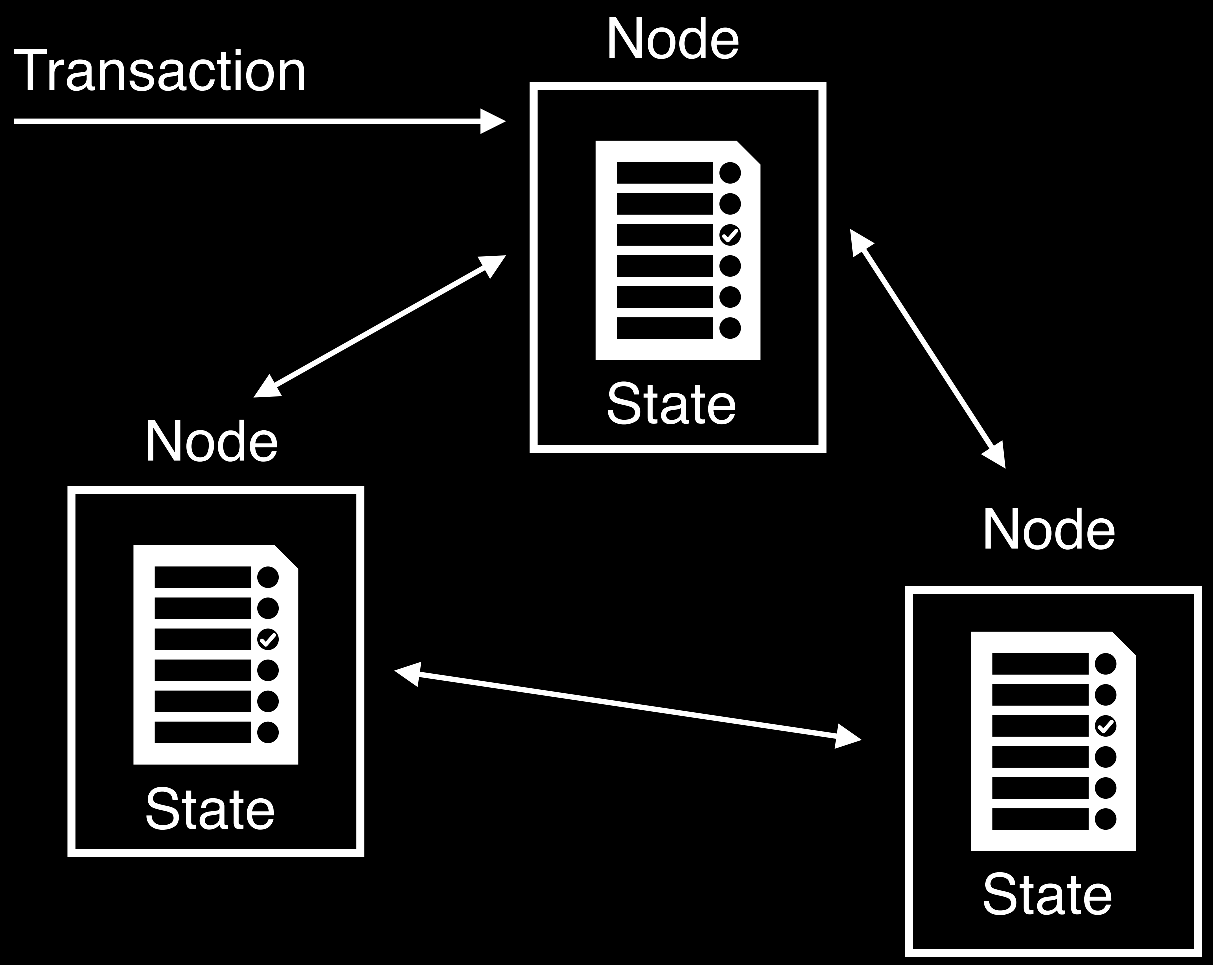
Blurring the Lines between Blockchains and Database Systems: the Case of Hyperledger Fabric

Ankur Sharma*
Felix Martin Schuhknecht
Divya Agrawal
Jens Dittrich

Big Data Analytics Group
Saarland University
bigdata.uni-saarland.de



Transaction

Node

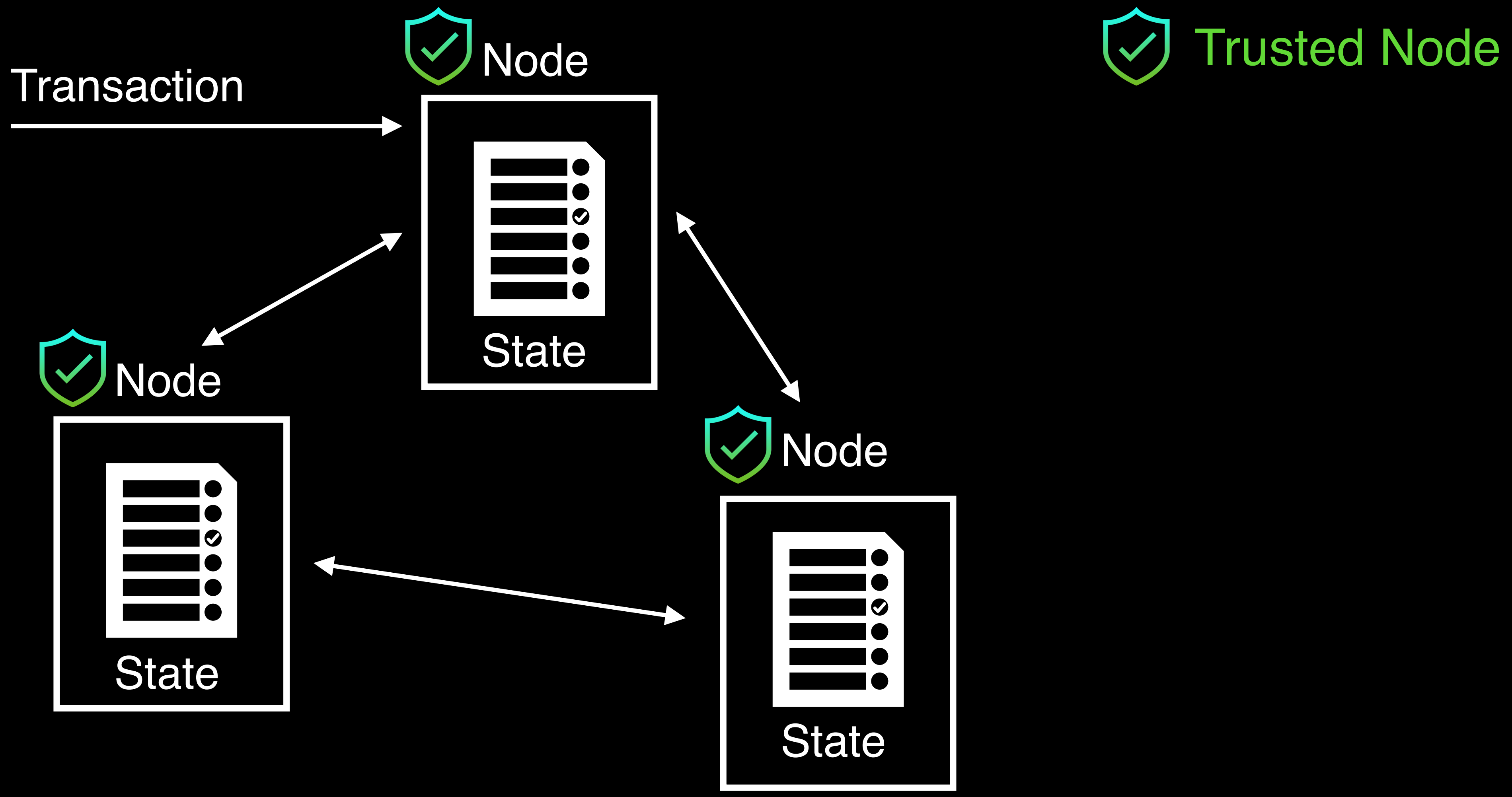
State

Node

State

Node

State



Trusted Node

Transaction

Node

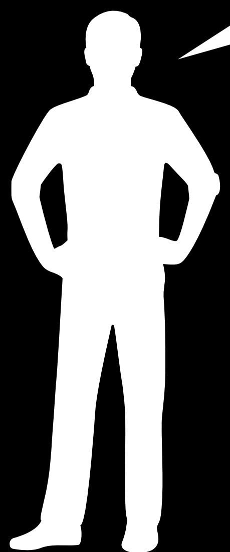
State

Node

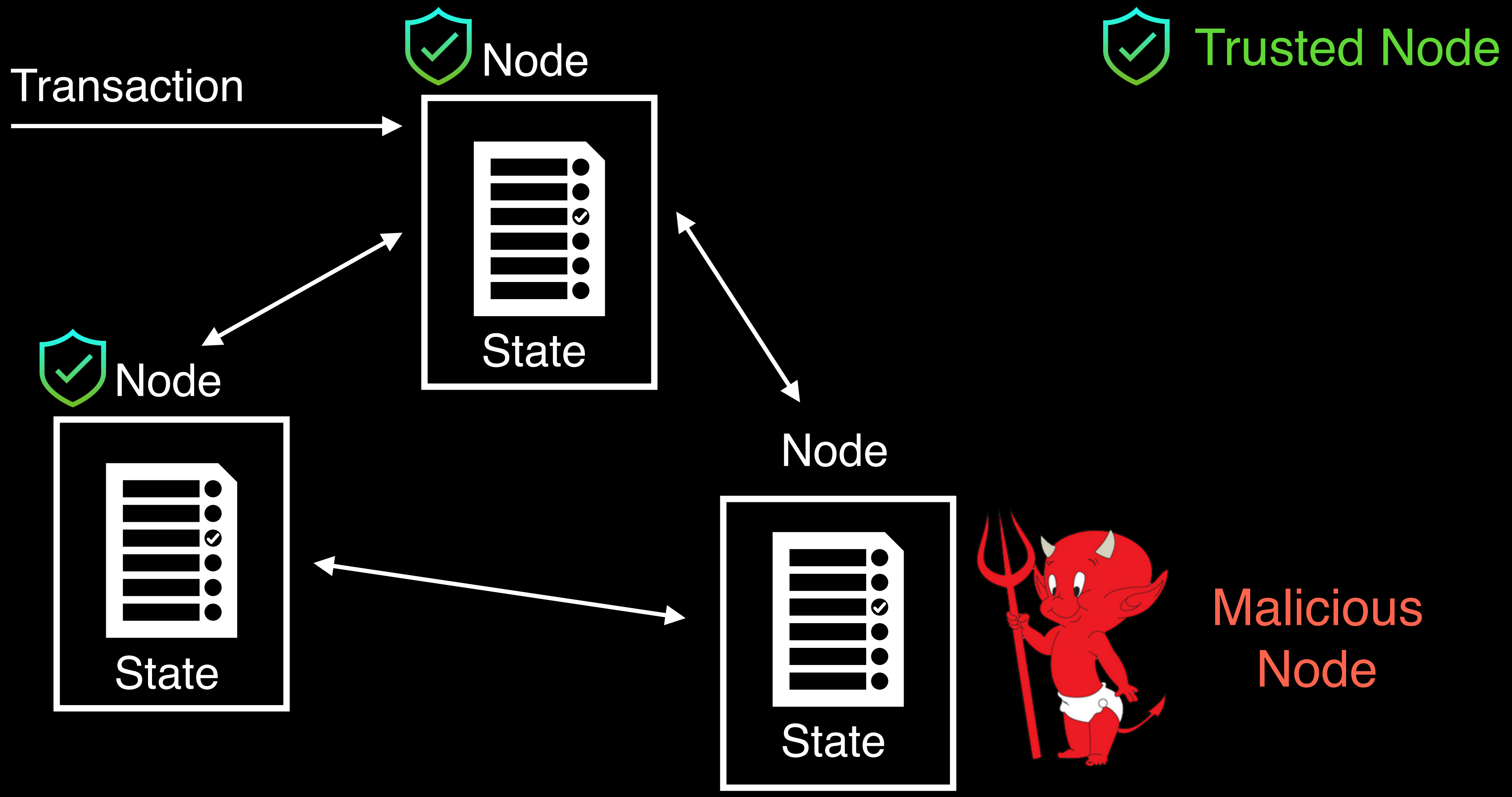
State

Node

State



This is a distributed database system.



Trusted Node

Transaction

Node

State

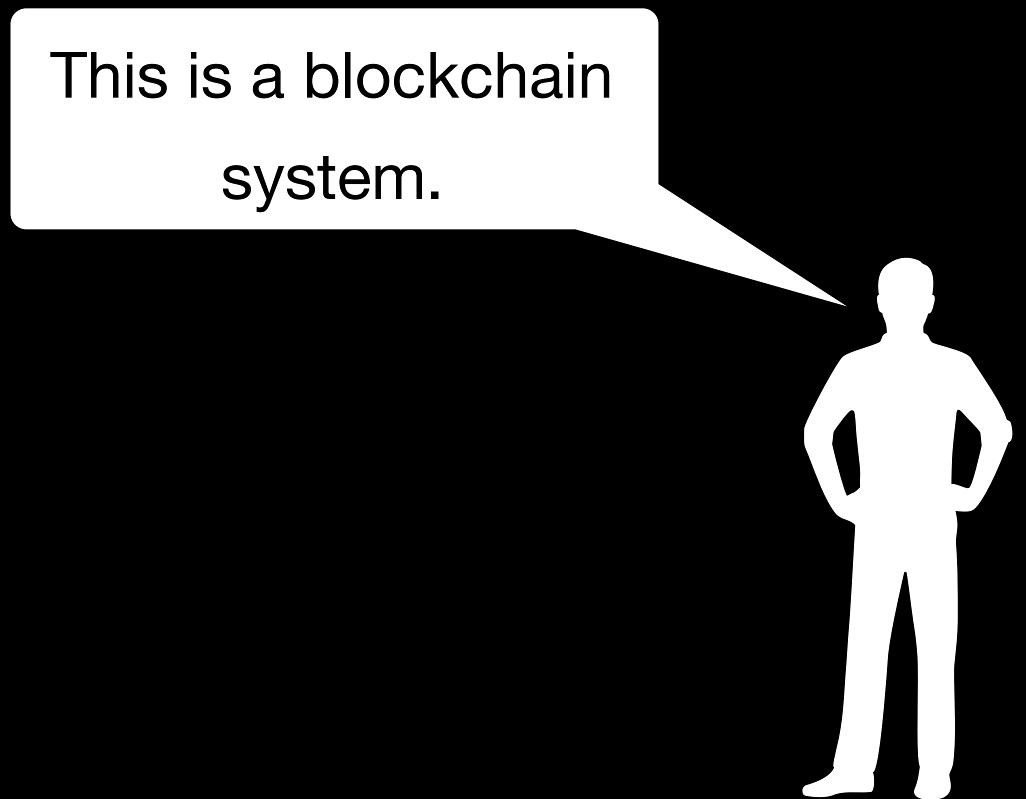
Node

State

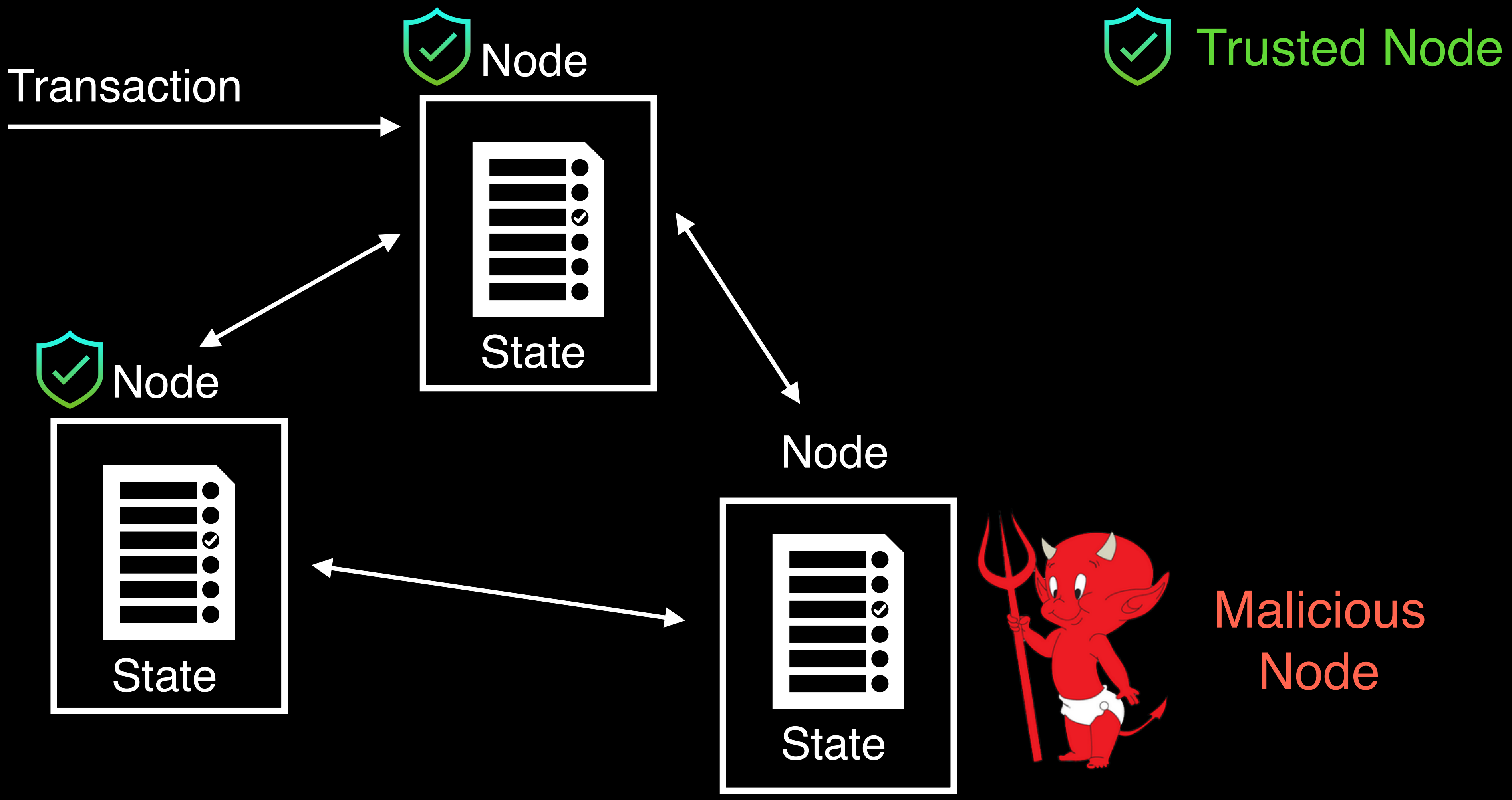
Node

State

Malicious Node



This is a blockchain system.



Trusted Node

Transaction

Node

State

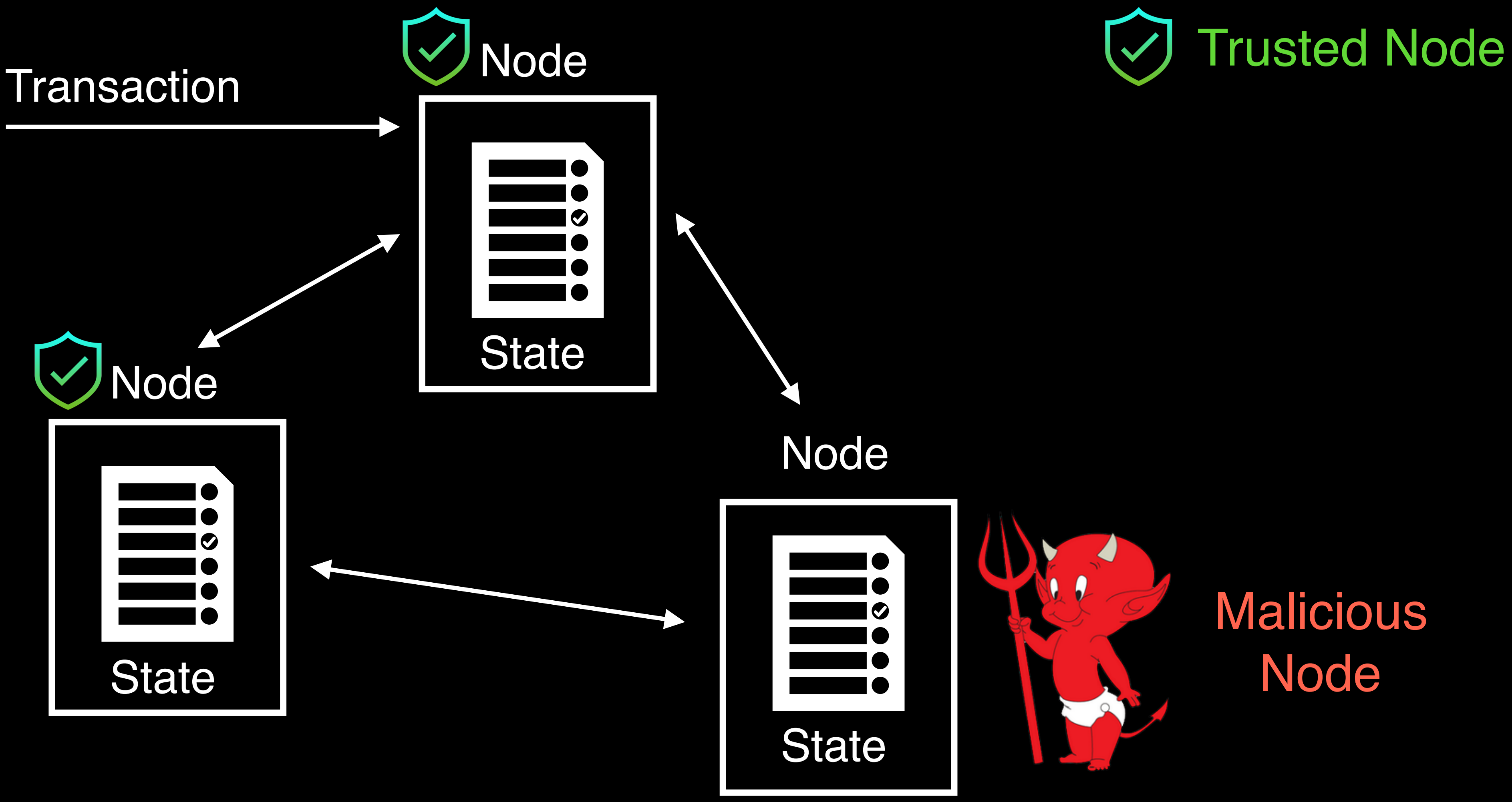
Node

State

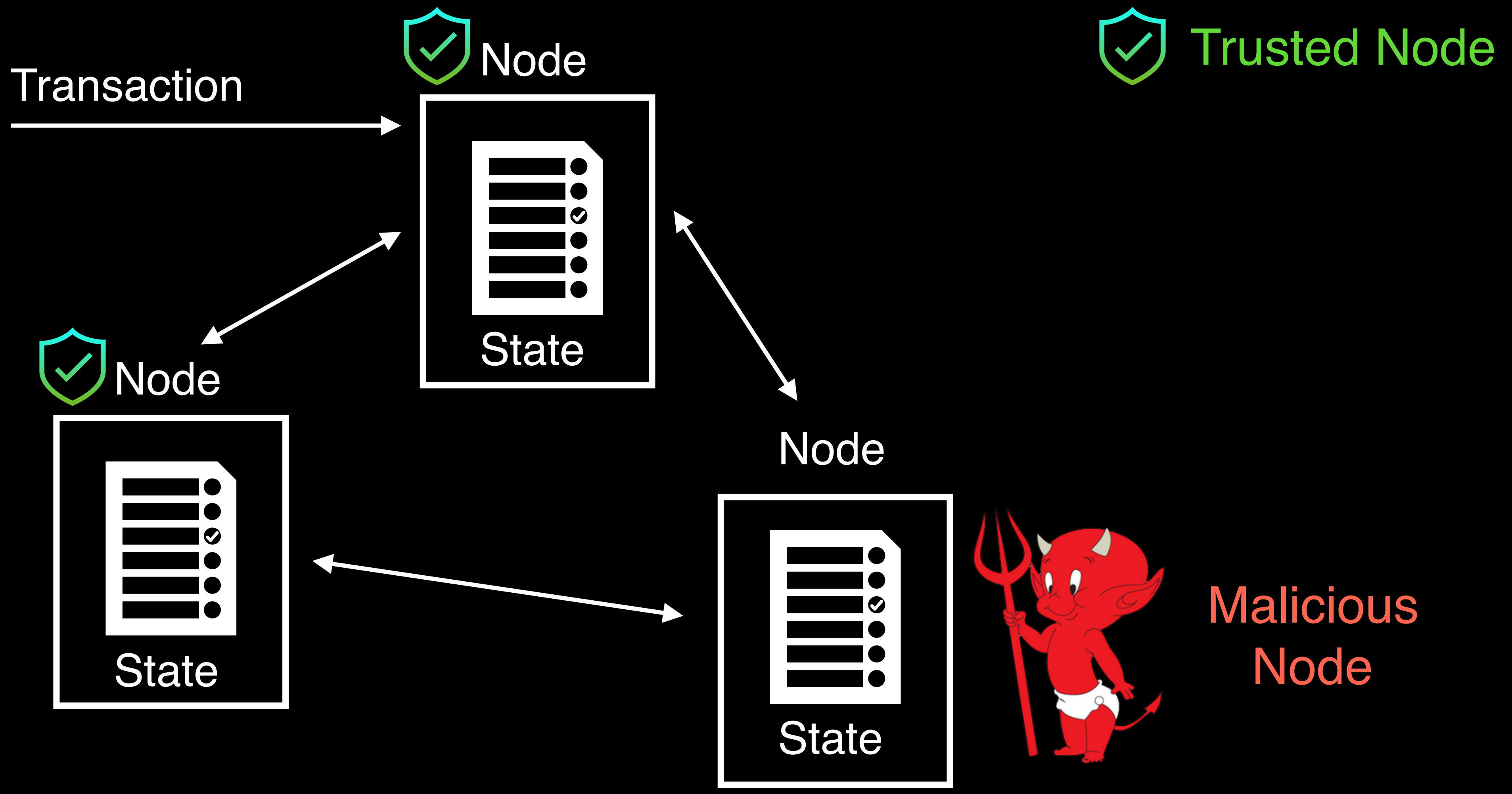
Node

State

Malicious Node

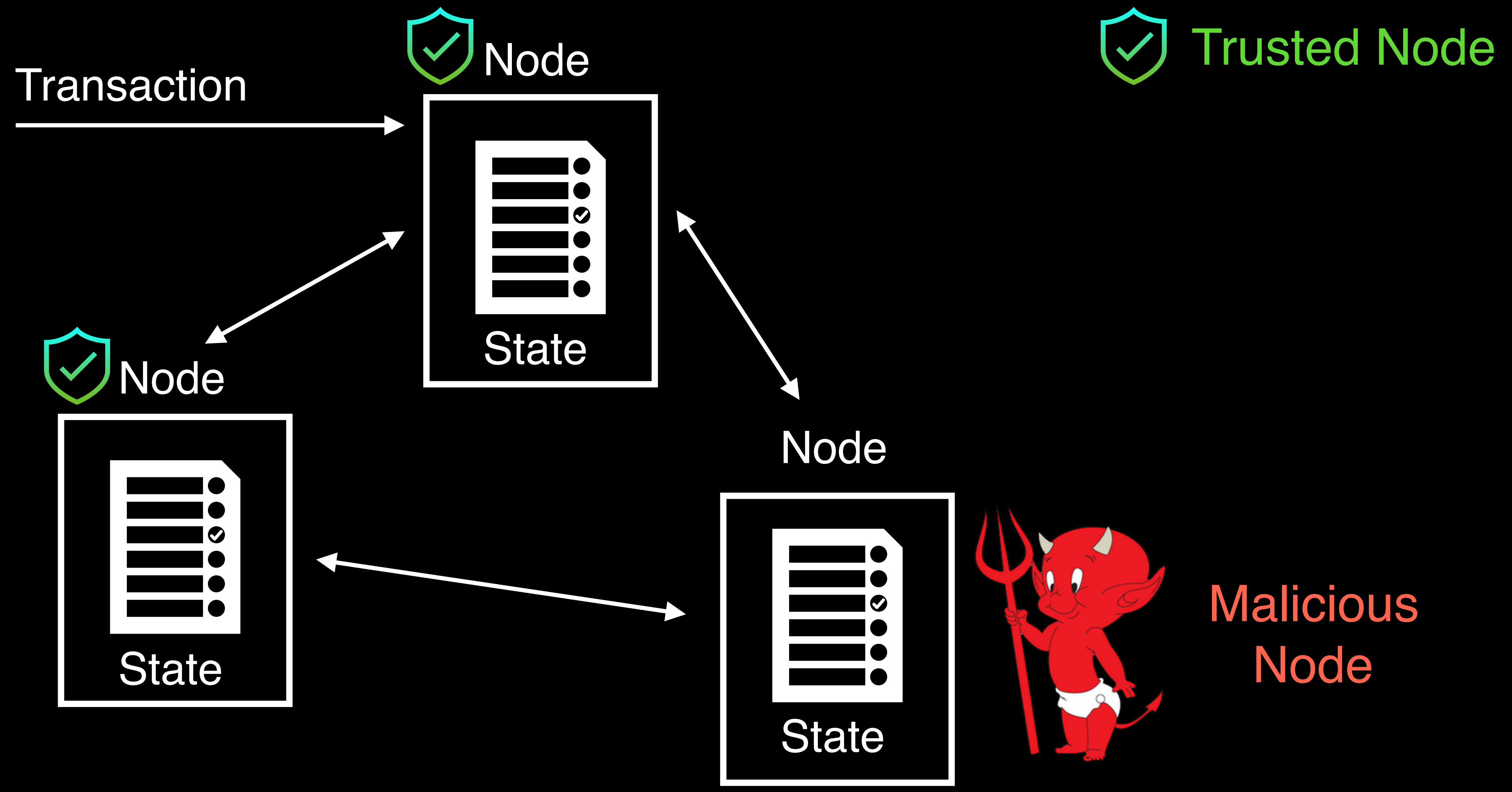


Can we say that the Blockchain Systems are next-gen Distributed Database Systems?



Can we say that the Blockchain Systems are next-gen Distributed Database Systems?

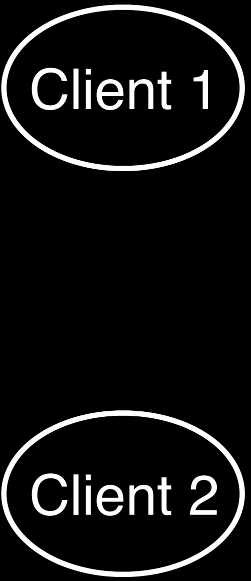
Not really!



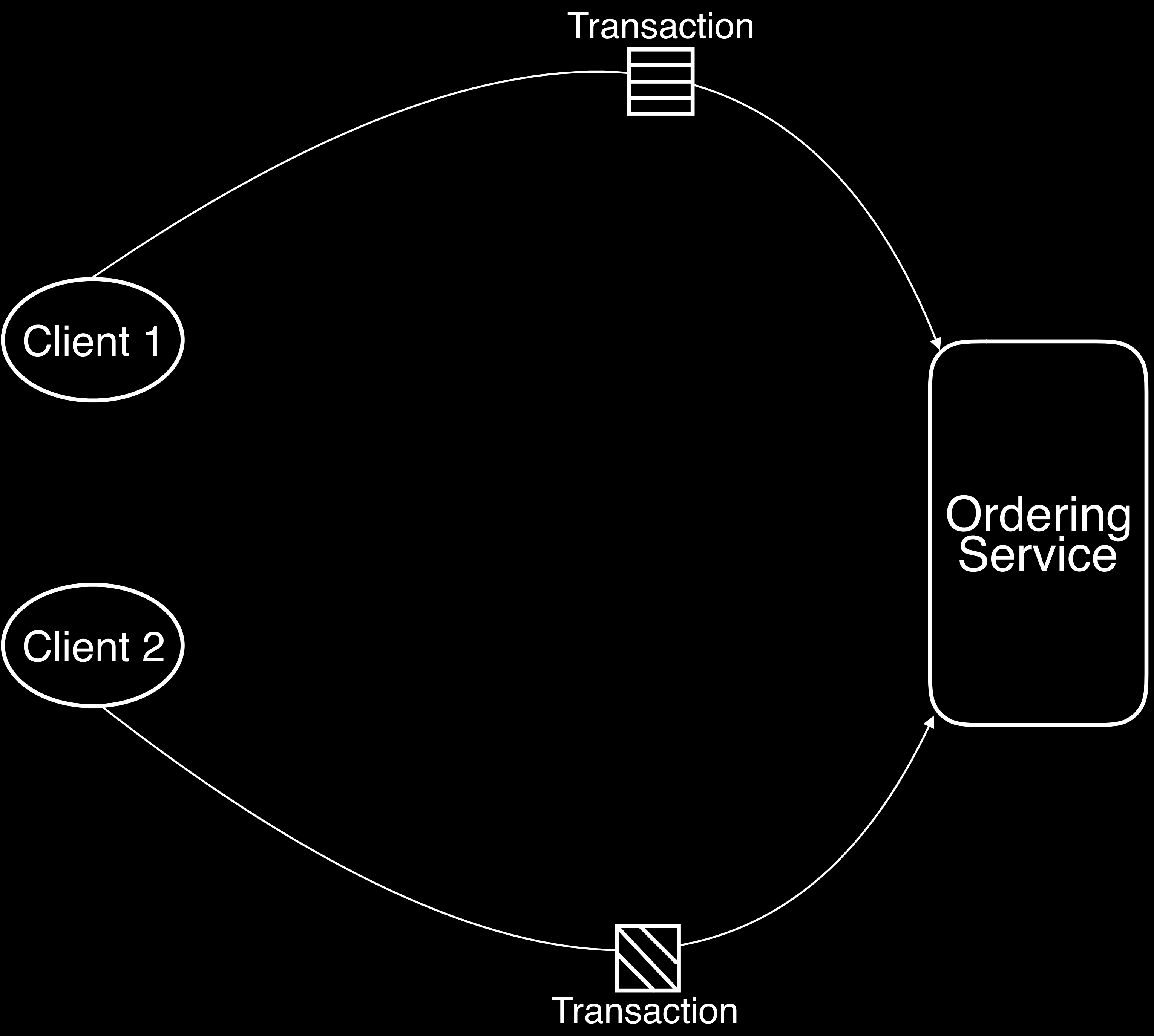
Can we say that the Blockchain Systems are next-gen Distributed Database Systems?

Not really! → **Outdated Transaction Processing Model**

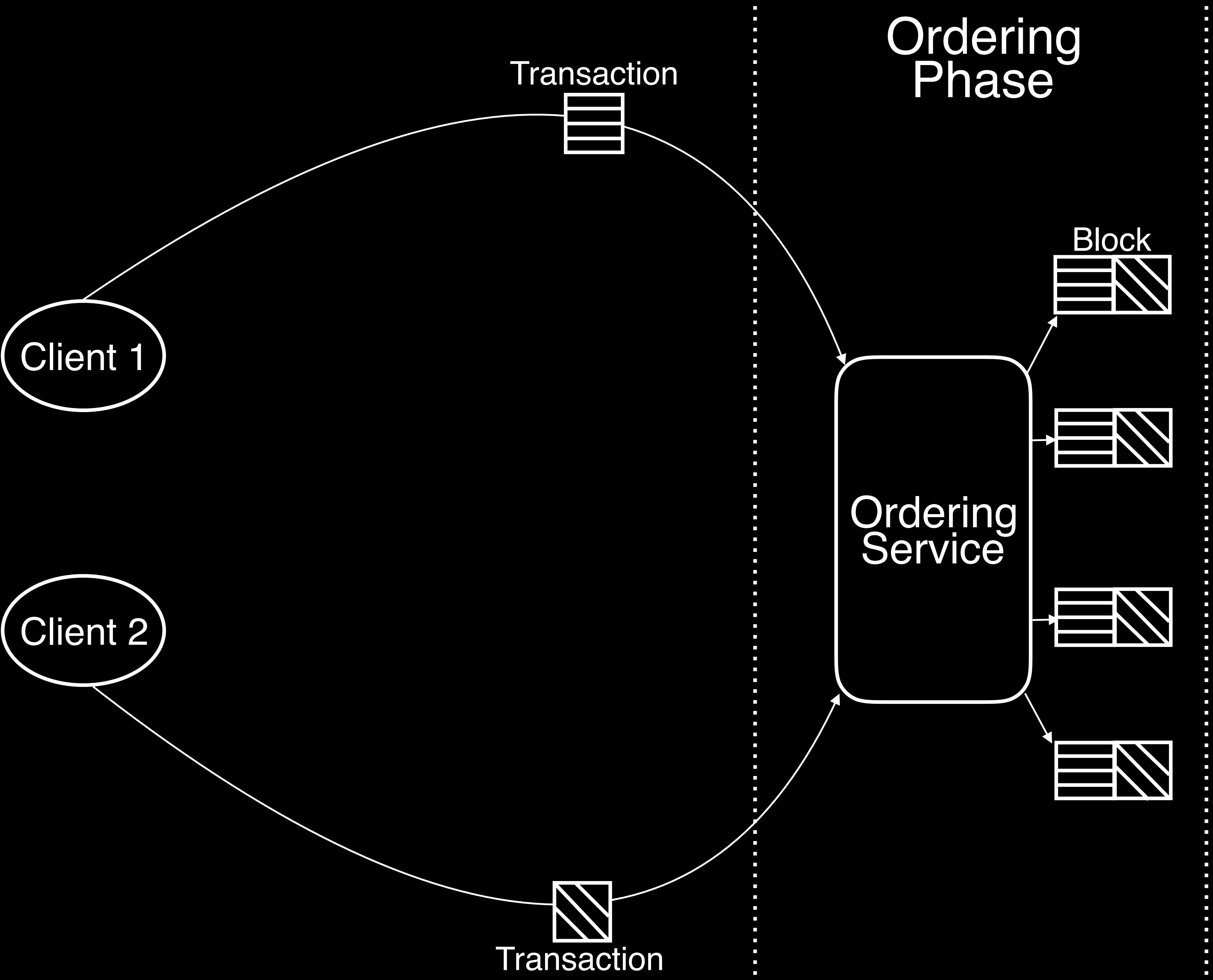
The order-execute model (Bitcoin, Ethereum, …)



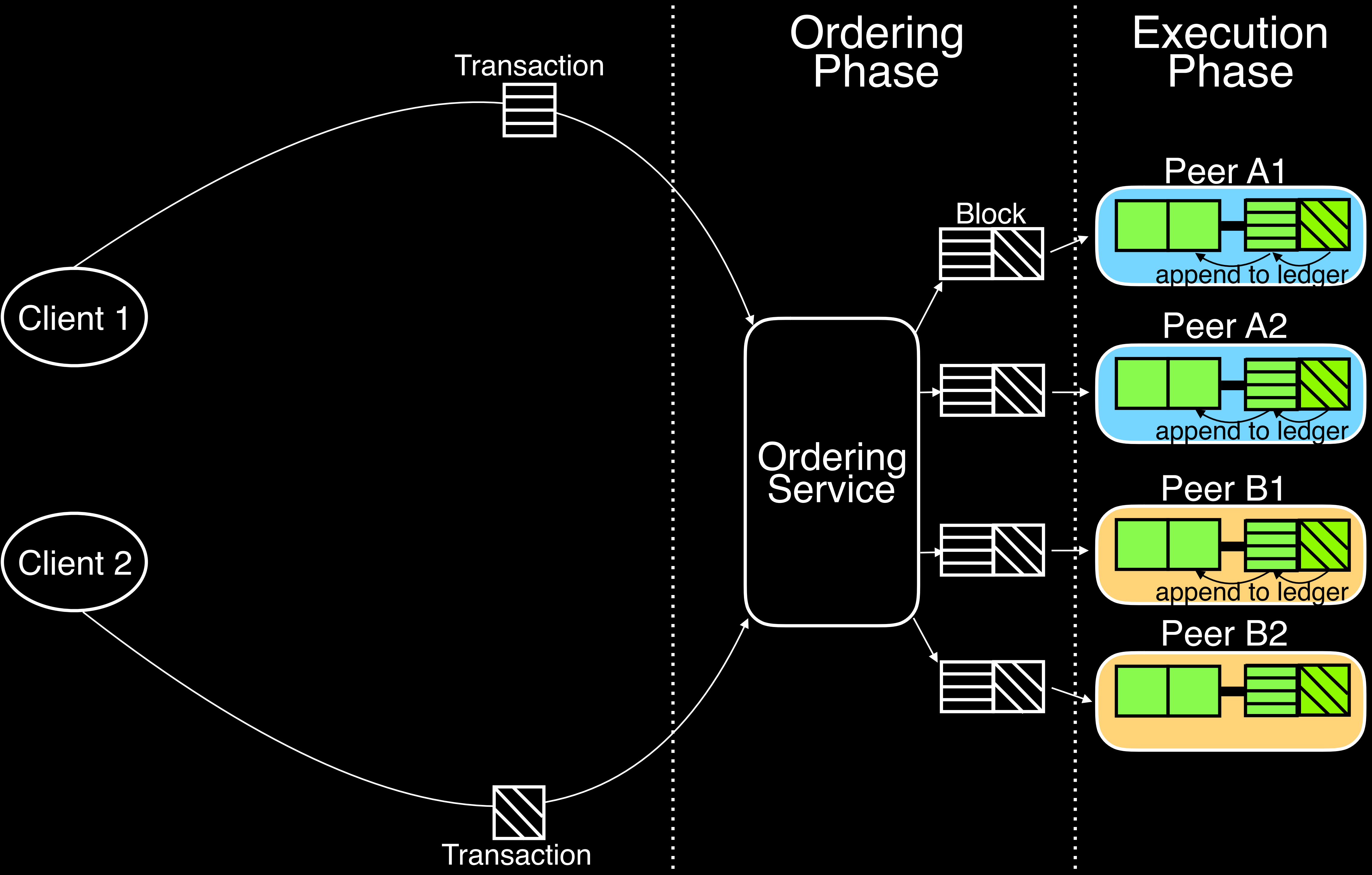
The order-execute model (Bitcoin, Ethereum, …)



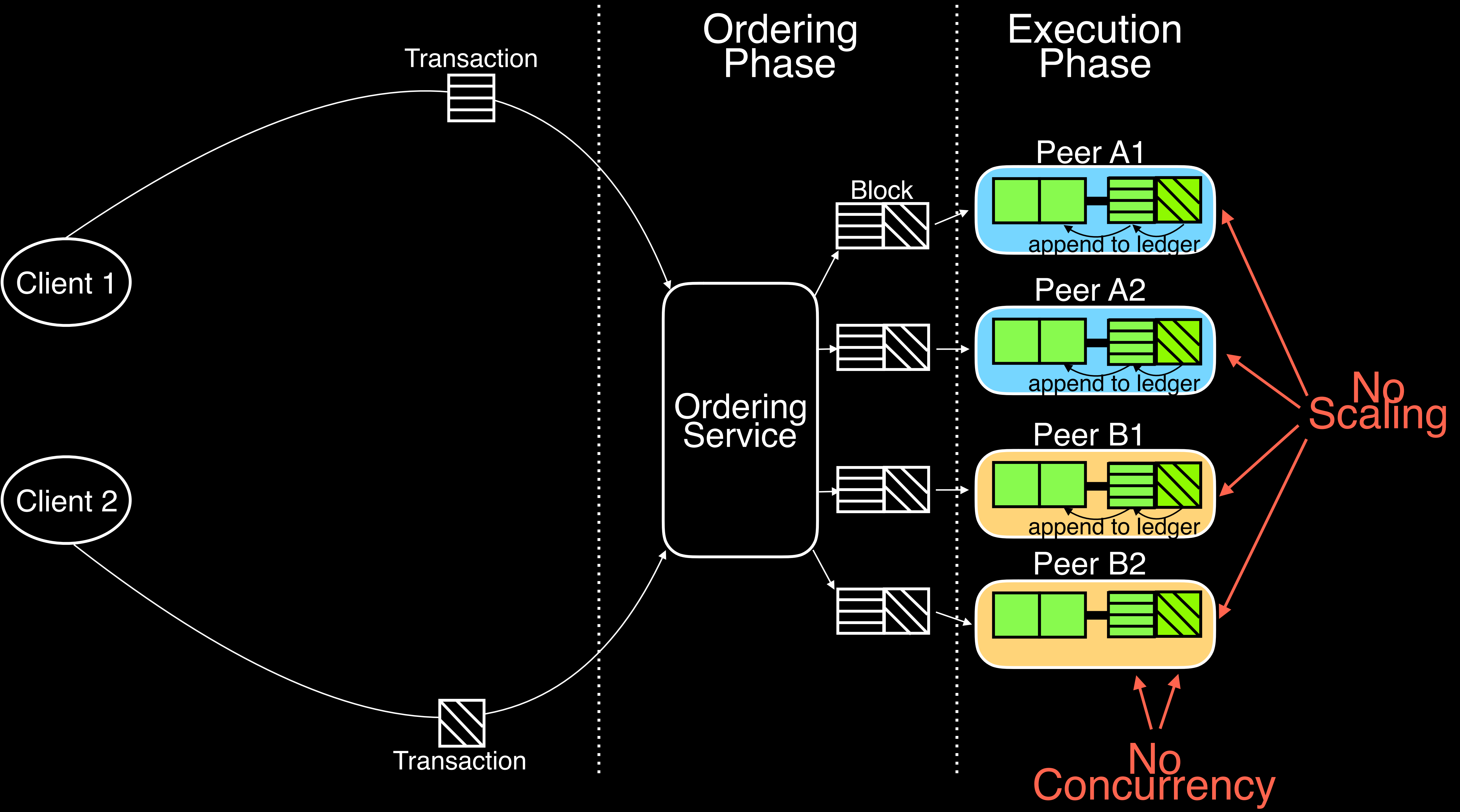
The order-execute model (Bitcoin, Ethereum, …)



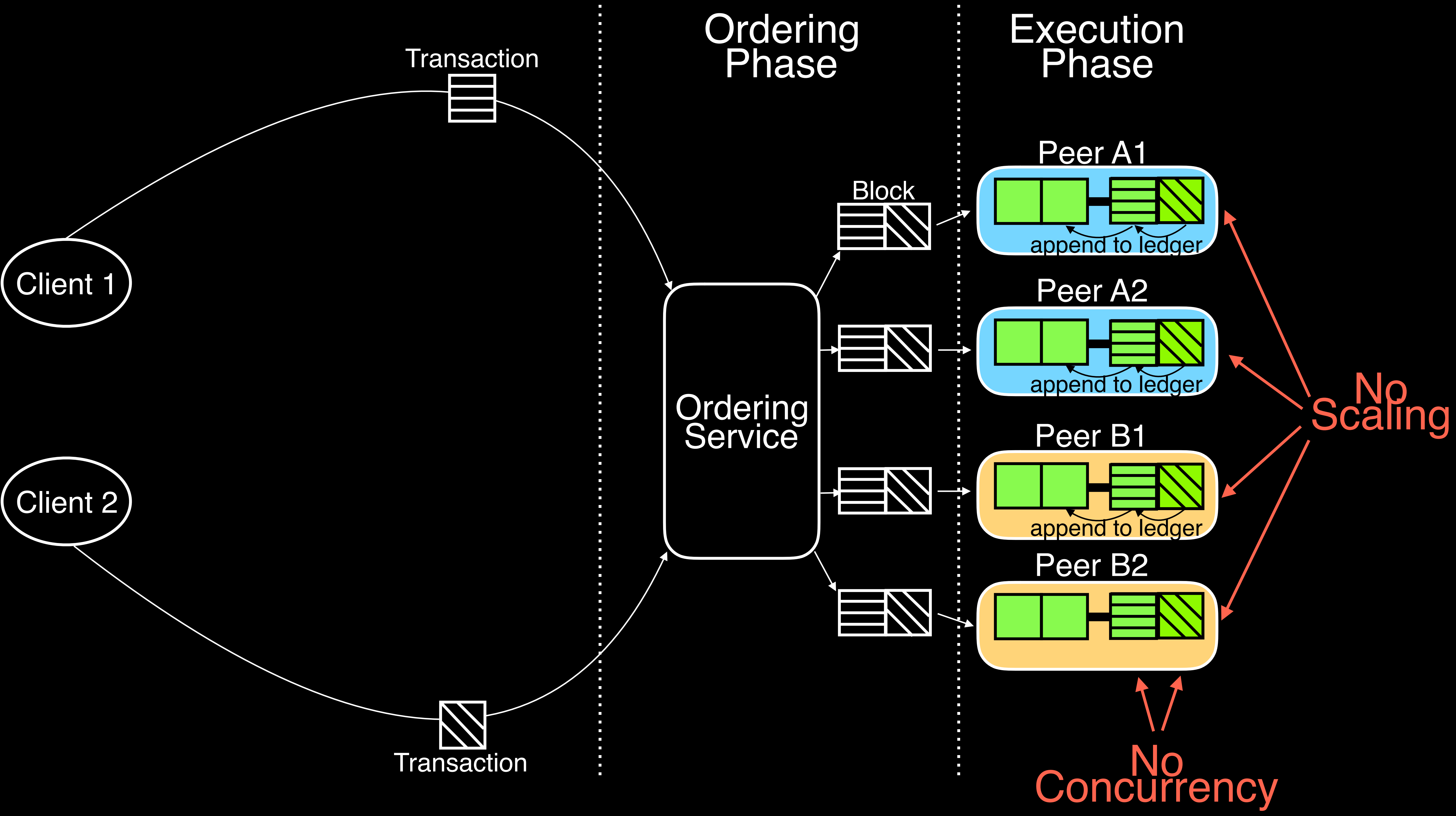
The order-execute model (Bitcoin, Ethereum, …)



The order-execute model (Bitcoin, Ethereum, …)

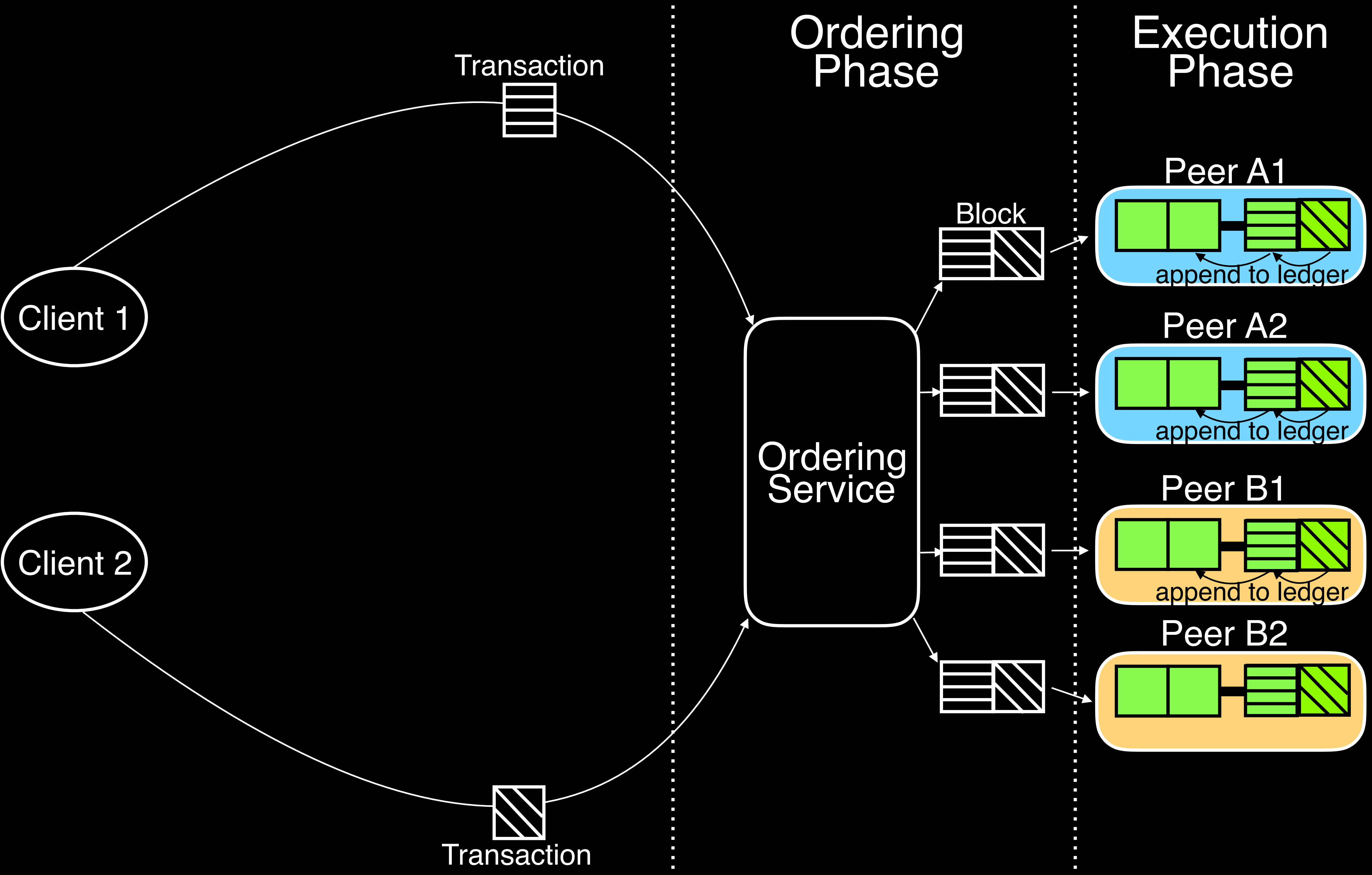


The order-execute model (Bitcoin, Ethereum, …)

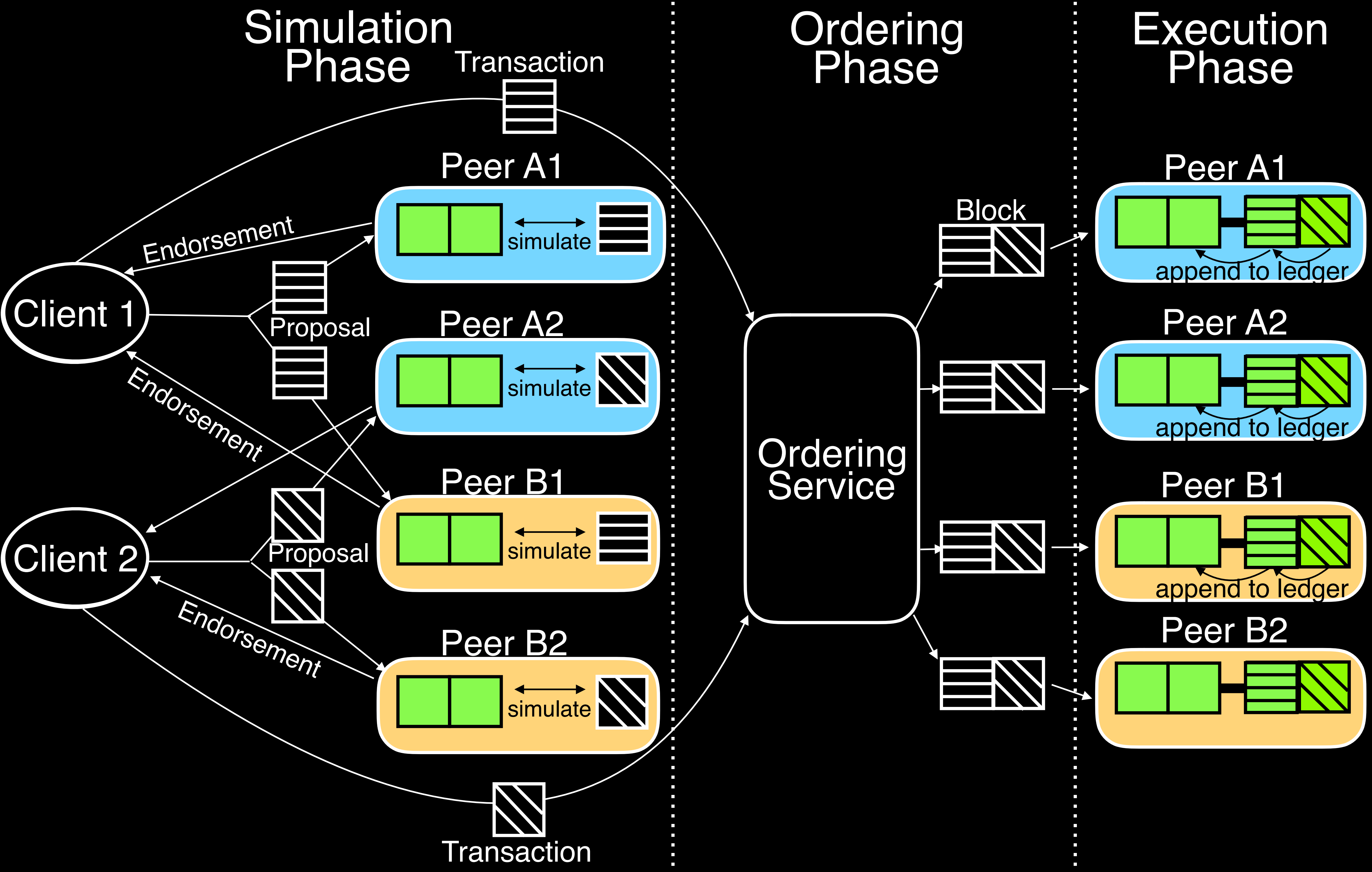


Well-established properties of database systems since decades!

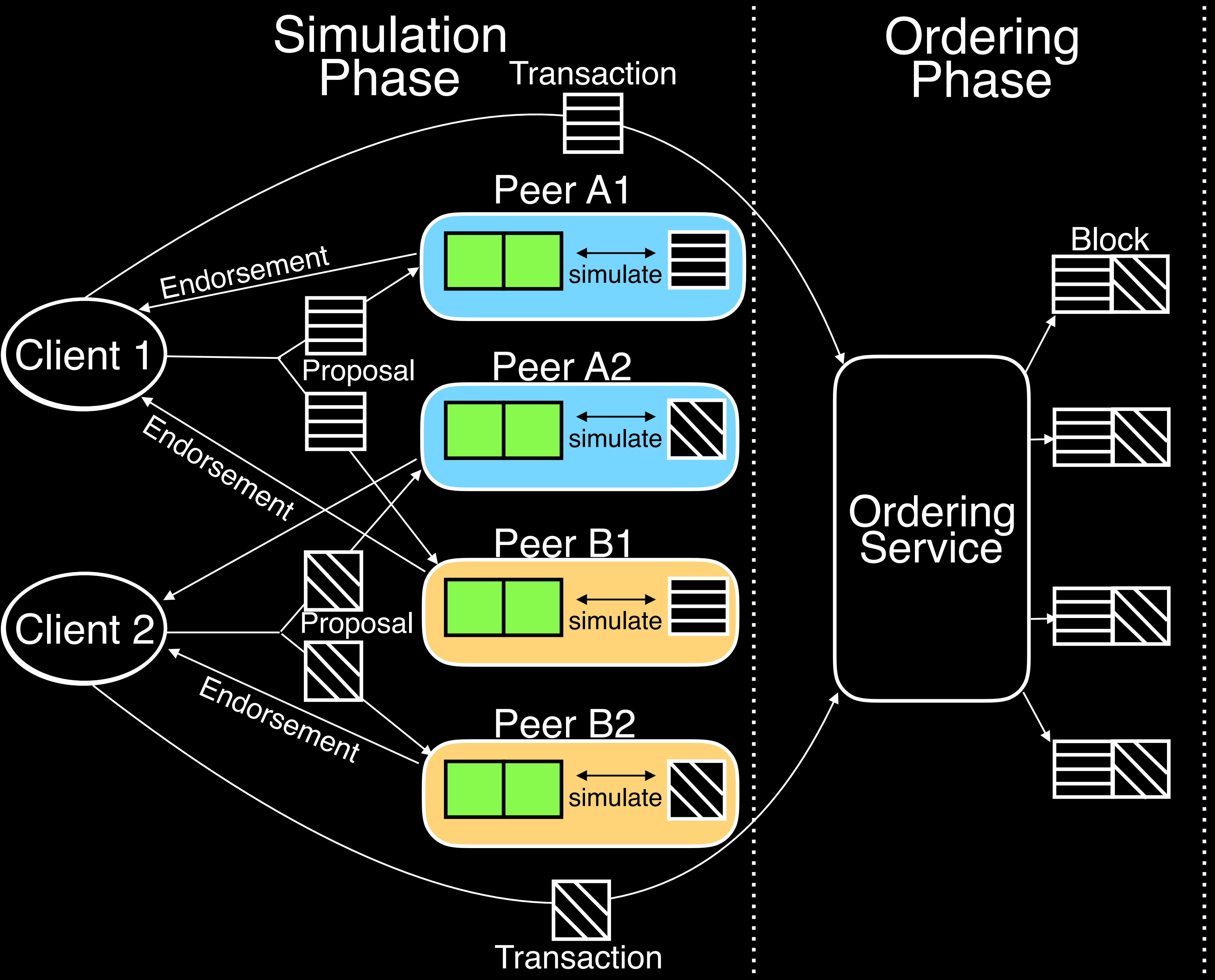
The order-execute model (Bitcoin, Ethereum, …)



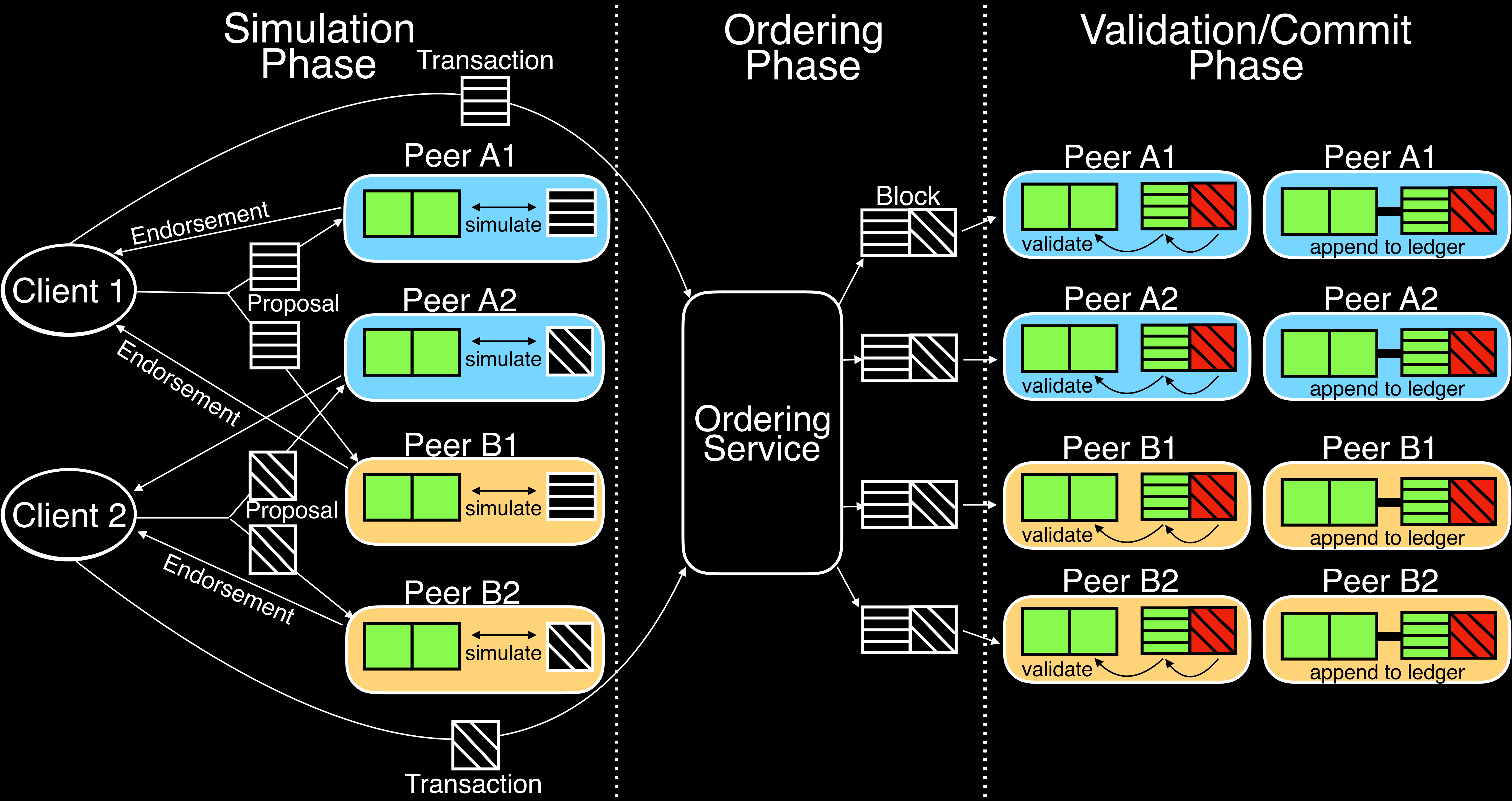
The order-execute model (Bitcoin, Ethereum, …)



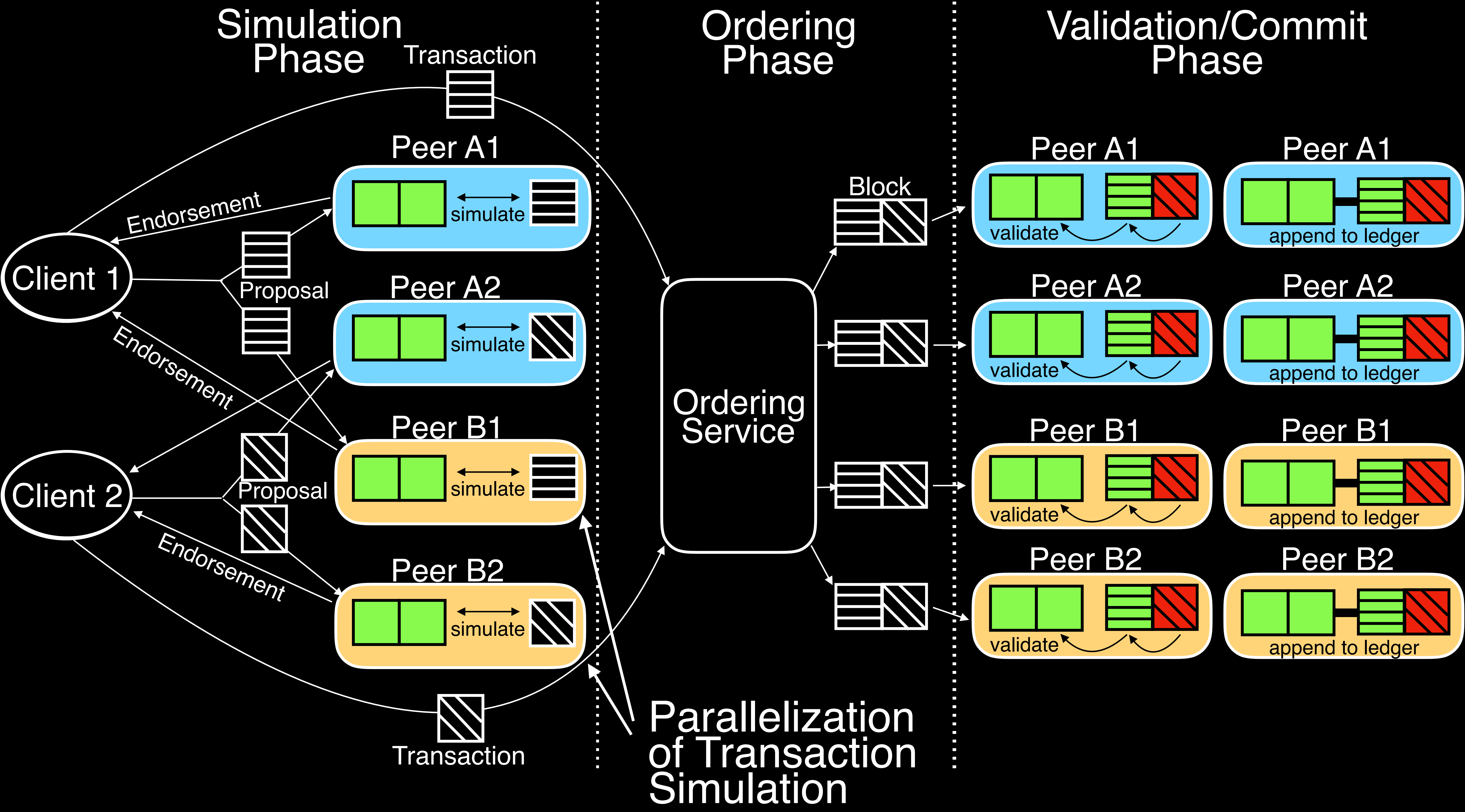
The order-execute model (Bitcoin, Ethereum, …)



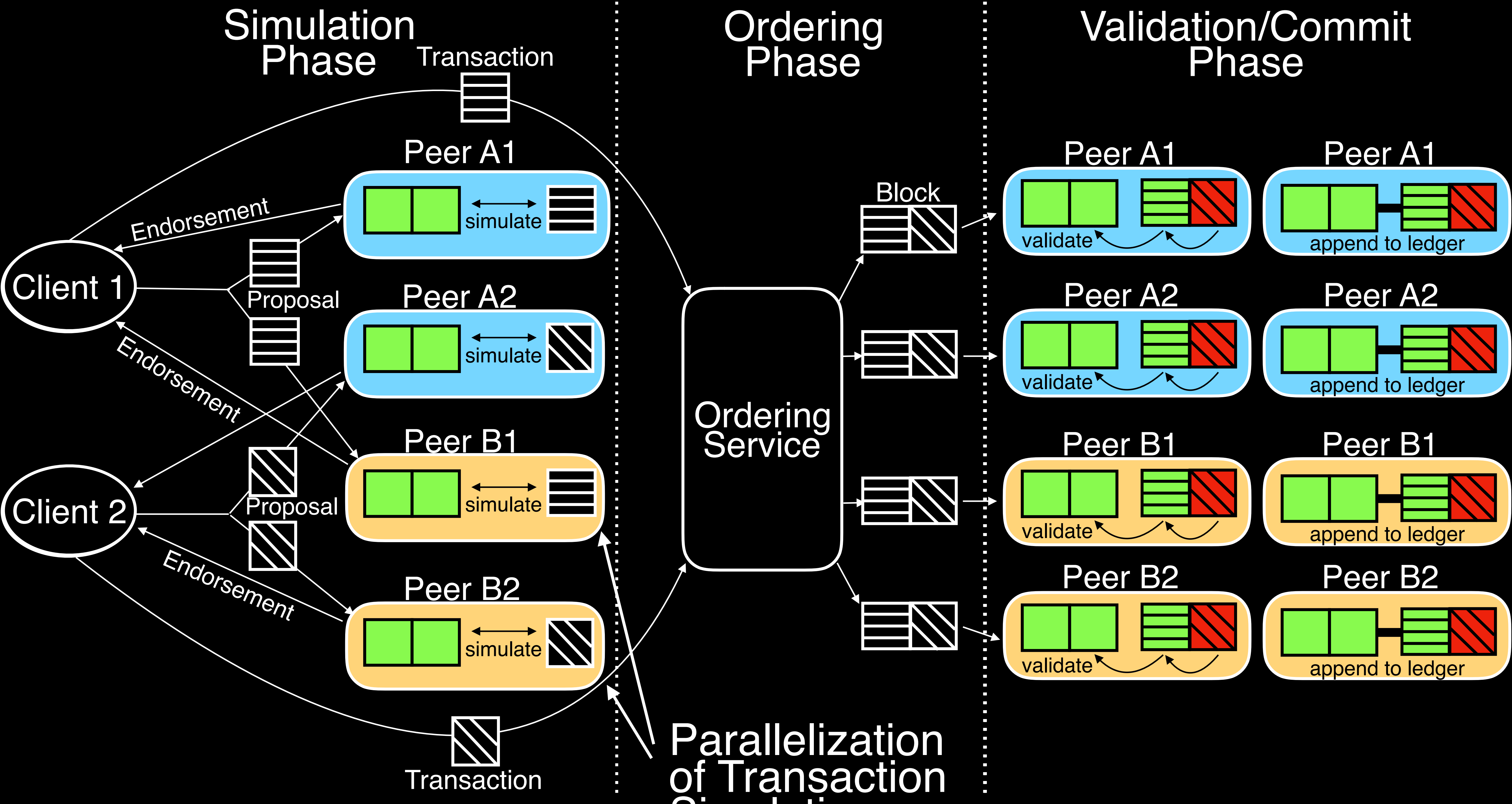
The Simulate-order-validate-commit model (Fabric)



The Simulate-order-validate-commit model (Fabric)



The Simulate-order-validate-commit model (Fabric)

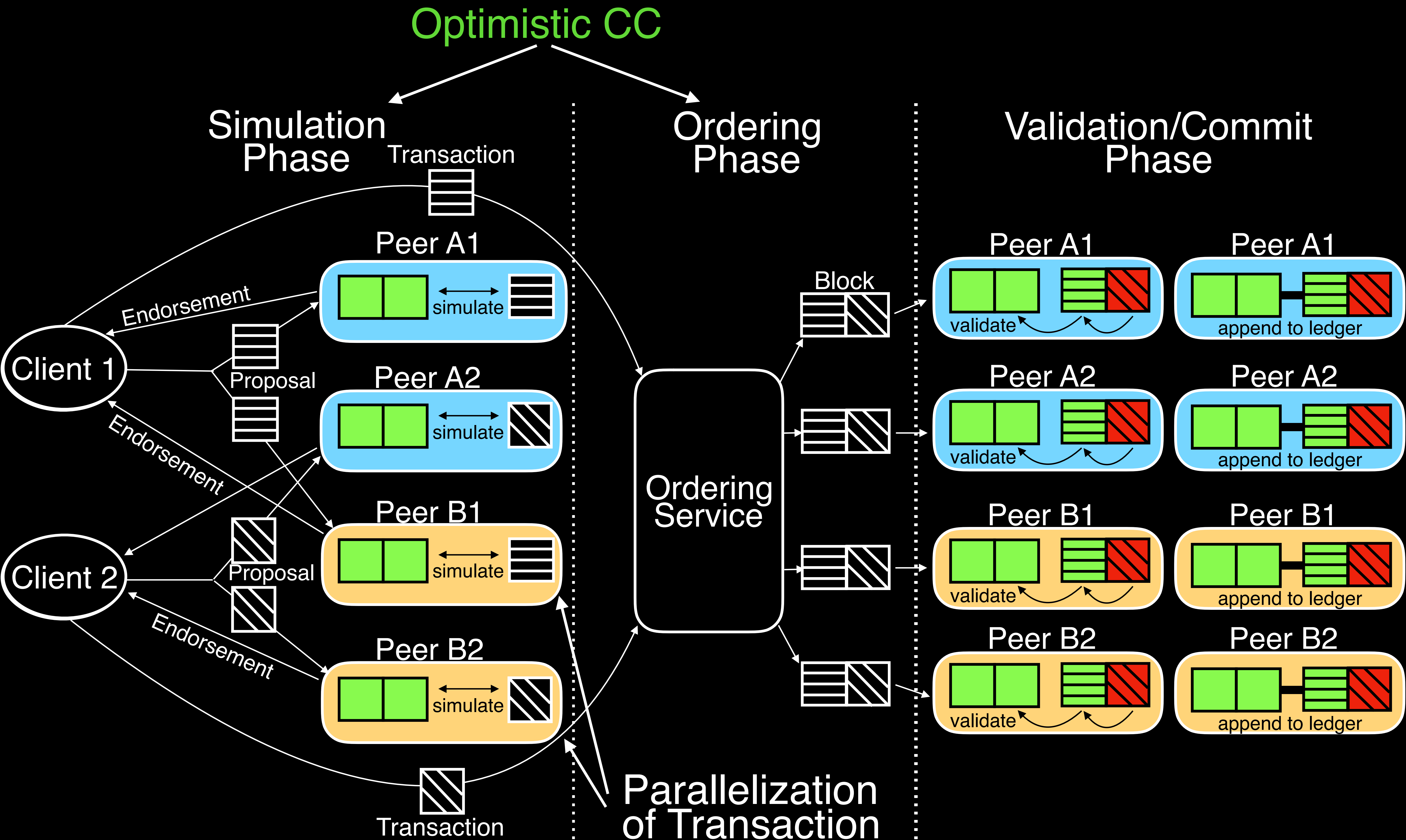


Parallelization of Transaction Simulation

→ Scaling!

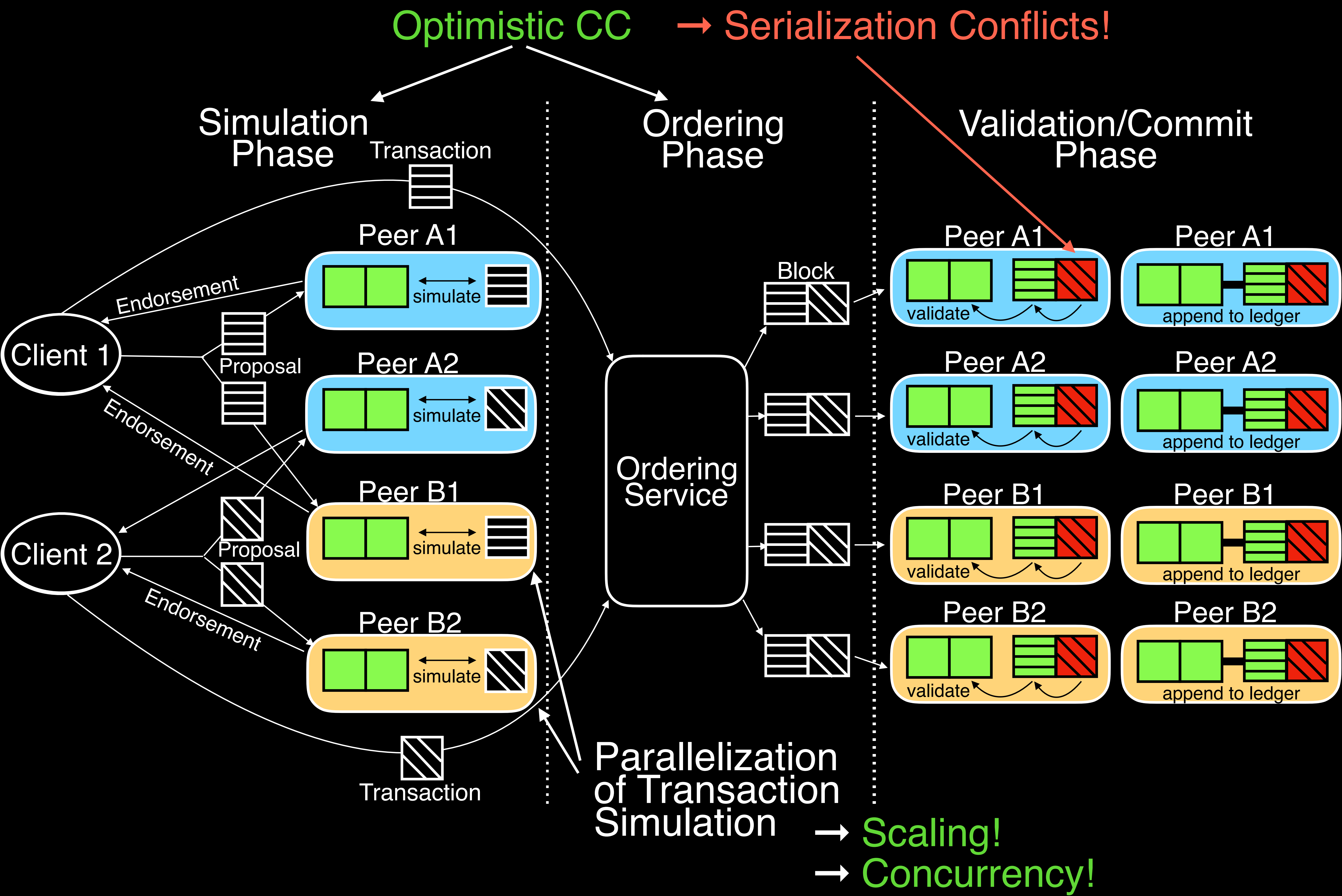
→ Concurrency!

The Simulate-order-validate-commit model (Fabric)

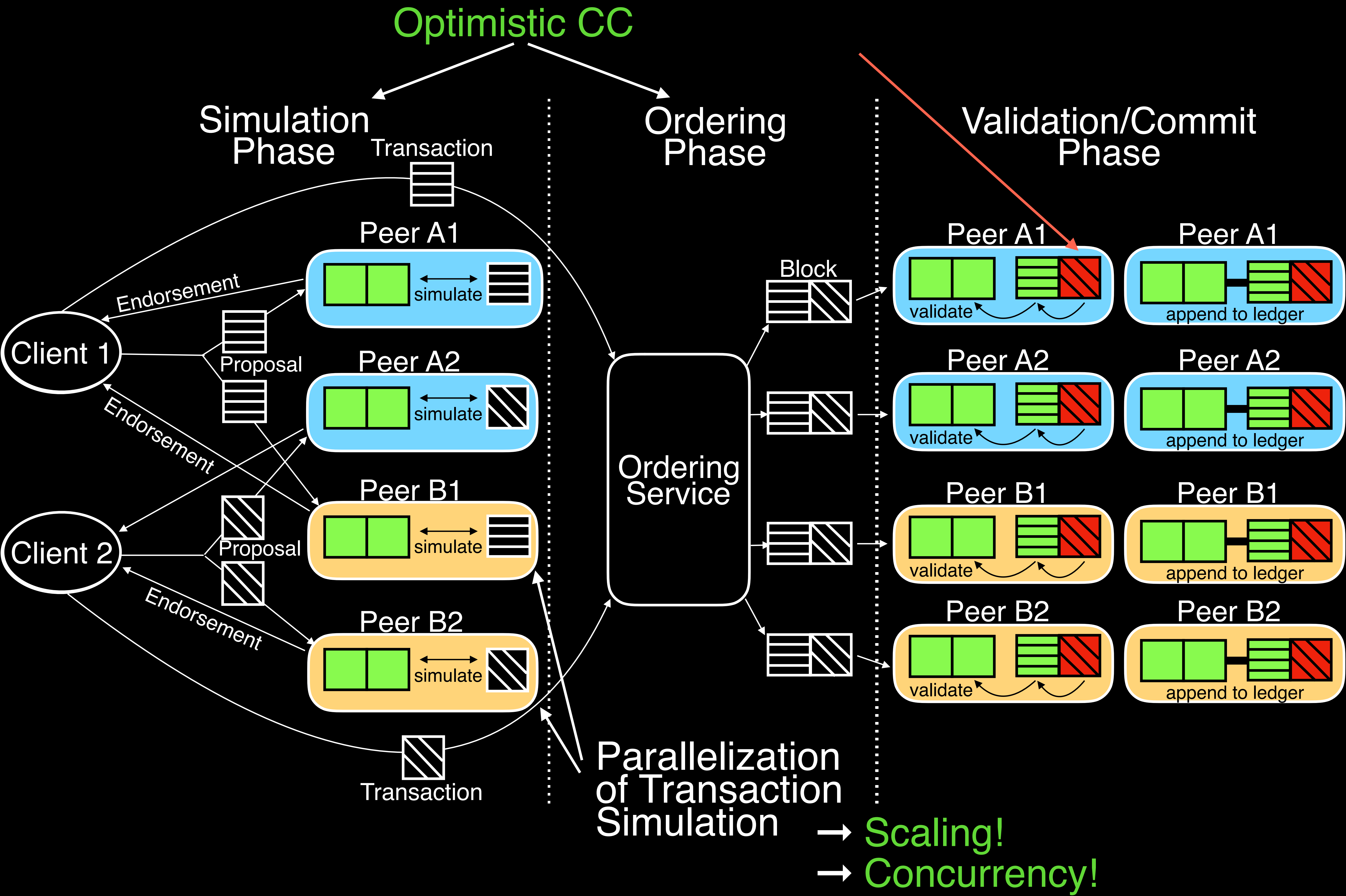


Parallelization of Transaction Simulation → Scaling!
→ Concurrency!

The Simulate-order-validate-commit model (Fabric)



The Simulate-order-validate-commit model (Fabric)

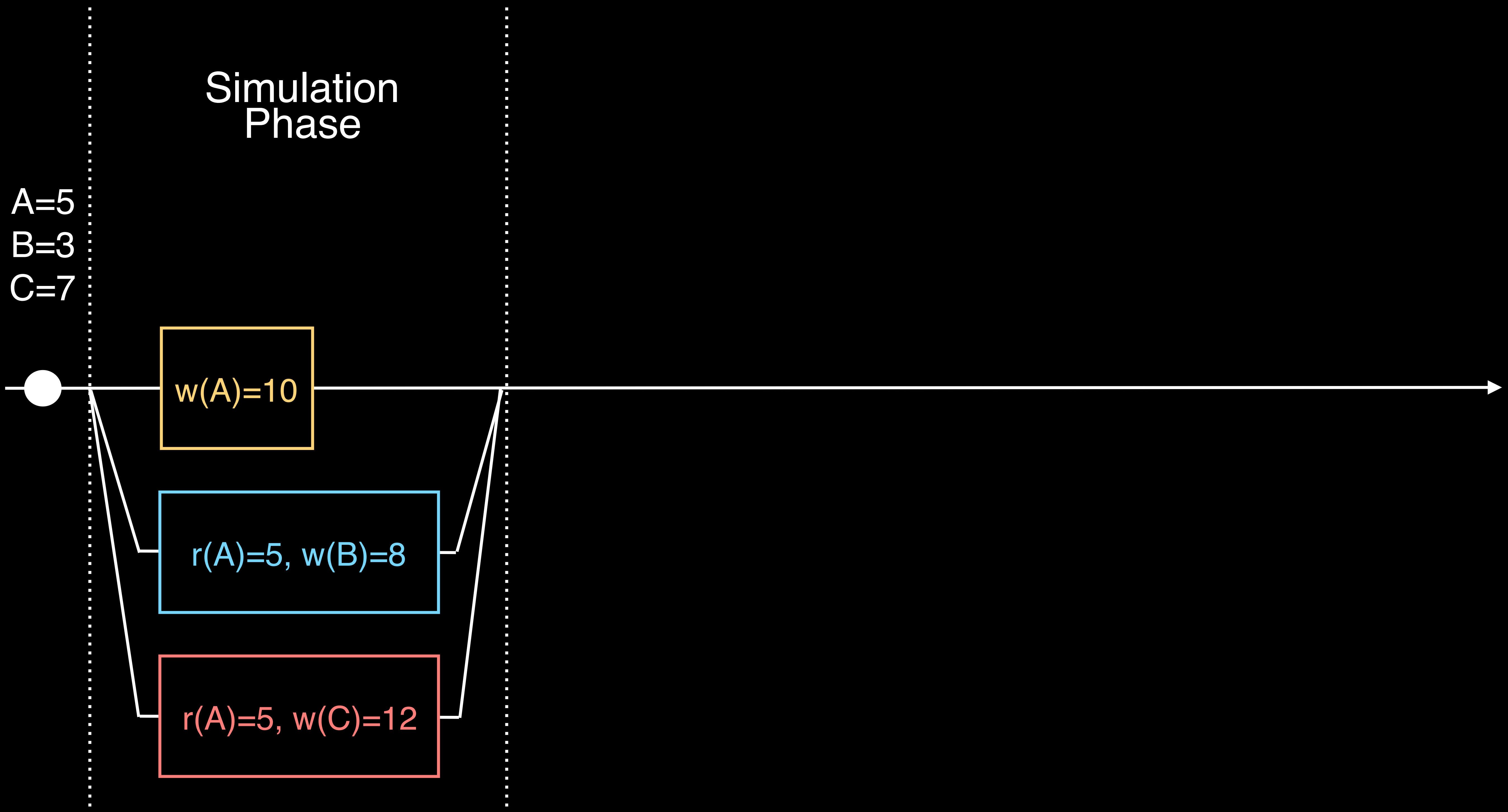


Serialization Conflicts

Serialization Conflicts

A=5
B=3
C=7

Serialization Conflicts



Simulation Phase

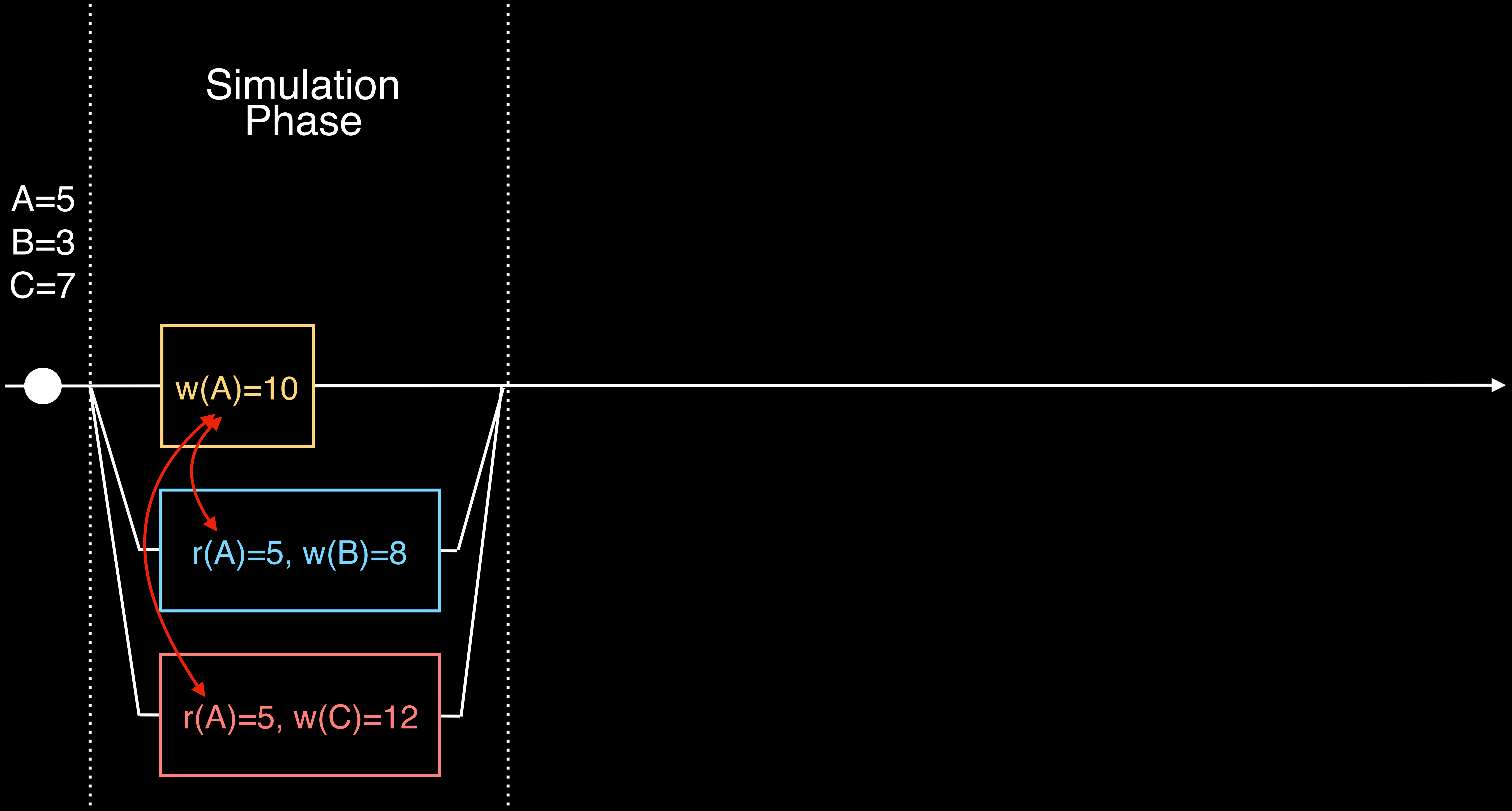
A=5
B=3
C=7

w(A)=10

r(A)=5, w(B)=8

r(A)=5, w(C)=12

Serialization Conflicts



Simulation Phase

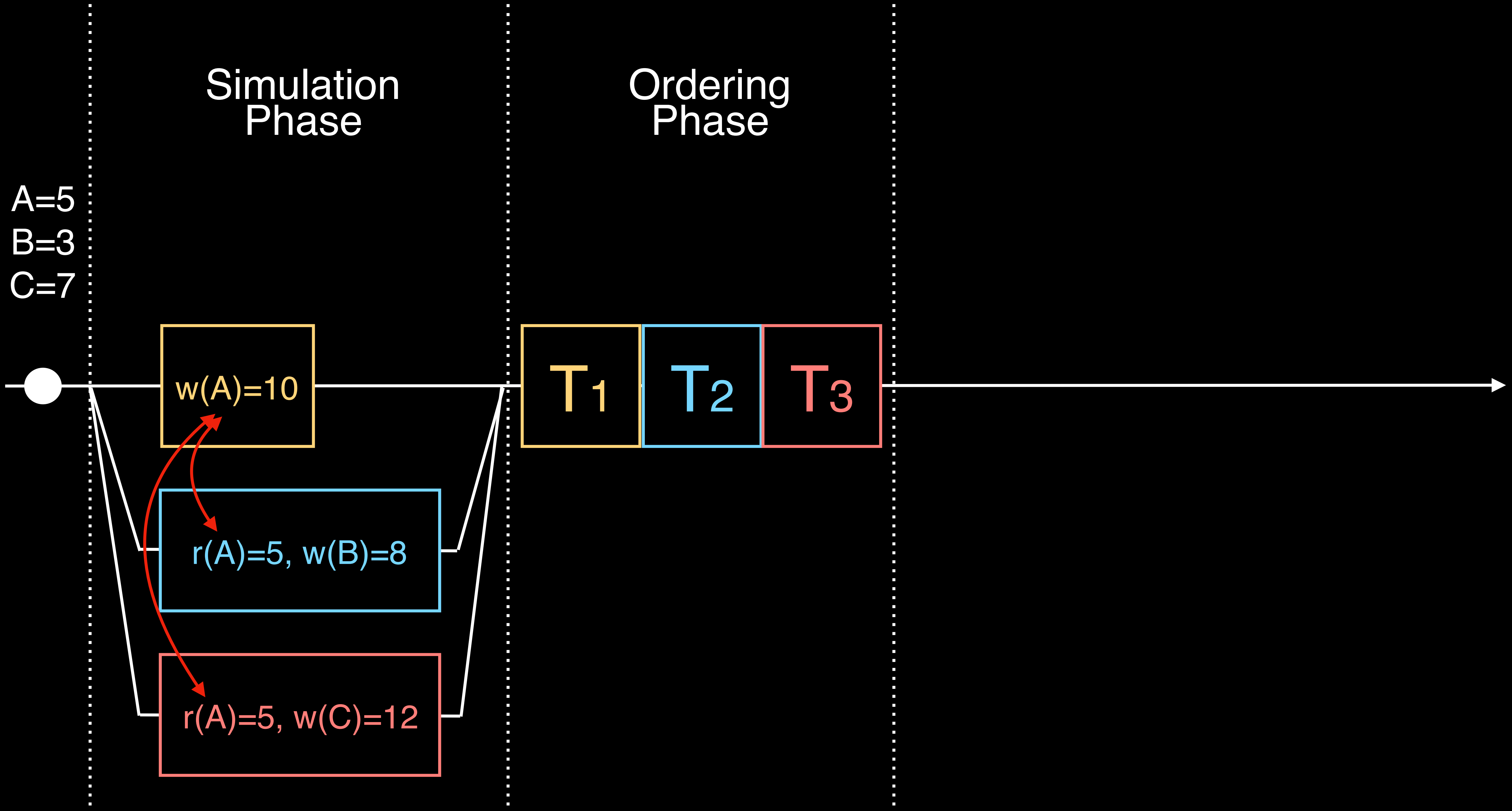
A=5
B=3
C=7

w(A)=10

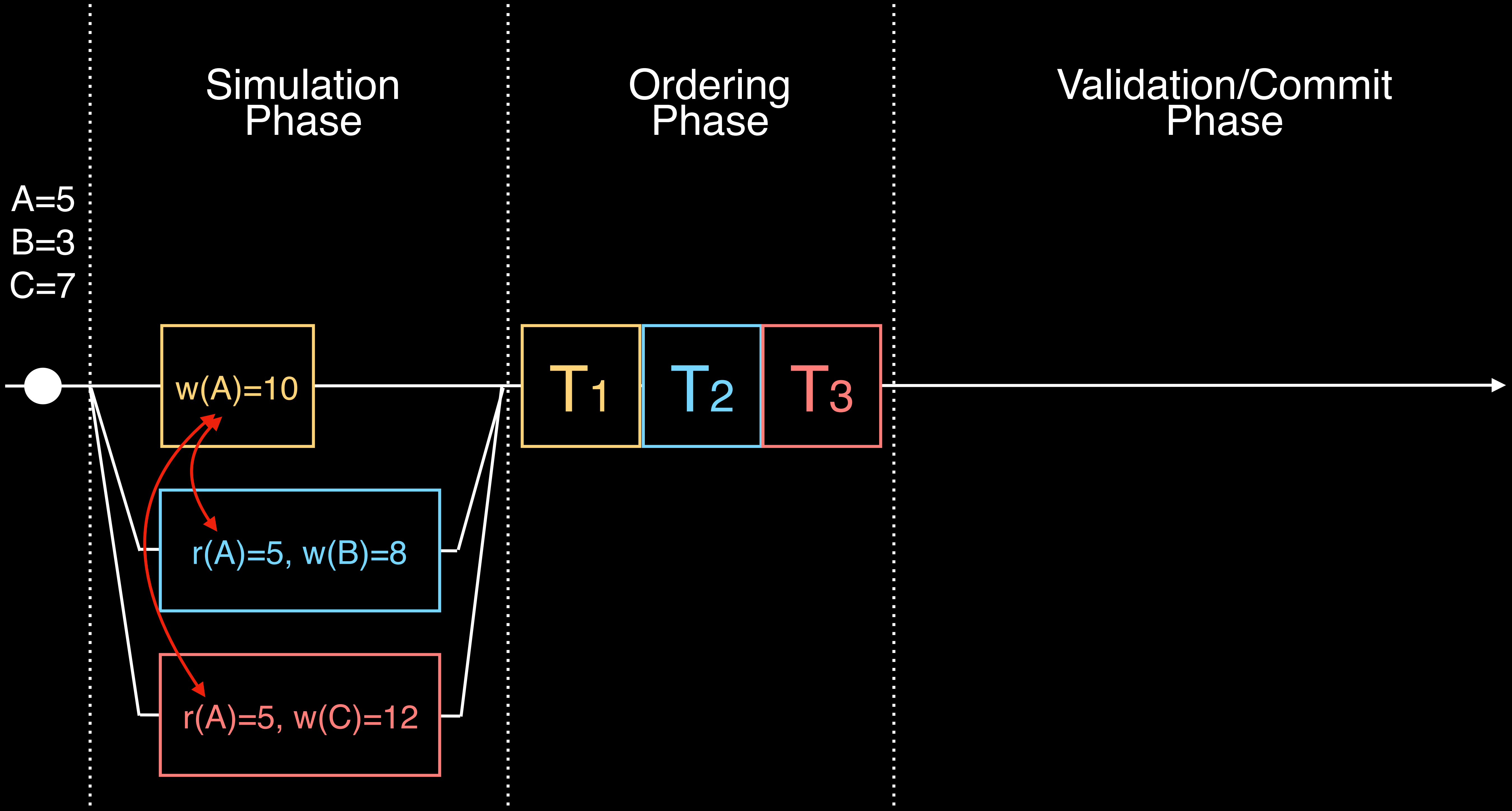
r(A)=5, w(B)=8

r(A)=5, w(C)=12

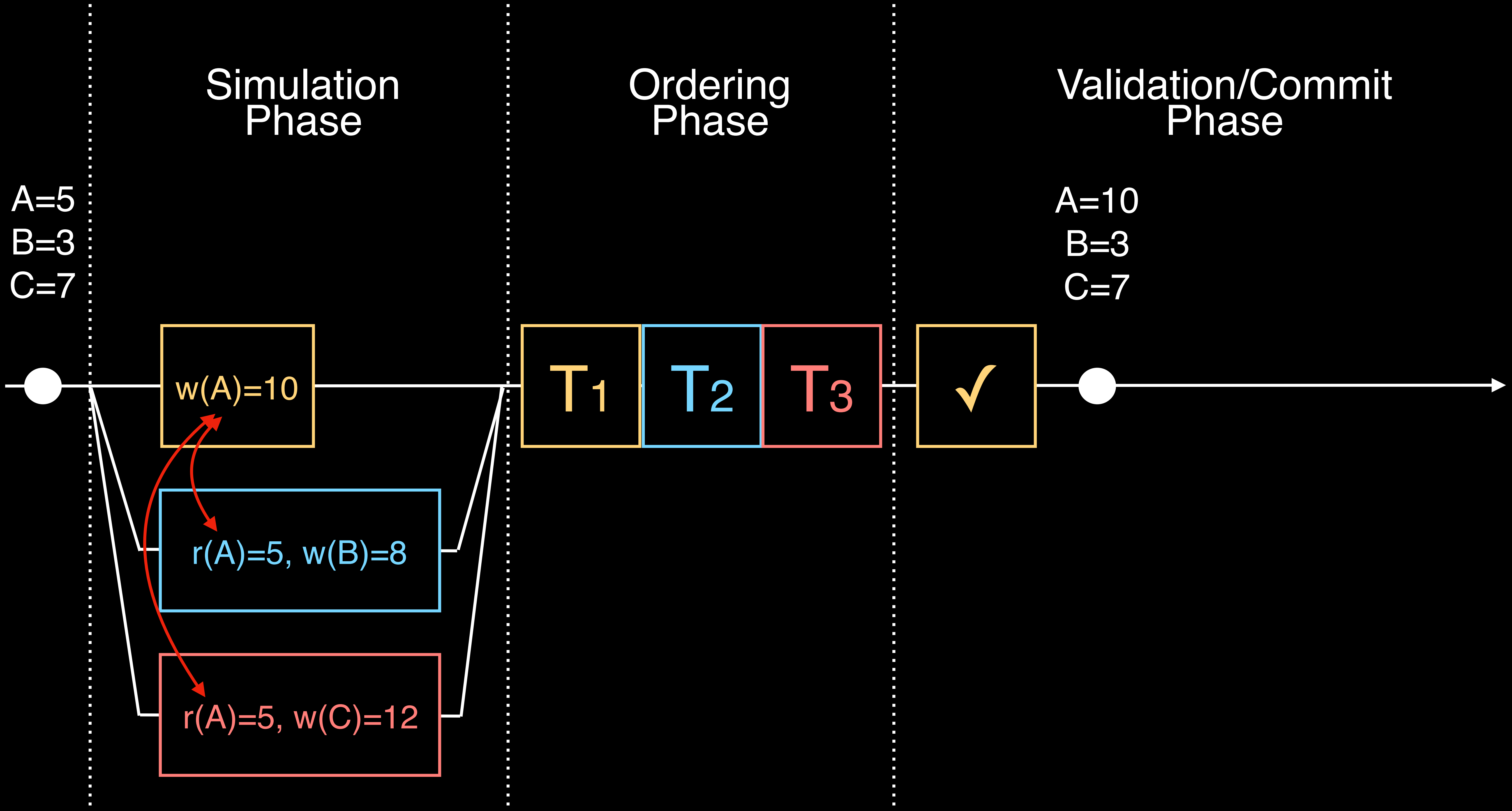
Serialization Conflicts



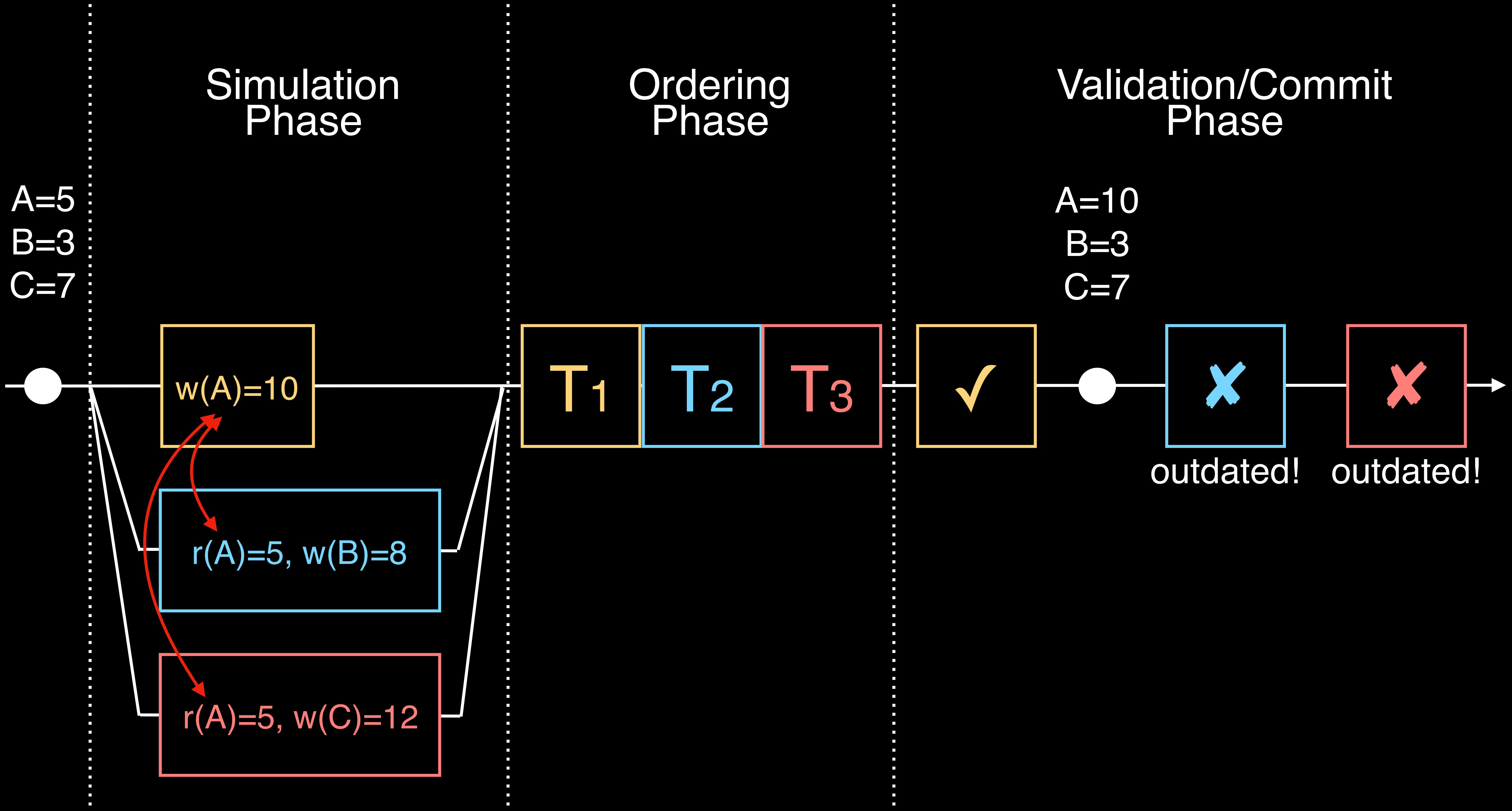
Serialization Conflicts



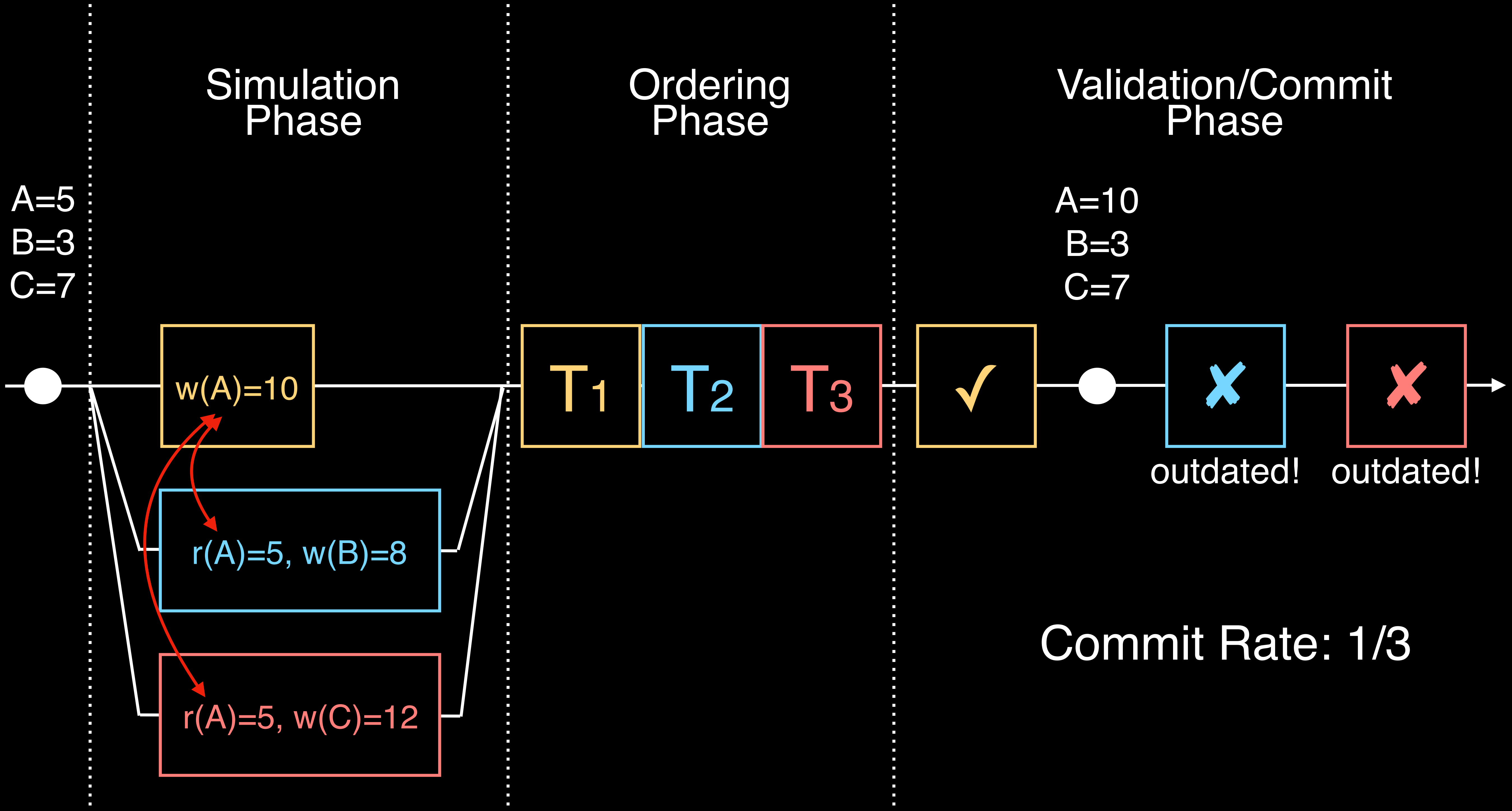
Serialization Conflicts



Serialization Conflicts

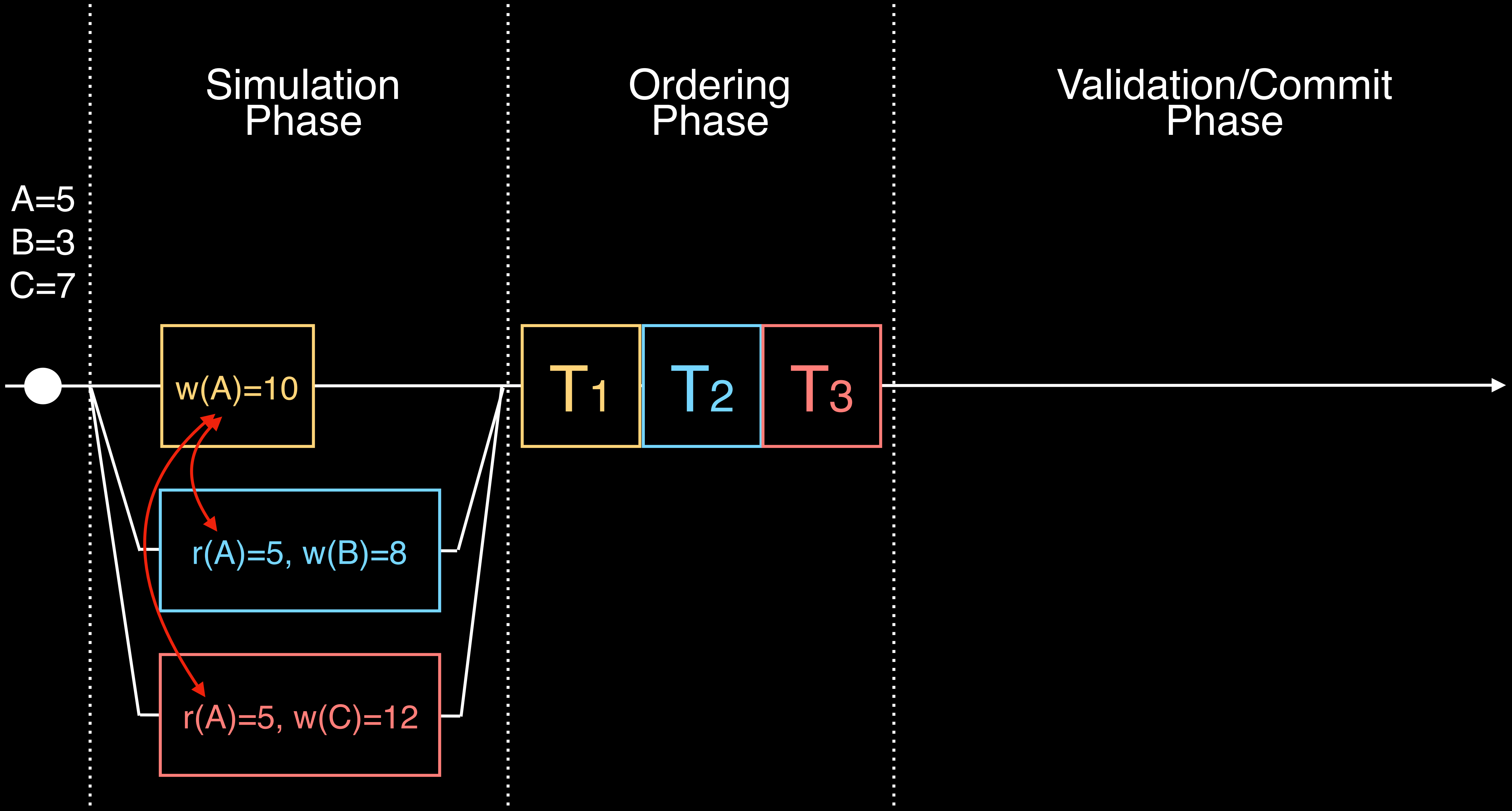


Serialization Conflicts

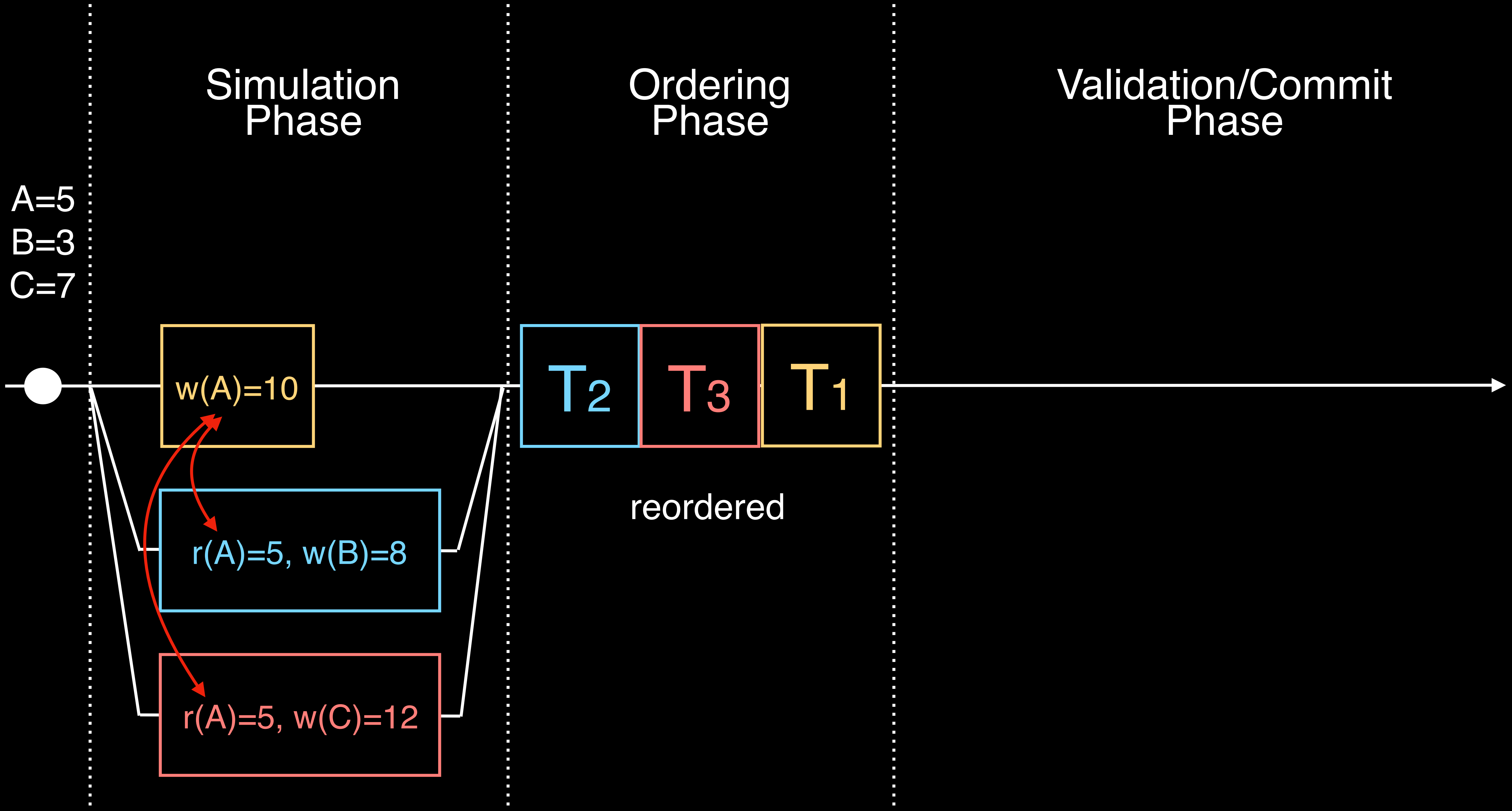


Commit Rate: 1/3

Serialization Conflicts



Serialization Conflicts



Simulation Phase

Ordering Phase

Validation/Commit Phase

A=5
B=3
C=7

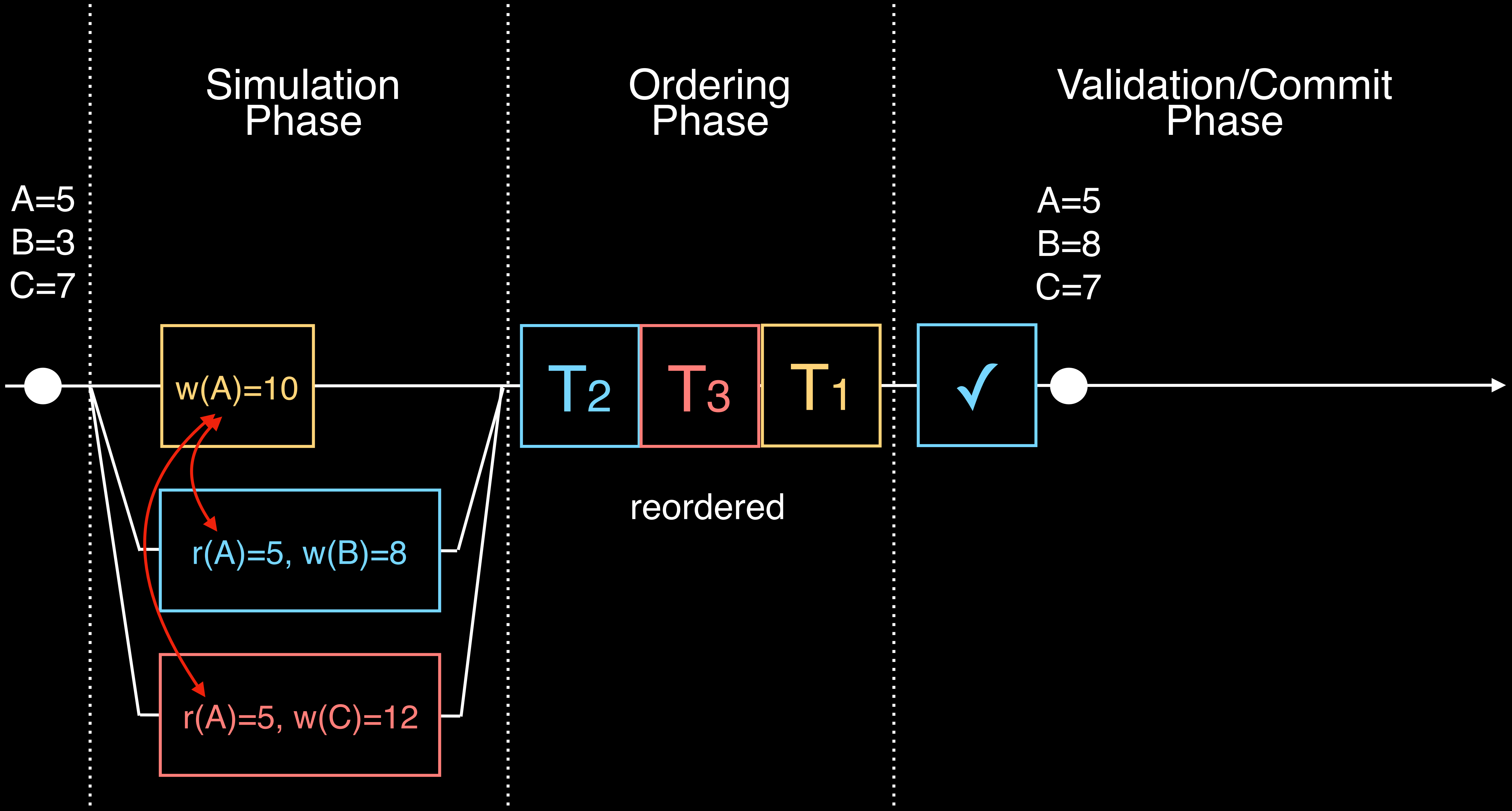
w(A)=10

r(A)=5, w(B)=8

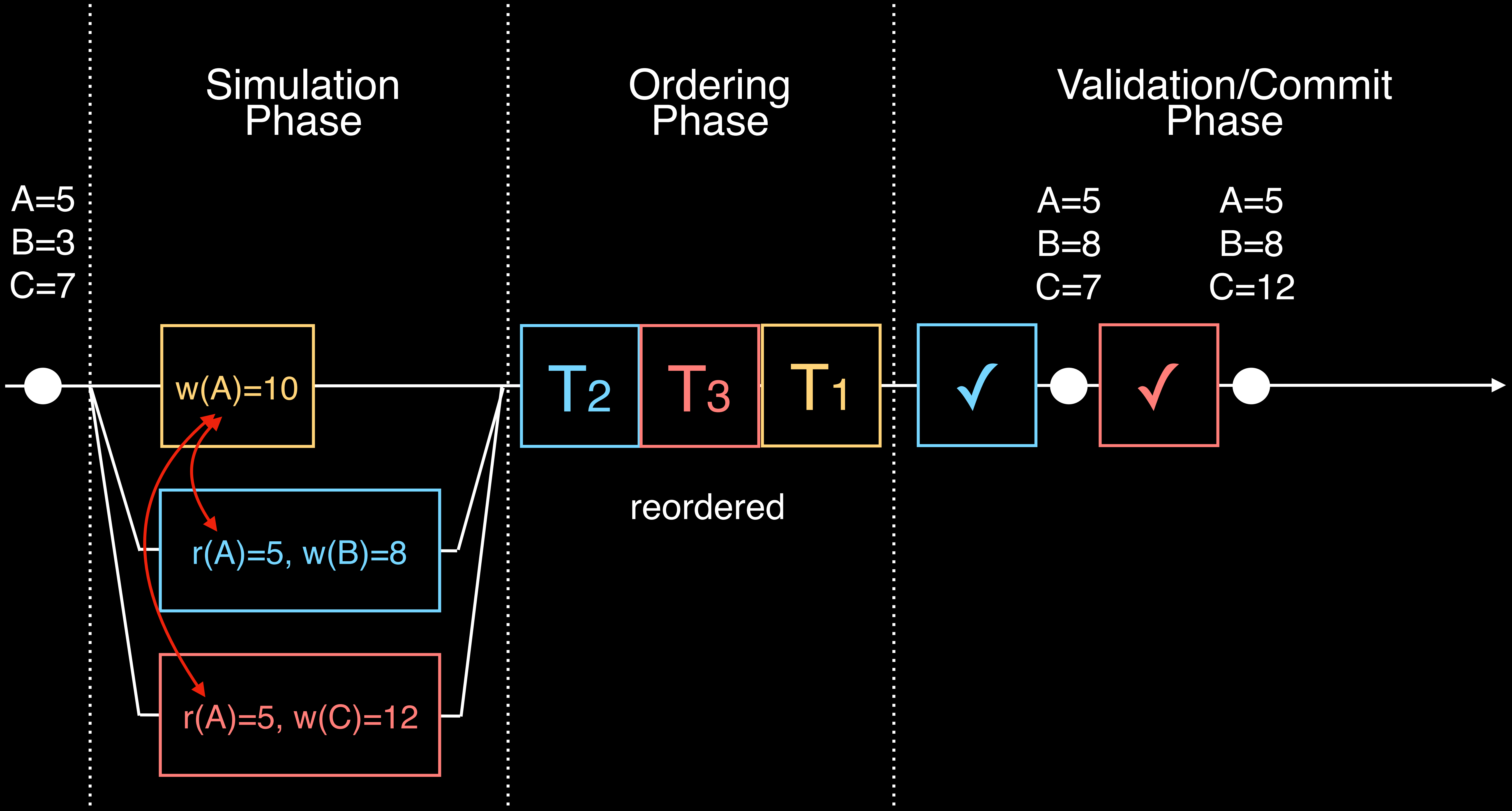
r(A)=5, w(C)=12

reordered

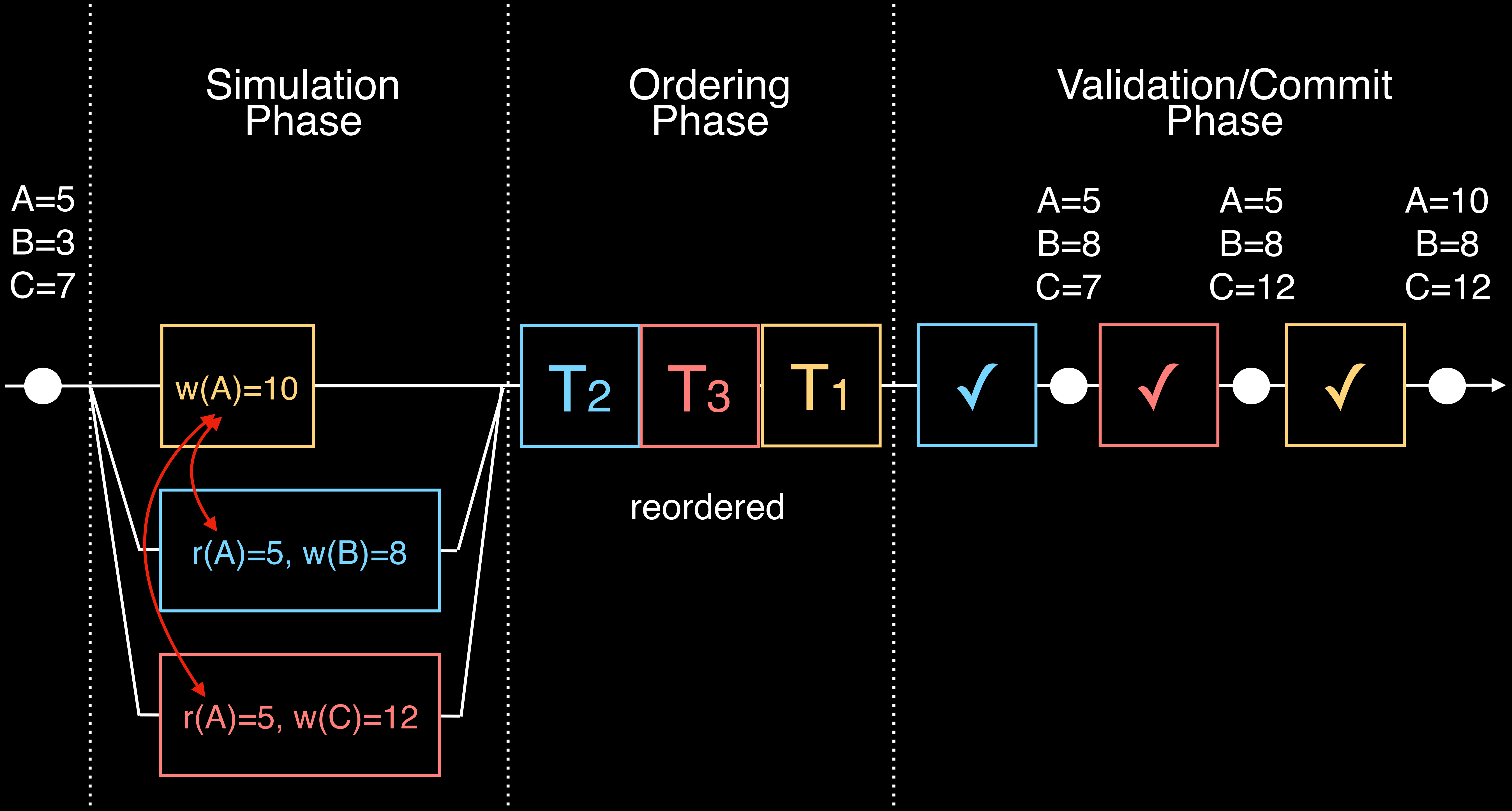
Serialization Conflicts



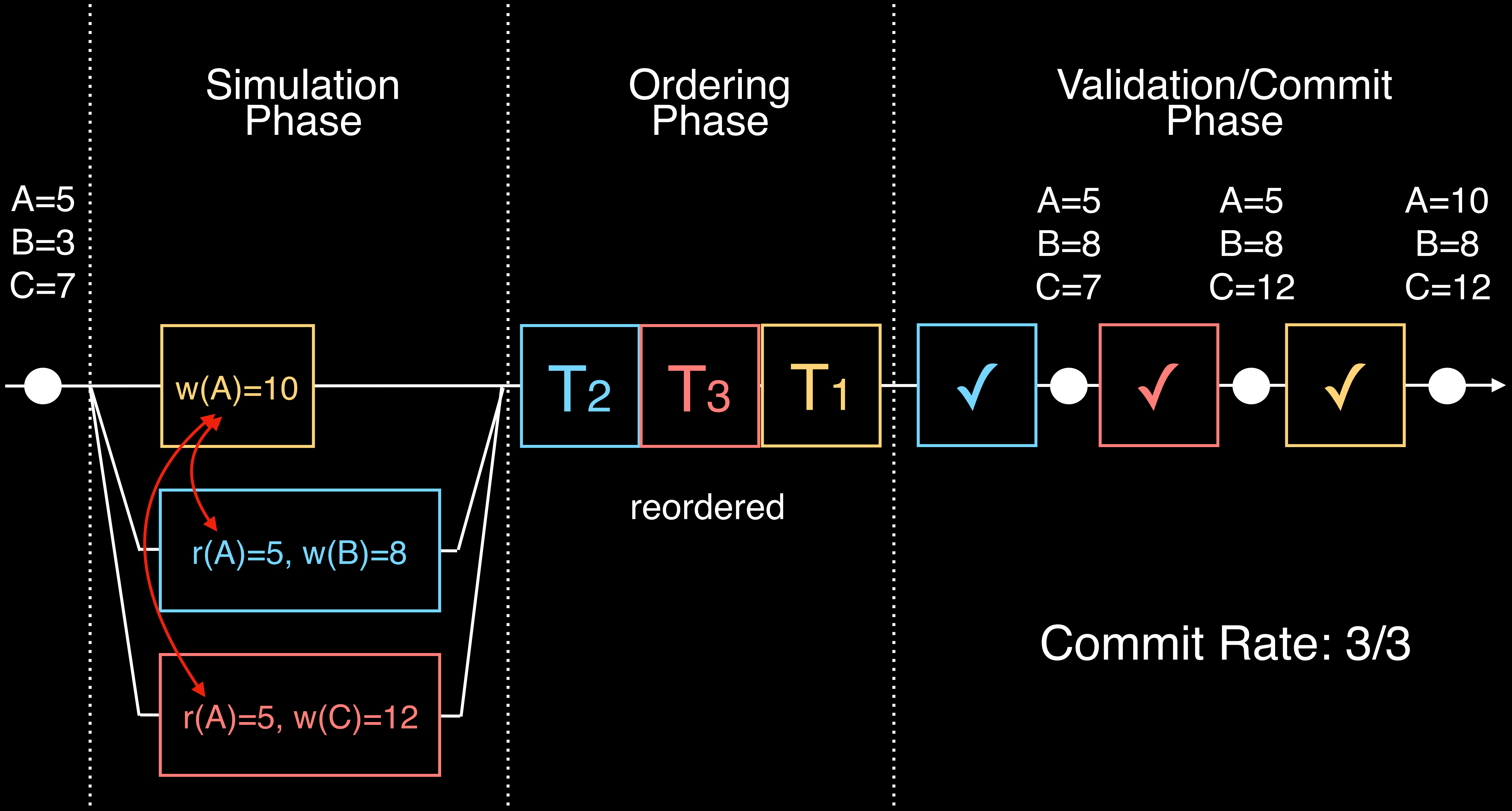
Serialization Conflicts



Serialization Conflicts



Serialization Conflicts



Commit Rate: 3/3

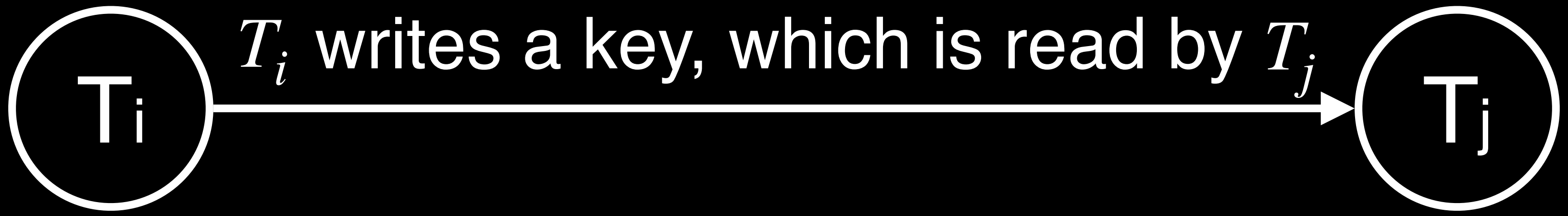
Fabric++: Reordering of Transactions

Fabric++: Reordering of Transactions

1. build conflict graph:

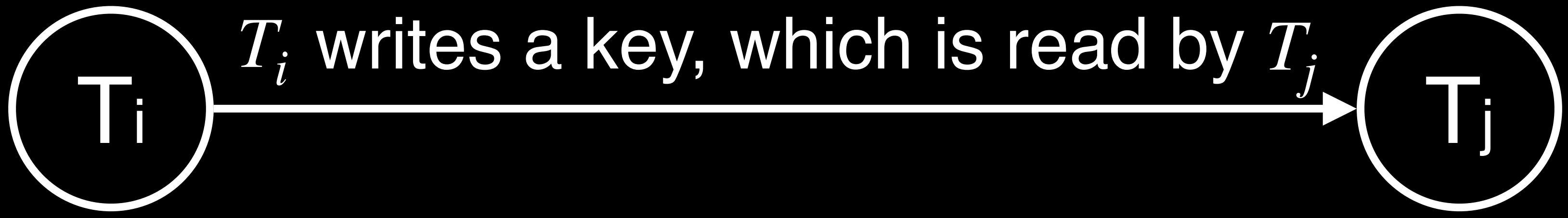
Fabric++: Reordering of Transactions

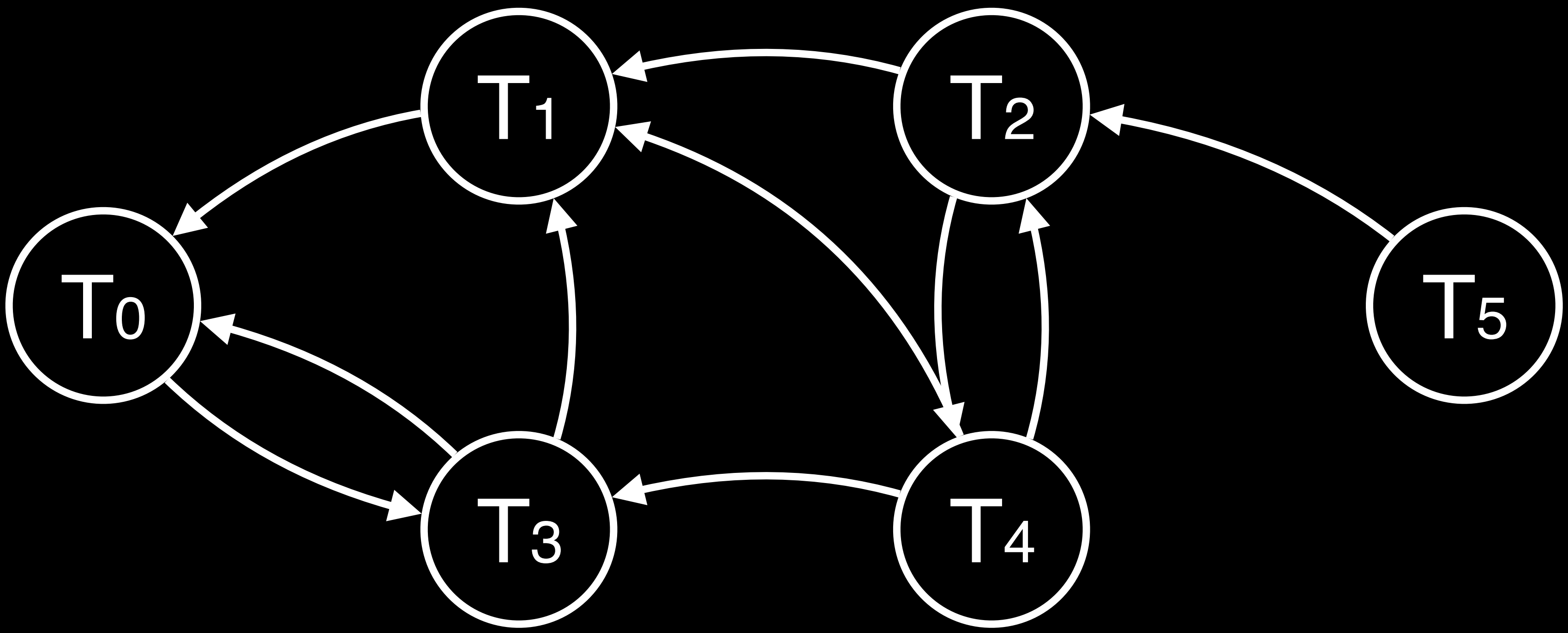
1. build conflict graph:



Fabric++: Reordering of Transactions

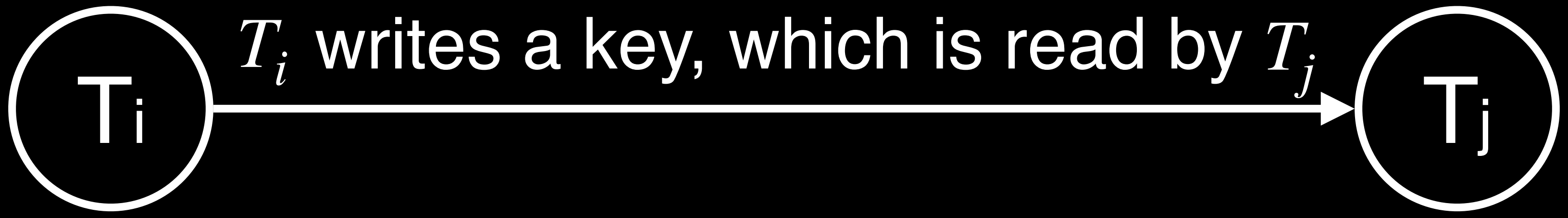
1. build conflict graph:

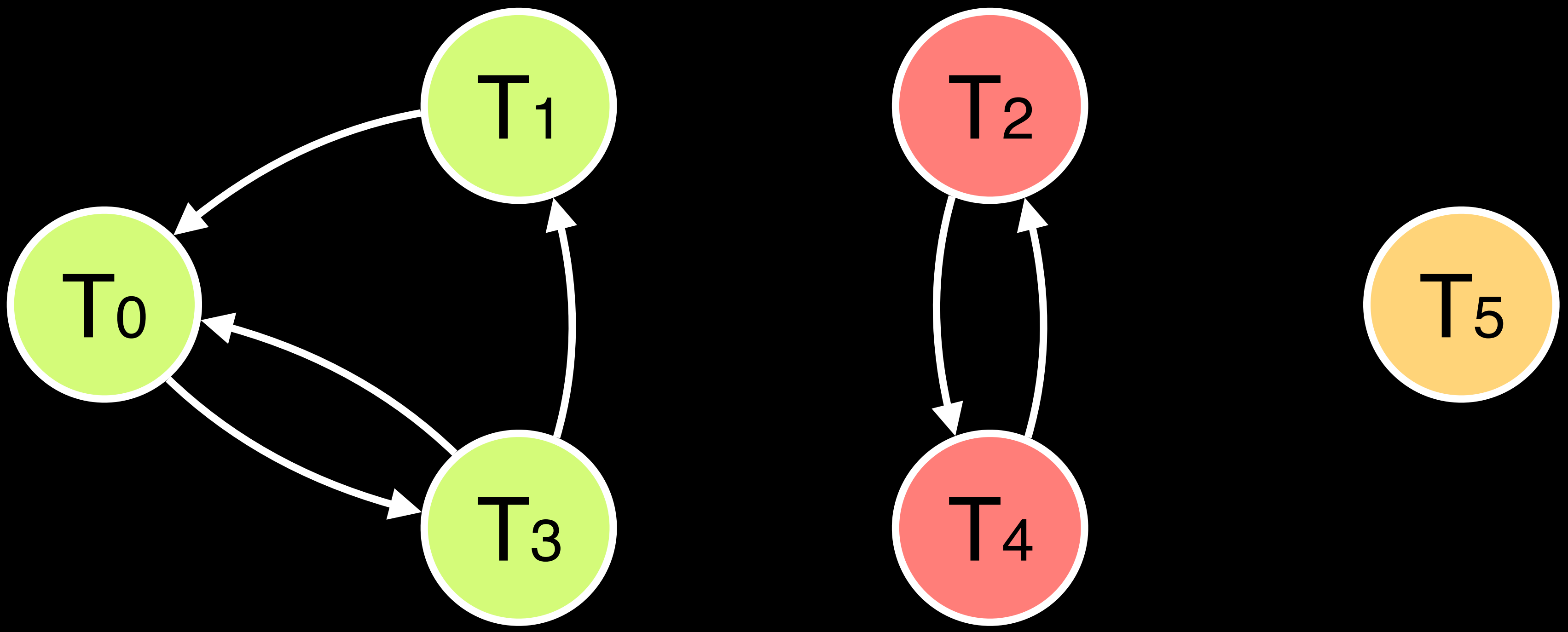




Fabric++: Reordering of Transactions

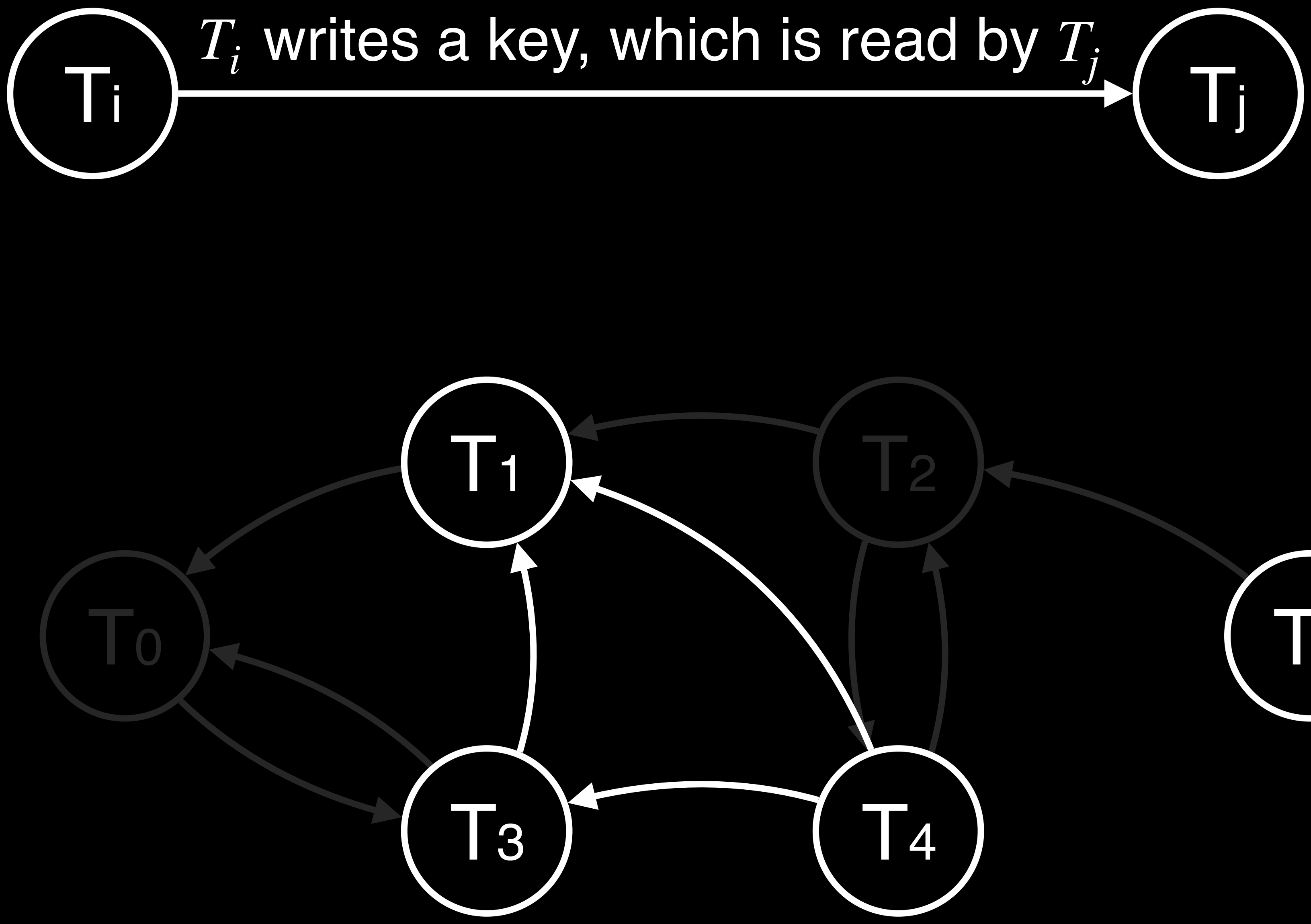
2. compute strongly connected subgraphs:





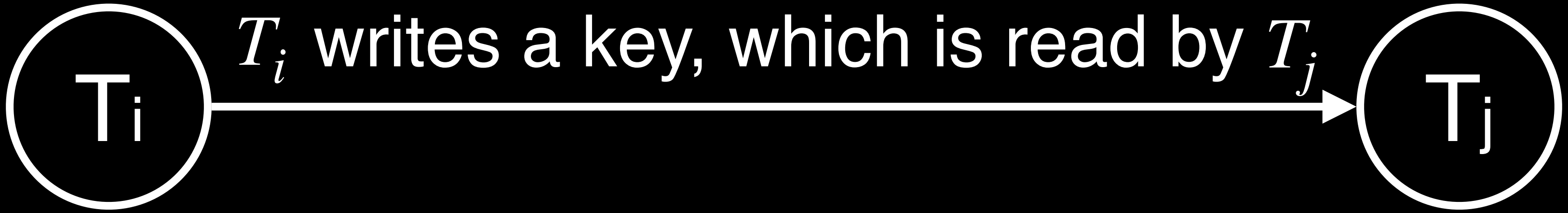
Fabric++: Reordering of Transactions

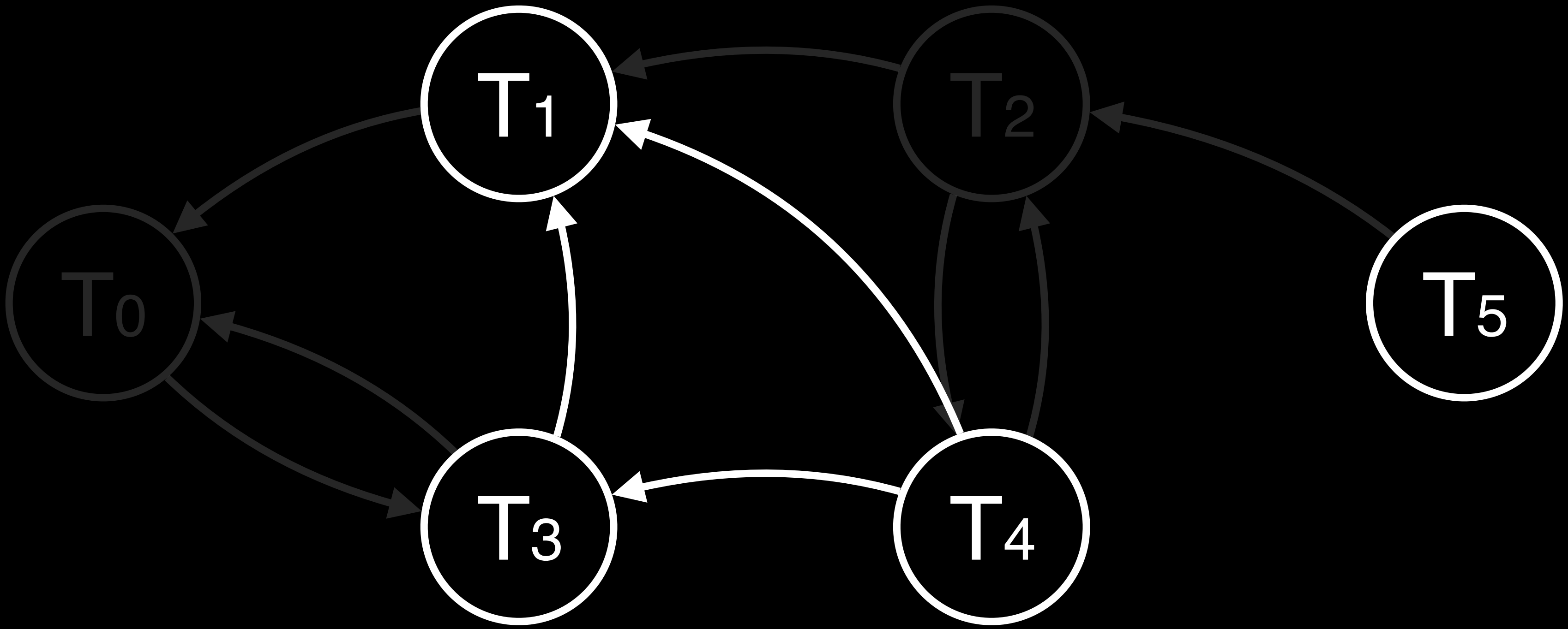
3. compute cycle-free conflicts graph:



Fabric++: Reordering of Transactions

3. compute cycle-free conflicts graph:



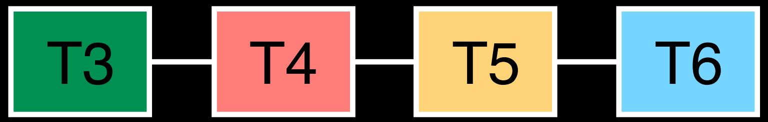


4. compute schedule: $T_5 \Rightarrow T_1 \Rightarrow T_3 \Rightarrow T_4$

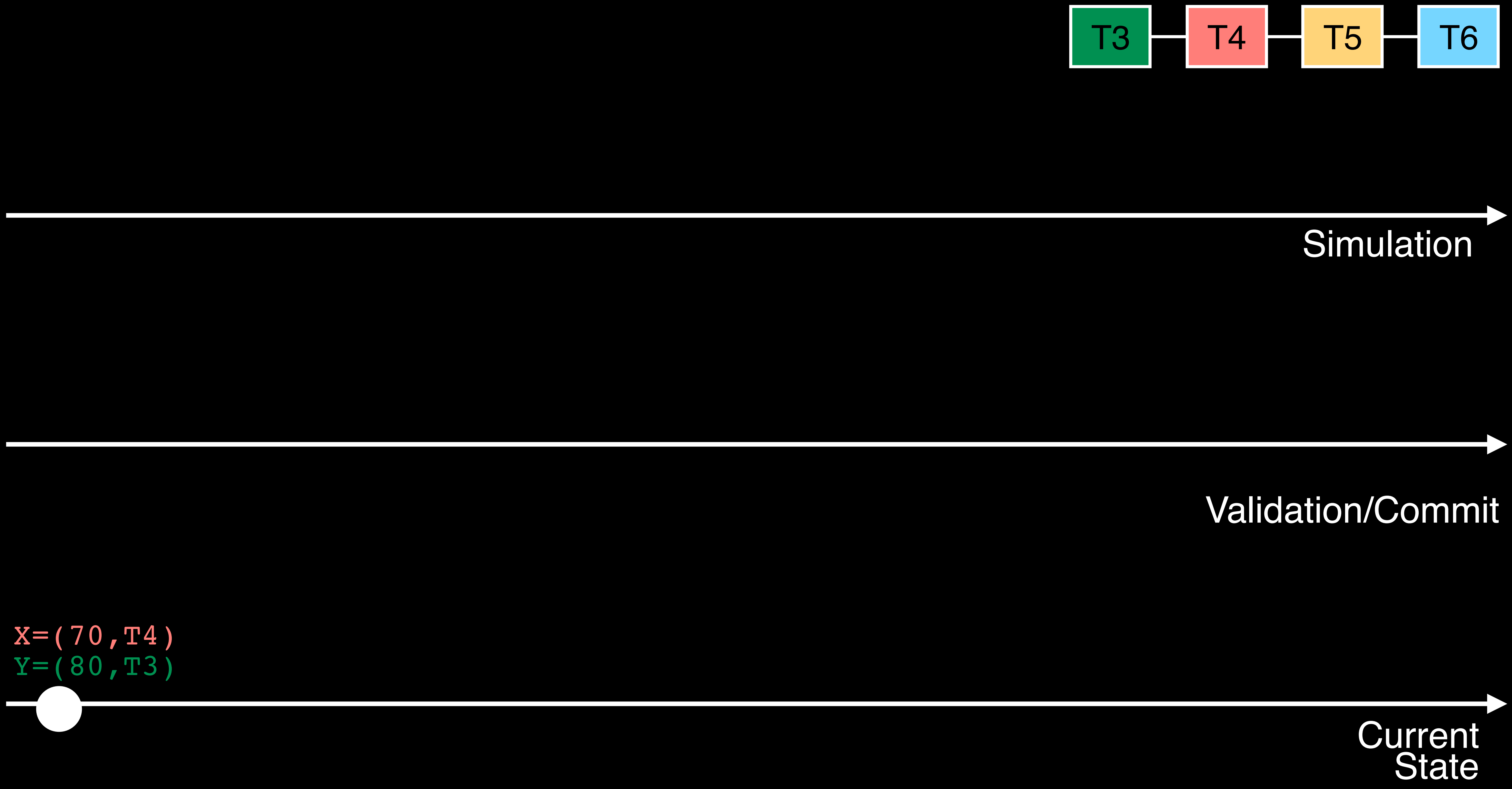
T_0 and T_2 aborted

Fabric: Lock-based Concurrency Control

Fabric: Lock-based Concurrency Control



Fabric: Lock-based Concurrency Control



T3 T4 T5 T6

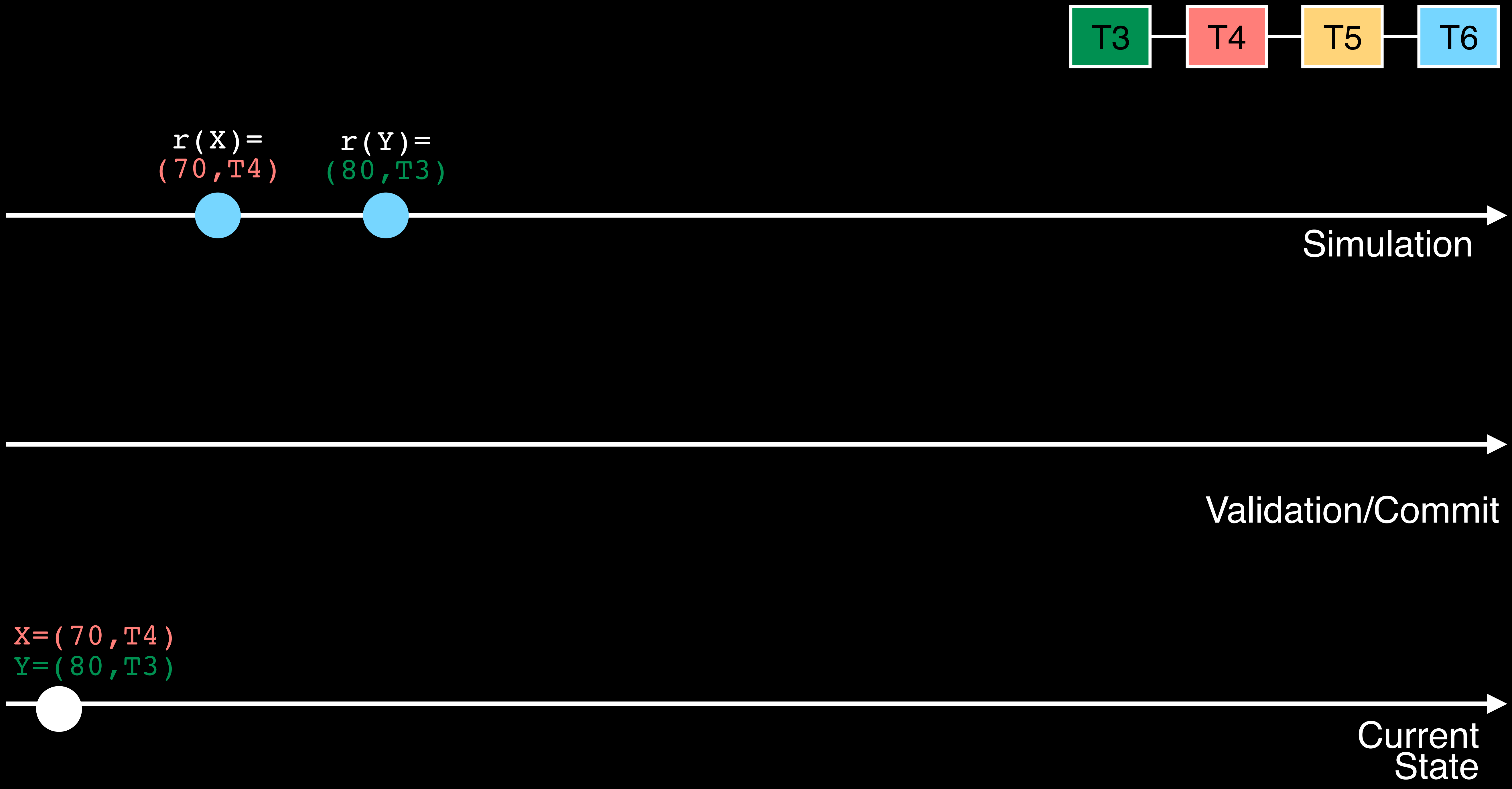
Simulation

Validation/Commit

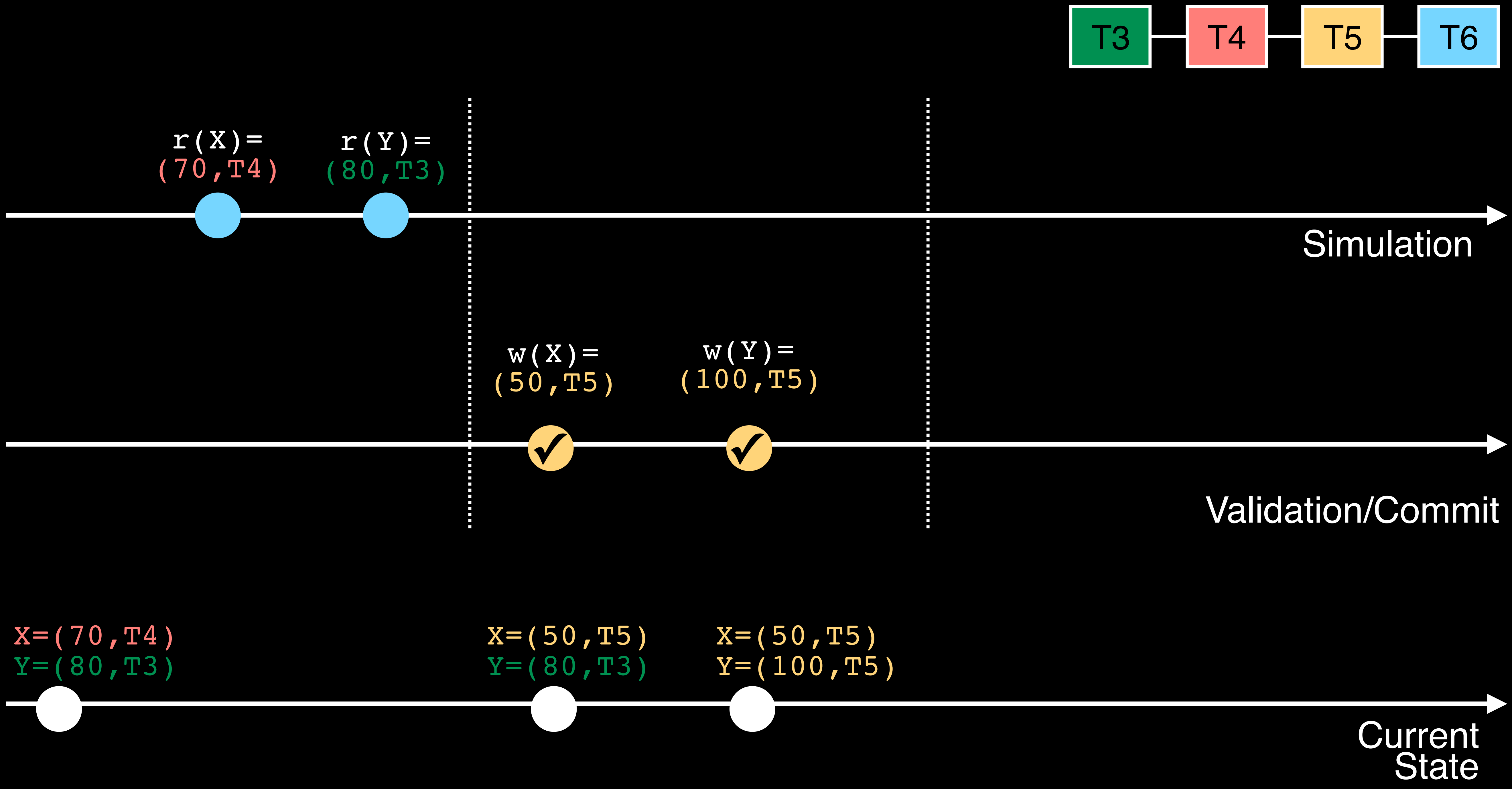
```
X=(70,T4)
Y=(80,T3)
```

Current State

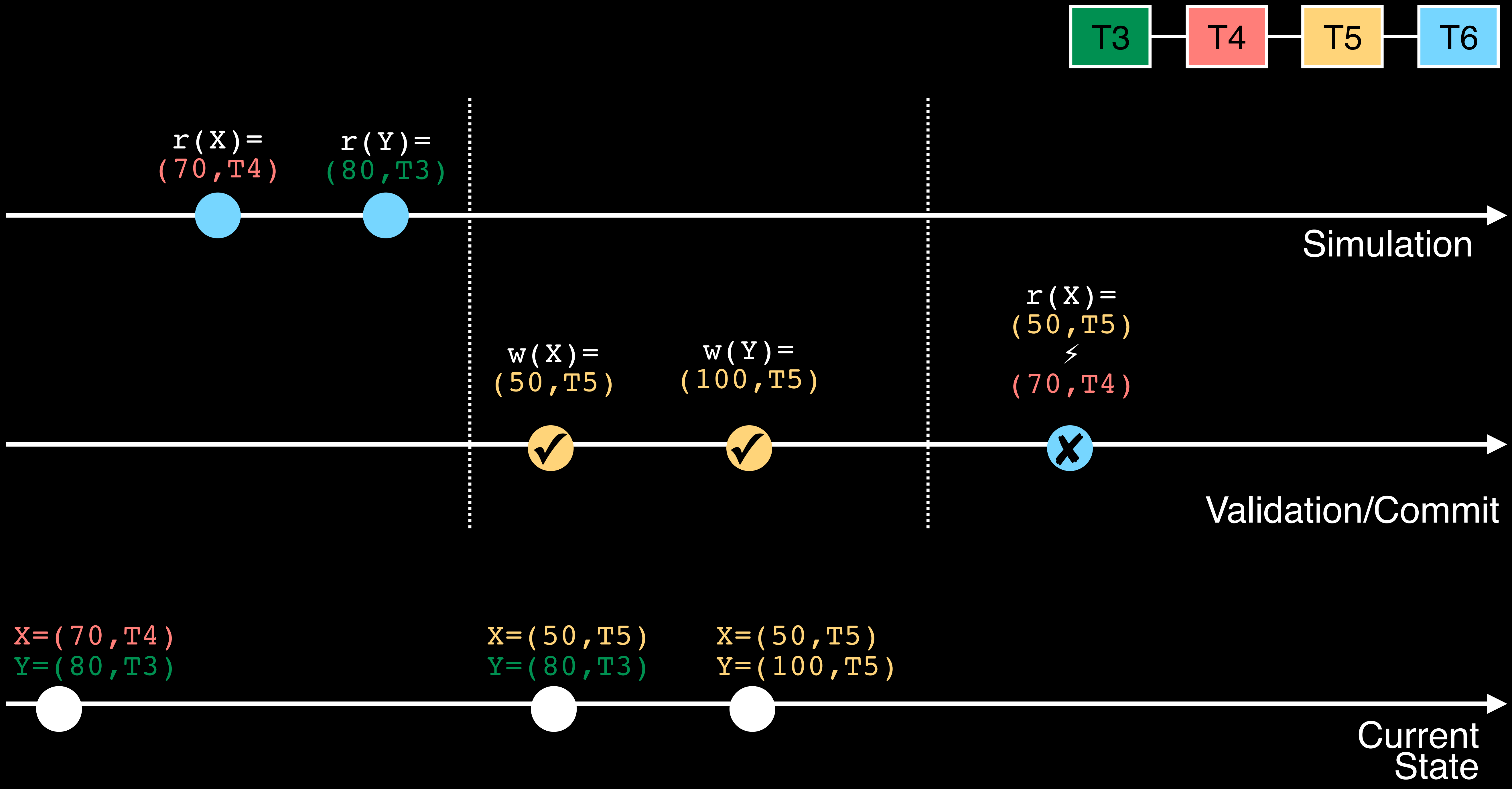
Fabric: Lock-based Concurrency Control



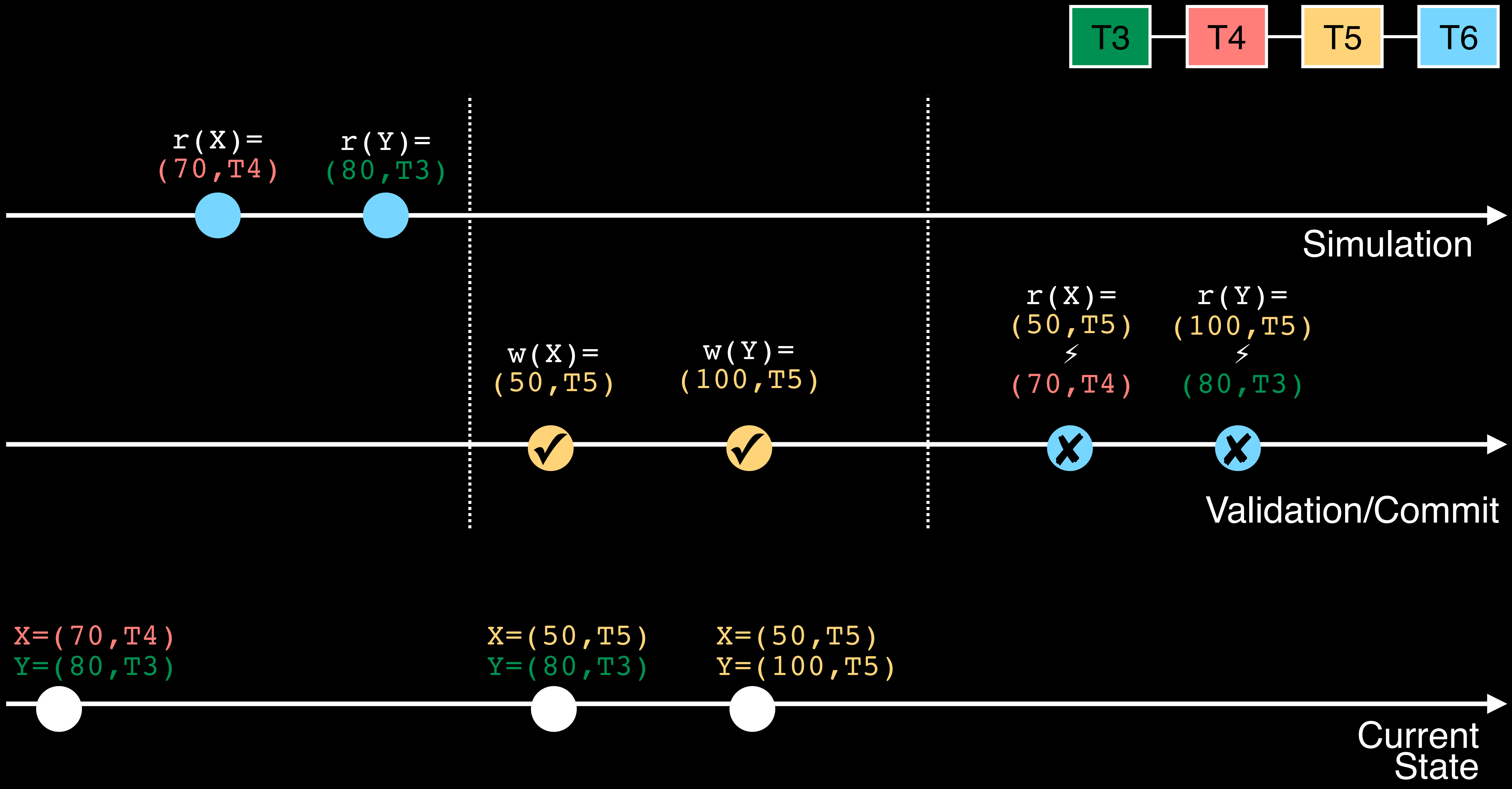
Fabric: Lock-based Concurrency Control



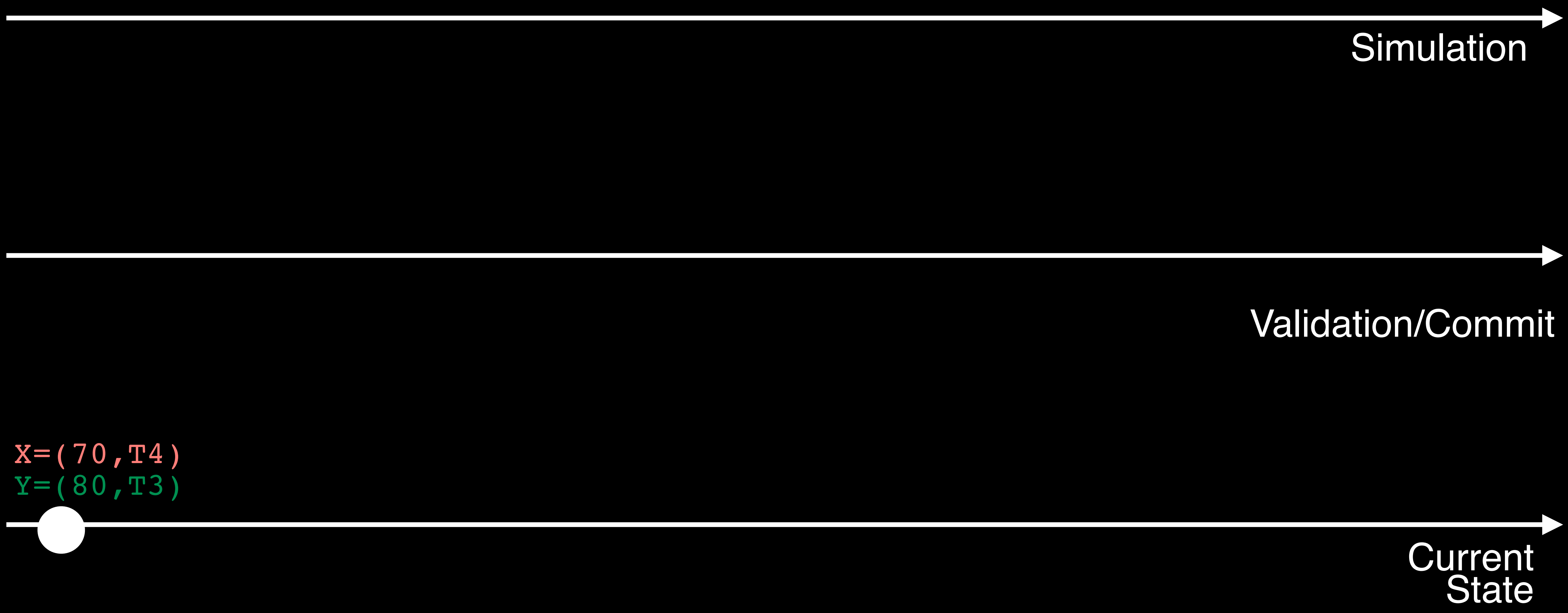
Fabric: Lock-based Concurrency Control



Fabric: Lock-based Concurrency Control



Fabric++: Multi-version Concurrency Control



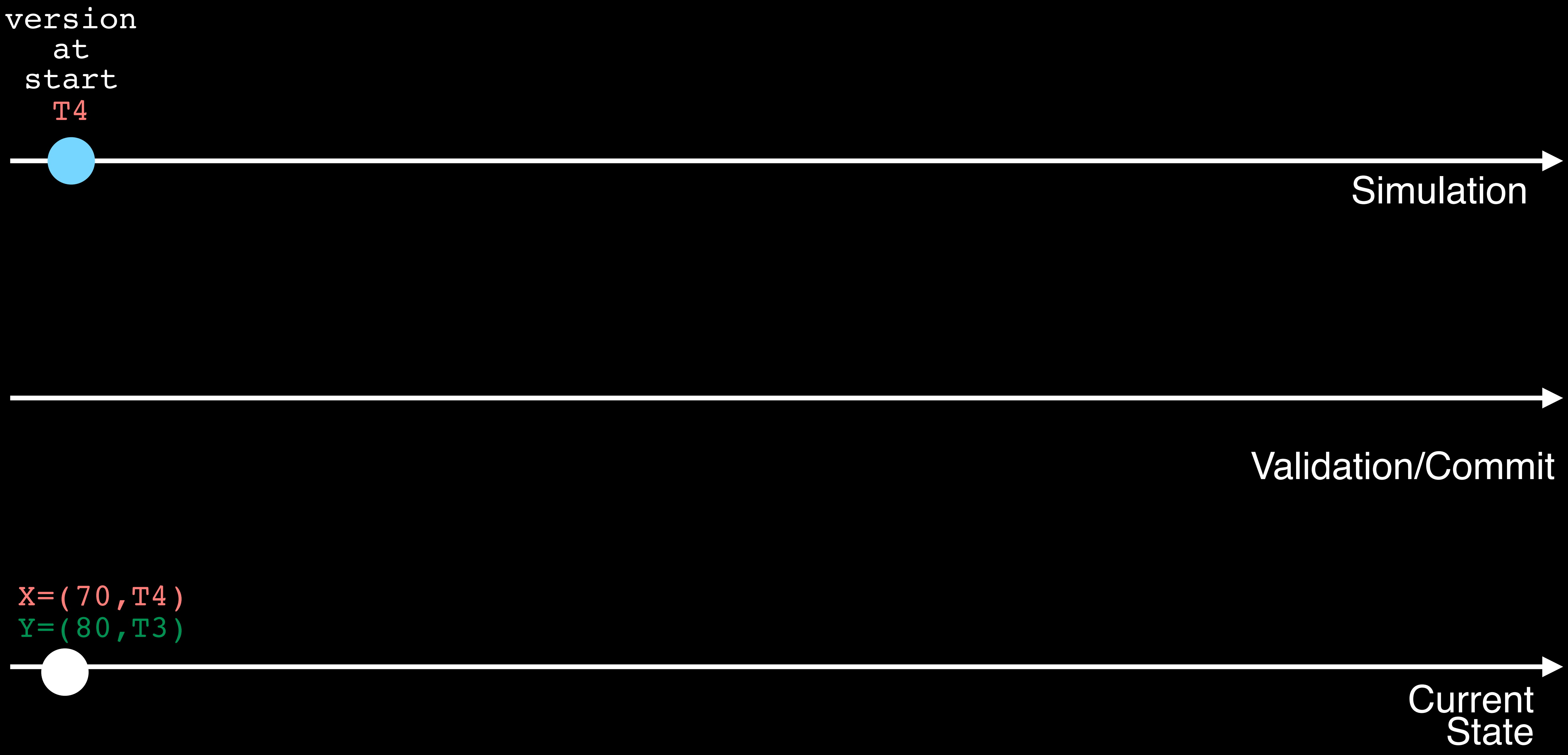
Simulation

Validation/Commit

X=(70,T4)
Y=(80,T3)

Current
State

Fabric++: Multi-version Concurrency Control



version
at
start
T4

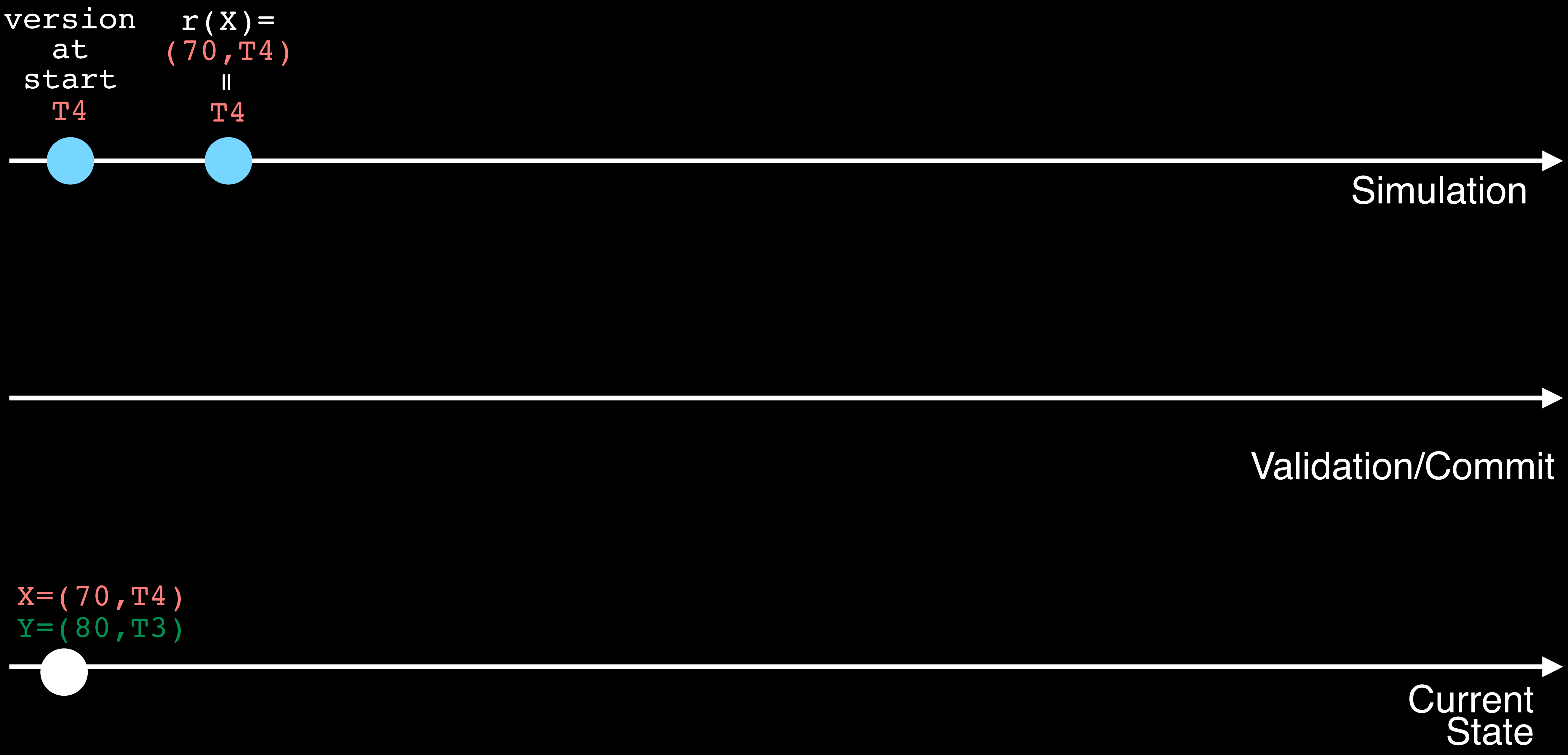
Simulation

Validation/Commit

X=(70,T4)
Y=(80,T3)

Current
State

Fabric++: Multi-version Concurrency Control



version at start T4

r(X)= (70,T4) = T4

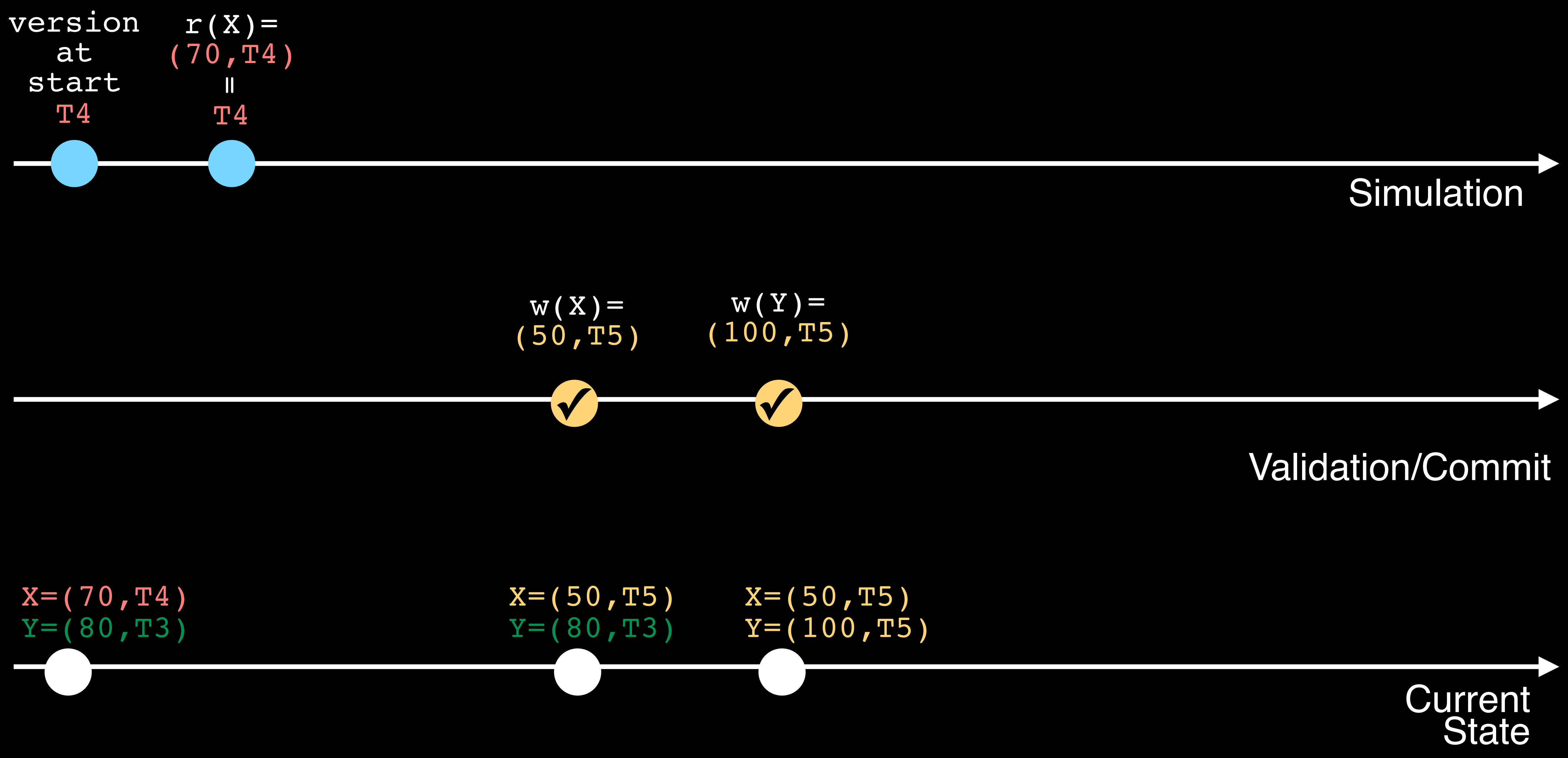
Simulation

Validation/Commit

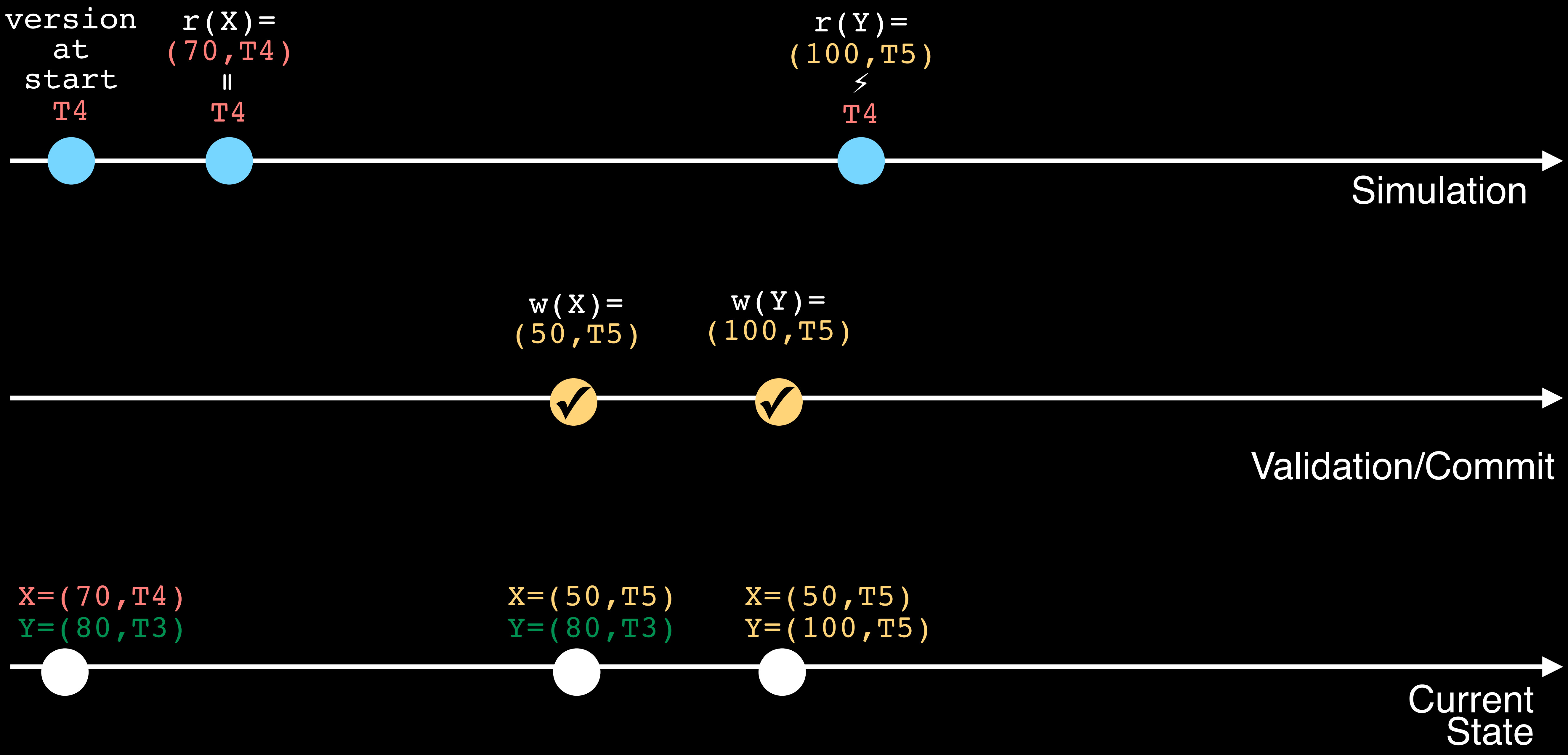
X=(70,T4)
Y=(80,T3)

Current State

Fabric++: Multi-version Concurrency Control



Fabric++: Multi-version Concurrency Control



version at start T4

r(X)= (70,T4) ‖ T4

r(Y)= (100,T5) ⚡ T4

Simulation

w(X)= (50,T5) ✓

w(Y)= (100,T5) ✓

Validation/Commit

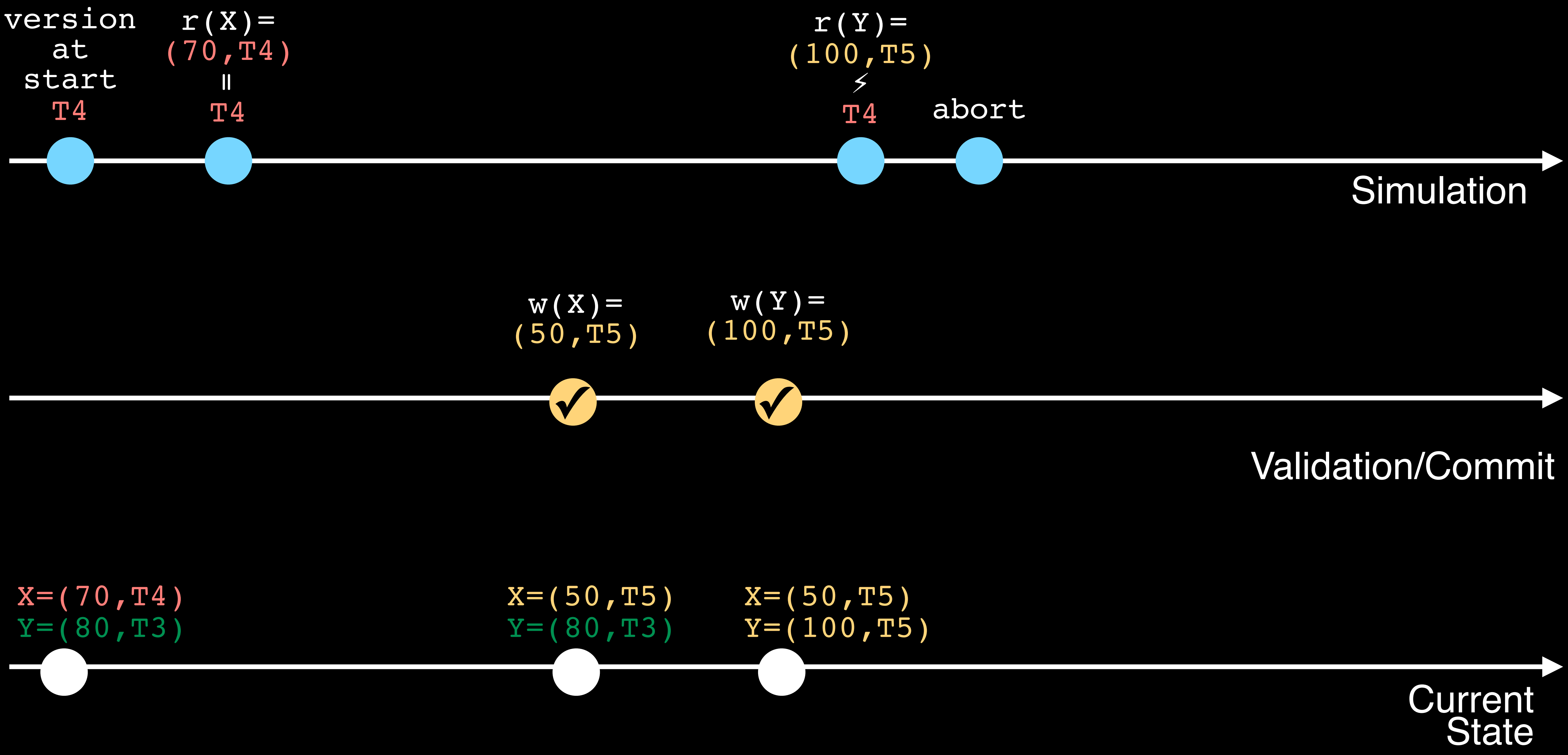
X=(70,T4)
Y=(80,T3)

X=(50,T5)
Y=(80,T3)

X=(50,T5)
Y=(100,T5)

Current State

Fabric++: Multi-version Concurrency Control



version at start T4

r(X)= (70,T4) = T4

r(Y)= (100,T5) ⚡ T4

abort

Simulation

w(X)= (50,T5)

w(Y)= (100,T5)

Validation/Commit

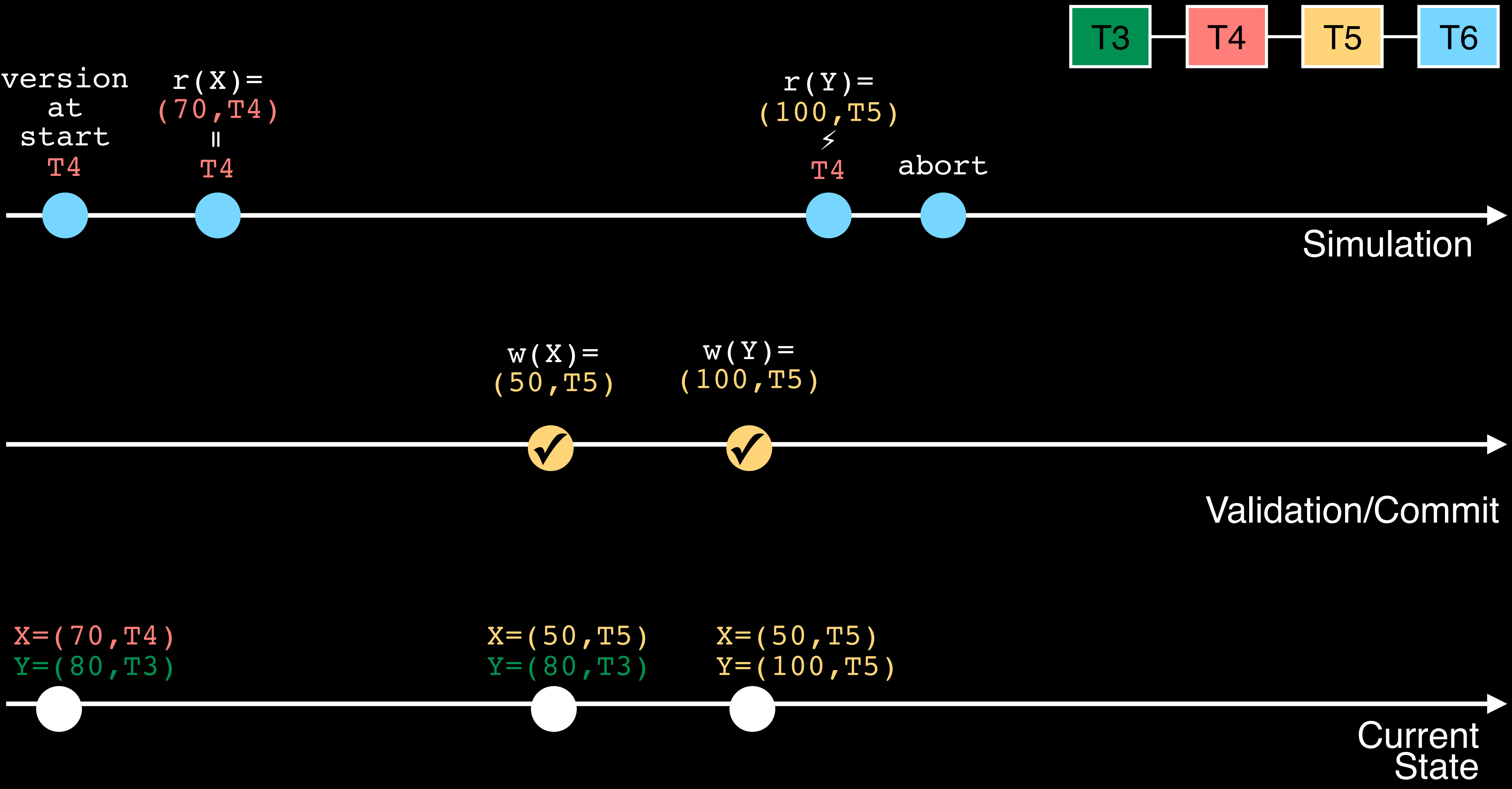
X=(70,T4)
Y=(80,T3)

X=(50,T5)
Y=(80,T3)

X=(50,T5)
Y=(100,T5)

Current State

Fabric++: Multi-version Concurrency Control



T3 T4 T5 T6

version at start T4

r(X)= (70,T4) = T4

r(Y)= (100,T5) ⚡ T4

abort

Simulation

w(X)= (50,T5)

w(Y)= (100,T5)

Validation/Commit

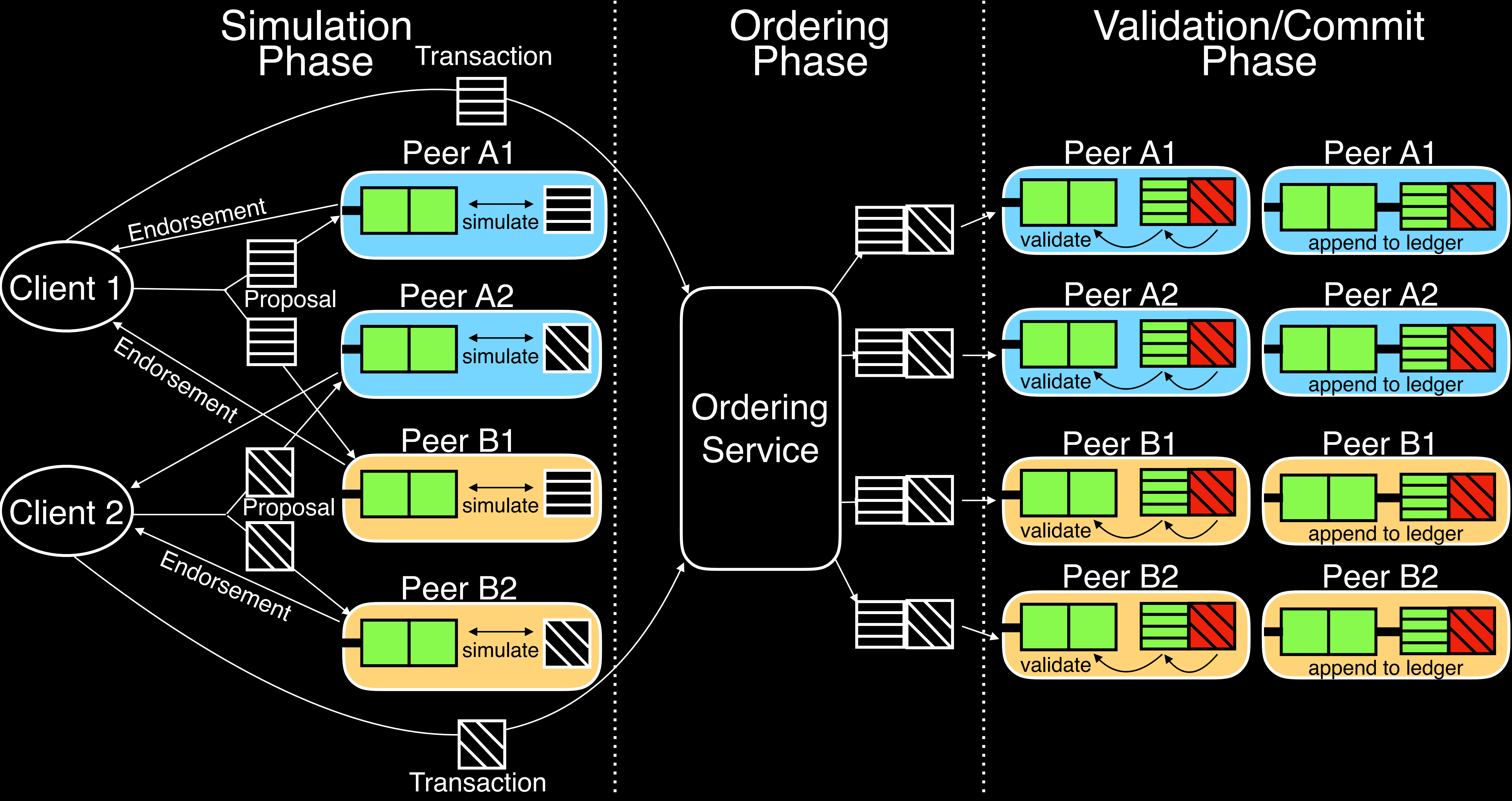
X=(70,T4)
Y=(80,T3)

X=(50,T5)
Y=(80,T3)

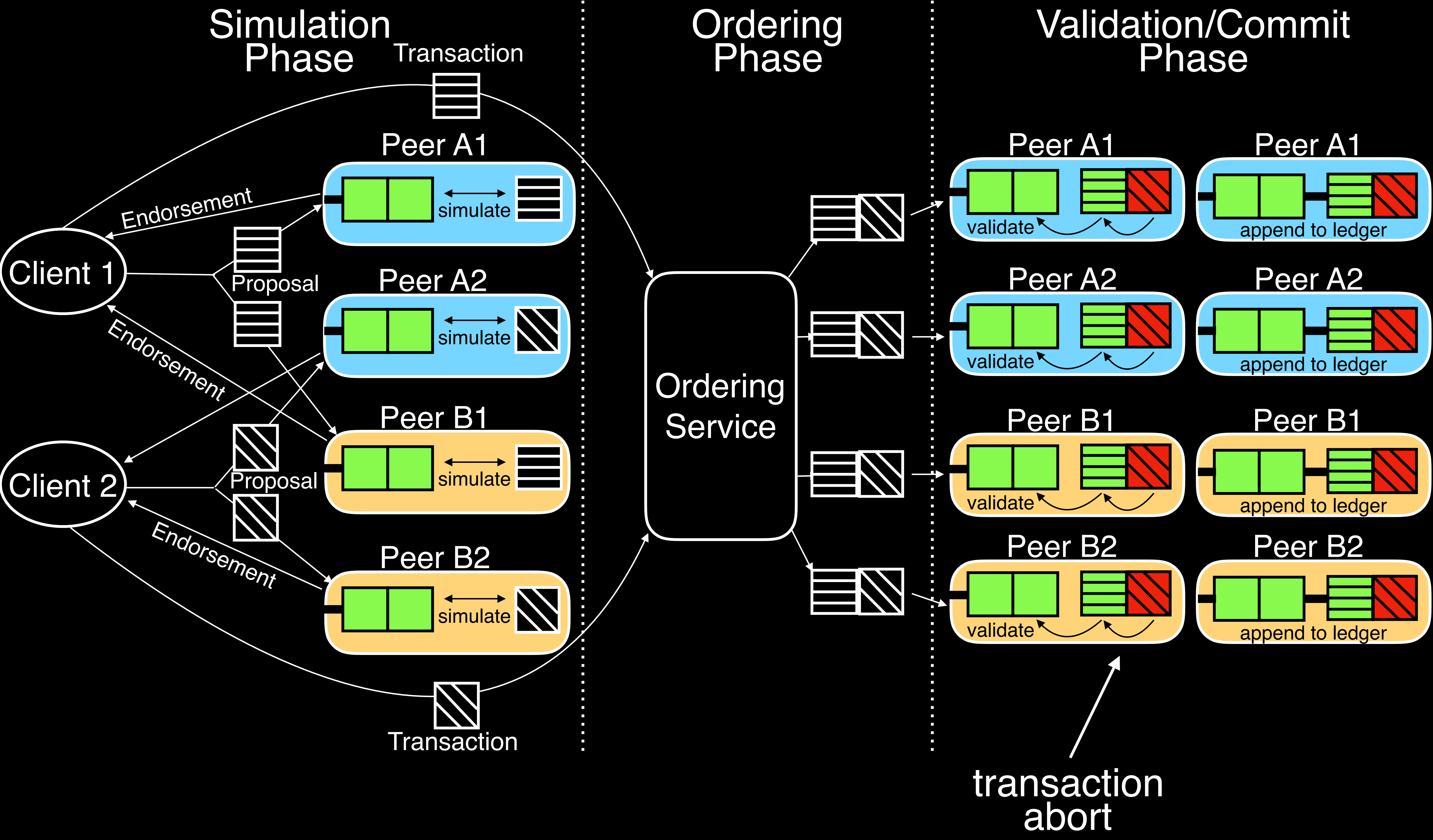
X=(50,T5)
Y=(100,T5)

Current State

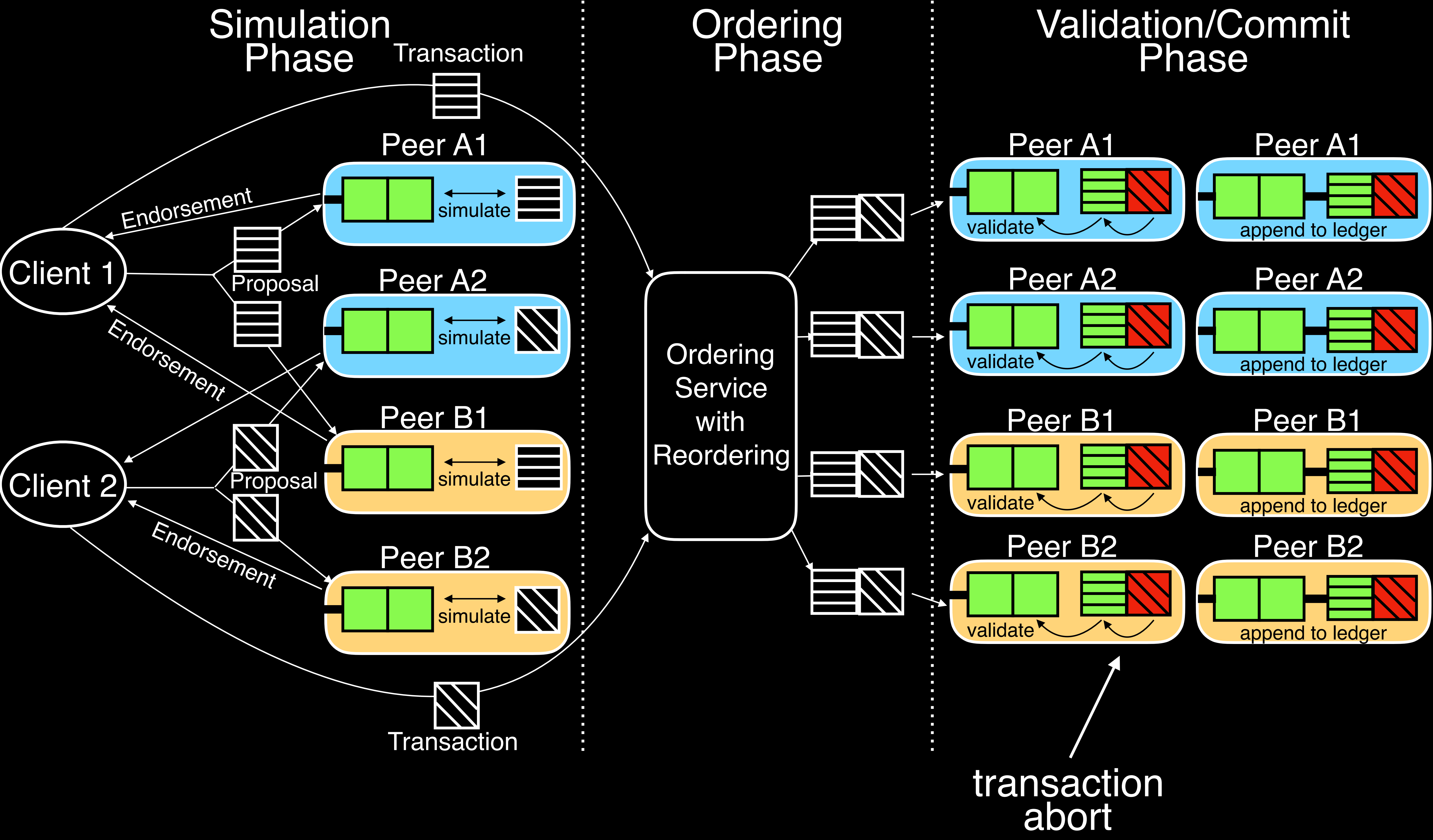
Early Abort of Transactions



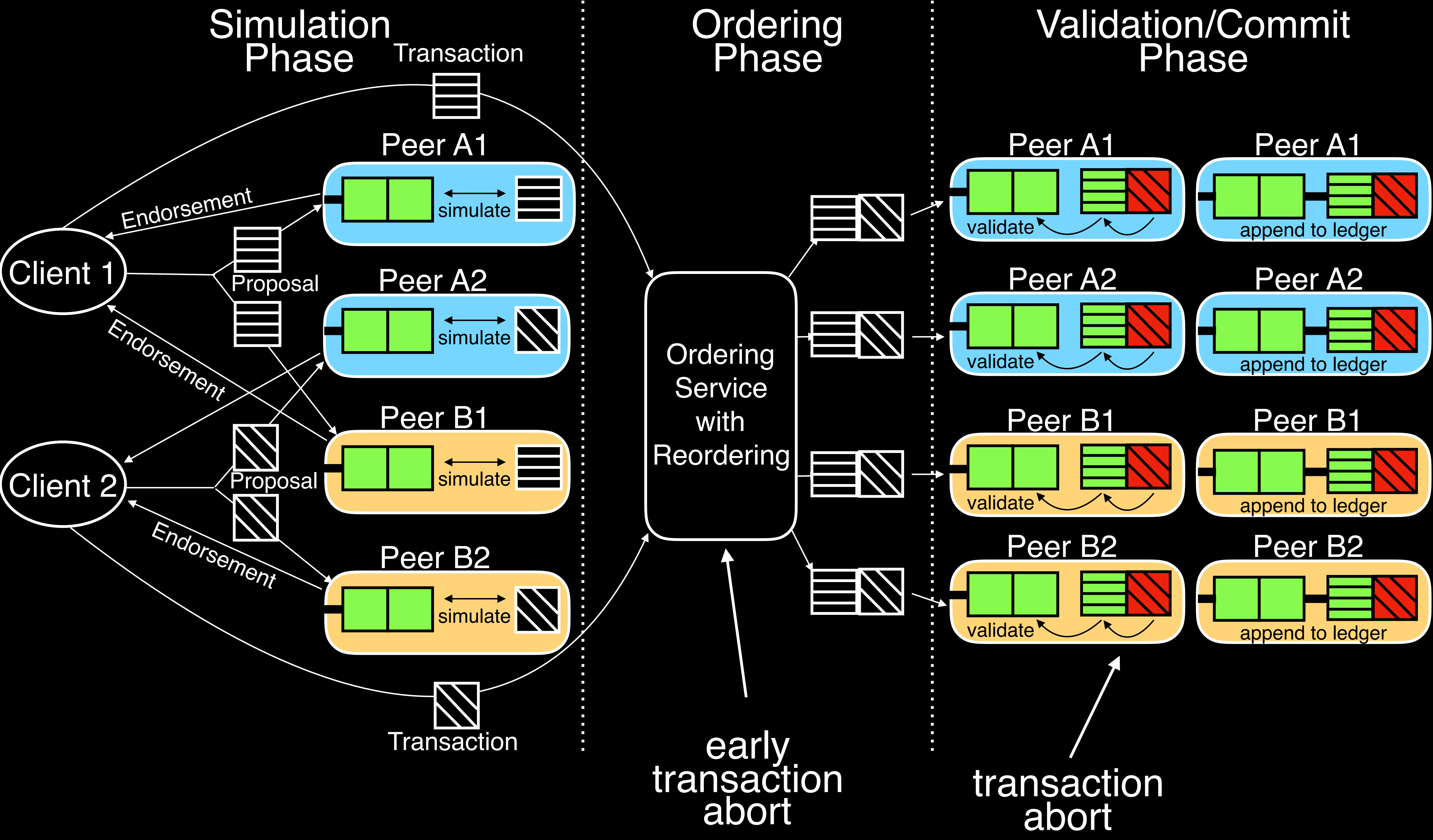
Early Abort of Transactions



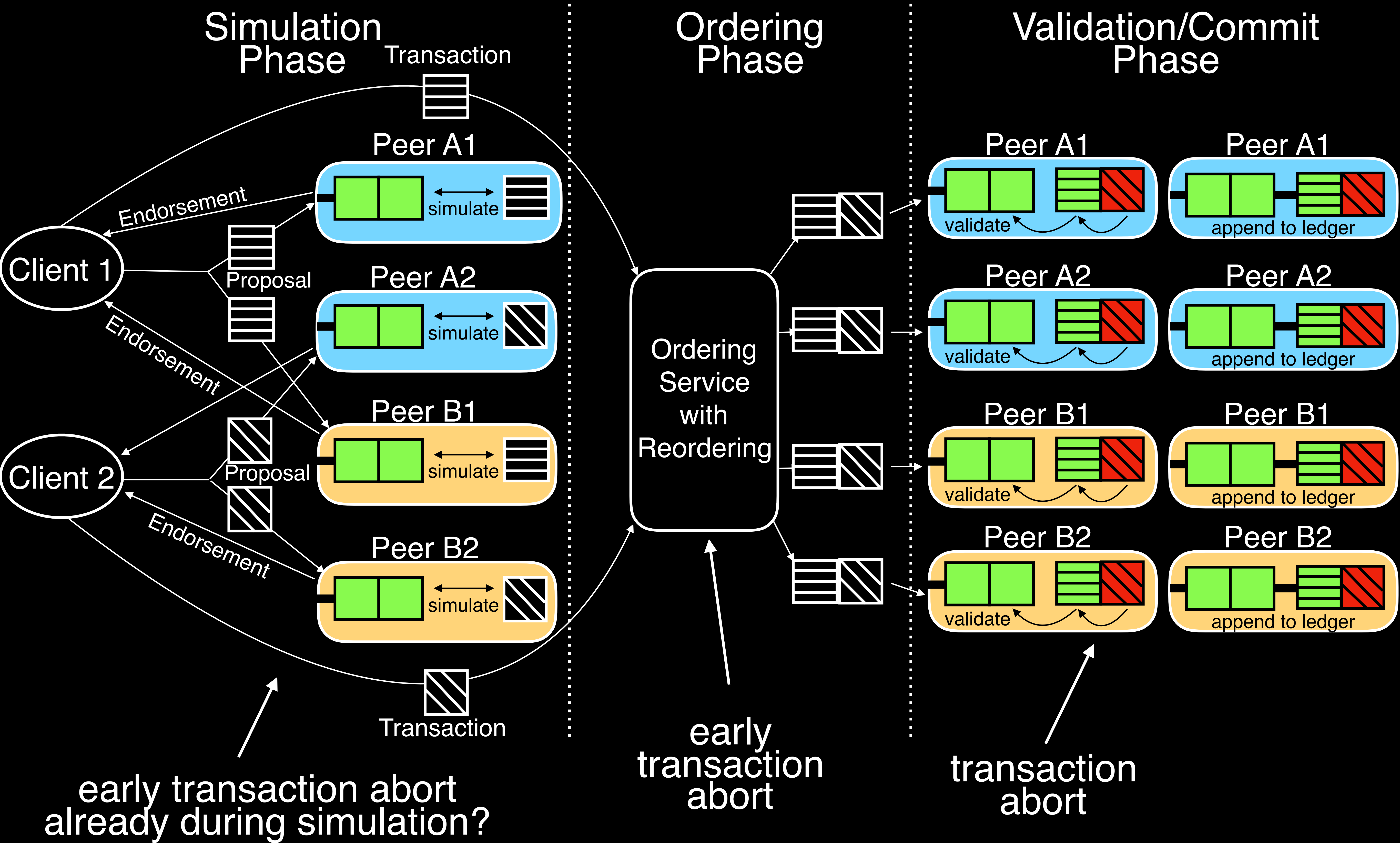
Early Abort of Transactions



Early Abort of Transactions



Early Abort of Transactions

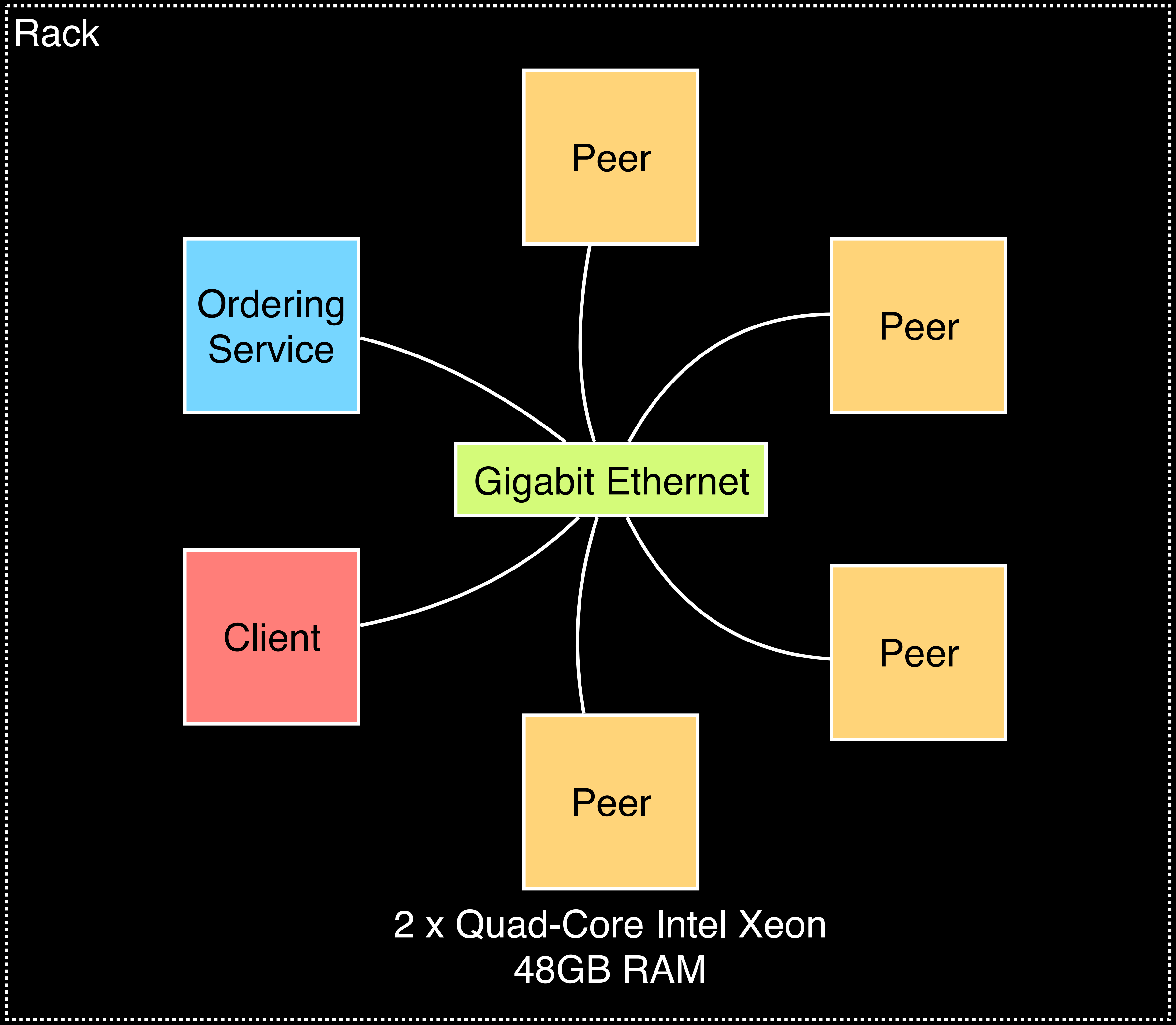


early transaction abort already during simulation?

early transaction abort

transaction abort

Experimental Evaluation: Setup



Experimental Evaluation: Workloads

Experimental Evaluation: Workloads

Smallbank: asset transfer scenario

- 6 transactions: 5 update transactions + 1 read-only transaction:

Workload Parameters	Values
Number of users (two accounts per user)	100.000
Probability for picking a modifying transaction (Pw)	95%, 50%, 5%
s-value of Zipf distribution for account picking	0.0 - 2.0

Experimental Evaluation: Workloads

Smallbank: asset transfer scenario

- 6 transactions: 5 update transactions + 1 read-only transaction:

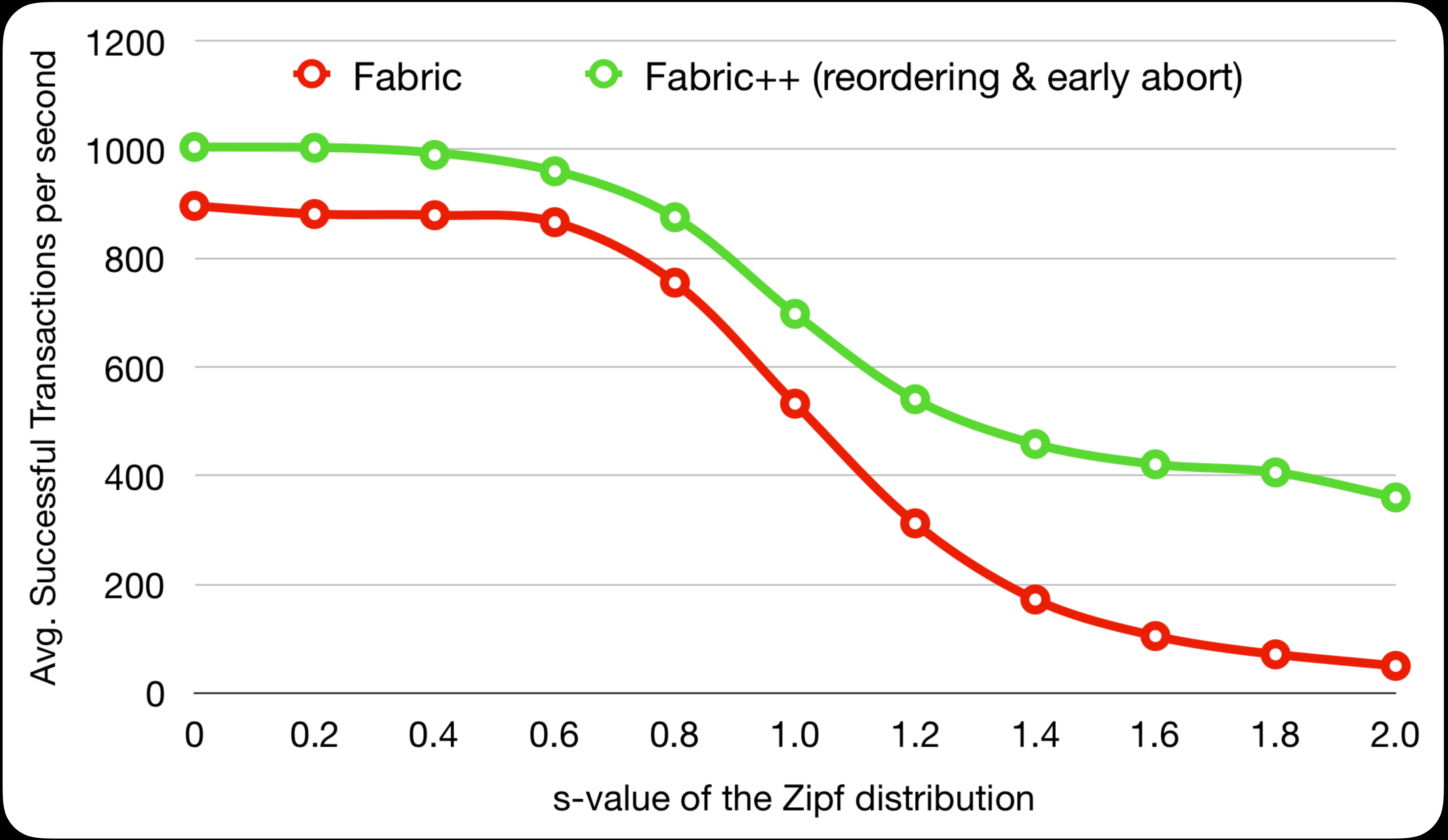
Workload Parameters	Values
Number of users (two accounts per user)	100.000
Probability for picking a modifying transaction (Pw)	95%, 50%, 5%
s-value of Zipf distribution for account picking	0.0 - 2.0

Custom:

- 1 highly-configurable transaction

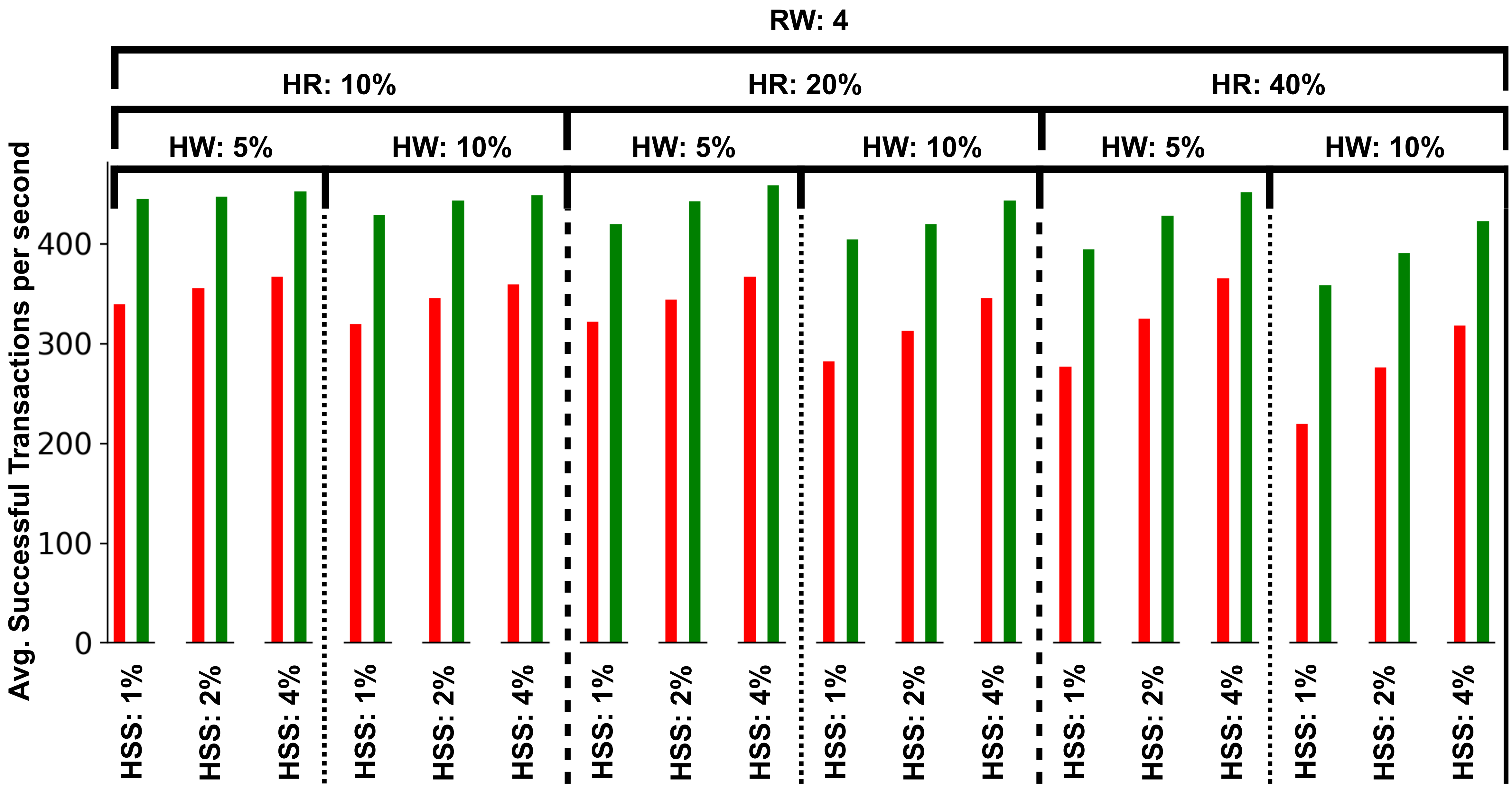
Workload Parameters	Values
Number of accounts balances (N)	10.000
Number of hot account balances (HSS)	1%, 2%, 4%
Number of read & written balances per transaction (RW)	4, 8
Probability for picking a hot account for reading (HR)	10%, 20%, 40%
Probability for picking a hot account for writing (HW)	5%, 10%

Successful Transactions (Smallbank)



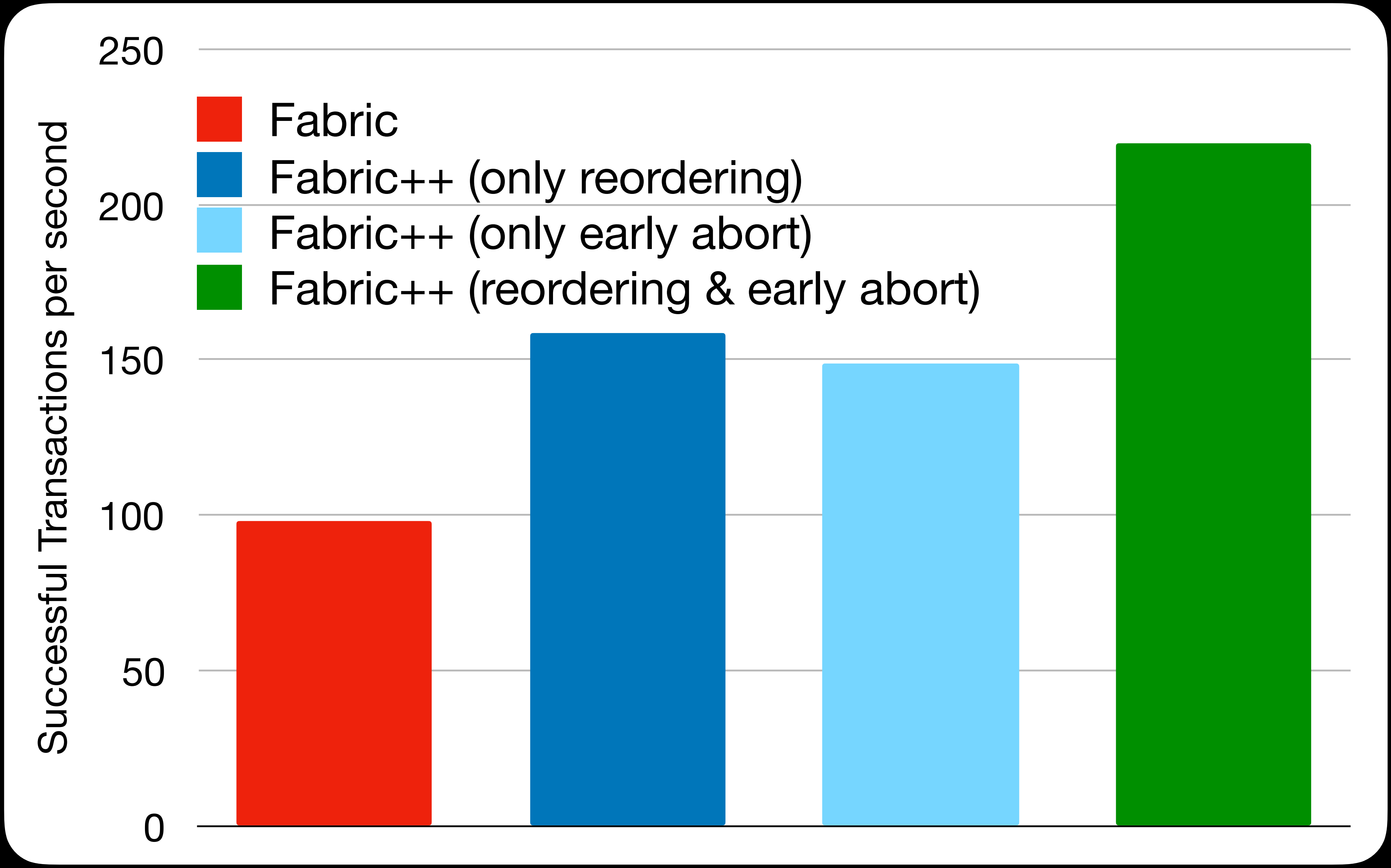
Smallbank balanced workload (Pw = 50%)

Successful Transactions (Custom Workload)



18 different configurations of workload

Optimization Breakdown



Custom Workload:
BS=1024, RW=8, HR=40%, HW=10%, HSS=1%

Conclusion

Fabric

Conclusion

Transaction Reordering

Fabric

Conclusion

Fabric++*

* Open Source. Available at tiny.cc/fabricpp

Conclusion

Fabric++*

Up to 12x Improvement in Successful Transaction's Throughput

* Open Source. Available at tiny.cc/fabricpp

Conclusion

Fabric++*

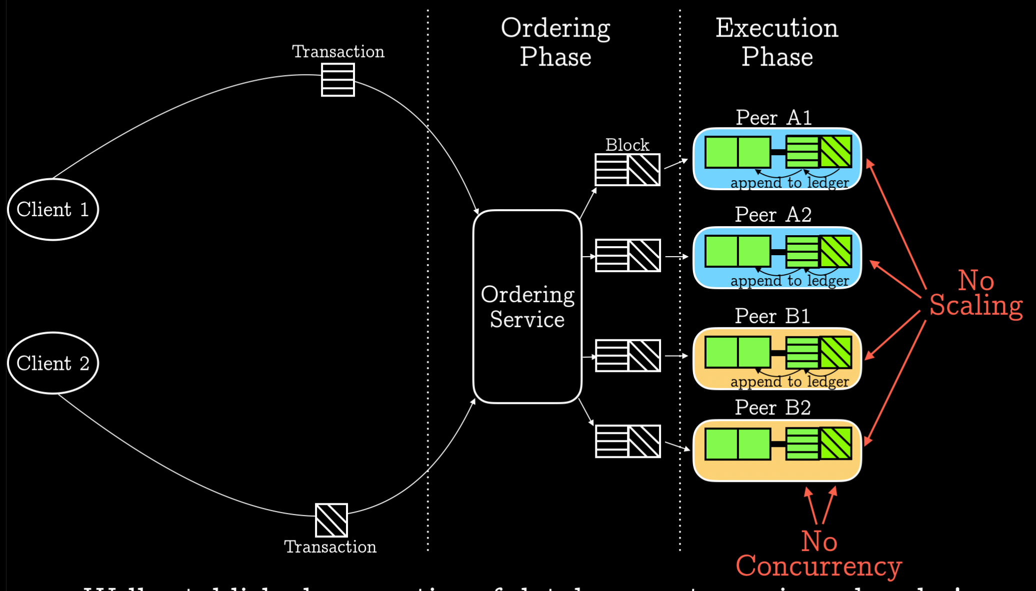
Up to 12x Improvement in Successful Transaction's Throughput

Up to 50% Less Latency

* Open Source. Available at tiny.cc/fabricpp

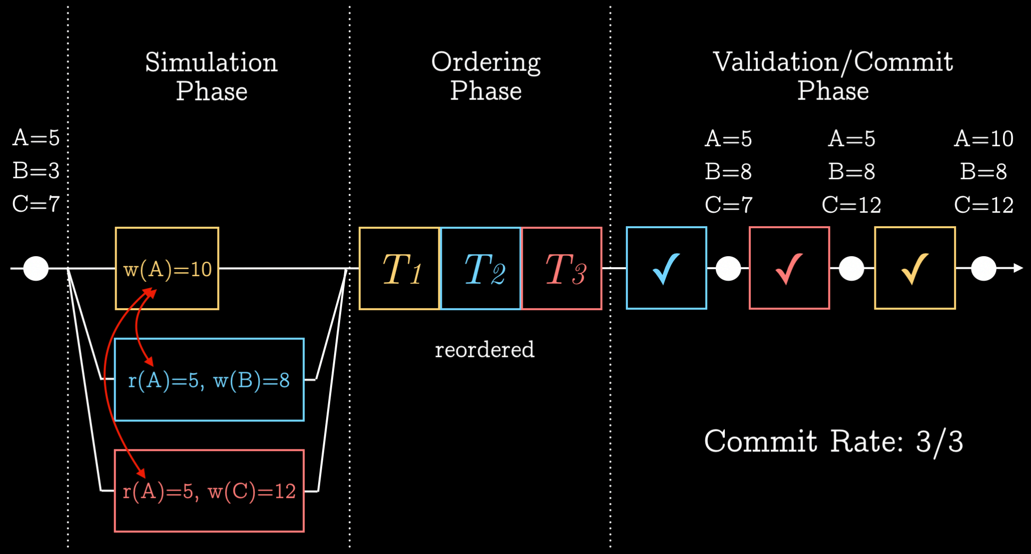
Summary

The order-execute model (Bitcoin, Ethereum, ...)

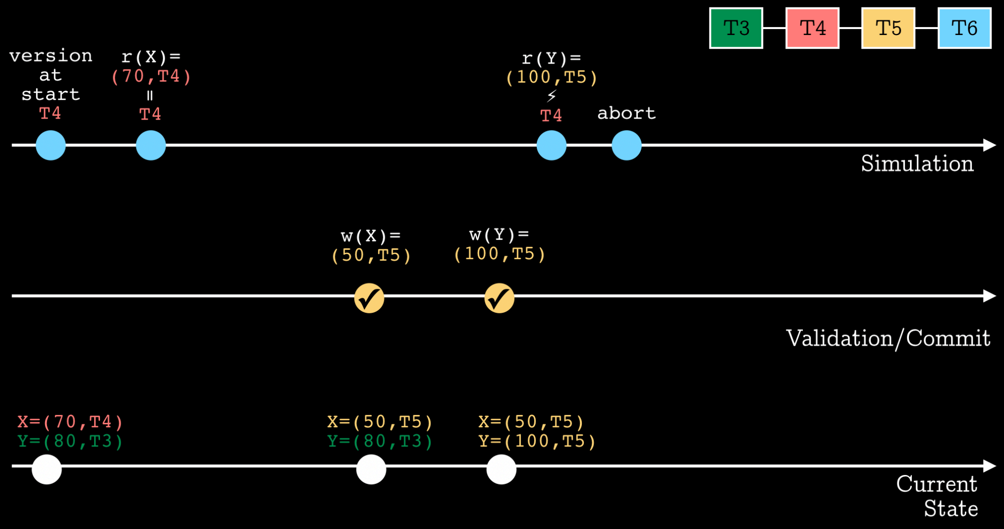


Well-established properties of database systems since decades!

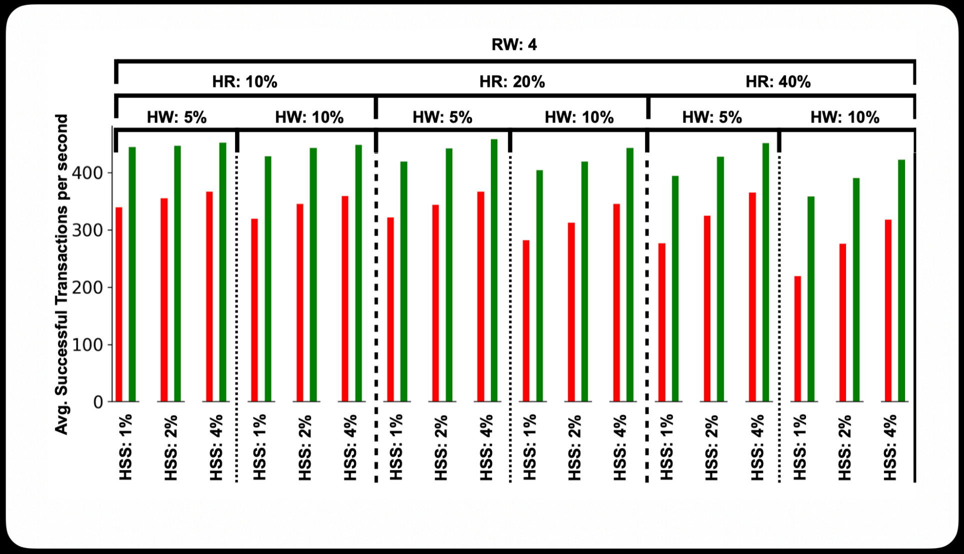
Serialization Conflicts



Fabric++: Multi-version Concurrency Control



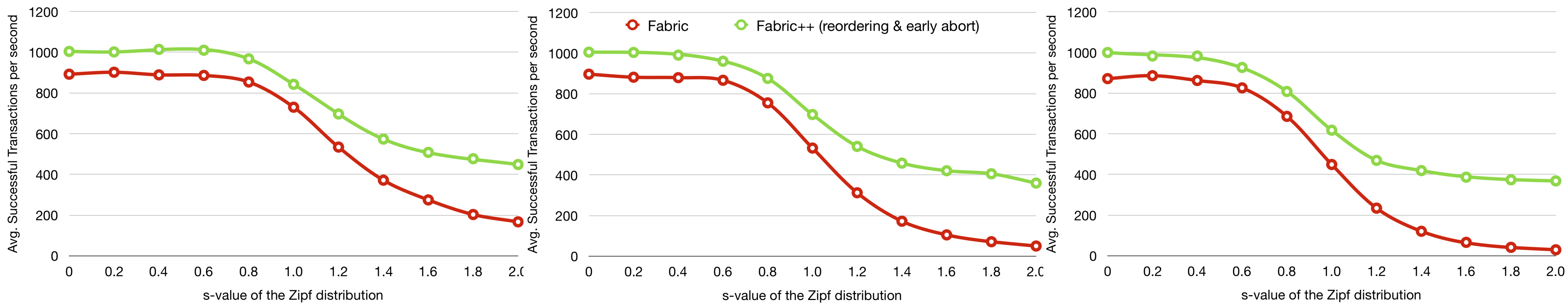
Successful Transactions (Custom Workload)



18 different configurations of workload

Backup Slides

Successful Transactions (Smallbank)

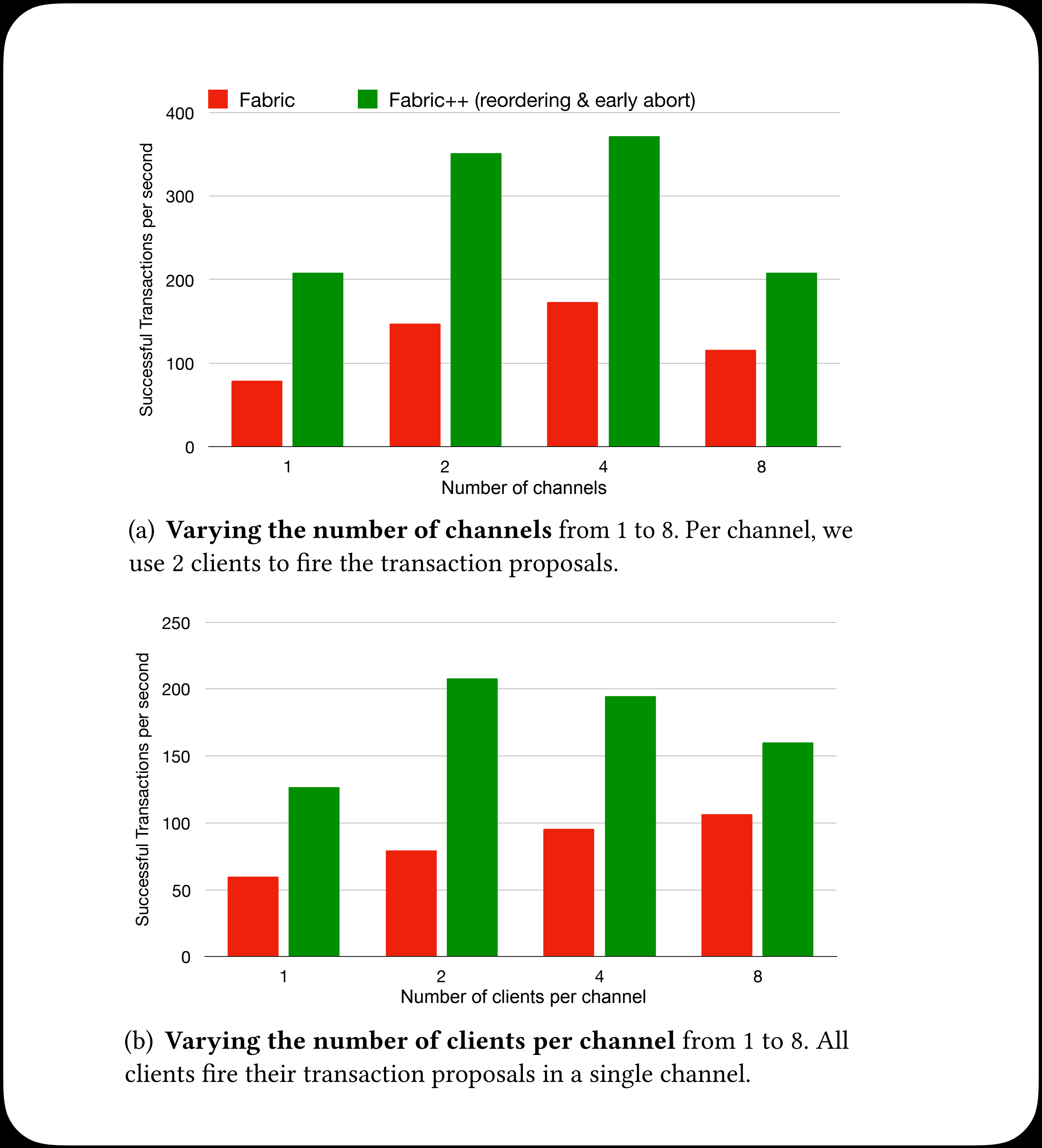


(a) $P_w = 5\%$ (read-heavy)

(b) $P_w = 50\%$ (balanced)

(c) $P_w = 95\%$ (write-heavy)

Scaling of Fabric++: Custom Workload



(a) **Varying the number of channels** from 1 to 8. Per channel, we use 2 clients to fire the transaction proposals.

(b) **Varying the number of clients per channel** from 1 to 8. All clients fire their transaction proposals in a single channel.