Byzantine Generals’ Problem:

Decrypting Leslie Lamports’ Solution

This blog explains Leslie Lamport’s solution to the Byzantine Generals’ problem that uses unsigned messages, providing a step-by-step walkthrough with visual examples.

Motivation

Distributed systems are everywhere in our modern digital infrastructure—from cloud computing to blockchain networks. However, a fundamental challenge in these systems is achieving consensus when some components might be faulty or malicious.

The Byzantine Generals’ Problem, introduced by Leslie Lamport, Robert Shostak, and Marshall Pease in 1982, elegantly captures this challenge through a military metaphor. Understanding this problem and its solution is crucial for anyone working with distributed systems, blockchain technology, or fault-tolerant computing.

This blog focuses specifically on the solution using unsigned messages—the most challenging scenario where messages carry no authentication. We’ll examine a non-trivial instance with the most adversarial case: where the commander is a traitor and strategically sends conflicting commands to create maximum confusion.

The unsigned message solution is particularly fascinating because it operates under the loosest guarantees. Unlike solutions that rely on cryptographic signatures or other forms of authentication, this approach must achieve consensus through pure logic and majority reasoning, making it both intellectually stimulating and practically relevant.

Introduction

Imagine a scenario where several Byzantine generals have surrounded an enemy city. They must decide collectively whether to attack or retreat. To succeed, all loyal generals must agree on the same plan of action—either all attack or all retreat. A split decision would be disastrous.

The generals can only communicate through messengers, and some generals might be traitors (including potentially the commanding general) who seek to prevent the loyal generals from reaching agreement. These traitors can:

- Send different messages to different generals
- Collude with other traitors
- Lie about messages they claim to have received

The challenge is to design an algorithm that ensures all loyal generals reach the same decision despite these traitors. This is the essence of the Byzantine Generals' Problem.

The solution we'll explore doesn't rely on message signatures or authentication. Instead, it uses a recursive message-passing approach called Oral Message (OM) algorithm that can withstand up to t traitors in a system with at least $3t + 1$ generals.

Terminology

Before diving into the solution, let's establish some terminology:

- **Generals**: The participants in the system, denoted as G_i where i is the general's identifier.
- **Commander**: The general who initiates the decision process (usually G_0).
- **Lieutenants**: All other generals who must follow the commander's decision.
- **Loyal ($\mathcal{L}$)**: Generals who follow the algorithm correctly.
- **Traitors ($\mathcal{T}$)**: Generals who may behave arbitrarily and maliciously.
- **Commands**: The possible decisions, typically Attack (**A**) or Retreat (**R**).
- **OM(m)**: The Oral Message algorithm with recursion depth m.

The Byzantine Generals' Problem has two key requirements:

1. **Agreement**: All loyal generals must agree on the same decision. On receiving equal number of attack and retreat messages the loyal generals default to predetermined decision, *Retreat*

2. **Validity**: If the commander is loyal, all loyal generals must follow the commander's order.

Lamport proved that with oral (unsigned) messages, we need at least $3t + 1$ generals to tolerate t traitors, and the algorithm requires $t+1$ rounds of message exchanges.

The Algorithm

The Oral Message algorithm (OM) works recursively:

OM(0): The commander sends a value to all lieutenants, and each lieutenant uses the value received.

OM(m), for $m > 0$:

1. The commander sends a value to each lieutenant.

2. Each lieutenant acts as a commander in the OM($m-1$) algorithm to share the value they received with all other lieutenants.

3. Each lieutenant determines the majority value reported for each general and uses that as their final decision.

The algorithm guarantees correct operation if $n \geq 3t+1$ where n is the total number of generals and t is the maximum number of traitors.

The recursive nature of this algorithm creates a tree of message exchanges that grows exponentially with m. For a system with n generals, the message complexity is $O(n^{m+1})$, which makes it practical only for small values of t and thus m.

The Problem

Let's examine a specific instance of the Byzantine Generals' Problem:

- **Total generals**: $N = 7$

- **Traitors**: $t = 2$

- **Loyal generals**: $N - t = 5$

In our scenario:

- Commander G_0 is a traitor
- Lieutenant G_1 is also a traitor
- Lieutenants G_2 through G_6 are loyal

This is a particularly challenging case because the commander, who initiates the algorithm, is malicious. The commander's strategy is to create maximum confusion by sending conflicting commands:

- Commander G_0 sends Attack (A) to generals G_2, G_4, and G_6
- Commander G_0 sends Retreat (R) to generals G_1, G_3, and G_5

Additionally, the second traitor G_1 will also attempt to confuse the loyal generals by inconsistently relaying messages during the second round.

With $t = 2$ traitors, we need at least $3t + 1 = 7$ generals, which is exactly what we have. According to the theory, OM(2) should be sufficient to achieve consensus among loyal generals. Let's see how this works in practice.

The Algorithm in Play

Round 1

In the first round, the commander G_0 (who is a traitor) sends commands to all lieutenants. Since the commander is a traitor, they can send different commands to different lieutenants. Figure 1 illustrates what happens in Round 1:

As shown in the diagram, the traitor commander G_0 sends:

- Retreat (R) to generals G_1 (traitor), G_3, and G_5
- Attack (A) to generals G_2, G_4, and G_6

At this point, each lieutenant only knows what the commander told them directly. The table in Figure 1 represents this initial knowledge state, with each lieutenant recording their received command on the diagonal.

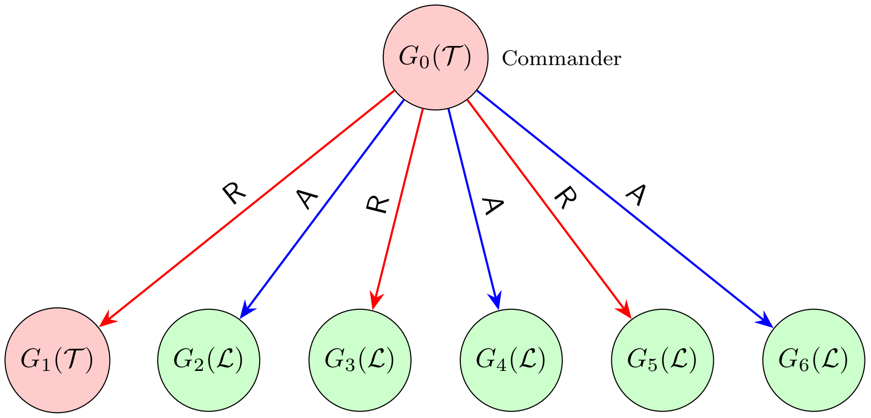

Figure 1: Round 1: Traitor Commander G_0 sends inconsistent messages to different lieutenants.

$G_1(\mathcal{T})$	$G_2(\mathcal{L})$	$G_3(\mathcal{L})$	$G_4(\mathcal{L})$	$G_5(\mathcal{L})$	$G_6(\mathcal{L})$
R					
	A				
		R			
			A		
				R	
					A

The grid above shows the initial values v_i that each lieutenant G_i (represented by row and column headers) receives from Commander G_0 and stores. This value is placed on the diagonal cell (G_i, G_i). Off-diagonal cells are empty, to be filled by messages exchanged between lieutenants in later algorithm phases (e.g., OM(m-1) rounds).

Round 2

In the second round, each lieutenant acts as the commander for the round and tells all other lieutenants what command they received from the original commander in Round 1.

Figure 2 illustrates a scenario where a traitor lieutenant sends conflicting messages during Round 2:

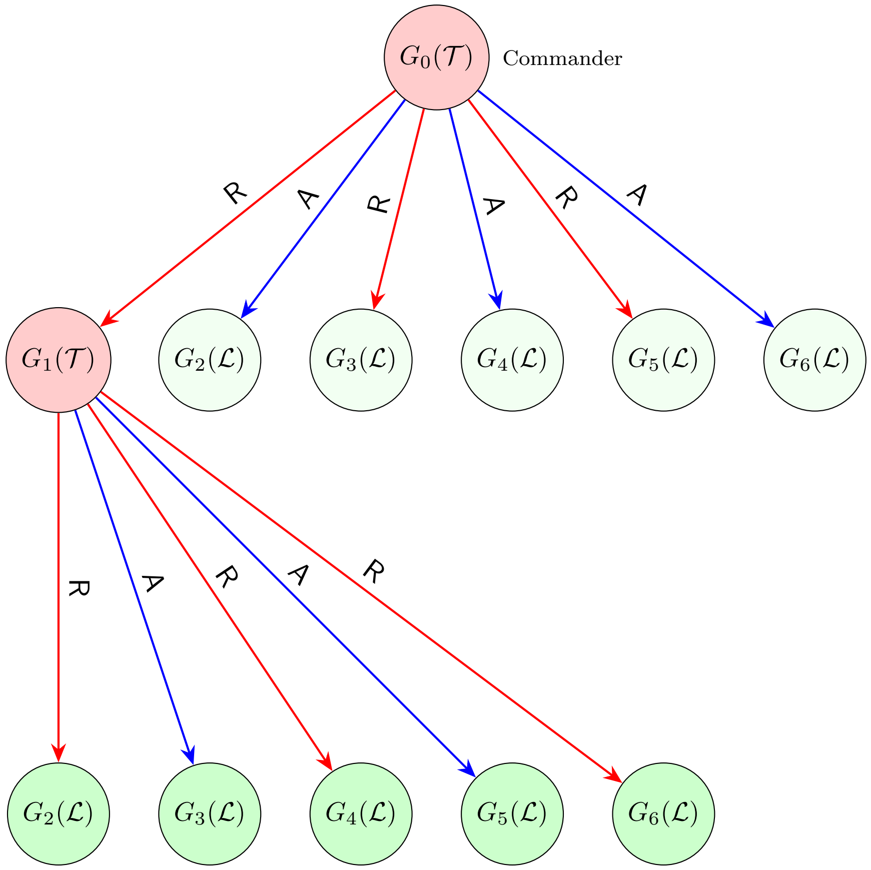

Figure 2: Round 2: Lieutenants exchange information about what they received from the commander. Traitor lieutenant G_1 deliberately sends conflicting messages to different receivers.

Receiver

Sender	G_1	G_2	G_3	G_4	G_5	G_6
G_1						
G_2		**R**	R	R	R	R
G_3		A	**A**	A	A	A
G_4		R	R	**R**	R	R
G_5		A	A	A	**A**	A
G_6		R	R	R	R	**R**
Majority		**R**	**R**	**R**	**R**	**R**

Legend: Rows (left) are senders, columns (top) are receivers. Each cell shows the message the sender tells the receiver that it received from G_1. Yellow diagonal:

sender's own message from G_1. Grey cells: G_1 is the acting commander and does not participate in consensus. Blue = Attack (A), Red = Retreat (R).

$G_1(\mathcal{T})$	$G_2(\mathcal{L})$	$G_3(\mathcal{L})$	$G_4(\mathcal{L})$	$G_5(\mathcal{L})$	$G_6(\mathcal{L})$
R	**R**	**R**	**R**	**R**	**R**
	A				
		R			
			A		
				R	
					A

After Round 2, each loyal lieutenant can determine the consensus by taking a majority vote of all reported values. Despite the conflicting information from the traitors, all loyal lieutenants reach the same conclusion: Retreat (R).

The majority from round $O(m-1)$ is then used as the truth in $O(m)$.

In a different scenario, we can observe how a loyal lieutenant shares consistent information during Round 2:

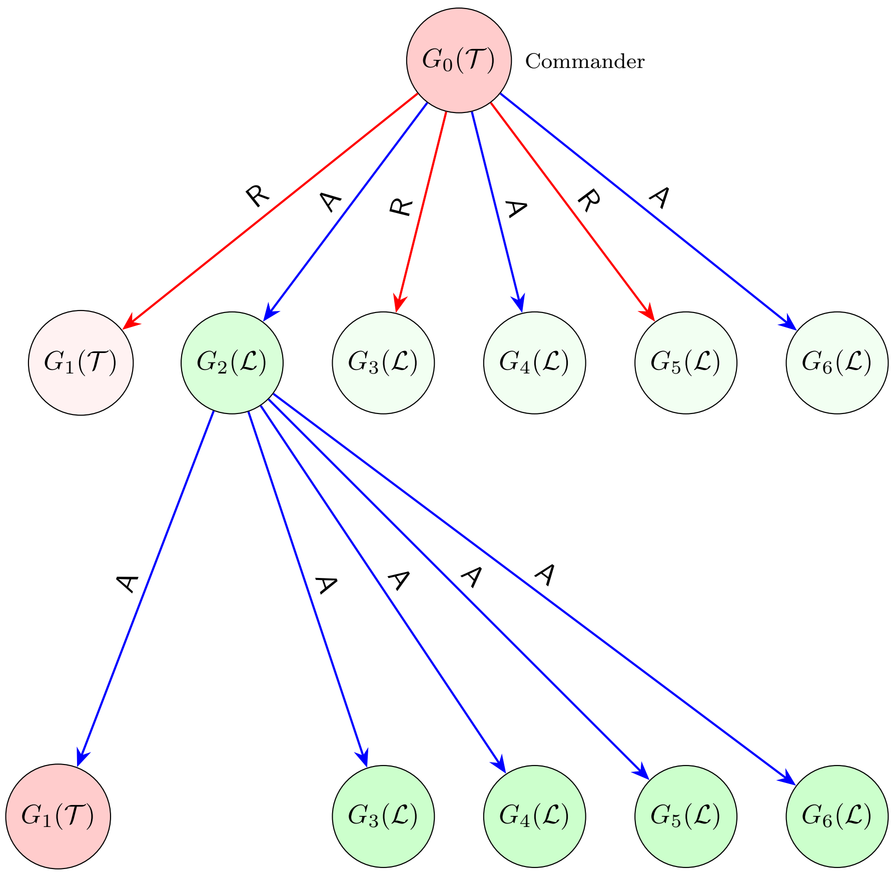

Figure 3: Round 2: Lieutenants exchange information about what they received from the commander. Loyal lieutenant G_2 faithfully relays the original message to all other lieutenants.

		Receiver					
		G_1	G_2	G_3	G_4	G_5	G_6
Sender	G_1	**A**		R	R	R	R
	G_2						
	G_3	A		**A**	A	A	A
	G_4	A		**A**	A	A	A
	G_5	A		A	A	**A**	A
	G_6	A		A	A	**A**	A
	Majority	**A**		**A**	**A**	**A**	**A**

Legend: Rows (left) are senders, columns (top) are receivers. Each cell shows the message the sender tells the receiver that it received from the commander. Yellow diagonal: sender's own message from the commander. Grey cells: G_2 is the acting commander and does not participate in consensus. Blue = Attack (A), Red = Retreat (R).

$G_1(\mathcal{T})$	$G_2(\mathcal{L})$	$G_3(\mathcal{L})$	$G_4(\mathcal{L})$	$G_5(\mathcal{L})$	$G_6(\mathcal{L})$
R	R	R	R	R	R
A	A	A	A	A	A
		R			
			A		
				R	
					A

After Round 2, each loyal lieutenant can determine the consensus by taking a majority vote of all reported values. Despite the conflicting information from the traitors, all loyal lieutenants reach the same conclusion: Retreat (R).

After all lieutenants have completed their communications in Round 2, the final information state of each general is represented in the following table:

$G_1(\mathcal{T})$	$G_2(\mathcal{L})$	$G_3(\mathcal{L})$	$G_4(\mathcal{L})$	$G_5(\mathcal{L})$	$G_6(\mathcal{L})$
R	R	R	R	R	R
A	A	A	A	A	A
R	R	R	R	R	R
A	A	A	A	A	A
R	R	R	R	R	R
A	A	A	A	A	A
R	R	R	R	R	R

After all lieutenants have exchanged their Round 2 messages, each loyal lieutenant can determine the consensus by taking a majority vote of all reported values. Despite the conflicting information sent by the traitor lieutenant G_1, all loyal lieutenants reach the same conclusion: Retreat (R), demonstrating the algorithm's ability to achieve consensus even with a traitor commander.

To summarize, The traitor lieutenant G_1 (who received "Retreat" from G_0) continues the deception by:

- Telling some lieutenants that it received "Attack" from the commander
- Telling other lieutenants that it received "Retreat" from the commander

Meanwhile, all loyal lieutenants honestly relay the commands they received. After this round, each lieutenant has:

- Their original command from the commander (from Round 1)
- Reports from all other lieutenants about what they claim to have received

The final table shows what each lieutenant knows after all Round 2 messages have been exchanged. The cell values show what message was conveyed.

Now, each loyal lieutenant can determine what command to follow by taking a majority vote of all reported values. The consensus among all loyal lieutenants is to "Retreat" (R).

This demonstrates the power of the OM(2) algorithm: despite having a traitor commander and a traitor lieutenant, all loyal lieutenants reach the same conclusion. The Byzantine Generals' Problem is solved for this instance!

A Second Example: Loyal Commander with Collaborating Traitors

Let's examine another crucial scenario where the commander is loyal but traitor lieutenants collaborate to disrupt consensus. This scenario demonstrates the algorithm's effectiveness at satisfying the validity condition when the commander is loyal.

- **Total generals**: $N = 7$
- **Traitors**: $t = 2$
- **Loyal generals**: $N - t = 5$

In this scenario:

- Commander G_0 is loyal
- Lieutenants G_1 and G_4 are traitors
- Lieutenants G_2, G_3, G_5, and G_6 are loyal

We'll implement the OM(2) algorithm for this scenario and observe how the loyal lieutenants achieve consensus despite the traitors' attempts to disrupt it.

Round 1: Loyal Commander Sends Consistent Messages

In the first round, the loyal commander G_0 sends the same command to all lieutenants. In this case, the commander chooses Attack (A). Figure 3 illustrates what happens in Round 1:

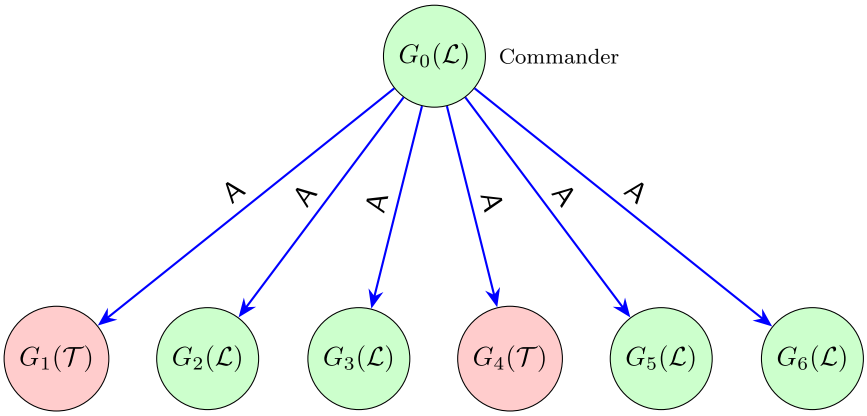

Figure 4: Round 1: Loyal Commander G_0 sends consistent Attack messages to all lieutenants.

$G_1(\mathcal{T})$	$G_2(\mathcal{L})$	$G_3(\mathcal{L})$	$G_4(\mathcal{T})$	$G_5(\mathcal{L})$	$G_6(\mathcal{L})$
A					
	A				
		A			
			A		
				A	
					A

The grid above shows the initial values v_i that each lieutenant G_i (represented by row and column headers) receives from Commander G_0 and stores. Since the commander is loyal, all lieutenants receive the same command (Attack). Each value is placed on the diagonal cell (G_i, G_i). Off-diagonal cells are empty, to be filled by messages exchanged between lieutenants in later algorithm phases.

As shown in the diagram, the loyal commander G_0 sends consistent Attack (A) messages to all lieutenants, including the two traitors (G_1 and G_4). This is a fundamental aspect of the algorithm - when the commander is loyal, they always send the same message to everyone.

At this point, each lieutenant only knows what the commander told them directly. The table in Figure 3 represents this initial knowledge state, with each lieutenant recording their received command in their own diagonal cell. Since the commander is loyal in our scenario, all cells contain Attack (A) commands. The off-diagonal cells remain empty because the lieutenants haven't yet exchanged messages with each other.

Round 2: Traitors Attempt to Disrupt Consensus

In the second round, each lieutenant acts as the commander for the round and tells all other lieutenants what command they received from the original commander in Round 1.

Figure 4 illustrates how traitors attempt to contradict the commander's orders during Round 2:

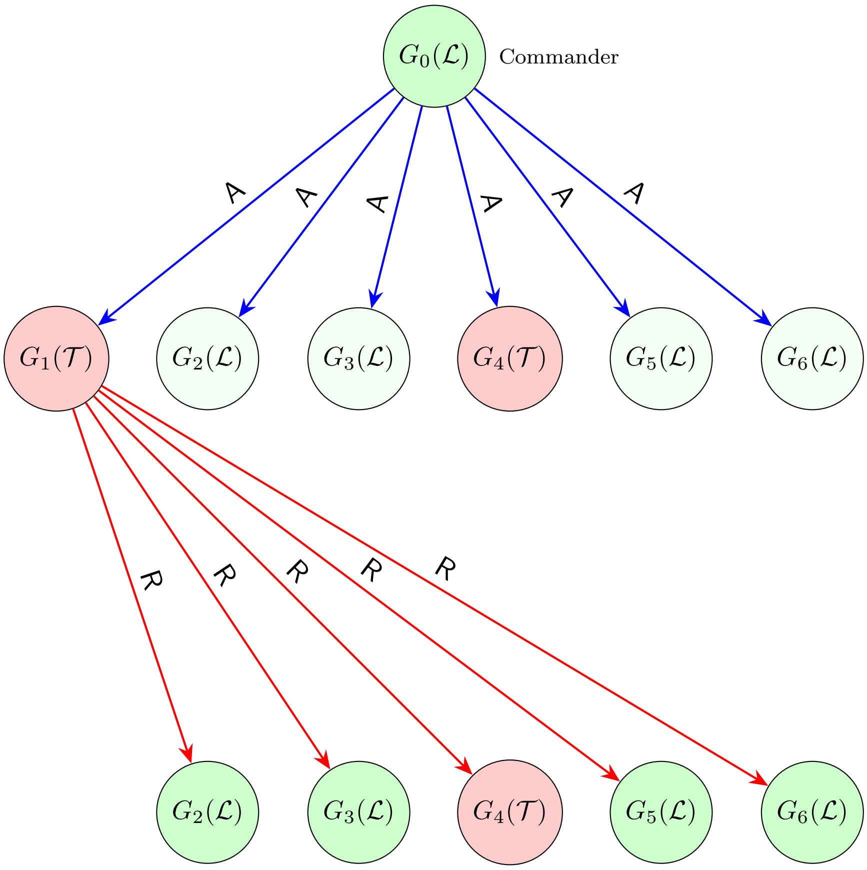

Figure 5: Round 2: Traitor lieutenants G_1 and G_4 deliberately contradict the loyal commander's original Attack order by sending Retreat messages to all other lieutenants.

		Receiver					
		G_1	G_2	G_3	G_4	G_5	G_6
Sender	G_1						
	G_2		R	**R**	**R**	**R**	**R**
	G_3		R	**R**	**R**	**R**	**R**
	G_4		A	A	**R**	A	A
	G_5		R	**R**	**R**	**R**	**R**
	G_6		R	**R**	**R**	**R**	**R**
	Majority		R	R	R	R	R

Legend: Rows (left) are senders, columns (top) are receivers. Each cell shows the message the sender tells the receiver that it received from the loyal commander. Yellow diagonal: sender's own message from the commander. Grey cells: G_1 is the acting commander and does not participate in consensus. Blue = Attack (A), Red = Retreat (R).

$G_1(\mathcal{T})$	$G_2(\mathcal{L})$	$G_3(\mathcal{L})$	$G_4(\mathcal{T})$	$G_5(\mathcal{L})$	$G_6(\mathcal{L})$
A	**R**	**R**	**R**	**R**	**R**
	A				
		A			
			A		
				A	
					A

After Round 2, each loyal lieutenant can determine the consensus by taking a majority vote of all reported values. Despite the traitors contradicting the commander's message, all loyal lieutenants reach the same conclusion: Attack (A), which matches the loyal commander's original order.

In another scenario, we see how loyal lieutenants faithfully relay the commander's original message:

In the second round, each lieutenant shares with all other lieutenants what they received from the commander in Round 1.

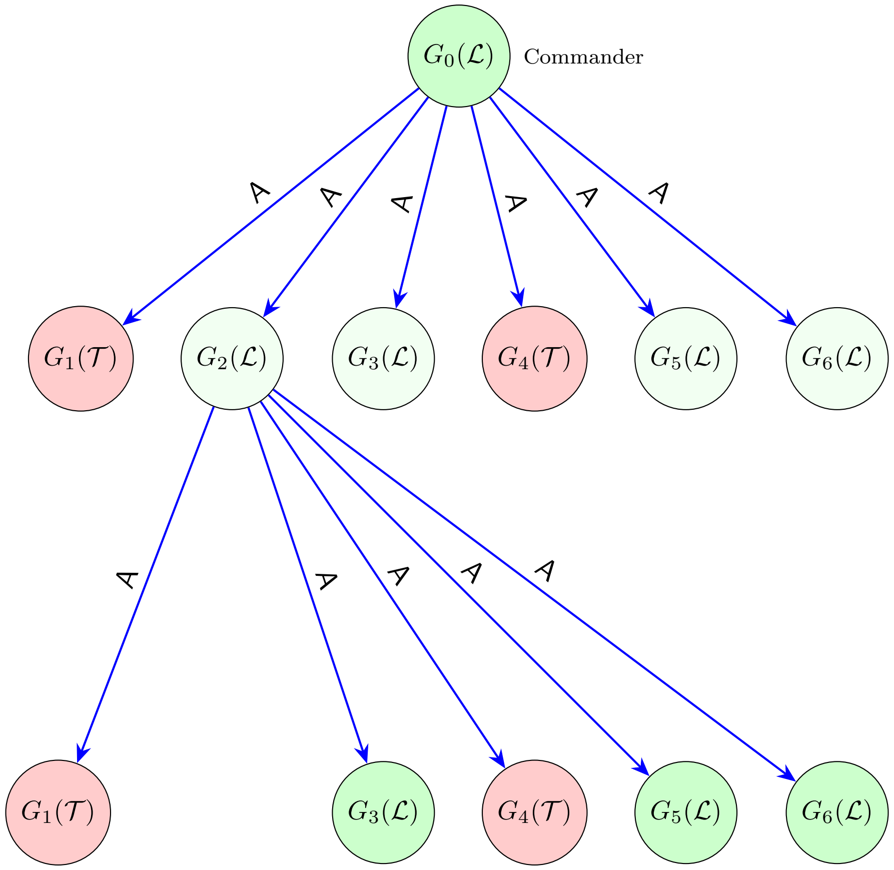

Figure 6: Round 2: Loyal lieutenants faithfully relay the Attack messages they received from the loyal commander, maintaining consistency despite traitors' attempts to disrupt consensus.

Receiver

Sender	G_1	G_2	G_3	G_4	G_5	G_6
G_1	A		**R**	**R**	**R**	**R**
G_2						
G_3	A		**A**	A	A	A
G_4	**R**		**R**	A	**R**	**R**
G_5	A		A	A	**A**	A
G_6	A		A	A	A	**A**
Majority	**A**		**A**	**A**	**A**	**A**

Legend: Rows (left) are senders, columns (top) are receivers. Each cell shows the message the sender tells the receiver that it received from the loyal commander. Yellow diagonal: sender's own message from the commander. Grey cells: G_2 is the acting commander and does not participate in consensus. Blue = Attack (A), Red = Retreat (R).

$G_1(\mathcal{T})$	$G_2(\mathcal{L})$	$G_3(\mathcal{L})$	$G_4(\mathcal{T})$	$G_5(\mathcal{L})$	$G_6(\mathcal{L})$
A	R	R	R	R	R
A	A	A	A	A	A
		A			
			A		
				A	
					A

After Round 2, each loyal lieutenant can determine the consensus by taking a majority vote of all reported values. Despite the traitors contradicting the commander's message, all loyal lieutenants reach the same conclusion: Attack (A), which matches the loyal commander's original order.

After all lieutenants have exchanged their information, we can represent the final state of knowledge with the following comprehensive table:

$G_1(\mathcal{T})$	$G_2(\mathcal{L})$	$G_3(\mathcal{L})$	$G_4(\mathcal{T})$	$G_5(\mathcal{L})$	$G_6(\mathcal{L})$
A	R	R	R	R	R
A	A	A	A	A	A
A	A	A	A	A	A
R	R	R	A	R	R
A	A	A	A	A	A
A	A	A	A	A	A
A	A	A	A	A	A

After all lieutenants have exchanged their Round 2 messages, each loyal lieutenant can determine the consensus by taking a majority vote of all reported values. Despite the conflicting information sent by the traitor lieutenants G_1 and G_4, all loyal lieutenants reach the same conclusion: Attack (A), which matches the loyal commander's original order and demonstrates the algorithm's ability to satisfy the validity condition.

This is where the traitors' malicious behavior becomes evident. Despite both traitor lieutenants (G_1 and G_4) receiving Attack (A) from the loyal commander, they attempt to create confusion:

- Traitor G_1 consistently contradicts the commander by telling all other lieutenants that it received "Retreat" (R) instead of the actual "Attack" (A) command
- Similarly, traitor G_4 also sends false information, telling all other lieutenants it received "Retreat" (R) when it actually received "Attack" (A)

Meanwhile, all loyal lieutenants (G_2, G_3, G_5, and G_6) honestly relay the Attack (A) command they received from the commander.

After this round, each lieutenant has:

- Their original command from the commander (from Round 1)
- Reports from all other lieutenants about what they claim to have received

The final table shows what each lieutenant knows after all Round 2 messages have been exchanged. The diagonal cells (highlighted in yellow) show what each lieutenant itself received from the commander. The off-diagonal cells show what each lieutenant was told by others.

Now, each loyal lieutenant can determine what command to follow by taking a majority vote of all reported values. As shown in the "Majority" row of the table, despite the conflicting information from the traitors, all loyal lieutenants reach the same conclusion: Attack (A), which matches the loyal commander's original order.

This demonstrates an important aspect of the OM(2) algorithm: when the commander is loyal, the algorithm ensures that all loyal lieutenants follow the commander's order, even if traitor lieutenants attempt to disrupt the process. The algorithm successfully satisfies both the Agreement and Validity requirements.

Conclusion

The Byzantine Generals' Problem represents one of the fundamental challenges in distributed computing. Through our examination of the Oral Message algorithm (OM), we've seen how it's possible to achieve consensus even in the presence of malicious actors.

Key insights from our exploration:

- With n generals and at most t traitors, consensus requires $n \geq 3t + 1$ and $m \geq t$ rounds of communication.
- The recursive nature of the algorithm allows loyal generals to filter out inconsistent information.
- Even with a traitor commander sending contradictory commands, loyal generals can still reach agreement.
- The solution works without requiring any form of message authentication or signatures.

This algorithm laid the foundation for many fault-tolerant distributed systems we rely on today, from cloud databases to blockchain networks. Modern consensus protocols like Practical Byzantine Fault Tolerance (PBFT) and those used in blockchain systems are direct descendants of Lamport's original solution.

Understanding these principles not only helps in designing robust distributed systems but also provides insight into the fundamental limits of what's achievable in the presence of Byzantine failures.