Leader Election Using Loneliness Detection (Extended Paper)

Luís Cruz[1]

[1]*MAP-i, Joint Doctoral Programme in Computer Science*

January 3, 2014

Abstract

The problem of Leader Election (LE) is studied in wireless single-hop networks with two types of collision detection — strong collision detection (SCD) and weak collision detection (WCD). It is shown that the main difference between these two detectors is the problem of Loneliness Detection (LD), describing an algorithm that implements SCD on top of WCD when such detector is provided.

Furthermore, the algorithm Bitwise Separator Protocol is presented showing that LD can be solved in $\mathcal{O}\log(\frac{u}{n})$. To prove that it is possible to solve LE in WCD systems, it is presented an algorithm that solves LE in SCD — Bitwise Leader Election Protocol (BLEP) — and then the algorithm that implements SCD on top of WCD is used to make the BLEP algorithm compatible with WCD channels, providing a solution in $\mathcal{O}\log(u)$.

1 Introduction

This document is an extended abstract of the work made by Ghaffari et al. (2012) which studies the leader election problem in single-hop radio networks with synchronised time slots for transmitting and receiving messages, that are subject to collision. The authors focus on the time cost of electing a leader its dependance on the number of actual processes in the network and the space ID of the processes. In the original work, both deterministic and randomised behaviors are analysed. However, in this document only the deterministic model is presented.

Leader election (LE) is a problem in which all the processes in the system unanimously select some process to be the leader. The time cost of solving LE varies according to the type of collision detection provided by the system. This problem has been well studied in single-hop systems that have no collision detectors and also single-hop systems with *strong collision detection* (SCD) wherein all processes can detect message collisions. However, this problem is under-explored in systems with *weak collision detection* (WCD), in which only listening processes can detect message collisions.

The best known time for deterministic LE for single-hop WCD systems is $\Theta(\log n)$ where n is known, based on the results for broadcast in multi-hop WCD systems (Schneider and Wattenhofer, 2010).

The primary challenge in LE in WCD systems is to find whether or not there is exactly one process in the system — *loneliness detection* (LD). To WCD system the following cases are indistinguishable:

- there is more than one process and all the processes send a message in the same round, resulting in a collision. Since with WCD only listening processes can detect collision and all the processes are transmitting, there is no process to detect collision;
- there is only one process, so any message sent by the process does not collide. However, there is no process listening so it is not possible to detect if the transmission was successful.

In both cases, the transmitting processes receive the same feedback. In SCD systems transmitting processes are also able to detect collisions and these two cases are distinguishable.

The formal definitions and the system model are presented in section 2. In section 3, the problem of Loneliness Detection (LD) is described, and it is showed that it is possible to implement an SCD system on top of a WCD system when having an implementation that solves the LD problem, and the algorithm Bitwise Separation Protocol is presented, solving LD in WCD channels in $\mathcal{O}\log(\frac{u}{n})$. The section 4 defines the properties of the problem of Leader Election (LE) and presents an algorithm that solves it in SCD systems in $\mathcal{O}(\log u)$ — Bitwise Leader Election Protocol, which is can be applied for WCD systems providing a solution in $\mathcal{O}\log(u)$.

2 System Modelling

The system is modeled using the deterministic Timed I/O Automata (TIOA) formalism (Kaynar et al., 2005). It considers a finite set of n processes with unique IDs from the finite space I of size u. The set $J \subseteq I$ the set of IDs of the n processes.

Processes can communicate by broadcasting messages from a fixed alphabet $\mathcal{M}$ that does not contain the special placeholder elements $\perp$ and $\top$, which will be used to denote silence and collision, respectively. Time is divided into *rounds*, and processes are synchronised, being able to detect the beginning and the ending of each round. It is assumed that all processes wake up at round 1, which starts at time 0. A process i transmits a message m at round r through the action $send(m, r)_i$, being considered a transmitting process. The process i receives a message m' through the action $receive(m', r)_i$. If in a round r a process i does not send a message, i.e., the process is a listener, it is assumed that process i executes action $send(\perp, r)_i$. Also, a process i that does not receive any message at round r inputs the action $receive(\perp, r)_i$.

The communication is made through a *wireless channel.* A wireless channel is a broadcast medium in which at most one process can successfully send a message in each round. The set of transmitting processes is denoted by T. If a single process sends a message m at round r, the event $receive(m, r)_i$ will occur in each process i of the system. If more than one process sends a message at round r, i.e., $|T| > 1$, it means that round r is experiencing a collision.

In WCD channels, in the case of a collision at round r, each listening process i receives the action of collision denoted by $receive(\top, r)_i$. Since the transmitting processes cannot detect collision, they will receive the same message they sent.

In the case of SCD channels, since all processes can detect messages collisions, when a collision occurs, each process i of the system inputs the action $receive(\top, r)_i$.

The wireless is assumed to be a *b-time-bounded channel*, i.e., each round takes at most a known time bound to complete. This is given by the nondecreasing function $B : \mathbb{N}^+ \rightarrow \mathbb{R}^{\geq 0}$ that maps each round r, to the upper bound $B(r)$.

3 Loneliness Detection

3.1 Specification

Loneliness detection (LD) informs all the processes in the system whether there exists one process in the system ($n = 1$). A solution to LD is an automaton that in some round r, for each process $i \in J$, outputs $alone(a_{LD}, r)_i$, where a_{LD} is Boolean, being *true* iff there is exactly one process in the system.

The LD problem is defined by the following two sets of properties:

Safety Properties Three safety properties are considered. First, if an $alone(true, *)_*$ event occurs, then there is exactly one process in the system; if an $alone(false, *)_*$ occurs, then there is more than one process in the system. Second, in any execution for every process i, the event $alone()$ occurs at most once. Third, in any execution for every pair of processes $i, j \in J$, if $alone(*, r_i)_i$ and $alone(*, r_j)_j$ occur, then r_i ad r_j are equal.

Liveness Property In any admissible execution, for each process i some $alone()_i$ event occurs.

In (Ghaffari et al., 2012) the authors demonstrate that the exactly difference between SCD and WCD systems is LD. They show that LD is trivially solved by presenting the algorithm *single-broadcast-LD* and present an algorithm that, given an LD service, implements SCD over a WCD channel.

3.2 SCD on WCD channel using LD

An algorithm that implements a deterministic SCD channel over a WCD channel is now presented. Basically the algorithm proceeds as follow. In the beginning of execution the LD service runs and each process i waits until it receives $alone(a_{LD}, r_{LD})$. Then, to emulate each SCD round r_s, two WCD rounds will be executed: the **Transmit** and the **Ack** rounds. Two extra set of actions are added to the automaton, having $sendWCD$ and $receiveWCD$ for the *actions* send $receive$ with WCD properties to distinguish from the $sendSCD()$ and $receiveSCD()$ that is implemented with SCD properties.

Transmit round. In the Transmit round each transmitting process sends their message m while the other processes are listening — all processes have the correspondent input on WCD channels as previously defined.

Ack round. In the Ack round the results are interpreted:

- If the process is a listener and received a message m' in the transmit round, then it outputs $sendWCD(\text{“}ack\text{”}, r_w)$. Round r_w denotes the round in the system with WCD properties and corresponds to $r_{LD} + 2r_s$.
- On the other hand, if the process is a transmitting process, it will be listening in this round r_w, waiting until it receives a message $p2msg$.
- After that, all the data is set to provided the correspondent sendSCD and receiveSCD events.
  - If m' was $\perp$, which means that the process is listening and that there is no collision, then it performs $receiveSCD(\perp, r_s)$.
  - If m' was not $\perp$, it means that it is a transmitting process. In the case of loneliness, $n = 1$, there is no collision, so the process performs $receiveSCD(m')$. If there are more processes in the system, $n > 1$, the message received in Ack round $p2msg$ is checked, and if $p2msg \neq \perp$ it means that some message was succesfully transmited and the process performs $receiveSCD(m', r_s)$. Otherwise, we are facing a message collision and the process performs $receive_i(\top, r_s)$

The proper proof is given in (Ghaffari et al., 2012) showing that the algorithm implements SCD on a WCD system using an LD service and that if the WCD channel is b_{WCD}-*time-bounded* and the LD service takes at most r_{LD} rounds, then the algorithm provides a *b-time-bounded* implementation, where $b(r) = b_{WCD}(r_{LD} + 2r)$.

3.3 Bitwise Separation Protocol

Bitwise Separation Protocol (BSP) is an algorithm that solves the deterministic LD problem in WCD systems. It is based on the ID space of the processes which has the size u. Each ID can be represented by a sequence of $\log_2(u)$ bits. The algorithm iterates over each bit of the id, starting from the least significant bit, $id[k = 1]$, to the most significant, $id[k = \log_2(u)]$. In each iteration two rounds proceed: the *transmission* and the *ack* rounds.

Transmission If the current bit is 1, the process sends a message m, $send(m, 2k-1)$. Otherwise, the process is listening, $send(\perp, 2k - 1)$. In the end of the round, the process receives a message m'

Ack If the process receives some message $m \neq \perp$ and was not transmitting in the previous message, i.e., $id[k] \neq 1$, it sends the message “*ack*” (note that this process knows already that it is not lonely, but it will try to propagate this info to the other processes). In the other case, $id[k] = 1$, the process will not transmit, performing $send(\top, 2k)$. The process receives a message m'' and if $m'' \neq \perp$, it means that it is not the only process in the systems, performing $alone(false, 2k)$ and halting.

If the process executes all the $\log_2(u)$ phases without recognising any other process, i.e., without performing $alone(false, *)$ it performs $alone(true, 2\log_2(u))$.

This means that if $n = 1$ then the algorithm takes $2\log_2(u)$ rounds to recognise it — $\mathcal{O}(\log(u))$.

In (Ghaffari et al., 2012), for the proof of correctness, it is demonstrated that BSP satisfies the safety and liveness properties of LD, showing also that it takes at most $2[\log_2(u) - \log_2(n) + 1]$ rounds for $1 < n \leq u$. This means that BSP solves the deterministic LD problem in $\mathcal{O}(\log \frac{u}{n})$. Also it is demonstrated that no algorithm can solve deterministic LD with less than $\log(u) - \log(n-1) - 1$ rounds, showing that the time complexity of BSP matches the lower bounds.

With this algorithm, it is proved that it is possible to have a WCD system that implements SCD has shown previously in 3.2. This implementation provides a B_{WCD} time-bounded deterministic SCD channel on a $BWCD$ time-bounded WCD channel where $B_{SCD}(n, r) = B_{WCD}(n, \log u - \log n + 1 + 2r)$ (Ghaffari et al., 2012).

4 Leader Election

The problem of leader election (LE) can be modeled by an automata in which every process i in the system eventually outputs $leader(l)_i$, where l is the ID of the leader process. The following two set of properties clearly define the process:

Safety Properties For every process $i \in J$ at most one $leader_i$ event occurs and for every pair of processes $i, j \in J$, if events $leader(l_i)_i$ and $leader(l_j)_j$ occur, then $l_i = l_j$ and $l_i \in J$.

Liveness Property In any admissible execution, the event $leader_i$ occurs for every process $i \in J$.

Since it is possible to implement an SCD on top of a WCD system, any algorithm that solves LE in SCD can be adapted for WCD. In this section, the algorithm *Bitwise Leader Election Protocol* (BLEP) is presented and the time complexity of such an algorithm for WCD systems is studied providing its upper bound.

4.1 Bitwise Leader Election Protocol

The BLEP algorithm proceeds by iterating over the binary sequence of the *id* of each process. It starts from the least significant bit, where $id_i[k]$ denotes the k-the bit of the process i's *id*. In the beginning of the execution any process can be a leader, but as it goes some processes are removed from the race. This is stated using the Boolean variable *active* — it is *true* when the process can still be a leader.

In each round r every process i that is still active and has $id[r]_i = 1$ transmits a message with its *id*; all other processes are listening. If in the end of the round, a listening process that is active receives a collision notification $\top$, it gives up the race and turns the Boolean *active* into $false$. On the other hand, if in the end of the round a process i successfully receives a message, this message will be the id of some process, which is our leader, performing $leader(m)_i$ and halts, stopping the execution. On the other hand, if the process receives silence $\perp$, it means that there was no transmitting process and it does nothing. The execution proceeds to round $r + 1$.

It is not necessary for the algorithm to know the size of the ID space I. The algorithm will run until it finds the leader, which in the worst case will be until the most significant bit of u, i.e., round $\log_2(u) + 1$. Note that this algorithm only works if the space $I \in [1, u]$, if the system has only one process i with $id_i = 0$, the algorithm is not able to find a leader.

In (Ghaffari et al., 2012), it is proved that BLEP solves deterministic LE, and it is R_{LE} round bounded were $R_{LE} = \log(u) + 1$. BLEP has a time complexity of $\mathcal{O}(\log u)$.

4.2 Leader Election in WCD Systems

Combining the implementation of SCD over WCD described in 3.2 and the Leader Election algorithm BLEP for SCD systems described in 4.1, it is possible to have a WCD system that solves LE. In this section, the time complexity of this solution is presented.

An implementation of SCD on top of WCD has the time bound function $B_{SCD}(n, r) = B_{WCD}(n, \log u - \log n + 1 + 2r)$. Besides, the BLEP algorithm solves LE in SCD systems in $\log u + 1$ rounds. The combination gives the time bound function

$$\begin{aligned} B_{SCD}(n, \log(u) + 1) &= B_{WCD}(n, \log u + 1 - \log n + 1 + 2(\log(u) + 1)) && (1) \\ &= B_{WCD}(n, \Omega(\log u - \log n)) && (2) \\ &= B_{WCD}(n, \mathcal{O}(\log u)) \text{ if } u >> n && (3) \end{aligned}$$

Therefore, there exists an algorithm that solves the deterministic LE and is R_{LE}-round-bounded where $R_{LE}(n) = B_{WCD}(n, \mathcal{O}(\log u))$.

5 Conclusion

In this document the work of Ghaffari et al. (2012) was briefly described regarding the deterministic behavior. The problem of deterministic Leader Election (LE) in wireless Single Hop systems subject to collisions was studied. The problem of Loneliness Detection was introduced and it was shown that the main distinction between Strong Collision Detection (SCD) systems and Weak Collision Detection Systems (WCD) is the Loneliness Detection (LD) problem.

The Bitwise Separation Protocol (BSP) was described providing an algorithm to solve the deterministic LD problem in $\mathcal{O}(\log \frac{u}{n})$.

It was shown that it is possible to implement an SCD systems over an WCD system using LD. This is an important approach because it can merge the research for both WCD and SCD systems.

The algorithm Bitwise Leader Election Protocol was introduced implementing a solution to LE in SCD systems in $(O)(\log u)$. Applying this algorithm to the SCD implementation on top of WCD system it was possible to study LE in WCD systems, giving the upper bound $(O)(\log u)$.

In this document it is considered that the channels are b-time-bounded and each round takes at most the known finite time b to finish. In the case of probabilistic channels, the duration of a round is not fixed *apriori* and other methods have to be applied. This study was out of the scope of this document but the reading of the full article (Ghaffari et al., 2012) is recommended.

References

Ghaffari, M., Lynch, N., and Sastry, S. (2012). Leader election using loneliness detection. *Distributed Computing*, 25(6):427–450.

Kaynar, D. K., Lynch, N., Segala, R., and Vaandrager, F. (2005). The theory of timed i/o automata. *Synthesis lectures on computer science. Morgan Claypool.*

Schneider, J. and Wattenhofer, R. (2010). What is the use of collision detection (in wireless networks)? In Lynch, N. and Shvartsman, A., editors, *Distributed Computing*, volume 6343 of *Lecture Notes in Computer Science*, pages 133–147. Springer Berlin Heidelberg.