

Exercise 4: Extreme Democracy

Task 1: Everyone Gets exactly one Vote...

The goal of this exercise is to prove correct the asynchronous safe broadcast algorithm by Bracha. It tolerates $f < n/3$ Byzantine faults, so we will assume that this condition holds.

Algorithm 1 Code of the safe broadcast algorithm at node v . The input message M is given to the designated source node s ; every node knows s . Any applied thresholds require messages from different nodes; duplicate messages from the same sender are dropped.

```
1: if  $v = s$  then
2:   send  $\text{init}(M)$  to all nodes (including self)
3: end if
4: Stage 1: wait until received
   • one  $\text{init}(M')$  message from  $s$ ,
   •  $n - f$   $\text{echo}(M')$  messages, or
   •  $n - 2f$   $\text{ready}(M')$  messages
   for some  $M'$ 
5: send  $\text{echo}(M')$  to all nodes (including self)
6: Stage 2: wait until received
   •  $n - f$   $\text{echo}(M')$  messages, or
   •  $n - 2f$   $\text{ready}(M')$  messages
   for some  $M'$  (including those from stage 1)
7: send  $\text{ready}(M')$  to all nodes (including self)
8: Stage 3: wait until received
   •  $n - f$   $\text{ready}(M')$  messages
   for some  $M'$  (including those from earlier stages)
9: output  $M'$ 
```

a) Show that if s is correct, eventually all correct nodes output M !

Hint: Argue that faulty nodes cannot make correct nodes send a “non- M ” message. Conclude that all nodes pass all stages for M .

b) Show that if a correct node broadcasts a $\text{ready}(M')$ message, no correct node broadcasts a $\text{ready}(M'')$ message for $M'' \neq M'$!

Hint: Use that correct nodes broadcast only one $\text{echo}(\cdot)$ message, but the first nodes broadcasting $\text{ready}(\cdot)$ messages must do so because of receiving many echoes!

c) Show that if a correct node outputs a message M' , eventually all correct nodes output M' !

Hint: Use b) to show that no correct node can pass stage 2 for $M'' \neq M'$. Then argue that eventually nodes get “pulled” through the first two stages because they receive sufficiently many $\text{ready}(M')$ messages.

d) Conclude that the algorithm correctly implements safe broadcast!

Task 2: ... and then a Random Decision is Taken!

Consider the following shared coin.

Algorithm 2 Simple weak shared coin (code at node v).

- 1: flip an unbiased coin
 - 2: send the result to everyone (also self)
 - 3: wait until received bits from $n - f$ different senders
 - 4: output the majority value (0 in case of a draw)
-

- a) Show that if $f < n/3$, this algorithm implements a weak shared coin with defiance 2^{-n} .
- b) Show that if $f \in \mathcal{O}(\sqrt{n})$, this algorithm implements a weak shared coin with constant defiance.

Hint: By the central limit theorem, the binomial distribution converges to a normal distribution for $n \rightarrow \infty$, in the sense that the relative error of approximating it by the normal distribution goes to 0. Check the standard deviation of the binomial distribution and make use of this connection.

- c) Show that if $f = \alpha\sqrt{n}$ for $\alpha \in [1, \sqrt{n}/3]$, then this algorithm implements a weak shared coin with defiance $2^{-\mathcal{O}(\alpha^2)}$.

Hint: Check out the section on tail bounds of the binomial distribution on Wikipedia.

- d) Use this to show that for every $f < n/4$, there is an asynchronous consensus algorithm tolerating up to f faults that terminates in expected time $2^{\mathcal{O}(\lceil f^2/n \rceil)}$.
- e) Can this approach be used to create an algorithm that tolerates any number of $f < n/4$ faults, but terminates faster if the actual number of faults is small? (An educated guess suffices, you don't need to prove your answer correct here.)

Task 3*: Lecturing the Lecturer

- a) Find out why Byzantine failures are called Byzantine!
- b) Conclude that the lecturer is biased towards always pointing at the same person. Which celebrities of distributed computing could/should be featured instead?¹
- c) Tell the tale of how Byzantine faults have been named and the heroes that have fought them throughout the decades in the exercise session!

¹And anyway, shouldn't he stop asking vague questions?