Teechain: A Secure Payment Network with Asynchronous Blockchain Access

Joshua Lind
Florian Kelbert

Oded Naor
Emin Gün Sirer

Ittay Eyal
Peter Pietzuch

joshua.lind11@imperial.ac.uk
Imperial College London
Blockchains aren’t scaling!

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

Max. throughput: 4 tx/s, latency: 10’s mins

Transactions per day

<table>
<thead>
<tr>
<th>Year</th>
<th>Transactions per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td></td>
</tr>
</tbody>
</table>
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

- Alice
- Carol
- Bob
- Dave
Off-chain scaling: Payment Networks

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

Payment channels:
- Point to point payments

---

Blockchain
Off-chain scaling: Payment Networks

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

Payment channels:
- Point to point payments
- Bi-directional payments

Blockchain
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

Alice ➔ Carol ➔ Dave
Carol ➔ Bob ➔ Alice

$5 ➔ $5 ➔ $5
$5 ➔ $2 ➔ $5

Blockchain
How do payment channels work?
- 3 phases: setup, payments, settlement
How do payment channels work?
- 3 phases: **setup**, payments, settlement

Alice

Bob

Write: deposit

Blockchain

Deposit $100
Background: Payment Channels

How do payment channels work?
- 3 phases: setup, payments, settlement

Alice
Pay: 1
A:$80
B:$20

Bob
+$20

Blockchain
deposit
$100
How do payment channels work?
- 3 phases: setup, **payments**, settlement

Alice

Pay: 1
A:$80
B:$20

Pay: 2
A:$85
B:$15

+$5

Bob

Blockchain

Deposit
$100
Background: Payment Channels

How do payment channels work?
– 3 phases: setup, payments, settlement

Alice

Pay: 1
A:$80
B:$20

Pay: 2
A:$85
B:$15

Pay: 3
A:$90
B:$10

+$5

Bob

Blockchain

Deposit

$100
Background: Payment Channels

How do payment channels work?
– 3 phases: setup, payments, settlement

Alice

Pay: 1
A:$80
B:$20

Pay: 2
A:$85
B:$15

Pay: 3
A:$90
B:$10

Bob

Write: final balance

Blockchain

Deposit
$100
**Roll-back attacks!**

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

---

**How do payment channels work?**

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

Alice ➔ Bob

Pay: 1

A: $80
B: $20

Pay: 2

A: $85
B: $15

Pay: 3

A: $90
B: $10

Write: old balance
Background: Payment Channels

Existing solutions to roll-back attacks:
- Monitor the blockchain (root-of-trust)
- React within reaction time ($\Delta$)
- Final balance on the blockchain

The root-of-trust

Blockchain

Deposit $100

Alice

Bob

Pay: 1
A: $80
B: $20

Pay: 2
A: $85
B: $15

Pay: 3
A: $90
B: $10
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

The **root-of-trust**
Background: Payment Channels

Existing solutions to **roll-back attacks**:
- Monitor the blockchain (**root-of-trust**)
- React within **reaction time** ($\Delta$)
- Final balance on the blockchain

![Diagram showing transactions between Alice and Bob]

The root-of-trust
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

The root-of-trust

Blockchain

Alice

Bob

Pay: 1
A:$80
B:$20

Pay: 2
A:$85
B:$15

Pay: 3
A:$90
B:$10

Pay: 2
A:$85
B:$15

Deposit $100

Read: old balance!
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

The root-of-trust

Write: final balance within reaction time Δ
Background: Reacting to roll-back attacks

Reaction times ($\Delta$) require synchronous blockchain access:

- **Assume**: parties can read/write within $\Delta$
- **But**: blockchains are best-effort. No read/write latency bounds!
Background: Reacting to roll-back attacks

Reaction times ($\Delta$) require synchronous blockchain access:

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

![Graph showing Bitcoin confirmation time (write latency) from 2016 to 2018](image)
Background: Reacting to roll-back attacks

Reaction times require Synchronous Blockchain Access:

- Assume: parties can read/write within $\Delta$
- But: Blockchains are best-effort. No read/write latency bounds!

Spam/Congestion attack!

Transactions took > 7 days to be written to the blockchain!
Background: Reacting to roll-back attacks

Reactions times require Synchronous Blockchain Access:

- Assume: parties can read/write within $\Delta$
- But: Blockchains are best-effort. No read/write latency bounds!

**Attackers can manipulate latencies:**

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

---

![Graph showing Bitcoin confirmation time (write latency) from 2016 to 2018.](image)
What value for reaction time ($\Delta$)?

Trade-off:

Large $\Delta$ (weeks): hard to attack, slow fund access
Small $\Delta$ (mins): easy to attack, quick fund access

Write: final balance within $\Delta$

Pay: 2
A: $85
B: $15

Pay: 3
A: $90
B: $10
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

**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*
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!

Alice

Blockchain

Deposit $100

Treasury

Bob

Write: deposit
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!

Alice

\[ A: 100 \quad B: 0 \]

Bob
Introduce another root-of-trust (RoT): treasury
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once → prevents roll-backs!

Alice

A: 100   B: 0

+$5

Bob

Treasury

Deposit $100
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): \textit{treasury}
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels once $\rightarrow$ prevents roll-backs!

Alice \hspace{1cm} Treasury \hspace{1cm} Bob

A: 95 \hspace{1cm} B: 5

Settle

Blockchain

Deposit $100
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

Alice
Treasury
Bob

Deposit $100
Settle
A:$95
B:$5
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!

Write: final balance
Challenge 1: Removing the blockchain as RoT

- Introduce another root-of-trust (RoT): Treasury
  - Controls funds, balances, and payments
  - Prevents misbehaviour
  - Only settles channels once → Prevents rollbacks!

Alice \rightarrow \text{Blockchain} \rightarrow Bob

Deposit $100

No roll-backs → no reaction times:
Asynchronous blockchain access

Treasury prevents roll-backs!

Bob: Write final balance

Settle
A: $95
B: $5

- Unbounded latency!
Challenge 1: Removing the blockchain as RoT

Introduce another root-of-trust (RoT):
- Controls funds, balances and payments
- Prevents misbehaviour
- Only settle channels → Prevents rollbacks!

Alice

Bob

Can we realize treasuries?

How do we realize treasuries for blockchains?

Treasury

Blockchain

Deposit $100

Settle
A:$95
B:$5

Write: Final Balance
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!)
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 \( m\text{-out-of-}n \) transactions
Challenge 2: Realizing treasuries for blockchains

Treasury committee:
- Choose \( n \) parties in the network
- Require \( m \) parties to agree before accessing funds
- Use \( m\text{-out-of-}n \) transactions

Blockchain

Alice \( \rightarrow \) Treasury Committee

Write: deposit

Bob
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

Alice

Bob

Treasury Committee

Blockchain

3-out-of-4 Deposit $100

A: 100  B: 0
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

Alice

Bob

Treasury Committee

Deposit $100

3-out-of-4

+$5

A: 100   B: 0
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

Blockchain
... 3-out-of-4
Deposit $100

Alice

Bob

Treasury Committee
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

Alice

Bob

Treasury Committee

Settle

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

Alice

Bob

Blockchain

3-out-of-4
Deposit
$100

Settle
A:$95
B:$5
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...

3-out-of-4 Deposit $100

Blockchain...

Settle
A: $95
B: $5

Alice

Bob

Write: final balance

Treasury Committee
Trust is distributed!

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

**Treasury committee:**
- Choose $n$ parties
- Require $m$ parties to agree before accessing funds
- Use $m$-out-of-$n$ transactions

Blockchain

3-out-of-4 Deposit $100

Alice

Bob

Treasury Committee
Treasury committee size?

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

Alice → Treasury Committee → Bob

Blockchain:

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

Settle:

- A: $95
- B: $5
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
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

“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
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!

![Diagram showing Alice adding $5 to the treasury committee funds, which is then matched by Bob]
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!)

Alice

$+$5

+$5

Bob

Treasury Committee
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

![Graph showing maximum throughput values]

- 150000
- 100000
- 50000
- 0
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

<table>
<thead>
<tr>
<th>Throughput (tx/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>150000</td>
</tr>
<tr>
<td>100000</td>
</tr>
<tr>
<td>50000</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

Lightning Network
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)

<table>
<thead>
<tr>
<th>Committee Size</th>
<th>Maximum Throughput (tx/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>London, New York, Haifa</td>
<td>130,000</td>
</tr>
</tbody>
</table>

**Diagram:**

- Throughput (tx/s)
- Techain (**n=1**): 130,000
- Lightning Network: 1000
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**

<table>
<thead>
<tr>
<th>Throughput (tx/s)</th>
<th>Lightning Network</th>
<th>Teechain (n=1)</th>
<th>Teechain (n=2)</th>
<th>Teechain (n=3)</th>
<th>Teechain (n=4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
<td>130,000</td>
<td>34,000</td>
<td>33,000</td>
<td>33,000</td>
</tr>
</tbody>
</table>
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)

- Throughput is limited! Throughput is limited by the CPU for chain replication

<table>
<thead>
<tr>
<th>Throughput (tx/s)</th>
<th>Lightning Network</th>
<th>Teechain (n=1)</th>
<th>Teechain (n=2)</th>
<th>Teechain (n=3)</th>
<th>Teechain (n=4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>150000</td>
<td>1000</td>
<td>34,000</td>
<td>33,000</td>
<td>33,000</td>
<td>33,000</td>
</tr>
<tr>
<td>110000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Latency (milliseconds)

<table>
<thead>
<tr>
<th>Throughput</th>
<th>London</th>
<th>New York</th>
<th>Haifa</th>
</tr>
</thead>
<tbody>
<tr>
<td>387</td>
<td>86</td>
<td>292</td>
<td>415</td>
</tr>
<tr>
<td>608</td>
<td>53</td>
<td>672</td>
<td>387</td>
</tr>
<tr>
<td>800</td>
<td>149</td>
<td>222</td>
<td>53</td>
</tr>
</tbody>
</table>
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 latency

![Graph showing maximum latency](image-url)
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

<table>
<thead>
<tr>
<th>Latency (milliseconds)</th>
<th>Lightning Network</th>
<th>Teechain (n=1)</th>
<th>Teechain (n=2)</th>
<th>Teechain (n=3)</th>
<th>Teechain (n=4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>387</td>
<td>86</td>
<td>292</td>
<td>415</td>
<td>672</td>
</tr>
</tbody>
</table>
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)

<table>
<thead>
<tr>
<th></th>
<th>Latency (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightning Network</td>
<td>387</td>
</tr>
<tr>
<td>Teechain (n=1)</td>
<td>86</td>
</tr>
<tr>
<td>Teechain (n=2)</td>
<td>292</td>
</tr>
<tr>
<td>Teechain (n=3)</td>
<td>415</td>
</tr>
<tr>
<td>Teechain (n=4)</td>
<td>672</td>
</tr>
</tbody>
</table>

**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](image)

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

Payment network deployment:
  - **Workload**: Bitcoin transaction history across graph

Committee chains

Throughput is limited by cost to replicate
Payment network deployment:

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

Throughput scales linearly:

- **30 machines:**
  - n=3 committee members
  - n=4 committee members

### Throughput (tx/s)

- **1 million tx/s**

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

**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**
Create temporary channels to avoid payment contention (see the paper!)
Does Teechain scale out?

Payment network deployment:
- **Workload**: Bitcoin transaction history across graph

![Diagram showing hub-and-spoke graph with nodes labeled Large, Medium, Small, and临时通道标注。Large/medium nodes use temporary channel optimization.](image)

![Graph showing throughput vs. maximum number of temporary channels.](image)

<table>
<thead>
<tr>
<th>Maximum number of temporary channels</th>
<th>Throughput (tx/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>500</td>
</tr>
<tr>
<td>6</td>
<td>750</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>10</td>
<td>1250</td>
</tr>
</tbody>
</table>
Does Teechain scale out?

Payment network deployment:
- **Workload**: Bitcoin transaction history across graph

**Diagram:**
- **Hub-and-spoke graph**: Large/medium nodes use temporary channel optimization
- **Throughput (tx/s)**
  - Baseline (no temporary channels)
  - n=4 committee members

**Throughput (tx/s)**
- 0
- 250
- 500
- 750
- 1000
- 1250

**Maximum number of temporary channels**
- 2
- 4
- 6
- 8
- 10
Does Teechain scale out?

Payment network deployment:
- **Workload:** Bitcoin transaction history across graph

**Temporary channels**

- **Large** nodes use temporary channel optimization

**Hub-and-spoke graph**

Large/medium nodes use temporary channel optimization

**Graph**

**Throughput (tx/s)**

- For **n=3 committee members**
  - Throughput: 250 tx/s

- For **n=4 committee members**
  - Throughput: 750 tx/s

**Graph**

**Maximum number of temporary channels**

- **n=4 committee members**
  - Throughput ranges from 250 to 1250 tx/s
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

Throughput (tx/s) vs. Maximum number of temporary channels

- n=4 committee members
Payment network deployment:
- **Workload**: Bitcoin transaction history across graph

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

**Best performance**
Performance requires high connectivity!

![Graph](image)

**Throughput (tx/s)**

- Large nodes: n=4 committee members

![Graph](image)

**Maximum number of temporary channels**

- 0
- 2
- 4
- 6
- 8
- 10
Summary

Blockchains are best-effort:
- Security shouldn’t rely on read/write latencies!
- Assume asynchronous blockchain access

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

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:

Alice → Treasury Committee → Bob

+$5

Alice

Bob
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: \((100 + 30) - 5 = 125\)
Payment Network deployment:
- **Workload**: Bitcoin transaction history across graph

Throughput is Limited

Throughput is limited by replication costs

- Throughput is limited by replication costs

**Diagram:**
- Throughput in tx/s vs. Maximum number of temporary channels
- Blue line: n=3 committee members
- Red line: n=4 committee members

**Legend:**
- Blue line: n=3 committee members
- Red line: n=4 committee members

**Notes:**
- Large/medium nodes use temporary channel optimization
- Hub-and-Spoke graph
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
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
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
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
New multi-hop payment protocol:

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

Problem: No concurrent payments!

Blocking other payments reduces throughput
Optimization: Temporary Channels!

Dynamically create channels quickly to allow concurrent payments (see the paper!)
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 modified multi-phase commit

Optimization 2: Transaction batching!

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

+$10
+$5
+$5

Alice
Bob
Carol