-2\tau\ln(16\mu\tau)}2^{-4\mu-3}\\
&\geq \tau^{-3\tau\ln\tau}
\end{aligned}$$

Now treat sequence z as a single process, and imagine splicing together $t = \max(|h|, 2) \leq 4\mu$ copies of z. By property 3.1,

$$\begin{aligned}
\Pr(\triangle^t\,\{z^t\}\, h) &\geq \prod_{j=0}^{t-1}\Pr(\triangle^{t-j}h_{(j)}\,\{z\}\,\triangle^{t-j-1}h_{(j+1)})\\
&\geq \tau^{-3t\tau\ln\tau}\\
&> \epsilon
\end{aligned}$$

since $\mu < (1/21)\sqrt{\log\log(1/\epsilon)}$.

By property 3.2, it follows that with probability at least ϵ computations of any ring R of n processes including the sequence z^t result in some process producing a string of messages with prefix h. But since processes terminate distributively and h is an accepting history, all such computations terminate with some processor concluding that it is alone. ∎

Corollary 4.2: Any error-free distributively terminating solitude verification algorithm uses $\Omega(n\sqrt{\log n})$ bits on every ring with a single contender.

Theorem 4.3: There is a constant $c > 0$ such that if $\pi_{1,n}$ is any sequence of (nondistributively terminating) processes with exactly one initiating process π_1, then $\pi_{1,n}$ requires $cn \min(\log \nu(n), \log\log(1/\epsilon))$, bits for confidence $1-\epsilon$.

Proof: Let the expected cost of accepting computations of $\pi_{1,n}$ on a ring be μn, where $1 \leq \mu < (1/100)\log(\nu(n)-1)$. We will suppose that $\mu < (1/100)\log M$, where $M < \min(\nu(n), \log^{1/4}(1/\epsilon))$, and for simplicity M is an integer. Since $\mu \geq 1$, M is quite large, and n is very large. We further presume that ϵ is small. Suppose $\pi_{1,n}$ asserts solitude with probability at least $1-\epsilon$. We construct a fooling sequence $\pi'_{1,n}$ for $\pi_{1,n}$.

By Lemma 3.3, there exists a cyclic permutation $\rho_{1,n} = \pi_{i,n}\pi_{1,i-1}$ and an accepting history h such that $\Pr(\Delta h \langle \rho_{1,n} \rangle h \Delta) \geq (1-\epsilon)/(8M^{1/25})$, where the cheapness threshold is $\chi = 2\mu n$.

Temporarily think of the shorter of the sequences $\pi_{i,1}$ and $\pi_{1,i-1}$ as a single process. Let $a = n/\lambda(M)$, where $\lambda(M)$ is the least common multiple of the positive integers not exceeding M. Since $M < \nu(n)$, a is a positive integer. Apply Lemma 3.4, with $t = n$, $\tau = \lfloor M/2 \rfloor$ and $\sigma = a$. Then there exist k and l, with $k < l$, without loss of generality in the interval $[1, i-1]$, an accepting history h^* and an integer m, with $1 \leq m < \tau$, such that $l - k = ma$ and $\Pr(\Delta h \langle \pi_{i,n}\, \pi_{1,k-1} \rangle\, h^* \langle \pi_{k,l-1} \rangle$ $h^* \langle \pi_{l,i-1} \rangle\, h\Delta) \geq p = 1/(8M^{26/25})$. By property 3.5, $\Pr(h^* \{\pi_{k,l-1}\}\, h^*) \geq p$ and $\Pr(h^* \langle \pi_{k,n}\, \pi_{1,l-1} \rangle\, h^*) \geq p$.

Let $D = l - k = ma$. Since $\tau \leq M/2$, D divides $n/2$. Now apply Lemma 3.6 to $\pi_{k,n}\pi_{1,l-1}$, using $t_0 = \lceil n/2 \rceil$, $\sigma = D$ and $\tau = M^2$, with π_1 as the distinguished process. We obtain a subsequence $z = \pi_{i_1} \cdots \pi_{i_r}$ of π having the following properties:

(a) there is exactly one occurrence of π_1 in z;

(b) $r \leq n/2$;

(c) $n/2 \equiv r \pmod D$;

(d) $n/2 - r < M^2 D$ and

(e) $\Pr(h^* \{z\}\, h^*) \geq q = p\left(\frac{1}{M^2}\right)^{1+2M^2 \ln \frac{\lambda(M)}{2}}$.

Since $\pi_{k,l-1}$ has length D, there is an integer $j < M^2$ such that $jD + r = n/2$ and $\Pr(h^* \{\pi^j_{k,l-1}\, z\}\, h^*) \geq q$. Let the fooling ring be $\pi'_{1,n} = (\pi^j_{k,l-1}\, z)^2$. This ring has two occurrences of the initiating process π_1, and incorrectly asserts solitude (since h^* is an accepting history) with probability at least q^2. By the prime number theorem, $\ln \lambda(M)$ is asymptotically equal to M, and in particular $\ln(\lambda(M)/2) < 2M$ for large M. So $q^2 > \epsilon$ for $\log M < (1/4)\log\log(1/\epsilon)$. ∎

Theorem 4.4: There is a constant $c > 0$ such that if $\pi_{1,n}$ is any sequence of (nondistributively terminating) processes with exactly one initiating process π_1, then $\pi_{1,n}$ requires $cn \min(\log\log n, \log\log\log(1/\epsilon))$ bits for confidence $1-\epsilon$.

Proof: This proof proceeds along the same lines as the previous two, but is more involved. Again, assume that n is very large and ϵ is very small. Suppose that the expected cost of accepting computations of $\pi_{1,n}$ on a ring is μn, where $1 \le \mu \le (1/200)\min(\log\log n, \log\log\log(1/\epsilon))$. Let $\lambda \doteq \lceil 2\mu \rceil$, and let the cheapness threshold be $\chi = \lambda n$.

By Lemma 3.3, there is a cyclic permutation $\rho_{1,n}$ of $\pi_{1,n}$ and an accepting history h such that $\Pr(\triangle h \langle \rho_{1,n} \rangle h \triangle) \ge (1-\epsilon)2^{-2\lambda-3}$. We construct a fooling sequence $\pi'_{1,n}$ for $\pi_{1,n}$. Construction of $\pi'_{1,n}$ is broken into several steps.

Step 1: The sequence $\pi'_{1,n}$ must have length exactly n. To aid in adjusting the size of the constructed sequence, we require two sequences of processes which can be spliced into another sequence. One is for making large adjustments in size, the other for fine tuning.

Step1A: (Find the fine tuning sequence θ.) By Lemma 3.4, with parameters $t = n$, $\sigma = 1$ and $\tau = \alpha = 2^{25\lambda}$, there exist integers i_1 and j_1 and a history h_1 such that $1 \le j_1 - i_1 < \alpha$ and

$$\begin{aligned} \Pr(\triangle h \langle \rho_{1,i_1-1} \rangle h_1 \langle \rho_{i,j-1} \rangle h_1 \langle \rho_{j_1,n} \rangle h \triangle) &\ge \alpha^{-1} \Pr(h \langle \pi_{1,n} \rangle h) \\ &\ge \alpha^{-1} 2^{-2\lambda-3} \\ &\ge \alpha^{-2}. \end{aligned}$$

Let $d_1 = j_1 - i_1$ and let $\theta = \rho_{i_1,j_1-1}$. By property 3.5, $\Pr(h_1 \{\theta\} h_1) \ge \alpha^{-2}$. Also, $\Pr(\triangle h \langle \pi_{1,i_1-1} \rangle h_1 \langle \pi_{i,n} \rangle h \triangle) \ge \alpha^{-2}$.

Step 1B: (Find the large adjustment sequence ϕ.) The length of ϕ is crucial to the argument. We will be collapsing $\rho_{1,n}$ down to a sequence of size close to $n' = \lfloor n/(d_1+1) \rfloor$. Sequence ϕ will aid in collapsing to a precisely chosen size in a small number of individual shrinking operations.

Let $M = (\alpha^2!)^4$. Notice that m divides M for all $1 \le m < \alpha^2$. Also, it is easily shown that $n > 2M\alpha$. We will apply Lemma 3.4 to the larger of ρ_{1,i_1-1} and $\rho_{i_1,n}$, which we can assume without loss of generality is $\rho_{i_1,n}$, treating the shorter one as a single process. The parameters used are $t = \lceil n/2 \rceil$, $\tau = \alpha^2$ and $\sigma = \sigma_2 = d_1 \left\lfloor \frac{n-n'}{d_1 M} \right\rfloor \ge d_1$. The lemma provides integers i_2 and j_2 and a history h_2, such that $\Pr(h_2 \{\rho_{i_2,j_2-1}\} h_2) \ge \alpha^{-4}$ and $\Pr(\triangle h \langle \pi_{1,i_1-1} \rangle h_1 \langle \pi_{i_1,i_2-1} \rangle h_2 \langle \pi_{i_2,n} \rangle h \triangle) \ge \alpha^{-4}$.

Let $d_2 = j_2 - i_2$ and let $\phi = \rho_{i_2,j_2-1}$. From Lemma 3.4, $d_2 = m\sigma_2$ where m is an integer, $1 \le m < \alpha^2$. Since m divides M, d_2 divides $M\sigma_2$.

Step 2: Now we do the collapsing. Let $x = \rho_{1,i_1-1}$, $y = \rho_{i_1,i_2-1}$ and $z = \rho_{i_2,n}$. From step 1 we have $\Pr(\triangle h \langle x \rangle h_1 \langle y \rangle h_2 \langle z \rangle h\triangle) \geq \alpha^{-4}$. The goal now is to shrink each of x, y and z until each of their lengths is at most $n'/3$. So apply Lemma 3.6 to each of x, y and z, in each case treating the rest of the sequence as a single process (or two processes in the case of y). Where appropriate, let the initiator be the distinguished process. Use parameters $t_0 = \lceil n'/3 \rceil$, $\sigma = d_2$, and $\tau = \tau_3 = \alpha^{6\alpha}$. The condition $48\chi \leq t_0 \log \tau$ of the lemma follows easily from the fact that $\alpha \leq n/25$. Condition $t_0 \geq 2\tau_3 d_2$ follows from the fact that $M \geq 12\alpha^{6\alpha+3}$.

The total number of individual shrinking operations applied cannot exceed n/d_2, since at least d_2 processors are removed in each shrink. Let x', y' and z' be the resulting sequences, and let $n'' = |x'\,y'\,z'|$. From Lemma 3.6,

(a) $n' - \tau_3 d_2 < n'' \leq n'$

(b) $n'' \equiv n \pmod{d_2}$ and

(c) $\Pr(\triangle h \{x'\} h_1 \{y'\} h_2 \{z'\} h\triangle) \geq \tau_3^{-n/d_2} \alpha^{-4} \geq \alpha^{-\alpha^{9\alpha^2}}$ since $n/d_2 \leq 2M < \alpha^{8\alpha^2}$.

Step 3: Notice that $n' - Md_1 \leq n - M\sigma_2 \leq n'$. The goal now is to pad the sequence y', obtaining a sequence y'' such that $n - Md_2 \leq |x'\,y''\,z'| \leq n'$.

If $n - Md_2 \leq n''$, then simply let $y'' = y'$. Otherwise, $n'' \equiv n \pmod{d_2}$, and $M\sigma_2 \equiv 0 \pmod{d_2}$, so $n'' \equiv n - M\sigma_2 \pmod{d_2}$. So there must be a positive integer $k < \tau_3$ such that $n'' + kd_2 = n - M\sigma_2$. Let $y'' = y'\,\phi^k$. Since $\Pr(h_2 \{\phi\} h_2) \geq \alpha^{-4}$, we have $\Pr(\triangle h \{x'\} h_1 \{y''\,z'\} h\triangle) > \alpha^{-\alpha^{9\alpha^2}} \alpha^{-4\tau_3} > \alpha^{-\alpha^{10\alpha^2}}$.

Step 4: Let $w = (x'y''\,z')^{d_1}$ be a sequence of d_1 copies of $x'\,y''\,z'$, and let $u = x'\,y''\,z'\,w$. From step 3 we have

(a) $n - Md_1(d_1+1) < |u| \leq n$,

(b) $|u| \equiv n \pmod{d_1}$, since $|x'\,y''\,z'| \equiv n \pmod{d_2}$ and d_1 divides d_2.

(c) $\Pr(\triangle^2 h \{x'\} h_1 \{y''\,z'\} \triangle h \triangle \{w\} h\triangle^2) > \alpha^{-\alpha^{10\alpha^2+1}}$.

Step 5: We are ready to do the fine tuning. Let $k < M(d_1+1) \leq M\alpha$ be a nonnegative integer such that $|u| + kd_1 = n$. Let the fooling sequence $\pi'_{1,n}$ be $x'\,\theta^k\,y''\,z'\,w$. Then $\Pr(\triangle h \{\pi_{1,n}\} h\triangle) > \alpha^{-\alpha^{10\alpha^2+1}} \alpha^{-2\alpha} > \epsilon$. Since h is an accepting history, and $\pi'_{1,n}$ has $d_1 + 1 > 1$ initiators, $\pi'_{1,n}$ wrongly asserts solitude with probability greater than ϵ. ∎

Corollary 4.5: Any error-free (non-distributively terminating) solitude verification algorithm uses $\Omega(n \log\log n)$ bits on every ring with a single contender.

5 Conclusions

We have presented upper and lower bounds that match to within a constant factor for the bit complexity of solitude detection on a ring of known size. A significant observation is that the bounds depend upon the specific requirements of the algorithm — whether it is error-free or error-tolerant, and whether it is distributively or nondistributively terminating.

It is perhaps not surprising that number theoretic properties of the ring size n influence the bit complexity of solitude detection when n is known exactly. Let $\nu(n)$ be the smallest nondivisor or n. For nondistributive termination with confidence $1-\epsilon$, the expected complexity of solitude detection is the minimum of that of three algorithms; specifically $\Theta(n\min(\log\nu(n)+\log\log\log(1/\epsilon),\ \log\log n,\ \log\log(1/\epsilon)))$ bits. Distributive termination can be achieved by appending a termination detecting phase to a nondistributively terminating algorithm. For distributive termination the bit complexity of solitude detection is $\Theta(n\min(\log\nu(n)+\sqrt{\log\log(1/\epsilon)},\ \sqrt{\log n},\ \log\log(1/\epsilon)))$.

When no error can be tolerated, the above results simplify to $\Theta(n\sqrt{\log n})$ bits for distributive termination and $\Theta(n\log\log n)$ bits for nondistributive termination.

It is interesting to note that the inherent bit complexity of solitude detection when n is known depends upon whether or not distributive termination is required. This contrasts with the case when n is only known to within a factor of two, where relaxing the requirements of the solution from distributive to nondistributive termination does not reduce the bit complexity of solitude detection [1,2].

It is commonplace to find probabilistic algorithms with a factor of $\log(1/\epsilon)$ in their complexity bounds. (Typically, the complexity measure is time, as opposed to bits.) Indeed, one solitude verification algorithm has bit complexity $O(n\log(1/\epsilon))$ [2]. Such an algorithm generally works by applying a method which gives a certificate of a given answer with some constant probability, and repeating it enough times to ensure a low probability of error. The probabilistic algorithms given here are more subtle than that. They reduce the dependence on the error probability ϵ well below a factor of $\log(1/\epsilon)$.

There are important distinctions between the models of computation used for the algorithms presented here and the models assumed for the lower bounds.

The first is a distinction between types of probabilistic algorithms. Our upper bounds are established in a weak system which can be termed *deterministic/probabilistic*. Every processor of a given type (contender or noncontender) runs the same algorithm. State changes occur either deterministically, as the result of receiving a message, or probabilistically, as the result of a coin toss. Two processors which are in the same state choose their next message randomly from the same distribution.

In contrast, the lower bounds are proved for a ***nondeterministic/probabilistic*** model. Conceptually, state changes are made as the result either of a coin toss or of receiving a message or of a nondeterministic choice. As is common practice, we have modelled nondeterminism as a single choice at the beginning of the algorithm, where the algorithm decides which deterministic/probabilistic process to assign to each processor. The requirement is only that, no matter which nondeterministic choices are made, the algorithm must reach an erroneous conclusion with low probability. Complexity is measured for the best possible assignment of processes to processors.

In the error-free case, nondeterminism subsumes randomization, and our lower bounds are really purely nondeterministic. But it can be advantageous for an error-tolerant algorithm to make use of randomization in addition to nondeterminism, since *all* nondeterministic choices must lead to a correct answer, while it suffices for *most* random choices to do so.

There are a number of ways to view nondeterminism as it is used here. Technically, our model requires nondeterministic choices to be made at the start of an algorithm. But such an algorithm can simulate decisions made on the fly by initially guessing a function from internal states to guesses. So, in fact, our lower bounds apply to algorithms which make nondeterministic choices on the fly, and they apply to the best case complexity.

One might, in general, hope for an algorithm which works on all rings, and which works especially efficiently on a ring in which processors are labeled in a particular way. The fact that our lower bounds apply to best case precludes such algorithms for solitude detection.

In addition to a naturally pleasing generality, there is an advantage to having nondeterministic lower bounds. It was pointed out in the introduction that solitude detection reduces to leader election in $O(n)$ bits. In the case of distributively terminating algorithms, the reduction is an obvious one. But reduction to a nondistributively terminating algorithm can be subtle. The problem is that an individual processor cannot know that the given nondistributively terminating leader election algorithm has terminated, and that it is time to proceed with solitude verification. If a processor begins the solitude verification phase prematurely, then it may cause extra messages to be sent.

The solution is elegant. Since the lower bounds on solitude verification hold for nondeterministic algorithms, they carry across nondeterministic reductions. Simply let each processor guess when leader election is finished. At that point, the leader checks for more than one contender. In the best case, there will be no premature "leaders".

The second important way in which the upper and lower bounds differ is in the type of error permitted. Our algorithms only admit one-sided error. That is, when there is exactly one contender the algorithms *always* confirm its solitude. The only allowable error is that of leading one or more of several contenders to an erroneous conclusion of solitude. The lower bounds, on the other hand, permit two-sided error, with probability of at most ϵ of any kind of error.

Thirdly, as pointed out in the introduction, although our algorithms all solve solitude detection, the lower bounds apply to the weaker problem of solitude verification. So algorithms that might have high communication complexity or might deadlock or even fail to terminate when there are two or more contenders, still require the same expected amount of communication when there is one contender. In fact if only solitude verification is needed, then step one of the algorithm in section 2 can be omitted. One result is a completely deterministic distributively terminating error-free solitude verification algorithm with bit complexity $O(n\sqrt{\log n})$ bits. The lower bound implies that even nondeterministic solutions must have at least this complexity.

Finally the solitude detection algorithms all use randomization that is restricted to selecting one of only two possible messages. The lower bounds indicate that more elaborate uses of randomization do not help to reduce the complexity of solitude detection.

References

[1] K. Abrahamson, A. Adler, R. Gelbart, L. Higham, and D. Kirkpatrick. *The Bit Complexity of Randomized Leader Election on a Ring.* Technical Report 86-3, University of British Columbia, Vancouver B.C., 1986.

[2] K. Abrahamson, A. Adler, L. Higham, and D. Kirkpatrick. Probabilistic solitude verification on a ring. In *Proc. 5th Annual ACM Symp. on Principles of Distributed Computing*, pages 161–173, 1986.

[3] D. Dolev, M. Klawe, and M. Rodeh. An $O(n \log n)$ unidirectional distributed algorithm for extrema finding on a circle. *J. Algorithms*, 3(3):245–260, 1982.

[4] A. Itai and M. Rodeh. Symmetry breaking in distributed networks. In *Proc. 22nd Annual Symp. on Foundations of Comput. Sci.*, pages 150–158, 1981.

[5] J. Pachl. *A Lower Bound for Probabilistic Distributed Algorithms.* Technical Report CS-85-25, University of Waterloo, Waterloo, Ontario, 1985.

[6] J. Pachl, E. Korach, and D. Rotem. Lower bounds for distributed maximum finding. *J. Assoc. Comput. Mach.*, 31(4):905–918, 1984.

[7] G. Peterson. An $O(n \log n)$ algorithm for the circular extrema problem. *ACM Trans. on Prog. Lang. and Systems*, 4(4):758–752, 1982.