? ? ?

JOLY Amaury
Encadrants : GODARD Emmanuel, TRAVERS Corentin

Aix-Marseille Université, Scille

25 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 :

— send(m) is the send primitive

— recv(m) is the reception primitive

A message send is eventualy 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 AB_recv(m1) and AB_recv(m2) for p_i, p_j
and AB_recv(m1) is before AB_recv(m2) for p_i
so AB_recv(m1) is before AB_recv(m2) for p_j

1.1.3 DenyList Properties

Property 6 *APPEND Validity a 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 PROVE(x) is valid iff the process p who sent the operation is such as* $p \in \Pi_V$*. And iff* $\exists$ *APPEND(x) who appears before PROVE(x) in Seq.*

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

Property 9 *READ Validity READ() return a list of tuples who is a random permutation of all valids PROVE() associated to the identity of the emiter process.*

1.2 Algorithms

We define k as the id of the round
the $getMax(..)$ function able to return the highest round played in the system.

Algorithm 1: AB_Broadcast

```
Input: le message m
while true do
    proves = READ()
    k = getMax(dump) + 1
    APPEND(k || m)
    if PROVE(k) then
        APPEND(k)
        return
    end
end
```

We define k_max as an intager
$getMax(..)$ function able to return the highest round played in the system.
$proves_r \subseteq proves$ s.a. $\forall PROVE(x) \in proves_r$, x is in the form $r||m$ with m who cannot be empty
$proves_r^i$ is the $PROVE(r||m)$ operation submited by the process i if exist

Algorithm 2: AB_Listen

```
while true do
    proves = READ()
    k_max = getMax(proves)
    for r=k+1 to k_max do
        APPEND(r)
        proves_r = {∀i, PROVE(r)_i ∈ READ()}
        for i = 1 to |P| do
            if ∃PROVE(r)_i ∈ proves_r then
                AB_Recv(m s.t. PROVE(r||m) ∈ proves)
            end
        end
    end
end
```