Self-Stabilizing Counting in Mobile Sensor Networks

Joffroy Beauquier, Julien Clément, Stephane Messika, Laurent Rosaz, Brigitte Rozoy

LRI, CNRS & Univ. Paris Sud,
91405 Orsay Cedex, France

15th March 2007
Summary

1 Introduction
   • Notations
   • Definitions

2 The TA model
   • Infinite memory
   • Finite or bounded memory

3 The TATP model
   • The ATATP model
   • The STATP model

4 Conclusion
Related works

- **SIVAM project**: The impact of the climate evolution
- **Wireless Distributed Sensor Networks for In-situ Exploration of Mars**: Mars exploration
- **Wireless Sensor Networks for Habitat Monitoring**: Studying of Leach's Storm Petrels
### Mobile sensor networks

#### Modelization

- A group of penguins evolves on an island, carrying on their body a small sensor.

- Whenever a penguin is close enough to the antenna, its sensor interacts with the antenna.

- Depending on the hypothesis, the sensors may or may not interact with each other when two penguins approach close enough.
The Model

A configuration of the network is given by

- $a$, content of the memory of the antenna

- $(p_1, ..., p_n)$, where $p_i$ is the state of the $i^{th}$ penguin.

Initial conditions

We assume that the penguins are arbitrarily initialized, but that an initial value can be chosen for the antenna.
Two kinds of events

- The meeting of penguin $i$ with the antenna.
  \[ p_i \rightarrow p'_i \text{ and } a \rightarrow a' \]

- The meeting of penguin $i$ with penguin $j$.
  \[ p_i \rightarrow p'_i \text{ and } p_j \rightarrow p'_j \]
Two different scenarios

- The Penguins-To-Antenna-Only model (TA)
- The Penguins-To-Antenna-And-To-penguins model (TATP)
  - The symmetric one (STATP) : two penguins meeting in the same state have to change to the SAME state.
  - The asymmetric one (ATATP) : two penguins meeting in the same state don’t have to change to the SAME state.
Definition: Configuration

A configuration is the state of all penguins at a time $t_0$.

Definition: Execution

An execution is an infinite sequence $(C_j)_{j \in \mathbb{N}}$ of configurations and an infinite sequence $(e_j)_{j \in \mathbb{N} \setminus \{0\}}$ of events such that $C_{j+1}$ is obtained after $e_j$ occurs on $C_j$. 
Definition: Rounds

A round is a sequence of consecutive events, during which every petrel meets the antenna at least once, and in the TATP model, every two petrels meet each other.

- The first round is the shortest round starting from initial configuration.
- The second round is the shortest round starting from the end of the first round, and so on.
Definition: Fairness
An execution is fair if every penguin communicates with the antenna infinitely often, and, in the TATP model, if every two penguins communicate with each other infinitely often.

- A daemon for a deterministic protocol is fair if every execution is fair.
- A daemon for a probabilistic protocol is strongly fair if the execution is fair.
- A daemon for a probabilistic protocol is weakly fair if, the measure of the set of the fair executions is 1.
**Definition: k-fairness**

An execution is k-fair,

- In TA model, if every penguin communicates with the antenna at least once in every $k$ consecutive events

- In TATP model, if furthermore every two penguins communicate with each other in every $k$ consecutive events.

A daemon is $k$-fair if the execution is $k$-fair.
Size of the penguin sensor memories

- The memory is infinite if it is unlimited.

- The memory is bounded if an upper-bound $P$ on the number of penguins is known, and if the number of different possible states of the memory is $\alpha(P)$.

- The memory is finite if the number of different possible states of the memory is $\alpha$. 
Summary

1. Introduction
   - Notations
   - Definitions

2. The TA model
   - Infinite memory
   - Finite or bounded memory

3. The TATP model
   - The ATATP model
   - The STATP model

4. Conclusion
Algo 1 : Infinite memory

Memory in the penguins sensor is
   number : integer

Memory at the antenna is
   registers indexed by N, initialized at 0
   PenguinNumber is maxregister[i]

When a penguin with number x approaches the antenna :
   number <- x+1
   R[number] <- R[number]+1
A fair daemon, any deterministic solution?

**Proposition**

The daemon is supposed to be fair. If the sensors have a finite memory then, there is no deterministic algorithm solving the counting problem.

The idea of the proof is to exhibit two executions resulting from two different initial configurations that will appear to be identical for the antenna.
Proof

Let us consider an execution $E$:

- $n$ sensors $(p_1, \ldots, p_n)$
- Initial configuration $I = (x_1, x_2, \ldots, x_n)$.

Lemma

There exists a state, denoted by $s$, that the sensor $p_n$ will visit infinitely often.

Proof: The sensors have a finite memory.
Proof (ctd)

Let $E'$ be the following execution:

- $n+1$ sensors $(p_1, \ldots, p_n, p_{n+1})$

- Initial configuration $I = (x_1, x_2, \ldots, x_n, s)$.

**Step 1**

The daemon holds back the sensor $p_{n+1}$ while letting the others $n$’s evolve as in $E$. As soon as $p_n$ reaches the state $s$ after having seen the antenna at least once, it replaces it by $p_{n+1}$ that was initialized in state $s$. 
Step 2

Then it lets $p_{n+1}$ and the other $n - 1$'s sensors evolve, while making act $p_{n+1}$ exactly like $p_n$ previously, until the state $s$ is reached by $p_{n+1}$ and the antenna seen at least once by $p_{n+1}$. Then it replaces it by $p_n$

and so on...
A fair daemon, any probabilistic solution?

Proposition

Suppose that the daemon is strongly fair. If the sensors have a finite memory, then it does not exist any probabilistic algorithm solving the counting problem.

The idea of the proof is that the antenna has a non null probability to lose.
Idea of the proof

Lemma

Let us consider a daemon $D$ with $n$ sensors $(p_1, \ldots, p_n)$ initialized in $I = (x_1, x_2, \ldots, x_n)$.

$$\exists s \exists \eta \in \mathbb{R} \quad P\{p \text{ goes infinitely often in } s\} \geq \eta$$
A $k$-fair daemon

Proposition

If we assume the daemon is $k$-fair, we get both deterministic and probabilistic solutions to the counting problem.
Algo 2 : Deterministic, \( k \)-fair daemon

Memory in the penguins sensor is \( \text{bitP} : \text{boolean} \)
Memory at the antenna is

- \( \text{cpt} \), \( \text{PenguinNumber} : \text{integer} \)
- \( \text{bitA} : \text{boolean} \), initialized at 0

The antenna does:
For \( i \) from 1 to infinity do
- \( \text{cpt} \leftarrow 0 \)
- do 2 power \( i \) times:
  - see arriving one sensor \( P \) with \( \text{bitP} \)
  - if \( \text{bitP} = \text{bitA} \) then \( \text{cpt} \leftarrow \text{cpt} + 1 \)
  - \( \text{bitP} \leftarrow \text{not(\text{bitP})} \)
- \( \text{PenguinNumber} \leftarrow \text{cpt} \)
- \( \text{bitA} \leftarrow \text{not(\text{bitA})} \)
Algo 2: Deterministic, $k$-fair daemon
Algo 3 : Probabilistic, $k$-fair daemon

Memory in the penguins sensor is
number, color: integer
Memory at the antenna is
  registers indexed by N, initiat empty
  PenguinNumber is card($k$ / register[$k$] not empty)
If penguin with number $n$ and color $c$ approaches the antenna:
  if $R[n] = \text{empty}$
    then $R[n] \leftarrow c$
  else if $R[n] = c$
    then color $\leftarrow \text{random1..PenguinNumber}$
        $R[n] \leftarrow \text{color}$
        $R[n+1] \leftarrow c$
  else number $\leftarrow n+1$
        $R[\text{number}] \leftarrow c$
Summary

1 Introduction
   • Notations
   • Definitions

2 The TA model
   • Infinite memory
   • Finite or bounded memory

3 The TATP model
   • The ATATP model
   • The STATP model

4 Conclusion
Bounded memory

Proposition

There are deterministic solutions to the counting problem with $\alpha(P) \geq P$
Algo 4 : Asymmetric algorithm with $\alpha(P) = P$

Memory in the penguins sensor is
number : integer in $[1..P]$
Memory at the antenna is
T array $[1..P]$ of boolean, init at 0 everywhere
PenguinNumber is the number of i such that T[i]=1
When a penguin with number x approaches the antenna :
T[x] <- 1
When two penguins meet :
If their numbers are the same integer x
then the number of one penguin becomes $x+1 \mod P$
Algo 4: Asymmetric algorithm with $\alpha(P) = P$
Algo 4 : Asymmetric algorithm with $\alpha(P) = P$
Algo 4 : Asymmetric algorithm with $\alpha(P) = P$
Algo 4: Asymmetric algorithm with $\alpha(P) = P$
Algo 4: Asymmetric algorithm with $\alpha(P) = P$
Algo 4: Asymmetric algorithm with $\alpha(P) = P$
Algo 5: Asymmetric algorithm with $\alpha(P) = P + 1$

Memory in the penguins sensor is

number : integer in $[0..P]$

Memory at the antenna is

T array $[1..P]$ of boolean, init at 0 everywhere

PenguinNumber is the number of i such that $T[i]=1$

If a penguin with number $x$ approaches the antenna:

if $x = 0$

then let $y$ be an integer such that $T[y]=0$

T[y] <- 1

number <- y

else $T[x] <- 1$

When two penguins meet:

If their numbers are the same

then the number of one penguin becomes 0
Self-Stabilizing Counting in Mobile Sensor Networks

The TATP model

The STATP model

Algo 6: Deterministic symmetric algorithm with \( \alpha(P) = 4P \)

Memory in the penguins sensor is
number : integer in \([1..2P]\)
Intention : (Keep, GiveUp)

Memory at the antenna is
T array \([1..2P]\) of (Free, Taken, GivenUp),
initialized at Free everywhere
PenguinNumber is the number of i such that
T[i]=Taken
Algo 6: Deterministic symmetric algorithm with \( \alpha(P) = 4P \)

When a penguin with number \( x \) approaches the antenna:
- Depending on Intention:
  - Keep: \( T[x] \leftarrow \text{Taken} \) /* even if \( T[x] \) was \( \text{GivenUp} \) */
  - GiveUp: \( T[x] \leftarrow \text{GivenUp} \)

  find a \( y \) such that \( T[y] = \text{Free} \)
  \( T[y] \leftarrow \text{Taken} \)
  number \( \leftarrow y \)
  Intention \( \leftarrow \text{Keep} \)

When two penguins meet:
- If their numbers are the same integer \( x \)
  and their both intentions are Keep
- Then their both intentions change to GiveUp
Proposition

There does not exist asymmetric algorithms with $\alpha(P) \leq P - 1$ solving the counting problem.
## The TA model

<table>
<thead>
<tr>
<th>model \ memory</th>
<th>Finite</th>
<th>Bounded</th>
<th>Bounded, k-fair daemon</th>
<th>Unbounded</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>deterministic</strong></td>
<td>impossible</td>
<td>impossible</td>
<td>Algorithm 2</td>
<td>Algorithm 1</td>
</tr>
<tr>
<td>convergence time</td>
<td></td>
<td></td>
<td>4k events</td>
<td>depends on initialization</td>
</tr>
<tr>
<td><strong>probabilistic</strong></td>
<td>impossible</td>
<td>impossible</td>
<td>Algorithm 3</td>
<td>unneeded</td>
</tr>
<tr>
<td>convergence time</td>
<td></td>
<td></td>
<td>exponential in $k$</td>
<td></td>
</tr>
</tbody>
</table>
The TATP model

<table>
<thead>
<tr>
<th>model \ memory</th>
<th>Finite</th>
<th>Bounded, $\alpha(P) &lt; P$</th>
<th>Bounded, $\alpha(P) \geq P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>symmetric deterministic</td>
<td>impossible</td>
<td>impossible</td>
<td>Algorithm 6</td>
</tr>
<tr>
<td>convergence time</td>
<td></td>
<td></td>
<td>$\alpha(P) = 4P$, 3 rounds</td>
</tr>
<tr>
<td>asymmetric deterministic</td>
<td>impossible</td>
<td>impossible</td>
<td>Algorithm 4 or 5</td>
</tr>
<tr>
<td>convergence time</td>
<td></td>
<td></td>
<td>$\alpha(P) = P$, $P+1$ rounds</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\alpha(P) = P + 1$, 3 rounds</td>
</tr>
</tbody>
</table>
Future works

- Choose the movement of the penguins
- Improve complexity bounds
- Implement the protocols