? ? ?

JOLY Amaury
Encadrants : GODARD Emmanuel, TRAVERS Corentin

Aix-Marseille Université, Scille

2 mai 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 $getMax(proves)$ return $MAX((\_, r) : \exists(\_, PROVE(r)) \in proves)$
$buffer$ a FIFO list with $buffer[front]$ returning the first element

Algorithm 1: Upon RB_deliver(m)

```
rcved = rcved ⋃ {m}
upon RB_deliver(PROP, r, S) from j
prop[r][j] = S
```

Algorithm 2: AB_Broadcast

Input: le message m
Data: rcved = ∅
delivered = ∅
r = 0

```
RB_cast(m)
rcved = rcvd ⋃ {m}
while true do
    r = r + 1
    RB_cast(PROP, r, S)
    PROVE(r)
    APPEND(r)
    proves = READ()
    winner^r = {j : (j, PROVE(r)) ∈ proves}
    wait until (∀j ∈ winner^k : prop[r][j])
    if ∃j ∈ winner^k : m ∈ prop[r][j] then
        break
    end
end
```

Algorithm 3: AB_Listen

```
r_prev = 0
while true do
    proves = READ()
    r_max = MAX({r : ∃i, (i, PROVE(r)) ∈ proves})
    for r = r_prev + 1 to r_max do
        APPEND(r)
        proves = READ()
        winner^k = {j : (j, PROVE(r)) ∈ proves}
        wait until(∀j ∈ winner^k : prop[r][j] ≠ ∅)
        M^r = (⋃_{j∈winner^k} prop[r][j]) \ delivered
                /* we assume M^r as an ordered list s.a. ∀m_1, m_2, if m_1 < m_2, m_1 appears
         before m_2 in M^r */
        foreach m ∈ M^r do
            delivered = delivered ⋃ {m}
            AB_deliver(m)
        end
    end
end
```