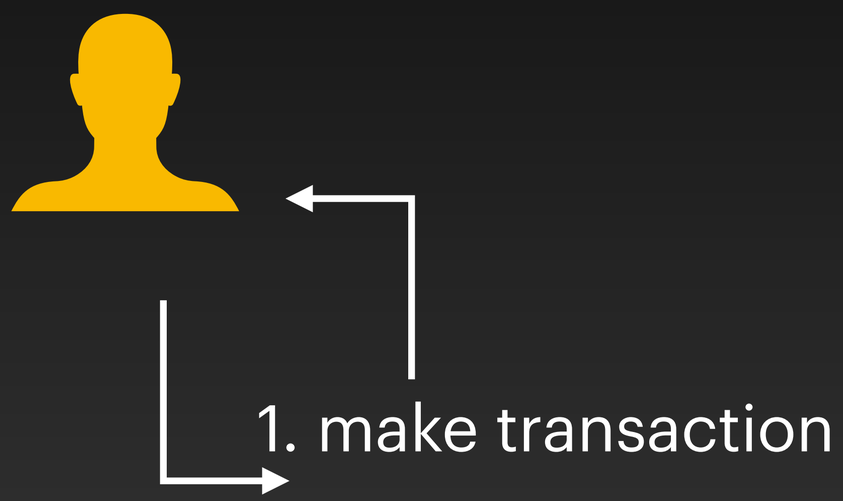
The Sui Smart Contract Platform

Alberto Sonnino

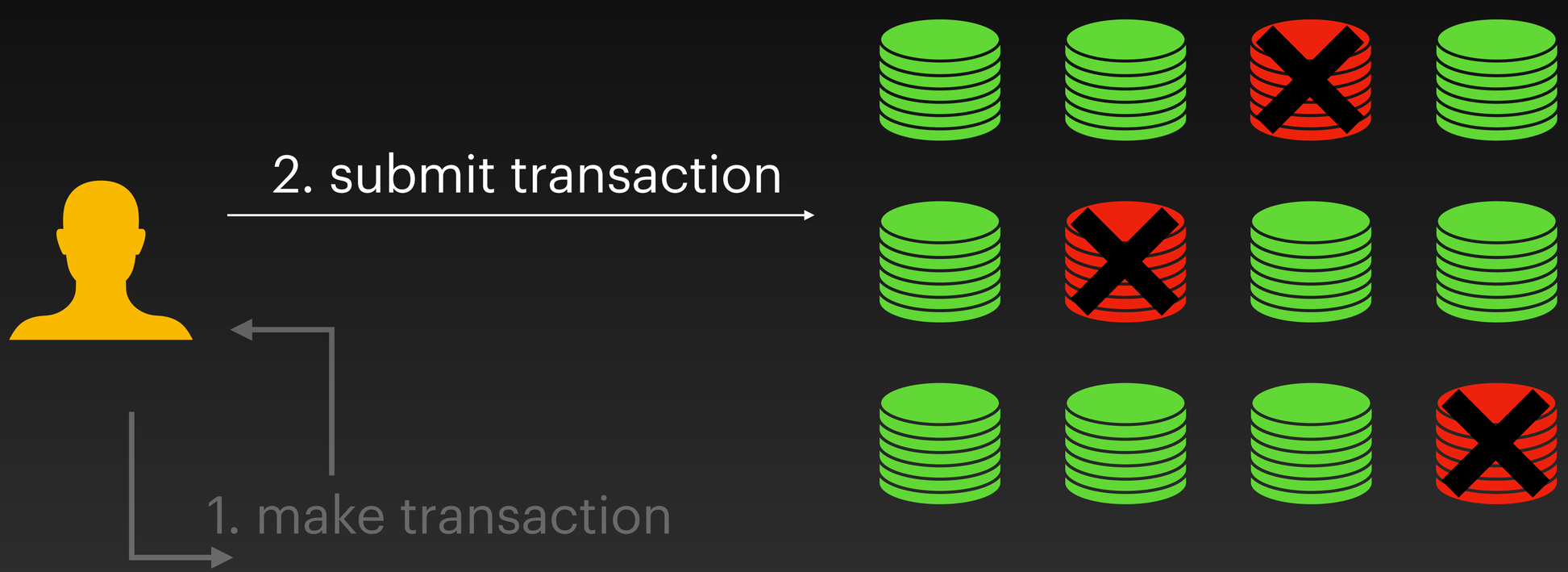
Byzantine Fault Tolerance

Byzantine Fault Tolerance

Blockchains

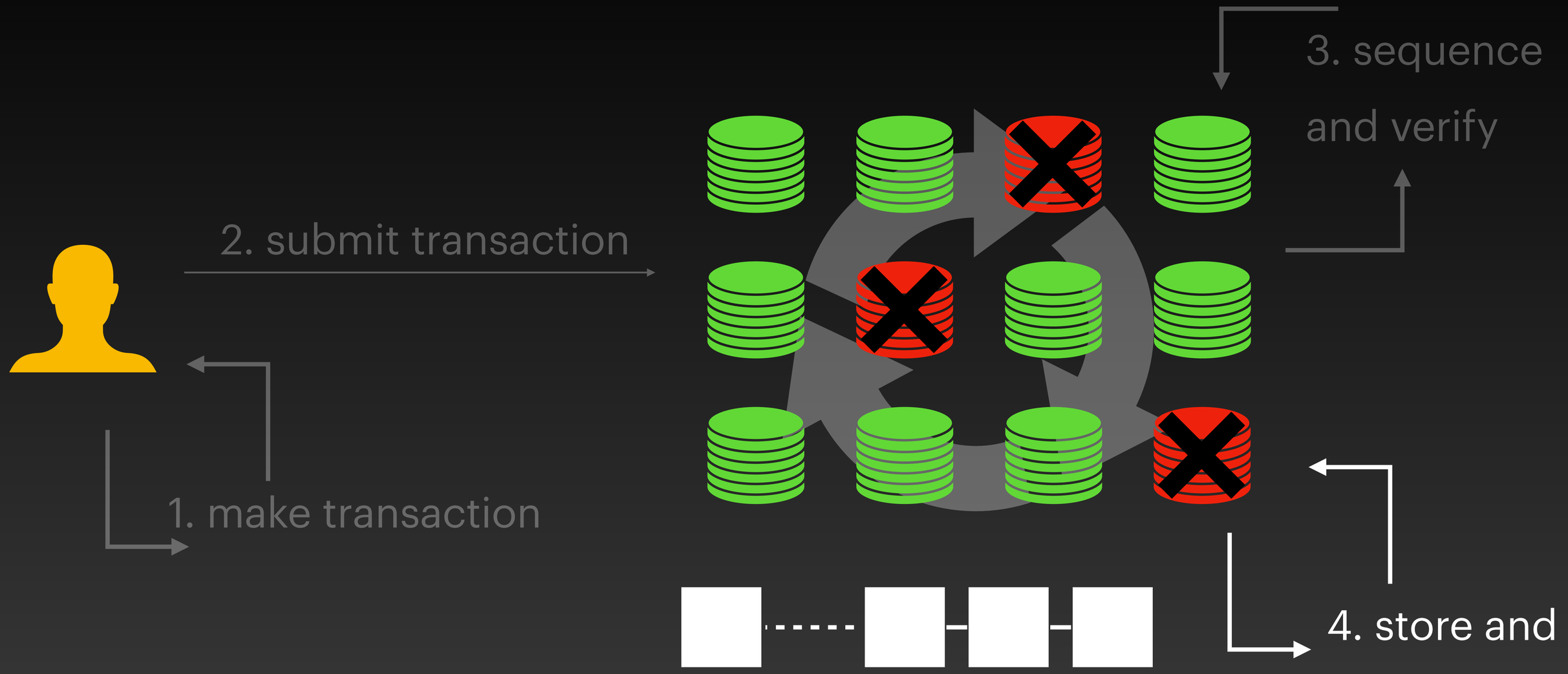


Blockchains

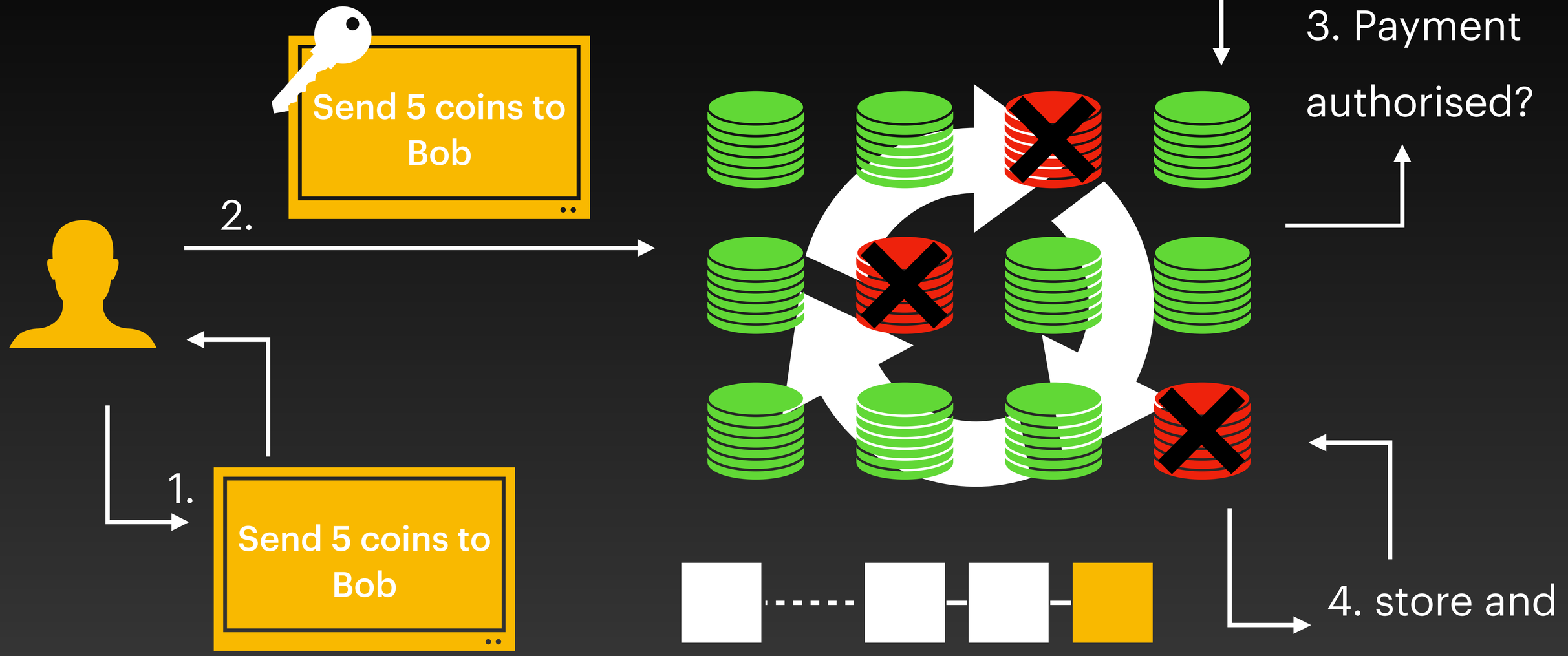


Blockchains

Blockchains



The Typical Example



Security Properties

Safety

Undesirable things never happen

Liveness

Desirable things eventually happen

Adversary

#1 The Network: Worst possible schedule

Properties

- **Synchronous**: A message sent will be delivered before a maximum (known) delay.
- **Asynchronous**: A message sent will eventually be delivered at an arbitrary time before a maximum (unknown) delay.
- **Partial Synchrony**: the network is asynchronous but after some time it enters a period of synchrony.

Challenges

- Theoretical models: Need careful implementation to ensure we approximate them, e.g., retransmissions.
- Memory: Naive implementations use infinite buffers. Identify conditions after which retransmissions are not necessary and buffers can be freed.
- Asynchrony means the protocol should maintain properties for any re-ordering of message deliveries.
- Unknown delay means delay should be adaptive to ensure robustness.

Adversary

#2 Bad Nodes: Arbitrary behaviour

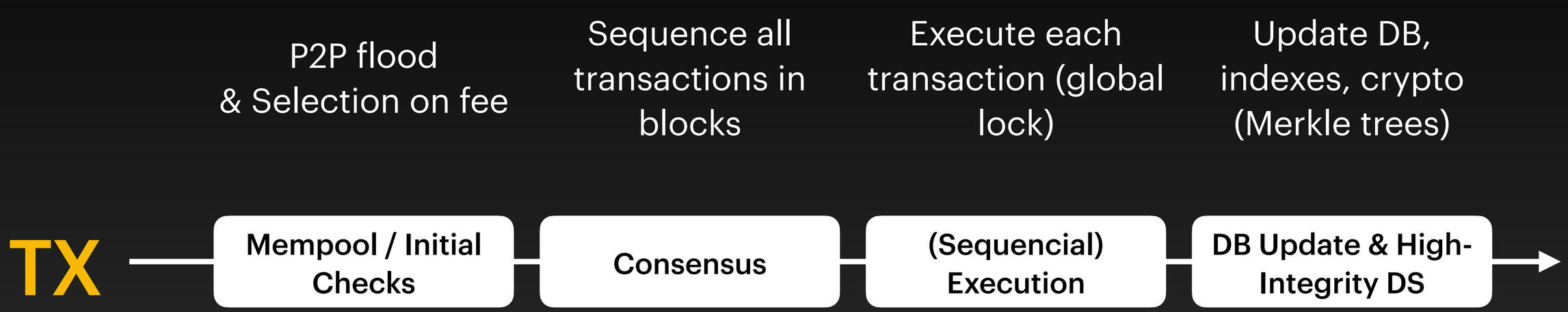
Properties

- **Correct / honest / good:** Will remain live and follow the protocol as specified by the designers of the system.
- **Byzantine:** will deviate arbitrarily from the protocol. May respond incorrectly or not at all.

Challenges

- **Crash & recover:** this is still a correct node with very high latency. Need persistence to ensure this
- **Rational:** honest validators may have some discretion. They may use it to maximise profit

Typical Architecture



Typical Architecture

P2P flood
& Selection on fee

Sequence all transactions in blocks

Execute each transaction (global lock)

Update DB, indexes, crypto (Merkle trees)

Overlay flooding slow and with significant redundancy

Seconds latency, traditionally low throughput

Single core does all computations. (eg EVM ~300 tps)

Added latency of store, blocks, and crypto computations

New Architecture

Consensus is 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

...

New Architecture

Consensus is required

Increment a publicly-accessible counter

Auctions

Market places

Collaborative in-game assets

...

New Architecture

The Sui System

Consensus only when you need to

New Architecture
Architecture

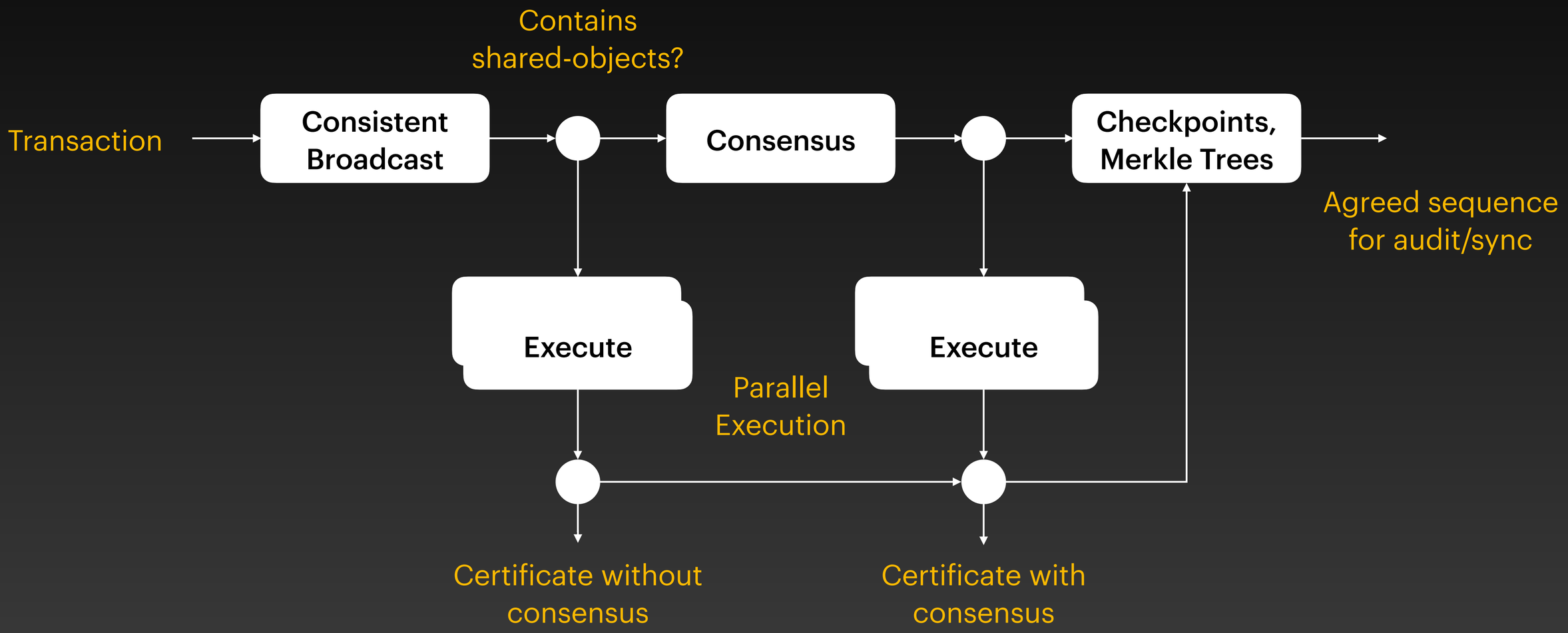
Owned Objects

- 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 my multiple entities
- e.g., A global counter
- **Need consensus**

The Sui System
Architecture



The Sui System
Transactions

Objects:

- Unique ID
- Version number
- Ownership Information
- Type (shared, owned)

The Sui System
Transactions

Objects:

- Unique ID
- Version number
- Ownership Information
- Type (shared, owned)

Transaction's content

Coin::Send

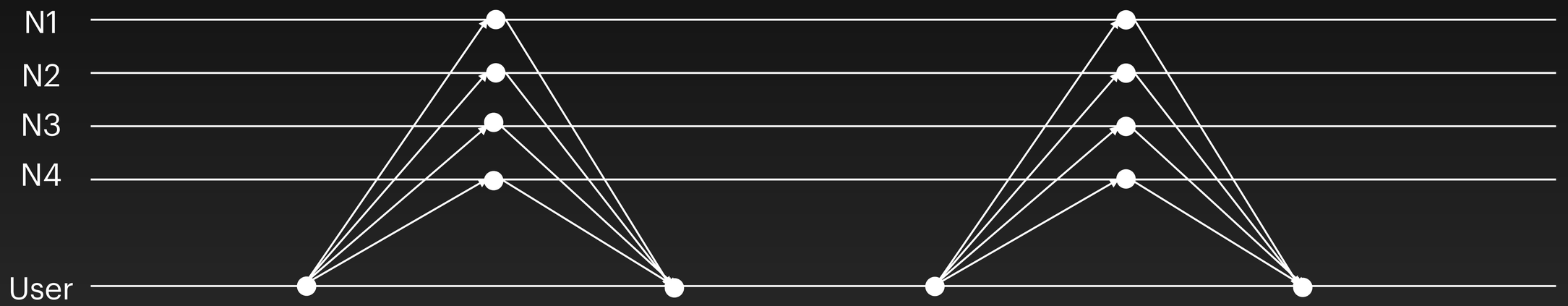
Alice's account

Bob's account, Balance=5

0.001, max=0.005

The Sui System

Consensus-less Path



Send T1:

Disseminate the transaction

Echo T1:

Nodes check and sign T1

Cert T1:

User gather >2/3 signatures into a certificate and disseminate it

Effect T1:

User gather >2/3 effect signatures for finality

The Sui System

Consensus-less Path

Example Transaction

T1

Inputs: O1, O2, O3

Output: Mutate O1, Transfer O2, Delete O3, Create O4

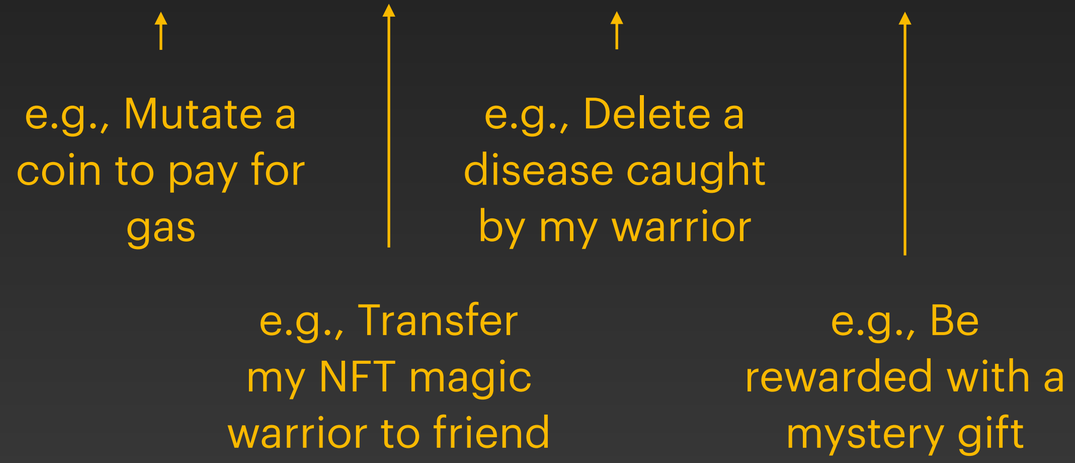
The Sui System
Consensus-less Path

Example Transaction

T1

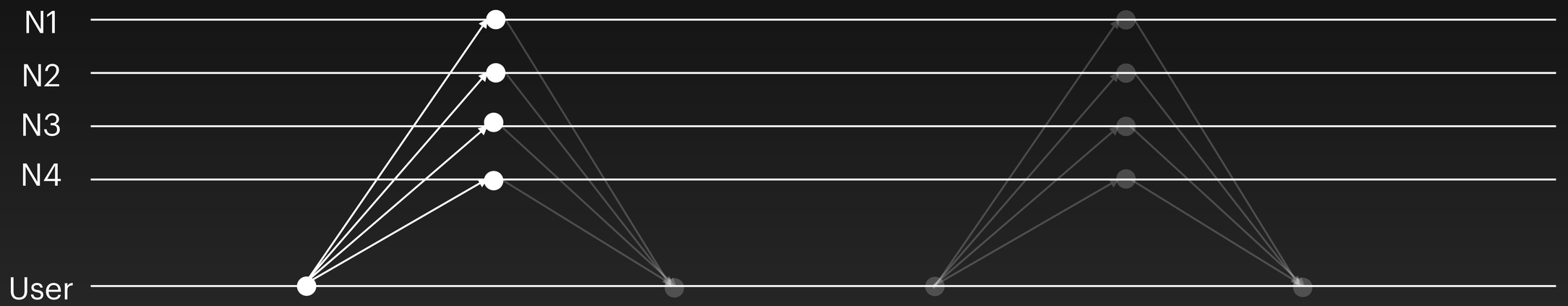
Inputs: O1, O2, O3

Output: Mutate O1, Transfer O2, Delete O3, Create O4



The Sui System

Consensus-less Path



Send T1:

Disseminate the transaction

Echo T1:

Nodes check and sign T1

Cert T1:

User gather >2/3 signatures into a certificate and disseminate it

Effect T1:

User gather >2/3 effect signatures for finality

The Sui System
Consensus-less Path

Step 1: Owned object locks & version exist at validator

O1

L1 = (O1, 10)

Sender=X : None

O2

L2 = (O2, 27)

Sender=X : None

O3

L3 = (O3, 1001)

Sender=X : None

We call these “locks”, and are initialised to None.

The Sui System
Consensus-less Path

Step 2: Validator V checks / signs transactions

O1

L1 = (O1, 10)

Sender=X : ~~None~~ T1

O2

L2 = (O2, 27)

Sender=X : ~~None~~ T1

O3

L3 = (O3, 1001)

Sender=X : ~~None~~ T1

Transaction: T1

Inputs: (O1, 10), (O2, 27), (O3, 1001)

Move call details

Signature of X

Checks T1 (Validity)

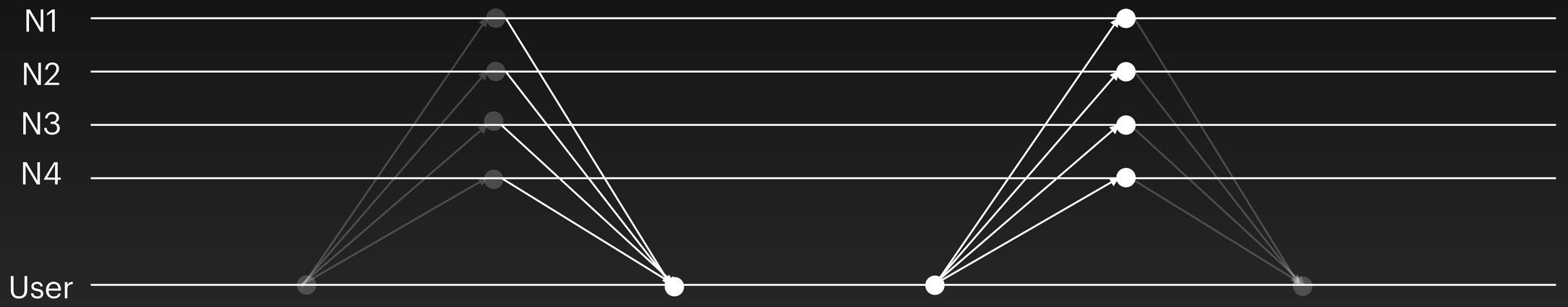
- Well-formed (syntactic)
- Valid Signature from X
- Valid execution function
- Version owned by X

Checks T1 (Broadcast)

- Objects exist and lock is None
- Set lock to T1

The Sui System

Consensus-less Path



Send T1:

Disseminate the transaction

Echo T1:

Nodes check and sign T1

Cert T1:

User gather >2/3 signatures into a certificate and disseminate it

Effect T1:

User gather >2/3 effect signatures for finality

The Sui System
Consensus-less Path

Step 3: Validator V process certificate

O1

L1 = (O1, 10)

Sender=X : ~~None~~ T1

O2

L2 = (O2, 27)

Sender=X : ~~None~~ T1

O3

L3 = (O3, 1001)

Sender=X : ~~None~~ T1

Transaction: T1

Inputs: (O1, 10), (O2, 27), (O3, 1001)

Move call details

Signature of X

Signature (V1, ... V4)

Checks T1 (Validity)

- Again!

Checks T1 (Broadcast)

- Objects exist (with any lock)
- Certificate signed by quorum

The Sui System
Consensus-less Path

Step 4: Validator V executes / signs effect

O1

L1 = (O1, 11)

Sender=X : None

O2

L2 = (O2, 28)

Sender=Y : None

O4

L3 = (O4, 1)

Sender=X : None

Transaction: T1

Inputs: (O1, 10), (O2, 27), (O3, 1001)

Move call details

Signature of X

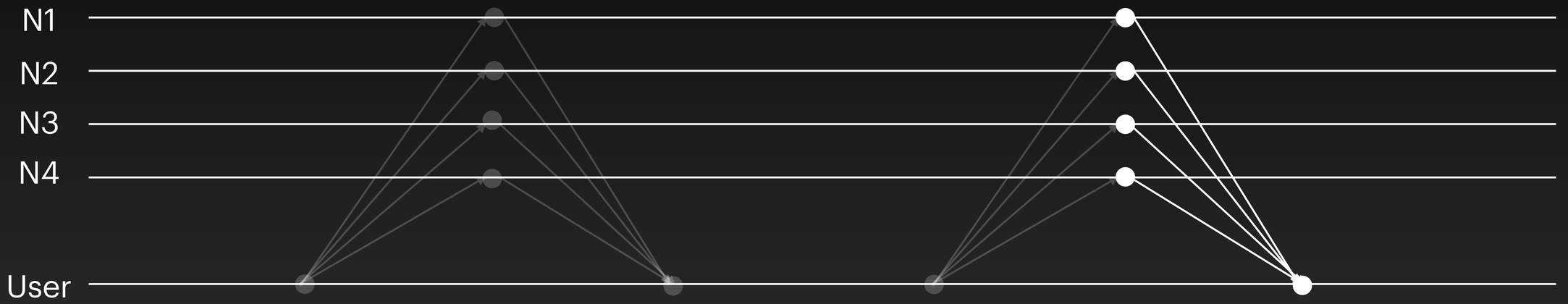
Signature (V1, ... V4)

Execute T1

- O1 mutated
- O2 transferred
- O3 deleted
- O4 created

The Sui System

Consensus-less Path



Send T1:

Disseminate the transaction

Echo T1:

Nodes check and sign T1

Cert T1:

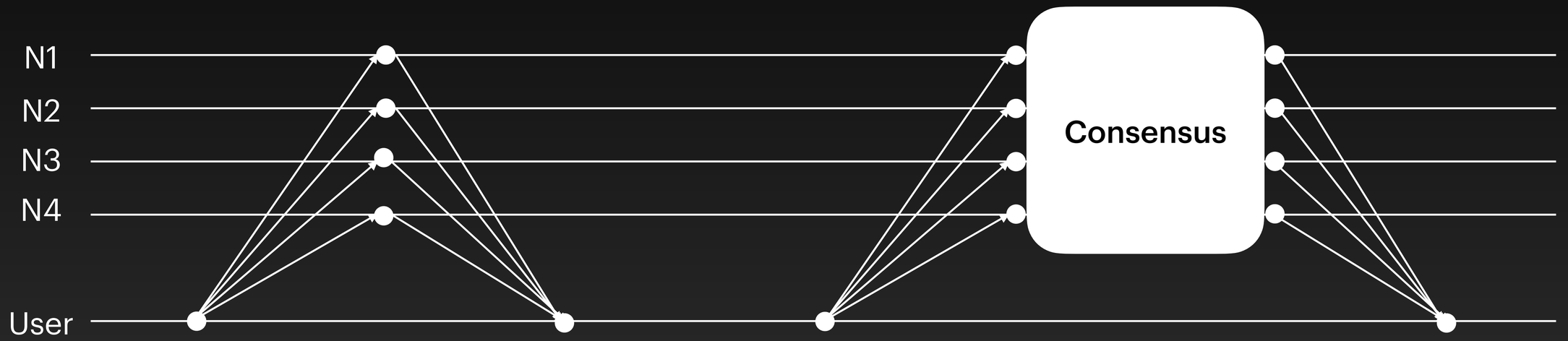
User gather >2/3 signatures into a certificate and disseminate it

Effect T1:

User gather >2/3 effect signatures for finality

The Sui System

Consensus Path



Send T1:

Disseminate the transaction

Echo T1:

Nodes check and sign T1

Cert T1:

User gather >2/3 signatures into a certificate and disseminate it

Effect T1:

User gather >2/3 effect signatures for finality

The Sui System
Consensus Path

Example Transaction

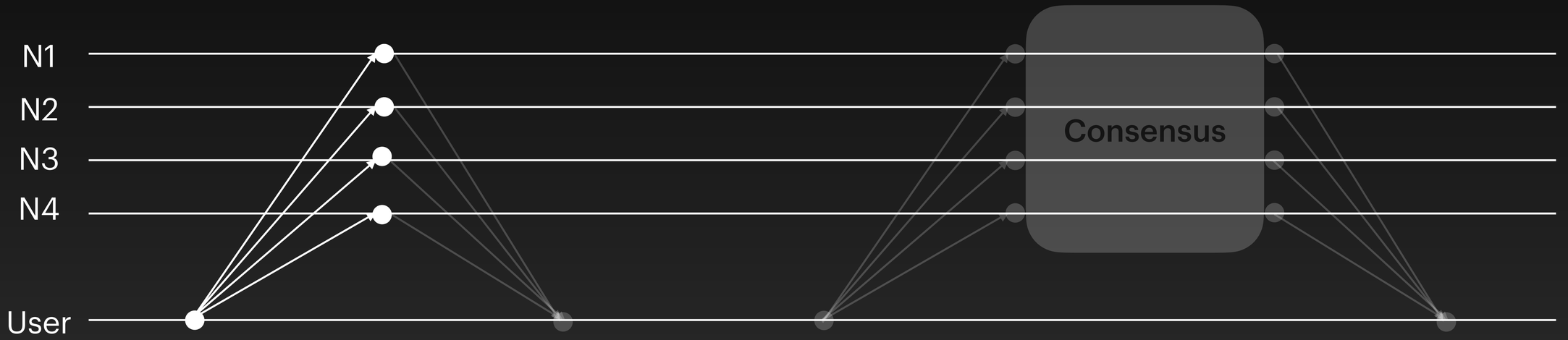
T2

Inputs: O1, S2

Output: Mutate O1, Mutate S2, Create O4

The Sui System

Consensus Path



Send T1:

Disseminate the transaction

Echo T1:

Nodes check and sign T1

Cert T1:

User gather >2/3 signatures into a certificate and disseminate it

Effect T1:

User gather >2/3 effect signatures for finality

The Sui System
Consensus Path

Step 1: Shared object locks exist at validator

O1

L1 = (O1, 10)

Sender=X : None

S2

L2 = (S2, *)

Sender=X

Do not check the version for shared objects

The Sui System
Consensus Path

Step 2: Validator V checks / signs transactions

O1

L1 = (O1, 10)

Sender=X : ~~None~~ T2

S2

L2 = (S2, *)

Sender=X

Transaction: T2

Inputs: (O1, 10), (S2, *)

Move call details

Signature of X

Checks T1 (Validity)

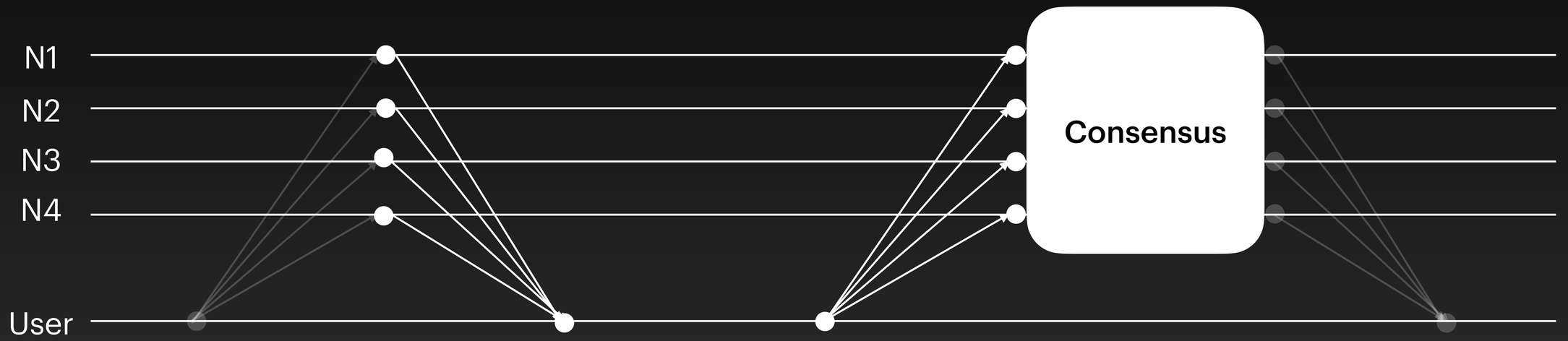
- Well-formed (syntactic)
- Valid Signature from X
- Valid execution function
- Version owned by X

Checks T1 (Broadcast)

- Objects exist and lock is None
- Set lock to T1

The Sui System

Consensus Path



Send T1:

Disseminate the transaction

Echo T1:

Nodes check and sign T1

Cert T1:

User gather >2/3 signatures into a certificate and disseminate it

Effect T1:

User gather >2/3 effect signatures for finality

The Sui System
Consensus Path

Step 3: After consensus, assign shared objects locks

O1

L1 = (O1, 10)

Sender=X : ~~None~~ T2

S2

L2 = (S2, 4)

Sender=X

Transaction: T2

Inputs: (O1, 10), (S2, *)

Move call details

Signature of X

Assign Shared Locks

- Every node sees the same sequence out of consensus
- So they can all assign the same shared object locks

The Sui System
Consensus-less Path

Same as before

Step 3: Validator V process certificate

O1

L1 = (O1, 10)

Sender=X : ~~None~~ T1

S2

L2 = (S2, 4)

Sender=X

Transaction: T2

Inputs: (O1, 10), (S2, *)

Move call details

Signature of X

Checks T1 (Validity)

- Again!

Checks T1 (Broadcast)

- Objects exist (with any lock)
- Certificate signed by quorum

The Sui System
Consensus-less Path

Same as before

Step 4: Validator V Applies / Signs Effect

O1

L1 = (O1, 11)

Sender=X : None

S2

L2 = (S2, 4)

Sender=X

O4

L3 = (O4, 1)

Sender=X : None

Transaction: T2

Inputs: (O1, 10), (S2, *)

Move call details

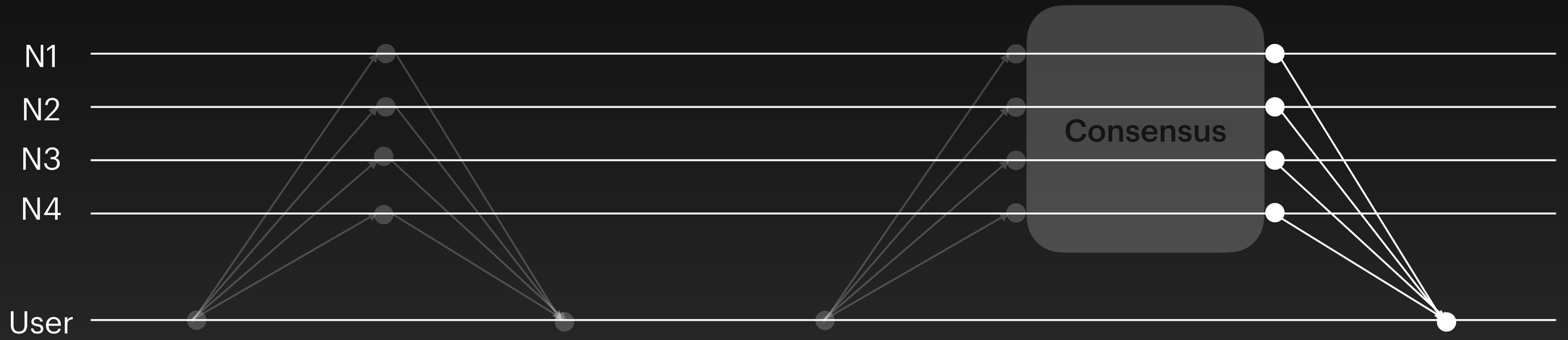
Signature of X

Execute T1

- O1 mutated
- O2 mutated
- O4 created

The Sui System

Consensus Path



Send T1:

Disseminate the transaction

Echo T1:

Nodes check and sign T1

Cert T1:

User gather >2/3 signatures into a certificate and disseminate it

Effect T1:

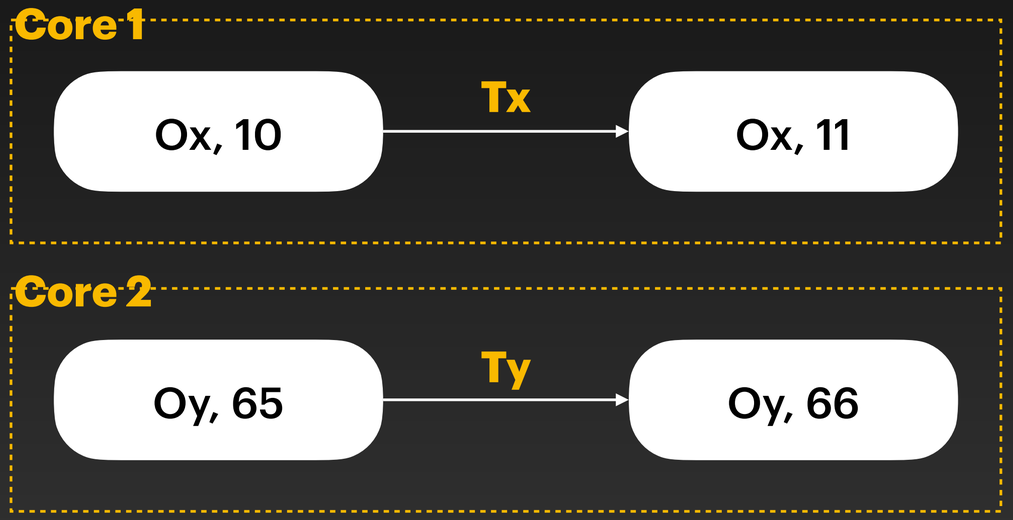
User gather >2/3 effect signatures for finality

The Sui System
Transaction Execution

- First, execute all precedent transactions
- Once there is a certificate, any validator can download Tx and execute

The Sui System
Transaction Execution

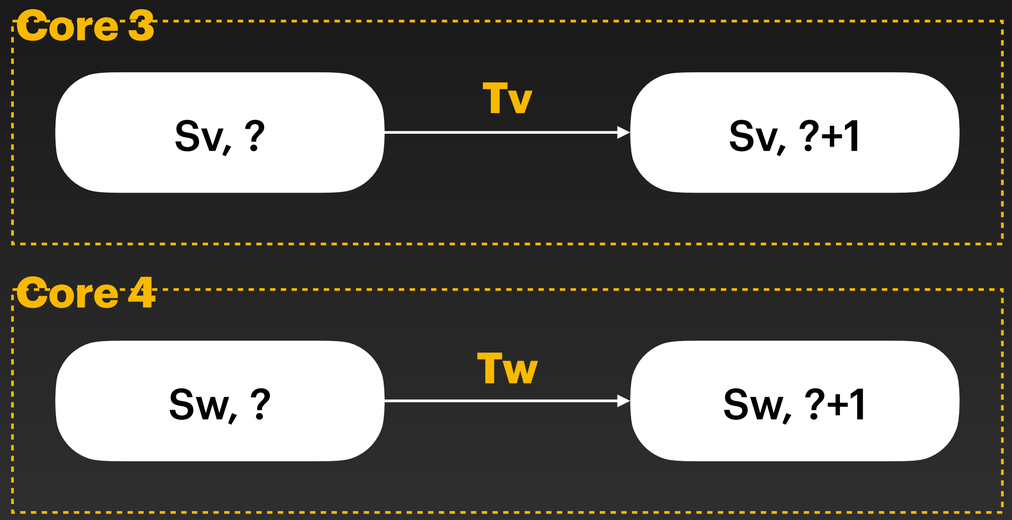
Owned-objects



Always executed in parallel

(once they inputs ID/version are known)

Shared-objects

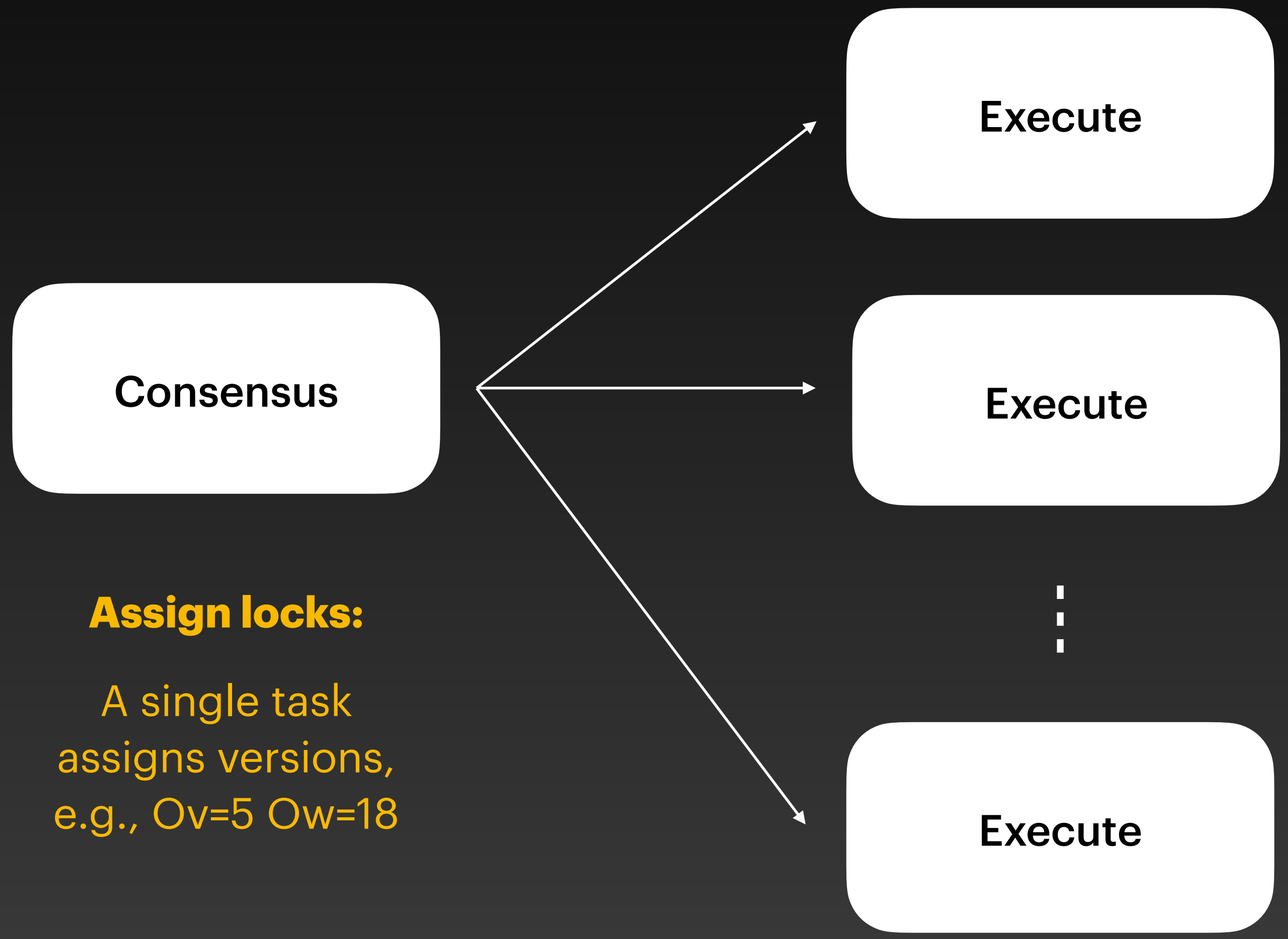


Often executed in parallel

(Sequentially for each shared object)

The Sui System

Shared objects



Assign locks:

A single task assigns versions, e.g., Ov=5 Ow=18

Execute:

Multiple tasks execute (if they can)

The Sui System

Shared objects

Schedule

Single task schedules transactions:

(Tx1, Sv) -> 5

(Tx1, Sw) -> 17

...

(Tx2, Sw) -> 6

Execute

Many tasks try to execute transactions:

(Tx1, Sv) == db[Sv]

db[Sv] += 1

Missing owned-objects dependency?

- Tell the client
- Synchronise
- Retry

Conclusion

The Sui System

- Separate owned and shared objects
- Only use consensus when you need to
- Execute in parallel whenever you can

- **Paper:** https://sui.io
- **Code:** https://github.com/mystenlabs/sui

alberto@mystenlabs.com