RELIABILITY OF DISTRIBUTED SYSTEMS

Mike Greenbaum and David Ponarovsky

January 2021

Exercises

1. In the trivial protocol, each of the parties draw a value uniformly from $\{0,1\}$, and output the sample. It's easy to see that the Agreement property holds, with probability $\alpha = \frac{1}{2^{n-f}}$ all the non faulty parties outputs the same value. For proving the Unpredictability property, consider the case in which the adv. controls over f parties. Denote by x_i the output of the the ith honest party. The draw process of any one of the parties is independent at the others running, (including communication) and therefore

$$\begin{aligned}
\mathbf{Pr}\left[x_i \text{ (event)}\right] &= \sum_{\text{others running}} \mathbf{Pr}\left[x_i \cap \text{others running}\right] \\
&= \sum_{\text{others running}} \mathbf{Pr}\left[x_i | \text{ others running}\right] \cdot \mathbf{Pr}\left[\text{ others running}\right] \\
&\overbrace{=}^{\text{independent}} \sum_{\text{others running}} \mathbf{Pr}\left[x_i \text{ (atomic)}\right] \cdot \mathbf{Pr}\left[\text{ others running}\right] = \left[x_i \text{ (atomic)}\right] = \frac{1}{2}
\end{aligned}$$

Hence the overall distribution over the non faulty outputs is a tensor product of independent running which is also a uniform and therefore the agreement is unpredictable.

2. (a) The Protocol:

Algorithm 1: Binary Asynchronous Byzantine Agreement

$x_0 \leftarrow$ given.
$g_0 \leftarrow 0$
$k \leftarrow 1$
while *live* **do**
  if $g_{k-1} = 0$ **then**
    $x_k, g_k \leftarrow BCA(x_{k-1}, k)$
  end
  else
    garbage, $g \leftarrow BCA(x_{k-1}, k)$
    $x_k, g_k \leftarrow x_{k-1}, \max\{g_{k-1}, g\}$
  end
  coin $\leftarrow$ *WeakCoin(k)*
  if g_k *is 2* **then**
    decide x_k *and send [decide, k,* x_k*] to all. (by reliable* ***Broadcast****)*
  end
  if x_k *is* $\perp$ **then**
    $x_k \leftarrow$ *coin.*
  end
  if *you hear* $n-2f$ *[decide,* $k^\star$*, x], for* $k^\star \leq k$ **then**
    decide x, and send [decide, k^**, x] to all. (by reliable* ***Broadcast****)*
  end
  if *you hear* $n-f$ *[decide,* $k^\star$*, x], for* $k^\star \leq k$ **then**
    terminate
  end
  $k \leftarrow k+1$
end

(b) Proof of the properties:

i. **Agreement** By the definition of the protocol, a non-faulty processor decides only if it got grade 2 or heard $n-2f$ messages $[decide, k, x]$. Notice that if a non-faulty processor heard $n-2f$ messages $[decide, k, x]$ then it heard at least one from a non-faulty processor.
By the agreement **GABCA** agreement, if a non-faulty processor outputs $(x, 2)$ then all non-faulty output (x, g) where $g \geq 1$.
Therefore, if one non-faulty got $(x, 2)$, no other value y will be chosen for any non-faulty processor which implies that all non-faulty decide messages contain x and therefore only value x can be decided.

ii. **Validity:** If all the parties have initialized with the same value, then by the validity property of **GABCA** all the non faulty parities will holds those values after the execution of **GABCA** and all non-faulty get grade 2 and decide.

iii. **Termination:** If one non-faulty terminates, it heard at least $n-f$ messages $[decide, k, x]$ which means it heard at least $n-2f$ non-faulty messages.
Therefore, all other non-faulty will hear at least $n-2f$ messages of the form $[decide, k, x]$ ans will send $[decide, k, x]$ themselves. Eventually, every non faulty will send $[decide, k, x]$ which means that every non-faulty will hear at least $n-f$ $[decide, k, x]$ and terminate itself.
Therefore, we explained why if one non-faulty terminates, all terminate.
If the adversary chooses to bind all $\perp$ then with probability α, all the non-faulty will get the same value and due to validity all get grade 2 and terminate. This takes $\frac{1}{\alpha}$ rounds in expectation.
If the adversary chooses to bind not b then with probability $\frac{1}{2}\alpha$, all the non-faulty will get the same value $1-b$ and due to validity all get grade 2 and terminate. This takes $\frac{2}{\alpha}$ rounds in expectation.
So in expectation, the algorithm will terminate after $\frac{2}{\alpha} = 2^{n-f+1}$ rounds.

3. (a) The protocol for processor i is the following:

i. Step 1: send $\langle val, x_i \rangle$

ii. Step 2: If you hear $\langle val, x \rangle$ from $f+1$ processors and you didn't send $\langle echo_1, x \rangle$ yet then send $\langle echo_1, x \rangle$ to everyone.

iii. Step 3:
First Time: if you hear $\langle echo_1, x \rangle$ from $n-f$ processors and you did not send any $\langle echo_2, * \rangle$, then send $\langle echo_2, x \rangle$ to all processors.
Second time: if you hear $\langle echo_1, x \rangle$ from $n-f$ processors and you already sent exactly one $\langle echo_1, y \rangle$ with $y \neq x$, then send $\langle echo_2, \perp \rangle$ to all processors.

iv. Step 4: if you hear $\langle echo_2, x \rangle$ from $f+1$ processors and you did not send any $\langle echo_3, * \rangle$ yet, then send $\langle echo_3, x \rangle$ to all processors.

v. Step 5: wait for $n-f$ $\langle echo_3, * \rangle$ messages then wait for either:
   A. $\langle echo_3, x \rangle$ from $n-f$ processors, then send $\langle echo_4, x \rangle$.
   B. $\langle echo_2, \perp \rangle$ from $n-f$ processors, then send $\langle echo_4, \perp \rangle$.

vi. Step 6: wait for $n-f$ $\langle echo_4, * \rangle$ messages then wait for
   A. $\langle echo_4, x \rangle$ from $n-f$ processors, then output $(x, 2)$
   B. $\langle echo_4, x \rangle$ from at least $f+1$ processors, then output $(x, 1)$
   C. $\langle echo_4, \perp \rangle$ from $n-f$ processors, then output $(\perp, 0)$

(b) We prove all the properties needed in Lemmas.

Lemma 1 At step 2, all non-faulty processors send $\langle echo_1, 0 \rangle$ or $\langle echo_1, 1 \rangle$ and not any other value.

Proof: Assume by contradiction that non-faulty processor i sends value $\langle echo_1, v \notin \{0, 1\} \rangle$, therefore due to how the protocol is defined, it heard $\langle val, v \rangle$ at least from $f+1$ processors.
Due to the fact that there are at most f faulty processors, i heard $\langle val, v \rangle$ from at least $f+1-f=1$ non-faulty processors. This is a contradiction because any non-faulty processor only sends $\langle val, x_i \rangle$ and $x_i \in \{0, 1\}$, therefore it couldn't have sent $\langle val, v \rangle$ where $v \notin \{0, 1\}$.

Lemma 2 At step 3, all non-faulty processors send $\langle echo_2, v \in \{0, 1, \perp\} \rangle$ and there doesn't exists a non-faulty s.t. it sends $\langle echo_2, 0 \rangle$ and $\langle echo_2, 1 \rangle$.

Proof: First notice that if non-faulty processor i sends $\langle echo_2, v \neq \perp \rangle$, then it heard at least $n-f \geq f+1$ messages of the form $\langle echo_1, v \rangle$. Therefore it heard at least $f+1-f=1$ messages of the form $\langle echo_1, v \rangle$

from non-faulty processors. Due to Lemma 1, We know that $v \in \{0,1\}$. Therefore, we proved that $v \in \{0,1,\perp\}$.
Assume by contradiction that non-faulty processor i sends $\langle echo_2, 0\rangle$ and $\langle echo_2, 1\rangle$. Assume wlog that i sent $\langle echo_2, 0\rangle$ before $\langle echo_2, 1\rangle$, therefore due to how the protocol is defined, after it sent $\langle echo_2, 0\rangle$ it should have sent only $\langle echo_2, \perp\rangle$ by contradiction that i sent $\langle echo_2, 1\rangle$.

Lemma 3 At step 4, all non-faulty processors send $\langle echo_3, v \in \{0,1\}\rangle$ and there doesn't exists a non-faulty s.t. it sends $\langle echo_3, 0\rangle$ and $\langle echo_3, 1\rangle$.
Proof: First notice that if non-faulty processor i sends $\langle echo_2, v\rangle$, then it heard at least $f+1$ messages of the form $\langle echo_2, v\rangle$. Therefore it heard at least $f+1-f=1$ messages of the form $\langle echo_2, v\rangle$ from non-faulty processors. Because we ignore messages of the type $\langle echo_2, \perp\rangle$, due to Lemma 2 we get that $v \in \{0,1,\perp\} \setminus \{\perp\} = \{0,1\}$.
Assume by contradiction that non-faulty processor i sends $\langle echo_3, 0\rangle$ and $\langle echo_3, 1\rangle$. Assume wlog that i sent $\langle echo_3, 0\rangle$ before $\langle echo_3, 1\rangle$, therefore due to how the protocol is defined, after it sent $\langle echo_2, 0\rangle$ it should have sent any messages by contradiction that i sent $\langle echo_2, 1\rangle$.

Lemma 4 Termination: if all non-faulty parties start the protocol, then all non-faulty parties output a value and terminate. We will reach termination in a constant number of rounds.
The protocol is live. (i.e. termination after a certain amount of rounds)
Proof: We will show that the protocol reaches the step of the protocol.
Proof that the algorithm will reach Step 2:
Notice that $n-f$ no-faulty processors send $\langle val, x_i \in \{0,1\}\rangle$ in Step 1 and that $n-f \geq 2\cdot(f+1)$ and therefore, by the pigeon principle, $\exists y \in \{0,1\}$ s.t. at least $f+1$ messages are $\langle val, y\rangle$ and therefore, each non-faulty will hear $f+1$ messages of the form $\langle val, y\rangle$ and will reach step 2 and therefore each non-faulty processor will hear at least $n-f$ messages of form $\langle val, y\rangle$ and will send $\langle echo_1, y\rangle$.
Proof that the algorithm will reach Step 3:
Notice that all non-faulty processors hear $\langle echo_1, y\rangle$ from $n-f$ non-faulty processors and therefore will reach Step 3.
Proof that the algorithm will reach Step 4:
As we proved in Lemma 2, every non-faulty process will send at least one of $\langle echo_2, 0\rangle$ or $\langle echo_2, 1\rangle$. Notice that there are at least $n-f \geq 2(f+1)$ non-faulty messages from step 3 in the format mentioned above. Therefore, Due to the pigeon principle, one of the messages $\langle echo_2, 0\rangle$ or $\langle echo_2, 1\rangle$ arrived at least $f+1$ times and we and we will reach step 4.
Proof that the algorithm will reach Step 5:
Notice that every non-faulty process will reach step 4 and will send a message of the format $\langle echo_3, *\rangle$ and therefore we will eventually get $n-f$ messages of the format $\langle echo_3, *\rangle$.
If in step 5, there are $n-f$ messages of the format $\langle echo_3, v\rangle$ then we send $\langle echo_4, x\rangle$ and reach step 5.
Else, it means that there are 2 non-faulty processors i, j that i sent $\langle echo_3, 0\rangle$ and j sent $\langle echo_3, 1\rangle$ by Lemma 3. Therefore, by definition of step 4, i heard at least $f+1$ messages of the form $\langle echo_2, 0\rangle$ and j heard at least $f+1$ messages of the form $\langle echo_2, 1\rangle$. Therefore, by definition, i heard at least $f+1-f=1$ messages of the form $\langle echo_2, 0\rangle$ from non-faulty processors and j heard at least $f+1$ messages of the form $\langle echo_2, 1\rangle$ from non-faulty processors.
Therefore, by definition of step 3, there is a process i' that heard $n-f$ at least $n-f$ messages of the form $\langle echo_1, 0\rangle$ and there is a process j' that heard $n-f$ at least $n-f$ messages of the form $\langle echo_1, 1\rangle$.
Therefore, process i' that heard $n-f$ at least $n-f-f \geq f+1$ messages of the form $\langle echo_1, 0\rangle$ from non-faulty processors and there is a process j' that heard $n-f-f \geq f+1$ at least $n-f$ messages of the form $\langle echo_1, 1\rangle$ from non-faulty processors.
Therefore, in step 2, at least $f+1$ non-faulty processors sent $\langle echo_1, 0\rangle$ and at least $f+1$ non-faulty processors sent $\langle echo_1, 1\rangle$. Therefore, eventually, every non-faulty processor will send both $\langle echo_1, 0\rangle$ and $\langle echo_1, 1\rangle$.
Therefore, in step 3, all non-faulty processors will send $\langle echo_2, \perp\rangle$ which means eventually there will be $n-f$ messages of the form $\langle echo_2, \perp\rangle$ and then the protocol will reach step 5.
Proof that the algorithm will reach Step 6:
Notice that all non-faulty processors will end step 5 due to what we proved before and then at least $n-f$ non-faulty processors will send $\langle echo_4, *\rangle$ and we will reach step 6 and end the protocol.

Lemma 5 There doesn't exists 2 non-faulty processors s.t. one outputs $v \neq \perp$ and the second one outputs $u \notin \{v, \perp\}$

Proof: Assume by contradiction that exists processes i, j s.t. i outputs value 0 and j outputs value 1. (Due to Lemma 3, we know that these are the only allowed values except $\perp$).

Then, i heard at least from $f+1$ processors the message $\langle echo_4, 0\rangle$ and j heard at least from $f+1$ processors the message $\langle echo_4, 1\rangle$.

Then, i heard at least from $f+1-f=1$ non-faulty processors the message $\langle echo_4, 0\rangle$ and j heard at least from $f+1-f=1$ non-faulty processors the message $\langle echo_4, 1\rangle$.

Therefore, by definition of Step 5, there is a non-faulty processor i' that sent $\langle echo_4, 0\rangle$ and a non-faulty processor i' that sent $\langle echo_4, 1\rangle$.

Therefore, by definition, i' heard at least from $n-f$ processors $\langle echo_3, 0\rangle$ and j' heard at least from $n-f$ processors $\langle echo_3, 1\rangle$.

Therefore, by quorum, there is a non-faulty processor that sent $\langle echo_3, 0\rangle$ and $\langle echo_3, 1\rangle$ which is a contradiction to Lemma 3.

Lemma 6 Binding property: The Bayesian opponent needs to choose the responses of the non-faulty between 1. no zeros 2. no ones 3. everyone outputs $\perp$.

Proof: This follows immediately from lemma 5, due to the fact that there can't be 2 non-faulty that one outputs 0 and the other outputs 1.

Lemma 7 Validity: If all non-faulty processor start with x they all output $(x, 2)$.

if a non-faulty processor outputs $x \neq \perp$, then some non-faulty processor had x as input.

If a non-faulty processor outputs a value with grade 2, then all non-faulty processors will output this value.

Proof: If all $n-f$ non-faulty processors have input x, then all of them send $\langle val, x\rangle$. Then in step 3, all $n-f$ non-faulty processors will send $\langle echo_2, x\rangle$. Then in step 4, all $n-f$ non-faulty processors will send $\langle echo_3, x\rangle$. Then in step 5, all $n-f$ non-faulty processors will send $\langle echo_4, x\rangle$. Finally, all $n-f$ non-faulty processors will output $(x, 2)$. (Assuming that the Bayesian opponent can't inject $\langle echo_4, *\rangle$ messages on his own, or else the proof in class also doesn't satisfy validity).

Assume that i is a non-faulty processor that outputs x. Due to how the algorithm works, someone had to hear $\langle val, x\rangle$ from $f+1$ processors (else the algorithm will not see x after step 1). Therefore, it heard at least one 1 non-faulty processor that sent $\langle val, x\rangle$ and therefore x is an input of a non-faulty processor.

If non-faulty processor chose $(v, 2)$, then all non-faulty parties chose $(v, 1)$ (because they heard from at least $n-f-f \geq f+1$ non-faulty $\langle echo_4, v\rangle$) and the next iteration everyone will agree on v.

Lemma 8 Agreement: If a non-faulty processor outputs $(x, 2)$ then all non-faulty processors output (x, g) where $g \geq 1$.

Proof: All we need to show due to Lemma 5 is that there is no non-faulty processor that outputs $\perp$. Assume by contradiction that i is a non-faulty processor that outputs $(\perp, 0)$. Therefore, i heard $n-f$ messages of form $\langle echo_4, \perp\rangle$ and the processor that outputted $(x, 2)$ heard $n-f$ messages of form $\langle echo_4, x\rangle$.

Therefore, by quorum, exists a non-faulty processor i' that sent both $\langle echo_4, x\rangle$ and $\langle echo_4, \perp\rangle$ which is a contradiction because each non-faulty sends at most 1 message of type $echo_4$.

(c) Notice that in each round, each non-faulty processor sends at most $O(1)$ bits to each other processor.

Therefore, in each round, each non-faulty processor sends at most $O(n)$ bits.

Therefore, in each round, the non-faulty processors send at most $O(n \cdot n) = O(n^2)$ bits.

Due to the number of rounds being constant, we get that the number of bits sent in the protocol is $O(n^2)$.