Mysticeti
The new core of the Sui blockchain

Alberto Sonnino
Tailoring the Talk

Do you know:

1. How blockchains work (roughly)?
2. What Byzantine Fault Tolerance (BFT) means?
3. What DAG-based consensus are?
4. How Narwhal / Bullshark work (roughly)?
Byzantine Fault Tolerance

$> \frac{2}{3}$
Byzantine Fault Tolerance

\[ \geq 2f+1 \]

\[ 3f+1 \]
Partial Synchrony

GST

time
Blockchains

1. make transaction
Blockchains

1. make transaction
2. submit transaction
1. make transaction
2. submit transaction
3. sequence and verify
Keeping the Talk Short

**In scope**
- Ordering (quorum-based)

**Not in scope**
- Nodes selection?
- Committee reconfiguration?
- Transactions execution?
- Transactions language?
- Financial incentives?
- etc
Why?
Latency

![Graph showing latency vs. throughput for different systems and configurations.](image-url)
In a year of running Sui:
Why?
Crash Faults

In a year of running Sui:

- How many Byzantine faults?
Why?
Crash Faults

In a year of running Sui:

• How many Byzantine faults? 0
Why?
Crash Faults

In a year of running Sui:

• How many Byzantine faults? 0

• How many Crash faults?
Why?
Crash Faults

In a year of running Sui:

• How many Byzantine faults? 0

• How many Crash faults? 😭
Why?
Engineering Complexity

Narwhal mempool

Client

Worker 1
Worker 2
Primary

Worker n

Batch

Tx

Batch

Batch
Lamport Diagram

message created by node 1

message from node 1 to node 2

time
The Mysticeti DAG

Uncertified DAG
The Mysticeti DAG
Block Creation

- Round number
- Author
- Payload (transactions)
- Signature
The Mysticeti DAG
Rule 1: Link to 2f+1 parents

- Total nodes: $3f+1 = 4$
- Quorum: $2f+1 = 3$
The Mysticeti DAG

Rule 2: Every node waits and links to leaders
The Mysticeti DAG

Rule 3: All node run in parallel
Main Ingredient:

All messages embedded in the DAG

- Fewer signatures
- Isolated engineering component
- Define interpretable patterns on the DAG
- Run multiple protocols on the same DAG
Interpreting DAG Patterns

Certificate

Blame
Two Protocols, One DAG

**Mysticeti-C Consensus**

- No rounds without leader
- Multiple leaders per round

**Mysticeti-FPC Adding Fast Finality**

- Interpret BCB on DAG
Mysticeti-C

The consensus protocol
End Goal
Ordering leaders

- We focus on ordering leaders: L1, L4, L7
• We focus on ordering leaders: L1 L4 L7
• Linearising the sub-DAG is simple
DAG Structure

wave 1
- r1
- r2
- r3

wave 2
- r4
- r5
- r6

wave 3
- r7

L1

L4

L7

propose
- vote
- certify

propose
- vote
- certify

propose
DAG Structure
DAG Structure
DAG Structure
Practical Implementation
Select only 2 leaders per round
Interpreting DAG Patterns

Reminder
Direct Decision Rule

On each leader starting from highest round:

- **Skip** if $2f+1$ blames
- **Commit** if $2f+1$ certifies
- **Undecided** otherwise
Direct Decision Rule

On each leader starting from highest round:

- **Skip** if 2f+1 blames
- **Commit** if 2f+1 certifies
- **Undecided** otherwise
Direct Decision Rule

On each leader starting from highest round:

- **Skip** if 2f+1 blames
- **Commit** if 2f+1 certifies
- **Undecided** otherwise
Direct Decision Rule

On each leader starting from highest round:

- **Skip** if $2f+1$ blames
- **Commit** if $2f+1$ certifies
- **Undecided** otherwise
Indirect Decision Rule
1. Find Anchor

- First block with round > r+2 that is **Commit** or **Undecided**

![Indirect Decision Rule Diagram](image)
1. Find Anchor

- First block with round > r+2 that is **Commit** or **Undecided**

2. Certified link

- **Commit** if B <-> certified link <-> A otherwise **Skip**
All Start at Undecided
Ignore Incomplete Waves
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule

Diagram showing relationships between elements labeled L1a, L1b, L2a, L2b, L3b, L4a, L4b, L5b, L6a, r1, r2, r3, r4, r5, and r6.
Apply Direct Rule
Apply Direct Rule
Ignore Missing Leader
Apply Direct Rule
Apply Direct Rule
<table>
<thead>
<tr>
<th></th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>r4</th>
<th>r5</th>
<th>r6</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L4a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L4b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L5a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L5b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Undecided
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Indirect Rule
Apply Indirect Rule
Apply Indirect Rule

Anchor
Apply Direct Rule
Apply Indirect Rule

Skipped
Apply Indirect Rule
Current Status
Commit Sequence
Stop at the first Undecided leader

leaders sequence: L1a L1b L2a L2b L3a L3b L4a L4b
Commit Sequence

Remove skipped leaders

leaders sequence: L1a L1b L2a L2b L3a L3b L4a L4b
Commit Sequence
Final leader sequence

leaders sequence: L1b L2a
leaders sequence: L1b L2a
output sequence:
leaders sequence:

L2a, L1b

output sequence:

L1b
Commit Sequence
Commit sub-dag

leaders sequence: L2a
output sequence: L1b
Commit Sequence
Commit sub-dag

leaders sequence: L2a

output sequence: L1b
Commit Sequence
Commit sub-dag

leaders sequence:

output sequence:
Mitigating slow leaders
Compute Reputation Scores

node 1: 3
Compute Reputation Scores

node 1: 3
node 2: 4
Compute Reputation Scores

node 1: 3  node 2: 4  node 3: 2
Compute Reputation Scores

node 1:  3  
node 2:  4  
node 3:  2  
node 4:  2
Future Leaders

node 1: 3  
node 2: 4  
node 3: 2  
node 4: 2
Security Intuition
At most $L_1$ or $L_1'$ can have a certificate pattern (quorum intersection)
- At most L1 or L1' can have a certificate pattern (quorum intersection).
- If L1 has 2f+1 certificate patterns, A always has a certified link to L1.
Security Intuition

- At most \( L_1 \) or \( L_1' \) can have a certificate pattern (quorum intersection).
- If \( L_1 \) has \( 2f+1 \) certificate patterns, \( A \) always has a certified link to \( L_1 \).
- After GST, the direct decision rule commits a block.
Security Intuition

Leader Timeout:
Wait for 2f+1 parents + 250 ms
Mysticeti-FPC

Adding a fast commit path
Consensus Not Required

Coins, balances, and transfers

NFTs creation and transfers

Game logic allowing users to combine assets

Inventory management for games / metaverse

Auditable 3rd party services not trusted for safety

...
Consensus Required

- Increment a publicly-accessible counter
- Collaborative in-game assets
- Auctions
- ...
Object Type

**Owned Objects**
- Objects that can be mutated by a single entity
- e.g., My bank account
- **Do not need consensus**

**Shared Objects**
- Objects that can be mutated by multiple entities
- e.g., A global counter
- **Need consensus**
System State

Objects:
  • Unique ID
  • Version number
  • Ownership Information
  • Type (shared, owned)
Fast Execution

owned: Tx1
shared: Tx2
owned: Tx3
shared: Tx4
shared: Tx5
owned: Tx6
Fast Execution

- owned: Tx1
- shared: Tx2
- owned: Tx3
- shared: Tx4
- shared: Tx5
- owned: Tx6

Certificate

Execute
Fast Execution

Certificate

Execute

owned: Tx1
shared: Tx2
owned: Tx3
shared: Tx4
shared: Tx5
owned: Tx6

L1
No Finality

- owned: Tx1
- shared: Tx2
- owned: Tx3
- shared: Tx4
- shared: Tx5
- owned: Tx6

Epoch Change

node 4: revert Tx1 and Tx3
Fast Path Finality (1)

- **Owned:** Tx1, Tx3, Tx6
- **Shared:** Tx2, Tx4, Tx5

Certicates

2f+1

Diagram showing connections among Tx1, Tx3, and various r1, r2, r3 nodes.
Mixed-Objects Transactions

owned: Tx1
shared: Tx2
owned: Tx3
shared: Tx4
mixed: Tx5
owned: Tx6
Mixed-Objects Transactions

- owned: Tx1
- shared: Tx2
- owned: Tx3
- shared: Tx4
- mixed: Tx5
- owned: Tx6

Commit:
- Tx2
- Tx4
Mixed-Objects Transactions

- owned: Tx1
- shared: Tx2
- owned: Tx3
- shared: Tx4
- mixed: Tx5
- owned: Tx6
Mixed-Objects Transactions

owned:  Tx1
shared:  Tx2
owned:  Tx3
shared:  Tx4
mixed:  Tx5
owned:  Tx6

2f+1 Certificates
Commit

consensus
Mixed-Objects Transactions

lock owned objects

commit the lock on owned objects
Preliminary Benchmarks

More to come soon
Implementation

- Written in Rust
- Networking: Tokio (TCP)
- Storage: custom WAL
- Cryptography: ed25519-consensus

https://github.com/mystenlabs/mysticeti
Implementation

- Synchronous core
- One Tokio task per peer (limiting resource usage)
- DTE simulator

https://github.com/mystenlabs/mysticeti
Preliminary Results

![Graph showing latency vs. throughput for different setups.]

- **Mysticeti-C - 10 nodes**
- **Mysticeti-C - 19 nodes**
- **bullshark - 10 nodes**
- **bullshark - 20 nodes**
## Engineering Benchmarks

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Committee</th>
<th>Load/TPS</th>
<th>P50</th>
<th>P95</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bullshark</td>
<td>137</td>
<td>5k</td>
<td>2.89 s</td>
<td>4.60 s</td>
</tr>
<tr>
<td>Mysticeti</td>
<td>137</td>
<td>5k</td>
<td>650 ms</td>
<td>975 ms</td>
</tr>
</tbody>
</table>

We ran it for 24h and it looks good 👍
Testing Strategy

- Compare performance & robustness
- Test mainnet change bullshark -> mysticeti
- Prepare for the worst mysticeti -> bullshark
Narwhal vs Mysticeti

Key differences & Insight
Narwhal vs Mysticeti

Narwhal

Mysticeti

Round 1

header

certificate

H V C

H V C

H V C

Block

B

B

B
Main Challenge
Possible equivocations
Main Challenge
Possible equivocations (even with 2f+1 support)
<table>
<thead>
<tr>
<th>Decision Rules</th>
<th>Upon interpreting the DAG...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bullshark</strong></td>
<td>• A leader is <strong>Commit</strong> or not</td>
</tr>
<tr>
<td></td>
<td>• Either directly or indirectly (recursion)</td>
</tr>
<tr>
<td><strong>Mysticet</strong></td>
<td>• A leader is <strong>Commit</strong>, <strong>Skip</strong>, or <strong>Undecided</strong></td>
</tr>
<tr>
<td></td>
<td>• Either directly or indirectly (recursion)</td>
</tr>
</tbody>
</table>
Linear Chain vs DAG
Quorum-Based Consensus

**Linear-Chain**
- Low latency
- Fragile to faults
- Complex leader-change

**DAG-Based**
- High latency
- Robust against faults
- No/Simple leader-change
Linear-Chain Consensus

Rough overview
Linear-Chain Consensus

Rough overview
Linear-Chain Consensus

Rough overview
Linear-Chain Consensus

Rough overview

• The leader does all the work
Linear-Chain Consensus
Rough overview

- The leader does all the work
- Complex leader-change
Linear-Chain Consensus

Rough overview

• The leader does all the work
• Complex leader-change
DAG-Based Consensus
Rough overview

r1

r2

r3

r4

r5
DAG-Based Consensus
Rough overview
DAG-Based Consensus

Rough overview
DAG-Based Consensus
Rough overview

[Diagram of a directed acyclic graph (DAG) with nodes labeled r1, r2, r3, r4, r5, L1, and L2 connected by directed edges.]
Mysticeti

- A single message type
- Interpret patterns on the DAG

- **Paper:** https://sonnino.com/papers/mysticeti.pdf
- **Code:** https://github.com/mystenlabs/mysticeti