Fundamental Properties of Cryptocurrency in Distributed Systems

Lewis Tseng
Boston College

Google search, 01/21/2021

1 Bitcoin equals

32,650.00 United States Dollar

Jan 21, 7:45 PM UTC · Disclaimer

1	Bitcoin
32650.00	United States Dollar

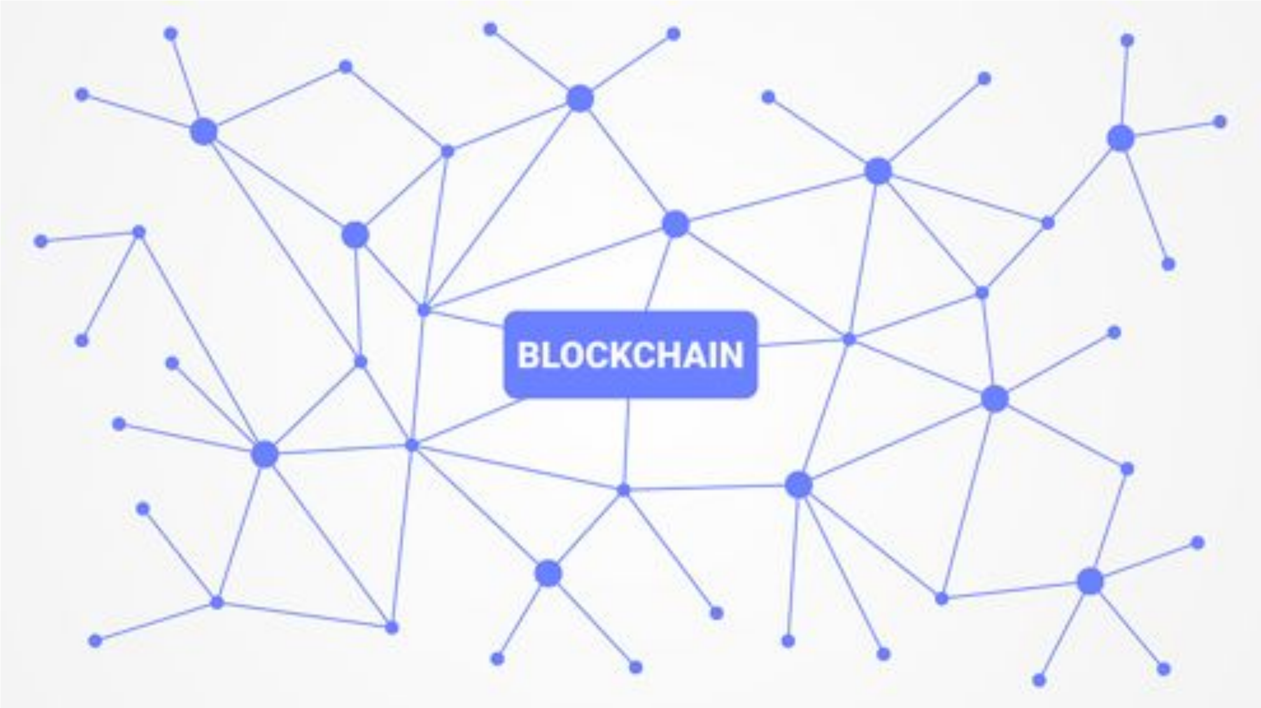
1D 5D 1M 1Y 5Y Max

Data provided by Morningstar for Currency and Coinbase for Cryptocurrency

Decentralized

Fault-tolerant

Secure



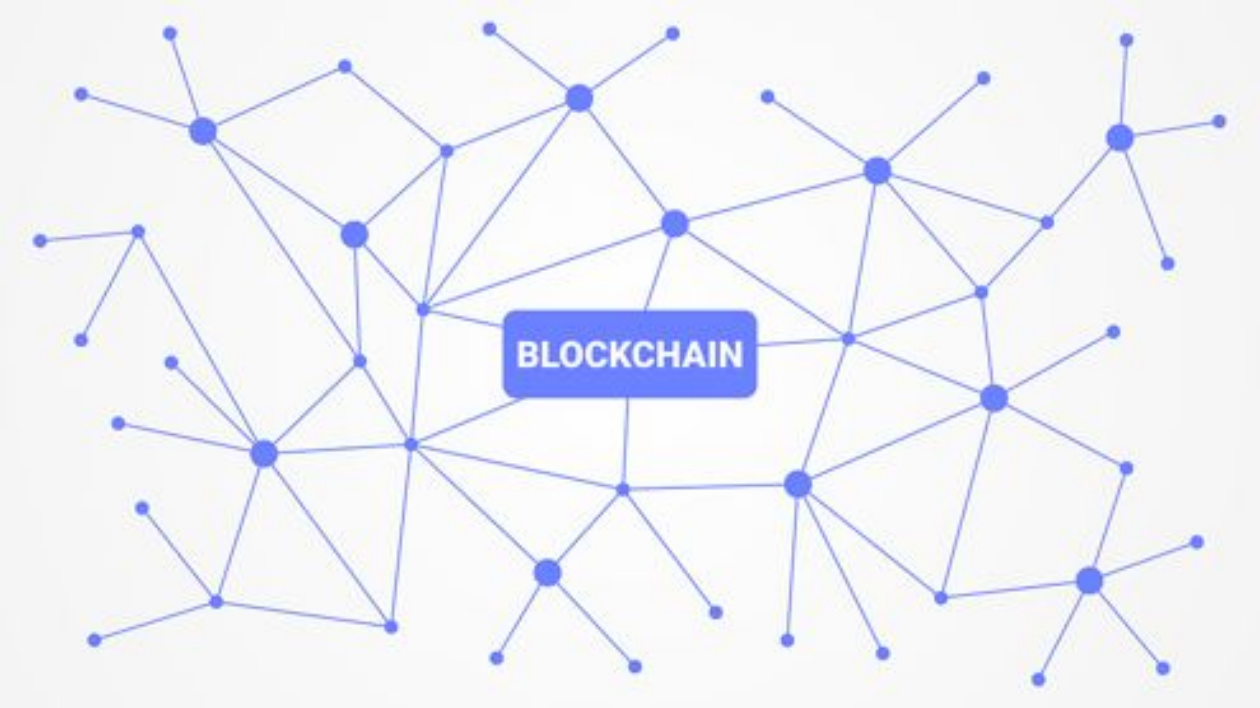
ways of sharing and storing information

Decentralized

Fault-tolerant

Secure

… many other



ways of sharing and storing information

Fundamental Questions

Blockchains and the Future of Distributed Computing

[Herlihy PODC 17]

- No formal abstraction of these objects has been proposed

Formalizing and Implementing Distributed Ledger Objects

[Anta et al. NETYS 18]

- What is the service that must be provided by a distributed ledger?
- What properties a distributed ledger must satisfy?
- What are the assumptions made by the protocols and algorithms on the underlying system?
- Does a distributed ledger require a linked cryptocurrency?

Fundamental Questions

Blockchains and the Future of Distributed Computing

[Herlihy PODC 17]

- No formal abstraction of these objects has been proposed

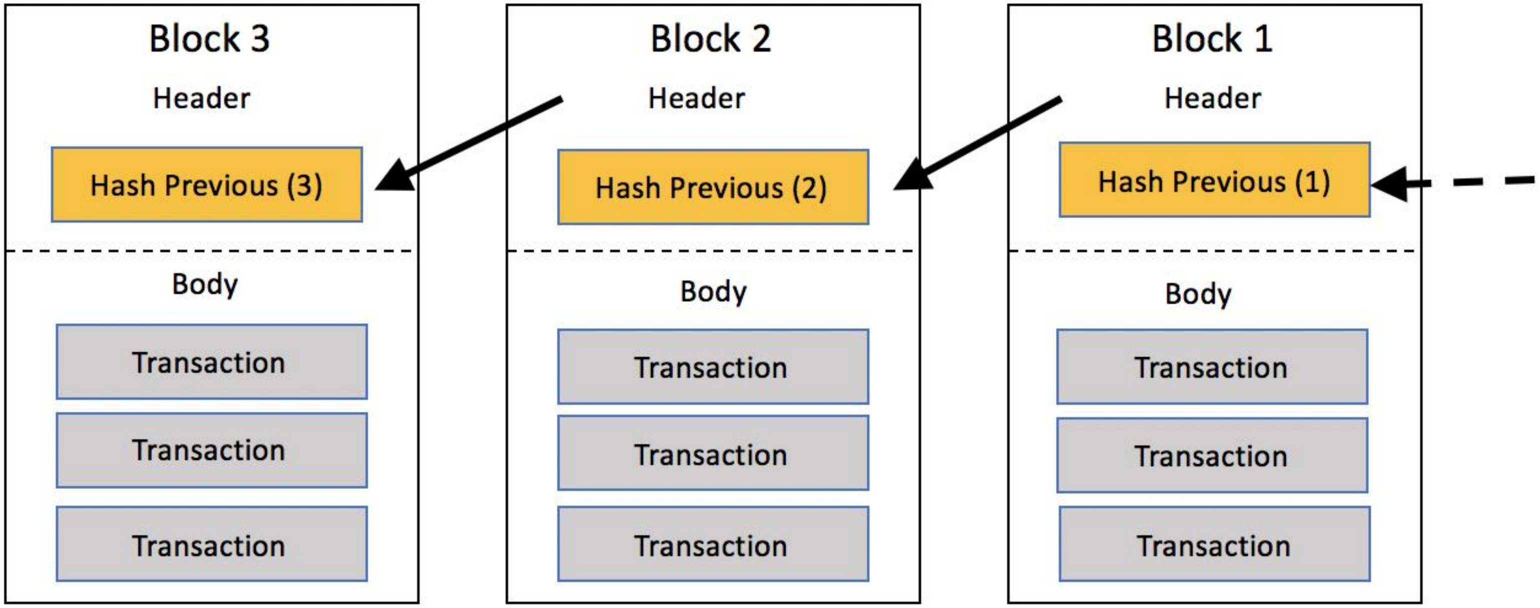
Formalizing and Implementing Distributed Ledger Objects

[Anta et al. NETYS 18]

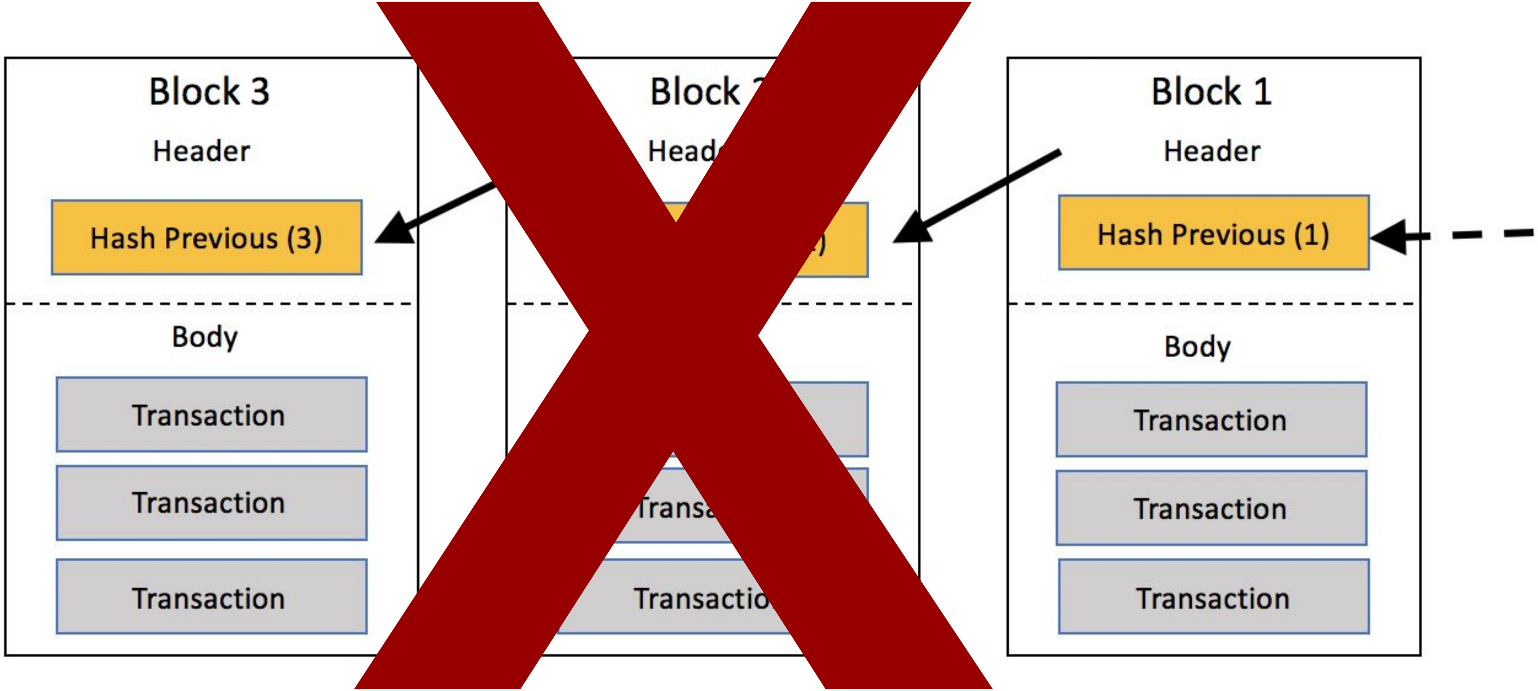
- What is the service that must be provided by a distributed ledger?
- **What properties a distributed ledger must satisfy?**
- What are the assumptions made by the protocols and algorithms on the underlying system?
- Does a distributed ledger require a linked cryptocurrency?

Why Blockchain is Hard?

Blockchain?

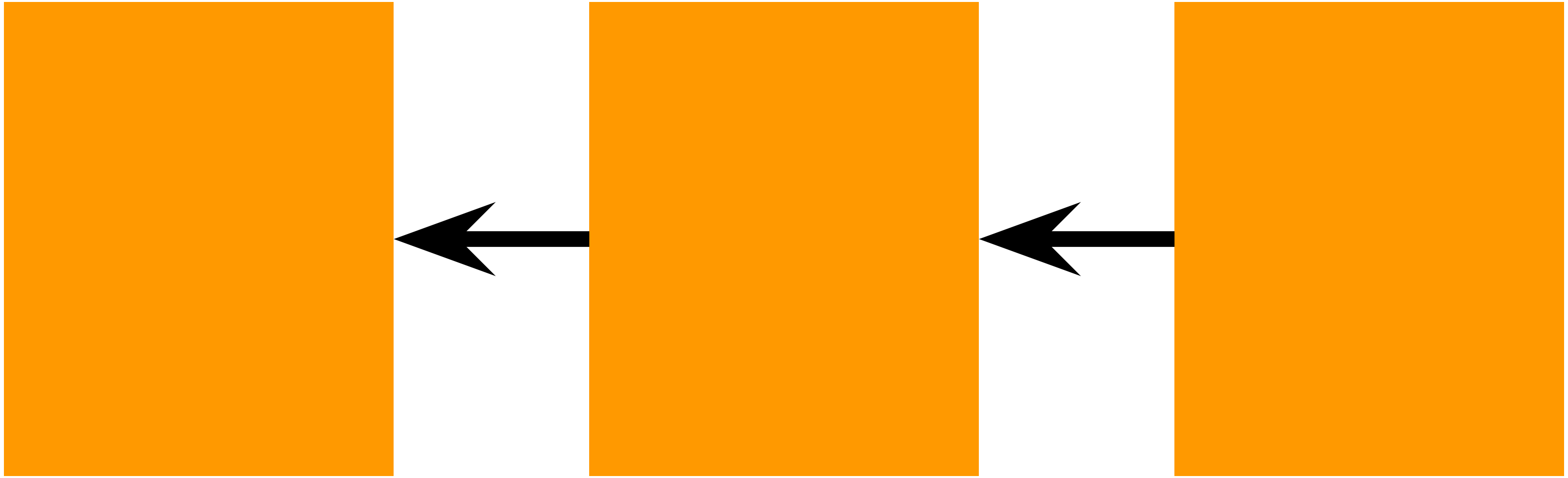

Block 3
Header
Hash Previous (3)
Body
Transaction
Transaction
Transaction
Block 2
Header
Hash Previous (2)
Body
Transaction
Transaction
Transaction
Block 1
Header
Hash Previous (1)
Body
Transaction
Transaction
Transaction

Blockchain?

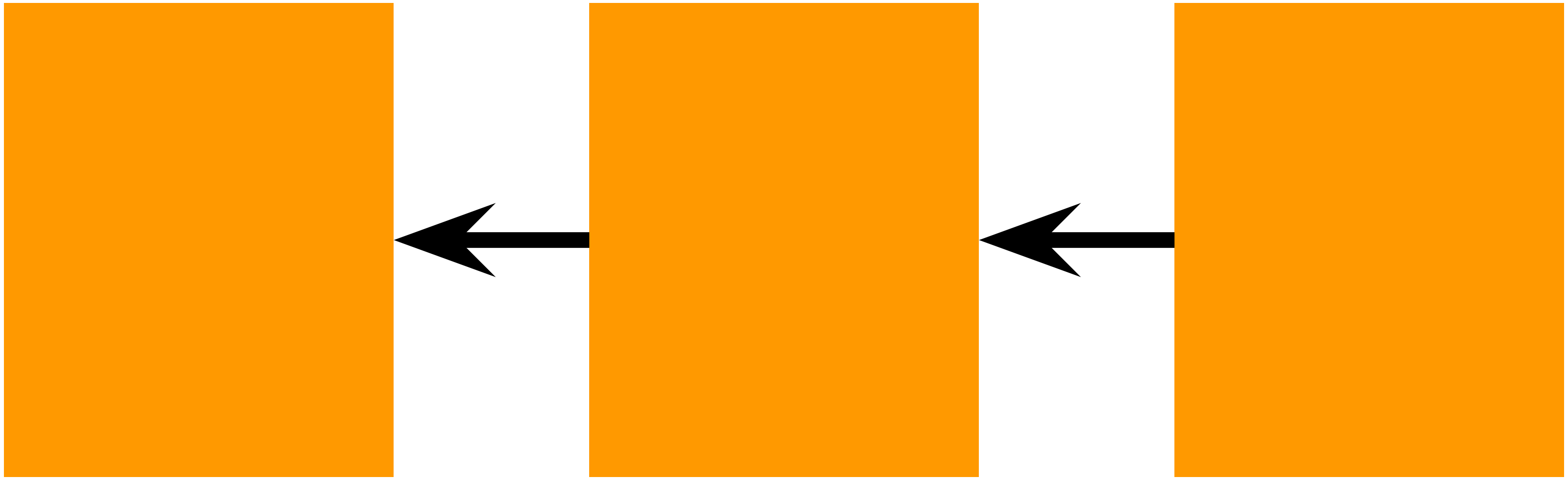


Block 3
Header
Hash Previous (3)
Body
Transaction
Transaction
Transaction
Block 1
Header
Hash Previous (1)
Body
Transaction
Transaction
Transaction

Blockchain? Block + Chain!



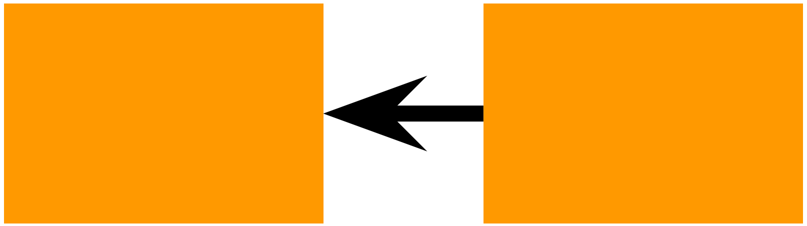
Blockchain? Block + Chain!



In this talk, no crypto detail

- What Block?
- How to link?
- How secure?
- How to mine?

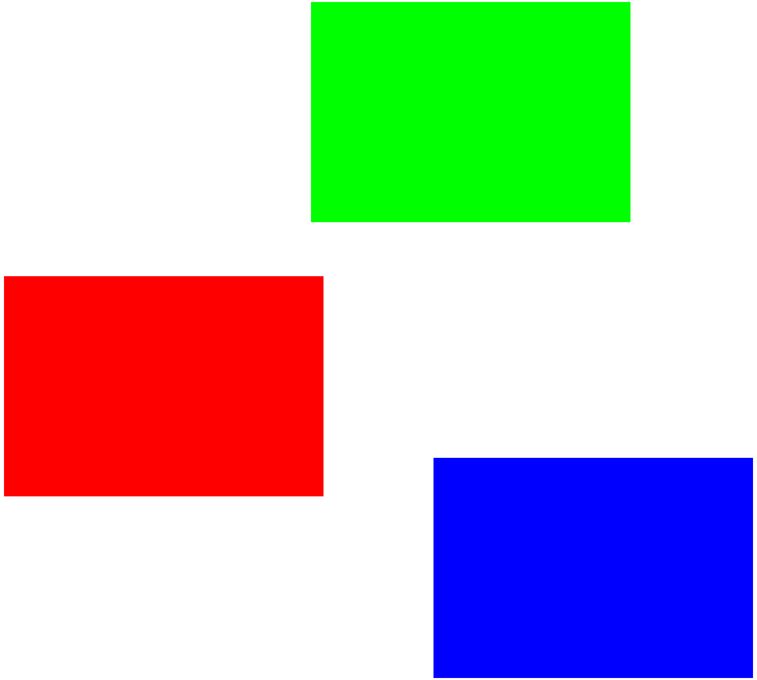
Block + Chain! [in Distributed Computing]



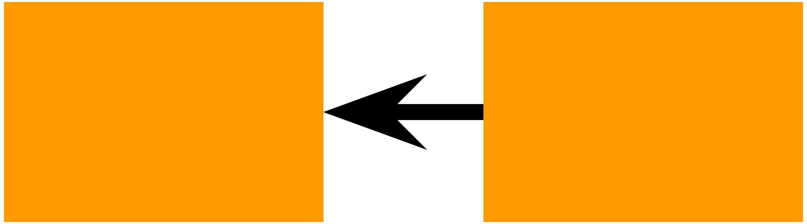
Bitcoin on a high-level:
In each round,

- Nodes exchange blocks (mining)
- Nodes “agree on” a block

Block + Chain!



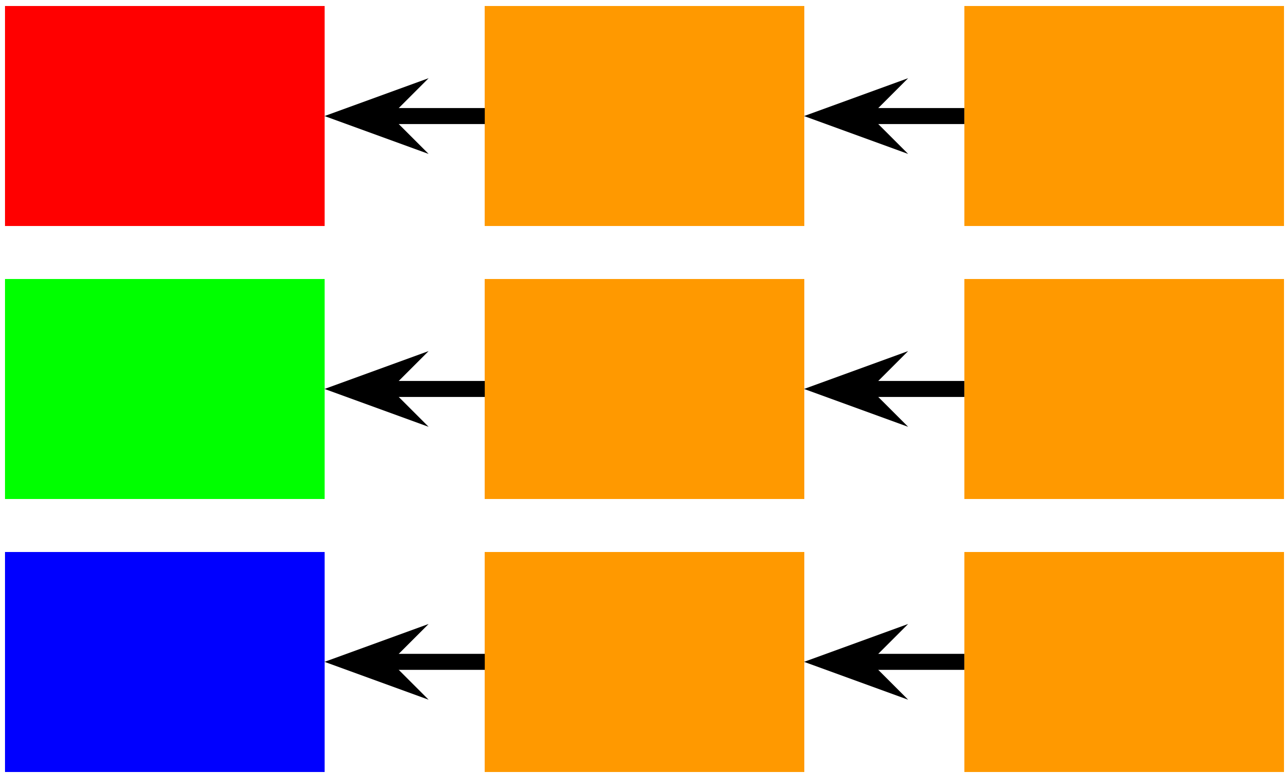
[in Distributed Computing]



Bitcoin on a high-level:
In each round,

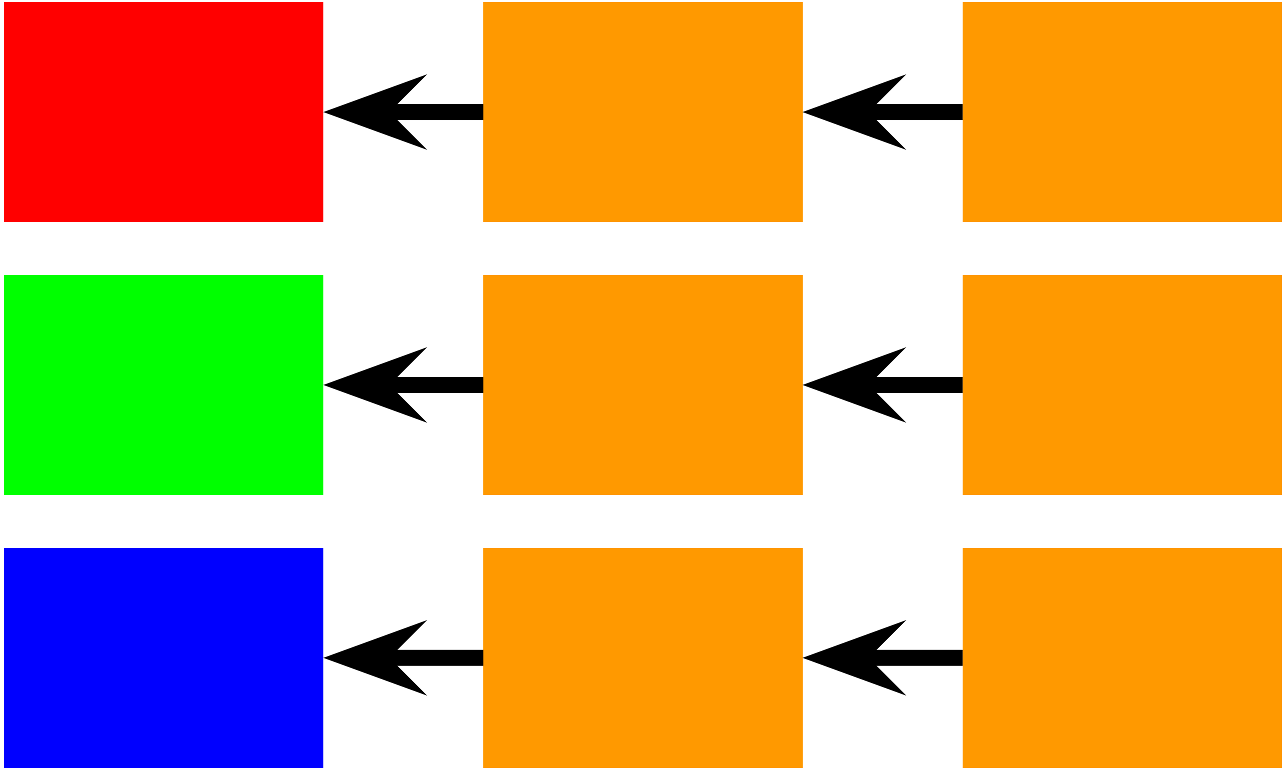
- Nodes exchange blocks (mining)
- Nodes “agree on” a block

Block + Chain! [in Distributed Computing]



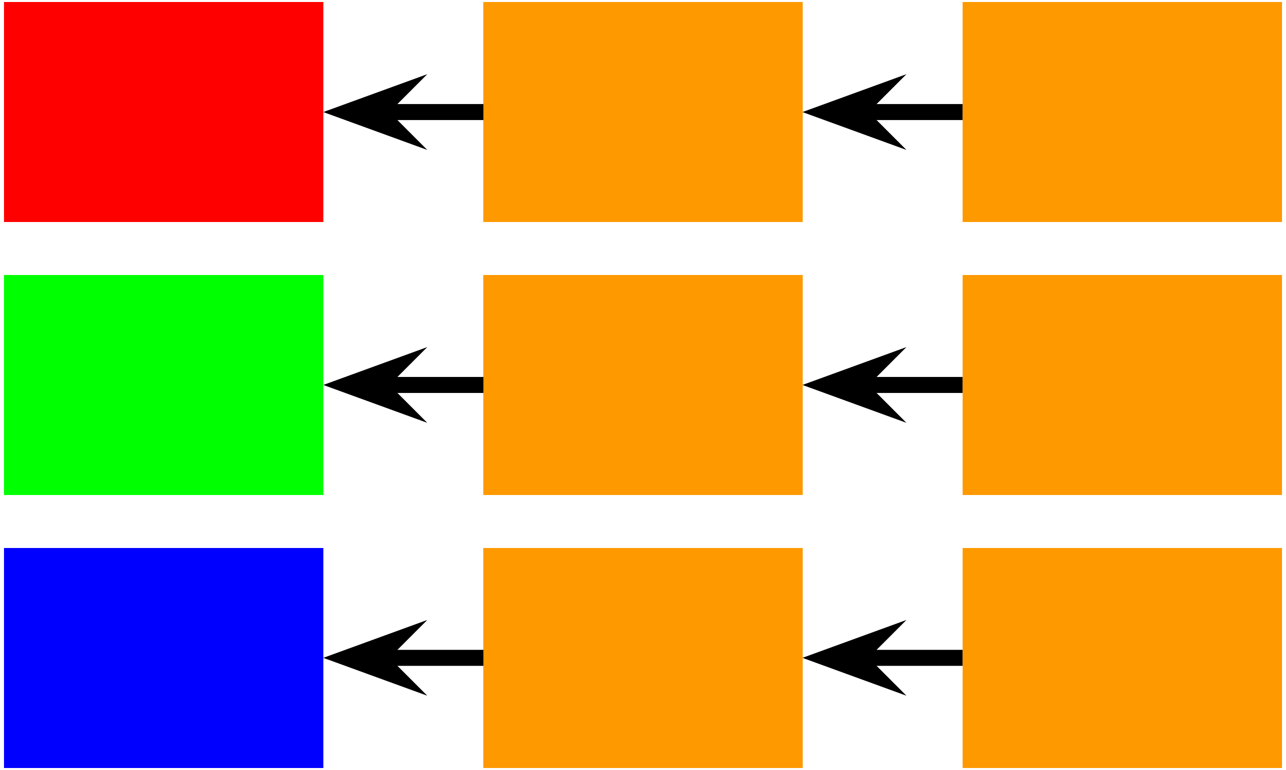
Block + Chain! [in Distributed Computing]

disagreement ⇒ double-spending attack



Block + Chain! [in Distributed Computing]

disagreement ⇒ double-spending attack



FLP Result [JACM 85]: Fault + Async. + Consensus = Impossible

Most common approach:
Proof-of-XXX

Is Consensus necessary?

Is Consensus necessary?
No

The Consensus Number of a Cryptocurrency

Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Dragos-Adrian Seredinschi [PODC 2019]

https://arxiv.org/pdf/1906.05574.pdf

Model

Asynchronous network: arbitrary message delay

Permissioned, static system: a fixed set of nodes **[1, …, n]**

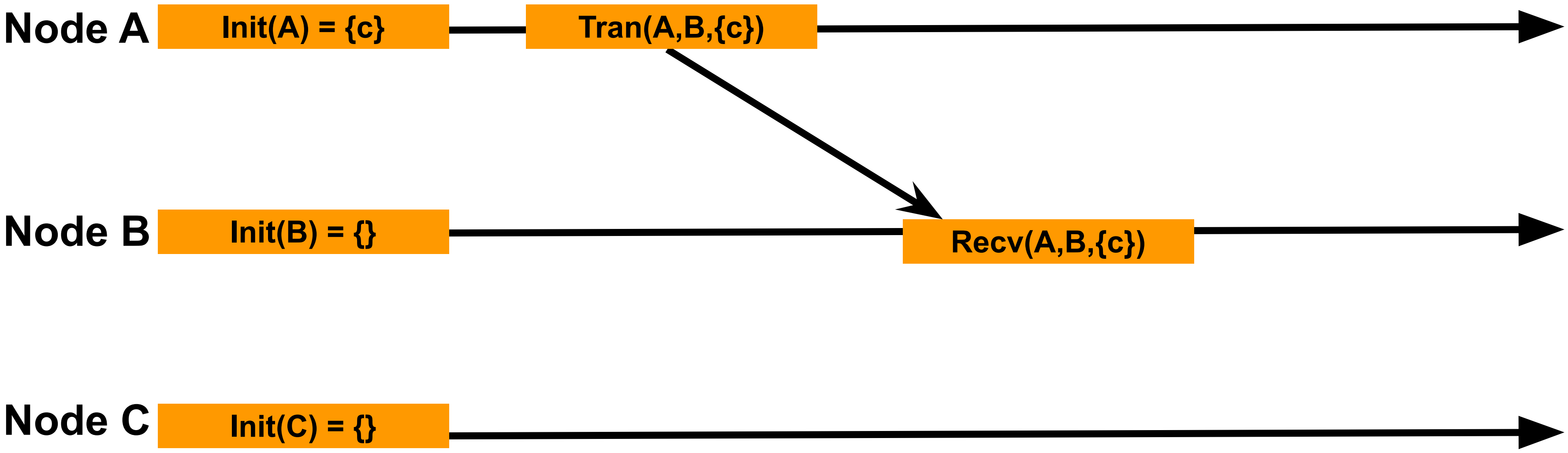
Crash fault: up to **f** fail-stop failures

Cryptocurrency: Working Definition

A cryptocurrency is a virtual asset that relies on cryptography tools to prevent counterfeit or double-spend.

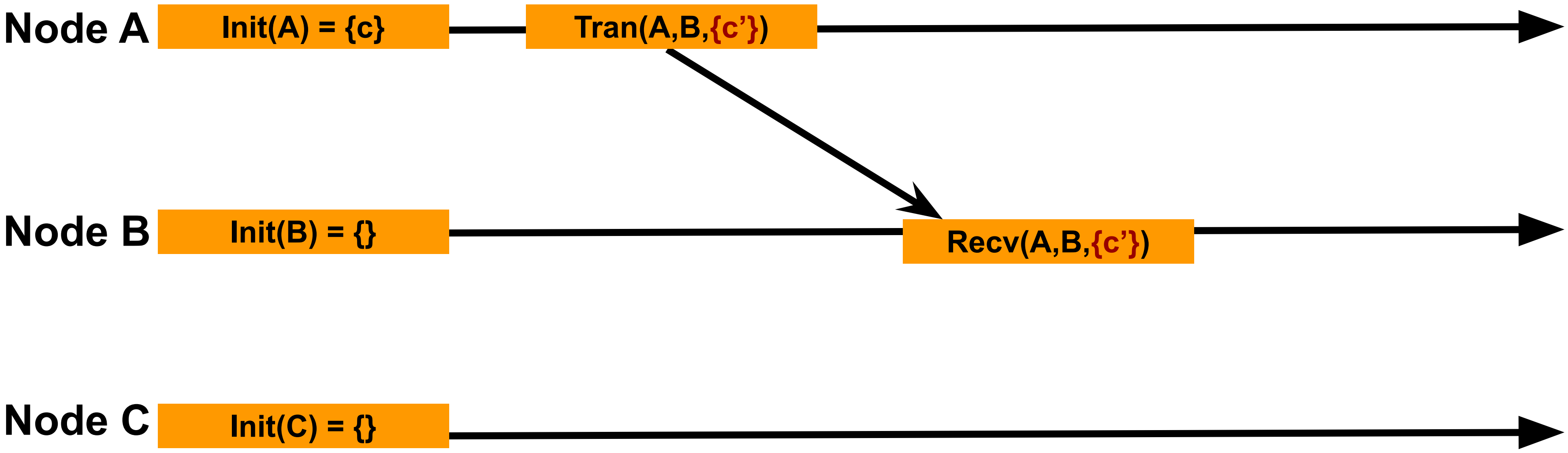
Cryptocurrency: Abstraction

A cryptocurrency is a virtual asset that relies on cryptography tools to prevent counterfeit or double-spend.



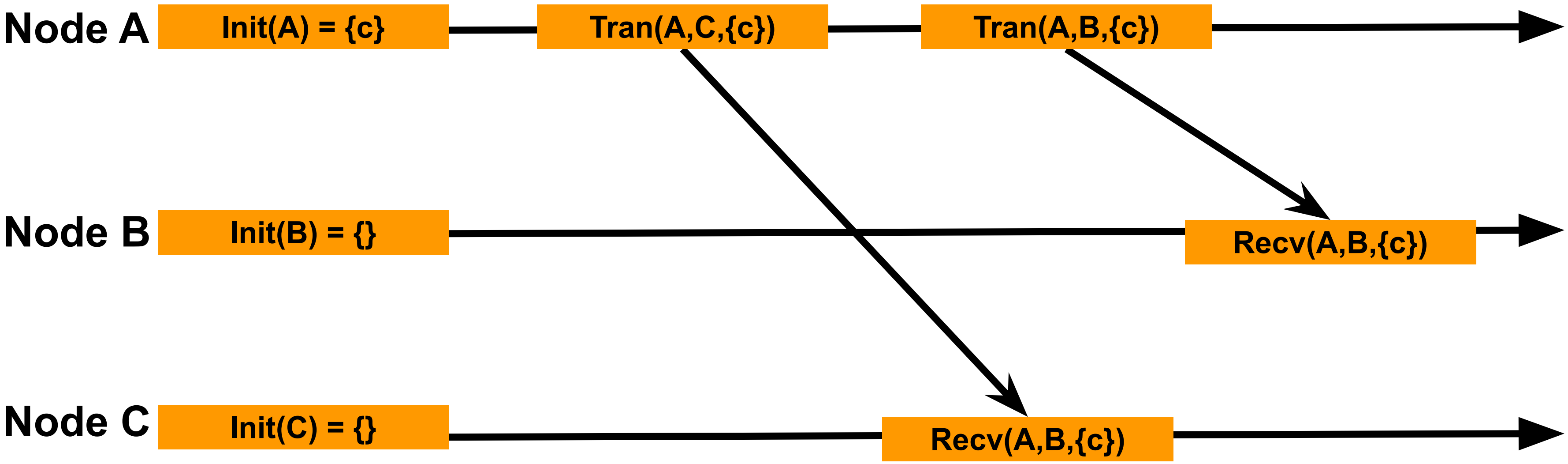
Cryptocurrency: Counterfeit

A cryptocurrency is a virtual asset that relies on cryptography tools to prevent **counterfeit** or double-spend.



Cryptocurrency: Double-spend

A cryptocurrency is a virtual asset that relies on cryptography tools to prevent counterfeit or **double-spend**.



What is the (concurrent) data structure?

Asset transfer object [PODC 2019]

- Each node has an account
- ATO state: balance of each account
- Transfer(A,B,x):
    - Decrease A's account by x
    - Increase B's account by x
- Read(A): return A's balance

What is the (concurrent) data structure?

Asset transfer object [PODC 2019]

- Each node has an account
- ATO state: balance of each account
- Transfer(A,B,x):
  - Decrease A's account by x
  - Increase B's account by x
- Read(A): return A's balance

Valid transfer/transaction: Transfer(A,B,x)

- Invoked by A
- balance(A) $\geq$ x

What is the (concurrent) data structure?

Asset transfer object [PODC 2019]

- Each node has an account
- ATO state: balance of each account
- Transfer(A,B,x):
  - Decrease A's account by x
  - Increase B's account by x
- Read(A): return A's balance

Valid transfer/transaction: Transfer(A,B,x)

- Invoked by A
- balance(A) $\geq$ x **No overdraft**

Atomic Snapshot Object [Afek et al. JACM 93]

Object partitioned into n segments

Each segment is “owned” by a node (single-writer)

Update: write a value to own segment

Scan: read values from all segments -- take a snapshot

Operations **linearizable** -- a total order that follows the real-time order

Atomic Snapshot Object [Afek et al. JACM 93]

Object partitioned into n segments

Each segment is “owned” by a node (single-writer)

Update: write a value to own segment

Scan: read values from all segments -- take a snapshot

Operations **linearizable** -- a total order that follows the real-time order

ID	**1**	**2**	**3**
Value			

ASO-based Cryptocurrency [PODC 2019]

Intuition:

- i's entry = <u>all outgoing transfers</u> at node i

ID	**1**	**2**	**3**
Value	{(1, 2, 100)}		

ASO-based Cryptocurrency [PODC 2019]

Read(A):

- S ← AS.scan
- Return A’s balance in S

Intuition:

- i’s entry = <u>all outgoing transfers</u> at node i

ID	1	2	3
Value	{(1, 2, 100)}		

ASO-based Cryptocurrency [PODC 2019]

Read(A):

- $S \leftarrow$ AS.scan
- Return A's balance in S

Intuition:

- i's entry = all outgoing transfers at node i

	Balance
1	0
2	100
3	0

ID	**1**	**2**	**3**
Value	{(1, 2, 100)}		

ASO-based Cryptocurrency [PODC 2019]

Read(A):

- S ← AS.scan
- Return A's balance in S

Intuition:

- i's entry = <u>all outgoing transfers</u> at node i

	Balance
1	0
2	50
3	50

ID	**1**	**2**	**3**
Value	{(1, 2, 100)}	{(2, 3, 50)}	

ASO-based Cryptocurrency [PODC 2019]

Read(A):

- $S \leftarrow$ AS.scan
- Return A's balance in S

	Balance
1	0
2	50
3	50

Transfer(A,B,x):

- $S \leftarrow$ AS.scan
- If valid transaction:

  $OP[A] \leftarrow OP[A] \cup \{(A,B,x)\}$

  AS.update(OP[A])

ID	**1**	**2**	**3**
Value	{(1, 2, 100)}	{(2, 3, 50)}	

ASO-based Cryptocurrency [PODC 2019]

Read(A):

- S ← AS.scan
- Return A's balance in S

	Balance
1	0
2	50
3	50

Transfer(A,B,x):

learn new incoming tx's

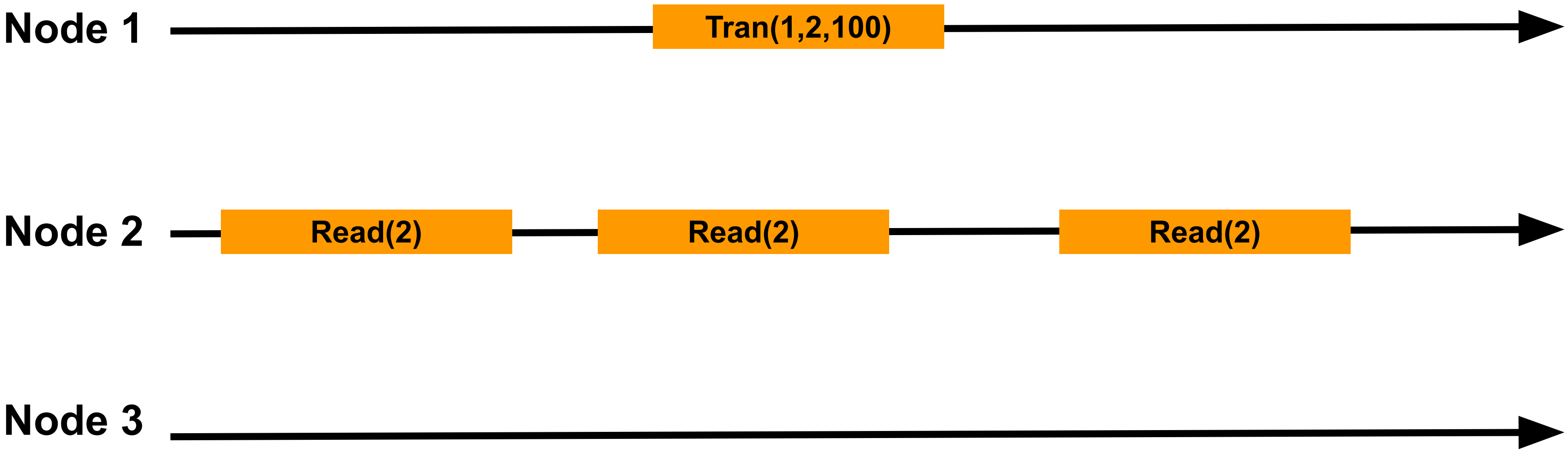
- S ← AS.scan
- If valid transaction:

    OP[A] ← OP[A] U {(A,B,x)}

    AS.update(OP[A])

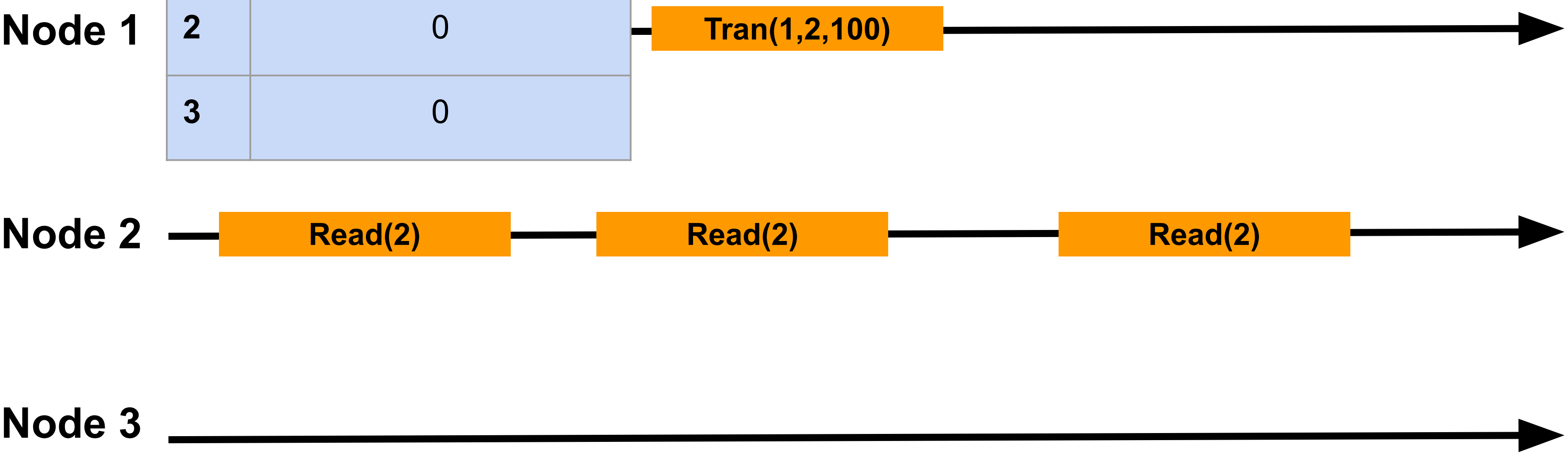
ID	**1**	**2**	**3**
Value	{(1, 2, 100)}	{(2, 3, 50)}	

Example

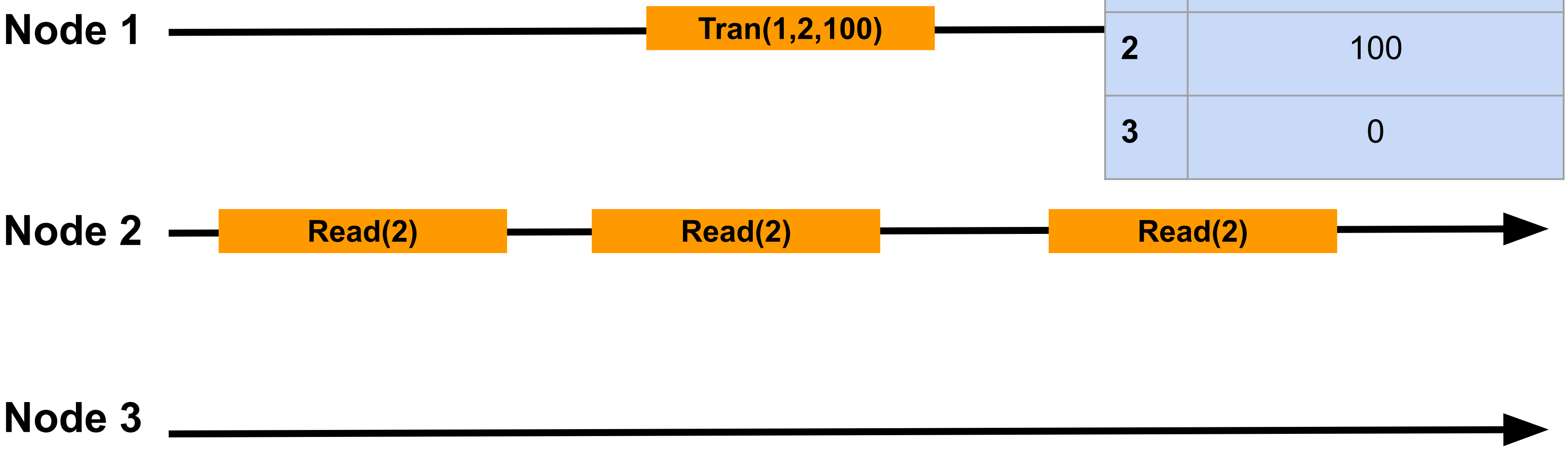


Example

	Balance
1	100
2	0
3	0



Example



	Balance
1	0
2	100
3	0

ASO-based Cryptocurrency [PODC 2019]

Read(A):

- $S \leftarrow$ AS.scan
- Return A's balance in S

	Balance
1	0
2	50
3	50

Transfer(A,B,x):

- $S \leftarrow$ AS.scan
- If valid transaction:

  $OP[A] \leftarrow OP[A] \cup \{(A,B,x)\}$

  AS.update(OP[A])

no overdraft

ID	**1**	**2**	**3**
Value	{(1, 2, 100)}	{(2, 3, 50)}	

ASO-based Cryptocurrency [PODC 2019]

Read(A):

- $S \leftarrow$ AS.scan
- Return A's balance in S

	Balance
1	0
2	50
3	50

Transfer(A,B,x):

- $S \leftarrow$ AS.scan
- If valid transaction:

  $OP[A] \leftarrow OP[A] \cup \{(A,B,x)\}$

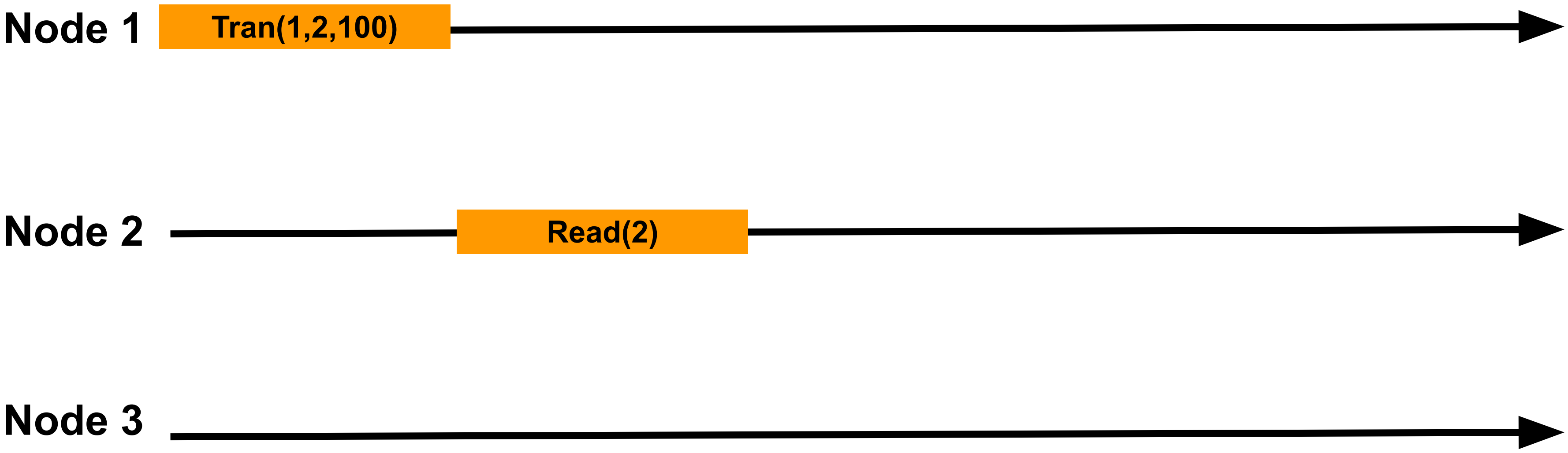
  AS.update(OP[A])

Update A's outgoing tx's

ID	**1**	**2**	**3**
Value	{(1, 2, 100)}	{(2, 3, 50)}	

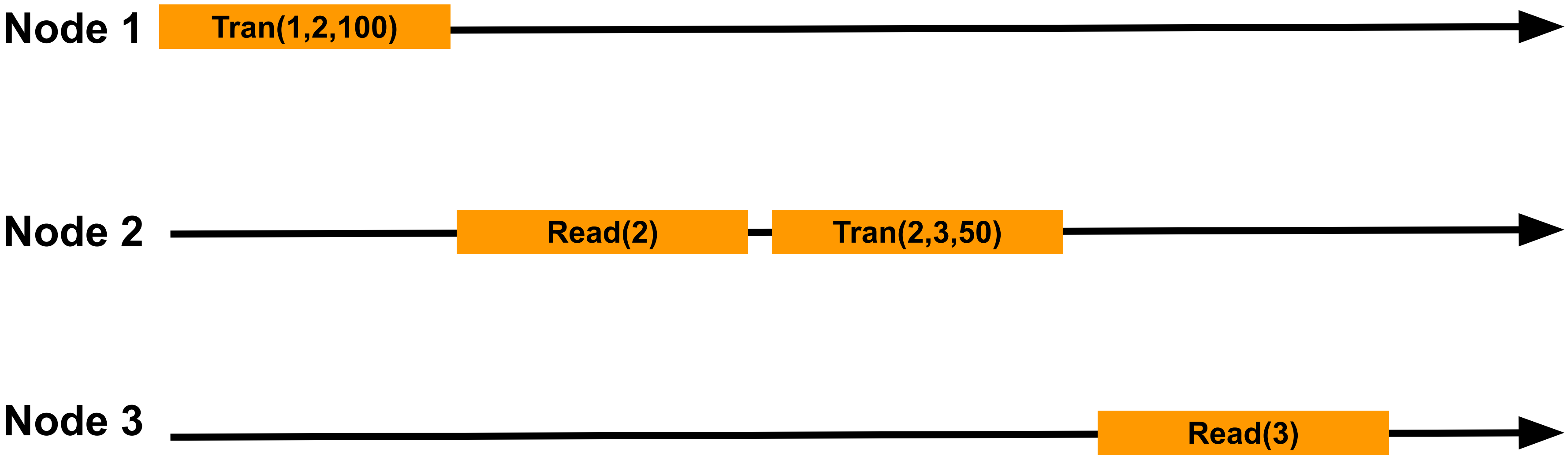
Example

	Balance
1	0
2	100
3	0



Example

	Balance
1	0
2	50
3	50



Is this possible?

Node 1 Tran(1,2,100)

	Balance
1	100
2	0
3	0

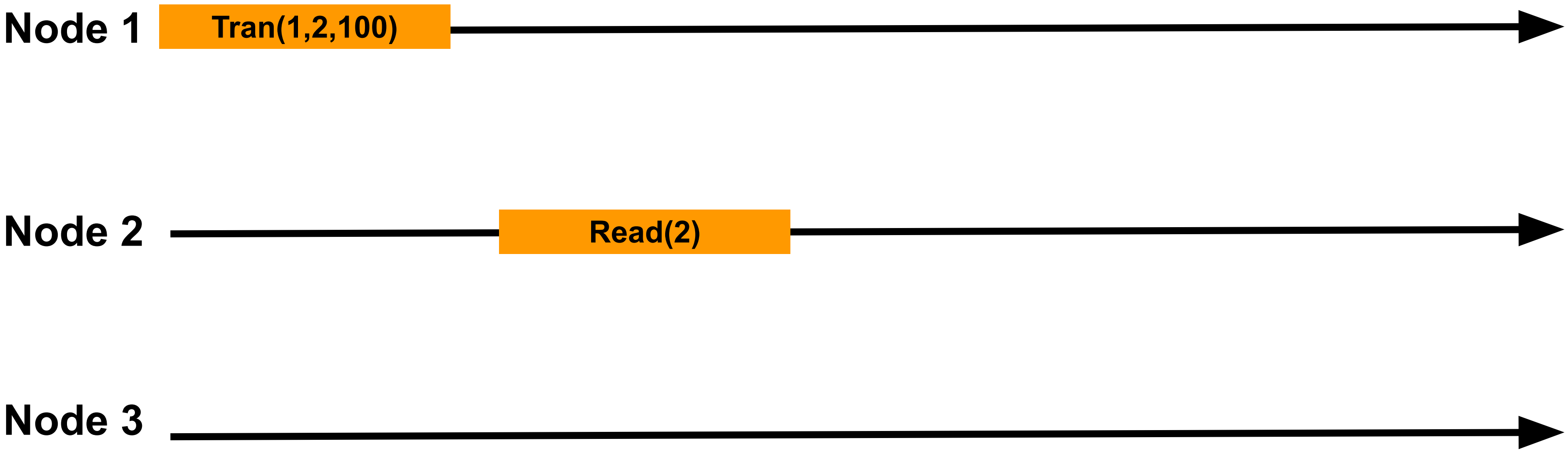
Node 2 Read(2)

Node 3 Read(2)

	Balance
1	0
2	100
3	0

Linearizability:
Total order + **Real-time order**

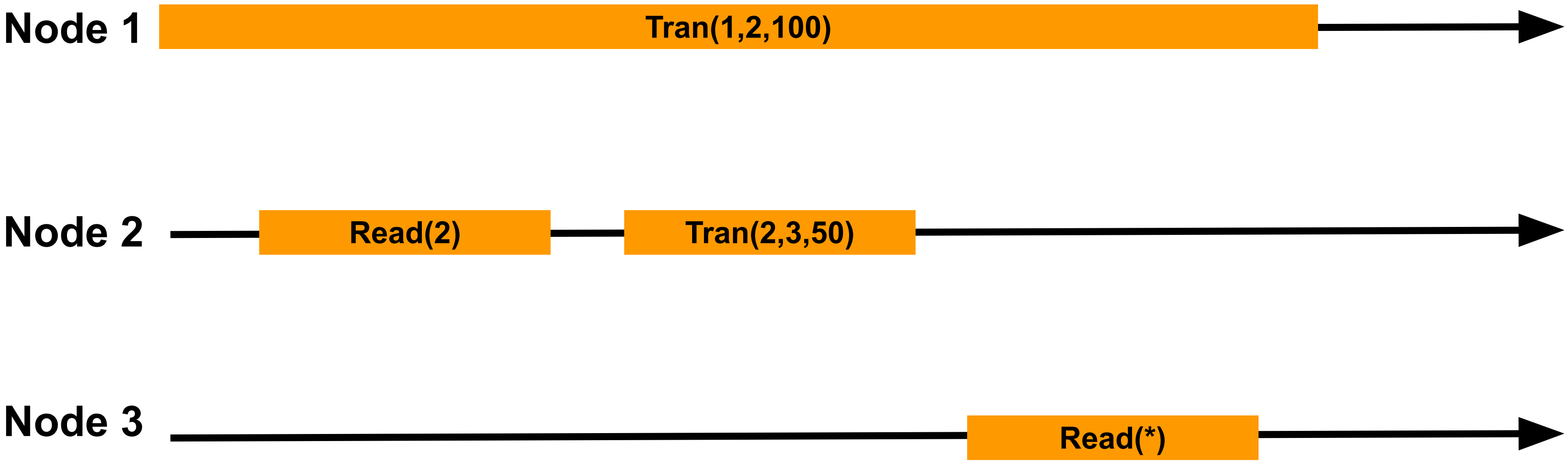
	Balance
1	0
2	100
3	0



Linearizability:

Total order + Real-time order

	Balance
1	0
2	50
3	50



So what?

ASO can be implemented in asynchronous systems!

It's all good, but ...

Is it scalable and highly available?

CAP Theorem [Brewer PODC 00, Gilbert/Lynch 02]

Consistency: right response to each request

Availability: termination eventually

Partition tolerance:

unreliable comm. network

CAP Theorem [Brewer PODC 00, Gilbert/Lynch 02]

Consistency: right

Availability: termi

Partition tolerance:

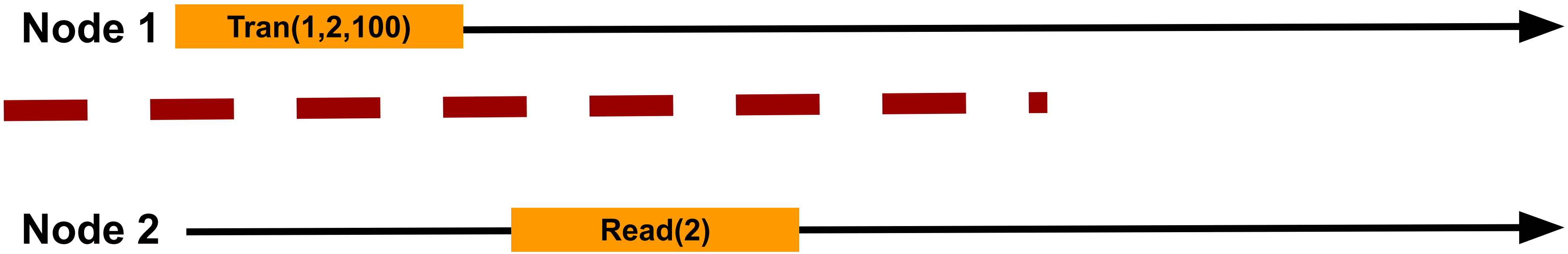
unrel

Impossible to have all three!

When there is a partition, choose consistency or availability

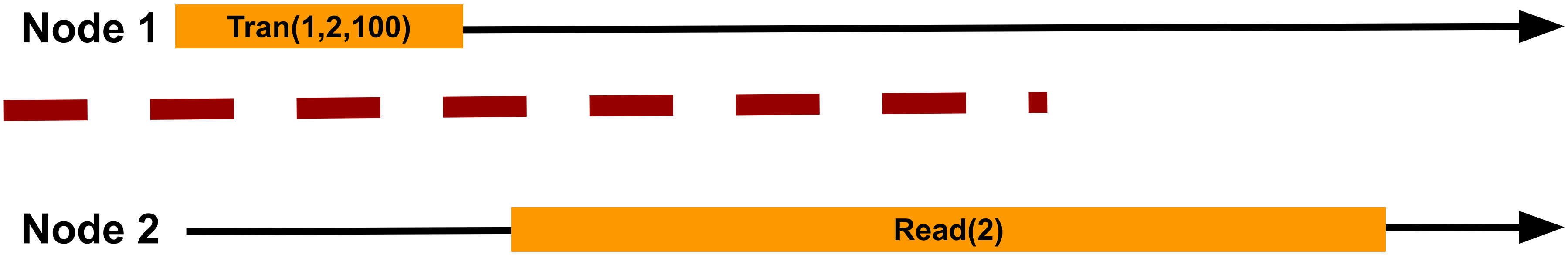
ASO-Crypto: Incorrect

	Balance
1	100
2	0
3	0



ASO-Crypto: Slow

	Balance
1	0
2	100
3	0



Cryptocurrency in a Partitioned Network?
(under submission)

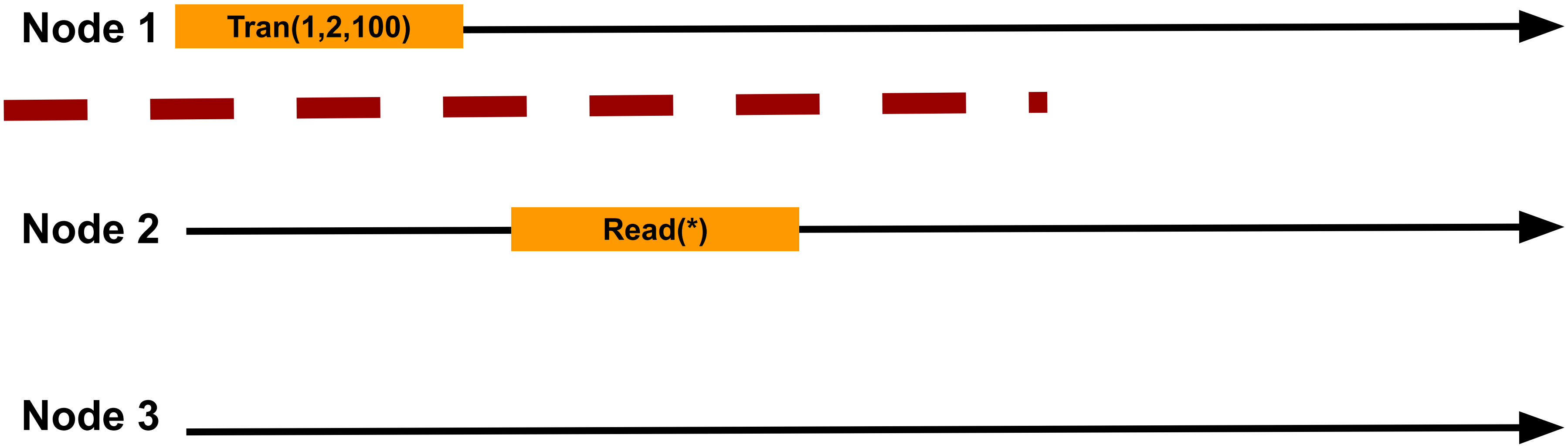
Key observation:

Pending transactions

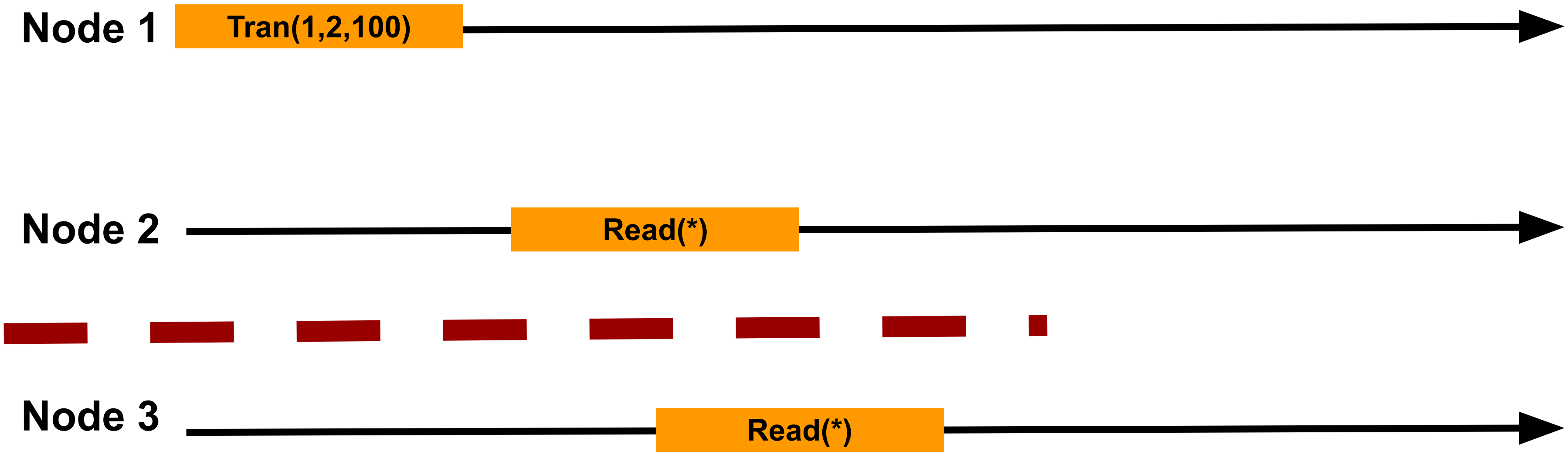
(Delivered but not applied)

Modern Bank

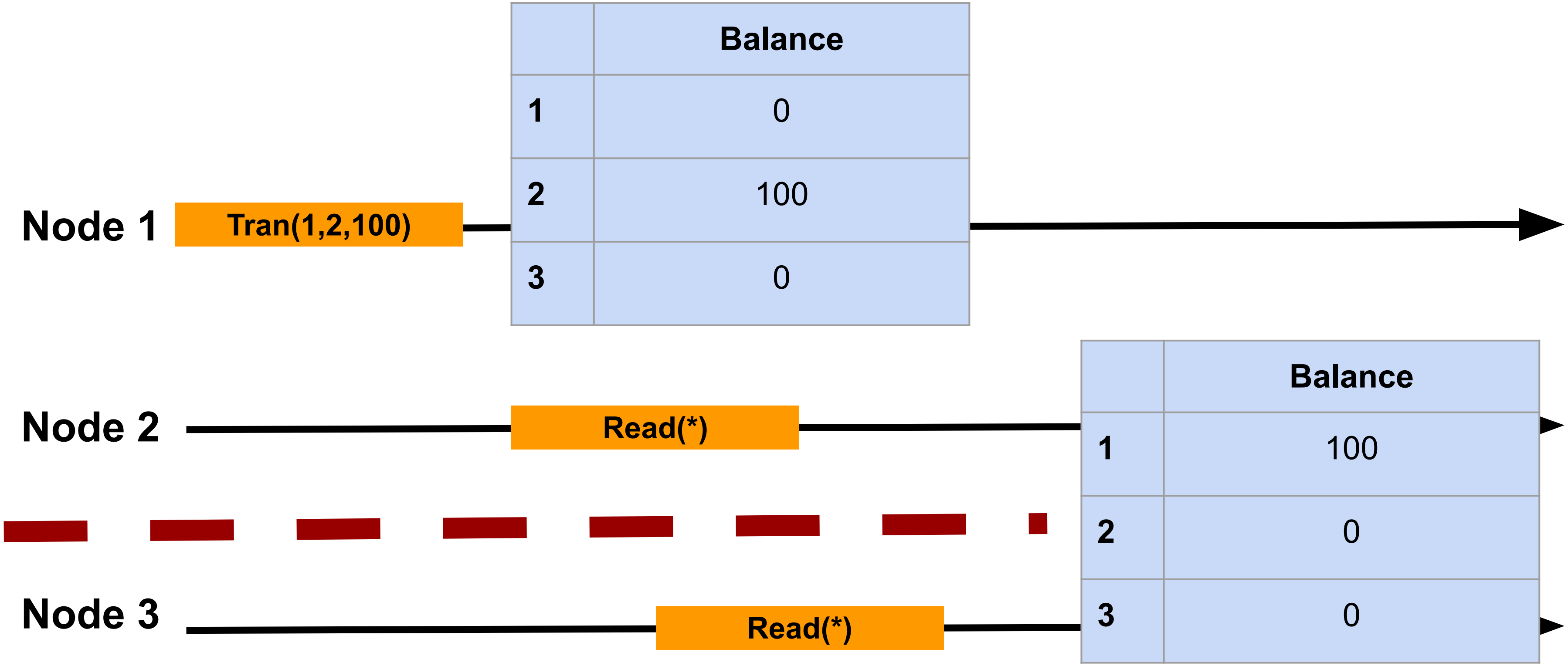
	Balance
1	100
2	0
3	0



Modern Bank



Modern Bank



Node 1
Tran(1,2,100)
Balance
1
0
2
100
3
0
Node 2
Read(*)
Balance
1
100
2
0
3
0
Node 3
Read(*)

Key tool:

abstracting consistency guarantees

Operations

Transfer

Read

Audit: return the “validity proof” of all the outgoing transactions

Valid transaction: no double-spend, no counterfeit, no overdraft

Properties

Eventual delivery:	Tx from the same partition is eventually applied
Local operation:	No communication needed to complete an operation
Read-my-write:	Read reflects the effect of all the prior outgoing transfers
Auditability:	One is able to present validity proof
Validity:	All applied transactions are valid

Properties under **C**AP framework

Eventual delivery: Tx from the same partition is eventually applied

Local operation: No communication needed to complete an operation

Read-my-write: Read reflects the effect of all the prior outgoing transfers

Auditability: One is able to present validity proof

Validity: All applied transactions are valid

Properties under CAP framework

Eventual delivery: Tx from the same partition is eventually applied

Local operation: No communication needed to complete an operation

Read-my-write: Read reflects the effect of all the prior outgoing transfers

Auditability: One is able to present validity proof

Validity: All applied transactions are valid

Properties under CA**P** framework

Eventual delivery: Tx from the same partition is eventually applied

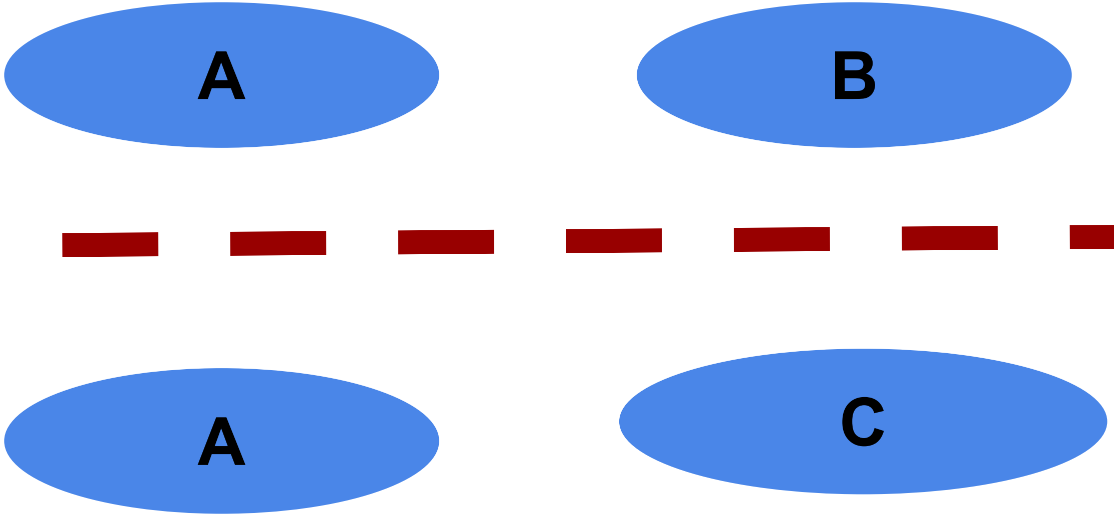
Local operation: No communication needed to complete an operation

Read-my-write: Read reflects the effect of all the prior outgoing transfers

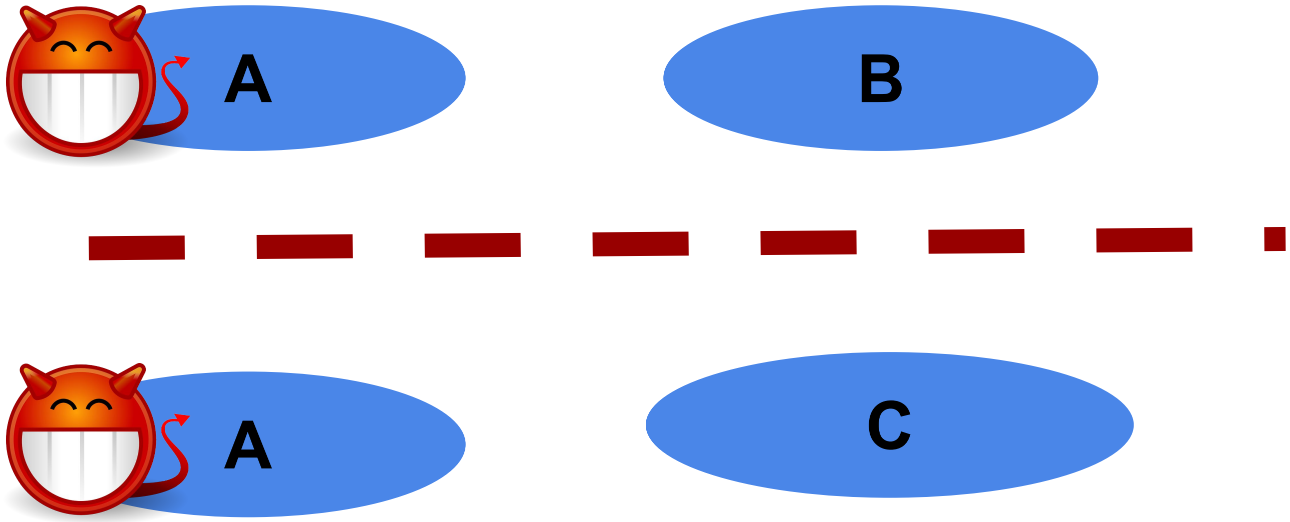
Auditability: One is able to present validity proof

Validity: All applied transactions are valid

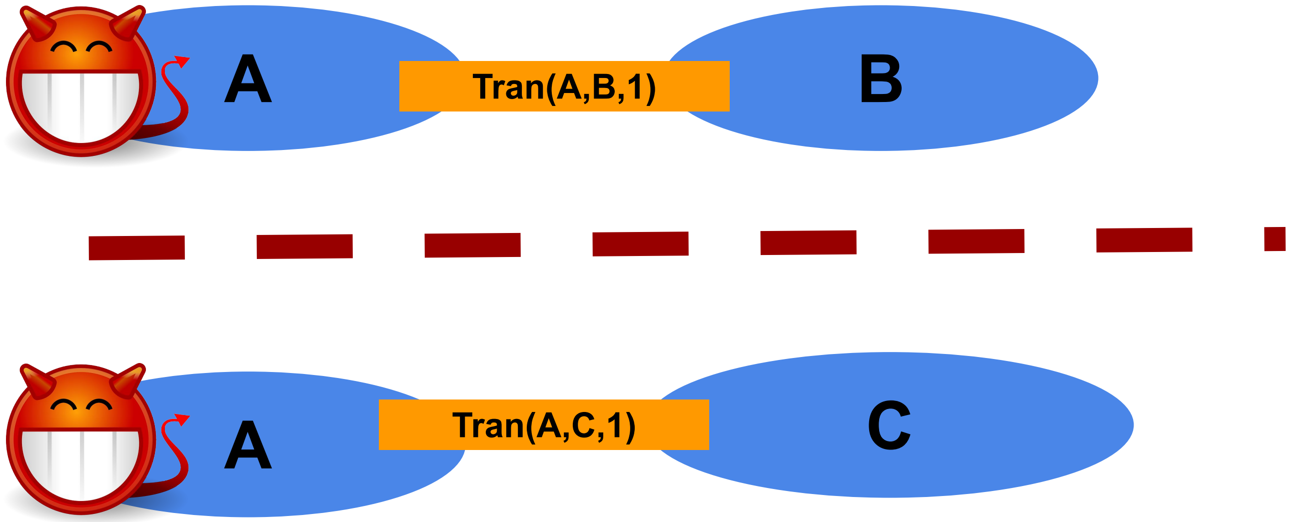
Thm1: Byzantine node + Eventual delivery + Partition-tolerance = impossible



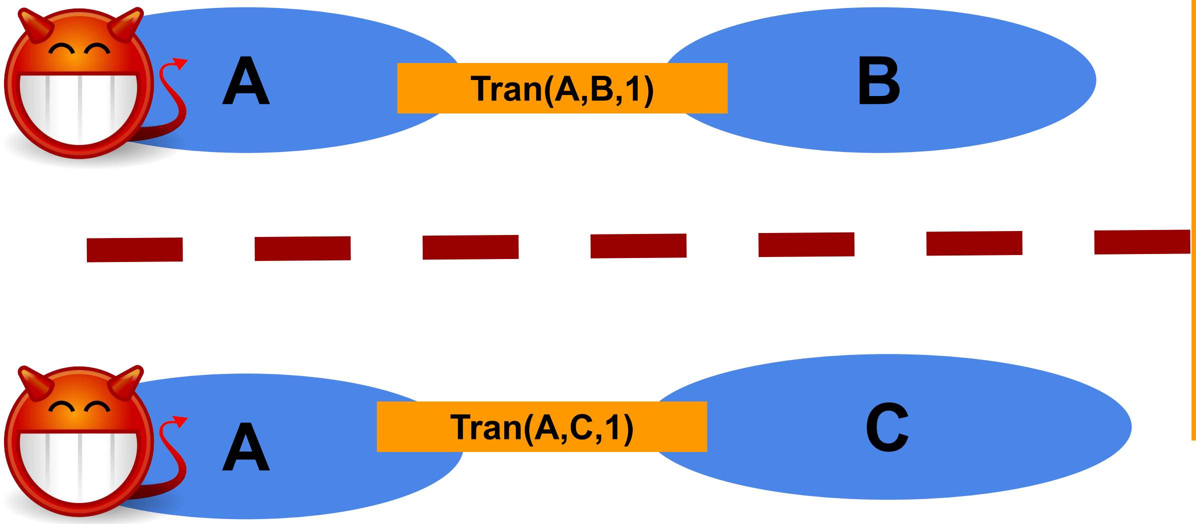
Thm1: Byzantine node + Eventual delivery + Partition-tolerance = impossible



Thm1: Byzantine node + Eventual delivery + Partition-tolerance = impossible



Thm1: Byzantine node + Eventual delivery + Partition-tolerance = impossible

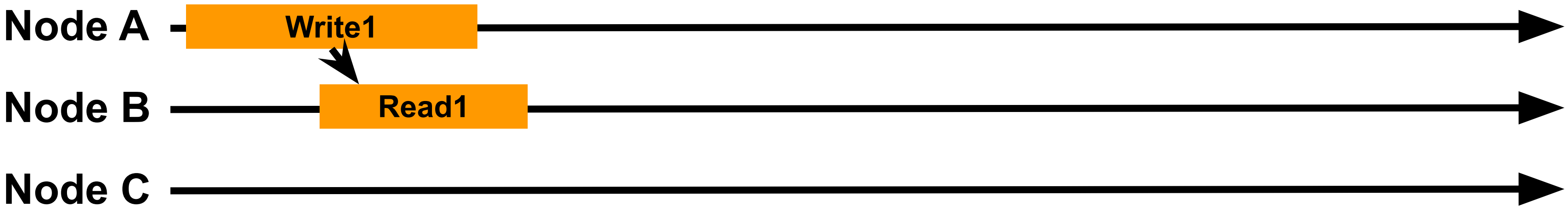


Don't apply:
violating eventual delivery

Apply:
double-spend

Causal consistency [Ahamad et al. DC 95]

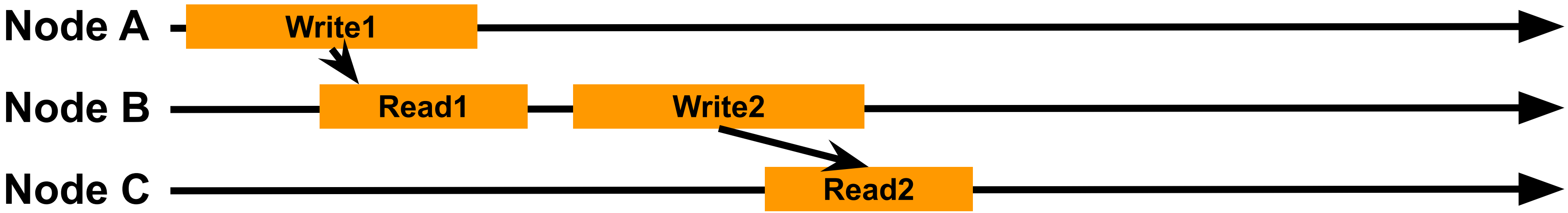
On a high-level, causal consistency lets each node observe the entire causal history (happens-before relation)



Write1 → Read1

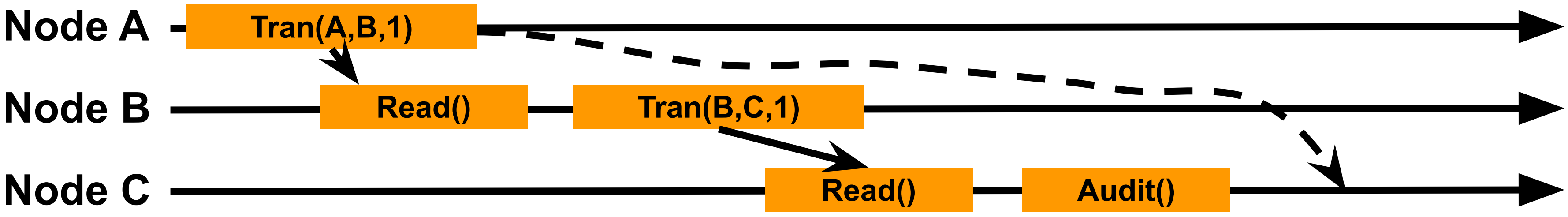
Causal consistency [Ahamad et al. DC 95]

On a high-level, causal consistency lets each node observe the entire causal history (happens-before relation)

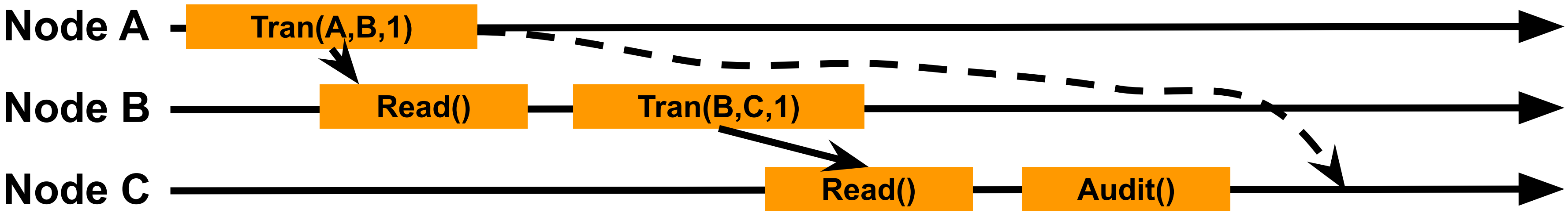


Write1 → Read1 → Write2 → Read2
⇒ Write1 → Read2

Thm2: Causal consistency is necessary



Thm2: Causal consistency is necessary



Audit() does NOT contain Tran(A,B,1)

⇒ Node C violates auditability!

Positive Results

CCC: Causal Cryptocurrency under Crash faults

- Similar to ASO-Crypto, but use causal memory underneath

CCB: Causal Cryptocurrency under Byzantine faults

- Byzantine causal memory [Tseng et al. NCA 19]
- Reliable broadcast [Bracha and Toueg JACM 85]
- Sequence number to stop double-spending
- PKI and digital signature
- A weaker form of eventual delivery:

  one needs to be able to talk to n-f correct nodes

Summary

ASO-Crypto: consensus not necessary

Our work: total order and strong consistency not necessary

causal consistency necessary

an inherent challenge of Byzantine crypto in partition

two implementations

Future Works

Implementation and evaluation

Permission-less systems

Probabilistic guarantees

Advice

Know fundamentals

- FLP
- CAP

Reach out to other communities

Be comfortable with formalism

Thanks!
Questions?

lewis.tseng@bc.edu