Rabia Errata[*]

Rabia [2] is a practical instantiation of a randomized consensus algorithm that builds on Ben-Or's randomized binary consensus protocol [1] and produces a state machine. The core idea is that Ben-Or is used to agree whether a variable v is 0 or 1, and 0 implies that a 'nop' command ($\perp$) should be committed, while 1 implies that an actual command should be committed. The protocol uses an initial proposal phase (Lines 1–8, referred to as phase 1 in the rest of this note) to determine if a quorum of nodes agree on a proposed command, and if so proposes $v = 1$ for the Ben-Or algorithm. Nodes should execute the command that a quorum agreed on if $v = 1$ succeeds.

Unfortunately, the current version has a slight bug. Observe, that if a node decides $v = 1$, then (a) a quorum agreed on a command q; and (b) the node should find and execute q. The paper defines a FINDRETURNVALUE (Algorithm 3) that is supposed to do this, and it does so by looking at the PROPOSAL messages received during Phase 1 and using the command q contained in $\lfloor \frac{n}{2} \rfloor + 1$ PROPOSAL messages. However, failures can result in situations where the protocol decides $v = 1$ but no live node has seen $\lfloor \frac{n}{2} \rfloor + 1$ PROPOSAL messages for the same command: a single node might have proposed $v = 1$ after observing a quorum propose the same command q, but might have failed during the Ben-Or phase. This shows a problem with adopting unmodified Ben-Or for this setting.

Our approach to fixing this problem is to use Ben-Or so it decides between $v = \perp$ and $v = q$, where q is the proposed command a quorum agrees on (if any, otherwise all nodes propose $v = \perp$). Observe, this is still a binary consensus problem, allowing us to largely reuse Ben-Or's protocol. The challenge lies in recovering q at the point in the protocol where Ben-Or uses a coin toss (Line 27). Here we rely on the fact that to arrive at line 27, the node must have seen no VOTE messages for values other than ?. We assume that a node always sees its own VOTE message (Line 21) because communication from a node to itself does not go over the network and is lossless. Under this assumption, to arrive at Line 27, the node must have sent a VOTE message with ?. This in turn means that none of the states s that the node received in the STATE messages (Line 16) appeared more than $\lfloor \frac{n}{2} \rfloor + 1$ times, which in turn means that the node must have received STATE messages with $\perp$ and q. Thus, the $states$ set (Line 15) for any node that executes Line 27 must contain both $\perp$ and q.

The updated algorithm is shown below, and does not use FINDRETURNVALUE.

```
1: ▷ WeakMVC invoked with proposal $q$ at a particular node.
2: ▷ Phase 1: Exchange protocols.
3: Send(PROPOSAL, $q$) to all
4: Wait for $\geq n - f$ PROPOSAL messages
5: **if** proposal $p$ appears more than $\lfloor \frac{n}{2} \rfloor + 1$ times **then**
6:     $state \leftarrow p$                                   ▷ Node will propose $p$
7: **else**
8:     $state \leftarrow \perp$                                   ▷ Node will propose $\perp$
```

[*]Authorship Note: The bug that this errata attempts to fix was found by Wen Zhang (UC Berkeley) and Aurojit Panda (NYU), though we were not the first; the algorithm described here was developed by Wen Zhang and Aurojit Panda; and was checked by the authors and Joseph Tassarotti (NYU). The current write-up was written by Aurojit Panda for use in the 2024 edition of NYU's graduate distributed systems class. If citing this note, please attribute authorship to Wen Zhang (UC Berkeley), Aurojit Panda (NYU), and Joseph Tassarotti (NYU).

```
 9: end if
10: ▷ Phase 2: Use Ben-Or to reach agreement.
11: $r \leftarrow 1$
12: while true do
13:     Send(STATE, $r$, $state$) to all
14:     wait until receiving $\geq n - f$ STATE messages for round $r$
15:     $states \leftarrow$ set of sates in received STATE messages from round $r$        ▷ $states$ has at most 2 elements.
16:     if state $s$ appears $\geq \lfloor \frac{n}{2} \rfloor + 1$ times in round-$r$ STATE messages then
17:         $v \leftarrow s$
18:     else
19:         $v \leftarrow ?$
20:     end if
21:     Send(VOTE, $r$, $v$) to all
22:     wait until receiving $\geq n - f$ VOTE messages for round $r$
23:     if non-? vote $v'$ appears $\geq f + 1$ times in round-$r$ VOTEs then
24:         return $v'$                                                             ▷ Termination
25:     else if non-? vote $v'$ appears at least once in round-$r$ VOTE messages then
26:         $state \leftarrow v'$
27:     else
28:         ▷ $states$ (Line 15) must have exactly two elements: $\perp$ and $p$ a proposal selected in Phase 1.
29:         $c \leftarrow CommonCoin(r)$
30:         if $c = 0$ then
31:             $state \leftarrow \perp$
32:         else
33:             $state \leftarrow$ non-$\perp$ element $p$ in $states$
34:         end if
35:     end if
36:     $r \leftarrow r + 1$
37: end while
```

References

[1] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *PODC*, pages 27–30, 1983.

[2] H. Pan, J. Tuglu, N. Zhou, T. Wang, Y. Shen, X. Zheng, J. Tassarotti, L. Tseng, and R. Palmieri. Rabia: Simplifying state-machine replication through randomization. In *SOSP*, 2021.