

???

JOLY Amaury

Encadrants : GODARD Emmanuel, TRAVERS Corentin

Aix-Marseille Université, Scille

30 avril 2025

1 Introduction

1.1 Model

1.1.1 Model Properties

The model is defined as Message-passing Aysnchronous.

There is n process. Each process is associated to a unique unforgeable id i .

Each process know the identity of all the process in the system

Each process have a reliable communication channel with all the others process such as :

- $\text{send}(m)$ is the send primitive
- $\text{recv}(m)$ is the reception primitive

A message send is eventually received

The system is Crash-Prone. There is at most f process who can crash such as $f < n$.

1.1.2 AtomicBroadcast Properties

Property 1 *AB_broadcast Validity* if a message is sent by a correct process, the message is eventually received by all the correct process.

Property 2 *AB_receive Validity* if a message is received by a correct process, the message is eventually received by all the correct process.

Property 3 *AB_receive safety No creation* if a message is received by a correct process, the message was emitted by a correct porcess.

Property 4 *AB_receive safety No duplication* each message is received at most 1 time by each process.

Property 5 *AB_receive safety Ordering* $\forall m_1, m_2$ two messages, $\forall p_i, p_j$ two process.
if $\text{AB_recv}(m_1)$ and $\text{AB_recv}(m_2)$ for p_i, p_j
and $\text{AB_recv}(m_1)$ is before $\text{AB_recv}(m_2)$ for p_i
so $\text{AB_recv}(m_1)$ is before $\text{AB_recv}(m_2)$ for p_j

1.1.3 DenyList Properties

Property 6 *APPEND Validity* a $\text{APPEND}(x)$ is valid iff the process p who sent the operation is such as $p \in \Pi_M$. And iff $x \in S$ where S is a set of valid values.

Property 7 *PROVE Validity* a $\text{PROVE}(x)$ is valid iff the process p who sent the operation is such as $p \in \Pi_V$. And iff $\exists \text{APPEND}(x)$ who appears before $\text{PROVE}(x)$ in Seq.

Property 8 *PROGRESS* if an $\text{APPEND}(x)$ is invoked, so there is a point in the linearization of the operations such as all $\text{PROVE}(x)$ are valids.

Property 9 *READ Validity* $\text{READ}()$ return a list of tuples who is a random permutation of all valids $\text{PROVE}()$ associated to the identity of the emitter process.

1.2 Algorithms

We define k as the id of the round

the $\text{getMax}(proves)$ return $\text{MAX}((_, r) : \exists(_, \text{PROVE}(r)) \in proves)$

$buffer$ a FIFO list with $buffer[front]$ returning the first element

Algorithm 1: AB_Broadcast

Input: le message m

```
1 k_max = getMax(READ())
2 while buffer ≠ [∅] do
3   (i, m) = buffer[front]
4   wait PROVE(k_max || m) = false
5   proves = READ()
6   if ((i, PROVE(k_max || m)) ∈ proves) ∧ ((i, PROVE(k_max)) ∈ proves) then
7     | buffer = buffer − {(i, m)}
8   end
9 end

10 while true do
11   proves = READ()
12   k = getMax(proves) + 1
13   PROVE(k || m)
14   APPEND(k || m)
15   if PROVE(k) then
16     | APPEND(k)
17     | return
18   end
19 end
```

$proves_r = (i, r) : \forall i, \exists(i, PROVE(r)) \in proves$
 $proves_r^i = ((i, r) : \exists(i, PROVE(r)) \in proves) ?? \text{NULL}$

Algorithm 2: AB_Listen

```
1 buffer = [∅]
2 k = 0

3 while true do
4   proves = READ()
5   k_max = getMax(proves)
6   for r=k+1 to k_max do
7     APPEND(r)
8     for i = 1 to |P| do
9       if (∃m : ∃(i, PROVE(r || m)) ∈ proves) then
10         if (∃(i, PROVE(r)) ∈ proves) then
11           | AB_Recv(m)
12         else
13           | buffer = buffer ∪ {(i, m)}
14         end
15       end
16     end
17   end
18 end
```
