State Machine Replication(SMR) in synchronous Network setting without PKI

Instructor: Nitin Awathare

September 16, 2024

While theory is excellent for proving possibilities, as shown with the Dolev-Strong protocol, it's arguably even better suited for impossibility results, defining the limits of what computers, algorithms, and protocols can achieve. In this lecture we are gonna discuss a impossibility result, called as FLM Impossibility. In some literature you will see Hexagon proof referring to the same.

To keep this lecture concise, let's make two simplifying assumptions: 1) We'll focus on deterministic protocols (no randomization), which doesn't change the impossibility result significantly, 2) We'll consider the simplest case of $n = 3$ and $f = 1$ (three nodes, one potentially Byzantine). While this might seem like it trivializes the result, it actually captures the full complexity of the general impossibility.

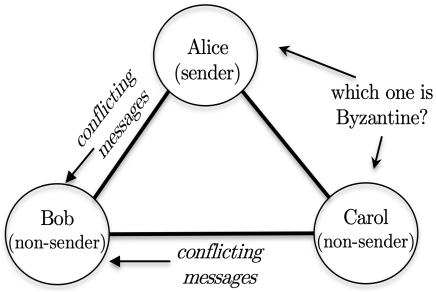

Figure 1: Scenario where Bob can't tell if Alice is Byzantine and Carol is honest, or if Carol is Byzantine and framing Alice.

Before the formal proof, let's build some intuition on why the result might hold and why the critical threshold is $n/3$. Consider three nodes: Alice, Bob, and Carol, each communicating with the others (see Figure 1). Suppose Alice is the sender. The key issue is that one node can identify conflicting information from the others but can't determine which is Byzantine. For example, Bob might receive conflicting messages from Alice and Carol. One scenario is Carol honestly echoing Alice's messages while Alice, being Byzantine, sends conflicting info to Bob. Alternatively, Alice might be honest and consistent, but Carol could be Byzantine, trying to frame Alice. Bob can't determine who's at fault and therefore can't decide on an output. If $n = 4$ and $f = 1$, Bob could use a majority vote among the other nodes to find the correct output.

If you're sharp, you might question this informal argument, especially Carol's ability to frame Alice, and suspect why such reasoning doesn't apply to the Dolev-Strong protocol. Try to identify the key step(s) in the next section's proof that don't apply to Dolev-Strong.

Impossibility results in computer science are challenging due to the vast design space—you must rule out every possible solution, no matter how creative. This is what made Turing's halting problem proof groundbreaking, and why the $P \neq NP$ conjecture remains unproven. The difficulty applies similarly to distributed protocols. For example, in trying to prove FLM impossibility by contradiction, we assume a protocol π exists for Byzantine broadcast with $n = 3$ and $f = 1$, but we must account for all possible forms π might take, no matter how complex.

In the proof, we can generically use the assumed protocol π by running it, simulating honest nodes through a Byzantine node's perspective. This "simulation" idea, common in theoretical proofs, is one key theme in the FLM proof of Theorem 1. The second theme is "indistinguishability," where an honest node can't distinguish between two plausible scenarios, causing it to be stuck since validity or agreement may require different outputs for each. The challenge is to create such a situation, applicable regardless of what π looks like.

The idea of FLM proof is to create two copies of Alice, Bob, and Carol, forcing them to participate in multiple conflicting scenarios. Thus, we deploy an allegedly correct Byzantine broadcast protocol π on six machines. To clarify, we need to specify the inputs the protocol π expects when it first runs on a node i. Inputs are usually resides on the locally stored initialization file. Following are the inputs that protocol π would get.

(in a locally stored initialization file, if you like)

- the names and IP addresses of two other nodes (recall the protocol π is designed for the case of n = 3);
- among node i and the two other nodes, which one is the sender (there should be exactly one sender);
- if node i is the sender, its private input.

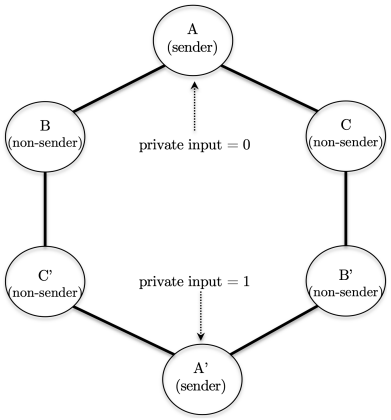

Figure 2: The hexagon thought experiment setup, where each machine runs protocol π with its two designated neighbors and, if a sender, with its specified private input.

The protocol π was designed assuming that node i's neighbors, j and k, list each other properly in their initialization files. For example, the initialization files at i's two neighbors, say j and k, list the nodes i,k and i,j, respectively. However, we could still run π at node i even if j's and k's files are inconsistent, though this might lead to unpredictable behavior or an infinite run. The hexagon experiment leverages this, running six copies of π on machines with inconsistent initialization files, as shown in Figure 2. The details of the thought experiment setup is as follows:

- Set up node A and initialize as: (i) neighbors B and C. This can be achieved with either names and IP address. (ii) A is the sender, B and C are non-senders. (iii) A's private input is 0.
- Set up node B with: (i) neighbors A and C'; (ii) A as the sender, B and C' as non-senders. (C' shares the same name as C but has a different IP address.)
- Set up node C with: (i) neighbors A and B'; (ii) A as the sender, B' and C as non-senders.
- Set up node C' with: (i) neighbors A' and B; (ii) A' as the sender, B and C' as non-senders.
- Set up node B' with: (i) neighbors A' and C; (ii) A' as the sender, B' and C as non-senders.
- Set up node A' with: (i) neighbors B' and C'; (ii) A' as the sender, B' and C' as non-senders; (iii) A''s private input is 1.

With the protocol π, nothing prevents you from buying six machines, installing π on each, and setting up their initialization files as described. Each file contains the exact info, i.e., neighbours, sender, and, sender's private input, that π expects. You can press play on all six machines simultaneously. While π wasn't designed for this setup, you're free to run the experiment and observe the nodes' outputs.

Since each node's initialization file in the hexagon experiment aligns with the protocol π's expectations, you can run π on all six nodes. However, the experiment setup ($n = 6$, not all nodes aware of others) doesn't match the intended setup for π ($n = 3$, all nodes with common knowledge). Thus, properties like validity from the original scenario may not hold in this new setup (e.g., with two senders and different private inputs, the concept of validity becomes unclear).

For our proof strategy, this distinction is crucial. We'll need to leverage the assumptions that π satisfies termination, agreement, and validity. Since achieving only two of the three is trivial, an impossibility proof must involve all three. But since we can't directly apply these properties in the hexagon experiment, we'll bridge the gap by showing that the hexagon encodes three distinct three-node Byzantine broadcast instances. In each instance, agreement or validity imposes constraints on pairs of nodes, which will extend to the hexagon. These constraints will conflict, leading to a contradiction, proving that the hexagon setup cannot have a consistent output. We will divide the proof strategy into 3 different scenarios, leading to 3 different worlds.

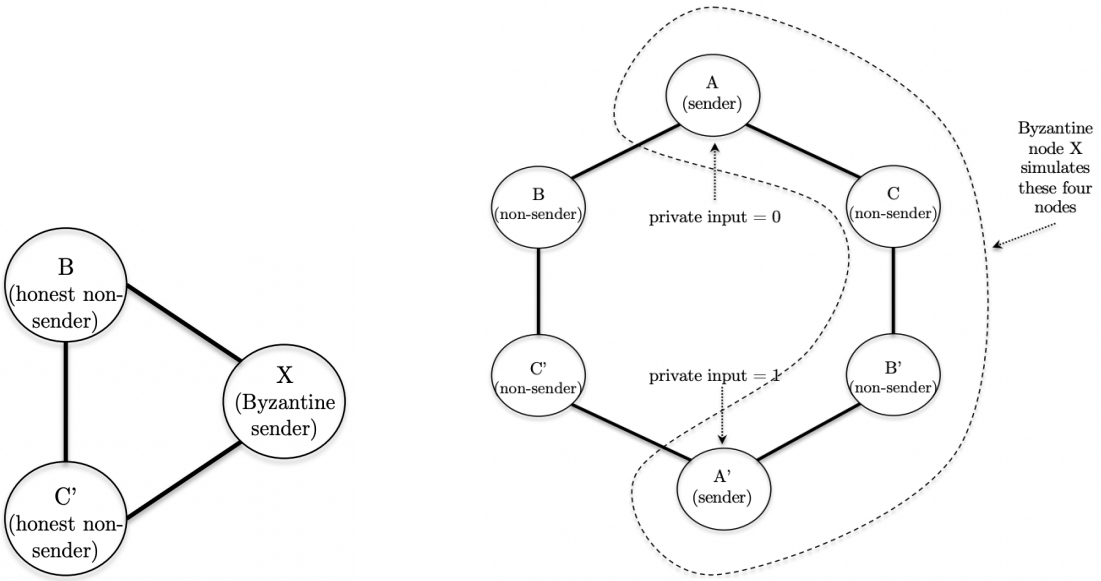

Figure 3: (a)Byzantine broadcast instance No. 1, (b)By simulating four nodes in the hexagon experiment, the Byzantine node (sender) X forces B and C' to behave identically in both the triangle (in Figure 3a) and the hexagon. The agreement property of π ensures that B and C' output the same value.

First World Instance : The first instance of Byzantine broadcast "embedded" in the hexagon experiment involves a Byzantine sender (node X) and two honest non-senders (B and C'). Since this is a valid Byzantine broadcast instance with $n = 3$, we can apply π's guarantees of termination and agreement, ensuring that the honest nodes B and C' eventually terminate and output the same value, regardless of X's strategy.

One strategy for X is to simulate the behavior of four nodes in the hexagon thought experiment, making B and C' believe they are in the hexagon. X runs four virtual copies (A, C, B', and A') of π, forwarding messages between its neighbors and virtual machines as if they were in the hexagon setup, as depicted in Figure 3. This keeps B and C' indistinguishable from the hexagon experiment and forces them to behave identically.

Since B and C' cannot tell the difference between the hexagon and the actual Byzantine broadcast instance, their outputs must be the same in both cases, by the property of agreement. Thus, in the hexagon experiment, B and C' will output the **same** value.

This establishes the first key part of the proof, and the remaining scenarios (world) follow the same pattern. In this scenario (worlds), we focused on the agreement property that the protocol π is assumed to follow. In the next two scenarios, we will examine validity, where one of the non-senders is Byzantine and the sender is honest.

Second World Instance : The honest nodes A and B in the triangle mirror their counterparts in the hexagon, with the honest sender A having a private input of 0. To keep A and B unaware of whether they are in the triangle or hexagon, the Byzantine non-sender X simulates the remaining four nodes of the hexagon by running four copies of π (on separate virtual machines) to mimic nodes C, B', A', and C', as depicted in Figure 4. X interacts with A as if it were C in the hexagon and with B as if it were C'.

Since π satisfies termination and validity in every bona fide three-node Byzantine broadcast instance (like the triangle), nodes A and B must eventually output 0 (A's input). Because A and B operate identically in the hexagon (due to X's simulation), they must also output 0 in the hexagon.

Thus, in the hexagon thought experiment, nodes A and B output 0.

Third World Instance : The third scenario mirrors the second, with an honest sender A' (private input 1), an honest non-sender C', and a Byzantine non-sender X, as depicted in Figure 5a. To obscure the context, Byzantine node X simulates the remaining hexagon nodes (Figure 5b) by running four copies of π, interacting with A' as B' and C' as B. Since π guarantees termination and validity, A' and C' must output 1 in the triangle, and by equivalence in the hexagon experiment, they also output 1 there.

Given these three scenarios (worlds), let's now turn to the formal statement of the FLM impossibility theorem.

Theorem 1. In the synchronous model with $f \geq n/3$, there is no Byzantine broad-cast protocol that satisfies termination, agreement, and validity.

Proof. We analyzed three different three-node Byzantine broadcast instances above, using π's guarantees to deduce constraints on the honest nodes' outputs. By simulating the hexagon thought experiment with a Byzantine node strategy, we transferred these constraints to the hexagon. The constraints are: (1) nodes B and C' must output the same value; (2) B must output 0; (3) C' must output 1. Since these constraints are mutually

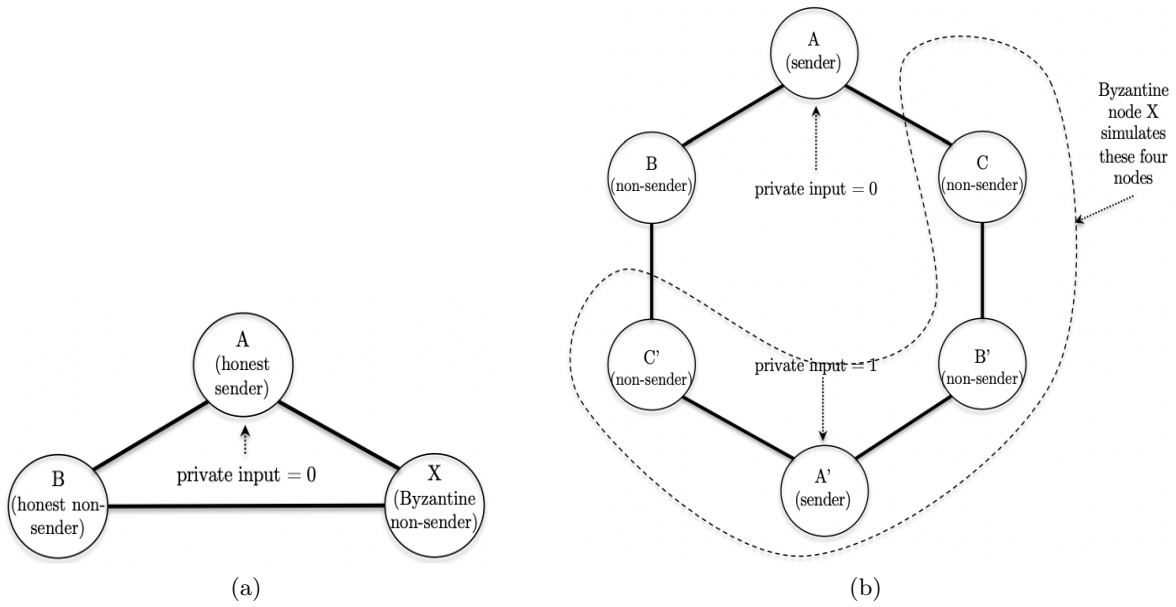

Figure 4: (a)Byzantine broadcast instance No. 2, (b)By simulating four nodes in the hexagon experiment, the Byzantine node X forces A and B to act identically in both the triangle and hexagon. The validity property of π ensures that A and B both output 0.

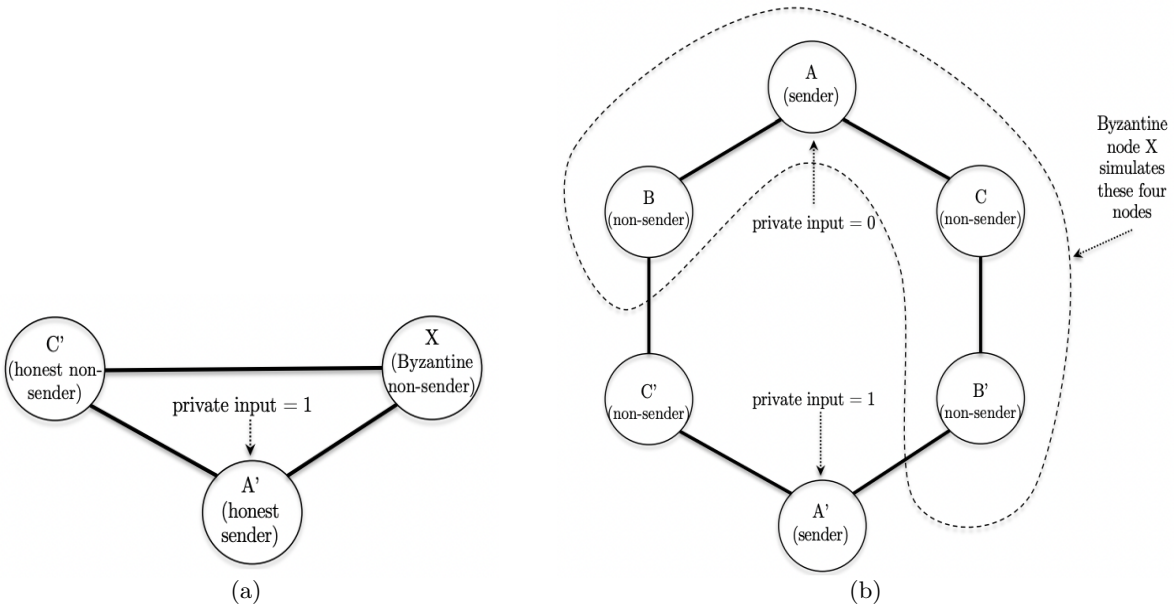

Figure 5: (a)Byzantine broadcast instance No. 3, (b)By simulating four nodes in the hexagon experiment, the Byzantine node X forces A' and C' to operate identically in both the triangle and hexagon. The validity property of π dictates that A' and C' both output 1.

inconsistent, the hexagon experiment cannot satisfy them, proving that no protocol π exists with $n = 3$ and $f = 1$ that guarantees termination, agreement, and validity.

□

The main result of previous Lecture is that, in the synchronous model, Byzantine broadcast can be solved with any number of Byzantine nodes. However, the main result of this lecture is that you can't solve Byzantine broadcast if at least one-third of the nodes are Byzantine. What explains this discrepancy?

The answer to this question is revealed in the lecture title. In the previous Lecture, we extended the usual permission assumption to include public key infrastructure (PKI), where nodes know each other's public keys in addition to their names and IP addresses. This is a trusted setup assumption that asserts correct public key distribution before the protocol starts, without specifying the distribution method.

Our analysis of the Dolev-Strong protocol in the previous Lecture relied on three assumptions: permissioned setting (all node names known initially), synchronous setting (reliable message delivery), and PKI assumption (nodes can verify each other's signatures). The proof discussed in this lecture does not violate the permissioned or synchronous setting assumptions. Therefore, Theorem 1 cannot hold under the PKI assumption, as it would

contradict our findings about the Dolev-Strong protocol. The proof discussed in this lecture must thus violate the PKI assumption. The question then is how common knowledge of public keys impacts the proof.

To make the hexagon thought experiment, discssued above, compatible with the PKI assumption, we need to include cryptographic keys in the nodes' initialization files. The updated file format is:

(I1) Names, IP addresses, and public keys of two other nodes. (I2) Identification of the sender among the node and the two others. (I3) If the node is the sender, its private input. (I4) A distinct public-private key pair for the node. The thought experiment should be modified accordingly as follows and the the same is detailed in Figure 6

- Node A:

  Neighbors: B and C (name, IP, public key). A is a sender; B and C are non-senders. Private input: 0. Key pair: (pk_A, sk_A).

- Node B:

  Neighbors: A and C' (name, IP, public key). A is a sender; B and C' are non-senders. Key pair: (pk_B, sk_B). Note: C' shares the name and key pair with C but has a different IP.

- Node C:

  Neighbors: A and B' (name, IP, public key). A is a sender; B' and C are non-senders. Key pair: (pk_C, sk_C).

- Node C':

  Neighbors: A' and B (name, IP, public key). A' is a sender; B and C' are non-senders. Key pair: ($pk_{C'}$, $sk_{C'}$).

- Node B':

  Neighbors: A' and C (name, IP, public key). A' is a sender; B' and C are non-senders. Key pair: ($pk_{B'}$, $sk_{B'}$).

- Node A':

  Neighbors: B' and C' (name, IP, public key). A' is a sender; B' and C' are non-senders. Private input: 1. Key pair: ($pk_{A'}$, $sk_{A'}$).

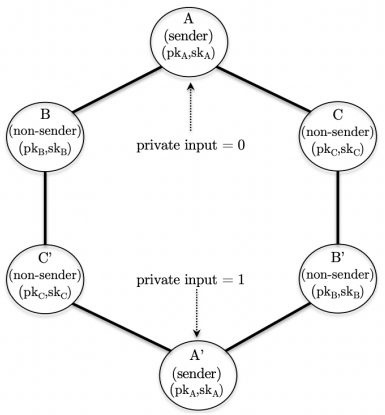

Figure 6: Extension of the hexagon thought experiment to incorporate the PKI assumption.

With the PKI assumption in place, we attempt to replicate the argument from the First world instance of the Hexagon proof discussed above. The proof, which hinges on simulating a hexagon, faces a challenge. Specifically, consider a Byzantine broadcast instance where node X simulates four vertices from the revised hexagon experiment. The goal is to keep honest nodes unaware of their actual environment, applying constraints from the triangle to the hexagon.

However, X faces a limitation: it can only simulate nodes A and A', as it lacks the private keys for nodes B' and C. The Byzantine Broadcast protocol π might require signing every message, but without the private keys for B' and C, X cannot fully simulate them. Thus, the proof's reliance on such simulations fails under the PKI assumption, as cryptographic constraints prevent X from performing the required simulations.

The contrast between what can be achieved with the PKI assumption versus without it is striking. While cryptographic assumptions like secure digital signatures and public key distribution fundamentally impact the

design and feasibility of consensus protocols, they are irrelevant for algorithmic problems like minimum spanning trees or NP-hard problems like the traveling salesman problem.

Impossibility results serve to highlight when models are too demanding or assumptions too limited. This lecture illustrates that to achieve a robust consensus protocol with many faulty nodes, a secure digital signature scheme and public key distribution are essential. The upcoming lectures will explore the need for assumptions about the reliability of communication networks (FLP impossibility), contrasting with our current focus on cryptographic assumptions.

With this we have concluded our discussion on FLM impossibility and/or hexagon proof.

References:

1. Foundations of Blockchains `https://timroughgarden.github.io/fob21/`.
2. Blockchain gets better: moving beyond Bitcoin
`https://www.comp.nus.edu.sg/features/2018-blockchain-gets-better/`

Disclaimer :