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
Byzantine Fault Tolerance

\[ \geq 2f+1 \]

\[ 3f+1 \]
Blockchains

1. make transaction
Blockchains

1. make transaction
2. submit transaction
Blockchains

1. make transaction
2. submit transaction
3. sequence and verify
Blockchains

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
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
The Mysticeti DAG
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 vote certify propose vote certify propose

wave 2
r4
r5
r6

L4

wave 3
r7

L7
Interpreting DAG Patterns

wave 1
r1  r2  r3

Certificate

Blame

propose  vote  certify
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 $2f+1$ certificates
- **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
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
     
     \[ B \leftrightarrow \text{certified link} \leftrightarrow A \]
     otherwise **Skip**
Apply Direct Rule
Mark all leaders as Undecided
Apply Direct Rule

Cannot decide incomplete waves
Apply Direct Rule
Start with latest block and go backward
Apply Direct Rule
Start with latest block and go backward
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule
Apply Direct Rule

Direct rule cannot decide
Apply Indirect Rule
Find anchor & Check certified links

Undecided
Apply Direct Rule
Apply Direct Rule
Apply Indirect Rule

Undecided
Apply Direct Rule
Apply Direct Rule
Apply Indirect Rule
Find anchor & Check certified links

Skipped
Apply Indirect Rule
Find anchor & Check certified links

Commit
Apply Indirect Rule
Find anchor & Check certified links
Apply Direct Rule
Apply Direct Rule
Apply Indirect Rule
Apply Indirect Rule
Find anchor & Check certified links

Commit
Apply Indirect Rule
Find anchor & Check certified links
Apply Direct Rule
Apply Direct Rule
Apply Indirect Rule
Find anchor & Check certified links
Apply Indirect Rule
Find anchor & Check certified links

Commit
Apply Indirect Rule
Find anchor & Check certified links

Commit
Apply Direct Rule
Apply Direct Rule
Current Status
Commit Sequence
Take all leaders in order

sequence: L1a L1b L1c L1d L2a L2b L2c L3a L3b L3c L3d L4a L4b L4c L4d
Commit Sequence
Stop at the first Undecided leader
Current Status
Remove skipped leaders

sequence: L1a L1b L1c L1d L2a L2b L2c L2d L3a L3b L3c L3d L4a L4b L4c L4d
Current Status
Remove skipped leaders

sequence: L1a L1c L1d L2a
Practical Implementation
Select only 2 leaders per round
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  node 4: 2
HammerHead
Future Leaders

node 1: 3    node 2: 4    node 3: 2    node 4: 2
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
- Marketplaces
- ...
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
Fast Execution

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

Certificate

Execute

Tx1
Tx3
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

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

2f+1 Certificates
Fast Path Finality

- 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

Diagram showing the relationships between transactions.
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

r1 r2 r3

Tx1

Tx5
Mixed-Objects Transactions

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

Certificates

2f+1 Commit

Tx5

L1

L2

r1 → r2 → r3 → r4 → r5
Mixed-Objects Transactions

lock owned objects

commit the lock on owned objects
Summary

Mysticeti

• A single message type
• Interpret patterns on the DAG

• Code: https://github.com/mystenlabs/mysticeti
EXTRA

Open Questions
Questions

- Anything obviously wrong?
- Is the protocol simple enough?
- What engineering challenges do you foresee?
- Suggested improvements?
- Is the fast path worth its complexity?
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

The graph compares the latency (s) with throughput (tx/s) for different node configurations:
- Mysticeti-C: 10 nodes and 19 nodes
- Bullshark: 10 nodes and 20 nodes

The graph shows that as the throughput increases, the latency also increases, with noticeable variance between the configurations.
EXTRA

Narwhal vs Mysticeti
Narwhal vs Mysticeti

Narwhal

Round 1

header certificate

Mysticeti

Round 1

Block
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)
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

r1     r2     r3     r4     r5

L1

L2
DAG-Based Consensus
Rough overview