Lecture 10

Consensus

One of the key components in the self-stabilizing pulse synchronization algorithm we will see two lectures from now is *consensus.* Consensus is a fundamental and extremely well-studied fault-tolerance primitive. There are a large number of variants of the problem, varying in terms of the model and the requirements on the solution. The common theme is the following question: In a system with faults, how can the non-faulty nodes agree on a decision that is consistent with given inputs?

Today, we study a very basic formulation of the consensus problem. We assume a synchronous system (like in Chapter 5 for approximate agreement) with n nodes and $f < n/3$ Byzantine faults, where nodes have unique identifiers $1, \dots, n$ known to all nodes. Each node is given a binary input $b_i \in \{0, 1\}$. To solve (binary) consensus, an algorithm must compute output values $o_i \in \{0, 1\}$ at all correct nodes $i \in V_g$ meeting the following conditions:

Agreement: There is $o \in \{0, 1\}$ so that $o_i = o$ for all $i \in V_g$. We refer to o as the output of the consensus algorithm.

Validity: If there is $b \in \{0, 1\}$ so that for all $i \in V_g$ it holds that $b_i = b$, then $o = b$.

Termination: There is $r \in \mathbb{N}$ satisfying that each $i \in V_g$ terminates and outputs o_i by the end of round r.

In general, the round r when all correct nodes have terminated may depend on the execution. However, we are interested in algorithms which guarantee termination within $R(f) \in \mathbb{N}$ rounds, regardless of the inputs and the behavior of faulty nodes. We refer to $R(f)$ as the *running time* or *round complexity* of the algorithm.

Remarks:

- For $f \geq n/3$, no algorithm solves consensus deterministically. The reasons are very similar to what we saw in Chapter 4.
- Even randomized algorithms fail with a large probability. One could say that the $n/3$ barrier is "hard." However, under cryptographic assumptions one can "force" faulty nodes to communicate consistently (either not sending a message or sending the same to everyone), so long as $f < n/2$. Then,

in the synchronous model, the task is trivial: All nodes broadcast their input, and choose the output as a function of the received values ensuring validity.

- Beside the round complexity, we care about the amount of communication. Relevant criteria here are the maximum message size (i.e., the number of bits in the largest message an algorithm uses) and the number of bits nodes send in total.

10.1 The Phase King Algorithm

Algorithm 10.1: Phase King Algorithm at node $i \in V_g$. Each broadcast takes one round. Note that faulty nodes are not required to broadcast, i.e., they can send conflicting messages to different nodes.

```
1  op_i := b_i
2  for j = 1 ... f + 1 do
3  |  // first broadcast
4  |  strong := 0
5  |  broadcast op_i (also to self)
6  |  if received at least n − f times b ∈ {0,1} then
7  |  |  op_i := b
8  |  |  strong := 1
9  |  // second broadcast
10 |  if strong = 1 then
11 |  |  broadcast op_i
12 |  if received fewer than n − f times op_i then
13 |  |  strong := 0
14 |  // king's broadcast part I
15 |  if i = j and received at least f + 1 times b ∈ {0,1} then
16 |  |  broadcast b
17 |  // king's broadcast part II
18 |  if i = j and received at most f times b ∈ {0,1} then
19 |  |  broadcast op_i
20 |  // if not sure, obey the king
21 |  if strong = 0 and received b ∈ {0,1} from node j then
22 |  |  op_i := b
23 return op_i
```

In the above algorithm, we refer to one iteration of the loop as a *phase.*

Lemma 10.1. *If, for some $b \in \{0,1\}$ and all $i \in V_g$, $op_i = b$ at the beginning of the phase, then the same holds at the end of the phase.*

Proof. As $|V_g| \geq n - f$, each $i \in V_g$ will not change op_i and set strong to 1 after the first broadcast. Thus, in the second broadcast, $|V_g| \geq n - f$ nodes will broadcast b, and all correct nodes will maintain strong $= 1$. Thus, op_i is not changed by the king's broadcast. □

Corollary 10.2. *Algorithm 10.1 satisfies validity.*

Proof. Suppose $b_i = b$ for some $b \in \{0,1\}$ and all $i \in V_g$. Then each $i \in V_g$ initializes $op_i := b$, which by inductive use of Lemma 10.1 never changes. Thus each $i \in V_g$ outputs b. □

Lemma 10.3. *Suppose node $j \in V_g$. Then there is some $b \in \{0,1\}$ so that $op_i = b$ for all $i \in V_g$ at the end of phase j.*

Proof. Fix the phase to be j. We claim that there is $b \in \{0,1\}$ satisfying that each $i \in V_g$ with strong $= 1$ after the first broadcast satisfies $op_i = b$. Otherwise, as correct nodes broadcast, it would hold that

$$2n - 2f = 2(n-f) \leq |V_g| + 2(n - |V_g|) = 2n - |V_g|\,,$$

i.e., $|V_g| \leq 2f < n - f$, as $f < n/3$ — which implied that there are $n - |V_g| > f$ faulty nodes.

Thus, only faulty nodes may send a value different from b in the second broadcast. We distinguish two cases. The first is that there is no $i \in V_g$ with strong $= 1$ after the second broadcast. In this case, each $i \in V_g$ sets $op_i := b' \in \{0,1\}$, where b' is the value broadcasted by the king, i.e., node j.

The other case is that some node received $n - f$ times b in the second broadcast, the king (i.e., node j) received at least $n - 2f \geq f + 1$ times b. On the other hand, there are at most f faulty nodes, so the king did not receive more than f times $1 - b$. It follows that the king broadcasts b in the king's broadcast. As $f < n - f$, any $i \in V_g$ with $op_i = 1 - b$ satisfies that strong $= 0$ when receiving this message and sets $op_i := b$. □

Corollary 10.4. *Algorithm 10.1 satisfies agreement.*

Proof. As there are at most f faults, $[f+2] \cap V_g \neq \emptyset$. Let $j \in [f+2] \cap V_g$. By Lemma 10.3, at the end of phase j, we have that there is some $b \in \{0,1\}$ so that $op_i = b$ for all $i \in V_g$. By inductive use of Lemma 10.1, these variables do not change any more. Hence all $i \in V_g$ output b. □

Theorem 10.5. *Algorithm 10.1 solves binary consensus in the synchronous model. It runs for $R(f) = 3(f+1) \in O(f)$ rounds and broadcasts 1-bit messages.*

Proof. Agreement and validity hold by Corollary 10.4 and Corollary 10.2, respectively. The running time bound follows from the facts that each phase takes three rounds, one for each broadcast, and that there are $f + 1$ phases. The message size bound is immediate from the pseudocode. □

Remarks:

- Depending on the precise model of communication, message size may also be 2 bits; the bound above exploits the option of sending no message, which may not always be possible.
- The message size is trivially optimal — but that does not mean the *overall* number of communicated bits is.
- Deterministic algorithms need to send $\Omega(nf)$ bits, which follows from the simple observation that each correct node needs to receive more than f bits to even know that anyone else would pick a certain output value.

- The running time is asymptotically optimal: no algorithm can be faster than $f+1$ rounds in the worst case. We will show this later in this lecture.
- Both the bit complexity and running time lower bound can be beaten by randomized algorithms; this is beyond the scope of this lecture, though.

10.2 Recursive Phase King

The Phase King protocol is doing well in terms of running time and resilience, i.e., f. However, it uses much more communication than the (trivial) lower bound of $\Omega(nf)$ requires. We can overcome this by avoiding to have all nodes communicate to each other for f times. The key to achieving this is to make sure that the "king" is more likely to be reliable. We do this by calling the protocol on roughly half of the participating nodes. There are not enough faulty nodes to make both instances fail, and in the recursive calls, fewer nodes need to send and receive messages. In the following recursive variant of the protocol, "broadcast" means to sent a message to all nodes participating in the instance.

Lemma 10.6. *Algorithm 10.2 satisfies agreement and validity.*

Proof. We reason similarly to our analysis of Algorithm 10.1, with the difference that the role of the "king" is now filled by V'. We show the claim by induction on $|V|$; it is trivial for $|V| = 1$. For $|V| > 1$, observe that the statement of Lemma 10.1 can be shown analogously for Algorithm 10.2.

Next, note that f is the number of faults we expect Algorithm 10.2 to withstand. If $|V| > 1$, define that a recursive call is successful if $|V'| > 3|V' \setminus V_g|$, i.e., there are few enough faults such that validity and agreement hold for the recursive call by the induction hypothesis. We show that at least one of the two recursive calls is successful. To this end, recall that for $j = 1$ V' is chosen such that the recursive call is successful, if $|V' \setminus V_g| \leq \lceil (f-1)/2 \rceil$. Concerning $f = 2$, note that

$$3\left\lceil \frac{f-1}{2} \right\rceil + 1 + 3\left\lfloor \frac{f-1}{2} \right\rfloor + 1 = 3(f-1) + 2 < 3f < n\,.$$

Hence, the second recursive call is successful, if $|V' \setminus V_g| \leq \lfloor (f-1)/2 \rfloor$. Overall, as there are at most

$$f < f + 1 = \left\lceil \frac{f-1}{2} \right\rceil + 1 + \left\lfloor \frac{f-1}{2} \right\rfloor + 1$$

faults, at least one of the recursive calls is successful.

Now recall Lemma 10.3. Instead of requiring that $j \in V_g$, we instead demand that the j^{th} recursive call succeeds. We proceed as in the proof of Lemma 10.3 until the case distinction. If there is no correct node with strong $= 1$, then by the agreement property, all correct nodes in V' broadcast the same value $b \in \{0, 1\}$, and because $|V'| > 3|V' \setminus V_g|$, this is for each node the majority value received from nodes in V'. Thus, each node sets $op_i = b$, as desired. On the other hand, if there is a correct node i with strong $= 1$, then the same reasoning as in Lemma 10.3 shows that each node in $V' \cap V_g$ uses input op_i for the recursive call. By validity, this means that this is the output of the recursive call, which is broadcasted to all nodes by the majority of nodes in

Algorithm 10.2: Recursive Phase King Algorithm at node $i \in V_g$. For simplicity, recursive calls on k nodes assume the identifiers to be $\{1, \dots, k\}$.

```
1  if |V| = 1 then
2  |  return b_i
3  op_i := b_i
4  for j = 1, 2 do
5  |  strong := 0
6  |  broadcast op_i (also to self)
7  |  if received at least n − f times b ∈ {0, 1} then
8  |  |  op_i := b
9  |  |  strong := 1
10 |  if strong = 1 then
11 |  |  broadcast op_i
12 |  if received fewer than n − f times op_i then
13 |  |  strong := 0
14 |  // recursive call
15 |  f := ⌈n/3⌉ − 1
16 |  if j = 1 then
17 |  |  V' := {1, ..., 3⌈(f − 1)/2⌉ + 1}
18 |  else
19 |  |  V' := {3⌈(f − 1)/2⌉ + 2, ..., |V|}
20 |  if i ∈ V' then
21 |  |  b'_i := 0
22 |  |  if received at least f + 1 times 1 then
23 |  |  |  b'_i := 1
24 |  |  denote by o'_i the output of recursive call on node set V' with
   |  |    inputs b'_i
25 |  |  broadcast o'_i
26 |  if strong = 0 then
27 |  |  set op_i to majority value received from nodes in V' (breaking a
   |  |    tie arbitrarily)
28 return op_i
```

V'. We conclude that Lemma 10.3 applies to Algorithm 10.2 with the above modification to the statement.

As we have also shown that at least one of the recursive calls succeeds, agreement and validity follow as in Corollaries 10.2 and 10.4, respectively. □

Lemma 10.7. *Algorithm 10.2 terminates in $\mathcal{O}(n)$ rounds.*

Proof. We claim that the total number of (recursive) calls of Algorithm 10.2 is $2n - 1$. This follows from the fact that the resulting binary recursion tree has n leafs (the instances with a single node) and each inner node except for the root has degree 3, i.e., the number of leaves equals the number of inner nodes plus 2.

As, apart from the recursive calls, each instance requires 6 rounds, the claim follows. □

Lemma 10.8. *The total number of bits communicated by the nodes in V_g when executing Algorithm 10.2 is $\mathcal{O}(n^2)$.*

Proof. Denote by $B(n)$ the number of bits communicated in an instance with n nodes. Clearly, in six rounds of communication, at most $6n^2$ bits are sent. Hence, $B(n) \leq 6n^2 + B(n_1) + B(n_2)$, where $n_1 + n_2 = n$. Note that in the algorithm $f \in n/3 + \Theta(1)$ and, in each of the two recursive calls, $|V'| \in 3f/2 + \Theta(1)$. Hence, if n is at least a sufficiently large constant, we have that $\max\{n_1, n_2\} \leq 2n/3$. Otherwise, the remaining number of recursive calls is constant, and we can bound $B(n) \in \mathcal{O}(n^2)$. Therefore, $B(n) \in 2B(2n/3) + \mathcal{O}(n^2)$ for all $n \in \mathbb{N}$. By the master theorem, this implies that $B(n) \in \mathcal{O}(n^2)$. □

Theorem 10.9. *Binary consensus in systems with $f < n/3$ faults can be solved in $\mathcal{O}(f)$ rounds with 1-bit messages, where the total number of communicated bits is $\mathcal{O}(nf)$.*

Proof. For $f \in \Omega(n)$, this is shown for Algorithm 10.2 by Lemmas 10.6, 10.7, and Lemma 10.8. For smaller values of f, run the algorithm on $3f + 1$ nodes, have them broadcast their output, and let each node output the majority value. This adds one round and $\mathcal{O}(fn)$ bits to the previously spent $\mathcal{O}(f)$ rounds and $\mathcal{O}(f^2)$ bits. □

10.3 Running Time Lower Bound

As promised earlier, we prove now that any (deterministic) consensus algorithm must run for at least $f + 1$ rounds in the worst case. In fact, we will show this for a much weaker fault model: crash faults.

Definition 10.10 (Crash Faults)**.** *If node $v \in V$* crashes in round $r \in \mathbb{N}$, *it operates like a non-faulty node in rounds $1, \ldots, r-1$, does nothing at all in rounds $r+1, r+2, \ldots$, and in round r sends an arbitrary subset of the messages it would send according to the algorithm.*

Crashing nodes fail in a well-organized fashion. They do not lie, we do not have to care about getting them up to speed again later, and by requiring that nodes always send messages to each other in each round, nodes will learn that a node failed from not receiving a message from the node. None of this affects the worst-case running time lower bound in any way — regardless of whether we consider Byzantine or crash faults, the bound of $f + 1$ rounds turns out to be tight.

We will show this lower bound now by a straightforward inductive argument. The key ingredient is the following definition.

Definition 10.11 (Pivotal Nodes)**.** *Observe that an execution in the synchronous model with crash faults is fully determined by specifying the node inputs and, for each node, whether it crashes and, if so, in which round and which of its messages of this round get sent. Given an execution $\mathcal{E}$ of a consensus algorithm with at most $n-2$ crash faults and a node $v \in V$ that does not crash in $\mathcal{E}$, we call v* pivotal in round r (of $\mathcal{E}$) *if changing $\mathcal{E}$ by crashing v in round r of $\mathcal{E}$ without v sending any messages results in an execution with different output (the execution does have an output, because at least one node does not crash).*

In order to anchor the induction, we need to show that such nodes exist.

Lemma 10.12. *There is a fault-free execution with a node that is pivotal in round* 1*.*

Proof. Consider executions $\mathcal{E}_i$, $i \in [n+1]$, which are fault-free with node $j \in V$ having input 0 if $j > i$ and input 1 otherwise. By validity, $\mathcal{E}_0$ has output 0 and $\mathcal{E}_1$ has output 1. Thus, there must be some $i \in [n]$ with the property that $\mathcal{E}_i$ has output 0 and $\mathcal{E}_{i+1}$ has output 1. Consider the execution $\mathcal{E}'$ obtained by crashing node $i+1$ in round 1, without $i+1$ getting any messages out. If $\mathcal{E}'$ has output 0, $i+1$ is pivotal in round 1 of execution $\mathcal{E}_{i+1}$; if $\mathcal{E}'$ has output 1, $i+1$ is pivotal in round 1 of execution $\mathcal{E}_i$. □

The induction step works the same way, except that the inputs are replaced by, for each node, the decision whether the pivotal node crashing in round r sends a message to the node or not.

Lemma 10.13. *Suppose* $0 \leq f \leq n-3$ *and* $\mathcal{E}$ *is an execution with* f *failing nodes, one in each round* $1, \ldots, f$*, that has a pivotal node in round* $f+1$*. Then there is an execution* $\mathcal{E}'$ *which differs from* $\mathcal{E}$ *only in that this pivotal node crashes in round* $f+1$ *and satisfies that there is a pivotal node in round* $f+2$*.*

Proof. For $i \in [n+1]$, define $\mathcal{E}_i$ by having the pivotal node of $\mathcal{E}$ crash in round $f+1$ and succeed in sending its message for that round to node $j \in \{1, \ldots, n\}$ if and only if $j > i$. As we crashed a pivotal node, we know that $\mathcal{E}_0$ and $\mathcal{E}_n$ have different outputs. Thus, there must be some i for which $\mathcal{E}_i$ and $\mathcal{E}_{i+1}$ have different outputs. Now consider the executions $\mathcal{E}'_i$ and $\mathcal{E}'_{i+1}$ obtained from $\mathcal{E}_i$ and $\mathcal{E}_{i+1}$, respectively, in which node $i+1$ crashes in round $f+2$ without sending any messages. The only difference between these executions is whether $i+1$ received the message from the crashing node in round $f+1$ or not; as $i+1$ does not get a message out telling anyone of this difference, the outputs of $\mathcal{E}'_i$ and $\mathcal{E}'_{i+1}$ are the same. Thus, either $\mathcal{E}_i$ and $\mathcal{E}'_i$ have different outputs or $\mathcal{E}_{i+1}$ and $\mathcal{E}'_{i+1}$ have different outputs, i.e., either $i+1$ is pivotal in round $f+2$ of $\mathcal{E}_i$ or it is pivotal in round $i+1$ or $\mathcal{E}_{i+1}$. □

Corollary 10.14. *Any consensus algorithm has an execution with a pivotal node in round* $\min\{f, n-2\}$*.*

Theorem 10.15. *Any consensus algorithm has worst-case running time at least* $\min\{f+1, n-1\}$*.*

Proof. Consider the execution $\mathcal{E}$ with a pivotal node in round $\min\{f, n-2\}$ guaranteed to exist by Corollary 10.14, as well as the execution $\mathcal{E}'$ obtained by crashing the pivotal node in round $\min\{f, n-2\}$. The two executions have different output, but at all nodes but the pivotal one, the only difference to be observed before round $\min\{f+1, n-1\}$ is whether the respective message from the pivotal node in round $\min\{f, n-2\}$ was received or not.

Assume for contradiction that, in both executions, the (at least two) non-crashed nodes terminate by the end of round $\min\{f, n-2\}$. Let $i, j \in V$ be two such nodes crashing in neither $\mathcal{E}$ nor $\mathcal{E}'$. These nodes must also terminate in the execution $\mathcal{E}''$ in which the pivotal node sends its message to i, but does not send its message to j: To i, this execution is indistinguishable from $\mathcal{E}$ before round $\min\{f+1, n-1\}$, and for j it is indistinguishable from $\mathcal{E}$. However, this

indistinguishability implies that they also output the same values as in $\mathcal{E}$ and $\mathcal{E}'$, respectively. As these values differ, this violates agreement and hence is a contradiction. We conclude that our assumption must be wrong and there is some execution of the algorithm in which not all nodes terminate before round $\min\{f+1, n-1\}$. □

Remarks:

- The above proof was for binary consensus, but it also works if more than two output values are possible. The salient point is that there are different outputs!

- The same arguments apply even if we restrict the fault model *further*. We could require that, in each round, nodes send their outgoing messages in some specific order; the algorithm could even choose. Thus, a crash would not result in an arbitrary subset of nodes receiving their messages, but rather a prefix of the message sequence being received. Still, the same lower bound applies, with the same proof, where the only difference is that the order in which we list the nodes when defining executions is given by the pivotal nodes' sending sequence for the respective round.

- As mentioned before, randomization can result in faster algorithms.

Bibliographic Notes

Consensus and its variants is a central problem in distributed computing. Thus, any list of references would be no more than a scratch in the tip of the iceberg. Some books addressing the topic are [Lyn96, Ray10]. The Phase King protocol by Berman, Garay, and Perry was introduce in [BGP89]. The recursive version is provided in [BGP92]. The time lower bound was shown by Fischer and Lynch [FL82].

A lower bound of $\Omega(nf)$ on the message complexity is shown in [DR85]. Arguably, this bound is trivial, but the interesting part is that the paper shows a tight bound of $\Theta(n + f^2)$ on the number of messages required with cryptographic signatures. In this case, the total number of bits is still $\Omega(nf)$, but the signatures permit to prove that an output is valid with a single message. As a remark on connectivity requirements, both with and without cryptography node degrees must be larger than f. However, without cryptography, it's not hard to see that the majority of neighbors of each non-faulty nodes must be non-faulty, while with cryptographic signatures, it is sufficient, if the network is still connected when removing faulty nodes. With cryptographic assumptions, it is also sufficient if $f < n/2$. Naturally, all of this requires the ability to perform decoding and encoding operations, cryptographic hardness assumptions (i.e., the adversary can't break protocols by brute force), and that the adversary can't directly obtain information about internal states of correct nodes.

Bibliography

[BGP89] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards Optimal Distributed Consensus (Extended Abstract). In *Symposium on Foundations of Computer Science (FOCS)*, pages 410–415, 1989.

[BGP92] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. *Bit Optimal Distributed Consensus*, pages 313–321. Springer US, 1992.

[DR85] Danny Dolev and Rüdiger Reischuk. Bounds on Information Exchange for Byzantine Agreement. *J. ACM*, 32(1):191–204, 1985.

[FL82] Michael J. Fischer and Nancy A. Lynch. A Lower Bound for the Time to Assure Interactive Consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982.

[Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., 1996.

[Ray10] Michel Raynal. *Fault-tolerant Agreement in Synchronous Message-passing Systems*. Morgan & Claypool Publishers, 2010.