Mysticeti
The new core of the Sui blockchain

Alberto Sonnino
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

> 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
Blockchains

1. make transaction

2. submit transaction

3. sequence and verify
1. make transaction

2. submit transaction

3. sequence and verify

4. store
Keeping the Talk Short

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

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

Low-latency DAG consensus with fast commit path
Lamport Diagram

message created by node 1

message from node 1 to node 2

node 1

node 2

node 3

node 4

time
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 nodes run in parallel
Main Ingredient:

All messages embedded in the DAG

- Fewer signatures
- Simpler synchronisation
- 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
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
DAG Structure

Wave 1:
- r1
- r2
- r3

L1
- Propose
- Certify

Wave 2:
- r4
- r5
- r6

L4
- Propose
- Vote

Wave 3:
- r7

L7
- Propose
- Vote
- Certify
DAG Structure
Practical Implementation
Select only 2 leaders per round
Interpreting DAG Patterns

wave 1

L1

Certificate

propose vote certify

Blame

Reminder
Direct Decision Rule

On each leader starting from highest round:

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

On each leader starting from highest round:

- **Skip** if 2f+1 blames
- **Commit** if 2f+1 certifies
- **Undecided** otherwise
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 \(2f+1\) certificates
- **Undecided** otherwise
Indirect Decision Rule
1. Find Anchor

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

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

Diagram showing connections between nodes labeled as L1a, L1b, L2a, L2b, L3b, L4a, L4b, and L5a, with lines indicating relationships or interactions between them.
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Ignore Missing Leader
Apply Direct Rule
Apply Direct Rule
Apply Indirect Rule

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

Skipped
Apply Indirect Rule
Apply Indirect Rule
Apply Direct Rule
Apply Indirect Rule

Skipped
Apply Indirect Rule

Anchor

Apply Indirect Rule
Apply Indirect Rule

- L1a
- L1b
- L2b
- L4a
- L4b
- L5a
- L5b
- L6a

no certified link

Anchor
Current Status
Commit Sequence

Take all leaders in order

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

Commit sub-dag

leaders sequence: L1b L2a

output sequence:
leaders sequence:

output sequence:
Commit Sequence
Commit sub-dag

leaders sequence: L2a

output sequence: L1b
leaders sequence: L2a

output sequence: L1b
Commit Sequence
Commit sub-dag

leaders sequence:
output sequence:
Slow Leaders are Annoying

Suffer from them only when under attack or bad network
HammerHead
Compute Reputation Scores
HammerHead
Compute Reputation Scores

node 1: 3
HammerHead
Compute Reputation Scores

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

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

dnode 1: 3
dnode 2: 4
dnode 3: 2
dnode 4: 2
HammerHead
Future Leaders

node 1: 3  node 2: 4  node 3: 2  node 4: 2
Security Intuition
Security Intuition

- At most $L_1$ or $L_1'$ can have a certificate pattern (quorum intersection)
Security Intuition

- 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
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
- Market places
- ...

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
Fast Execution

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

Certificate

Execute
Fast Execution

Certificate

Execute
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)
Fast Path Finality (2)

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

commit of certificate
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

Certificate

Execute

L1

Tx1

Tx5
Mixed-Objects Transactions

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

```
2f+1
```

Commit

```
Certi
```

consensus
Mixed-Objects Transactions

- **Lock owned objects**
- **Commit the lock on owned objects**
Mysticeti

- A single message type
- Interpret patterns on the DAG

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

Preliminary Benchmarks
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
Evaluation
Experimental setup on AWS

m5d.8xlarge
Preliminary Results

![Graph showing latency vs. throughput for different node configurations. The graph compares Mysticeti-C and bullshark with 10 nodes, Mysticeti-C with 19 nodes, and bullshark with 20 nodes. The y-axis represents latency in seconds, and the x-axis represents throughput in transactions per second (tx/s).]
EXTRA

Narwhal vs Mysticeti
Narwhal vs Mysticeti

Narwhal

Mysticeti

Round 1

header certificate

Block

Round 1

G1

G2

G3
Main Challenge
Possible equivocations
Main Challenge
Possible equivocations (even with 2f+1 support)
Decision Rules
Upon interpreting the DAG...

Bullshark
- A leader is **Commit** or not
- Either directly or indirectly (recursion)

Mysticeti
- A leader is **Commit**, **Skip**, or **Undecided**
- Either directly or indirectly (recursion)
EXTRA

Linear 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
DAG-Based Consensus
Rough overview
DAG-Based Consensus
Rough overview
DAG-Based Consensus
Rough overview