10.1 Byzantine Randomized Consensus

(a) Optimal Resilience for Byzantine Consensus?

We assume that we have four processes p_1 to p_4, of which p_1 is the only faulty one. Process p_2 proposes the value 0 but processes p_3 and p_4 are proposing 1. At last we assume that the propose of any process to itself does not affect the decision process because it is delayed.
In the first round process p_3 has received from p_1 and p_2 the proposed value 0 and from process p_4 it receives 1. Therefore $m = 2$ and $v^\star = 0$ and if *coin* happens to be 0, then p_3 will decide on the value 0.
The process p_4 has received the proposed value 0 from p_2 and 1 from p_1 and p_3. Therefore $m = 2$ and $v^\star = 1$ and if *coin* happens to be 1, then p_4 will decide on 1.
In this scenario the processes p_3 and p_4 decide on different values which is violating the *Agreement Property* and therefore this algorithm does not achieve the fault tolerance of $N > 3f$.

(b) Guru Resilience?

We assume that we have five processes p_1 to p_5, of which p_1 is the only faulty one. Furthermore $O = 3$ and $G = 2$. Every process needs to receive 4 VOTE messages, in order to make a decision. Processes p_2 and p_3 proposes the value 0 but processes p_4 and p_5 are proposing 1. As before the message of a process to itself is delayed and does not affect the all in all decision. In the first round process p_2 receives one message from p_3 with the proposed value of 0 and the proposed value of 1 from the other three processes. If *coin* is 1 the process p_2 will therefore decide on the value 1.
On the other hand process p_4 will receive 0 from p_1 to p_3 and 1 from p_5. If the *coin* is equal to 0 the process will decide on 0.
As in the previous exercise this violates the *Agreement Property* and the algorithm does not achieve the fault tolerance of $N > 4f$.

(c) Actual Resilience of *Gurucoin*

In order for a scenario to be able to violate the *Agreement Property*, the following must happen:

> One process which decides on a value, receives the proposed value of 0 from $\lceil \frac{N-f}{2} \rceil$ processes. The other $\lfloor \frac{N-f}{2} \rfloor$ non-faulty processes propose 1. In the end it has received [VOTE, R, 0] from $\lceil \frac{N-f}{2} \rceil + f$ processes and [VOTE, R, 1] from $\lfloor \frac{N-f}{2} \rfloor$ processes to totally receive $N - f$ messages. For another processes the faulty processes will VOTE for the other value and the opposite would be the case.

The two processes can only decide if $m \geq G$, therefore if

$$\lceil \frac{N-f}{2} \rceil + f < N - 3f$$

holds it is not possible that the processes decide on different values. For $N = kf + 1$ we have:

$$\lceil \frac{(k-1)f+1}{2} \rceil + f < (k-3)f + 1, \qquad k > 4 \text{ (from b)}$$

We now can compute this for every k until the inequality holds:

$\vdots$

$\underline{k = 7}$:

$$\lceil \frac{6f+1}{2} \rceil + f = 4f + 1 < 4f + 1 \textit{ is clearly wrong}$$

$\underline{k = 8}$:

$$\lceil \frac{7f+1}{2} \rceil + f < 5f + 1$$

$\underline{k = 9}$:

$$\lceil \frac{8f+1}{2} \rceil + f = 5f + 1 < 6f + 1$$

For every $k < 7$ the same would be the result as for $k = 7$ or $k = 8$. With these computations we can see that k must be $geq9$ s.t. the inequality holds.

PROBABILISTIC TERMINATION:
Clearly the algorithm will eventually terminate with a probability of 1.

STRONG VALIDITY:
If all processes propoose the same value, then $m \geq G$, thus all correct processes will decide on this value.

INTEGRITY:
Since the value of *decided* is set to TRUE once a decision is made, a correct process cannot decide more than once.