Imperial College
London

LSDS
Large-Scale Data
& Systems Group

# Teechain: A Secure Payment Network with Asynchronous Blockchain Access

Joshua Lind Oded Naor Ittay Eyal

Florian Kelbert Emin Gün Sirer Peter Pietzuch

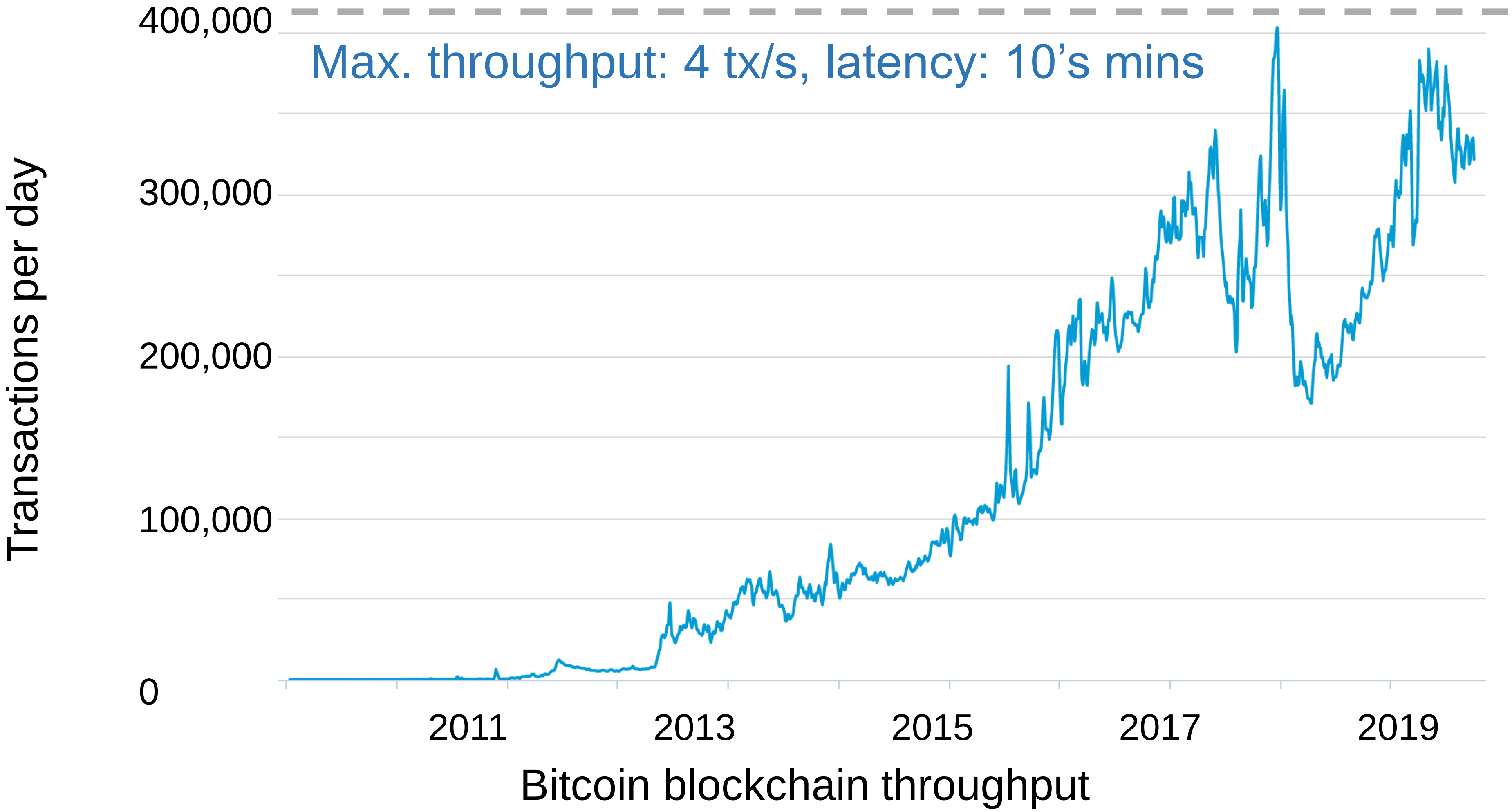
joshua.lind11@imperial.ac.uk
Imperial College London

Cornell University

TECHNION
Israel Institute
of Technology

# Blockchains aren’t scaling!

**Consensus is slow**: all nodes must agree on all transactions!



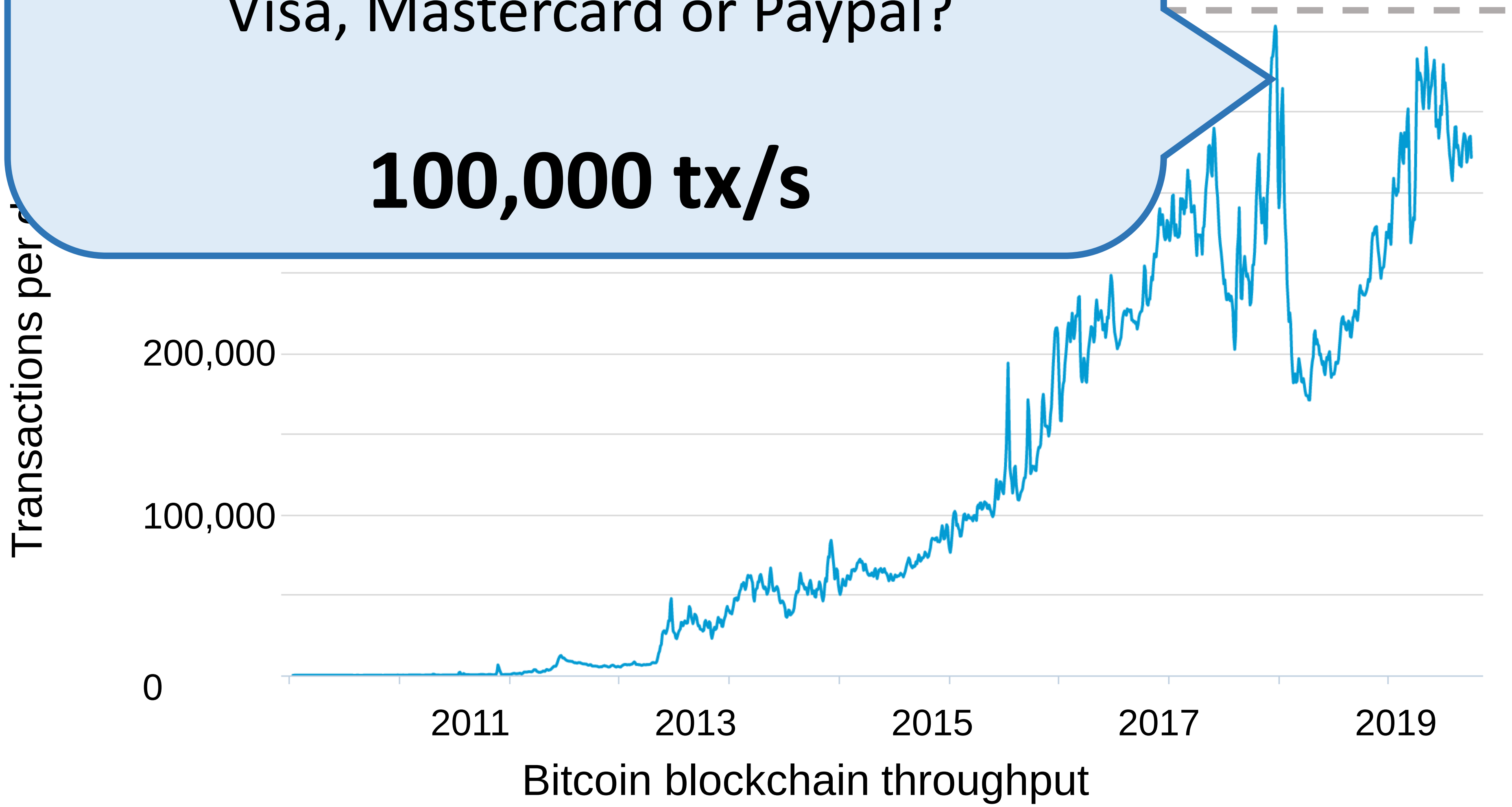
caling!

ansactions!

## Consensus isn’t scaling!

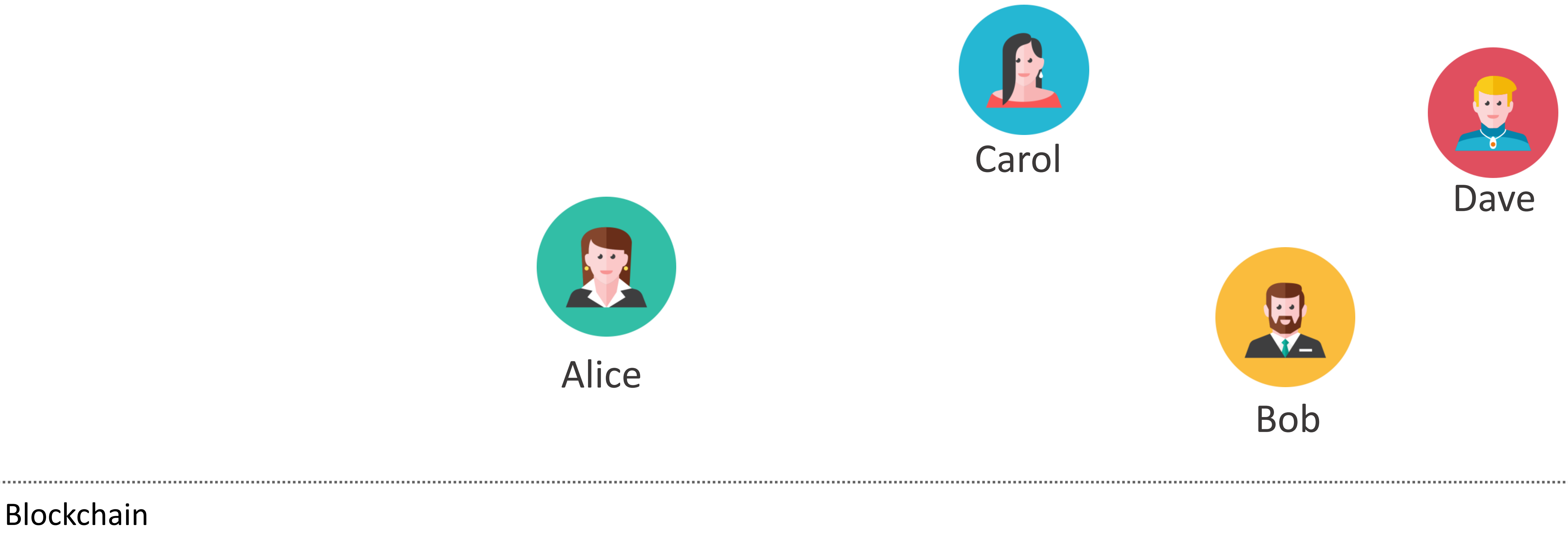
How can we reach the throughput of Visa, Mastercard or Paypal?

**100,000 tx/s**



# Off-chain scaling: Payment Networks

- Execute payments **off-chain!**
  - Parties pay each other directly



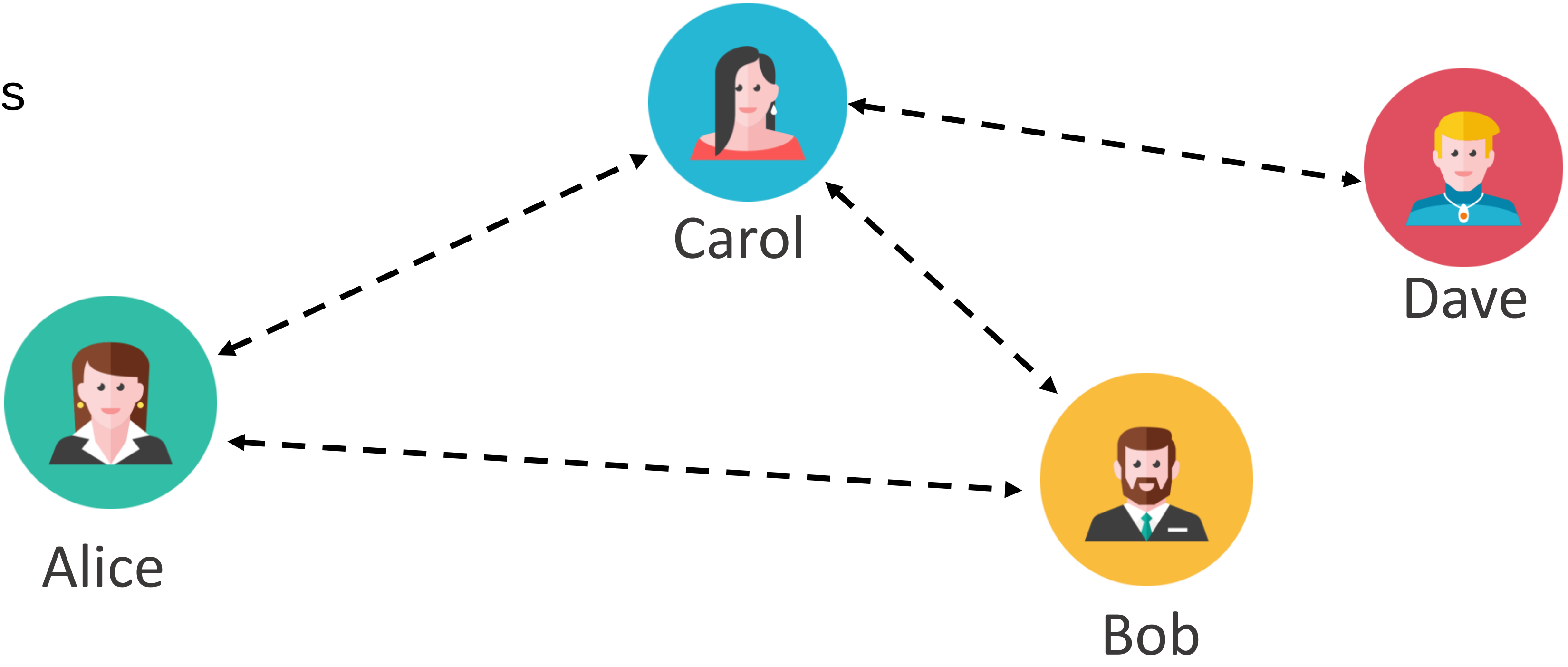
# Off-chain scaling: Payment Networks

Execute payments **off-chain!**

- Parties pay each other directly

Payment channels:

- Point to point payments



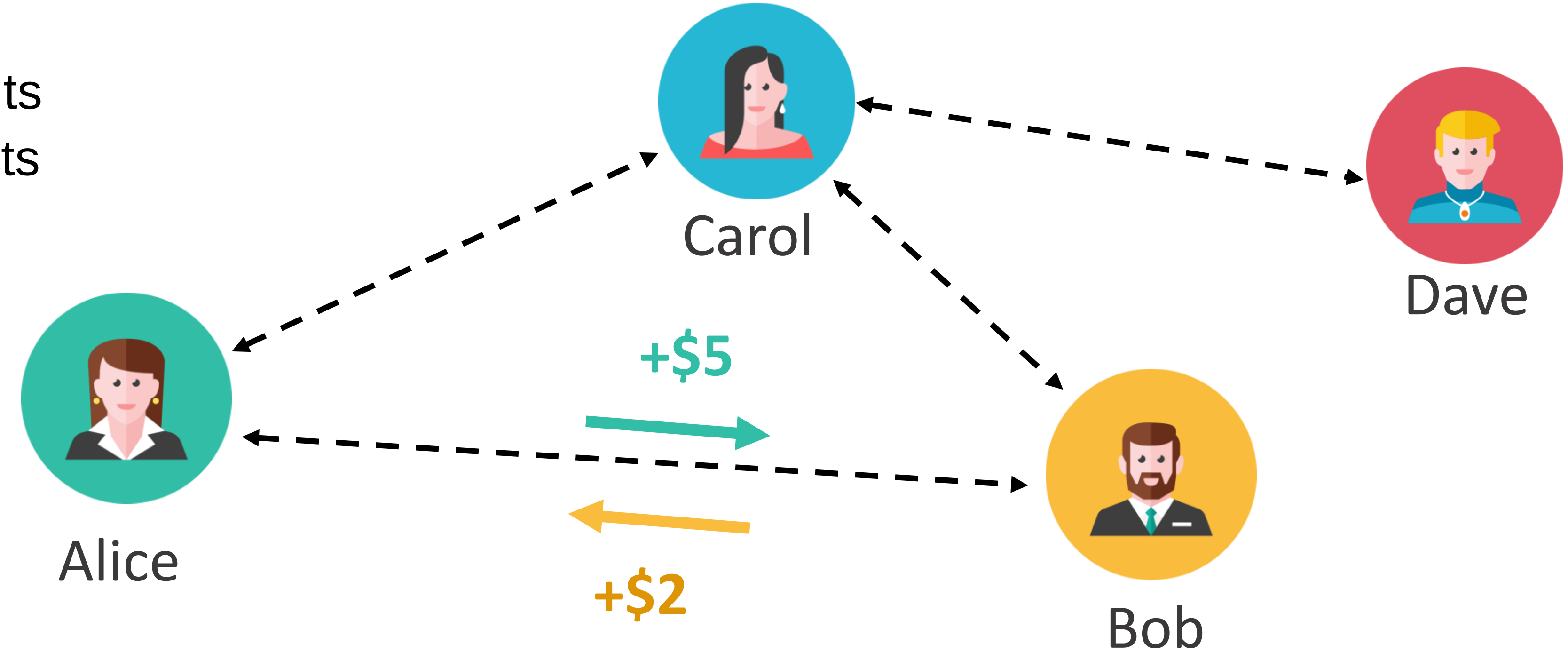
# Off-chain scaling: Payment Networks

Execute payments **off-chain!**

- Parties pay each other directly

Payment channels:

- Point to point payments
- Bi-directional payments



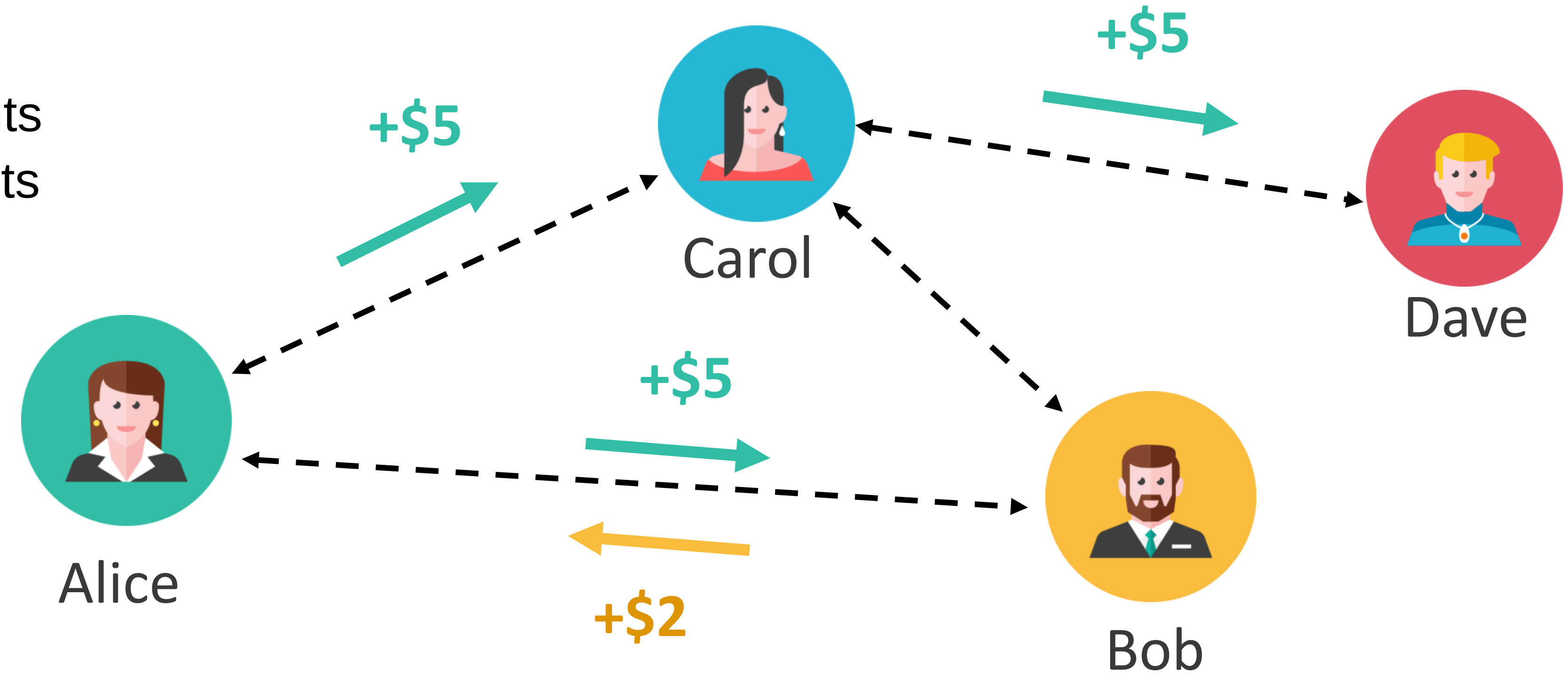
# Off-chain scaling: Payment Networks

Execute payments **off-chain!**

- Parties pay each other directly

Payment channels:

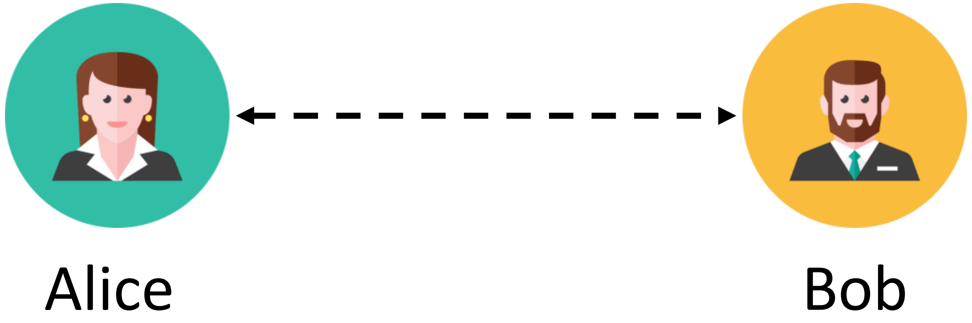
- Point to point payments
- Bi-directional payments
- Multi-hop payments



# Background: Payment Channels

**How do payment channels work?**

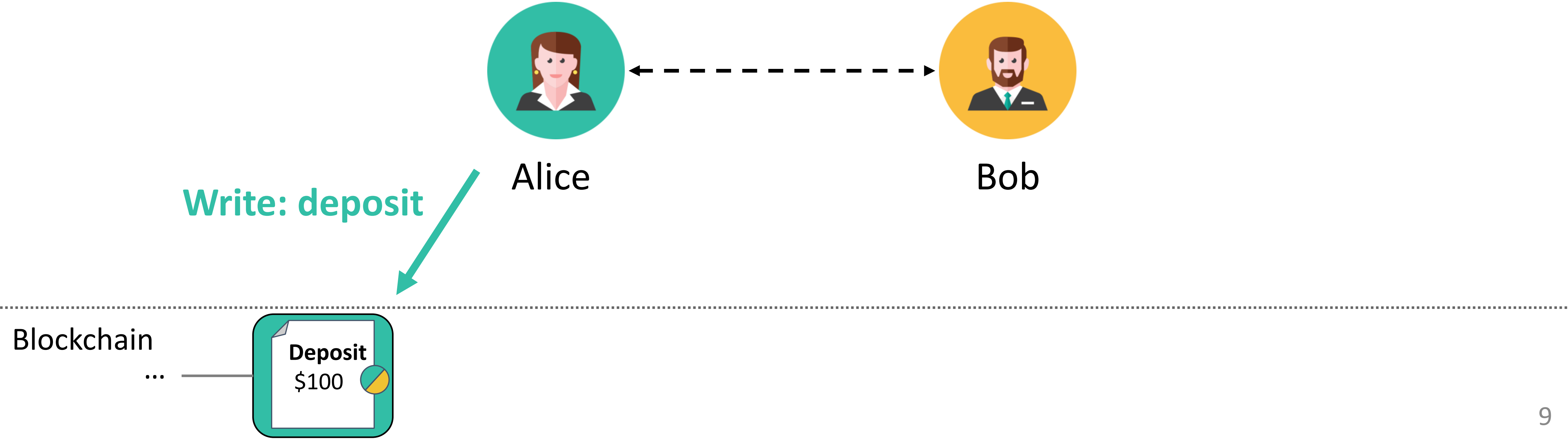
- 3 phases: setup, payments, settlement



# Background: Payment Channels

**How do payment channels work?**

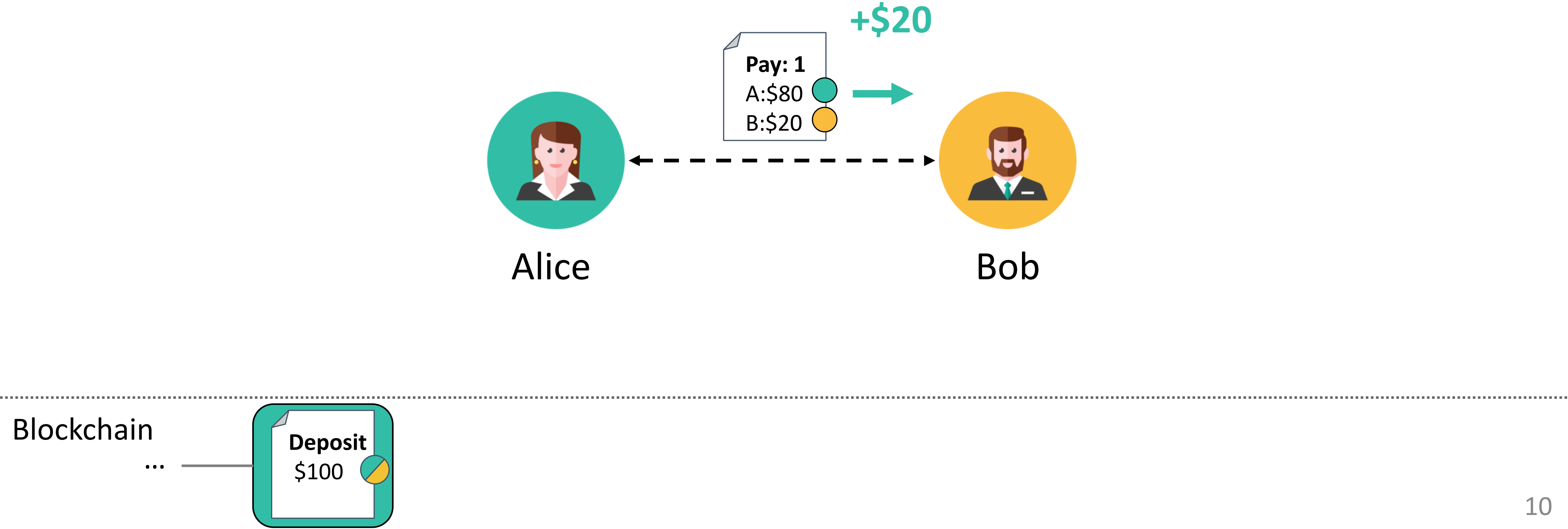
– 3 phases: **setup**, payments, settlement



# Background: Payment Channels

**How do payment channels work?**

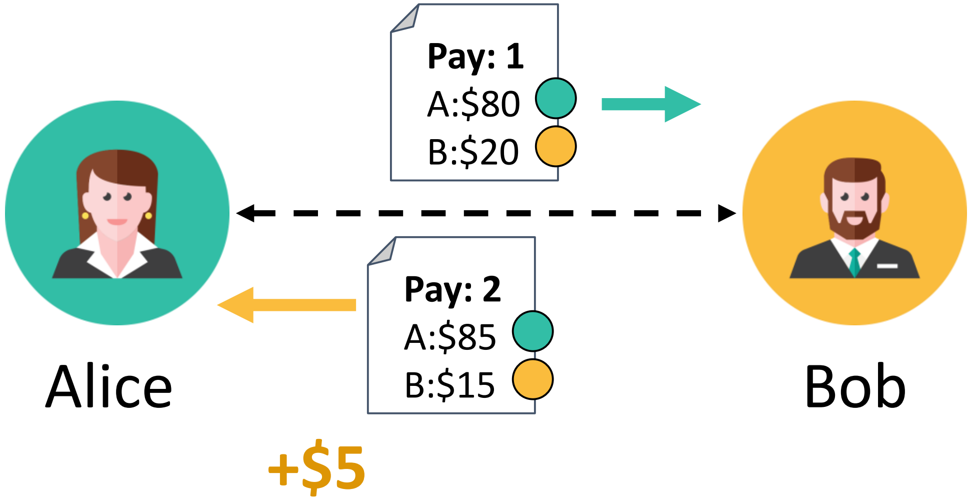
– 3 phases: setup, **payments**, settlement

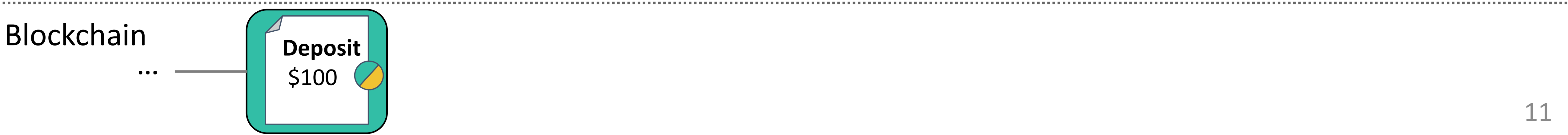


# Background: Payment Channels

## How do payment channels work?

– 3 phases: setup, **payments**, settlement

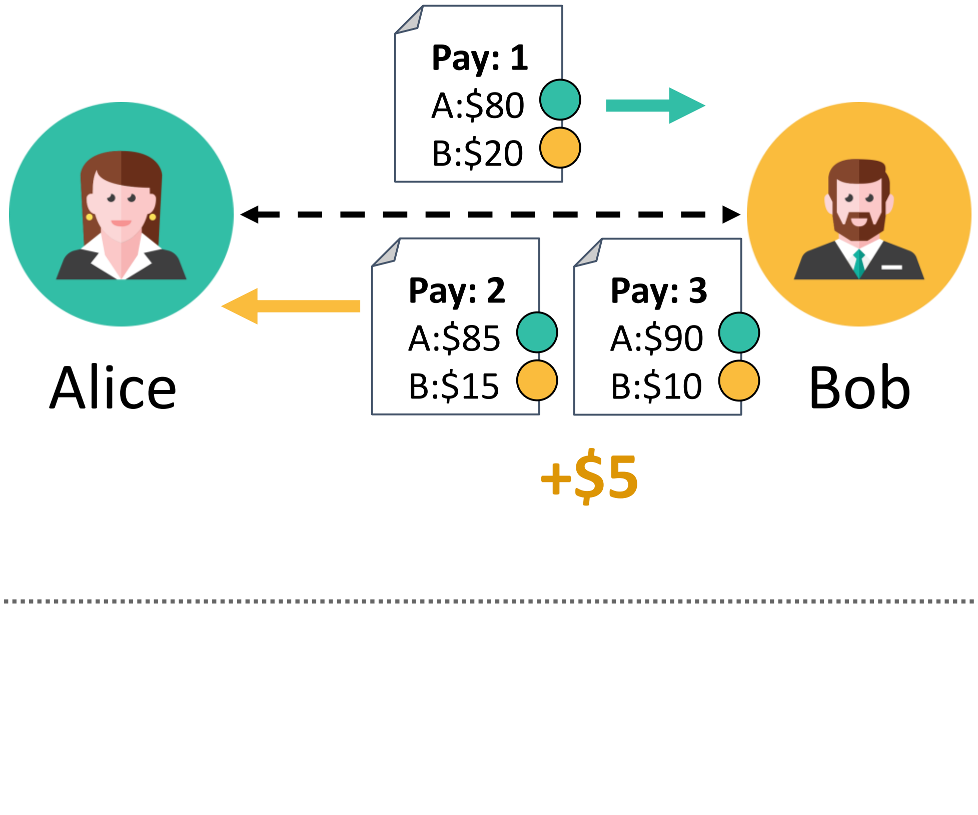




# Background: Payment Channels

## How do payment channels work?

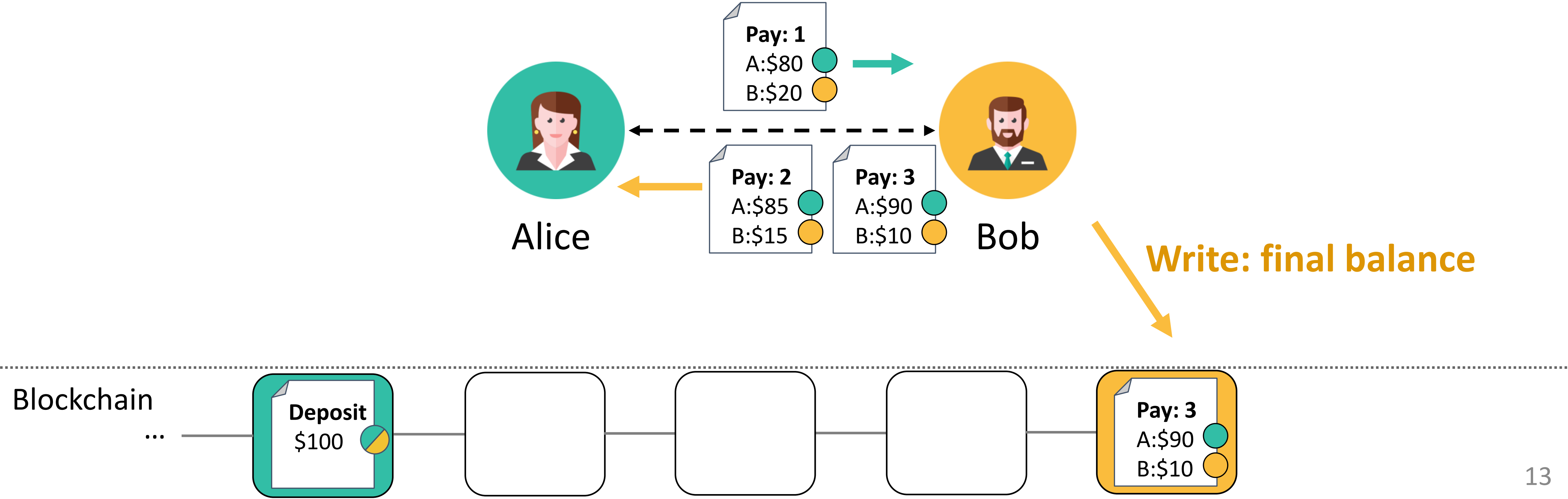
– 3 phases: setup, **payments**, settlement



# Background: Payment Channels

**How do payment channels work?**

- 3 phases: setup, payments, **settlement**

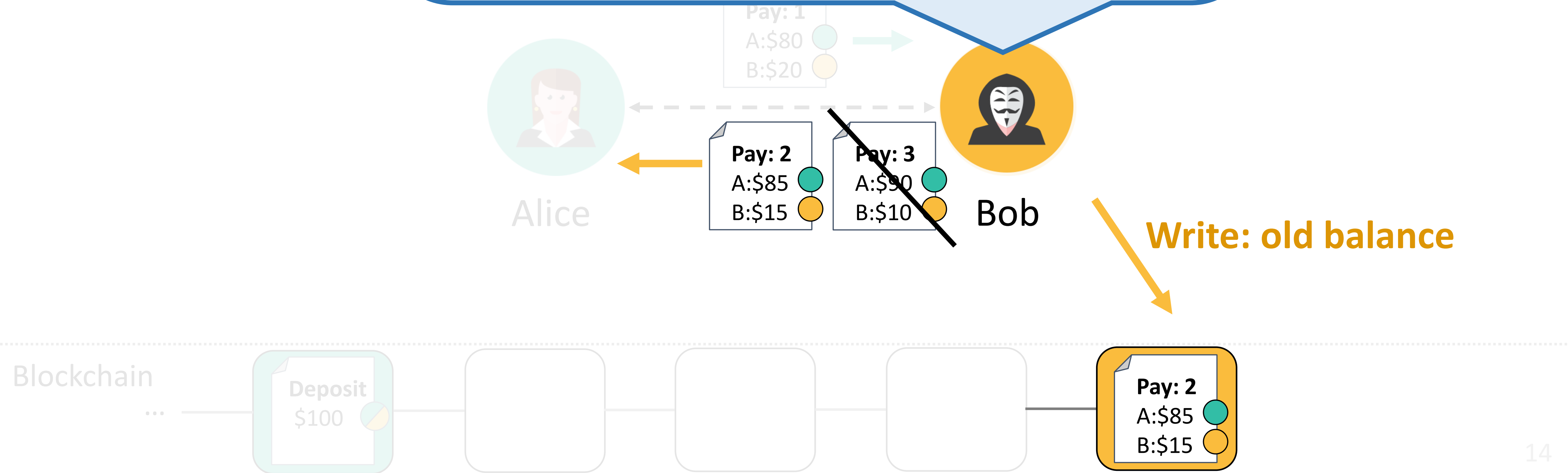


How do paym

– 3 phases: se

## Roll-back attacks!

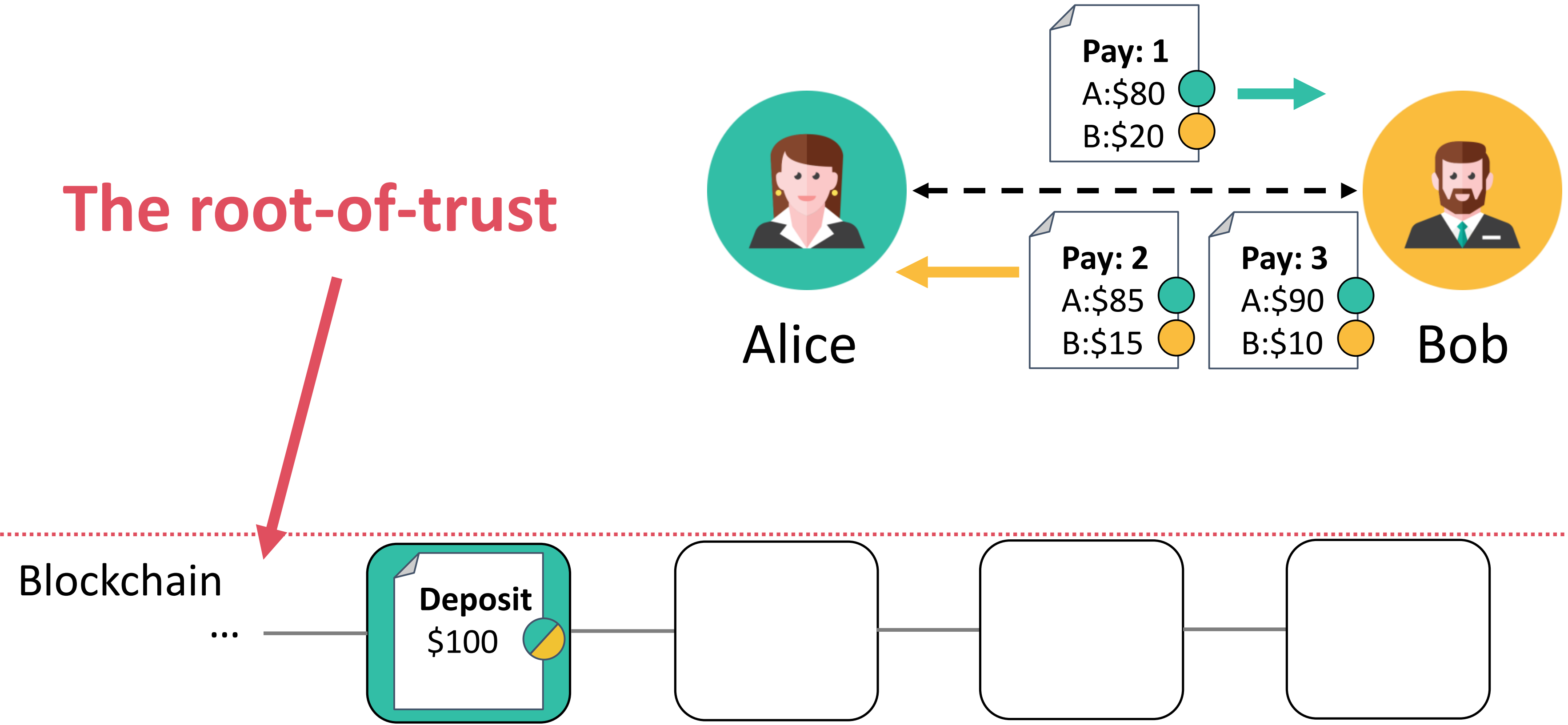
What if Bob misbehaves and writes an old balance to the blockchain?



# Background: Payment Channels

Existing solutions to **roll-back attacks**:

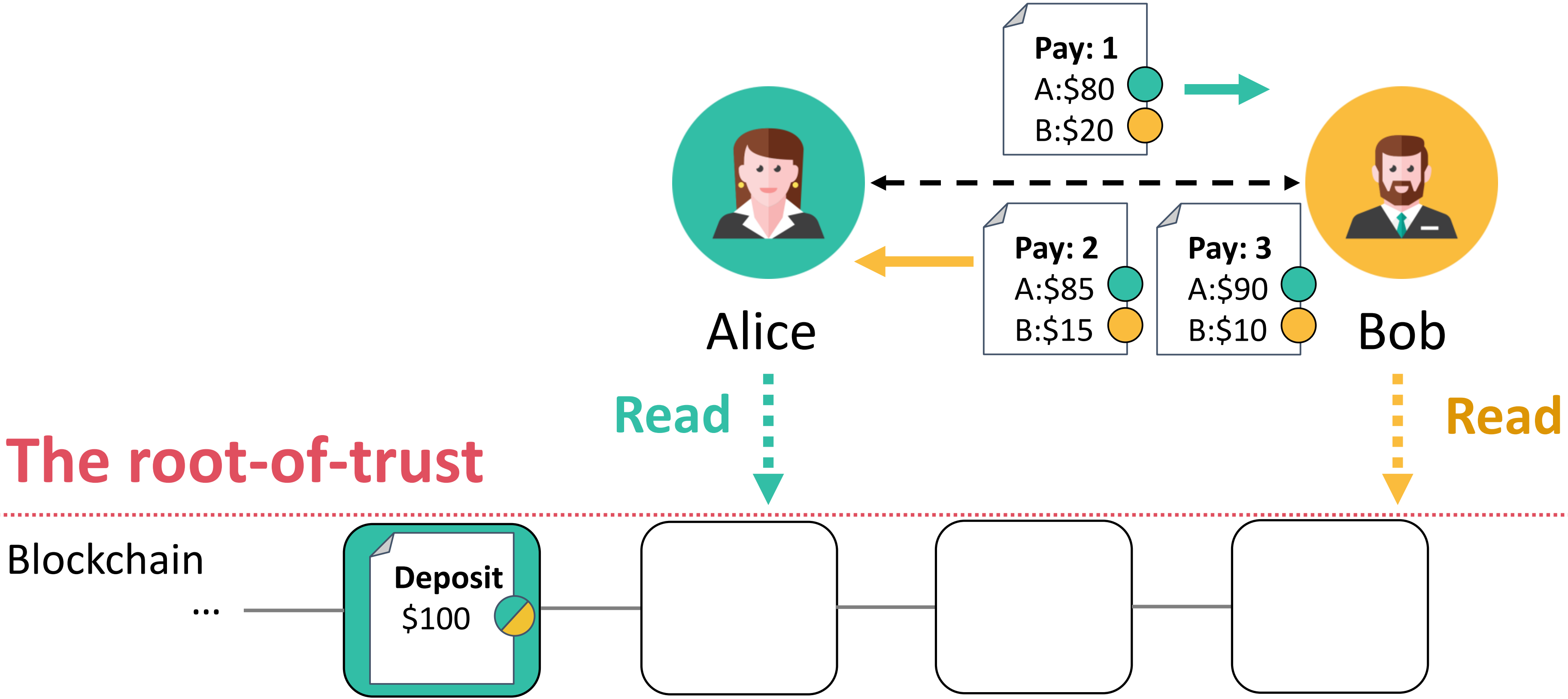
- Monitor the blockchain (**root-of-trust**)
- React within **reaction time (Δ)**
- Final balance on the blockchain



# Background: Payment Channels

Existing solutions to **roll-back attacks**:

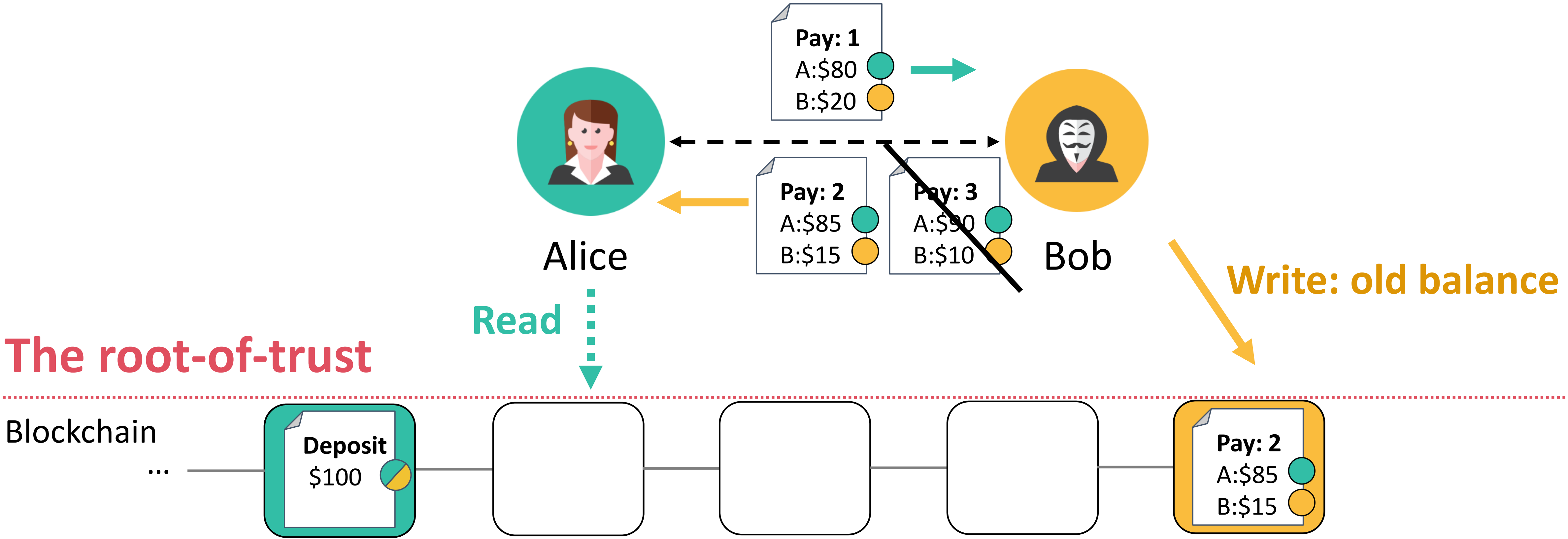
- Monitor the blockchain (**root-of-trust**)
- React within **reaction time (Δ)**
- Final balance on the blockchain



# Background: Payment Channels

Existing solutions to **roll-back attacks**:

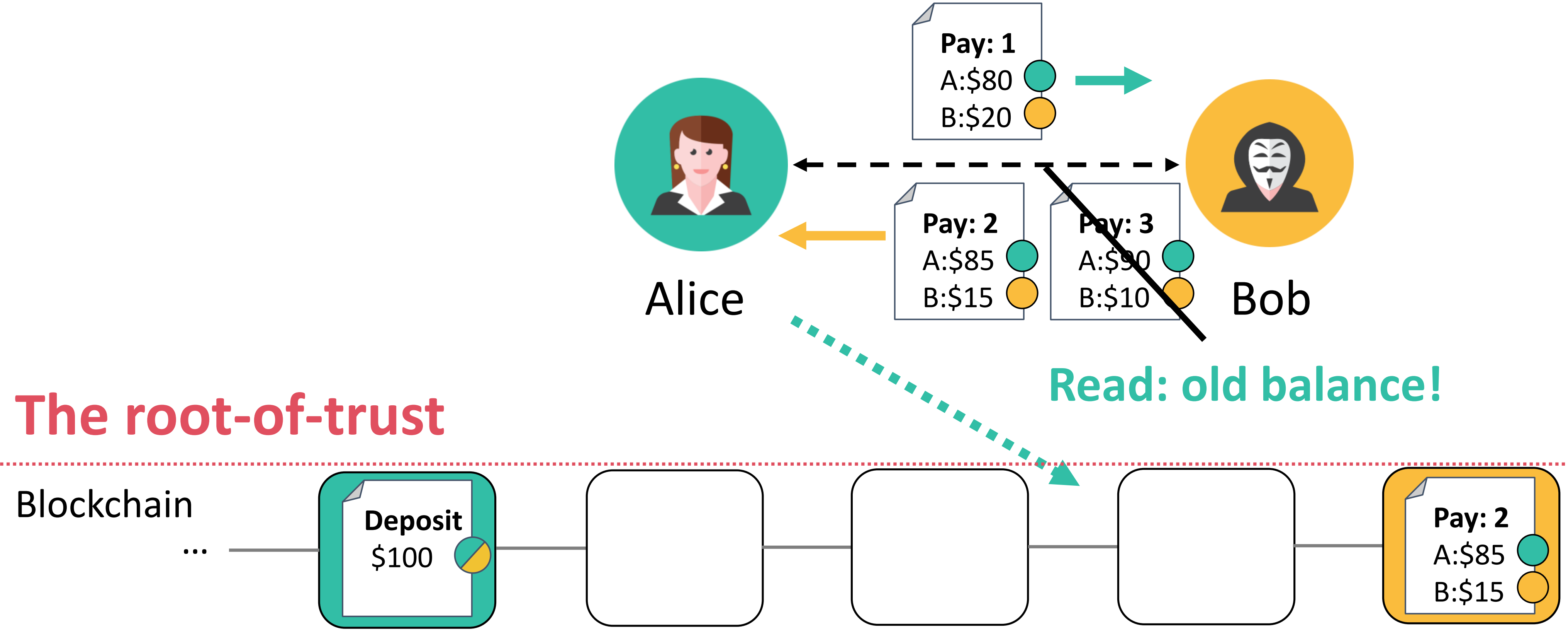
- Monitor the blockchain (**root-of-trust**)
- React within **reaction time (Δ)**
- Final balance on the blockchain



# Background: Payment Channels

Existing solutions to **roll-back attacks**:

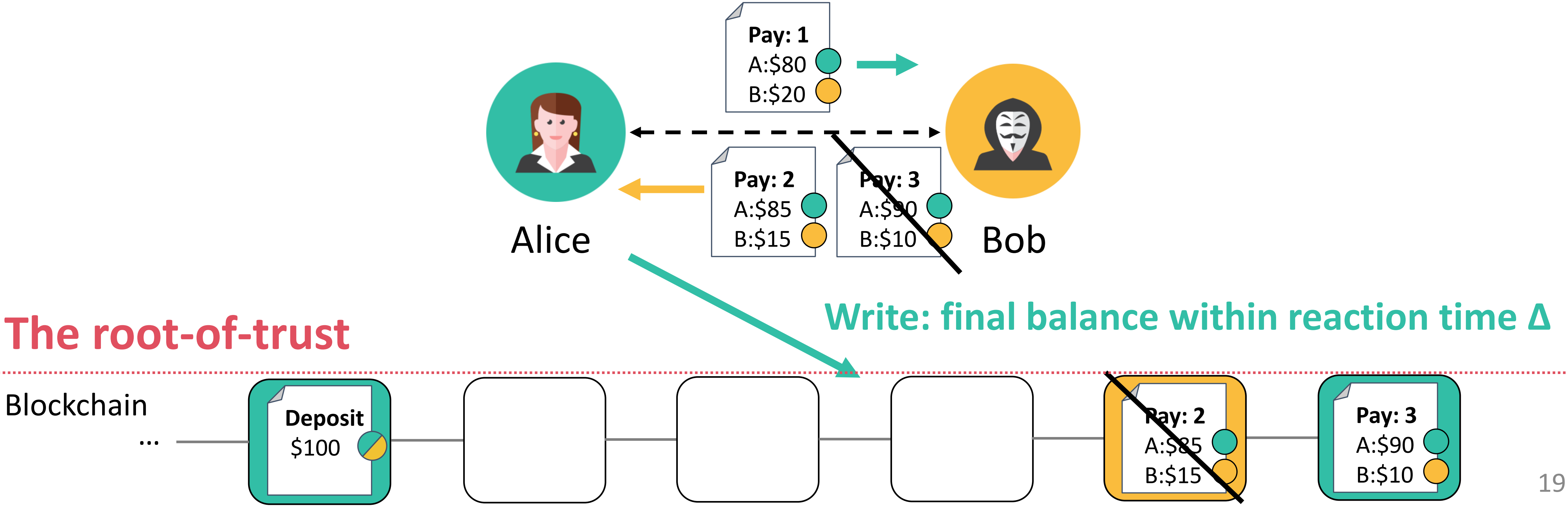
- Monitor the blockchain (**root-of-trust**)
- React within **reaction time (Δ)**
- Final balance on the blockchain



# Background: Payment Channels

Existing solutions to **roll-back attacks**:

- Monitor the blockchain (**root-of-trust**)
- React within **reaction time (Δ)**
- Final balance on the blockchain



Pay: 1
A:$80
B:$20

Alice

Bob

Pay: 2
A:$85
B:$15

~~Pay: 3
A:$90
B:$10~~

The root-of-trust

Write: final balance within reaction time Δ

Blockchain

…

Deposit
$100

~~Pay: 2
A:$85
B:$15~~

Pay: 3
A:$90
B:$10

# Background: Reacting to roll-back attacks

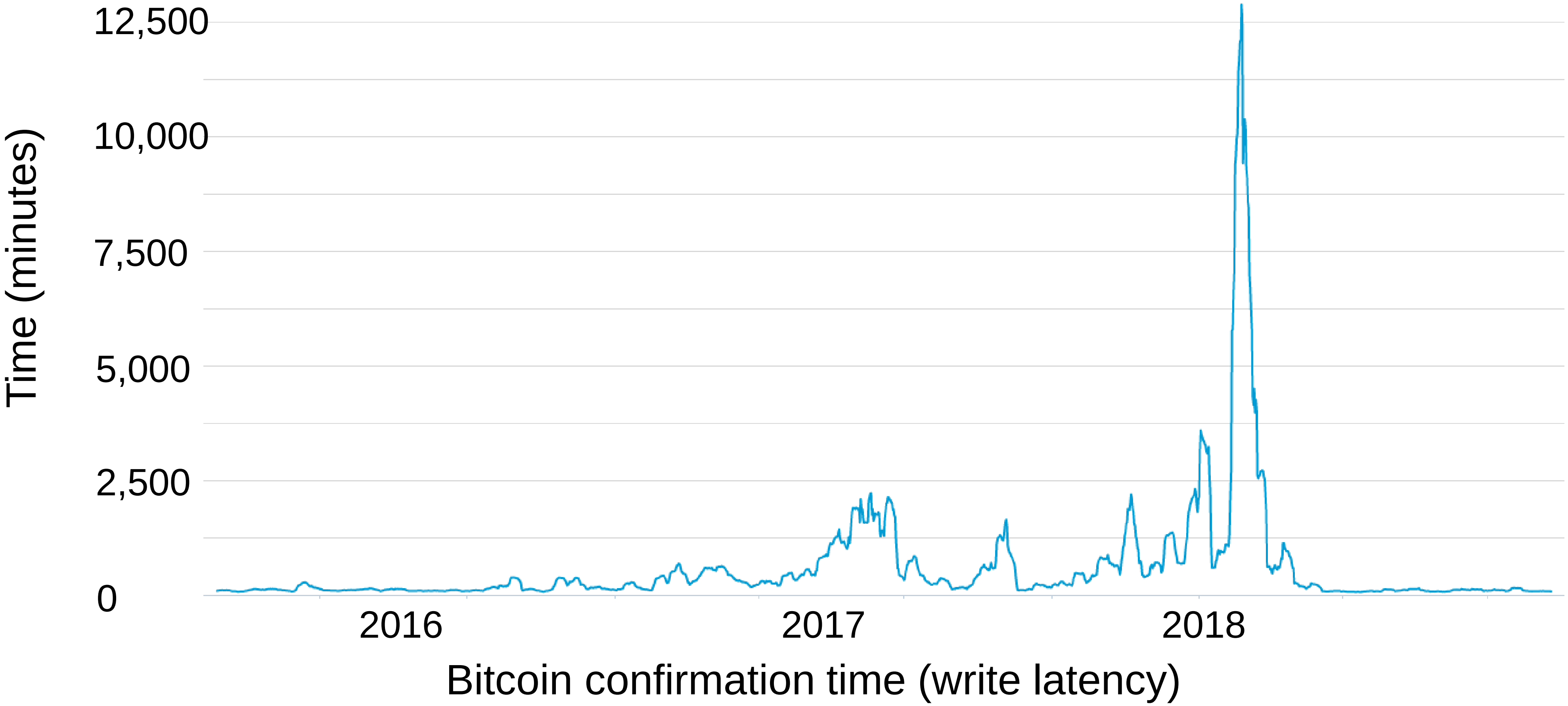
**Reaction times (Δ) require synchronous blockchain access:**

- **Assume:** parties can read/write within Δ
- **But**: blockchains are best-effort. No read/write latency bounds!

# Background: Reacting to roll-back attacks

**Reaction times (Δ) require synchronous blockchain access:**

- **Assume:** parties can read/write within Δ
- **But**: blockchains are best-effort. No read/write latency bounds!



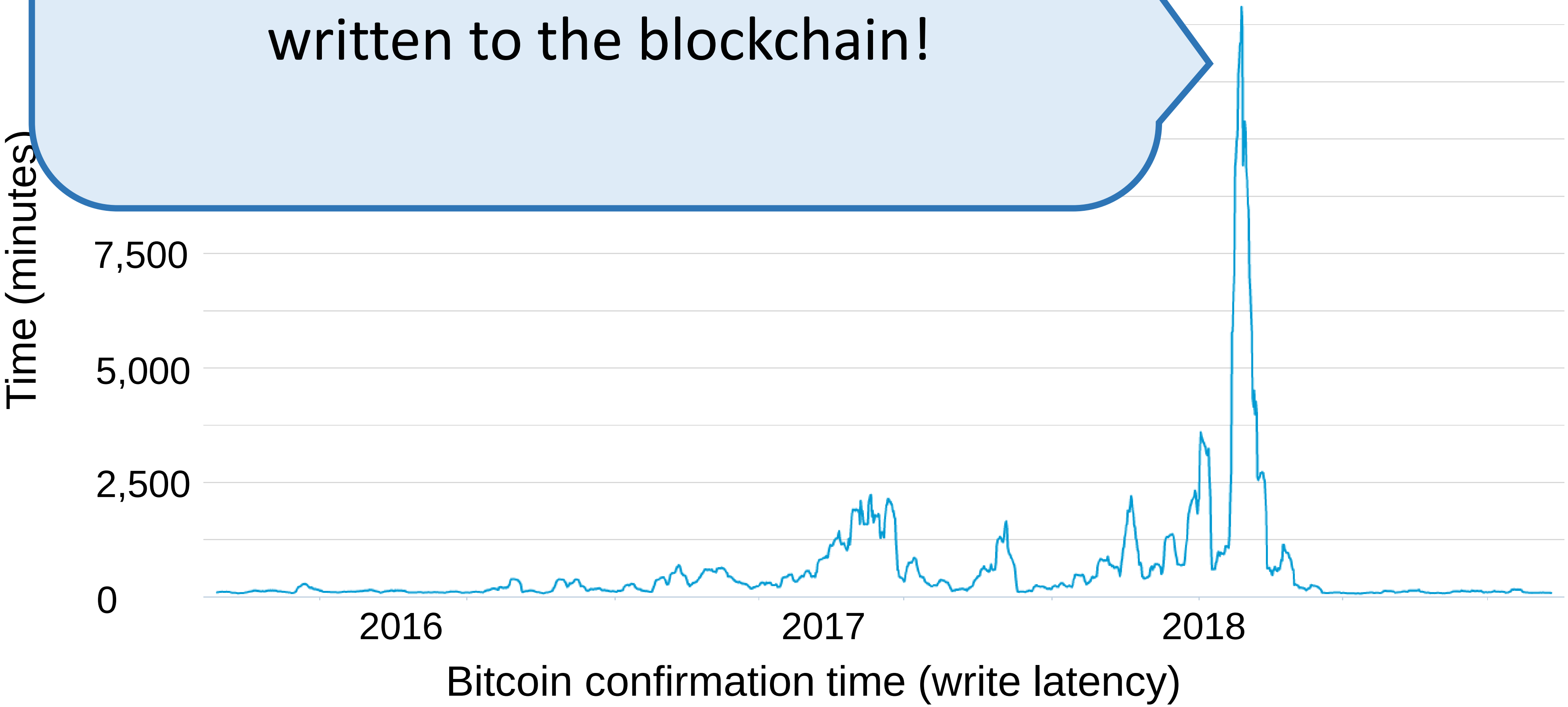
# Background: Reacting to roll-back attacks

R Access:

nds!

## Spam/Congestion attack!

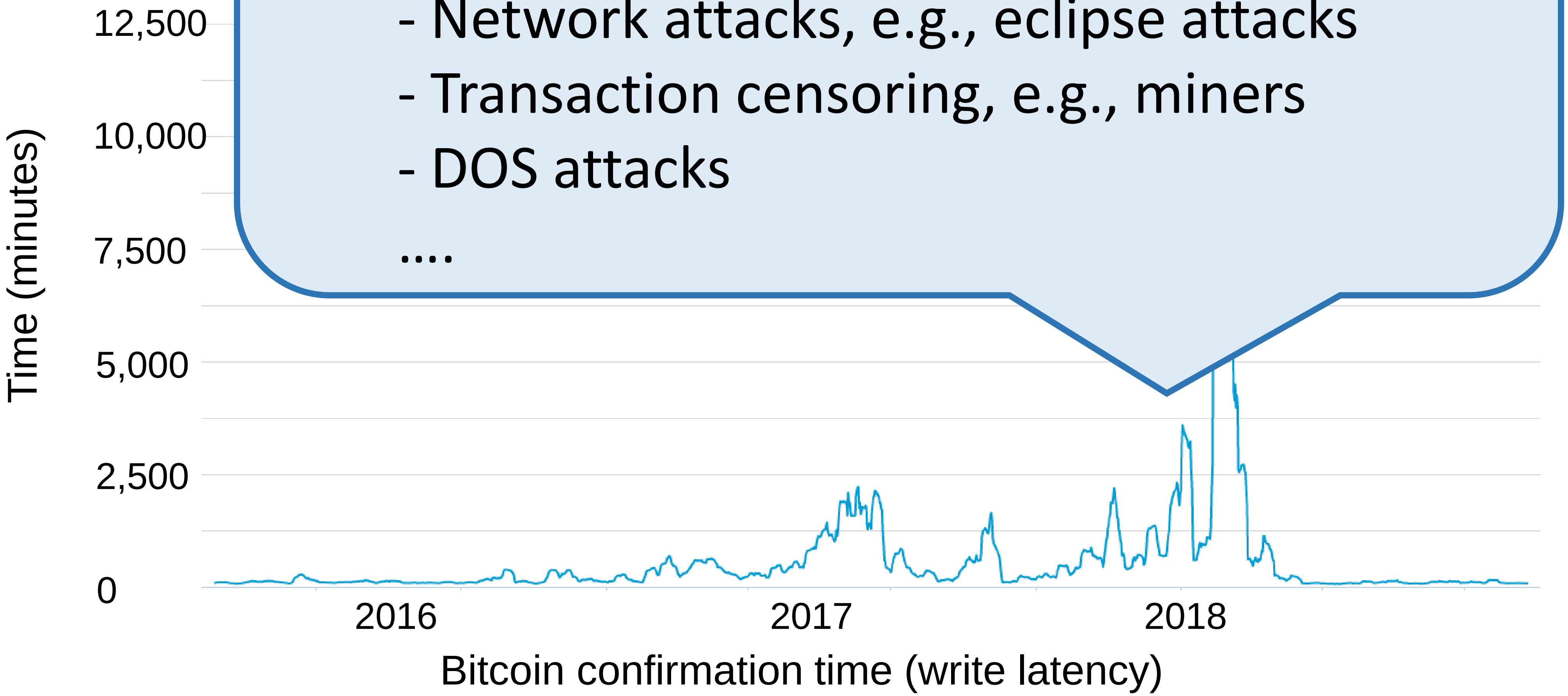
Transactions took **> 7 days** to be written to the blockchain!



# Background: Reacting to roll-back attacks

**Reaction t**

- **Assume:**
- **But**: Bloc



**Attackers can manipulate latencies:**

- Network attacks, e.g., eclipse attacks
- Transaction censoring, e.g., miners
- DOS attacks

….

# ll-back attacks

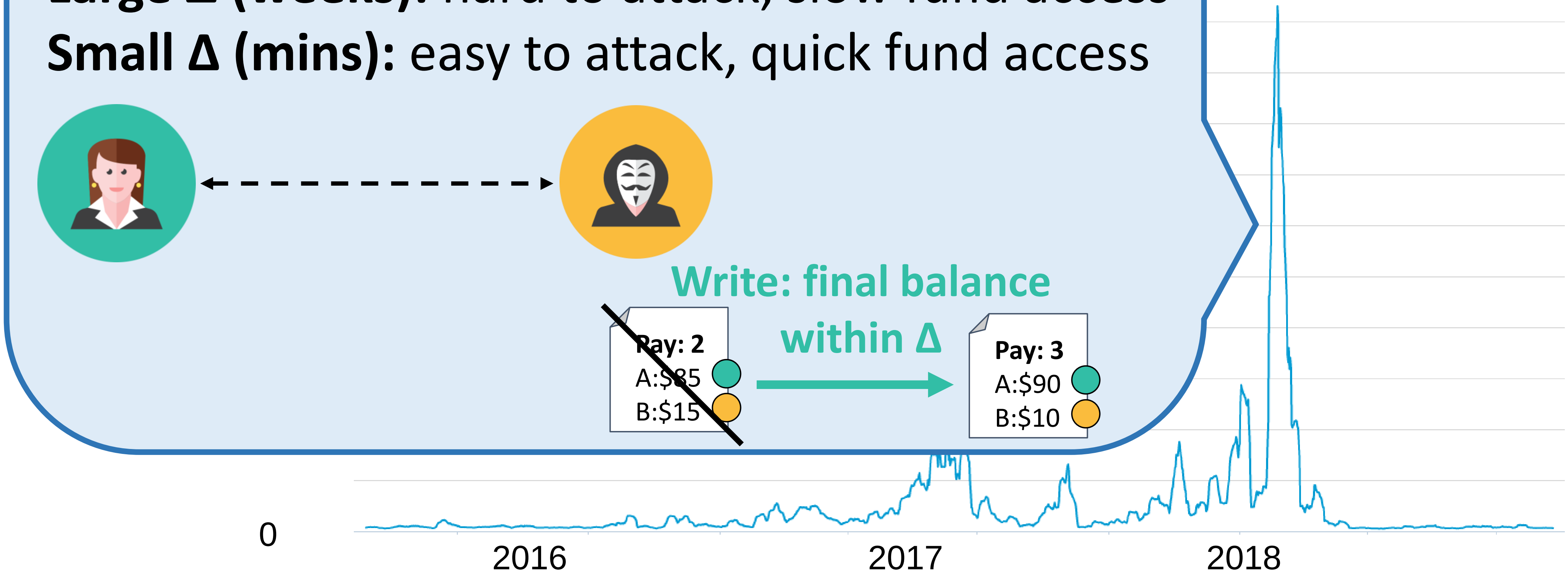
## What value for reaction time (Δ)?

**Trade-off:**

**Large Δ (weeks):** hard to attack, slow fund access

**Small Δ (mins):** easy to attack, quick fund access

nds!



Write: final balance within Δ

~~Pay: 2~~
~~A:$85~~
~~B:$15~~

Pay: 3
A:$90
B:$10

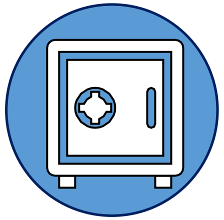
Bitcoin confirmation time (write latency)

# Teechain: Challenges and roadmap

**Asynchronous blockchain access** (no read/write latency bounds):

**Challenge 1:** removing the blockchain as root-of-trust (RoT)

**Idea:** ***treasury*** as new RoT for payments



Treasury

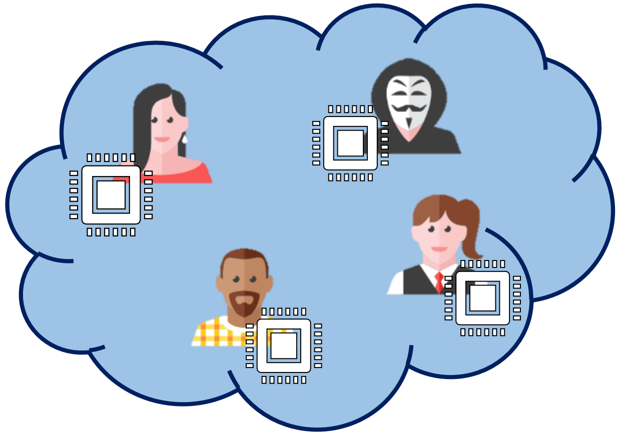
**Challenge 2:** realizing treasuries for blockchains

**Idea:** decentralized ***treasury committees***

**Idea:** ***trusted execution*** to secure committees

**Challenge 3**: consensus in treasury committees

**Idea:** ***force-freeze chain replication***

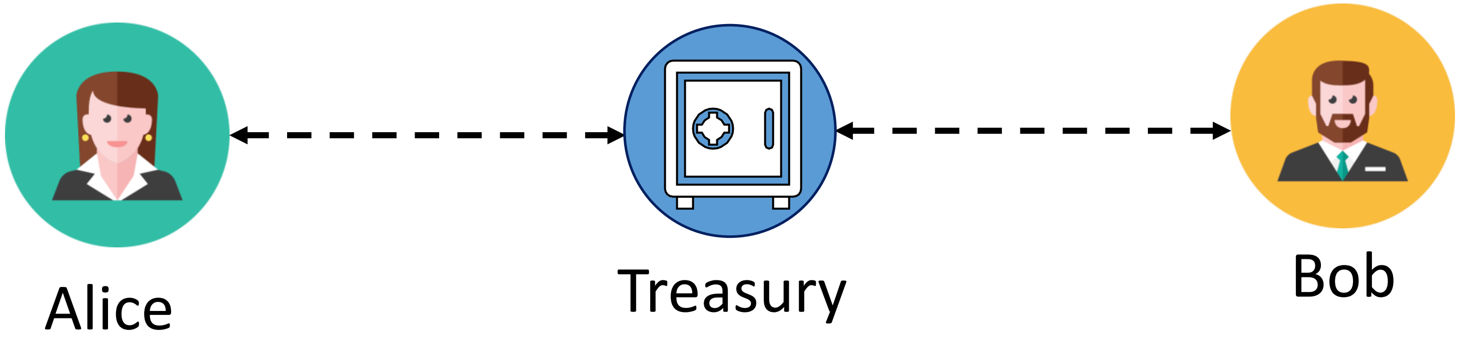


Treasury Committee

# Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT): **treasury**

- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!



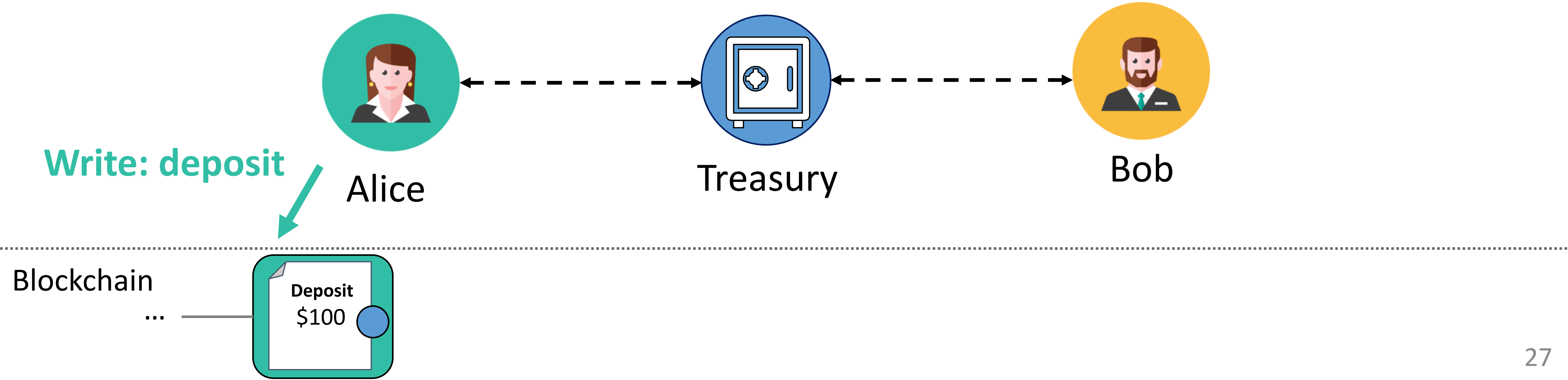
Blockchain

…

# Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT): **treasury**

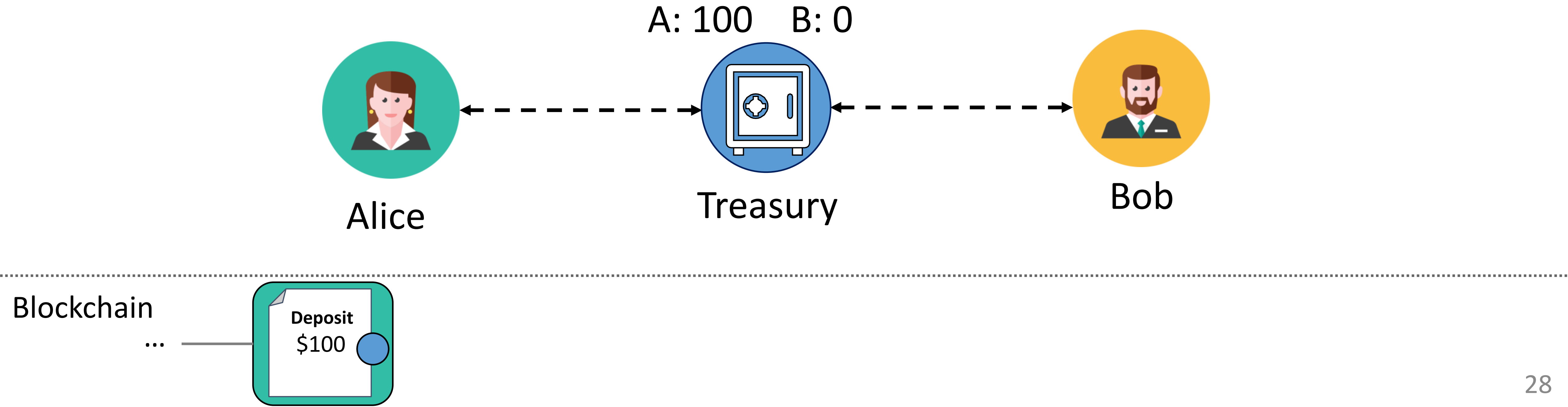
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!



# Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT): **treasury**

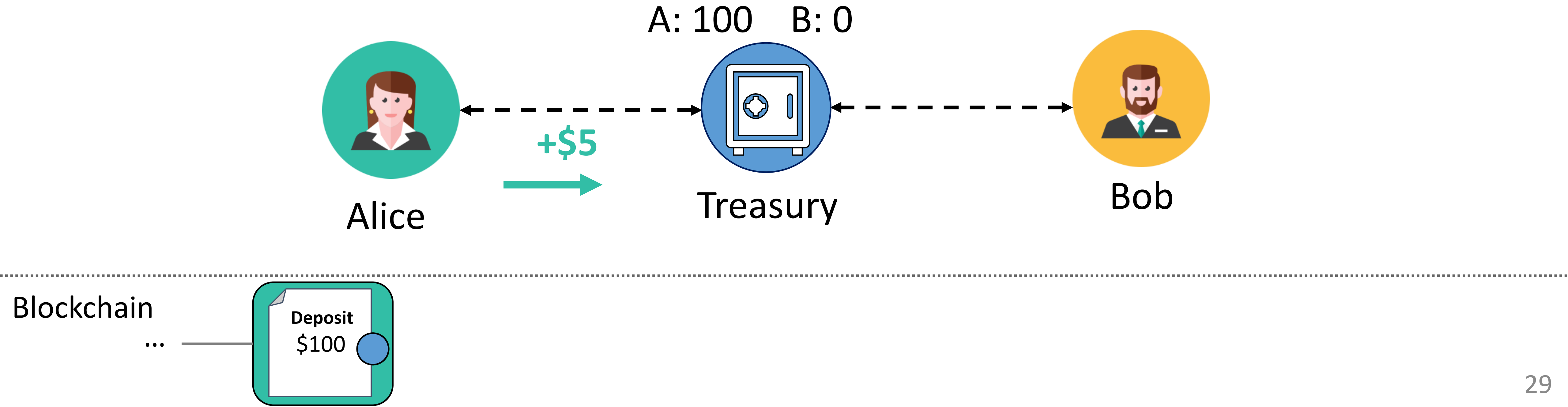
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!



# Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT): **treasury**

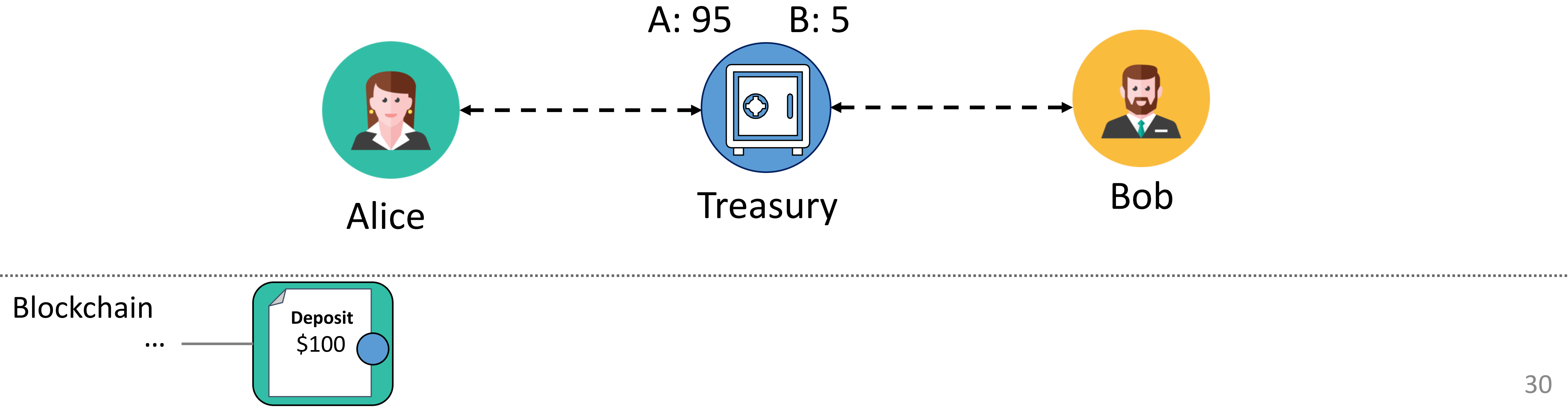
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!



# Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT): **treasury**

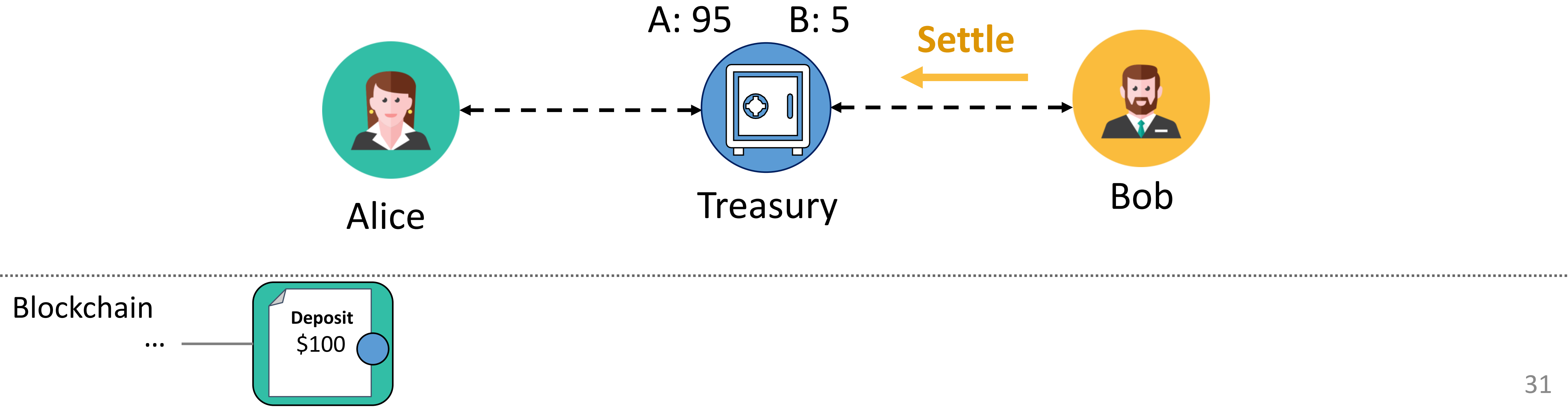
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!



# Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT): **treasury**

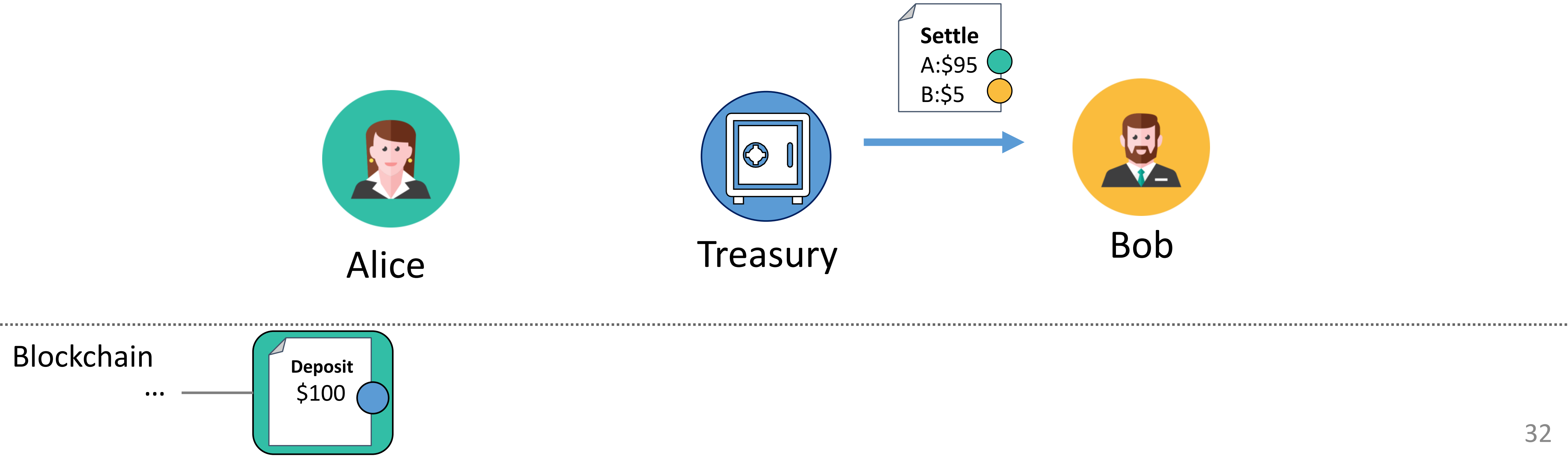
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!



# Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT): **treasury**

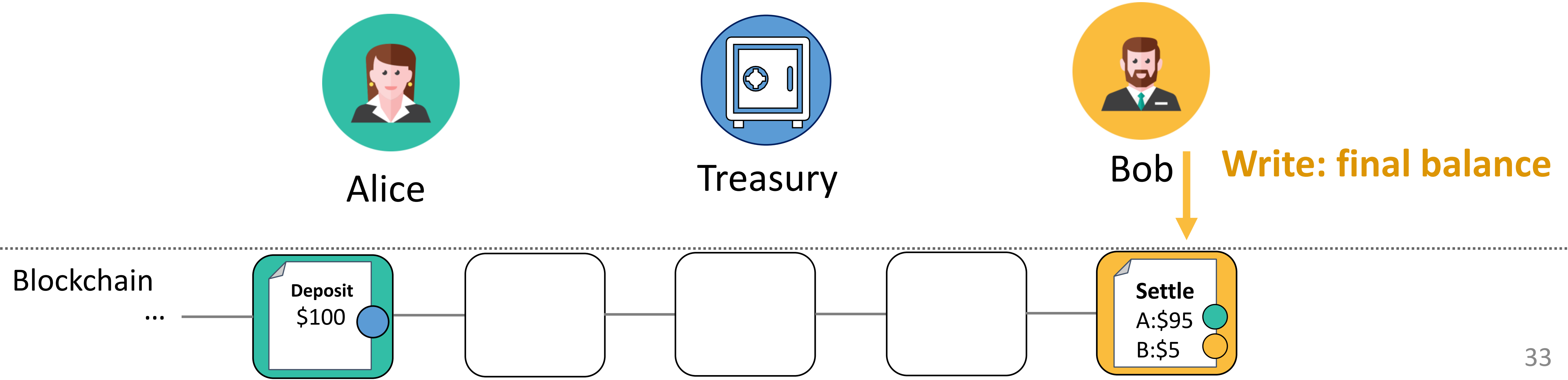
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!



# Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT): **treasury**

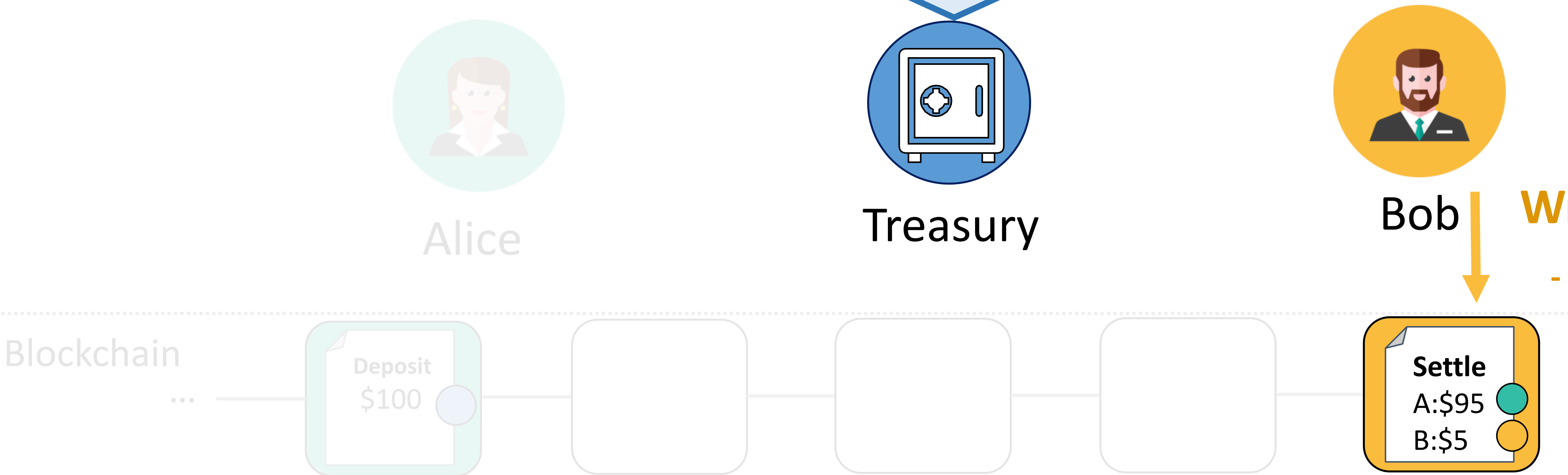
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!



# Challenge 1: Removing the blockchain as RoT

## Treasury prevents roll-backs!

No roll-backs → no reaction times:
<u>Asynchronous blockchain access</u>



**Write: final balance**

**- Unbounded latency!**

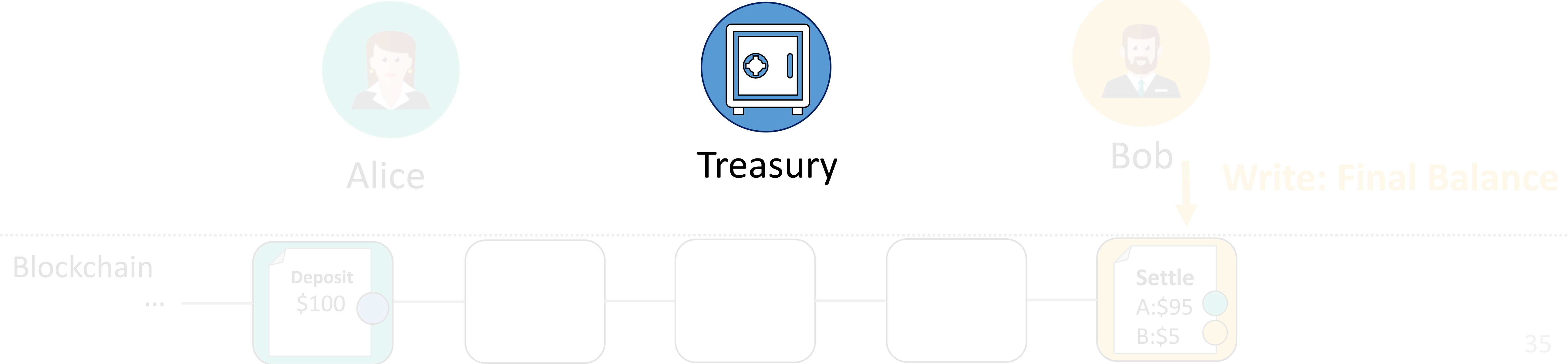
# Challenge 1: Removing the blockchain as RoT

Introduce another roo

- Controls funds, balan
- Prevents misbehavio
- Only settle channels c

## Can we realize treasuries?

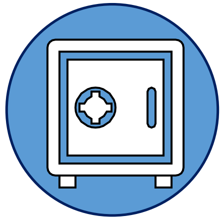
How do we realize treasuries for blockchains?



# Challenge 2: Realizing treasuries for blockchains

**Design treasury to**:

- Avoid absolute trust (parties are selfish!)
- Avoid centralization
- Integrate with most blockchains (e.g. no smart contracts!)



Treasury

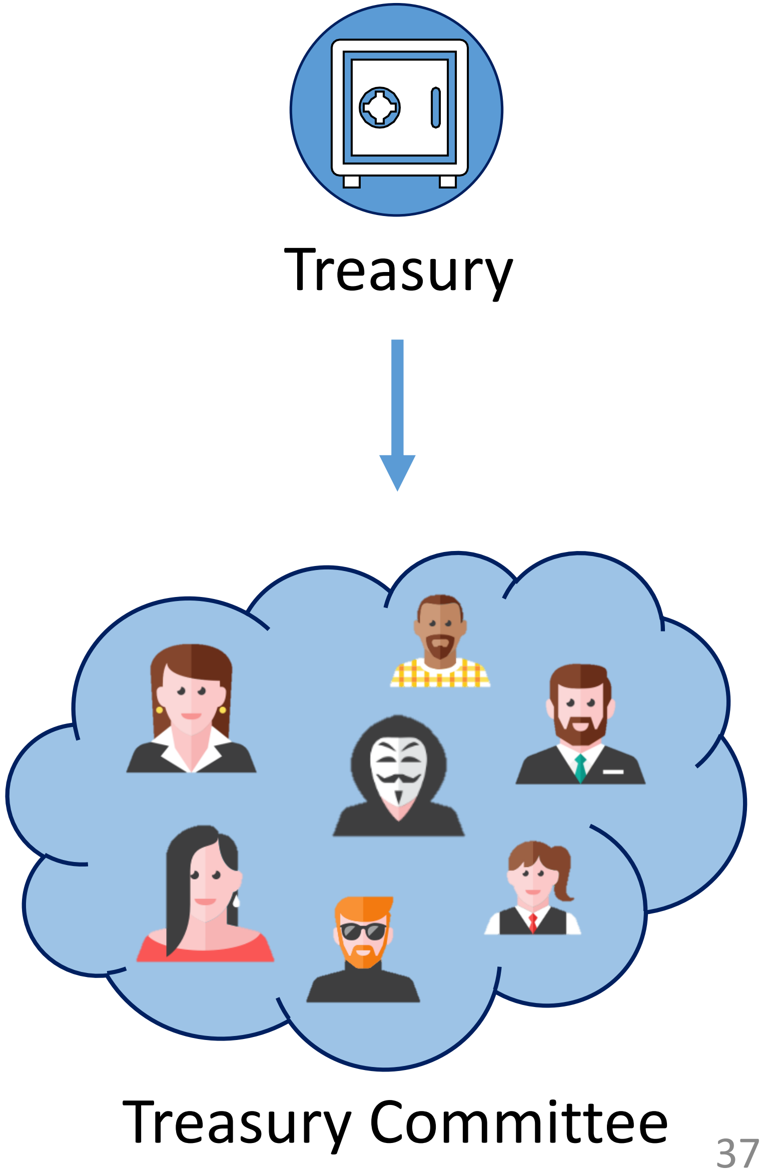
# Challenge 2: Realizing treasuries for blockchains

**Design treasury to**:

- Avoid absolute trust (parties are selfish!)
- Avoid centralization
- Integrate with most blockchains (e.g. no smart contracts!)

**Use a committee!**

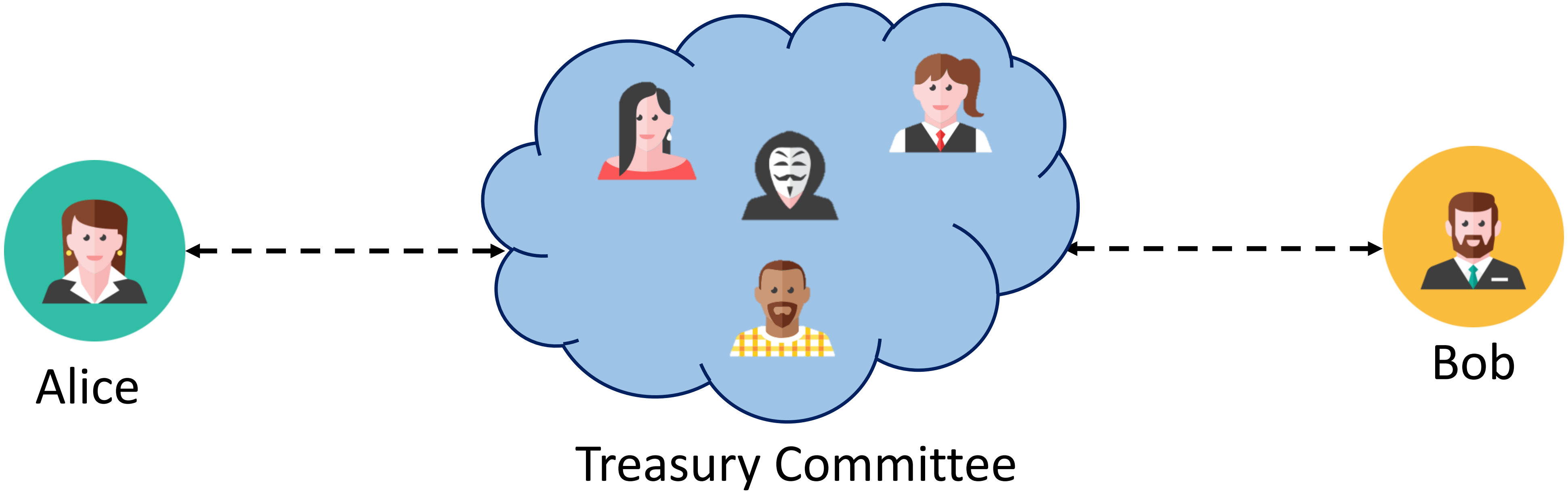
- **General solution**: well studied for blockchains
- **Decentralized**: distribute trust
- **Fault tolerant**: crash and Byzantine failures



# Challenge 2: Realizing treasuries for blockchains

**Treasury committee:**

- Choose **n** parties in the network
- Require **m** parties to agree before accessing funds
- Use **m-out-of-n** transactions



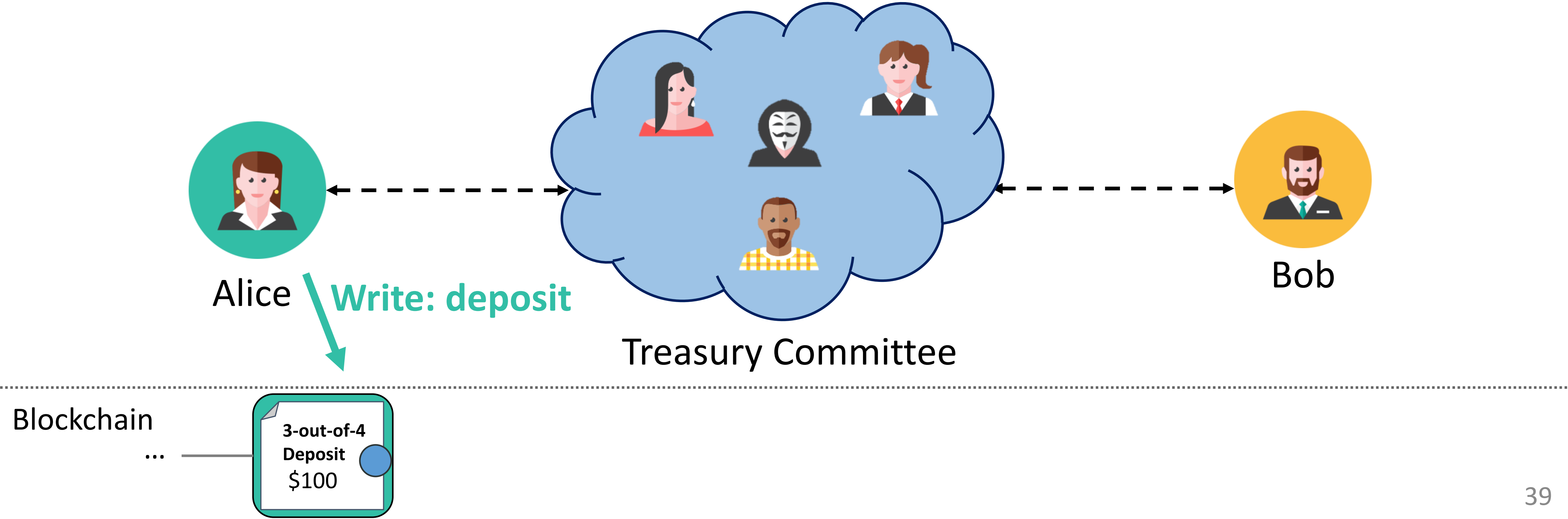
Blockchain

…

# Challenge 2: Realizing treasuries for blockchains

**Treasury committee:**

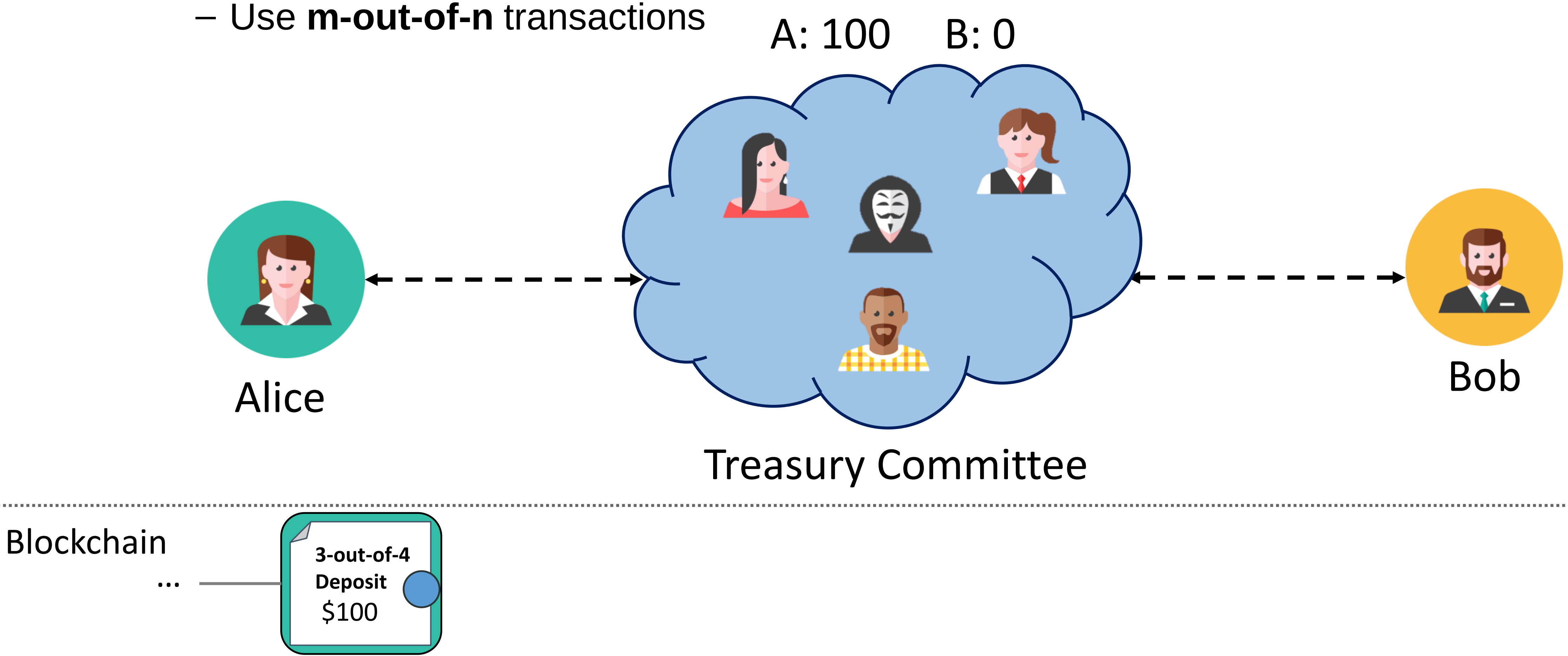
- Choose **n** parties in the network
- Require **m** parties to agree before accessing funds
- Use **m-out-of-n** transactions



# Challenge 2: Realizing treasuries for blockchains

**Treasury committee:**

- Choose **n** parties in the network
- Require **m** parties to agree before accessing funds
- Use **m-out-of-n** transactions



A: 100 B: 0

Alice

Treasury Committee

Bob

Blockchain

...

3-out-of-4

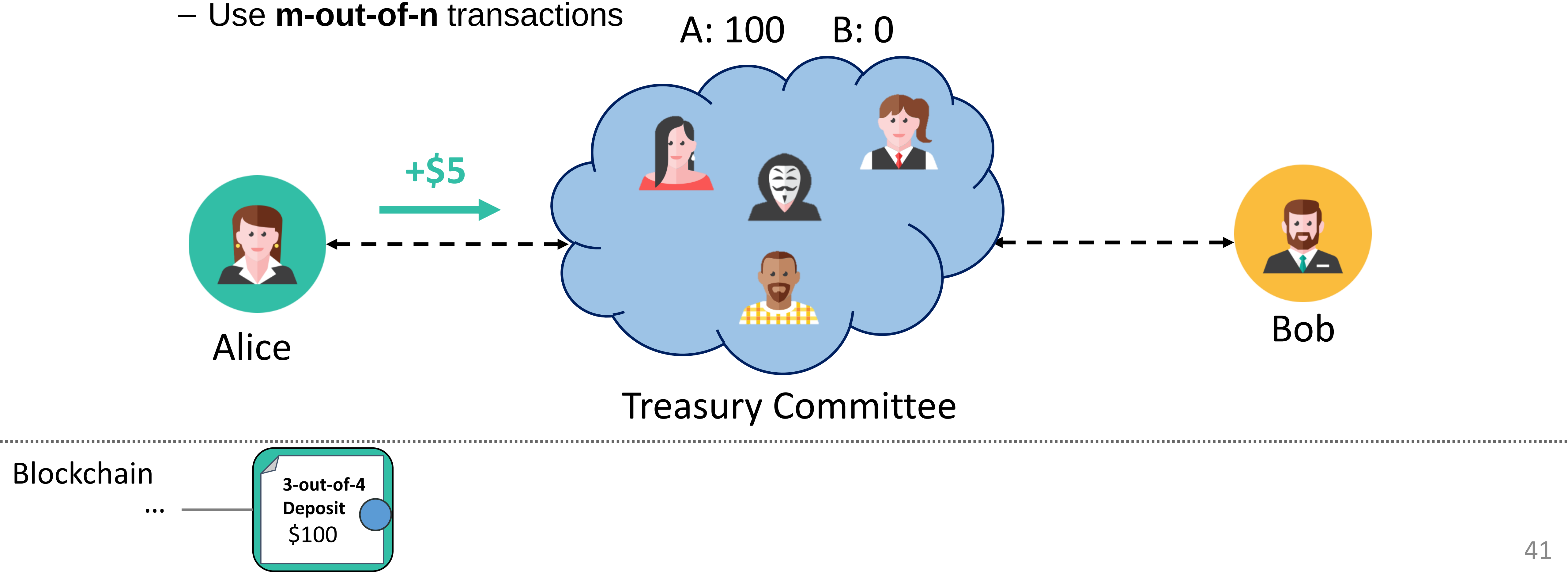
Deposit

$100

# Challenge 2: Realizing treasuries for blockchains

**Treasury committee:**

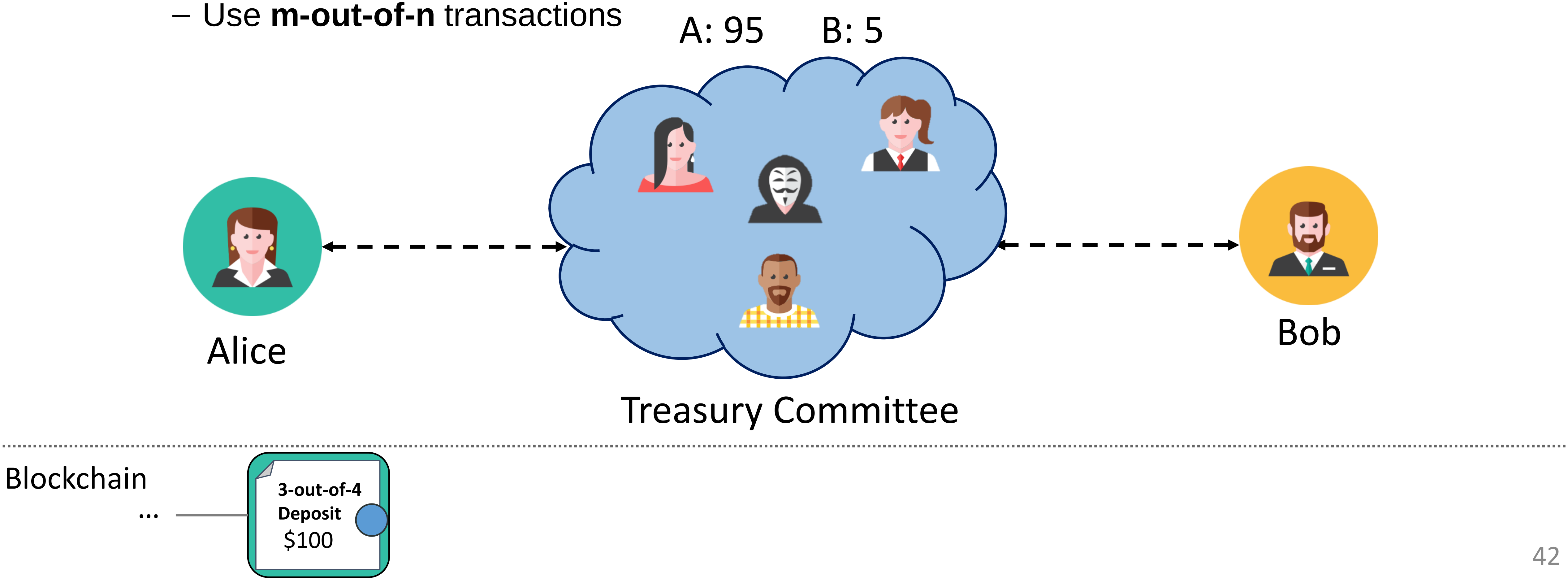
- Choose **n** parties in the network
- Require **m** parties to agree before accessing funds
- Use **m-out-of-n** transactions



# Challenge 2: Realizing treasuries for blockchains

**Treasury committee:**

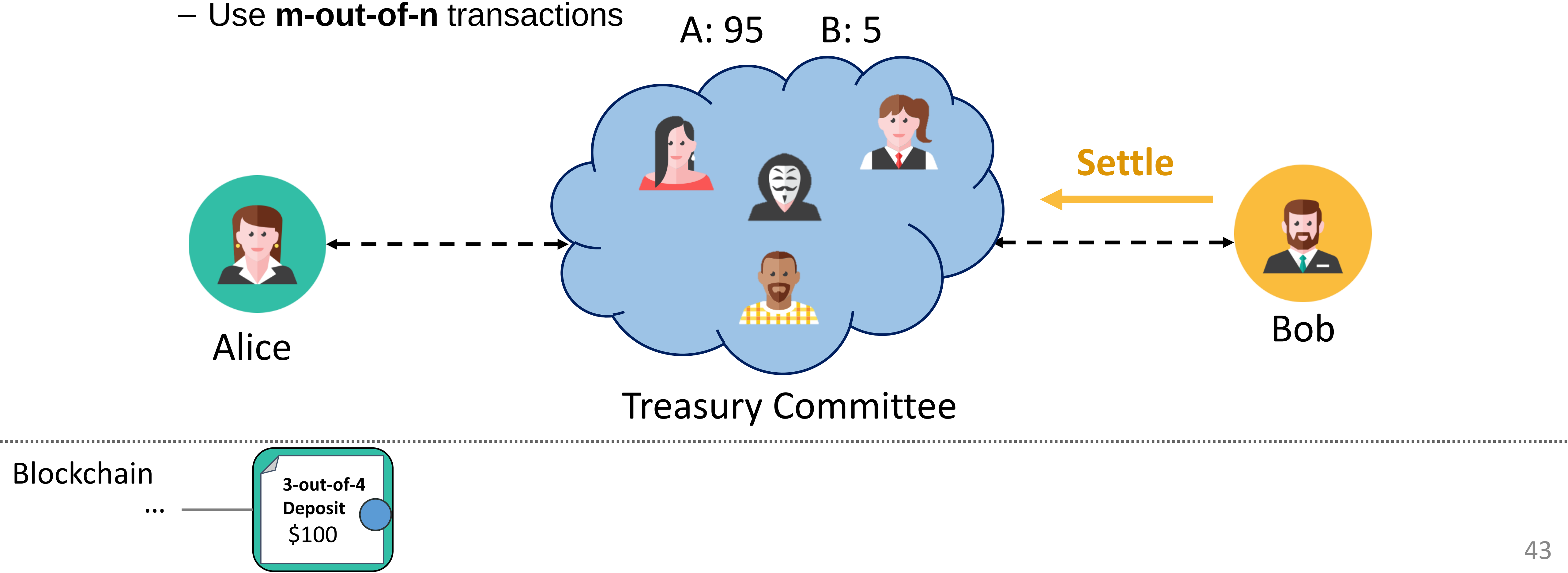
- Choose **n** parties in the network
- Require **m** parties to agree before accessing funds
- Use **m-out-of-n** transactions



# Challenge 2: Realizing treasuries for blockchains

**Treasury committee:**

- Choose **n** parties in the network
- Require **m** parties to agree before accessing funds
- Use **m-out-of-n** transactions

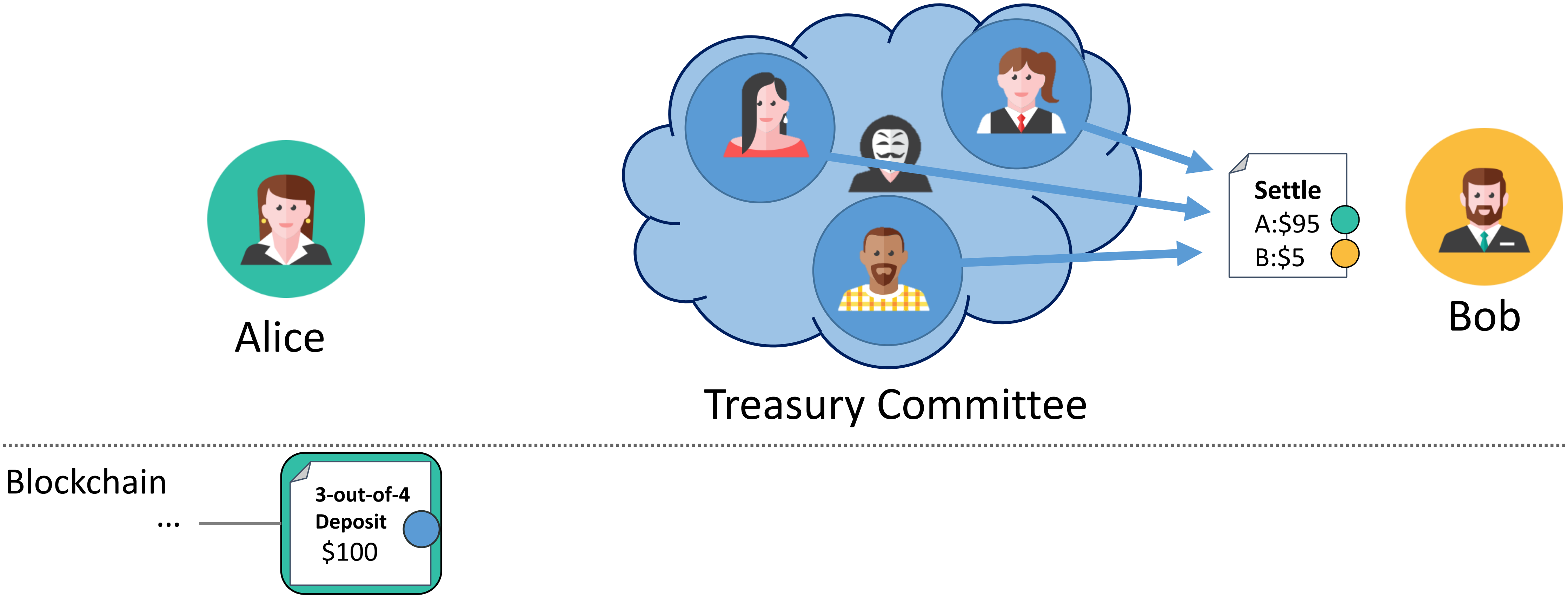


# Challenge 2: Realizing treasuries for blockchains

## Treasury committee:

- Choose **n** parties in the network
- Require **m** parties to agree before accessing funds
- Use **m-out-of-n** transactions

A: 95 B: 5



Alice

Treasury Committee

**Settle**
A:$95
B:$5

Bob

Blockchain

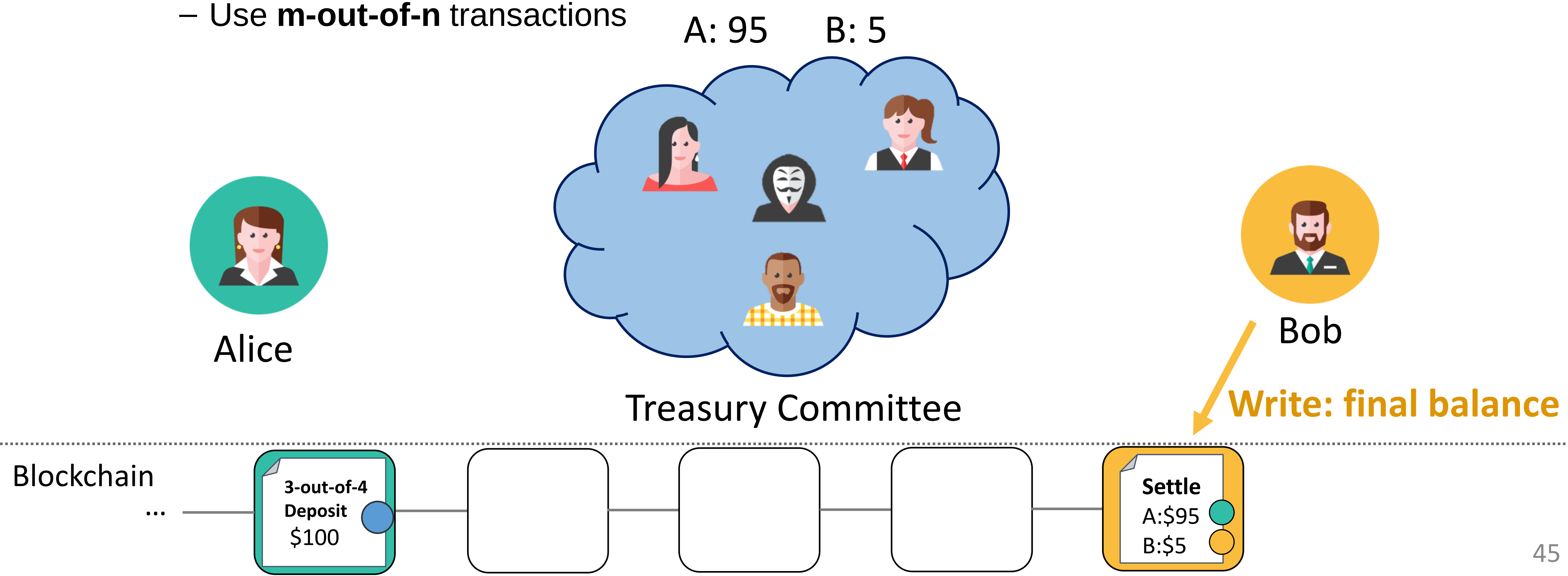
...

**3-out-of-4 Deposit**
$100

# Challenge 2: Realizing treasuries for blockchains

**Treasury committee:**

- Choose **n** parties in the network
- Require **m** parties to agree before accessing funds
- Use **m-out-of-n** transactions



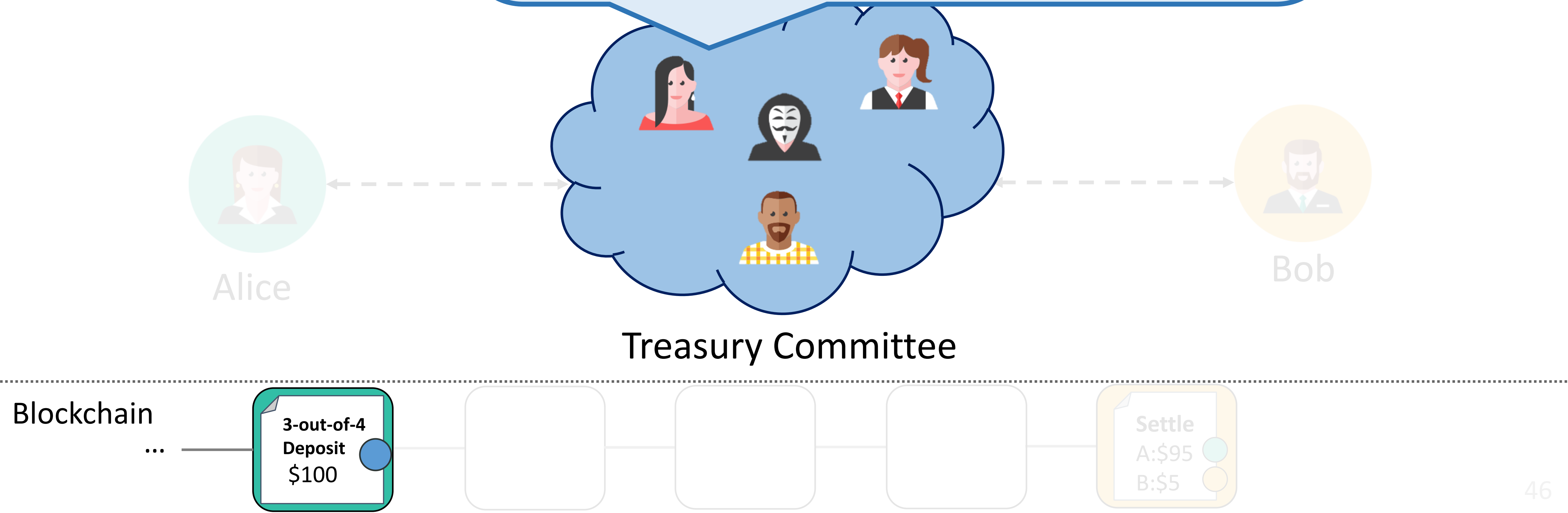
# Challenge 2 chains

Treasury commit

- Choose **n** parties
- Require **m** parties
- Use **m-out-of-n** t

## Trust is distributed!

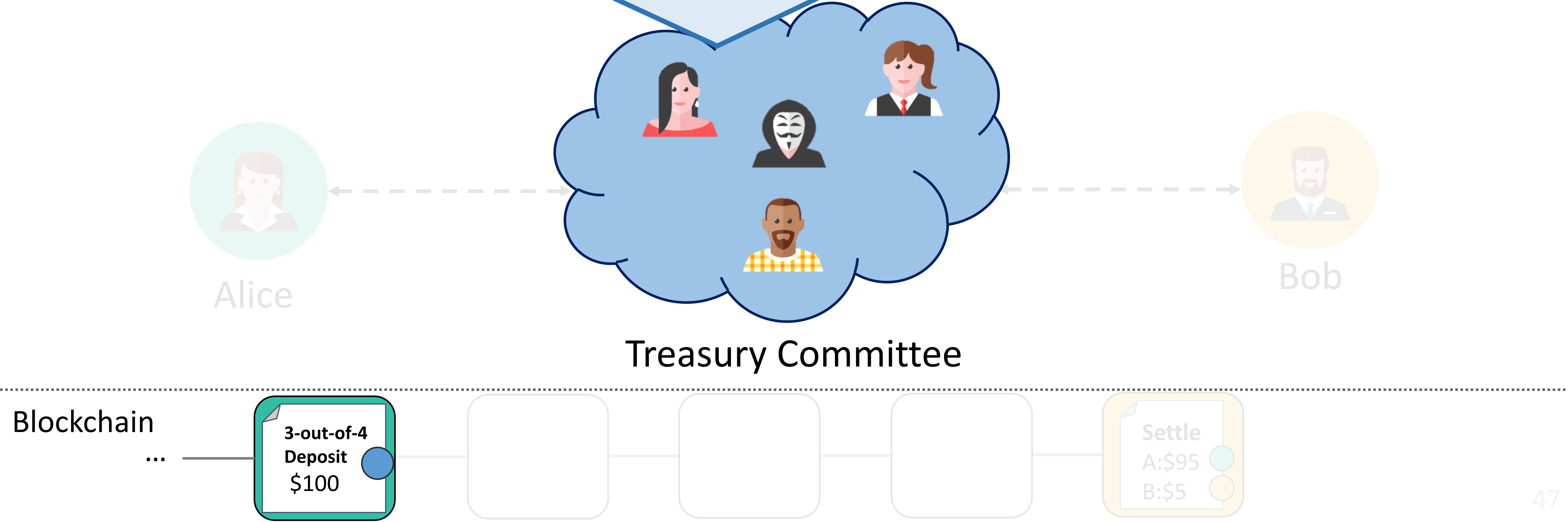
**m** treasuries must collude together to steal the deposit!



ries for blockchains

## Treasury committee size?

How large should **m** and **n** be?



# Challenge 2: Realizing treasuries for blockchains

**Existing solutions**:

- **Large committees** for security: e.g. Elastico, Algorand..
- **But this is difficult at scale!** (consensus..)

# Challenge 2: Realizing treasuries for blockchains

**Existing solutions**:

- **Large committees** for security: e.g. Elastico, Algorand..
- **But this is difficult at scale!** (consensus..)

**Smaller committees? Use trusted execution!**

- Confidentiality + integrity guarantees
- Only trust hardware and manufacturer (don’t trust people!)

**Many trusted execution environments (TEEs)**:

- **Commodity:** Intel SGX, ARM TrustZone, AMD SEV..
- **Up-and-coming**: KeyStone Enclave, Multizone, OP-TEE, Sanctum..

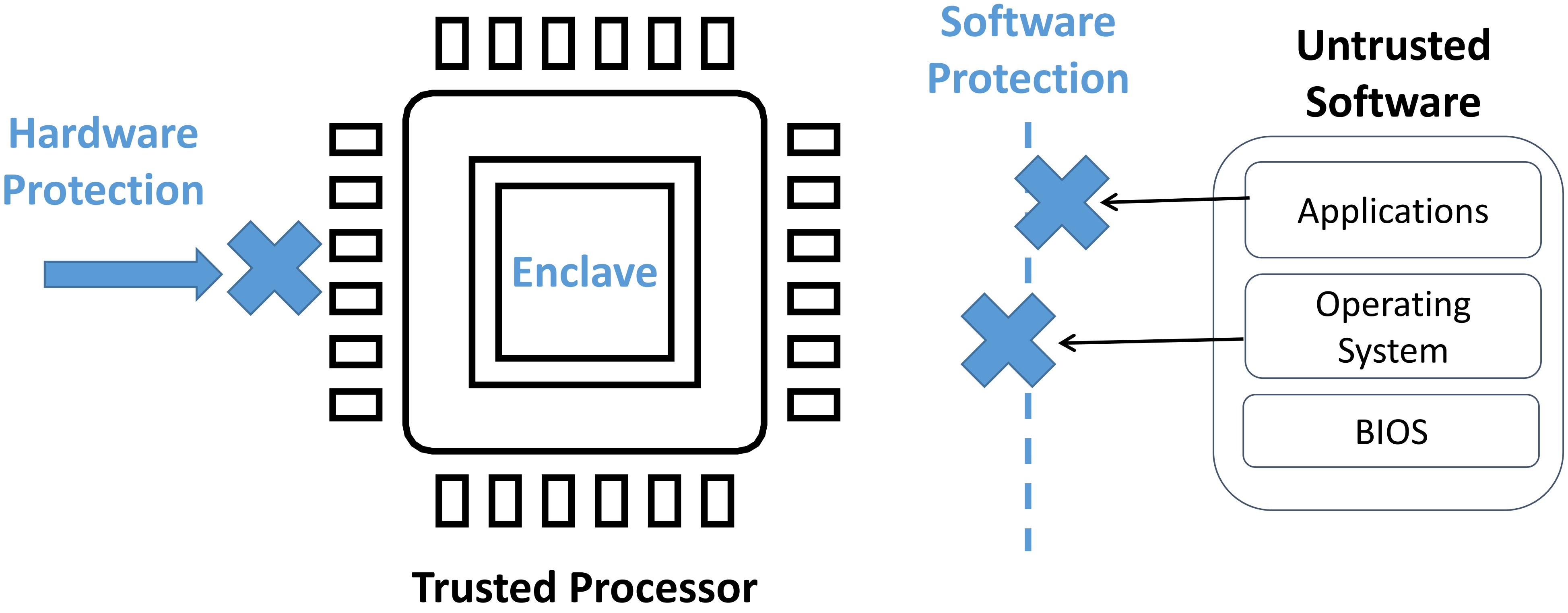
# Background: Intel software guard extensions (SGX)

Intel SGX provides **confidentiality and integrity** for **enclaves**:

- **Software protection**: OS, BIOS, other applications
- **Physical attacks**: DRAM, Disk, System Bus

intel

SGX



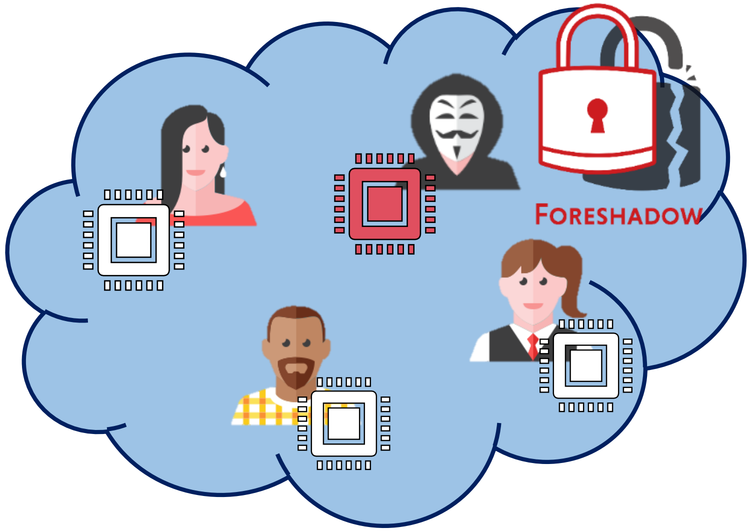
# Challenge 2: Realizing treasuries for blockchains

## Use TEEs to secure committee members

- **Increase attack costs**: reduce committee size

## TEEs are not silver bullets:

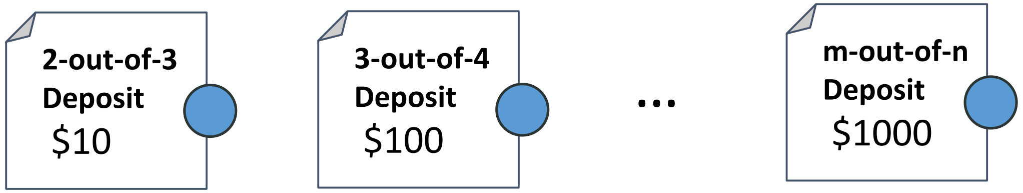
- **Existing attacks:** e.g. Foreshadow [USENIX SEC'18]
- **Combine TEEs + committees**: defence-in-depth



Treasury Committee

## "Configurable security" per deposit:

- **Parties decide m-out-of-n:** no "one size fits all"
- **TEE heterogeneity:** avoid centralization/attacks
- **Weigh-up deposit risk: e.g.,**
  - **$10:** 2-out-of-3 committee
  - **$100:** 3-out-of-4 committee



**2-out-of-3 Deposit** $10

**3-out-of-4 Deposit** $100

…

**m-out-of-n Deposit** $1000

# Challenge 3: Consensus in treasury committees

**How do we maintain treasury agreement?**

- Peer-to-peer network → not fully connected (e.g. NATs and firewalls)

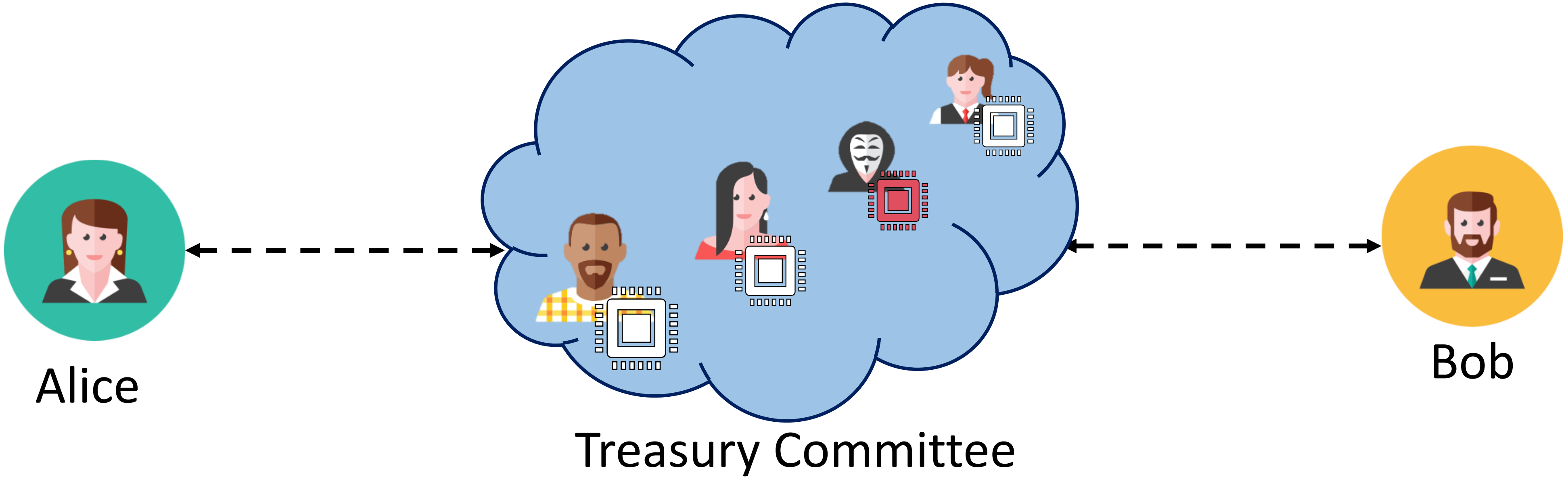
# Challenge 3: Consensus in treasury committees

## How do we maintain treasury agreement?

– Peer-to-peer network → not fully connected (e.g. NATs and firewalls)

## **Use chain replication**:

– **Strong consistency**: using a chain topology
– **Efficient:** update in **O(n) messages**
– **Easy to reason about:** avoid bugs!



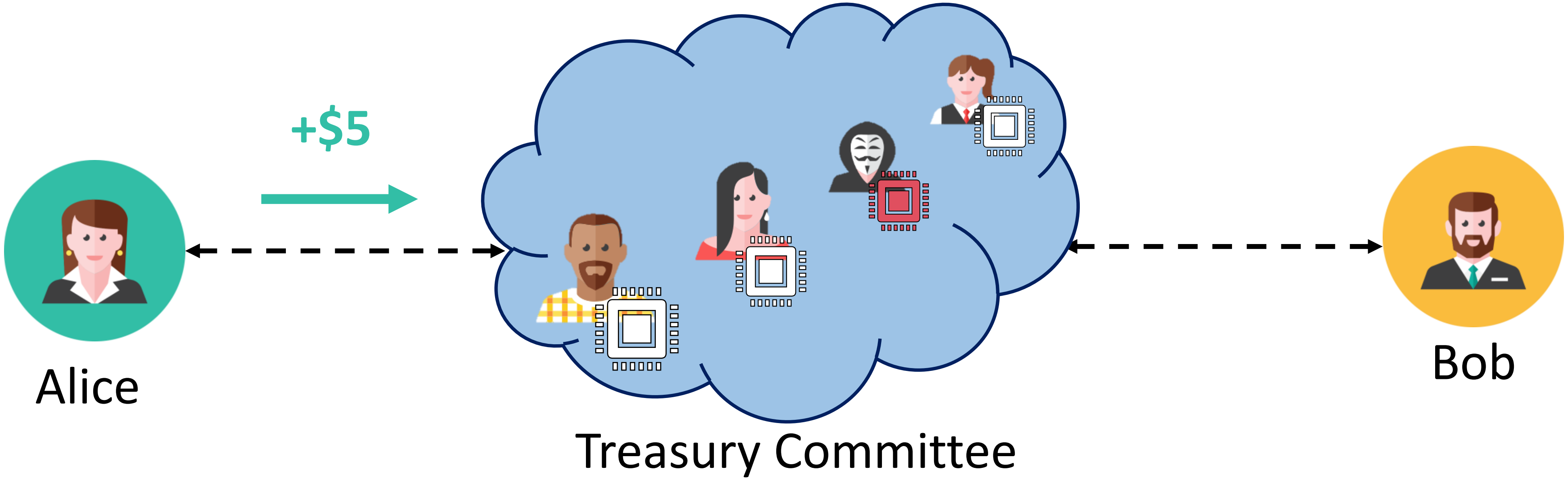
# Challenge 3: Consensus in treasury committees

## How do we maintain treasury agreement?

- Peer-to-peer network → not fully connected (e.g. NATs and firewalls)

## Use chain replication:

- **Strong consistency**: using a chain topology
- **Efficient:** update in **O(n) messages**
- **Easy to reason about:** avoid bugs!



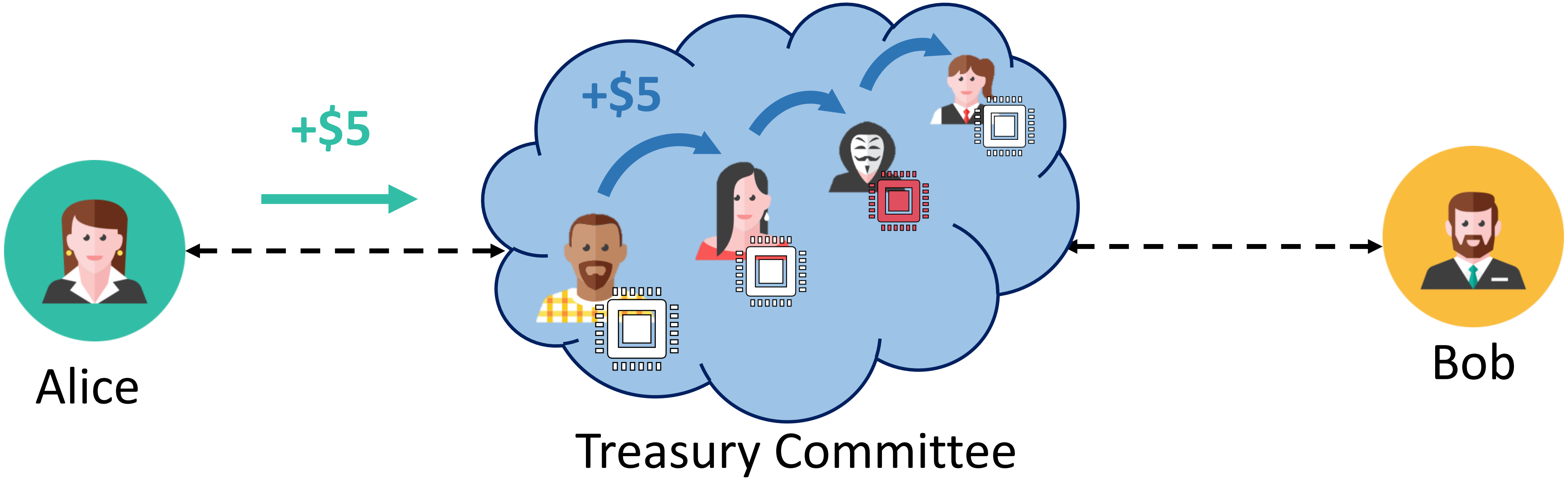
# Challenge 3: Consensus in treasury committees

**How do we maintain treasury agreement?**

- Peer-to-peer network → not fully connected (e.g. NATs and firewalls)

**Use chain replication**:

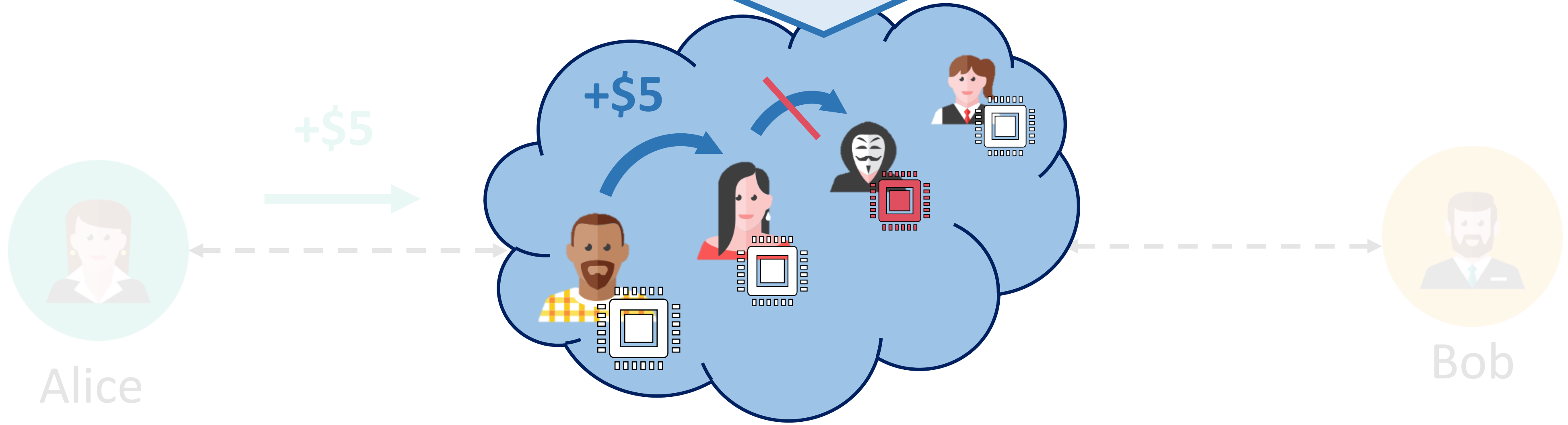
- **Strong consistency**: using a chain topology
- **Efficient:** update in **O(n) messages**
- **Easy to reason about:** avoid bugs!



# Challenge 3: Consensus in treasury committees

## What about failures?

Failures allow roll-back/replay attacks:
Introduce **force-freeze chain replication**
(see the paper!)

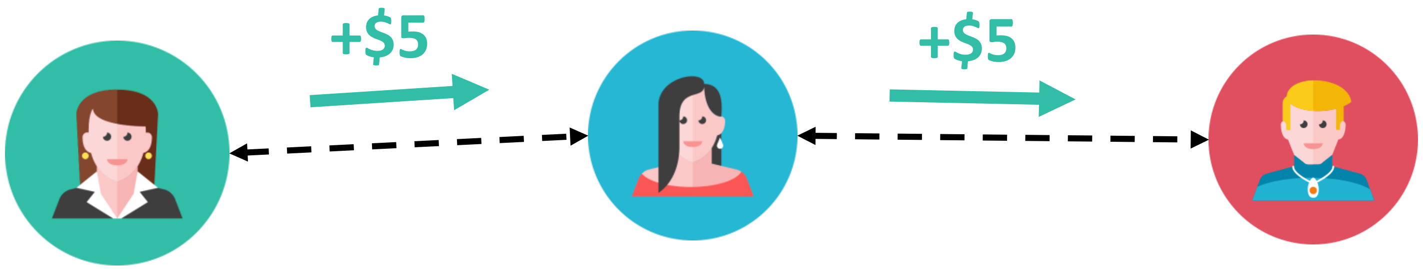


Treasury Committee

# See the paper!

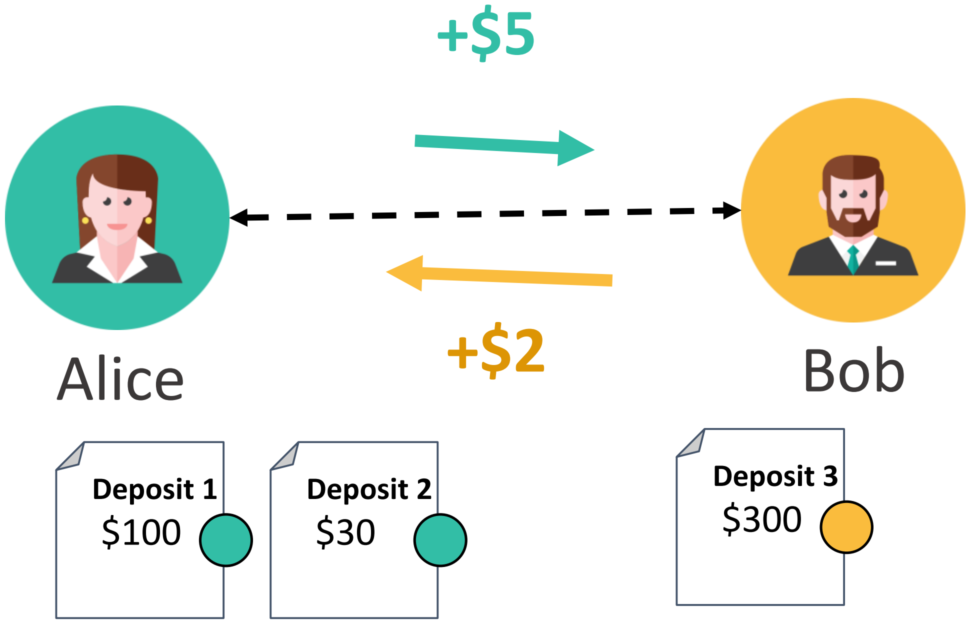
**Multi-hop protocol**

- Multi-phase commit



**Dynamic fund deposits**

- Add/remove funds dynamically!



**More features/optimizations!**

# Teechain: Implementation

**Teechain Network**:

- Bitcoin BTC blockchain (ported **Bitcoin core**)
- Intel SGX (20k C++ LoC inside TEE)
- 65k untrusted C++ LoC

**Open-source** (available and functional badges)

- **Github**: https://github.com/lsds/Teechain
- **Visit us**: teechain.network

# Teechain: Evaluation

Evaluation questions:

1. How well do **payment channels** perform?
2. How well do **multi-hop payments** perform?
3. Does Teechain **scale out**?

Baseline comparison:

- State of the art **Lightning Network** for Bitcoin
- Requires **synchronous blockchain access**

Experimental setup:

- 35 SGX machines across London, New York and Haifa
- Intel Xeon E3-1280 v5 32GB RAM

# How well do Payment Channels perform?

Payment channel: London -- New York

- Maximum **throughput** (tx/second) and **latency** (ack)
- Vary committee sizes (**n** members: London, New York, Haifa)

# How well do Payment Channels perform?

Payment channel: London -- New York

– Maximum **throughput** (tx/second) and **latency** (ack)

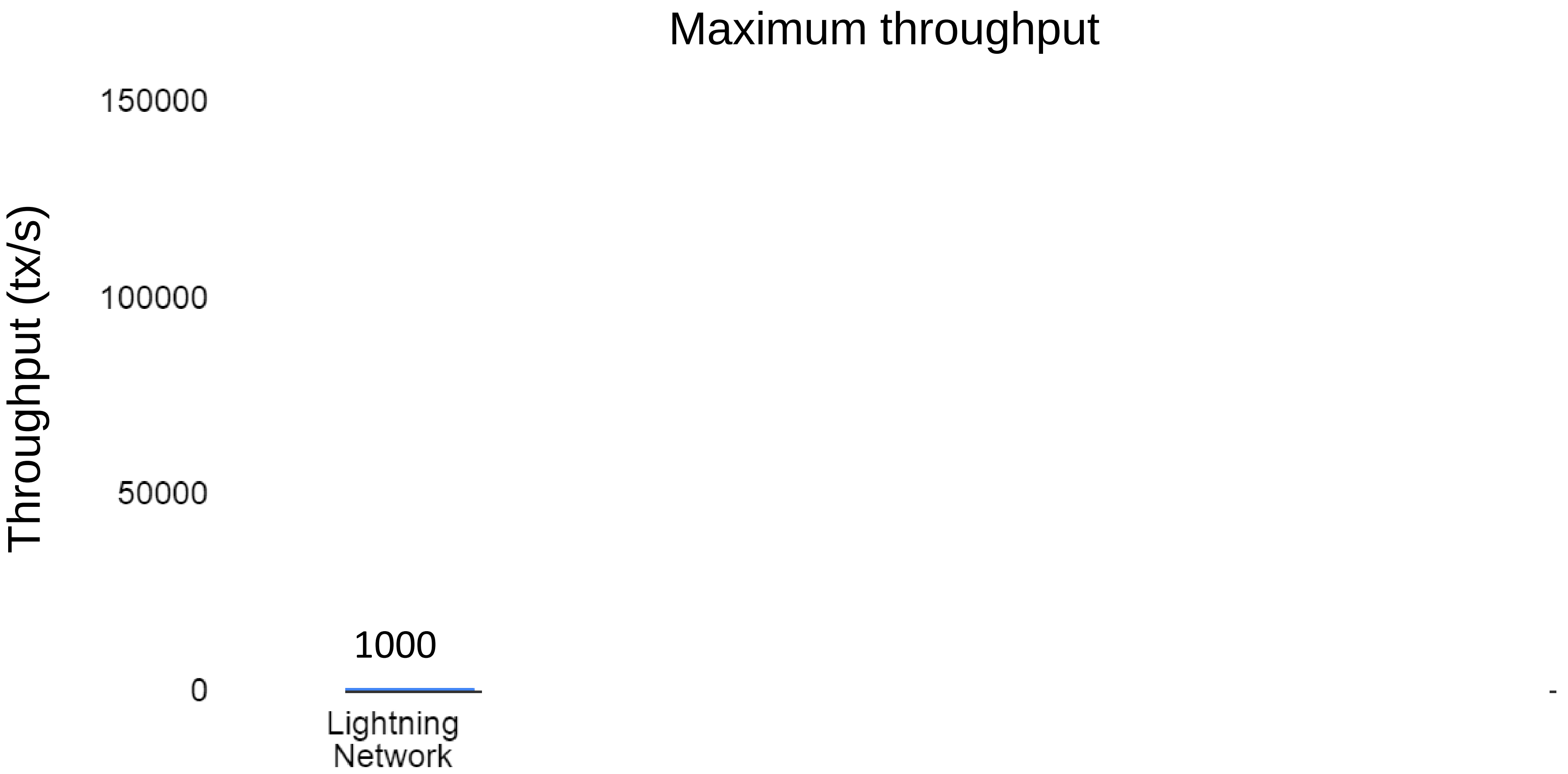
– Vary committee sizes (**n** members: London, New York, Haifa)

Maximum throughput

# How well do Payment Channels perform?

Payment channel: London -- New York

- Maximum **throughput** (tx/second) and **latency** (ack)
- Vary committee sizes (**n** members: London, New York, Haifa)

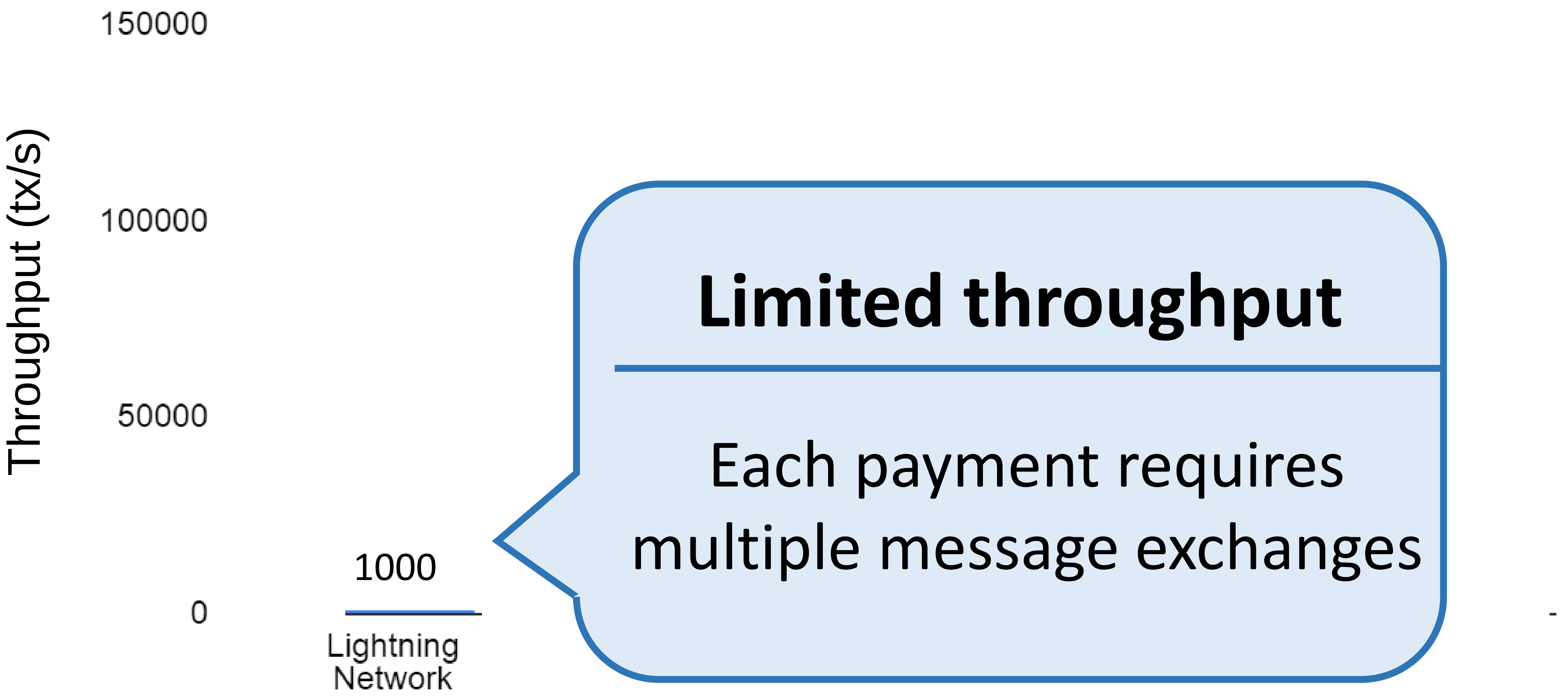


# How well do Payment Channels perform?

Payment channel: London -- New York

- Maximum **throughput** (tx/second) and **latency** (ack)
- Vary committee sizes (**n** members: London, New York, Haifa)

Maximum throughput



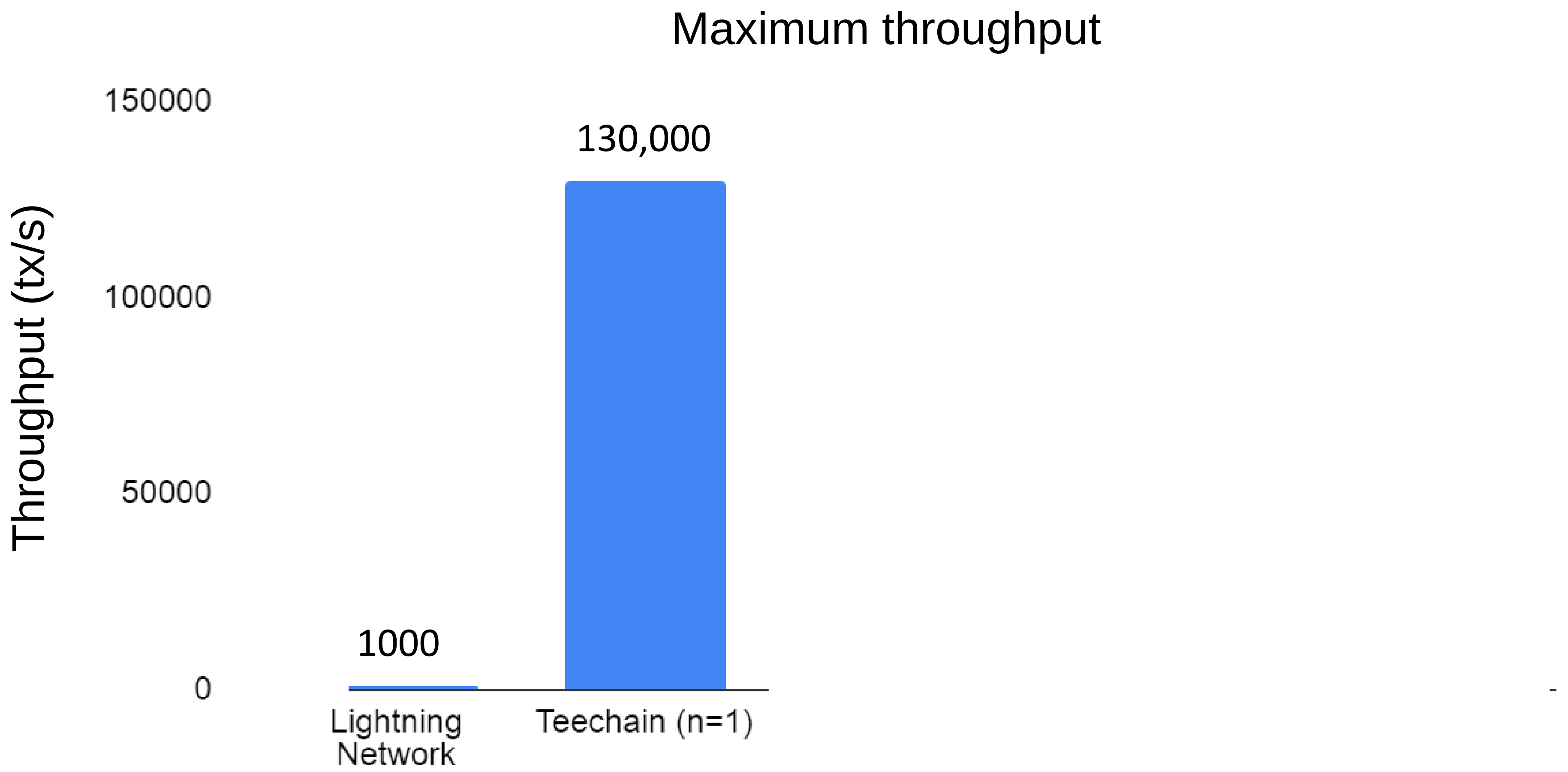
**Limited throughput**

Each payment requires multiple message exchanges

# How well do Payment Channels perform?

Payment channel: London -- New York

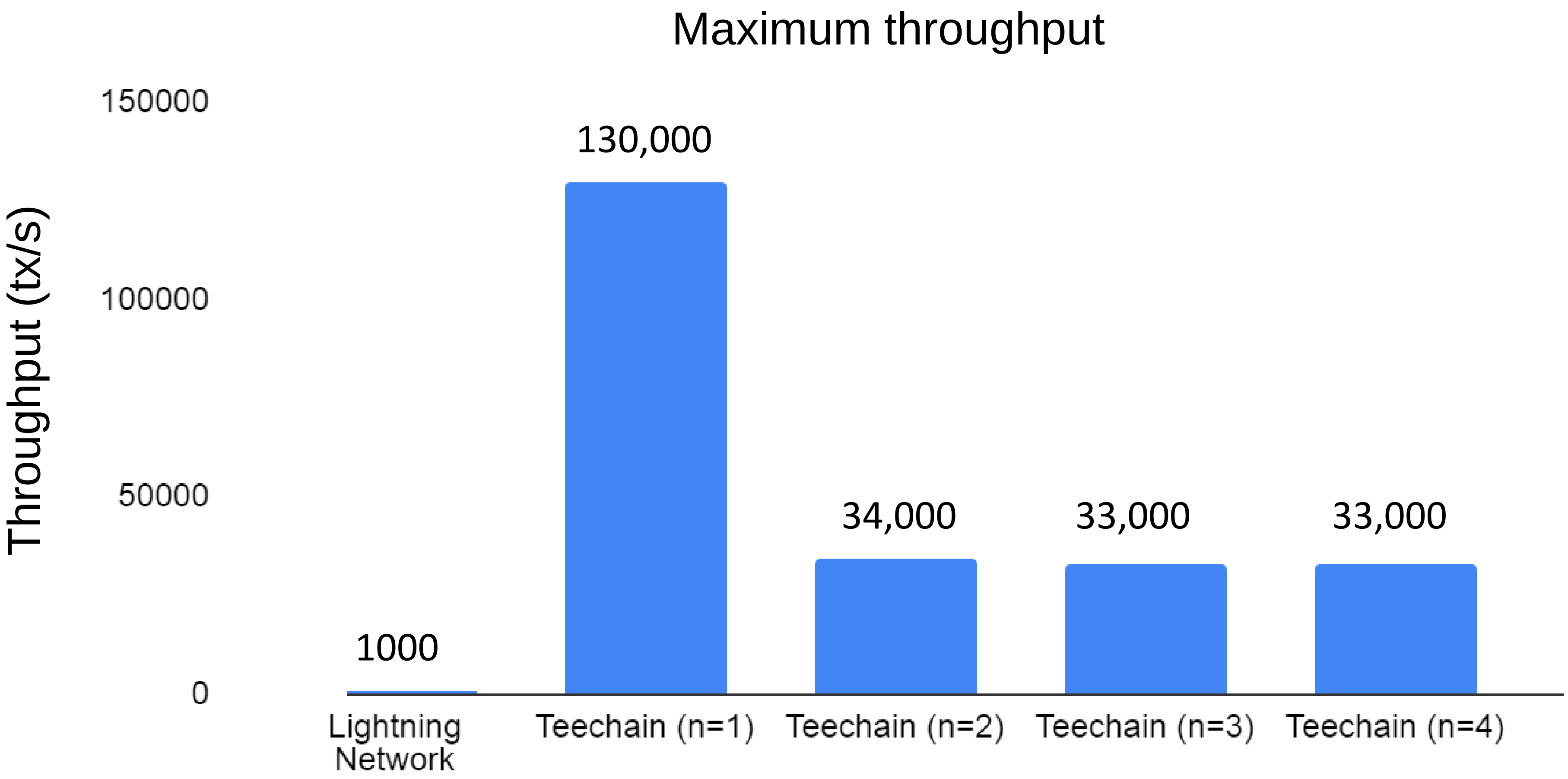
- Maximum **throughput** (tx/second) and **latency** (ack)
- Vary committee sizes (**n** members: London, New York, Haifa)



# How well do Payment Channels perform?

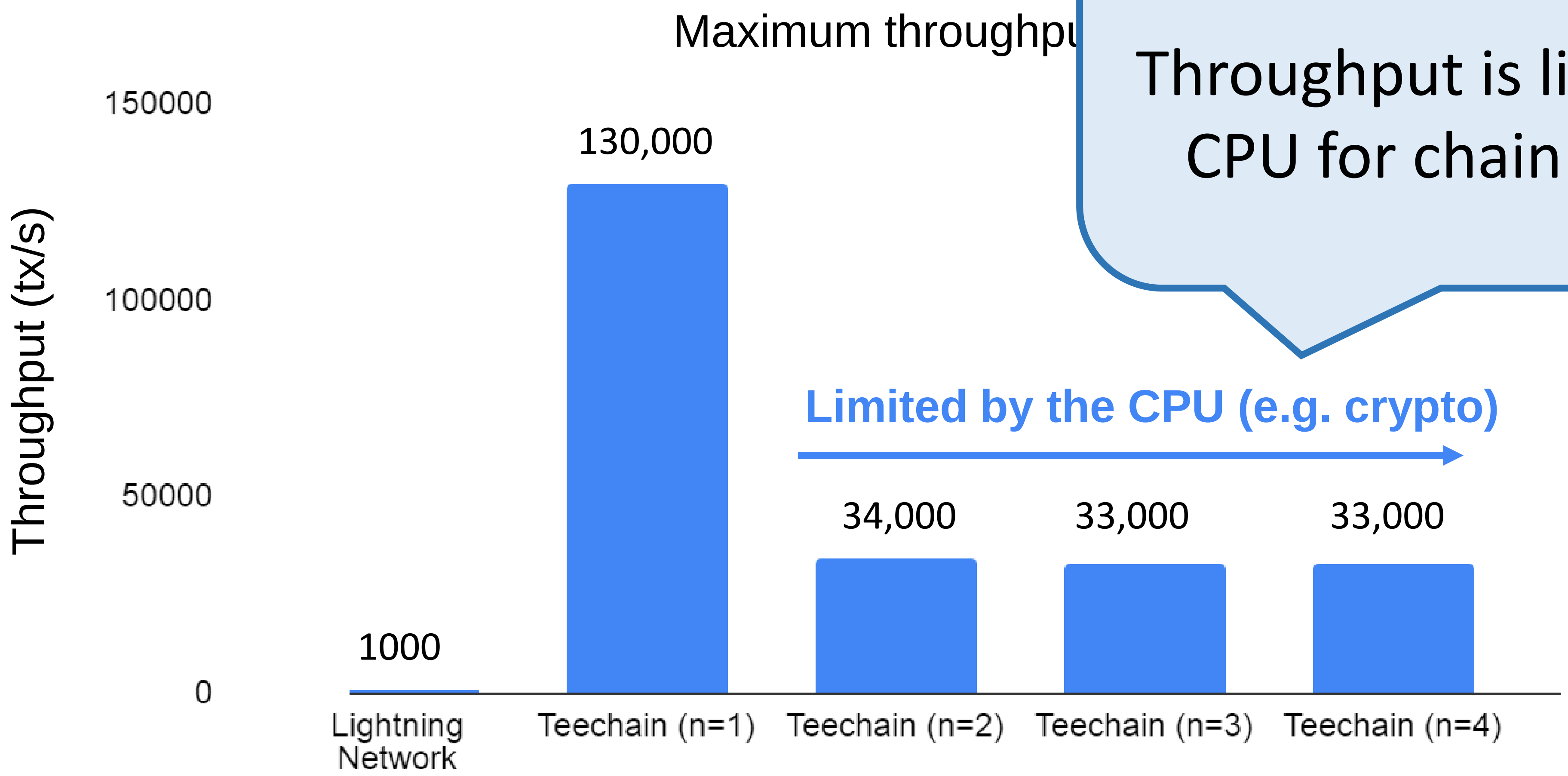
Payment channel: London -- New York

- Maximum **throughput** (tx/second) and **latency** (ack)
- Vary committee sizes (**n** members: London, New York, Haifa)



# How well do Payment Channels perform?

- Payment channel: London -- New York
  - Maximum **throughput** (tx/second) and **latency** (ack)
  - Vary committee sizes (**n** members: London, New York, Ha



**Limited by the CPU (e.g. crypto)**

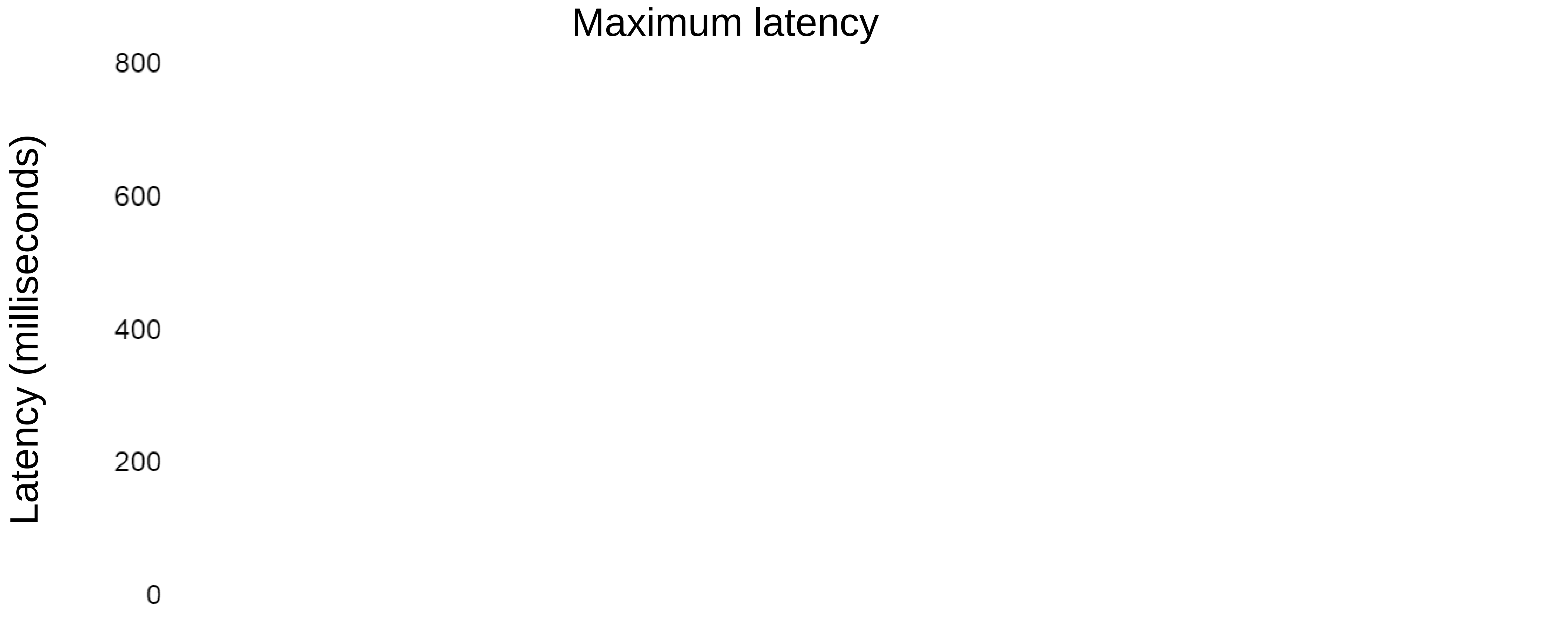
**Throughput is limited!**

Throughput is limited by the CPU for chain replication

# How well do Payment Channels perform?

Payment channel: London -- New York

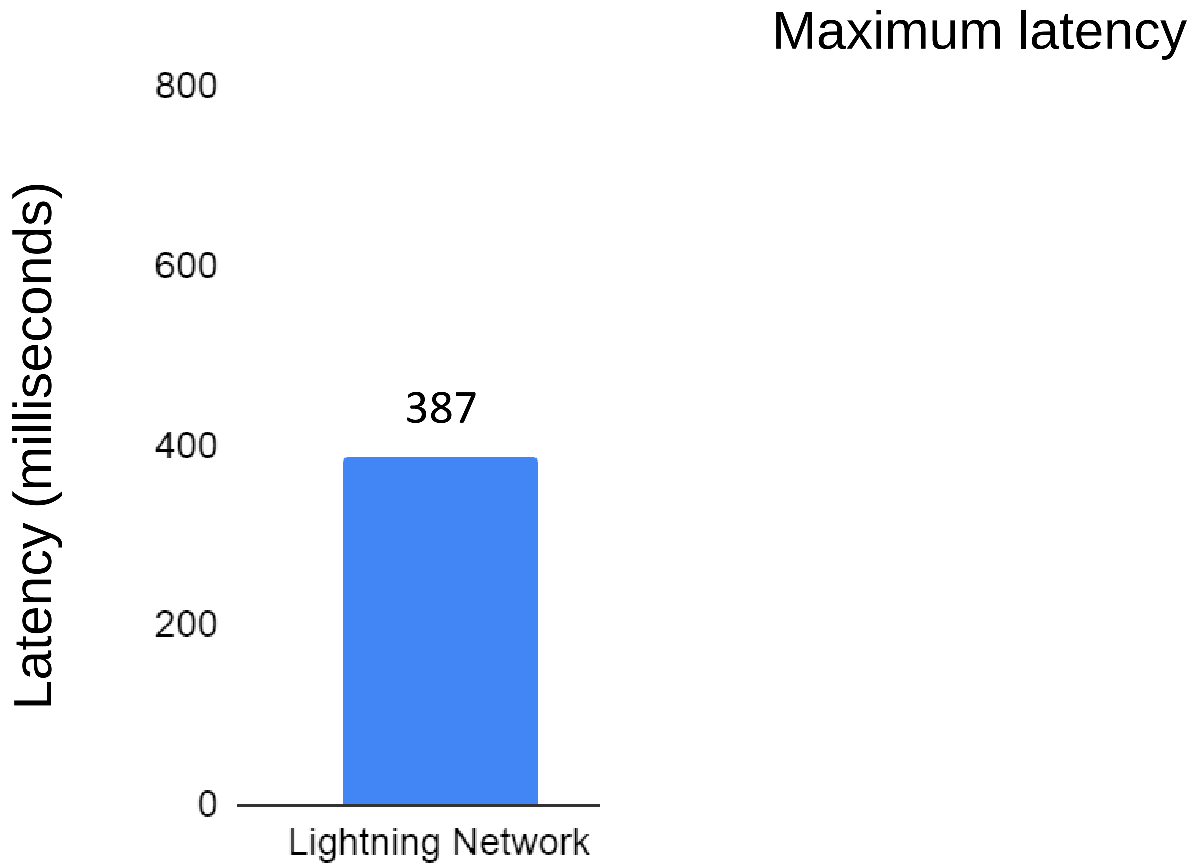
- Maximum **throughput** (tx/second) and **latency** (ack)
- Vary committee sizes (**n** members: London, New York, Haifa)



# How well do Payment Channels perform?

Payment channel: London -- New York

- Maximum **throughput** (tx/second) and **latency** (ack)
- Vary committee sizes (**n** members: London, New York, Haifa)



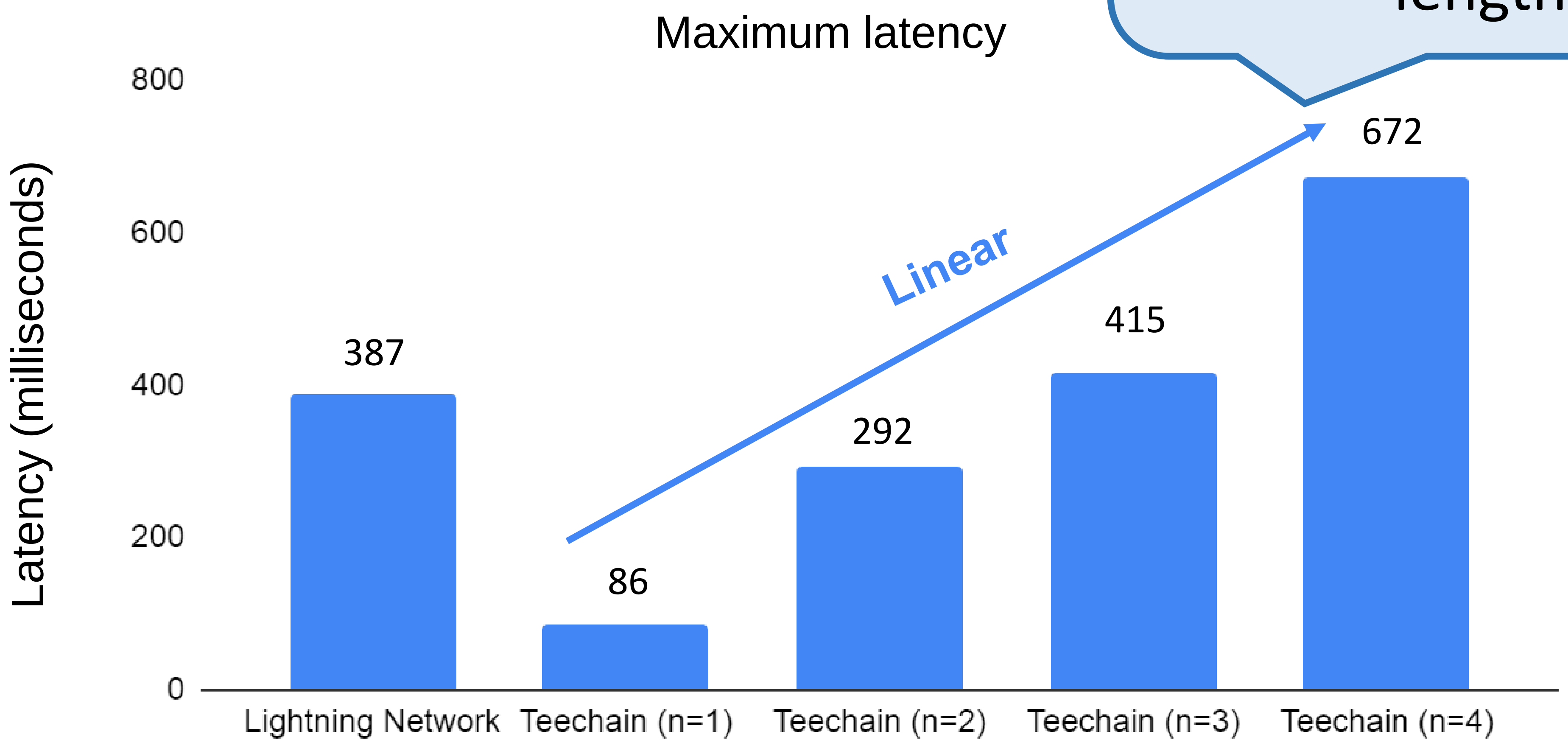
# How well do Payment Channels perform?

Payment channel: London -- New York

- Maximum **throughput** (tx/second) and **latency** (ack)
- Vary committee sizes (**n** members: London, New York, Haifa)

**Latency grows**

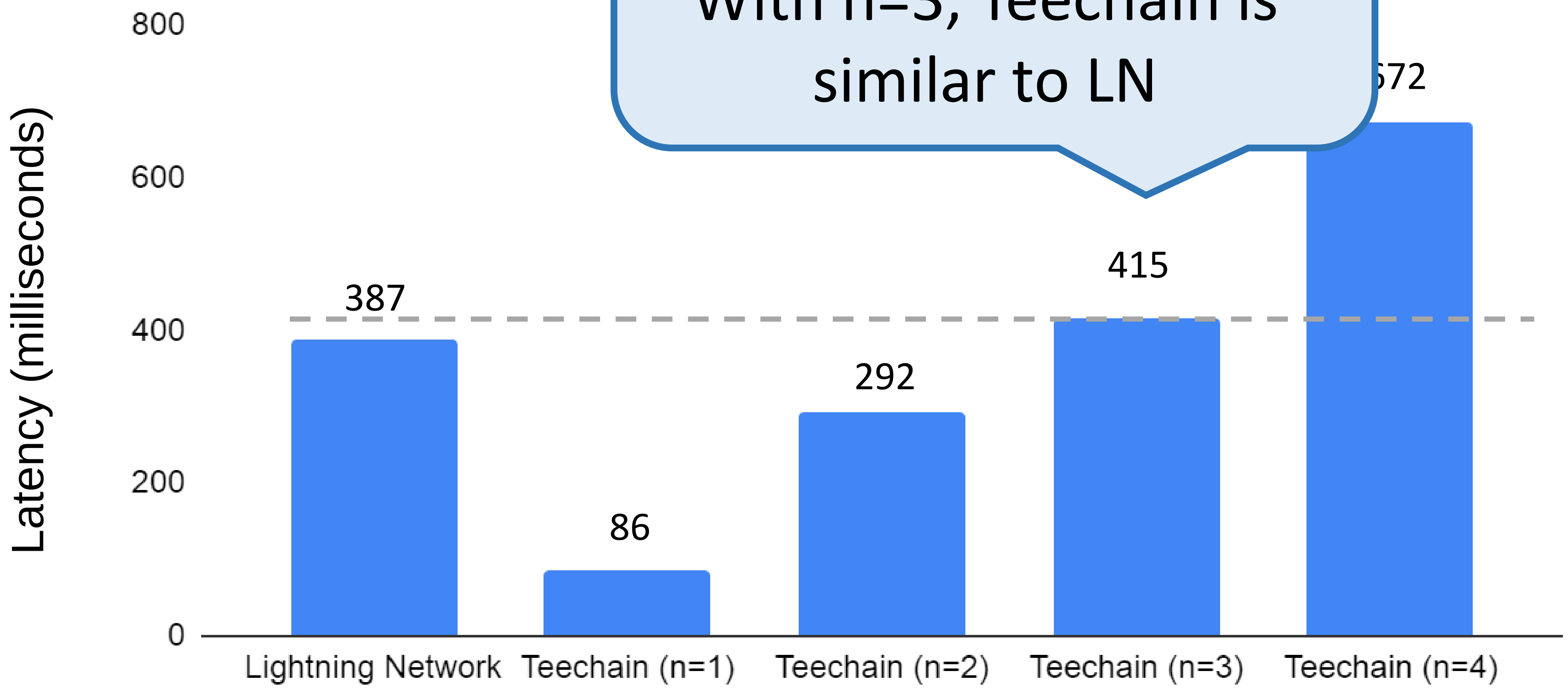
Proportional to chain length



# How well do Payment Channels perform?

Payment channel: London -- New York

- Maximum **throughput** (tx/second) an
- Vary committee sizes (**n** members: L



**Similar latency**

With n=3, Teechain is similar to LN

# Does Teechain scale out?

Payment network deployment:

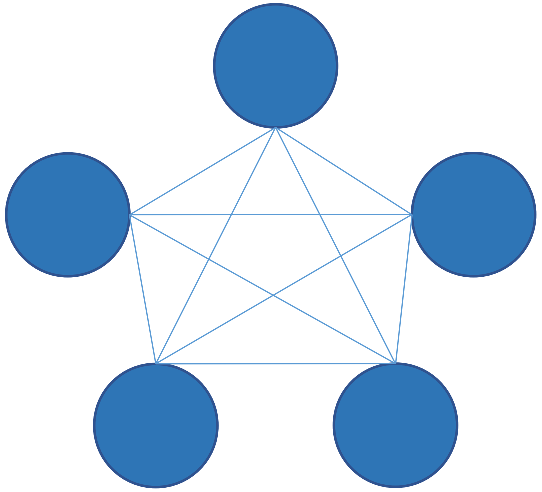
– **Workload:** Bitcoin transaction history across graph

– **Overlay topologies:** Complete vs. hub-and-spoke

# Does Teechain scale out?

Payment network deployment:

– **Workload:** Bitcoin transaction history across graph



**Complete graph**

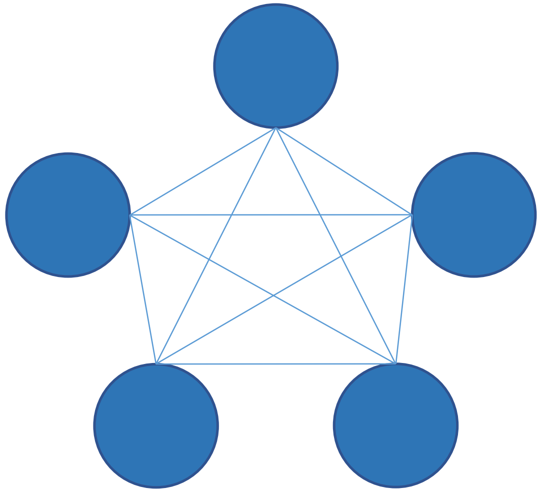
e.g. n=5, 10 channels

(no multi-hop payments)

# Does Teechain scale out?

Payment network deployment:

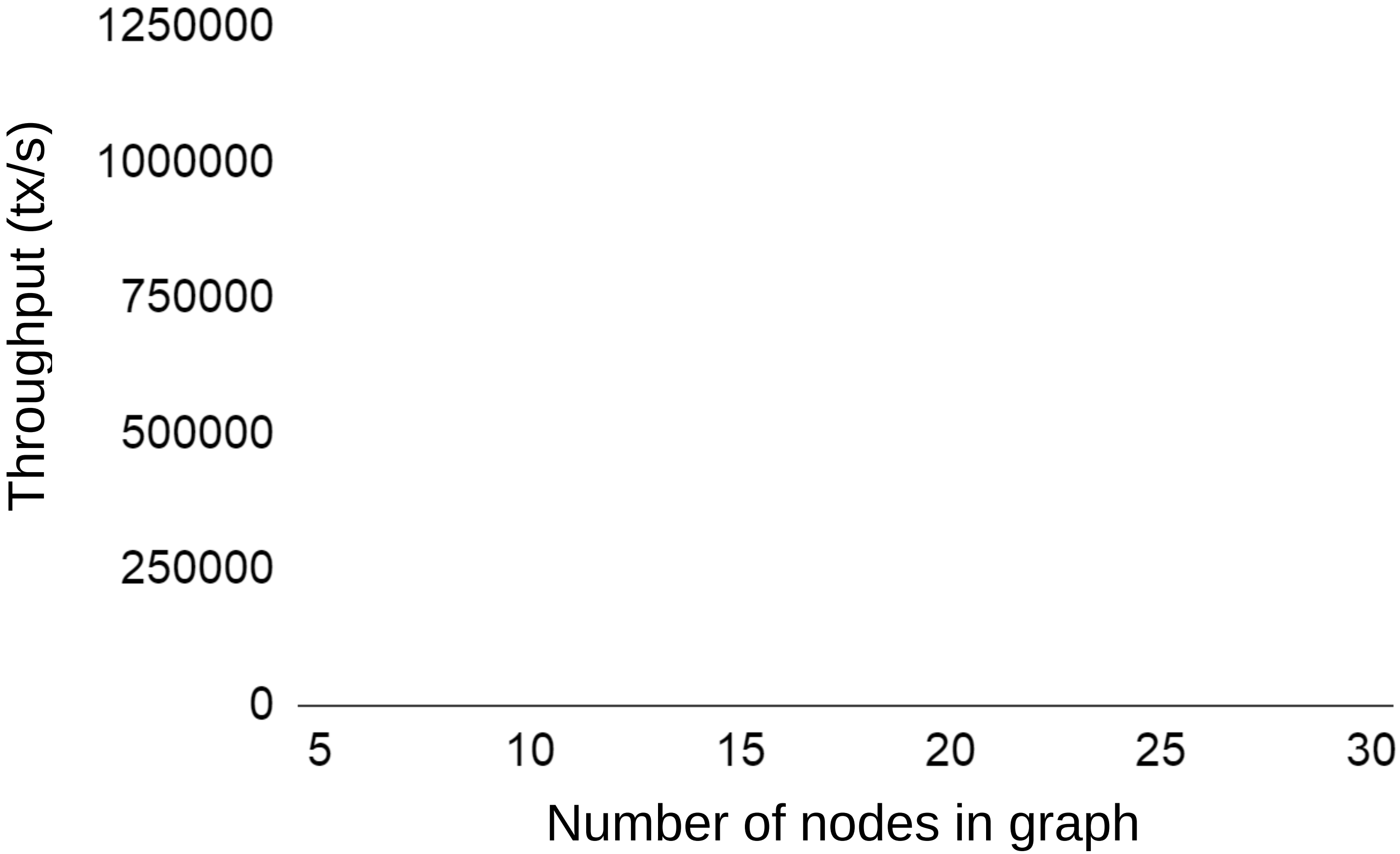
– **Workload:** Bitcoin transaction history across graph



**Complete graph**

e.g. n=5, 10 channels

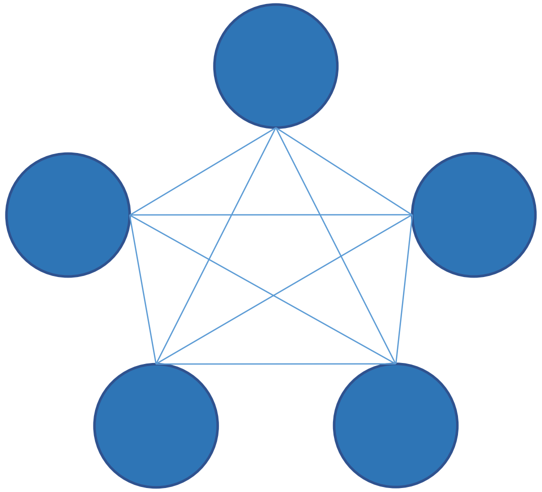
(no multi-hop payments)



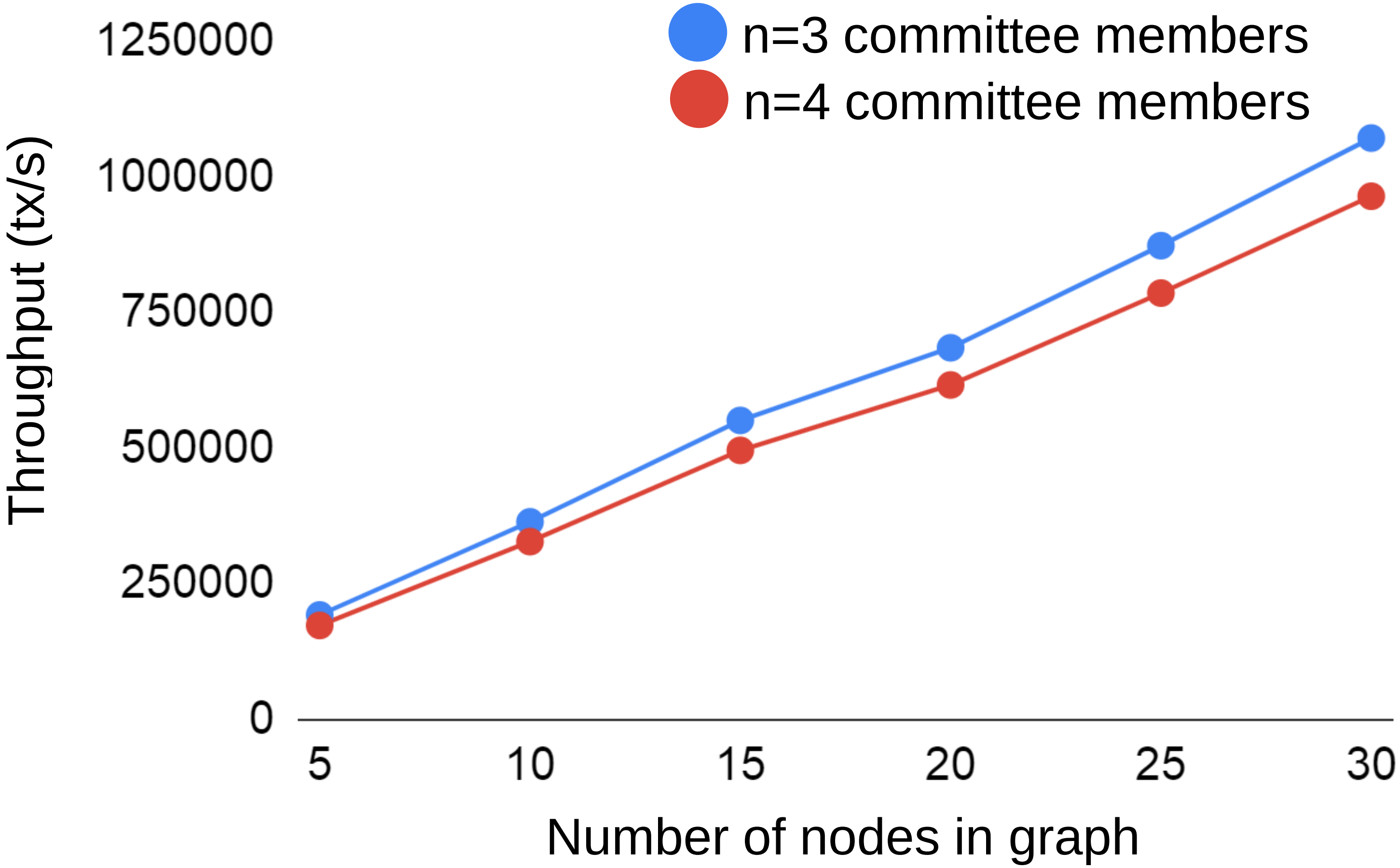
# Does Teechain scale out?

Payment network deployment:

– **Workload:** Bitcoin transaction history across graph



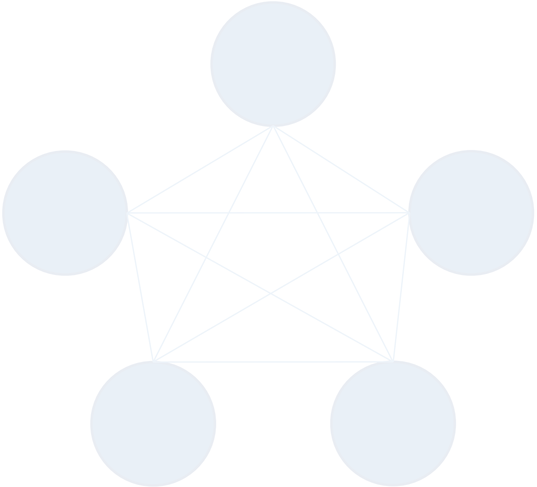
**Complete graph**
e.g. n=5, 10 channels
(no multi-hop payments)



# Does Teechain sc

Payment network deployment:

- **Workload:** Bitcoin transaction history across graph

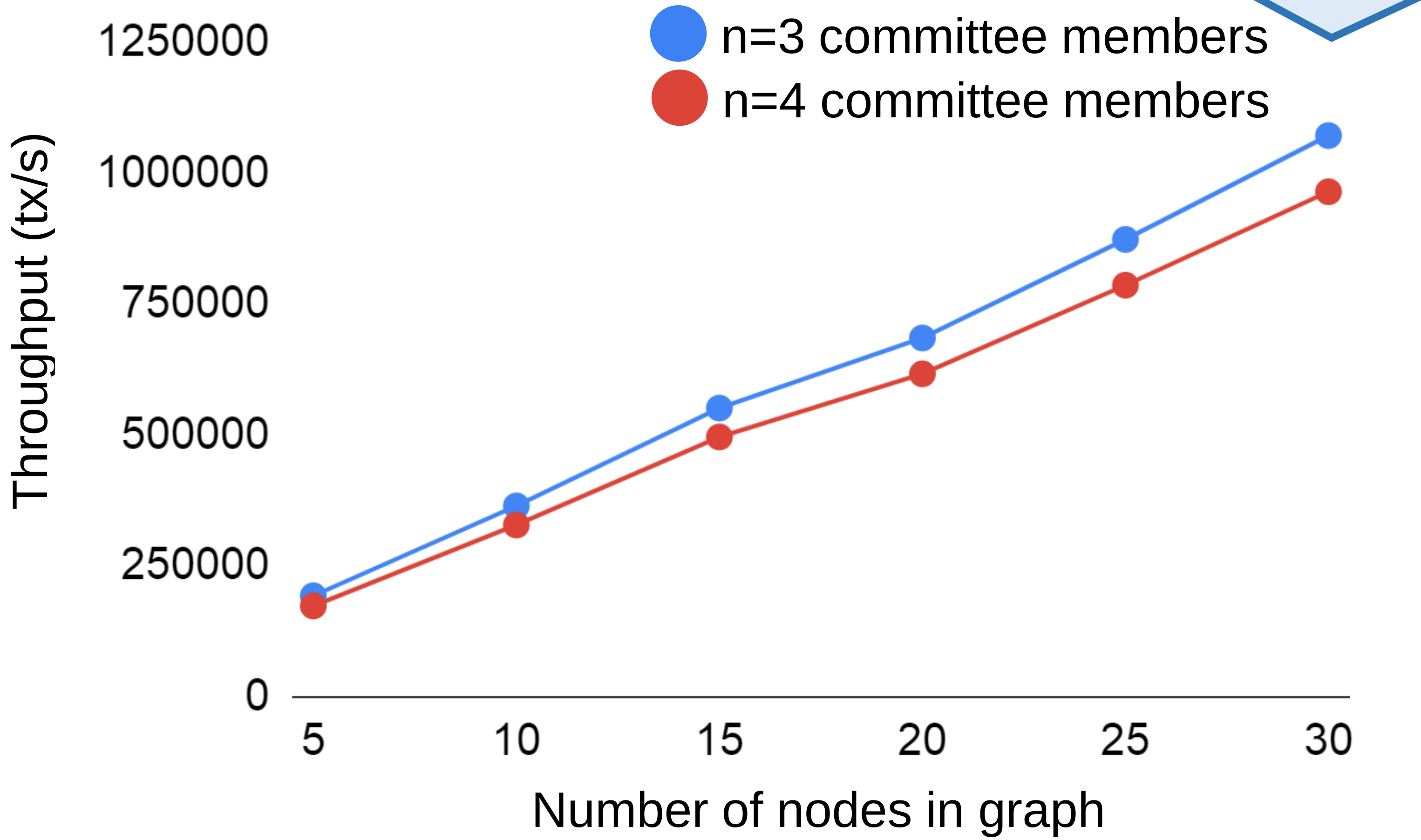


**Complete graph**

e.g. n=5, 10 channels
(no multi-hop payments)

## Committee chains

Throughput is limited by cost to replicate



# Does Teechain scale out?

Payment network

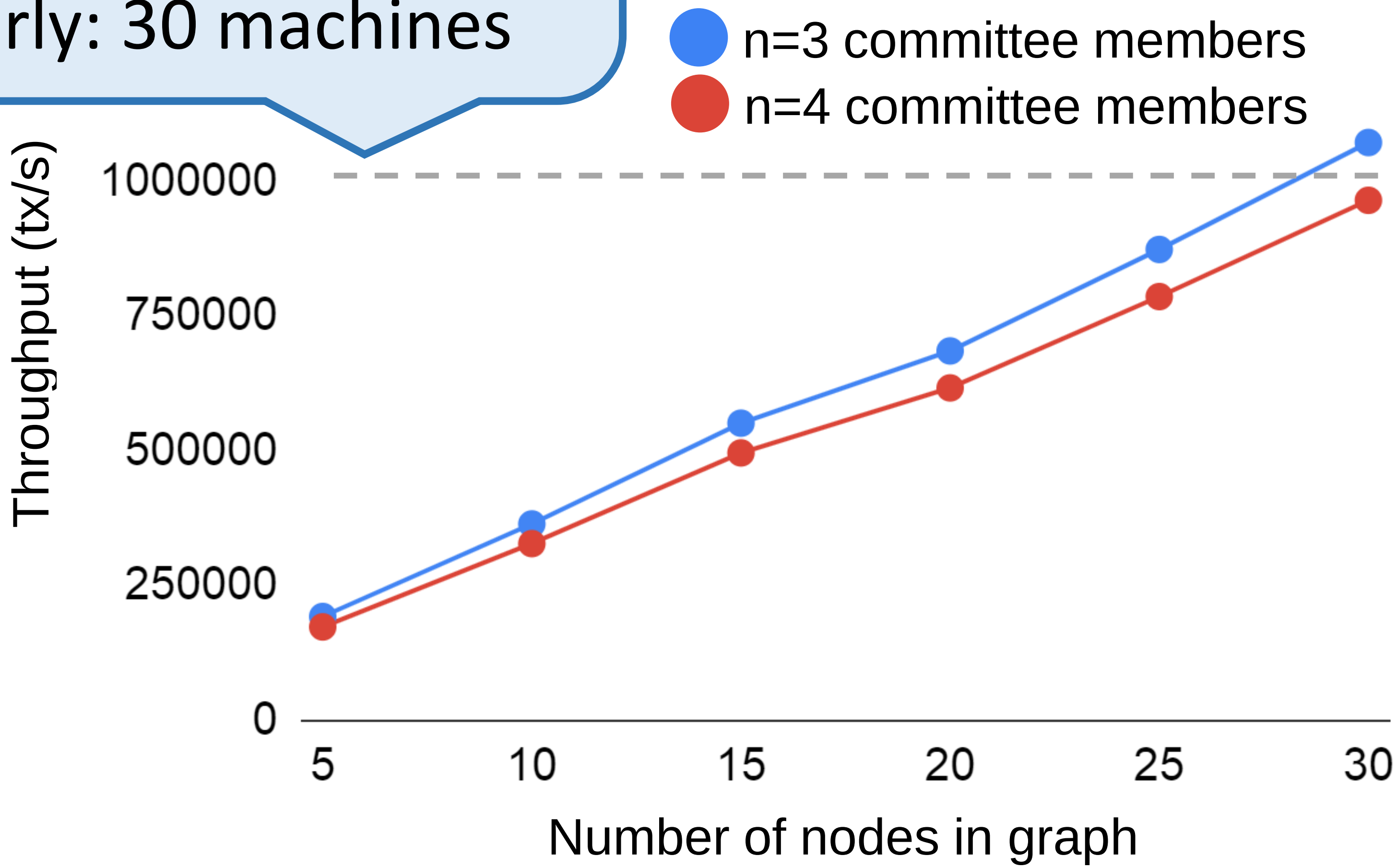
– **Workload:** Bitc

**Complete graph**

e.g. n=5, 10 channels
(no multi-hop payments)

**1 million tx/s**

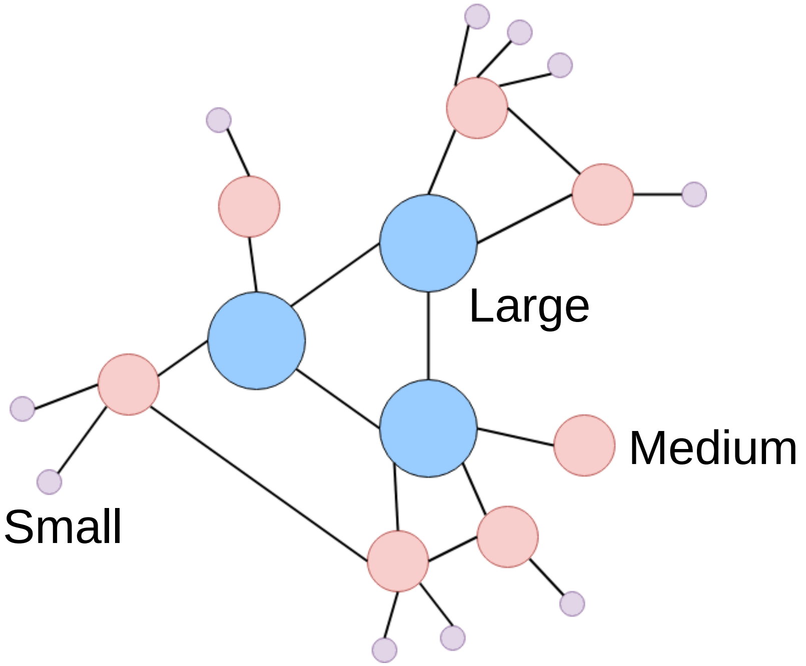
Throughput scales linearly: 30 machines



# Does Teechain scale out?

Payment network deployment:

– **Workload:** Bitcoin transaction history across graph



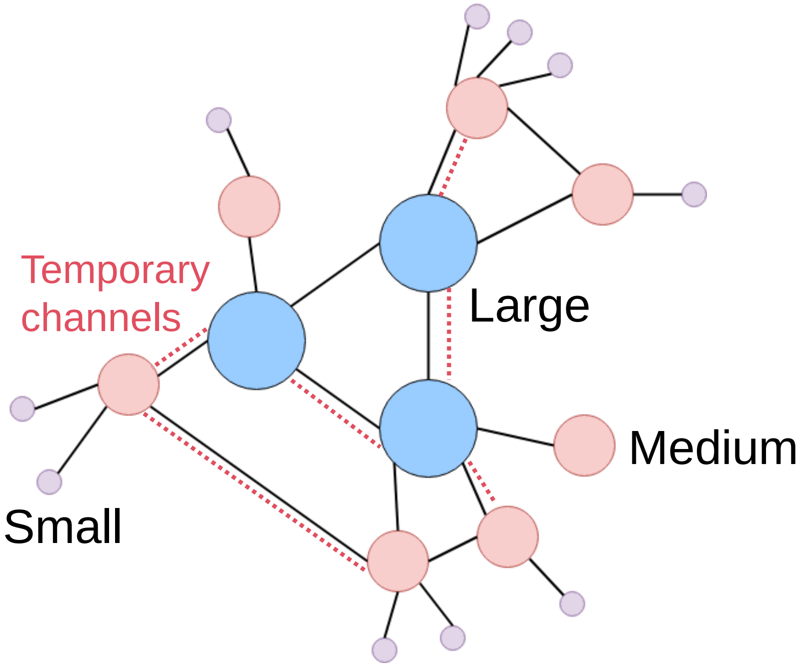
**Hub-and-spoke graph**

Large/medium nodes use temporary channel optimization

# Does Teechain scale out?

Payment network deployment:

– **Workload:** Bitcoin transaction history across graph



**Hub-and-spoke graph**
Large/medium nodes use temporary channel optimization

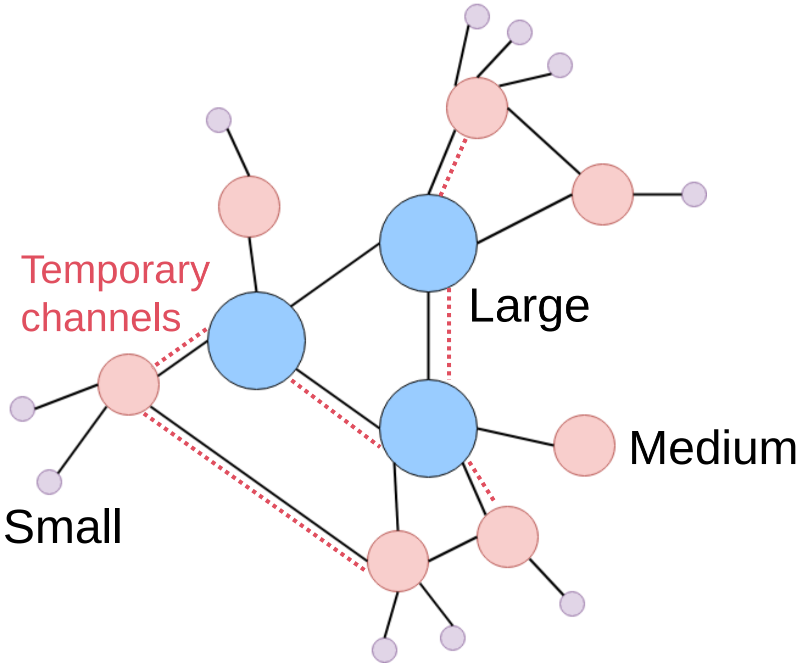
## Optimization: Temporary Channels

Create temporary channels to avoid payment contention
(see the paper!)

# Does Teechain scale out?

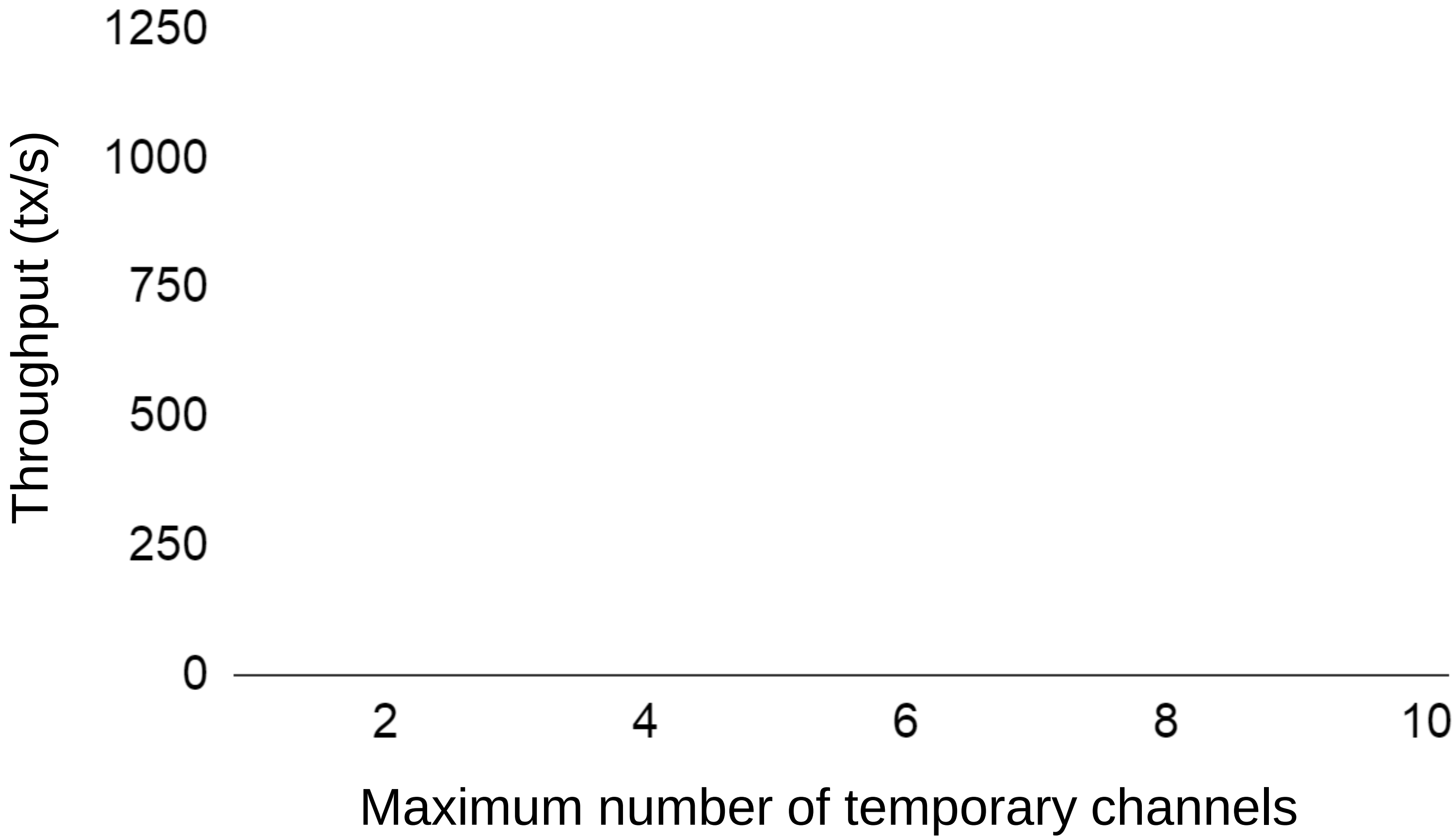
Payment network deployment:

– **Workload:** Bitcoin transaction history across graph



**Hub-and-spoke graph**

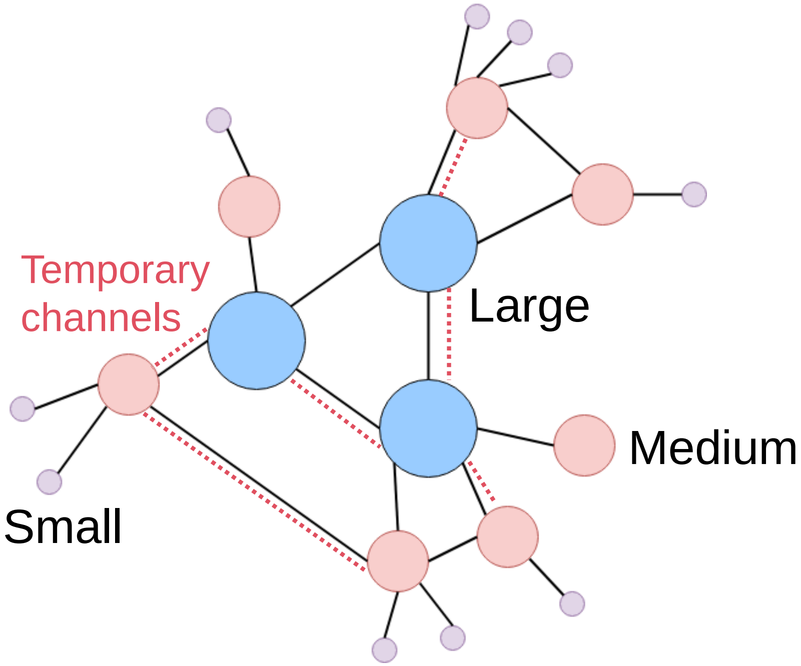
Large/medium nodes use temporary channel optimization



# Does Teechain scale out?

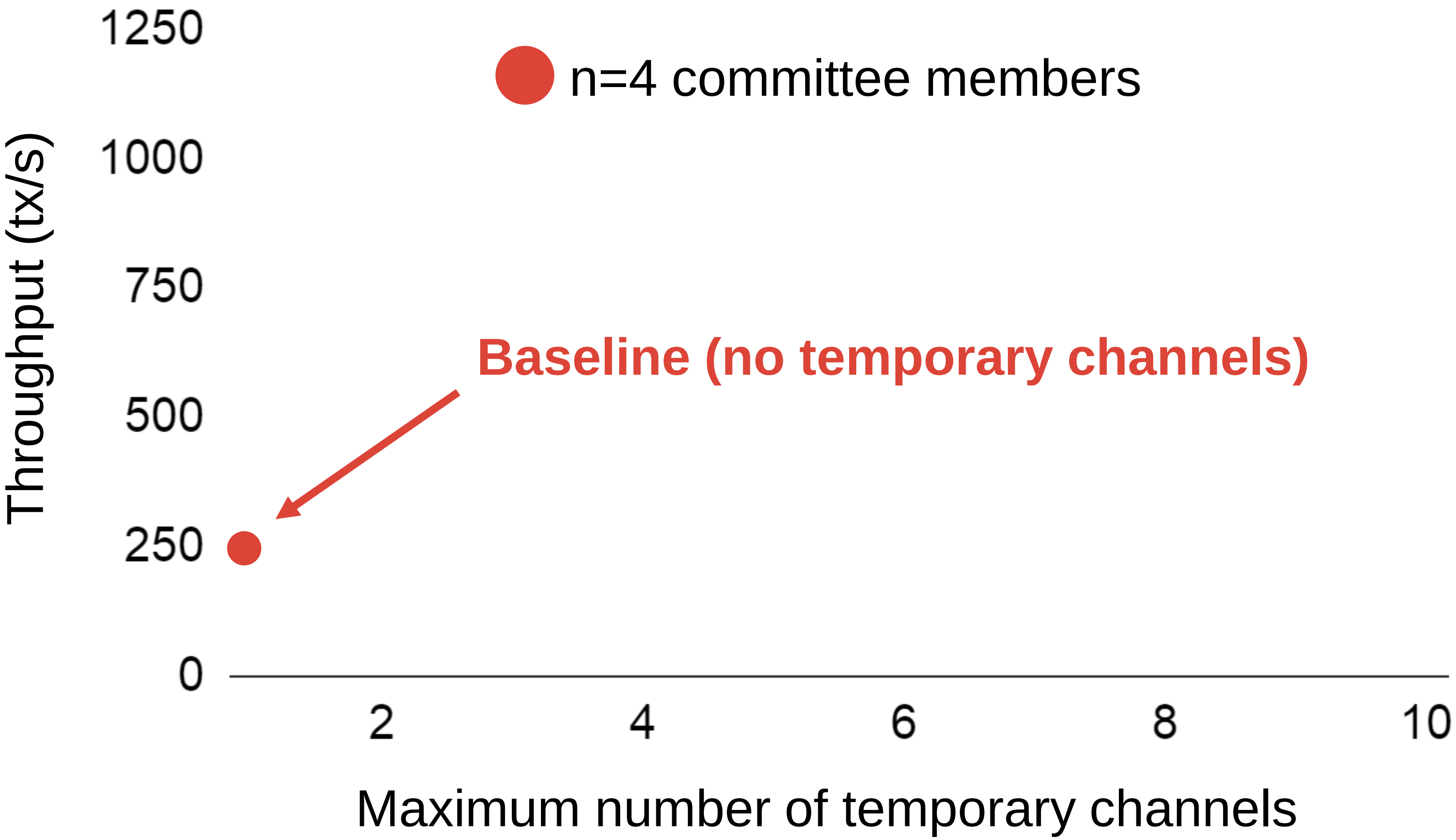
Payment network deployment:

– **Workload:** Bitcoin transaction history across graph



**Hub-and-spoke graph**

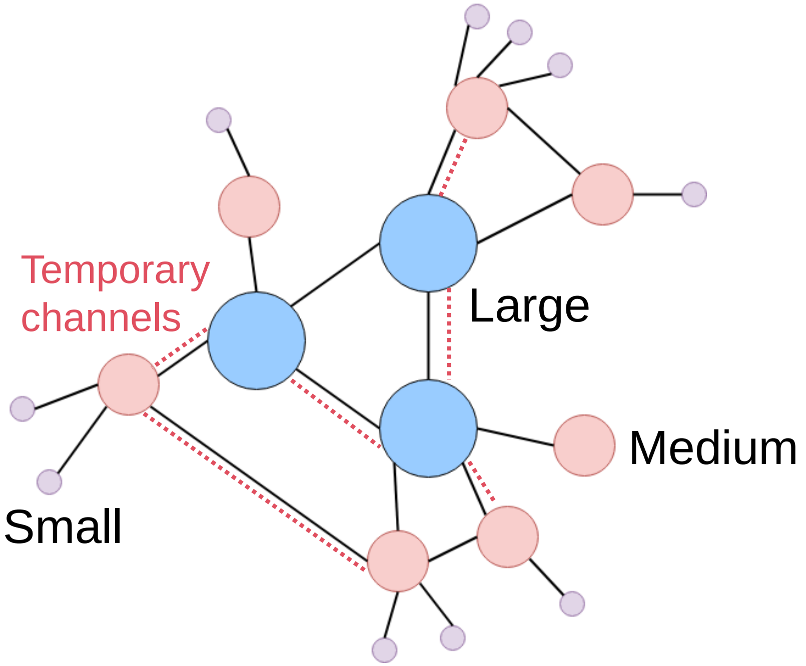
Large/medium nodes use temporary channel optimization



# Does Teechain scale out?

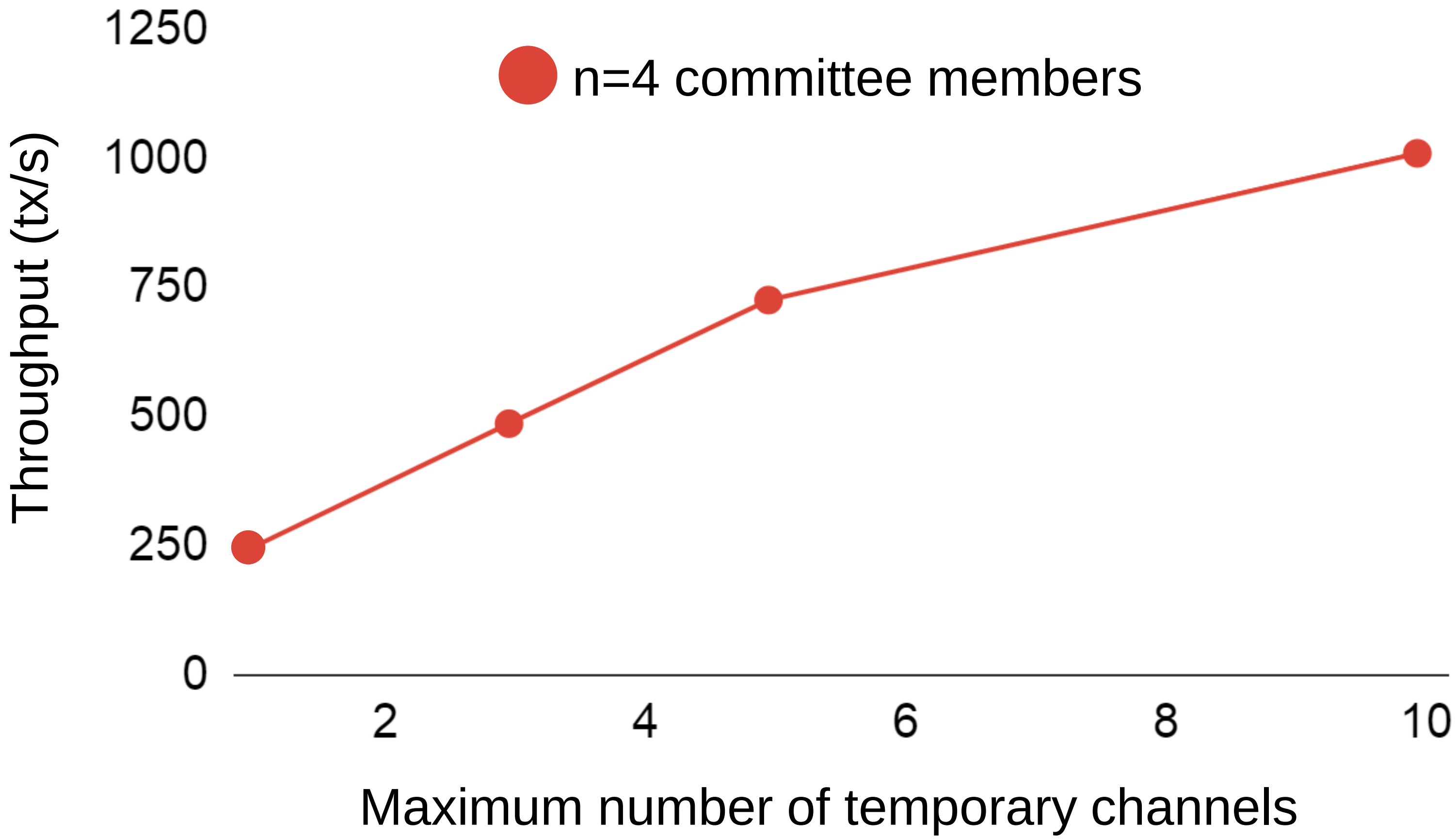
Payment network deployment:

– **Workload:** Bitcoin transaction history across graph



**Hub-and-spoke graph**

Large/medium nodes use temporary channel optimization



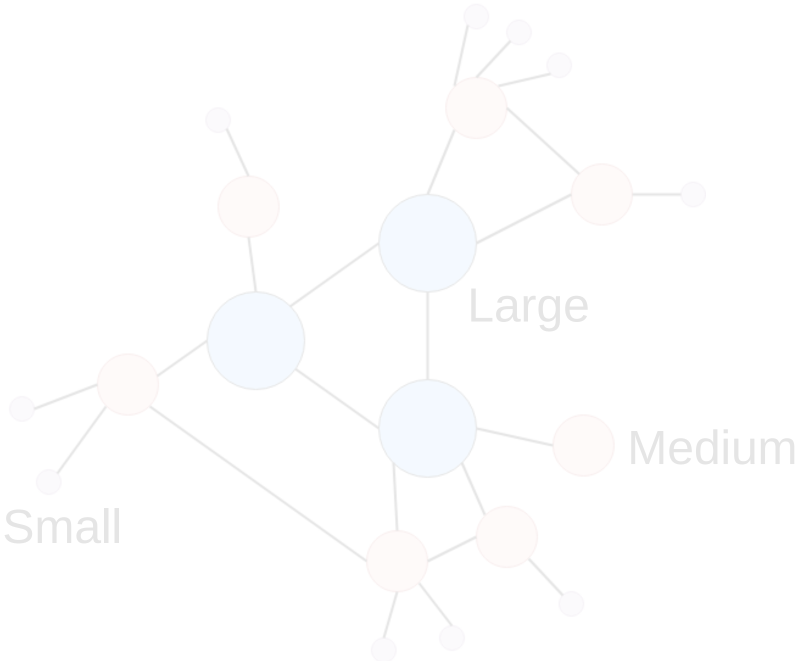
# Does Teechain scale out?

Payment network deployment:

– **Workload:** Bitcoin transaction history across graph

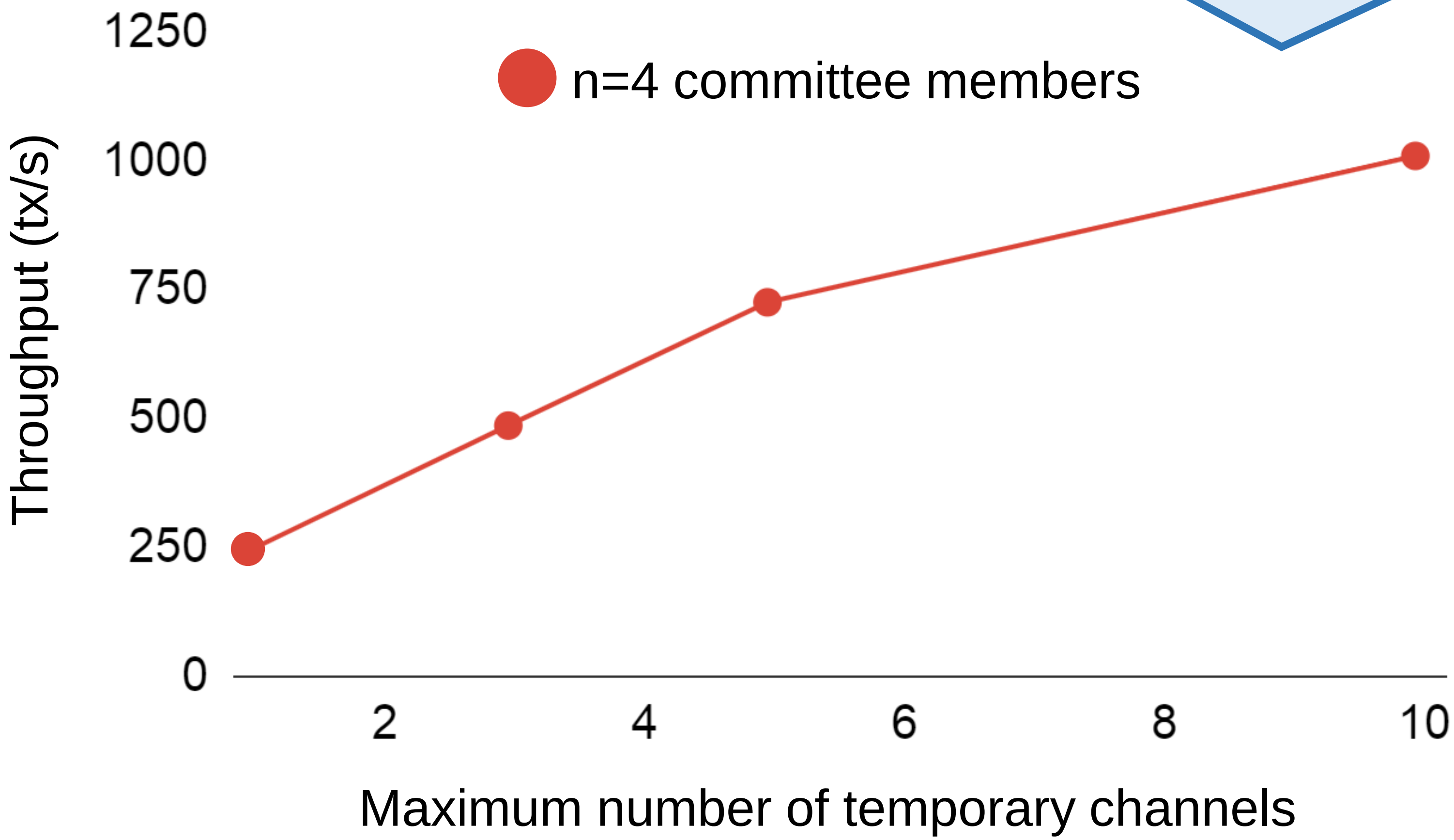
## Optimization

Temporary channels alleviate congestion!



Hub-and-spoke graph

Large/medium nodes use temporary channel optimization



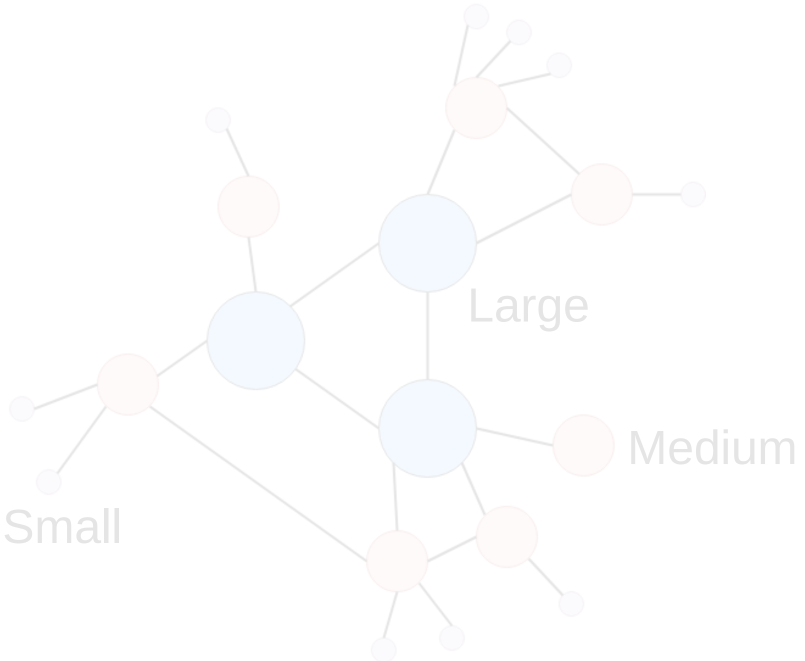
# Does Teechain scal

**Best performance**

Performance requires high connectivity!

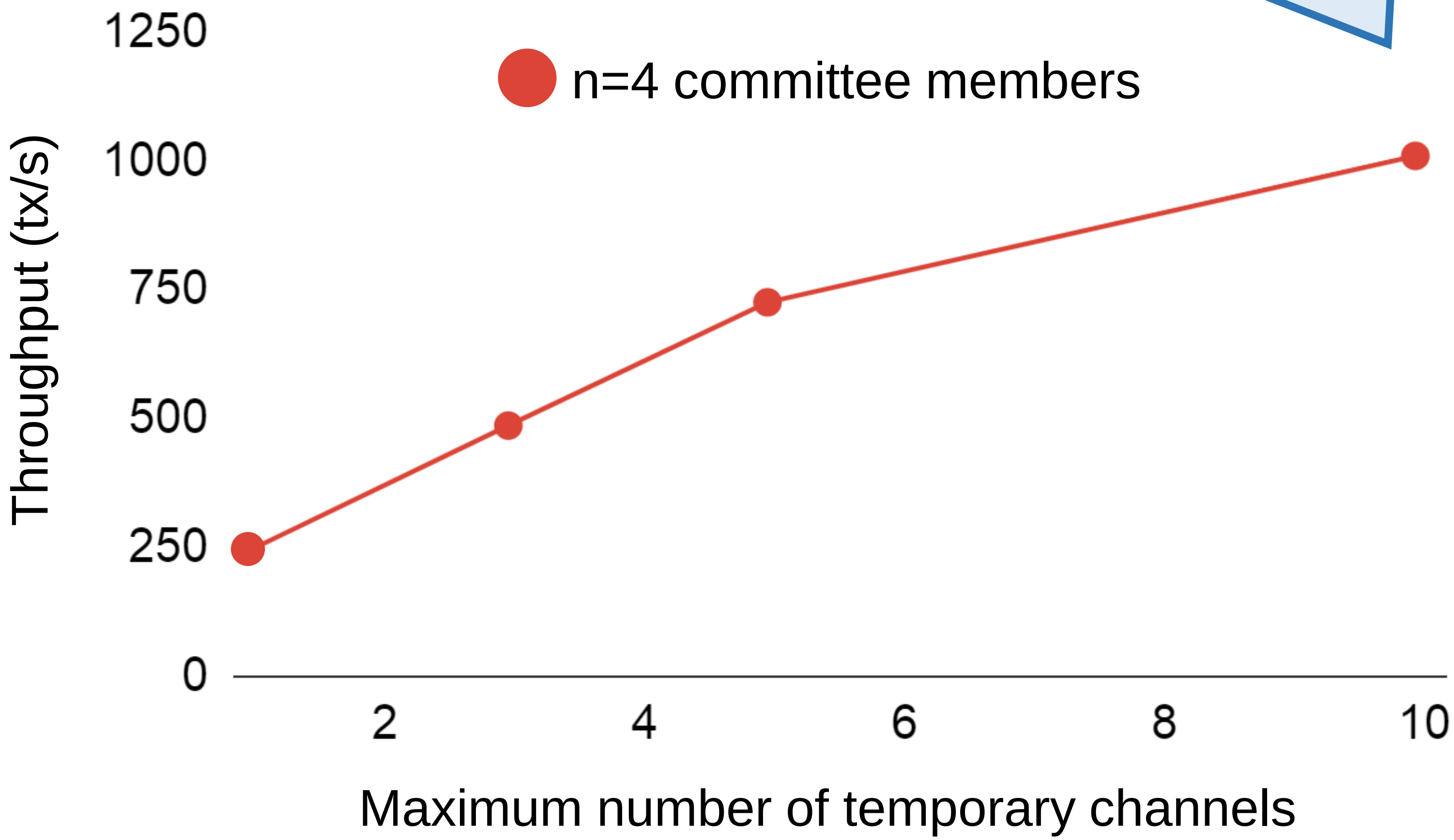
Payment network deployment:

– **Workload:** Bitcoin transaction history across graph

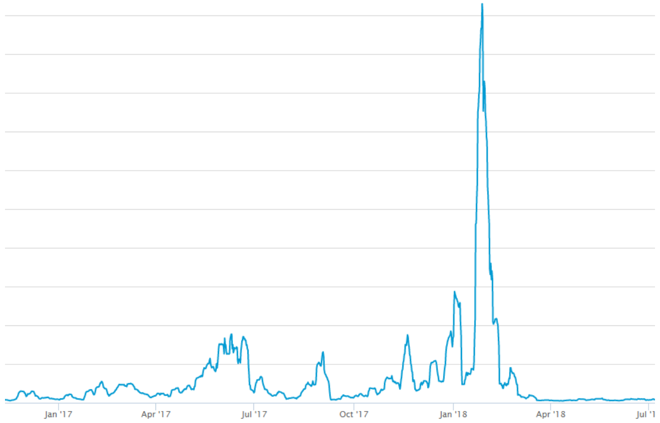


Hub-and-spoke graph

Large/medium nodes use temporary channel optimization



# Summary



## Blockchains are best-effort:

- Security shouldn’t rely on read/write latencies!
- Assume *asynchronous blockchain access*

intel SGX

## TEEs are not silver bullets:

- Must allow for some degree of failures!
- Committees compliment TEEs

## Open-source online:

- https://github.com/lsds/Teechain
- Contact us: **teechain.network**

Artifacts Available acm

Artifacts Evaluated acm Functional

# Thank you!

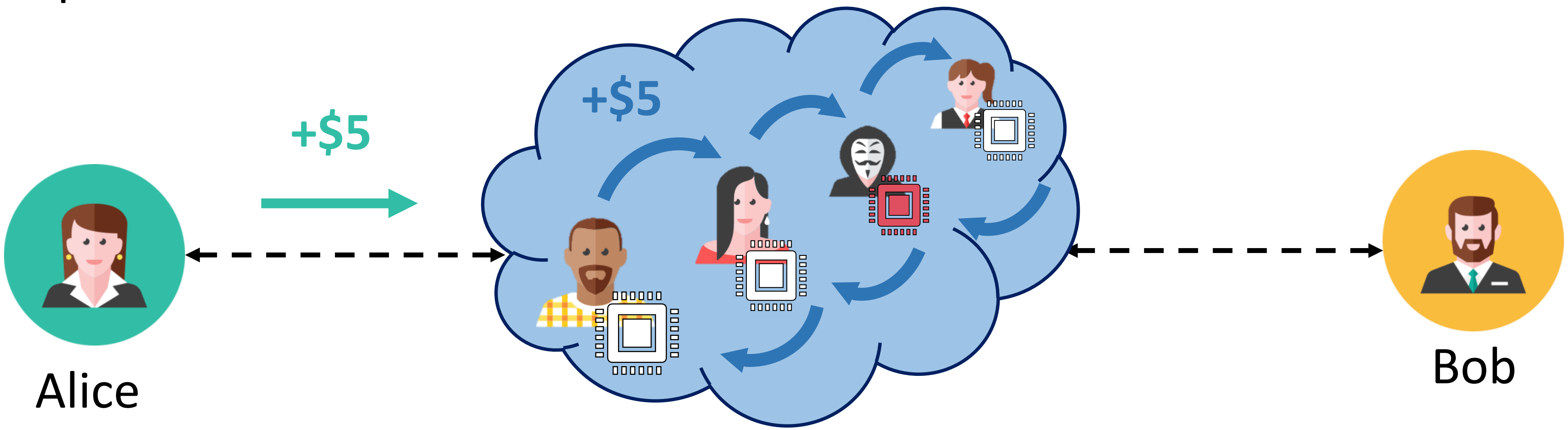
# Additional slides

# Chain replication: An overview

**On each payment channel update:**

- Replicate state of the head in the chain and propagate it down the chain

Update state:



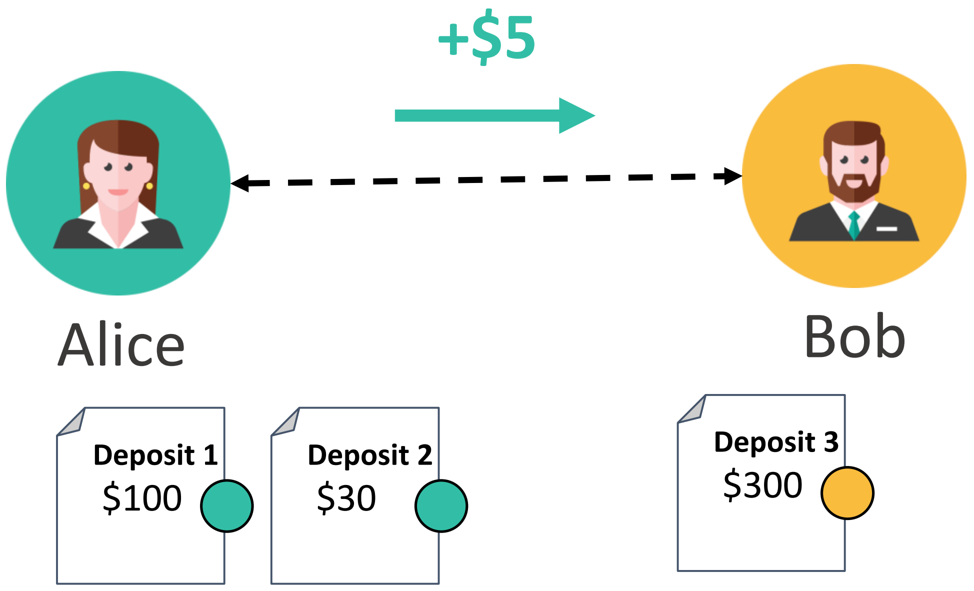
Treasury Committee

# Dynamic deposits: An overview

**Teechain supports dynamic deposits:**

- Deposits can be added/removed from payment channels
- New deposits can be created at runtime

Collateral = Amount deposited – Amount spent



Alice: (100 + 30) – 5 = **125**

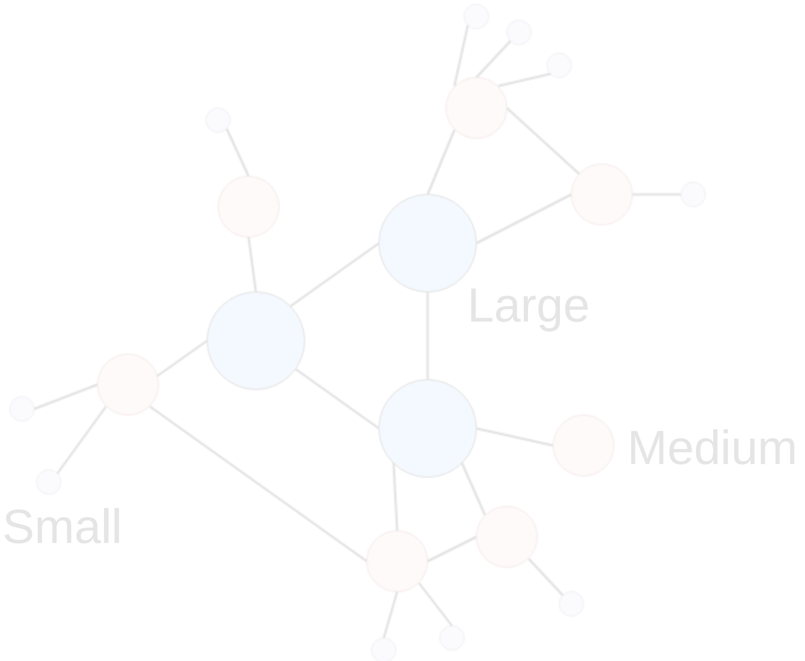
# Does Teechain Sc

## Throughput is Limited

Throughput is limited by replication costs

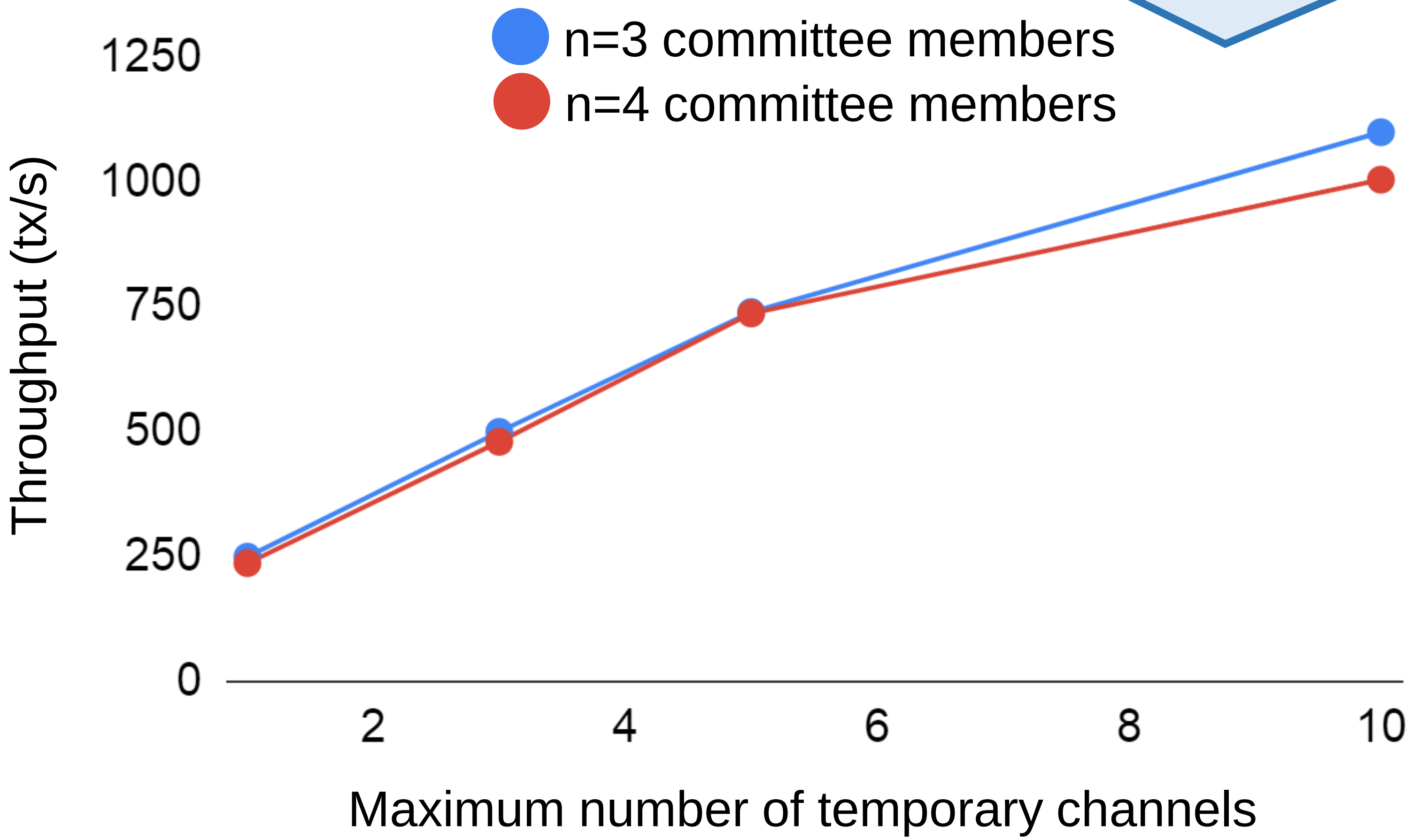
Payment Network deployment:

– **Workload:** Bitcoin transaction history across graph



Hub-and-Spoke graph

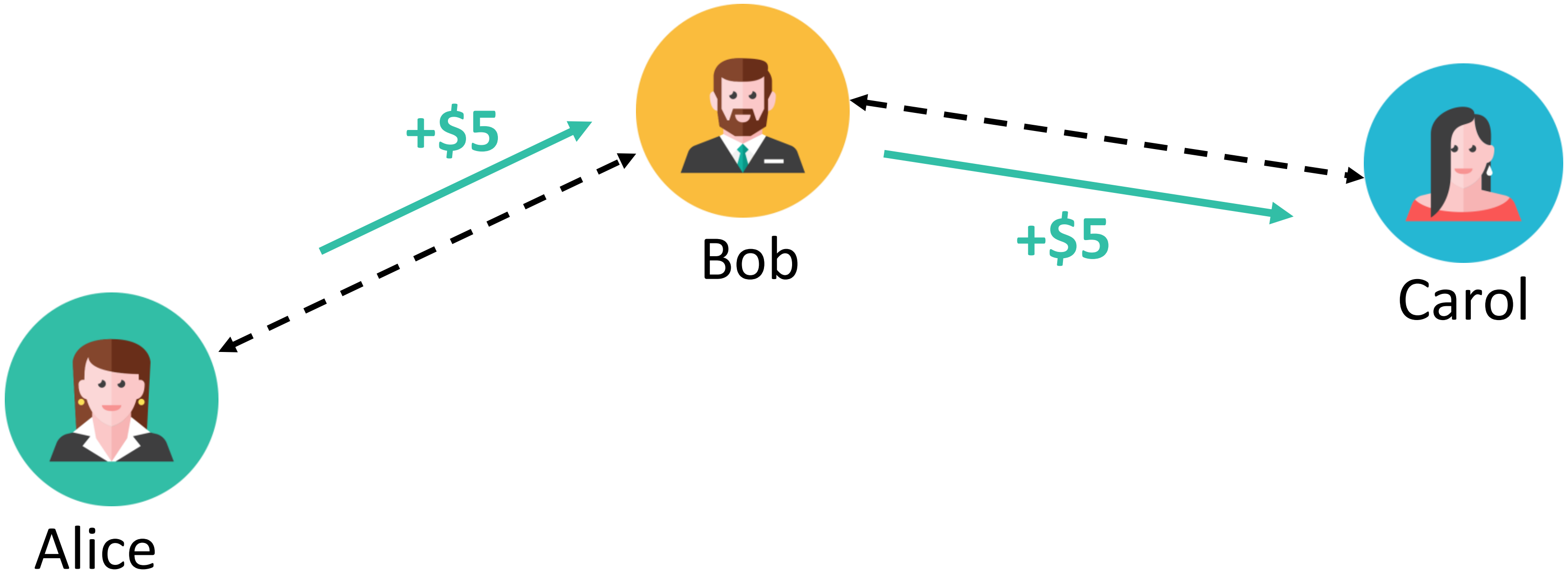
Large/medium nodes use temporary channel optimization



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

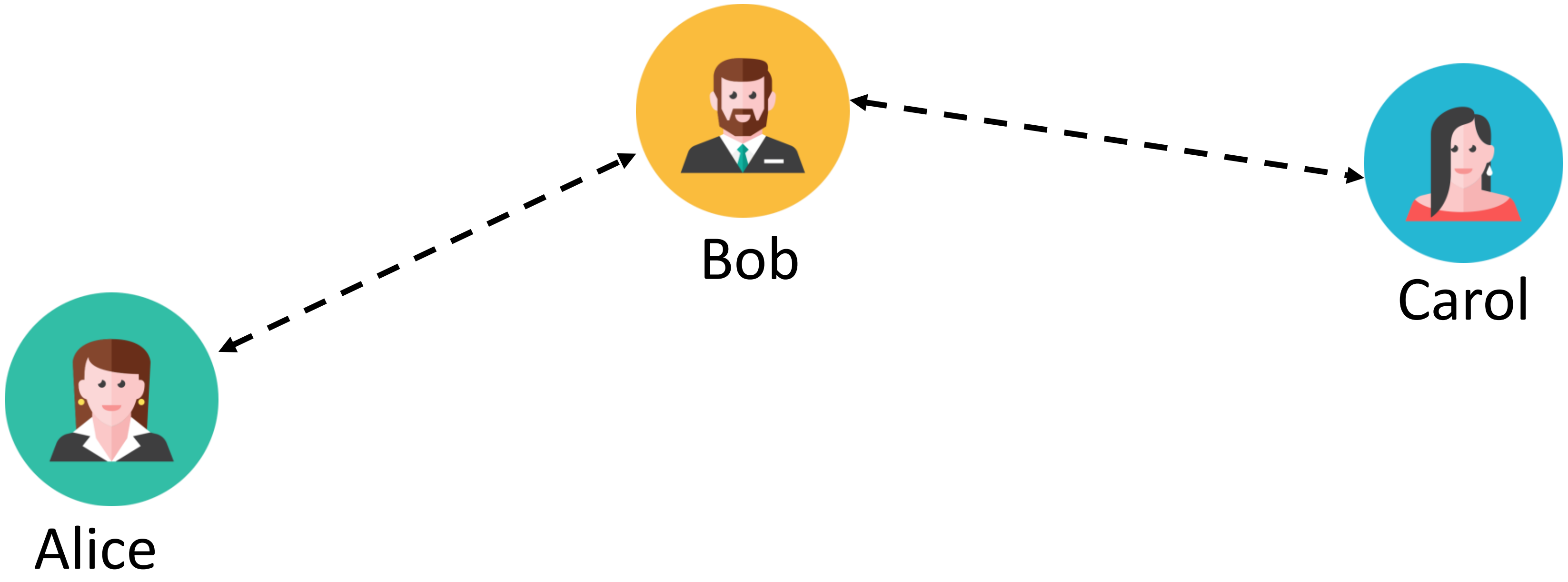
- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

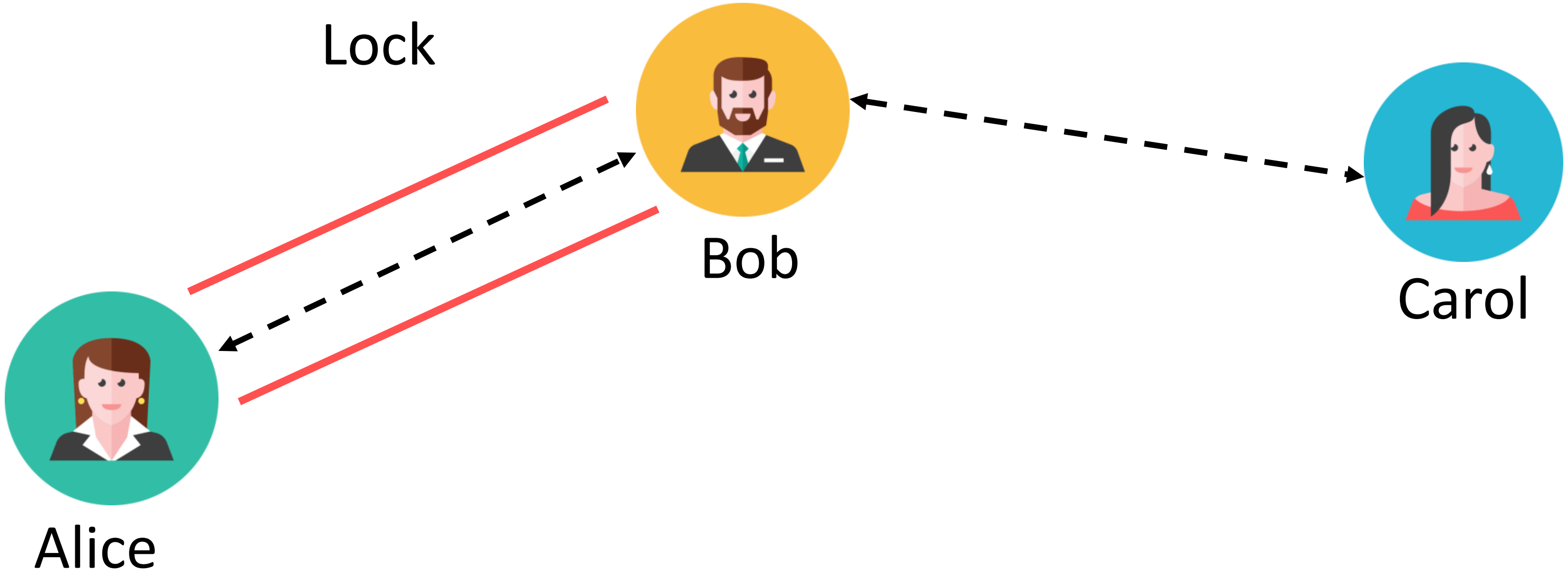
- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path
- **Our solution:** Lock payment path and execute multi-phase commit



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

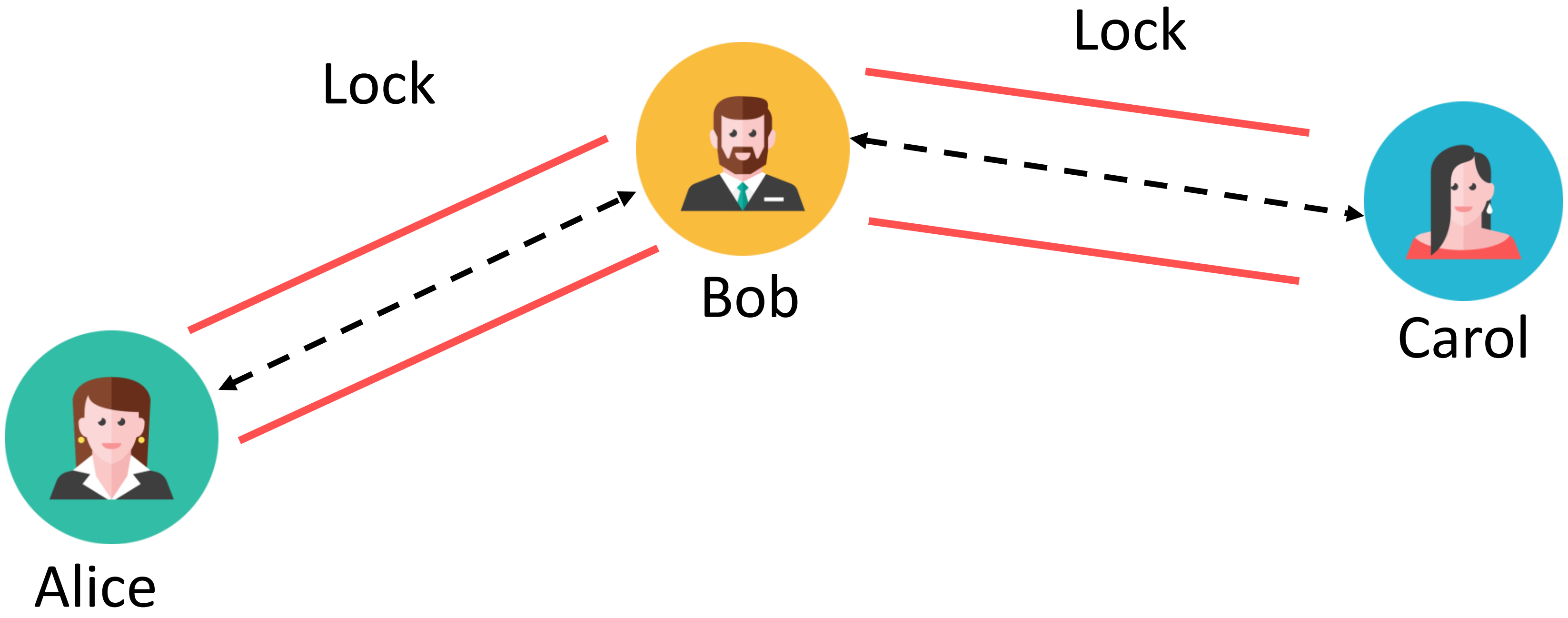
- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path
- **Our solution:** Lock payment path and execute multi-phase commit



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

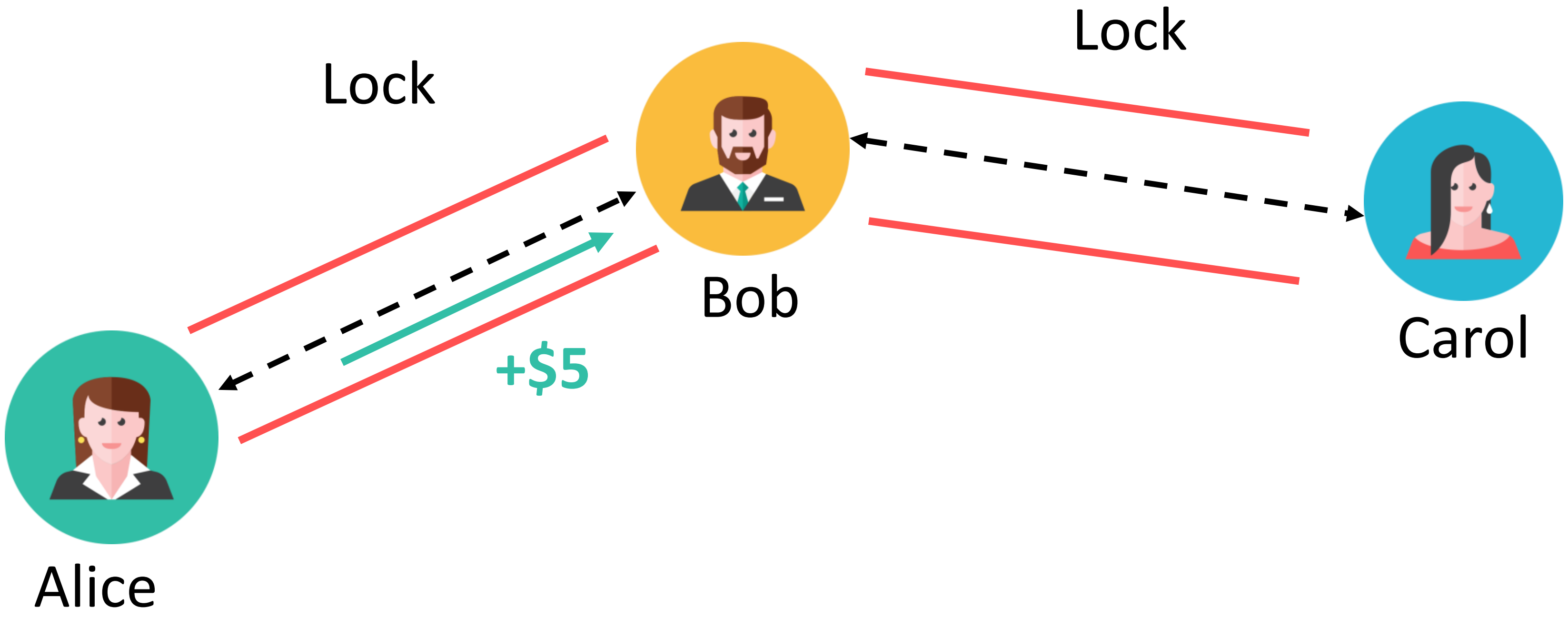
- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path
- **Our solution:** Lock payment path and execute multi-phase commit



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

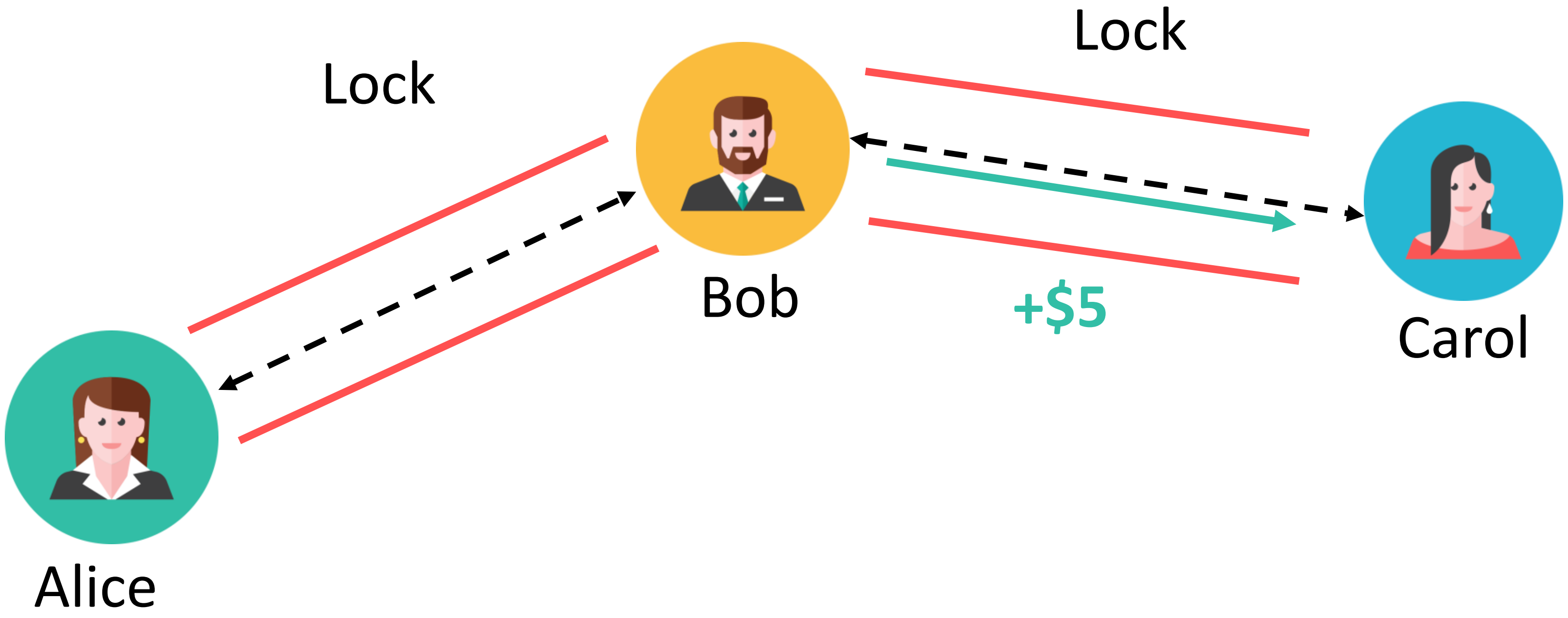
- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path
- **Our solution:** Lock payment path and execute multi-phase commit



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

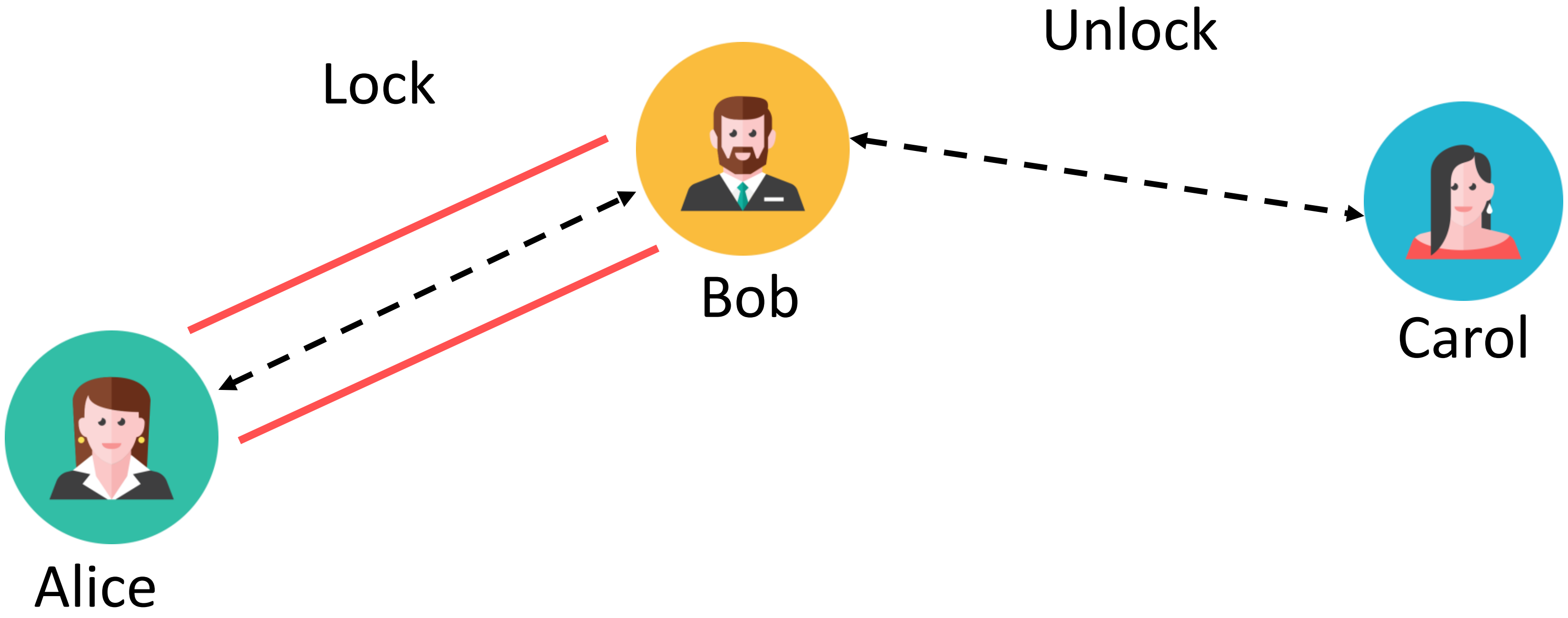
- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path
- **Our solution:** Lock payment path and execute multi-phase commit



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

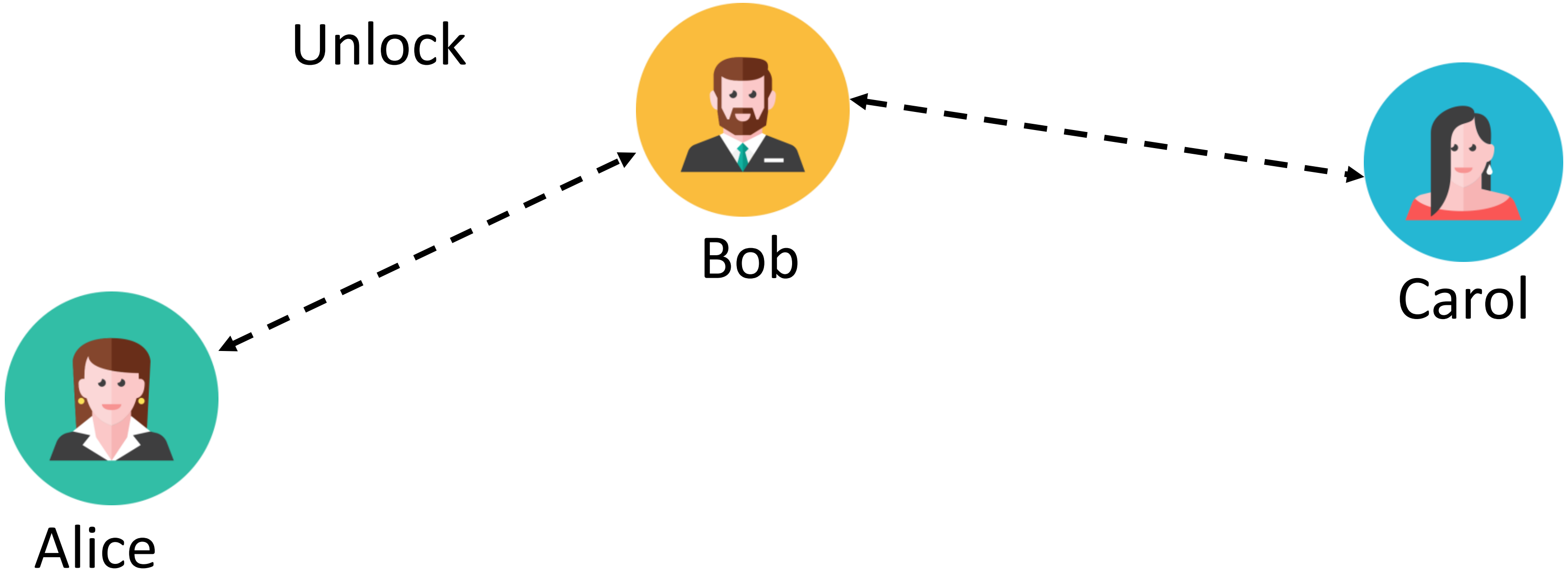
- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path
- **Our solution:** Lock payment path and execute multi-phase commit



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

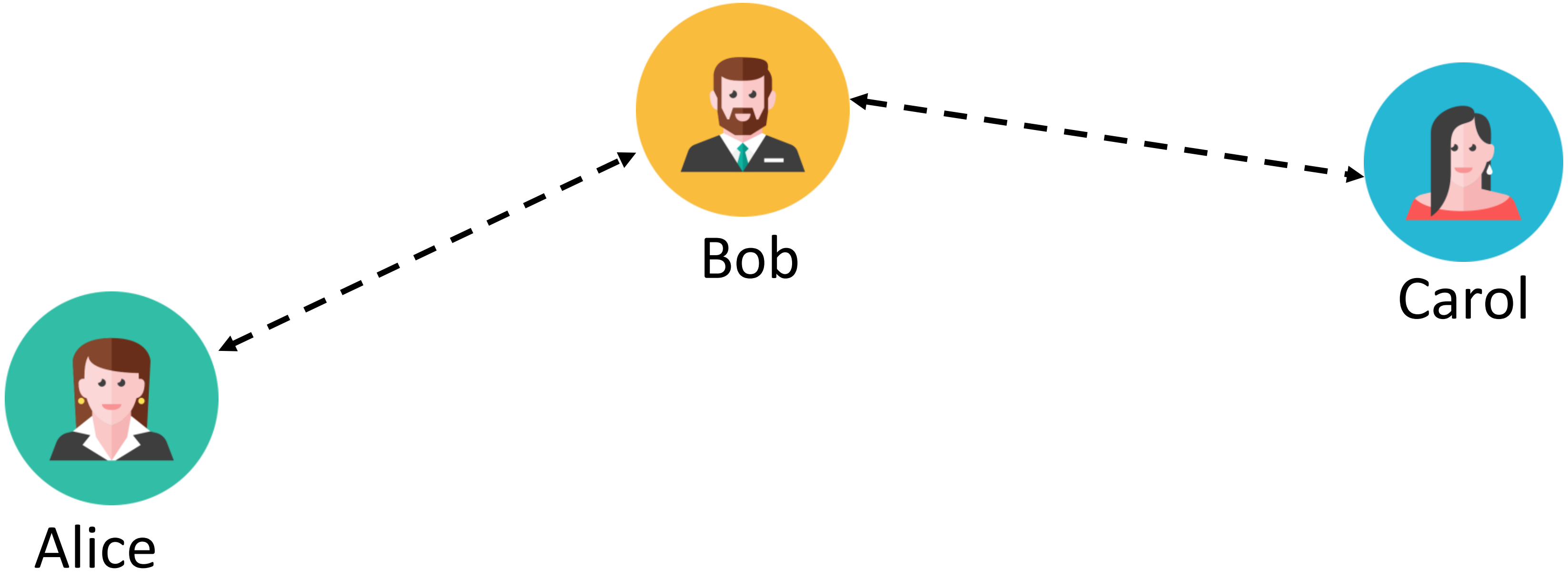
- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path
- **Our solution:** Lock payment path and execute multi-phase commit



# Multi-hop with asynchronous blockchain access

New multi-hop payment protocol:

- Maintains **asynchronous blockchain access**
- **Challenge:** Ensure atomic payments across multi-hop path
- **Our solution:** Lock payment path and execute multi-phase commit



# Mu ain access

New

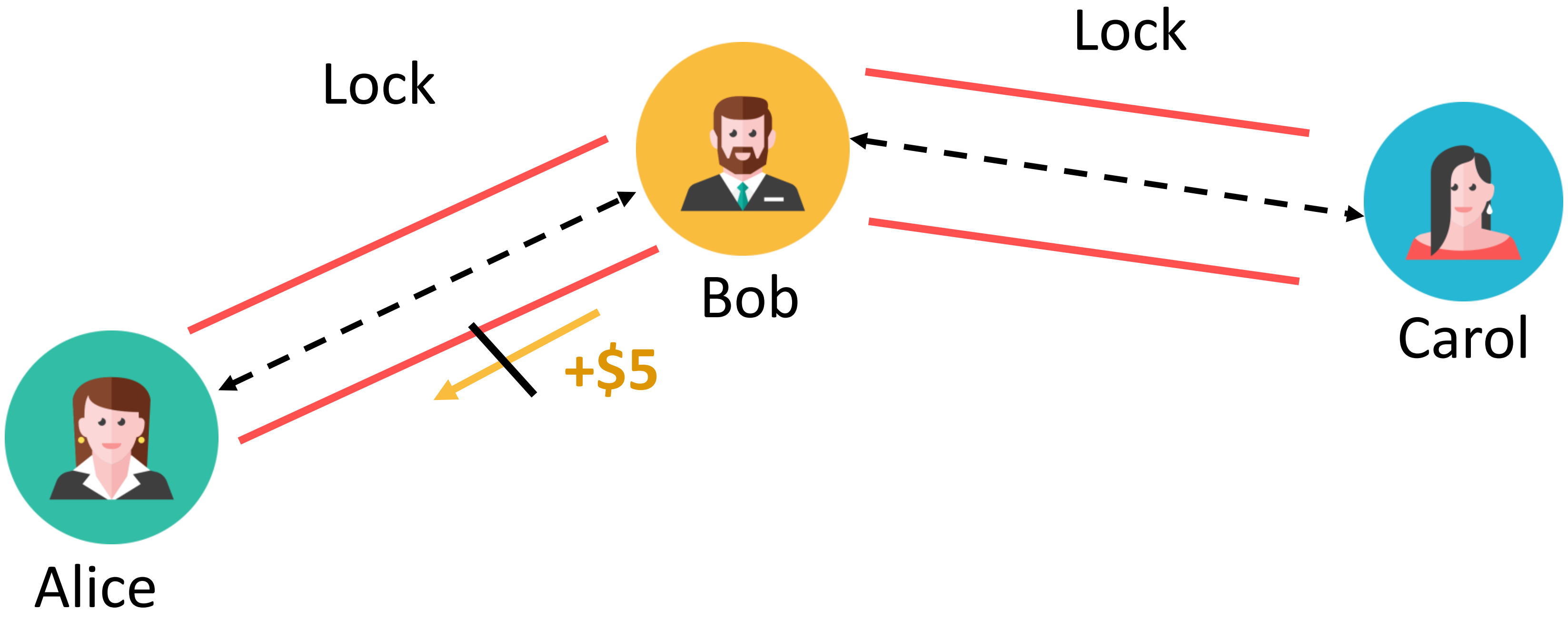
– M

– C

– C ommit

## Problem: No concurrent payments!

Blocking other payments reduces throughput



Mu ain access

New

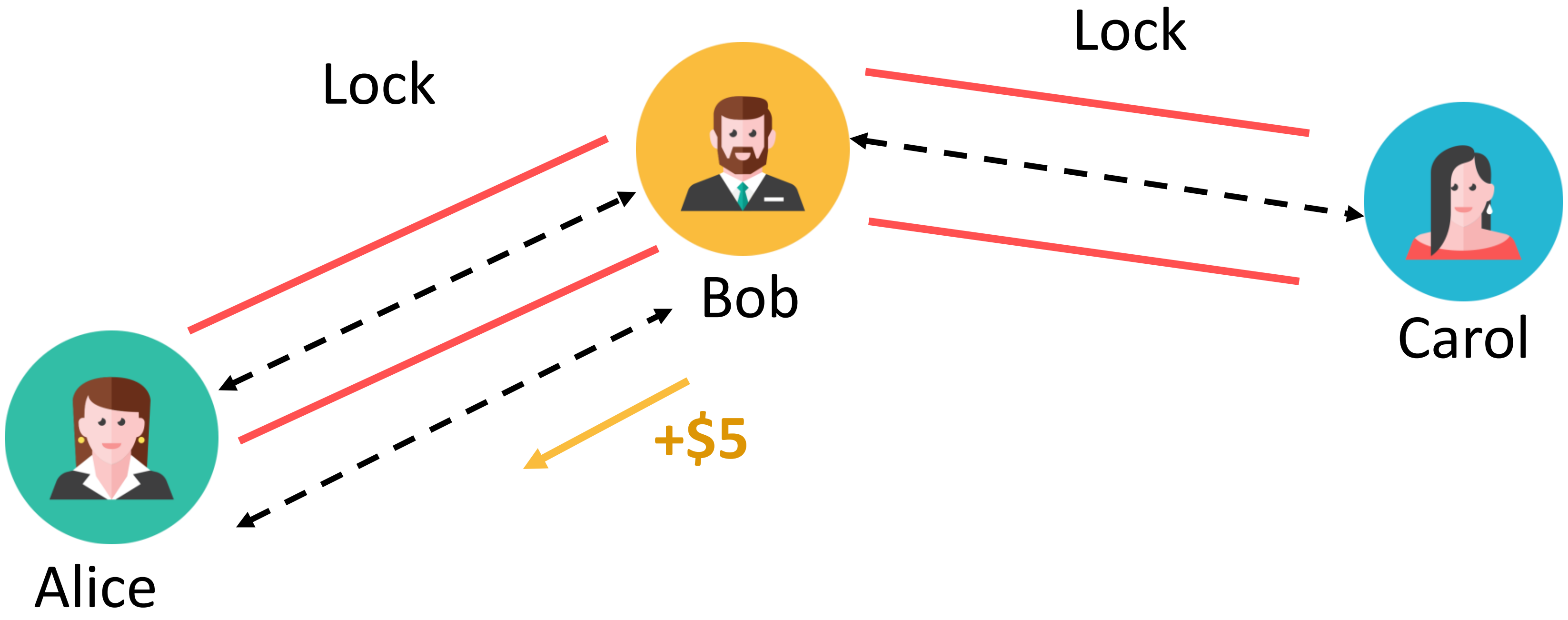
– M

– C

– C ommit

## Optimization: Temporary Channels!

Dynamically create channels quickly
to allow concurrent payments
(see the paper!)



Mu ain access

New

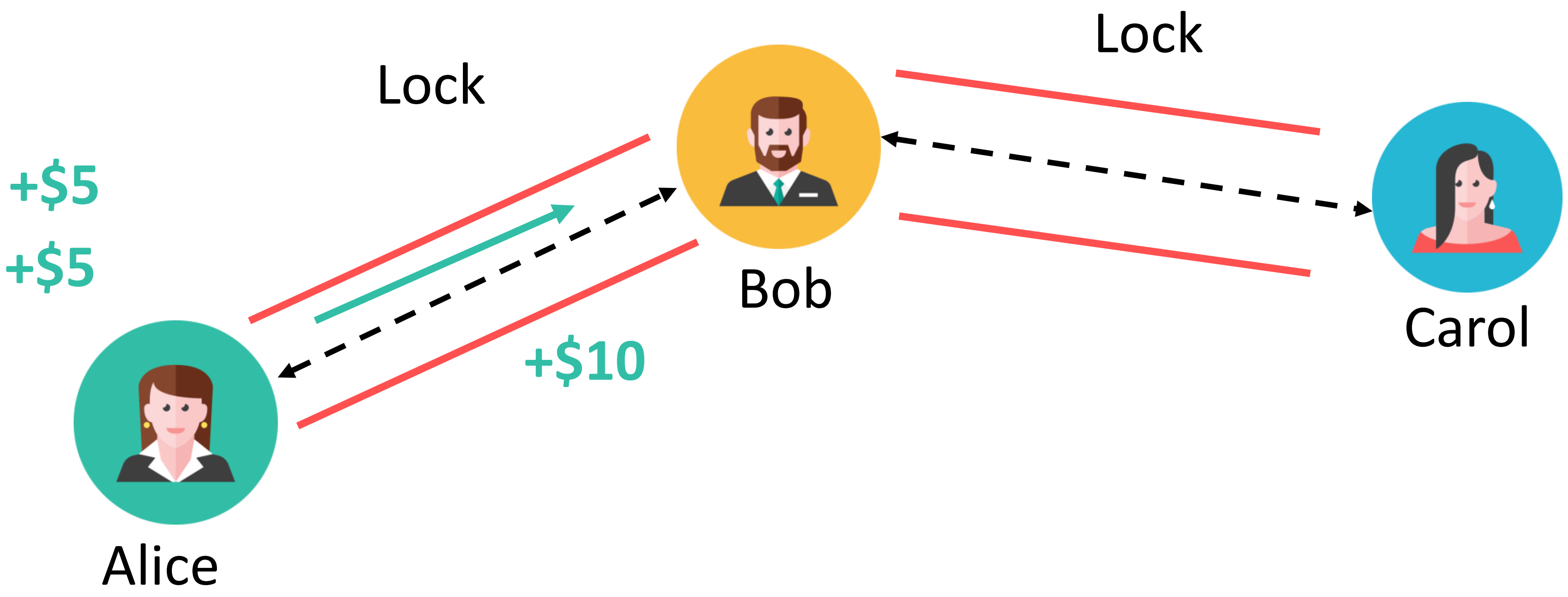
– M

– C

– C ommit

# Optimization 2: Transaction batching!

Batch payments from the same sender to the same recipient (see the paper!)



Lock
Lock
+$5
+$5
+$10
Bob
Carol
Alice