COMS 6998-006 (Foundations of Blockchains)
Homework Set 2 (Sol)

Matin M.Babaei Parsa Dini

September 9, 2023

Problem 1

A. Reduction of BA to BB (In Asynchronous Model with $f < n/3$):

We want to show that given a Byzantine Broadcast protocol (BB) that satisfies validity and agreement, we can construct a Byzantine Agreement protocol (BA) that also satisfies validity and agreement.

Construction of the BA Protocol:

Assumptions:

We have a Byzantine Broadcast (BB) protocol that guarantees validity and agreement. Steps of the Construction:

Each node, acting as a sender, broadcasts its input to all other nodes using the BB protocol. As receivers, nodes in the BA protocol receive the inputs broadcasted by all other nodes using the BB protocol. The nodes follow the BB protocol's validity and agreement properties to determine the majority value among the received inputs. The majority value is taken as the output of the Byzantine Agreement protocol.

Proof of Validity:

If all honest nodes have the same input (which is required for validity in BA), the BB protocol's validity guarantees that all nodes will receive this input. Thus, all nodes will agree upon this input as the majority value.

Proof of Agreement:

The BB protocol's agreement property ensures that all honest nodes will receive the same input if the sender is honest. Since the nodes agree upon the majority value, it means they will all agree upon the same input.

The FLP impossibility result states that in an asynchronous system, it is impossible to have a deterministic consensus protocol that ensures both termination and agreement in the presence of even one faulty process.

However, in the asynchronous model with $f < n/3$, the reduction of BA to BB shows that consensus can be achieved when the number of faulty processes is less than one-third of the total. This implies that, in this specific context, the BB problem can be solved deterministically, which contradicts the FLP result.

Therefore, the reduction of Byzantine Agreement to Byzantine Broadcast in the asynchronous model with $f < n/3$ demonstrates that the BB problem can be solvable in scenarios where the FLP impossibility result would suggest otherwise. This highlights the significance of the specific conditions and assumptions in determining the feasibility of consensus in distributed systems.

B. No, the reduction from Byzantine Broadcast (BB) to Byzantine Agreement (BA) that was shown to work in the synchronous model does not directly extend to the asynchronous model. There are

fundamental differences between the synchronous and asynchronous models that make the reduction not straightforward or feasible in the asynchronous context.

In the synchronous model, all processes take steps in lockstep, and message delays are bounded. This enables the reduction to work by having each node broadcast its input to all other nodes, and then having nodes agree on the majority value of the received inputs. The synchronized nature of the model ensures that each node receives messages from all others in a known and consistent timeframe.

However, in the asynchronous model, processes do not have synchronized steps, and message delays are unbounded. This lack of synchronization and the potential for unpredictable message delivery times pose significant challenges. The reduction would require solving the Byzantine Broadcast problem in the asynchronous model, which itself is a complex and difficult problem.

In the asynchronous Byzantine Broadcast problem, achieving agreement on a single message among nodes is challenging due to the lack of synchronization and potential for unpredictable message delays. This makes it difficult to devise a protocol that guarantees all nodes receive the same message in a timely manner, especially in the presence of faulty nodes.

While the reduction from BB to BA works in the synchronous model, it doesn't extend naturally to the asynchronous model due to the fundamental differences in the nature of these models. Achieving consensus and agreement is significantly more challenging in the asynchronous context, where the FLP impossibility result comes into play.

It is notable that in the asynchronous model, Byzantine broadcast turns out to be trivially unsolvable. Actually, this is a good exercise for you to work out. Intuitively, honest non-senders cannot distinguish between a Byzantine sender who is giving them the silent treatment and an honest sender whose messages have been delayed a very long time. Thus, a protocol which generally does not even exist can not be reduced.

C. No deterministic BB protocol always terminates while satisfying both validity and agreement in the asynchronous model, even for f = 1:

In the asynchronous model, it is not possible to have a deterministic BB protocol that always terminates while satisfying both validity and agreement, even when there is only one faulty node (f = 1). This is because, in the absence of any messages from the adversarial sender, honest nodes cannot distinguish between a slow or crashed sender, and they must wait indefinitely to ensure validity and agreement.

D. We prove this by using the FLP impossibility result and reducing the State Machine Replication (SMR) problem to the Byzantine Agreement (BA) problem:

Proof: FLP Impossibility and SMR Protocol

Suppose we have a protocol for State Machine Replication (SMR) that satisfies both consistency and liveness in the asynchronous model, even for f = 1.

1. FLP Impossibility Result:

The FLP impossibility result states that in the asynchronous model, no deterministic consensus protocol can ensure both termination and agreement if even a single process is faulty.

2. Reduction from SMR to BA:

We can use the FLP impossibility result to show that there is no protocol for SMR that satisfies both consistency and liveness, even for f = 1.

- Assume there exists a protocol for SMR that satisfies both consistency and liveness for f = 1.

- We can construct a Byzantine Agreement (BA) problem using this SMR protocol:

1. Each node in the SMR network simulates a process in the BA problem.
2. Each SMR command to be replicated can be considered as a proposed value in the BA problem.
3. Nodes follow the SMR protocol to replicate and agree upon a sequence of commands.
4. The output of the SMR protocol is the sequence of agreed-upon commands.

- By using the SMR protocol, we have essentially constructed a deterministic BA protocol that satisfies both validity and agreement, which contradicts the FLP impossibility result. Since the FLP impossibility result has been proved and shown to be true, The contradiction between the existence of the SMR protocol and the FLP impossibility result implies that there is no protocol for SMR that can simultaneously guarantee both consistency and liveness in the asynchronous model, even when there is only one faulty process.

  Using the FLP impossibility result, we've demonstrated that it is impossible to have an SMR protocol that satisfies both consistency and liveness in the asynchronous model, even for f = 1. This highlights the inherent challenges and limitations of achieving consensus and replication in the presence of faults and asynchrony.

Problem 2

A. Proof for n = 2 (and f = 1):

In this scenario, we have two nodes (n = 2), and one of them can fail (f = 1). To prove that consensus is impossible in this case, we can consider the following explicit adversary strategies:

Node A starts with an initial value, say "0." Node B starts with an initial value, say "1." The adversary makes Node A crash at the beginning of the execution, causing it to stop sending messages. Node B keeps running, but it doesn't know if Node A is alive or crashed. Node B cannot decide on a value because it doesn't have enough information to determine if it should choose "0" or "1" as the consensus value. This demonstrates that even with just one node failure, consensus cannot be guaranteed in an asynchronous network.

B. Restricted Crash-Fault Adversaries:

In the proof from the lecture, the FLP impossibility result is established for general asynchrony, which includes arbitrary unbounded message delays and the possibility of nodes behaving maliciously. When considering only crash-fault adversaries, the proof does not immediately apply because it relies on the possibility of nodes behaving in a more arbitrary and malicious manner, including not sending messages or sending contradictory messages.

C. To modify the proof to hold for crash-fault adversaries, you would need to simplify the adversarial behavior. Specifically, you can assume that adversaries only cause nodes to crash and don't engage in more complex, arbitrary actions. The modified proof would essentially follow the same logic as the original proof but with the added constraint that the adversarial actions are limited to crashes.The core idea remains the same: you'd show that in an asynchronous network with crash-fault adversaries, you cannot guarantee consensus, even with just one faulty node. Note that when a node can not send a message, talking about consensus seems trivial and pointless.

D. Introducing public key-private key pairs and assuming that all nodes' public keys are common knowledge does not change the FLP impossibility result. The FLP result is about the fundamental limitations of achieving consensus in asynchronous networks, regardless of the cryptographic tools or assumptions in use. The impossibility is related to the uncertainty caused by message delays and node failures, which cryptographic tools cannot fully address.

Therefore, the FLP impossibility result remains valid even when considering PKI or cryptographic mechanisms. It's a fundamental limitation of achieving consensus in asynchronous systems with crash or arbitrary faults.

Problem 3

A. To prove that the protocol is well-defined and that no node will receive at least f + 1 phase-r(b) messages for two different values v in any round r, we can use a proof by contradiction.

Assume the opposite, that there exists a round r and a node i that receives at least f + 1 phase-r(b) messages for two different values, let's say v1 and v2, where v1 v2.

Let's analyze the situations leading to this contradiction:

Case 1: At least f + 1 messages are honest: Since f + 1 messages agree on different values (v1 and v2), there must be at least f + 1 honest nodes that sent messages agreeing on different values. However, this contradicts the assumption that there are at most f Byzantine nodes. Thus, this case is not possible.

Case 2: At most f messages are honest: In this case, all messages sent by honest nodes are the same (either v1 or v2) because there are at most f Byzantine nodes and they can't prevent more than f honest nodes from sending the same value. However, this means that there can't be f + 1 phase-r(b) messages agreeing on different values, which contradicts our initial assumption.

Since both cases lead to contradictions, our initial assumption that there exists a round r and a node i that receives at least f + 1 phase-r(b) messages for two different values v1 and v2 must be false. Therefore, the protocol is well-defined and guarantees that no node will ever receive at least f + 1 phase-r(b) messages for two different values v.

B. To prove that the protocol satisfies validity whenever it terminates, we need to show that if the protocol terminates with an output value, that value must be the input of some honest node. Validity in the context of consensus protocols means that the decided value must be a value proposed by some honest node.

Let's analyze different scenarios that protocol possibly terminates:

Case 1: Termination due to more than (n + f)/2 phase-r(a) messages agreeing on a common value:

In this case, more than (n + f)/2 phase-r(a) messages agree on a common value v. This means that at least (n + f)/2 + 1 messages are from honest nodes (since $f < n/5$), and thus at least one of those messages is from an honest node proposing v. This guarantees validity. Note:

$$f < \frac{n}{5} \Rightarrow n > 5f \Rightarrow 4f < n + f \Rightarrow f < \frac{n+f}{4} \Rightarrow f < \frac{1}{2}(\frac{n+f}{2}) \ (Majority\ Vote)$$

Case 2: Termination due to more than (n + f)/2 phase-r(b) messages agreeing on a common value:

Similar to Case 1, this scenario guarantees validity because it means that at least (n + f)/2 + 1 messages are from honest nodes agreeing on a value v, which must have been proposed by an honest node.

Case 3: Termination due to at least f + 1 phase-r(b) messages agreeing on a common value:

In this case, we know that at least f + 1 messages are from honest nodes. The agreement on a common value v ensures that there's at least one honest node among these that proposed v, guaranteeing validity.

Case 4: Termination due to setting xi with 50/50 probability:

Even in this case, the protocol guarantees validity. This is because the probability that an honest node sets xi to a specific value (either 0 or 1) is 1/2, and since there are more honest nodes than Byzantine nodes, the collective effect will favor the value proposed by honest nodes.

In all termination scenarios, the protocol ensures that the decided value, if any, must have been proposed by some honest node. Therefore, the protocol satisfies validity whenever it terminates.

Problem 4

This proof can be done by numerous point of views. Some are explained below and others are left to the reader.

1st View.

A. The goal is to prove that if there is a Byzantine Agreement (BA) protocol that guarantees agreement, validity, and eventual termination in the "GST version" of the partially synchronous model (where Δ is known), then there is also such a protocol in the "unknown Δ version" (i.e., one that satisfies agreement, validity, and termination, no matter what Δ is).

Let's outline the proof for this statement:

Proof:

Assume that there exists a Byzantine Agreement protocol that works in the GST version of the partially synchronous model, where Δ (maximum message delay) is known. This protocol guarantees agreement, validity, and eventual termination post-GST.

Now, we want to create a protocol that works in the "unknown Δ version" of the partially synchronous model, where Δ is not known in advance. The key idea is to adaptively set an upper bound for Δ based on observed network behavior, ensuring that the protocol performs correctly for any Δ within that bound. Adaptive Δ Setting:

In the "unknown Δ version," the protocol description cannot depend on Δ. However, it can adaptively set an upper bound for Δ based on observed message delays. Initially, the protocol can set a conservative Δ bound (e.g., a high value) and monitor message delays post-GST. If the protocol observes that messages consistently arrive well within the initially set Δ bound, it can gradually decrease the Δ bound to optimize performance. This adaptive approach ensures that the protocol can operate effectively regardless of the actual Δ.

The protocol retains the same fundamental agreement, validity, and termination properties as in the GST version. It ensures that all honest nodes eventually agree on a common value, that the agreed-upon value is valid, and that the protocol eventually terminates.

This adaptive Δ setting approach allows the protocol to adapt to different Δ values without relying on knowing Δ in advance. As a result, the protocol works effectively in the "unknown Δ version" of the partially synchronous model, satisfying agreement, validity, and termination for any Δ. Thus, by adaptively setting an upper bound for Δ based on observed message delays, a BA protocol designed for the "GST version" of the partially synchronous model can also operate successfully in the "unknown Δ version," ensuring agreement, validity, and termination for any Δ value within the chosen bound. This demonstrates the adaptability and robustness of the protocol in varying network conditions.

B. To prove that if there is a Byzantine Agreement (BA) protocol that guarantees agreement, validity, and termination in the "unknown Δ version" of the partially synchronous model with f faulty nodes (regardless of Δ), then there is also such a protocol in the original "GST version" (i.e., one that satisfies agreement, validity, and eventual termination), we can use a simple argument.

Proof:

Assume there exists a Byzantine Agreement protocol that works in the "unknown Δ version" of the partially synchronous model, where Δ (maximum message delay) is not known in advance. This protocol guarantees agreement, validity, and termination for any Δ value.

We want to show that this protocol can also work in the original "GST version" of the partially synchronous model, where Δ is known. Since the protocol is designed to work for any Δ value, it can certainly work when Δ is known because it can adapt to a known Δ as one of the possible scenarios. This adaptation ensures that the protocol maintains its properties, including agreement, validity, and termination.

By the nature of the protocol's design for the "unknown Δ version," it is already capable of handling different Δ values, including the case where Δ is known (GST version). The protocol's fundamental properties, such as agreement, validity, and termination, are not dependent on whether Δ is known or unknown. Therefore, they are guaranteed in the GST version as well.

Thus, the protocol that guarantees agreement, validity, and termination in the "unknown Δ version" can also operate effectively in the original "GST version" with known Δ. The protocol's properties

remain intact, and it continues to satisfy agreement, validity, and eventual termination. In conclusion, if there is a Byzantine Agreement protocol that ensures agreement, validity, and termination in the "unknown Δ version" of the partially synchronous model for any Δ value, then the same protocol can be applied to the original "GST version" with known Δ, guaranteeing the same properties. This demonstrates the protocol's versatility and applicability in different scenarios, whether Δ is known or unknown.

2nd View. Let's discuss this in another literature. The partial synchrony model comes in two flavours: GST and Unknown Latency.

The GST flavour of Partial Synchrony

The Global Stabilization Time (GST) model for Partial Synchrony assumes that:

1) There is an event called GST that occurs after some finite time.

2) There is no bound on message delays before GST, but after GST all message must arrive within some known bound Δ.

The Unknown Latency flavour of Partial Synchrony

In the Unknown Latency(UL) flavour, the system is always synchronous, but the bound of the maximum delay is unknown to the protocol designer. The way the Unknown Latency models the real world is somewhat problematic, it essentially strives to set the latency of the protocol to be as large as the worst case latency that will ever occur.

Unlike the GST flavour where Δ can be set conservatively, in the UL flavour the estimation of latency needs to grow based on the worst case behavior of the system. For example, many early academic BFT systems had mechanisms that double the protocol's estimation of Δ each time there was a timeout.

From Unknown Latency to Global Stabilization Time

Assume a protocol Q that obtains safety and liveness in the Unknown Latency flavour. Does Q also obtain safety and liveness in the GST flavour?

Yes! consider an execution in the GST flavour and let Γ be the maximum of Δ and the time until GST starts. Observe that by definition, Q has safety and liveness assuming that the unknown latency is Γ , because its true for any finite latency.

Note that this reduction is extremely wasteful: the value of Γ may be huge. Many systems may strive to adjust their estimate of Δ both up and down.

From Global Stabilization Time to Unknown Latency

For completeness, here is the failed proof approach: The idea was to start with some estimation Λ of the actual UL parameter Δ . Each time a protocol timeout expires, increment Λ .

For safety, we need to argue that incrementing Λ does not break safety, one trivial way to do that is assume this is a property of P . It is not clear that a generic protocol has this property. For example, as currently stated, P may do very different things for different values of Λ .

For liveness, the idea was to say that at some point Λ will grow to be large enough. At that point and onwards, we need to argue that the protocol obtains liveness. Again one trivial way to do that is assume this is a property of P . It is not clear that a generic protocol has this property. For example, perhaps liveness of P is lost when the timeouts are too large?

Nevertheless, this idea of dynamically incrementing the estimation of the maximum latency is a well used technique:

In terms of efficiency:

1) incrementing Λ too slowly means that it may take a lot of time to reach consensus. This is a type of a denial-of-service that slows down the system.

2) incrementing Λ too aggressively (say by doubling) means that while we may reach $\Lambda > \Delta$ quickly, it may still happen that there are at least f more timeouts due to malicious primaries. So Λ may grow exponentially. This again may cause a denial-of-service that slows down the system.

In practice systems often adjust their estimate of Δ both up and down (sometimes using an explore-exploit type online learning algorithm).

Problem 5

A. We first fix n = 3 and f = 1.

To prove impossibility result, we always have to consider the worst case ever. Since the protocol is run under partial synchronous assumptions, We assume that the byzantine node doesn't send message, so we can only wait to hear from 2 other nodes before taking any action.

Thus, to avoid getting tricked, more than 50% of the 2 nodes need to be honest which is simply not possible:

Taking majority vote: $\frac{1}{2}(n-f) \not> f \longrightarrow \frac{1}{2}(3-1) \not> 1$ Thus it is not possible!

B. The intuition in (a) can be easily modified to be applied to all $f \geq \frac{n}{3}$ and $n \geq 3$:

To prove impossibility result, we always have to consider the worst case ever. Since the protocol is run under partial synchronous assumptions, We assume that the f byzantine nodes don't send message, so we can only wait to hear from $n-f$ other nodes before taking any action.

Thus, to avoid getting tricked, more than 50% of the $n-f$ nodes need to be honest which is simply not possible:

Taking majority vote: $f < \frac{1}{2}(n-f)$ Thus $f < \frac{n}{3}$.

C. Main reasons:

(a) It holds whether Public Key Infrastructure(PKI) assumption exists or not.

(b) It holds in the partially synchronous model while the impossibility result from Lecture 3 holds only for synchronous model.

D. The problem statement is related to the impossibility of achieving consensus with deterministic protocols in the partially synchronous model when the number of Byzantine nodes exceeds one-third of the total nodes (f $\geq \frac{n}{3}$). This result extends to the State Machine Replication (SMR) problem, which is a crucial component of many blockchain systems.

To prove this impossibility result, we can use a proof by contradiction. Let's assume that there exists a deterministic protocol for the SMR problem that satisfies consistency (at all times) and eventual liveness (required only after GST, which stands for "Global Stabilization Time").

Now, we will show that this assumption leads to a contradiction, which means that such a protocol cannot exist when f $\geq \frac{n}{3}$.

Consistency Guarantee: The protocol guarantees consistency at all times. This means that all honest nodes in the network will agree on the same sequence of commands, even in the presence of Byzantine nodes.

Eventual Liveness: The protocol guarantees eventual liveness. This means that the system will make progress and eventually terminate with a decision.

Now, consider the following scenario: Assume there are n nodes in the network, out of which f nodes are Byzantine, and f $\geq \frac{n}{3}$.

During GST, the network reaches a point where f Byzantine nodes decide to halt the progress maliciously. They refuse to move forward, causing the protocol to stall.

The remaining honest nodes, which are less than ($\frac{2n}{3}$), cannot achieve consensus because they don't have enough votes to overcome the Byzantine nodes' influence (f $\geq \frac{n}{3}$).

Since the protocol guarantees consistency, it cannot proceed without having all honest nodes agree on the next state transition.

However, due to the Byzantine nodes' refusal to make progress, the protocol remains stuck, violating the guarantee of eventual liveness.

This contradiction arises from the assumption that a deterministic protocol can achieve both consistency and eventual liveness when f $\geq \frac{n}{3}$. Therefore, the original assumption must be false, and no deterministic protocol for the SMR problem can satisfy both consistency and eventual liveness when f $\geq \frac{n}{3}$.

Problem 6

***Note: We denote the byzantine fault by "f" and crash-fault adversary by "k".**

A. We first fix n = 2 and k = 1. To prove impossibility result, we always have to consider the worst case ever. Since the protocol is run under partial synchronous assumptions, We assume that the faulty node are actually a crash-fault adversary, so we can only wait to hear from 1 other nodes before taking any action.

Thus, to avoid getting tricked, more than 50% nodes need to be honest which is simply not possible: Taking majority vote: $\frac{1}{2}(n) \not> k \longrightarrow \frac{1}{2}(2) \not> 1$ Thus it is not possible!

B. Here we do a stronger proof by taking byzantine nodes into consideration. Like what we saw in *Problem 6*: Taking majority vote: $f + k < \frac{1}{2}(n - f)$ Thus $n > 3f + 2k$. Study ***this paper*** for more.

Consequently if f = 0: $n > 2k$

Thus, the impossibility result holds when: $k \geq \frac{n}{2}$ while $(n \geq 2)$