Lean Consensus: Ejemplo Práctico

¿Qué es Lean Consensus?

Lean Consensus combina dos mecanismos:

- **Casper FFG** (Friendly Finality Gadget): Proporciona finalidad económica
- **LMD-GHOST** (Latest Message-Driven Greedy Heaviest Observed SubTree): Selecciona la cadena canónica

Ejemplo Paso a Paso

Configuración Inicial

Tenemos 4 validadores con sus stakes:

- **Validator A**: 32 ETH
- **Validator B**: 32 ETH
- **Validator C**: 32 ETH
- **Validator D**: 32 ETH

Total stake: 128 ETH **Supermayoría (2/3)**: 85.33 ETH (necesitamos al menos 3 validadores)

Epoch 0: Bloque Génesis

```
Genesis (Justified ✓, Finalized ✓)
```

Este es nuestro punto de partida. El bloque génesis está tanto justificado como finalizado por definición.

Epoch 1: Primera Propuesta

```
Genesis (J✓, F✓)
   |
   A1 ← Propuesto por Validator A
```

Votos FFG para A1:

- Source: Genesis (J✓, F✓)
- Target: A1

Validator	Vota por A1	Stake
A	✓	32 ETH
B	✓	32 ETH
C	✓	32 ETH
D	✓	32 ETH

Resultado: A1 recibe 128 ETH → **A1 queda JUSTIFICADO** (>2/3)

```
Genesis (J✓, F✓)
   |
```

```
   A1 (J✓)
```

Epoch 2: Fork en la Cadena

Dos validadores proponen bloques simultáneamente:

```
Genesis (J✓, F✓)
   |
   A1 (J✓)
   |
   +--B2a ← Propuesto por Validator B
   |
   +--C2b ← Propuesto por Validator C
```

¿Cuál cadena seguir? Aquí entra **LMD-GHOST**.

Votos LMD (Latest Message):

- Validator A: vota por B2a
- Validator B: vota por B2a
- Validator C: vota por C2b
- Validator D: vota por B2a

Cálculo del peso (stake acumulado):

- **B2a**: 32 + 32 + 32 = 96 ETH
- **C2b**: 32 ETH

LMD-GHOST selecciona B2a (más peso)

```
Genesis (J✓, F✓)
   |
   A1 (J✓)
   |
   B2a (canonical) ← 96 ETH
   |
   C2b (orphan) ← 32 ETH
```

Epoch 3: Finalización

Los validadores votan por B2a:

Votos FFG para B2a:

- Source: A1 (J✓)
- Target: B2a

Validator	Vota	Stake
A	✓	32 ETH
B	✓	32 ETH
C	✓	32 ETH
D	✓	32 ETH

Resultado: B2a recibe 128 ETH → **B2a queda JUSTIFICADO**

Regla de Finalización: Como A1 estaba justificado y ahora B2a (hijo directo) también está justificado, entonces **A1 queda FINALIZADO**.

```
Genesis (J✓, F✓)
   |
   A1 (J✓, F✓) ← ¡Ahora finalizado!
   |
   B2a (J✓)
```

Ahora construimos sobre B2a:

```
Genesis (J✓, F✓)
   |
   A1 (J✓, F✓)
   |
   B2a (J✓)
   |
   D3 ← Propuesto por Validator D
```

Epoch 4: Ataque del 51%

Supongamos que los Validators C y D se vuelven maliciosos e intentan revertir la cadena:

```
Genesis (J✓, F✓)
   |
   A1 (J✓, F✓) ← FINALIZADO (no se puede revertir)
   |
   +--B2a (J✓)
   |  |
   |  D3
   |
   +--X2 (cadena atacante)
      |
      Y3
```

¿Pueden revertir A1?

NO. Porque A1 está **finalizado**. Para revertirlo necesitarían:

1. Violar las reglas de slashing de Casper FFG
2. Perder sus stakes (32 ETH cada uno = 64 ETH)

Condición de Slashing:

```
Si votan por X2 con source < A1, cometen "surround voting"
→ Penalización: pérdida total del stake
```

Ejemplo de Slashing

El Validator C intenta hacer doble voto:

Voto 1:

- Source: A1 (epoch 1)
- Target: B2a (epoch 2)

Voto 2 (malicioso):

- Source: Genesis (epoch 0)
- Target: X2 (epoch 2)

```
Detección: Genesis < A1 y X2 = B2a (mismo epoch)
→ SLASHING: Validator C pierde sus 32 ETH
```

Propiedades Clave

1. Safety (Seguridad)

Los bloques finalizados **nunca** se pueden revertir sin que los atacantes pierdan >1/3 del total stake.

2. Liveness (Vivacidad)

La cadena continúa creciendo siempre que >2/3 de los validadores estén honestos y online.

3. Accountable Safety

Si dos bloques conflictivos se finalizan, podemos identificar exactamente qué validadores violaron las reglas (≥1/3 del stake) y slashearlos.

Resumen del Flujo

```
1. PROPUESTA
   └─> Validator propone bloque

2. LMD-GHOST
   └─> Validadores votan por el fork con más stake
   └─> Se selecciona la cadena canónica

3. JUSTIFICACIÓN (Casper FFG)
   └─> Si >2/3 del stake vota por un checkpoint
   └─> El checkpoint queda JUSTIFICADO

4. FINALIZACIÓN (Casper FFG)
   └─> Si checkpoint hijo está justificado
   └─> El checkpoint padre queda FINALIZADO

5. SLASHING
   └─> Detecta violaciones (double vote, surround vote)
   └─> Penaliza validadores maliciosos
```

Parámetros de Ethereum 2.0

- **Epoch**: 32 slots (~6.4 minutos)
- **Slot**: 12 segundos
- **Finalización típica**: 2 epochs (~12.8 minutos)

- **Mínimo para justificar**: >2/3 del stake activo
- **Penalización por slashing**: Pérdida del stake + expulsión

Conclusión

Lean Consensus combina:

- **LMD-GHOST**: Para elegir la mejor cadena en tiempo real (liveness)
- **Casper FFG**: Para garantizar finalidad económica (safety)

Esta combinación permite que Ethereum 2.0 tenga tanto finalidad probabilística rápida como finalidad absoluta económica.