TxProbe: Discovering Bitcoin's Network Topology Using Orphan Transactions

Sergi Delgado-Segura, Surya Bakshi, Cristina Pérez-Solà, James Litton, Andrew Pachulski, Andrew Miller and Bobby Bhattacharjee

sr_gi

MOTIVATION

What topology reconstruction seeks to answer?

- Is the network really decentralized?
- Are there supernodes in the network?
- Are there weak spots that can be easily isolated?

Currently, we do not know

WHAT DO WE KNOW ABOUT THE TOPOLOGY?

WHAT DO WE KNOW ABOUT THE TOPOLOGY?

Number of nodes and location of them

WHAT DO WE KNOW ABOUT THE TOPOLOGY?

Number of nodes and location of them

GLOBAL BITCOIN NODES DISTRIBUTION

Reachable nodes as of Thu Feb 07 2019 10:26:44 GMT+0000 (Greenwich Mean Time).

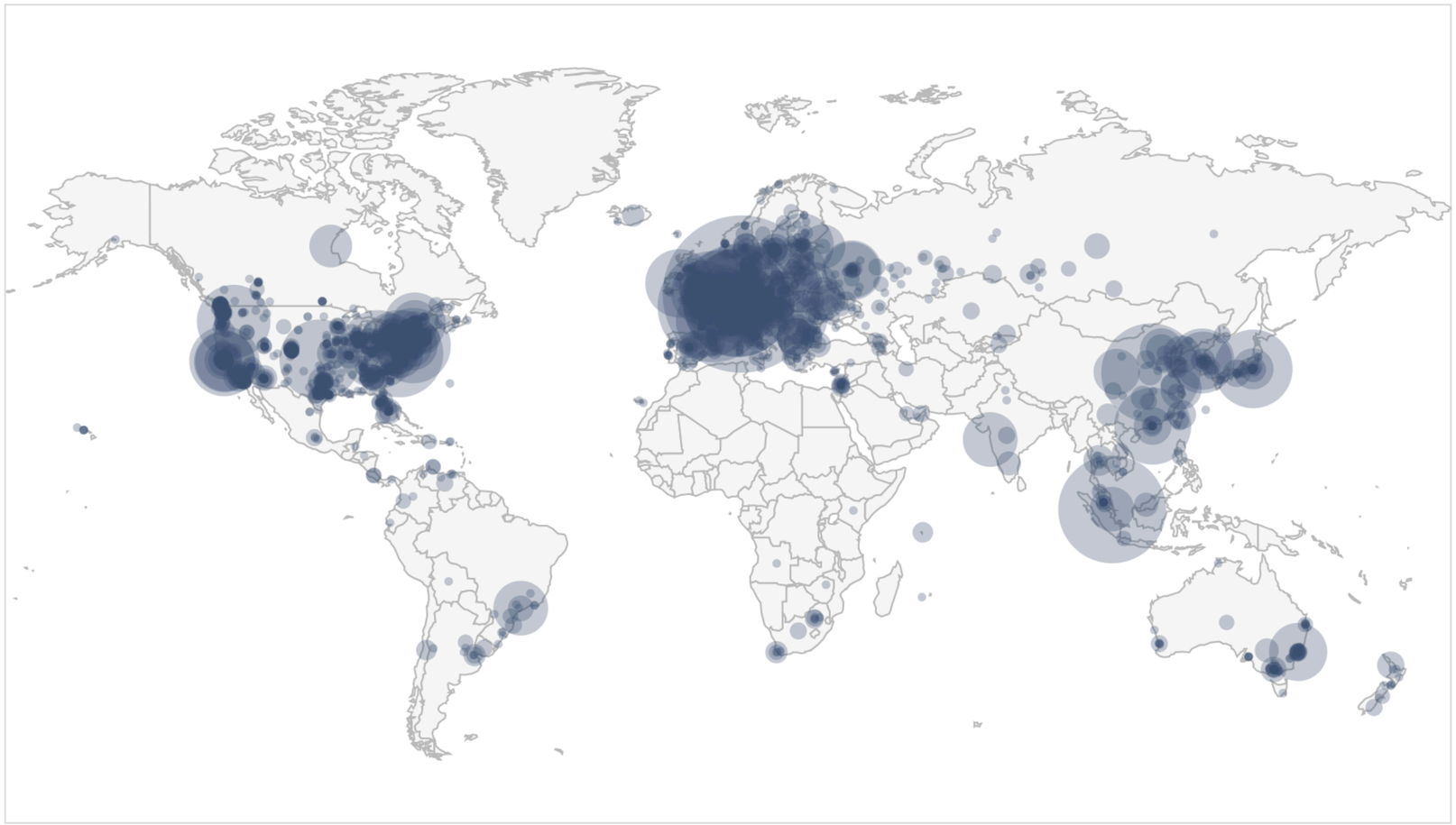
10365 NODES

24-hour charts »

Top 10 countries with their respective number of reachable nodes are as follow.

RANK	COUNTRY	NODES
1	United States	2570 (24.79%)
2	Germany	1968 (18.99%)
3	France	689 (6.65%)
4	Netherlands	514 (4.96%)
5	China	411 (3.97%)
6	Canada	384 (3.70%)
7	United Kingdom	355 (3.42%)
8	Singapore	321 (3.10%)
9	Russian Federation	277 (2.67%)
10	Japan	228 (2.20%)

More (100) »



THE TOPOLOGY SHOULD LOOK RANDOM

How Bitcoin (Core client) nodes choose their peers?

- Pseudorandomly from the **addrman**
- **8 outbound** connections by default

  No pair of nodes in the same **/16** (IPv4)
- **117 inbound** connection by default (no IP restriction here)

Bitcoin forks based on the Core client follow the same approach

TRANSACTION PROPAGATION IN BITCOIN

Our inferring technique is based on transaction propagation

We take advantage of how some kind of transactions (**orphans** and **double-spending)** are handled by nodes

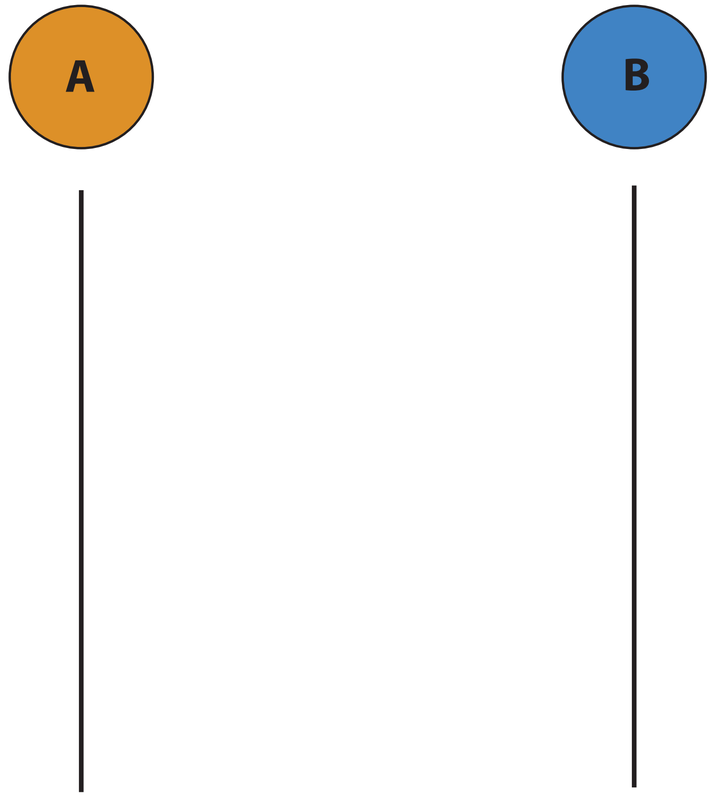
How does it roughly work?

TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

Transaction in mempool are **eventually propagated** throughout the node neighborhood

A node will **wait up to 2 minutes** for a response to a **getaddr message** before asking any other peer

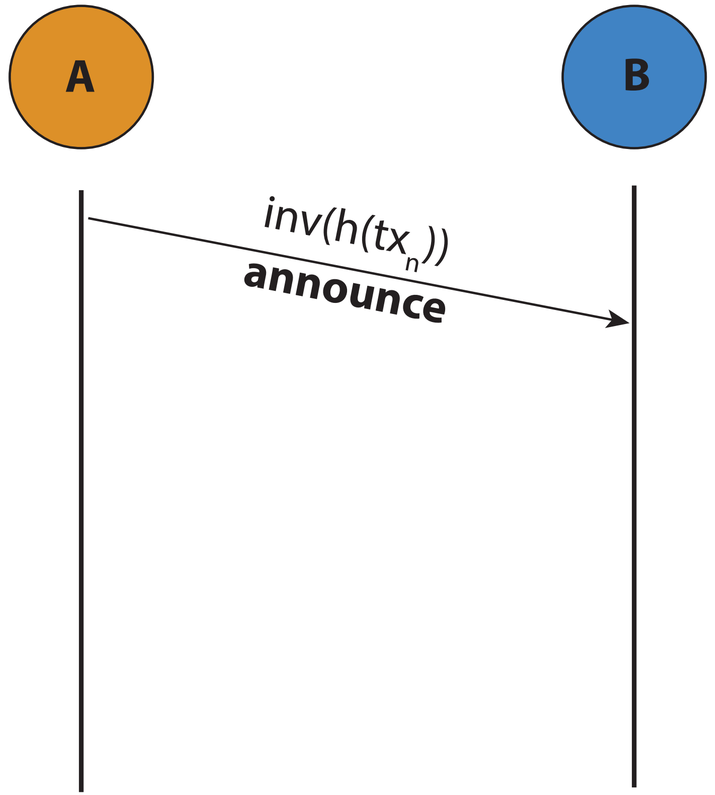


TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

Transaction in mempool are **eventually propagated** throughout the node neighborhood

A node will **wait up to 2 minutes** for a response to a **getaddr message** before asking any other peer

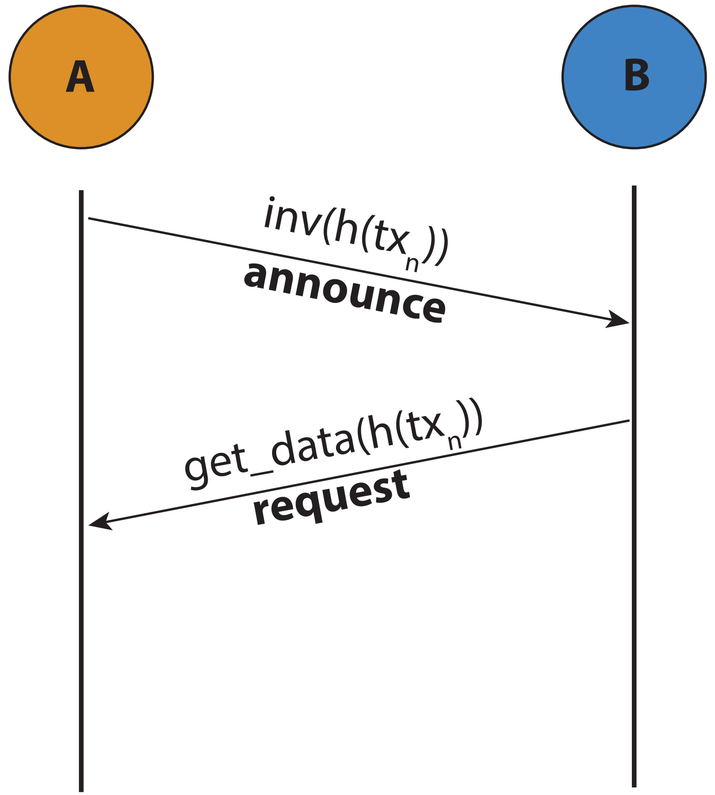


TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

Transaction in mempool are **eventually propagated** throughout the node neighborhood

A node will **wait up to 2 minutes** for a response to a **getaddr message** before asking any other peer

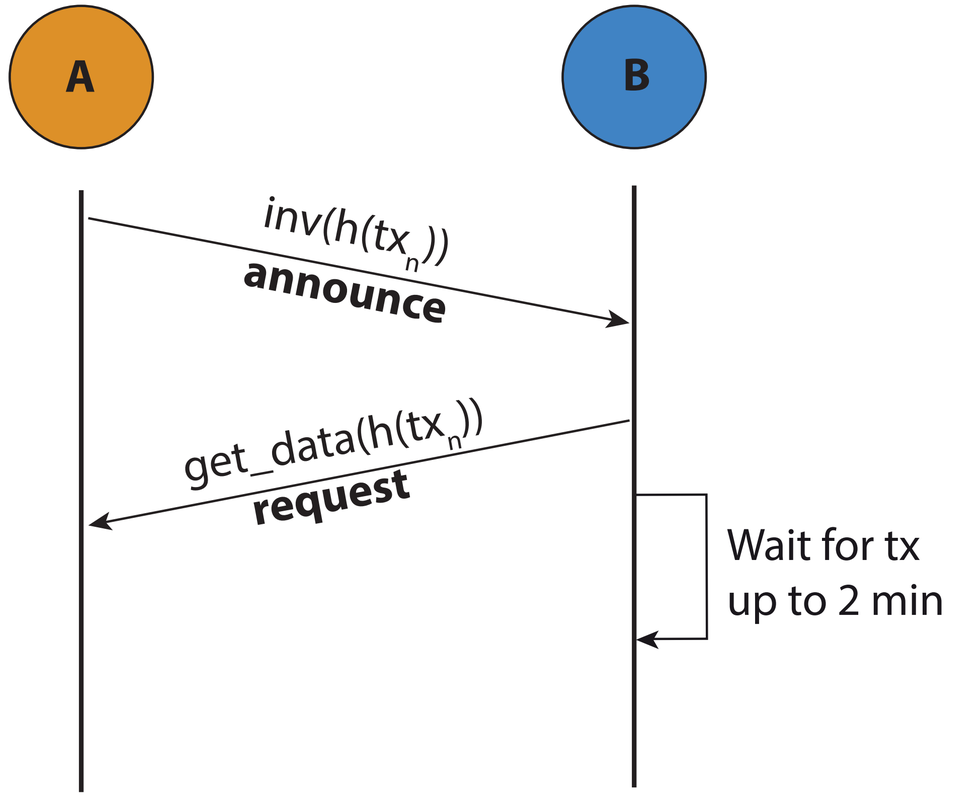


TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

Transaction in mempool are **eventually propagated** throughout the node neighborhood

A node will **wait up to 2 minutes** for a response to a **getaddr message** before asking any other peer

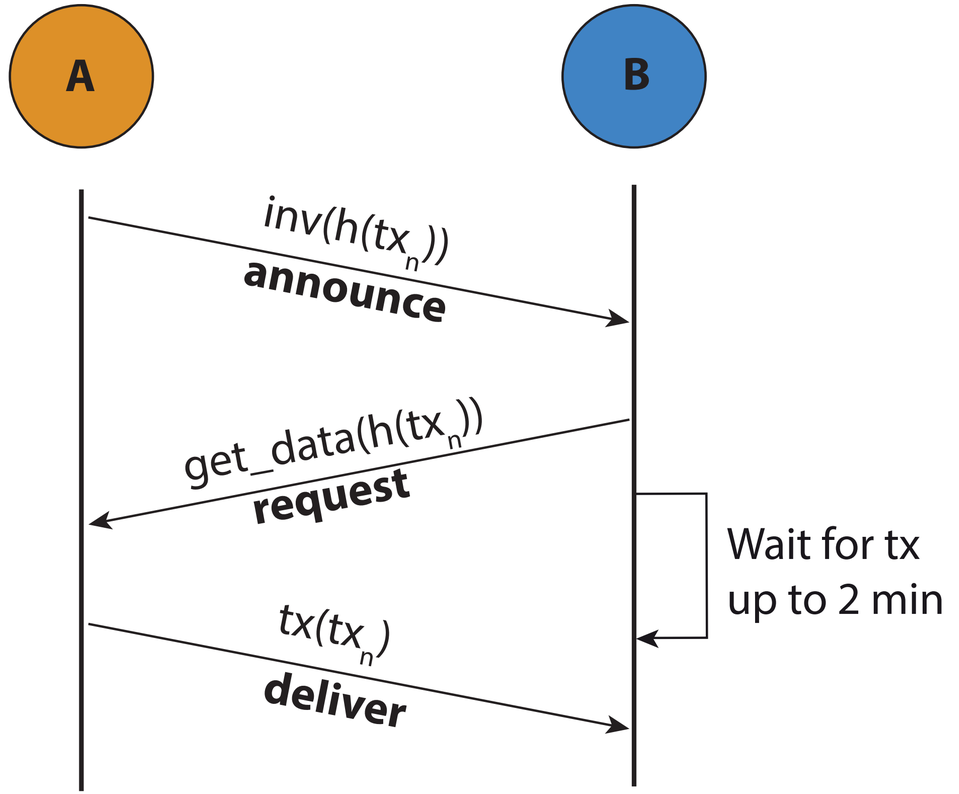


TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

Transaction in mempool are **eventually propagated** throughout the node neighborhood

A node will **wait up to 2 minutes** for a response to a **getaddr message** before asking any other peer



ORPHAN TRANSACTIONS

ORPHAN TRANSACTIONS

A transaction is orphan if **some of the referenced UTXOs are unknown**

ORPHAN TRANSACTIONS

A transaction is orphan if **some of the referenced UTXOs are unknown**

They can not be validated, so they are stored in a separated data structure known as **MapOrphanTransactions**

ORPHAN TRANSACTIONS

A transaction is orphan if **some of the referenced UTXOs are unknown**

They can not be validated, so they are stored in a separated data structure known as **MapOrphanTransactions**

Transactions in MapOrphanTransactions are **NOT forwarded to any node**

ORPHAN TRANSACTIONS

A transaction is orphan if **some of the referenced UTXOs are unknown**

They can not be validated, so they are stored in a separated data structure known as **MapOrphanTransactions**

Transactions in MapOrphanTransactions are **NOT forwarded to any node**

If the same transactions is offered again to the node **(inv message)**, it will not ask back for it **(getdata)**

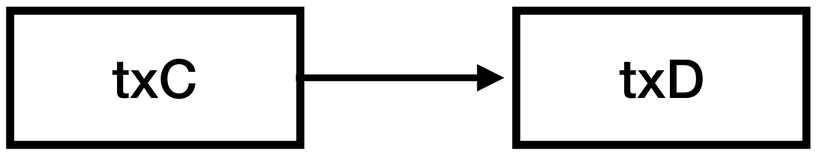
ORPHAN TRANSACTIONS

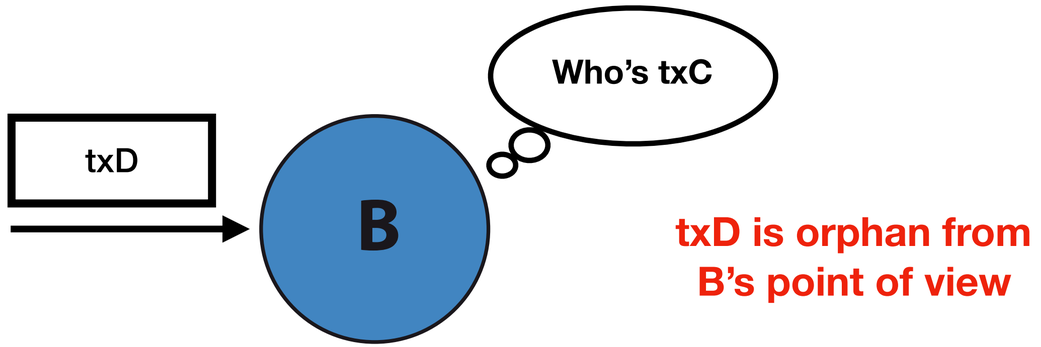
A transaction is orphan if **some of the referenced UTXOs are unknown**

They can not be validated, so they are stored in a separated data structure known as **MapOrphanTransactions**

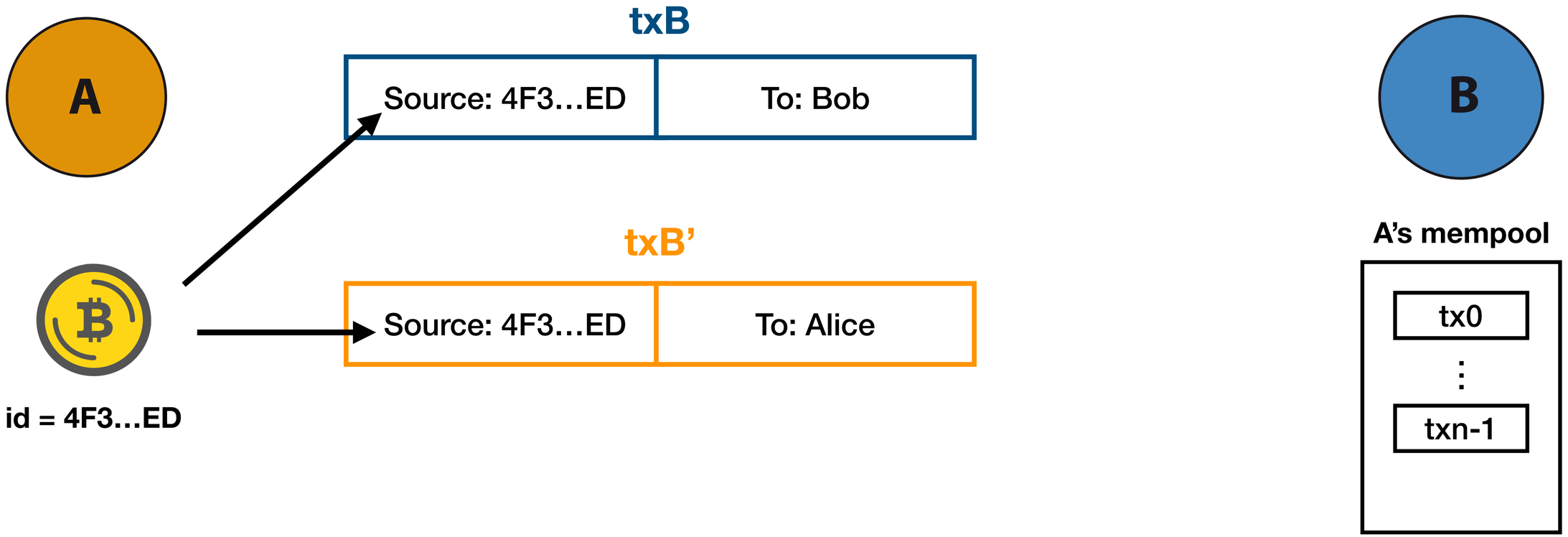
Transactions in MapOrphanTransactions are **NOT forwarded to any node**

If the same transactions is offered again to the node **(inv message)**, it will not ask back for it **(getdata)**

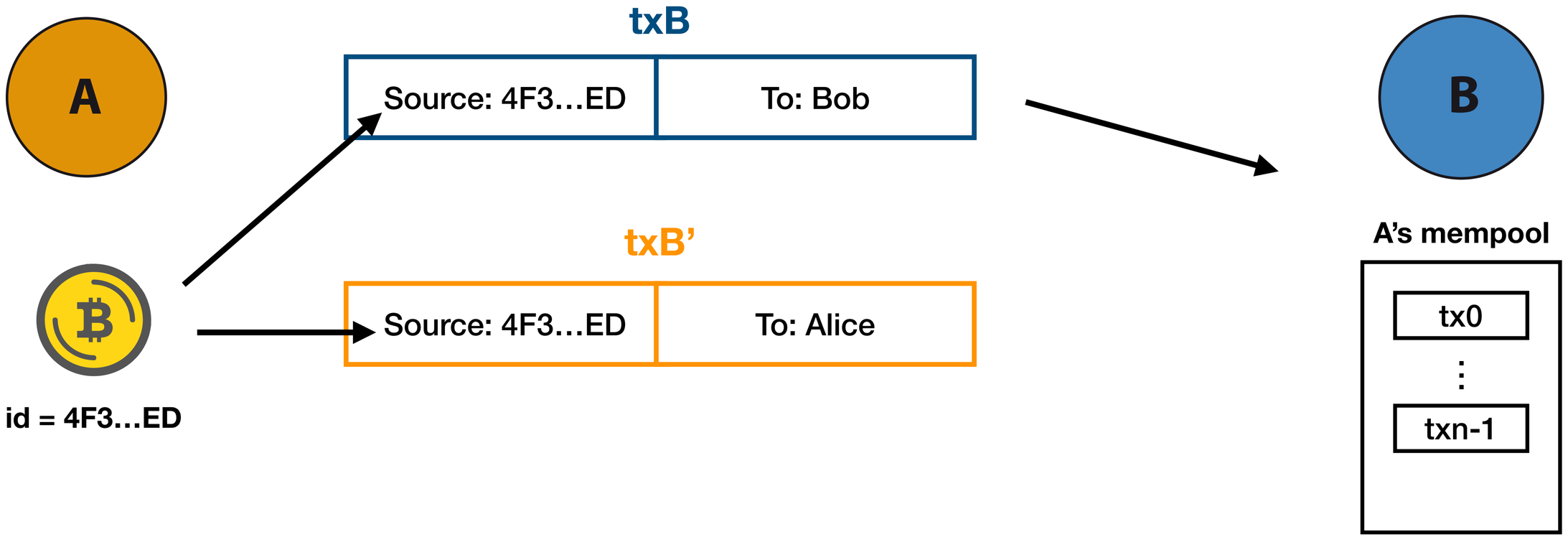




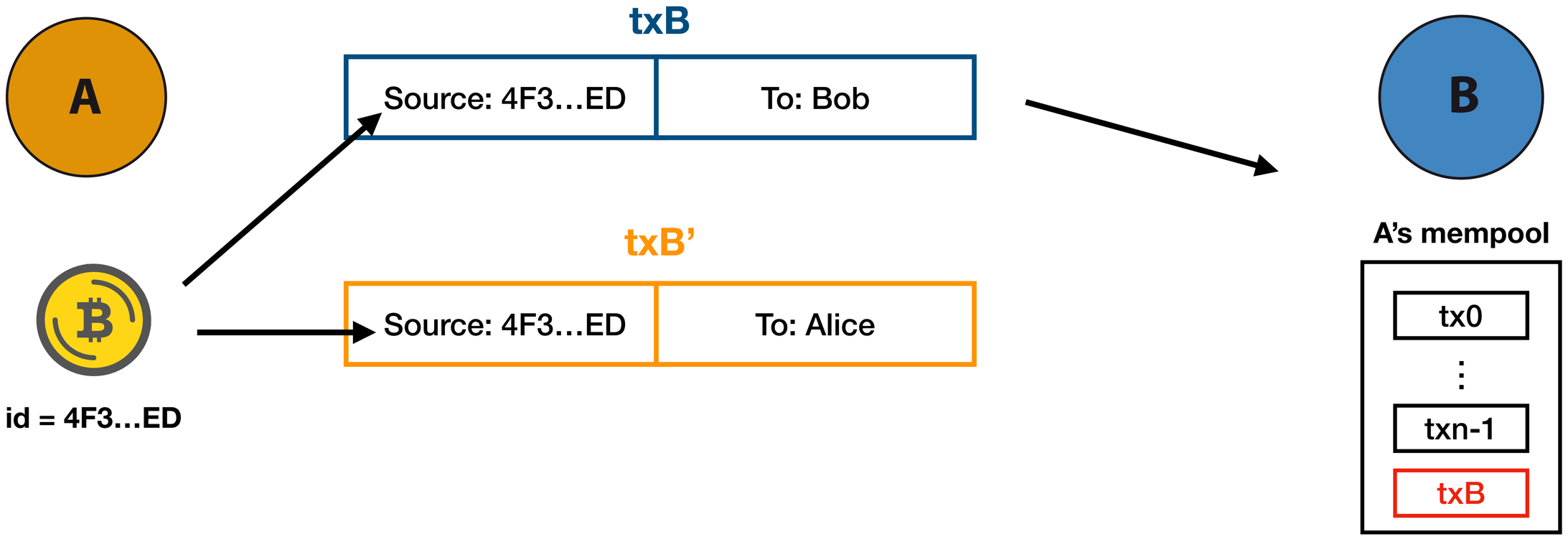
DOUBLE-SPENDING TRANSACTIONS



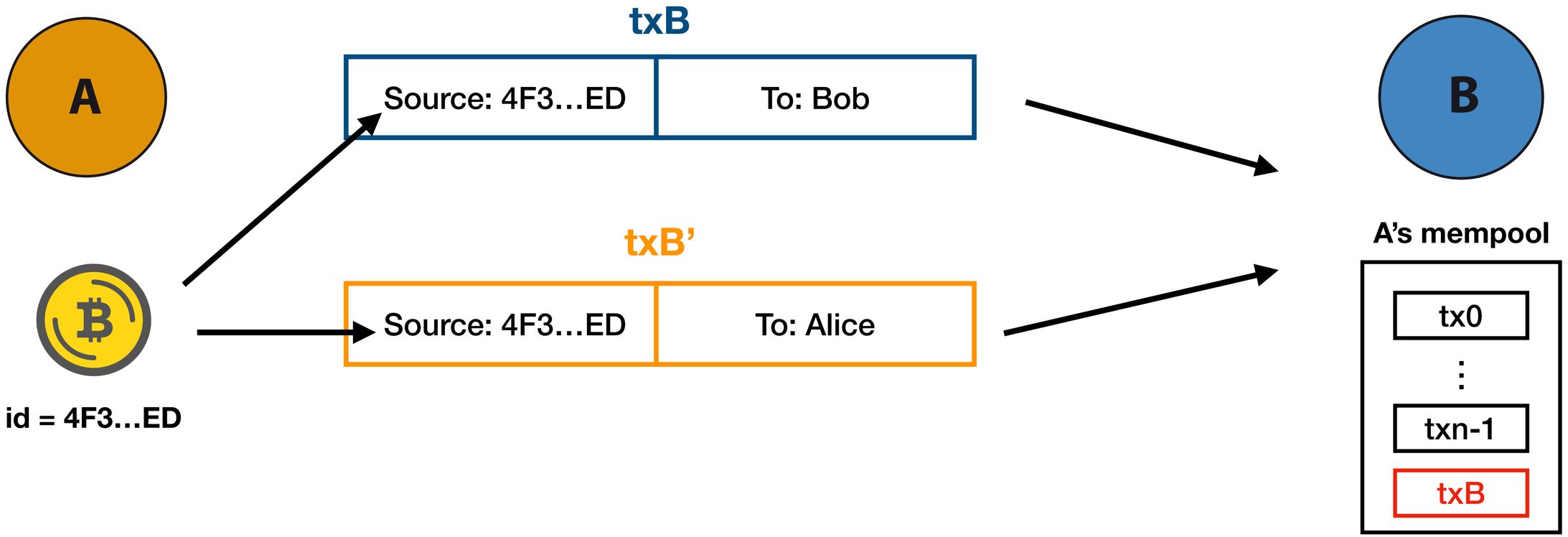
DOUBLE-SPENDING TRANSACTIONS



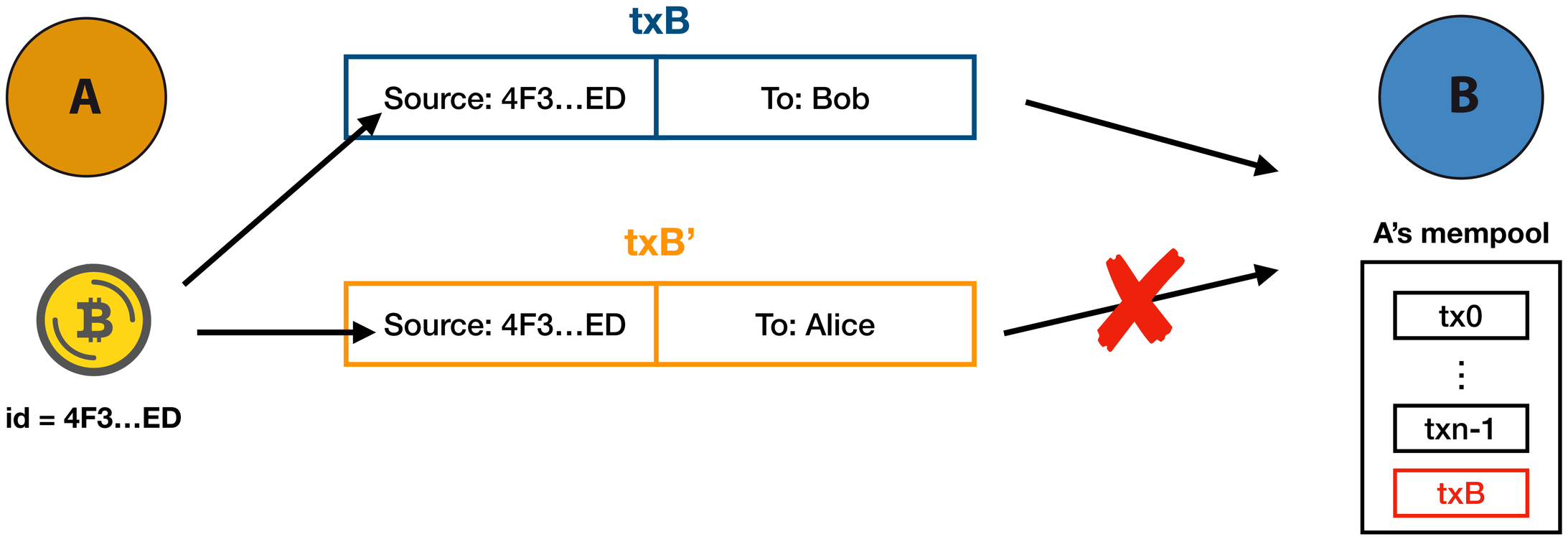
DOUBLE-SPENDING TRANSACTIONS



DOUBLE-SPENDING TRANSACTIONS



DOUBLE-SPENDING TRANSACTIONS



A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

Three transactions

Observation tool

A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

Observation tool

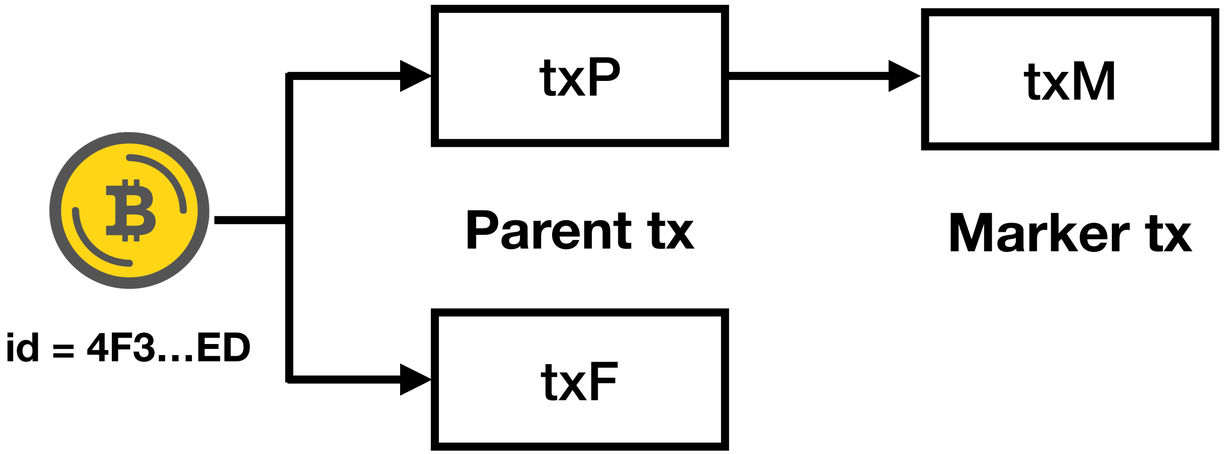
Three transactions

A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

Observation tool

Three transactions

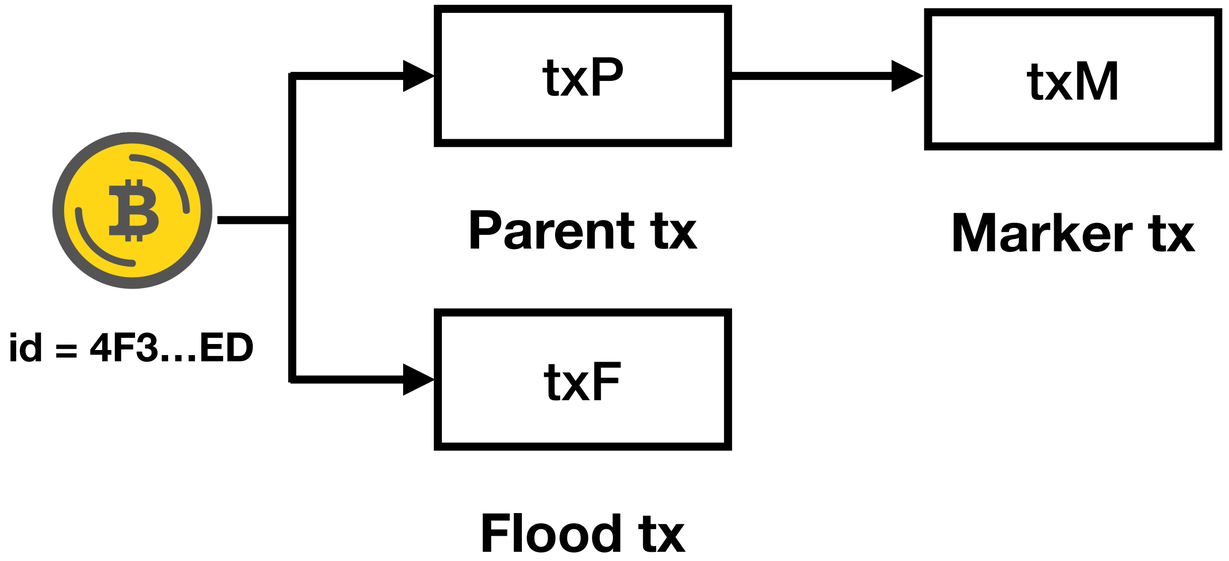


A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

Observation tool

Three transactions

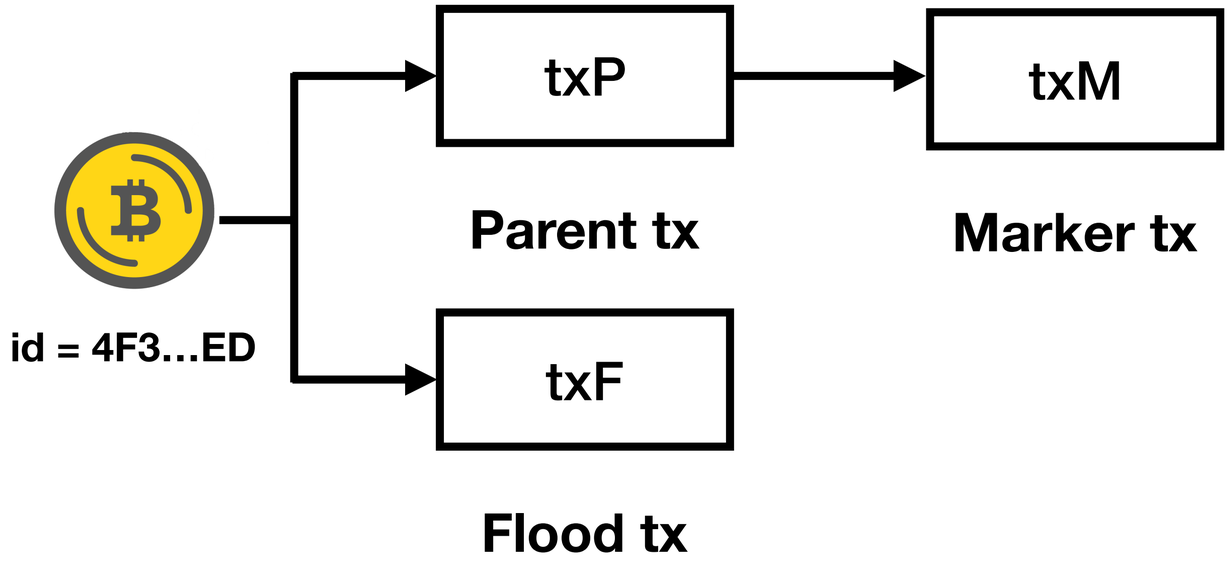


A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

Observation tool

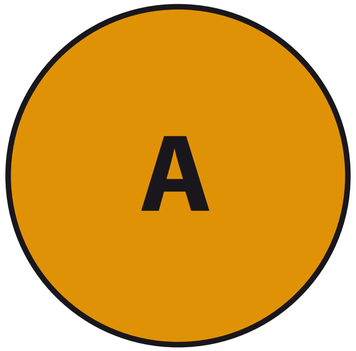
Three transactions

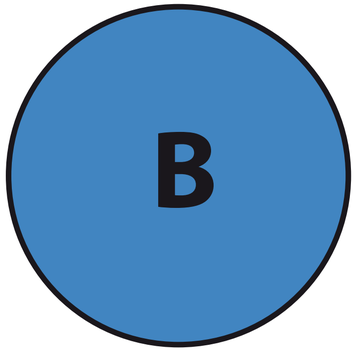


Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, Bobby Bhattacharjee

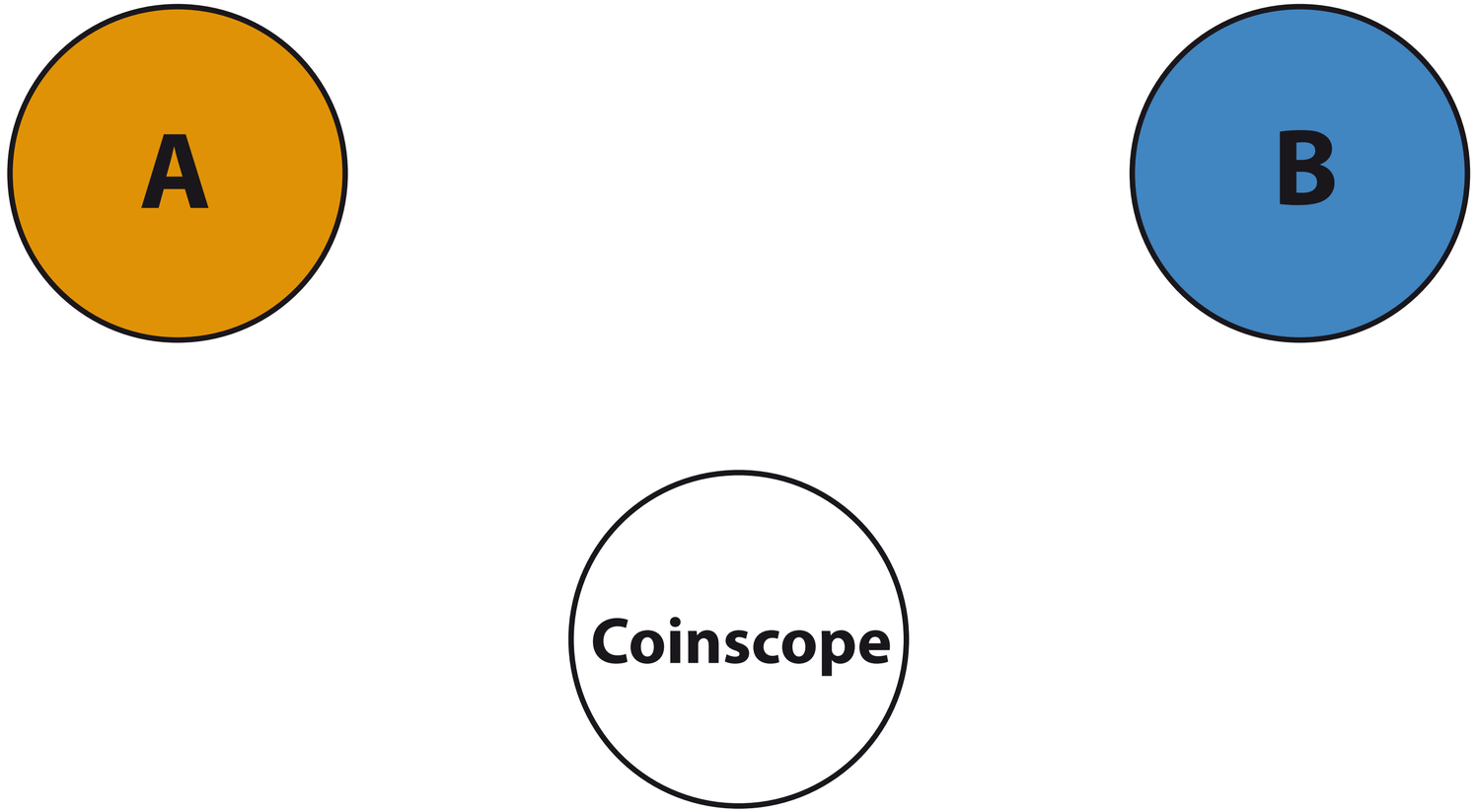
Discovering Bitcoin's Public Topology and Influential Nodes

NEGATIVE INFERRING TECHNIQUE

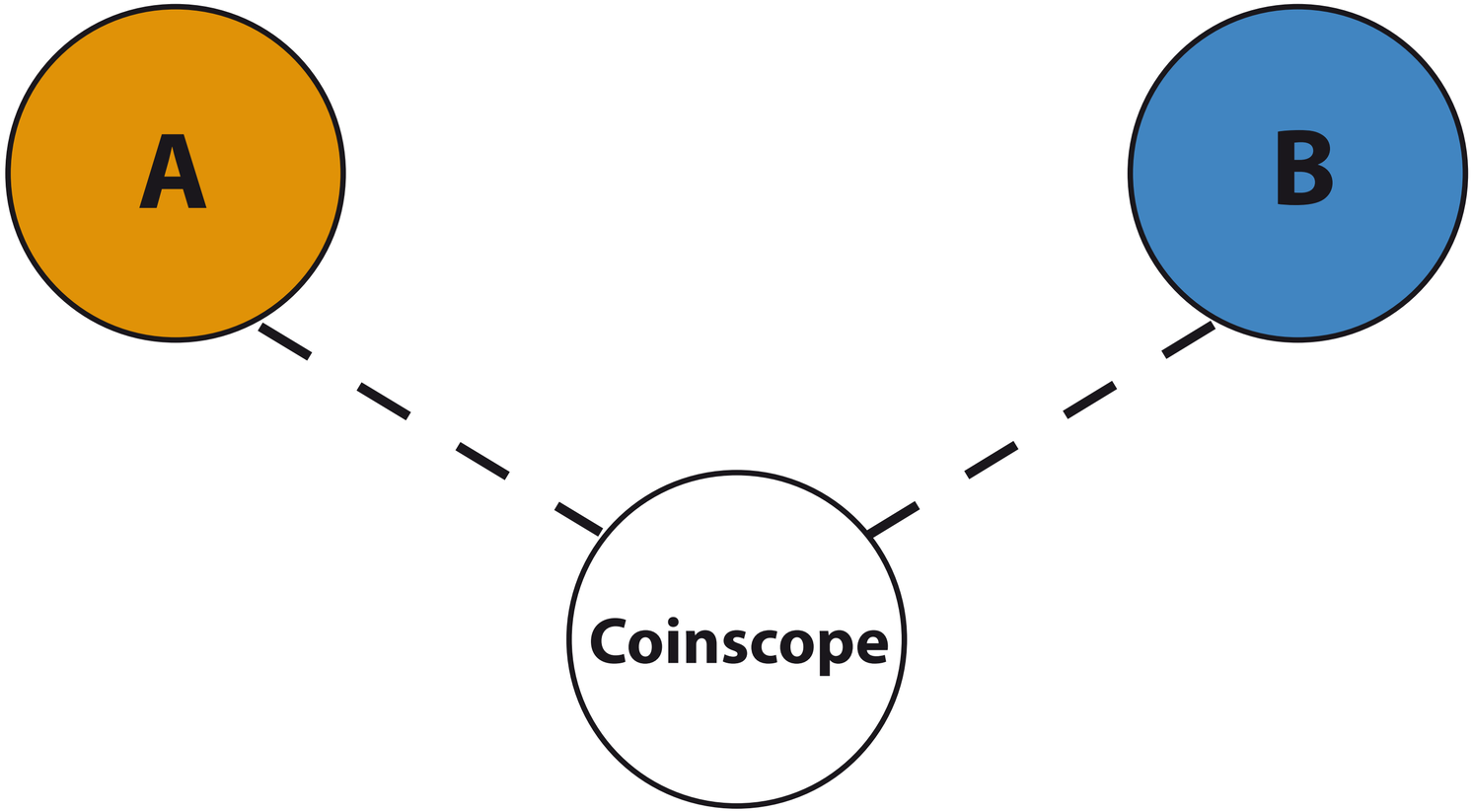




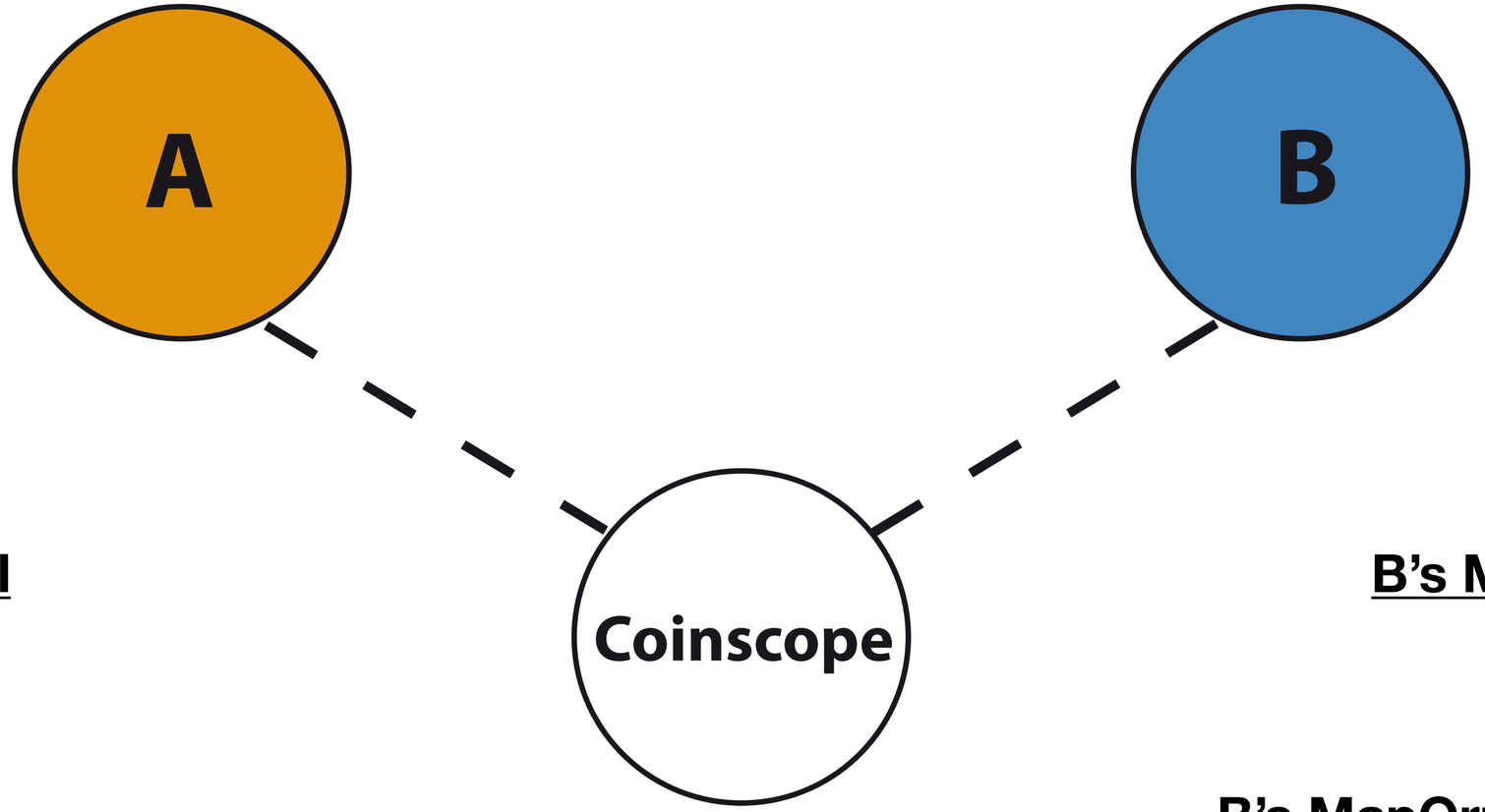
NEGATIVE INFERRING TECHNIQUE



NEGATIVE INFERRING TECHNIQUE



NEGATIVE INFERRING TECHNIQUE



A's Mempool

∅

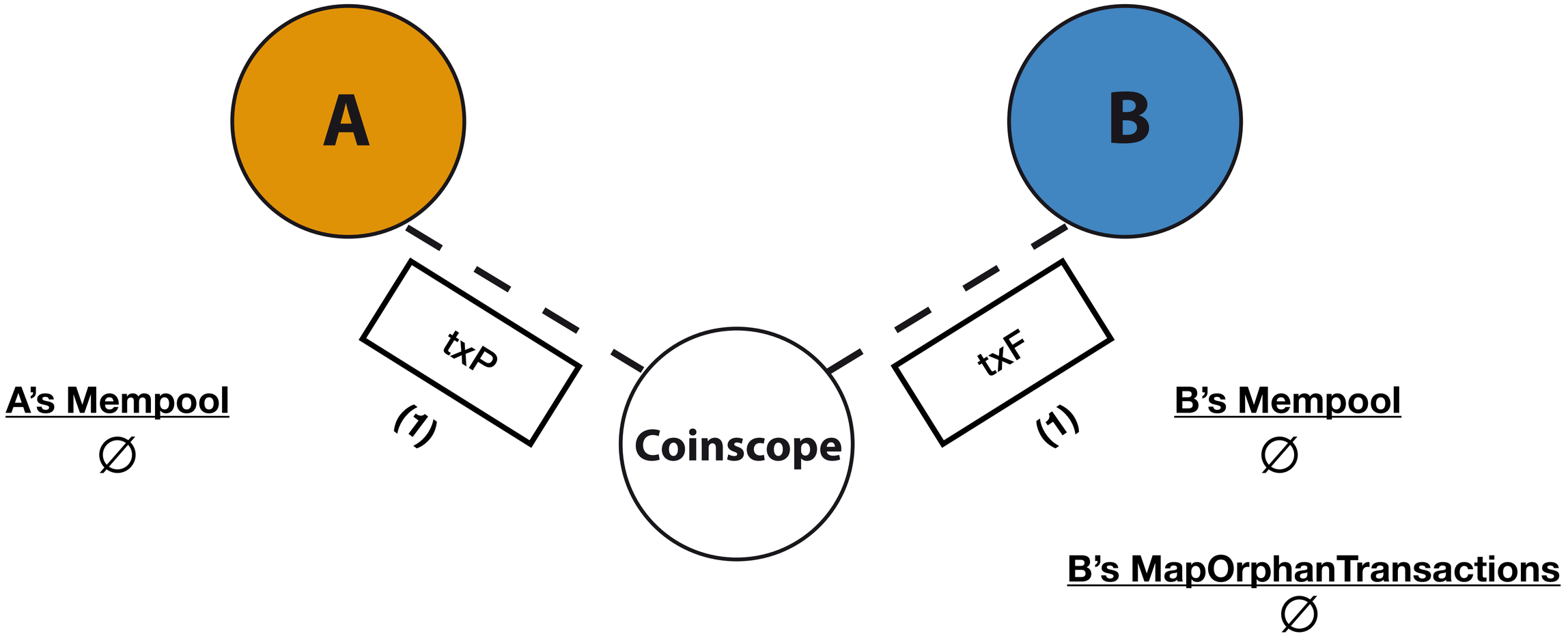
B's Mempool

∅

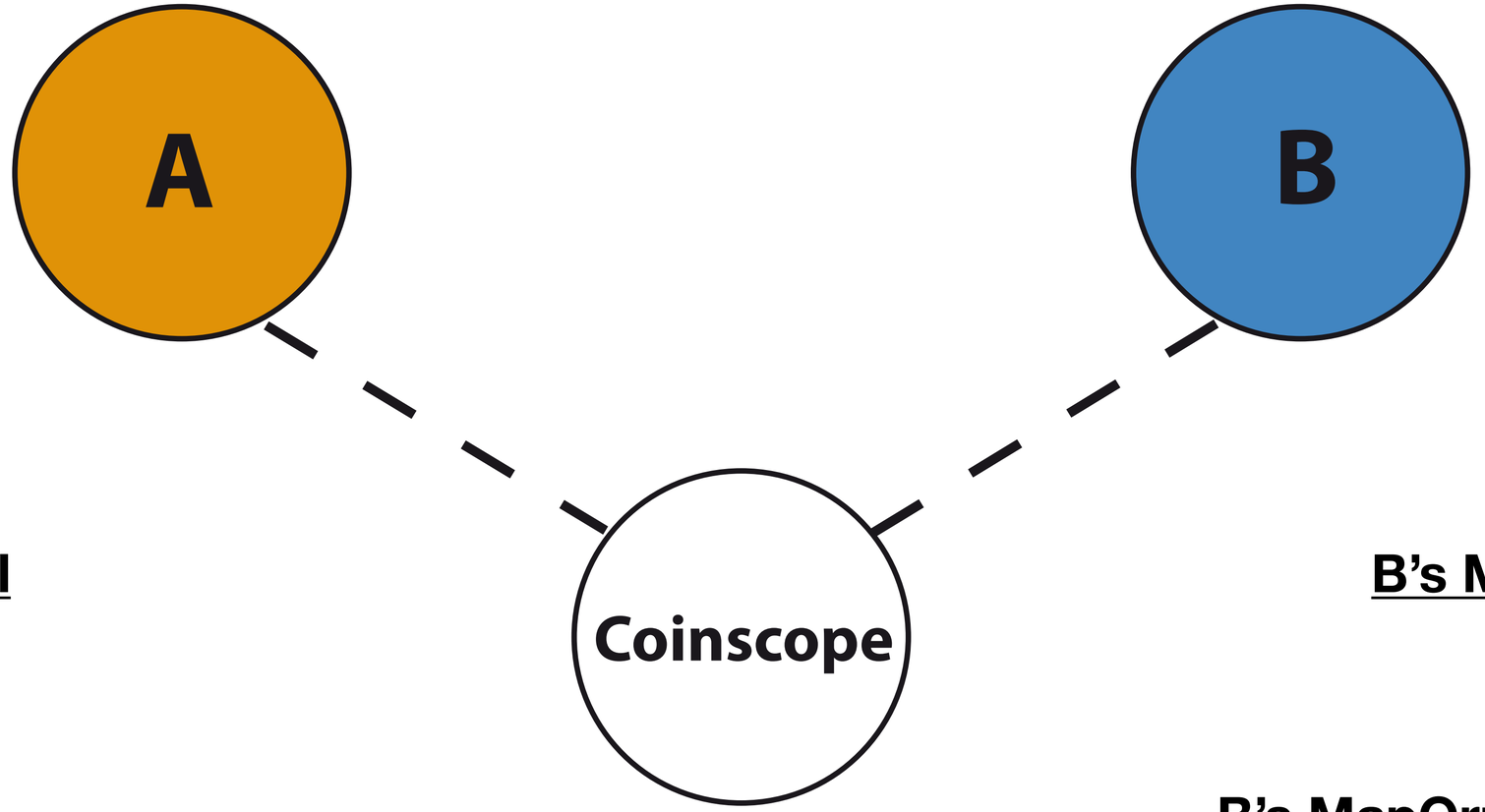
B's MapOrphanTransactions

∅

NEGATIVE INFERRING TECHNIQUE



NEGATIVE INFERRING TECHNIQUE



A's Mempool

∅

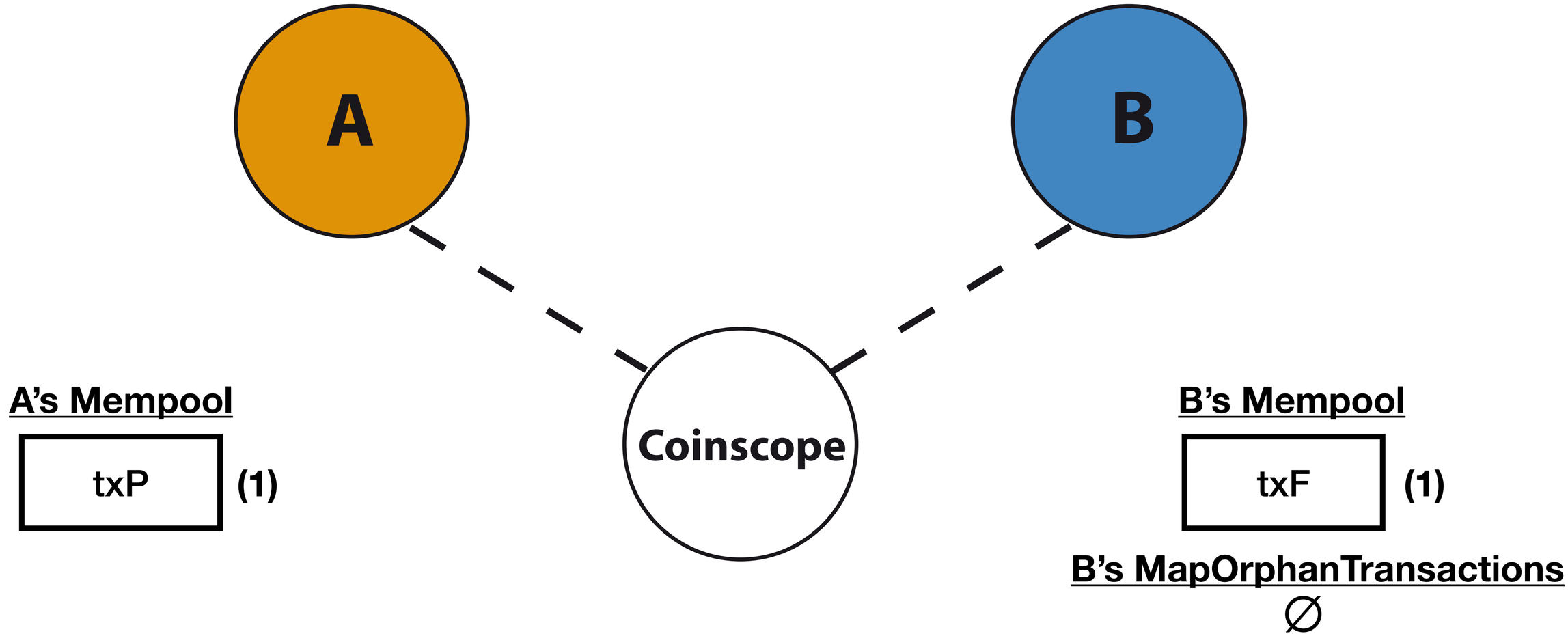
B's Mempool

∅

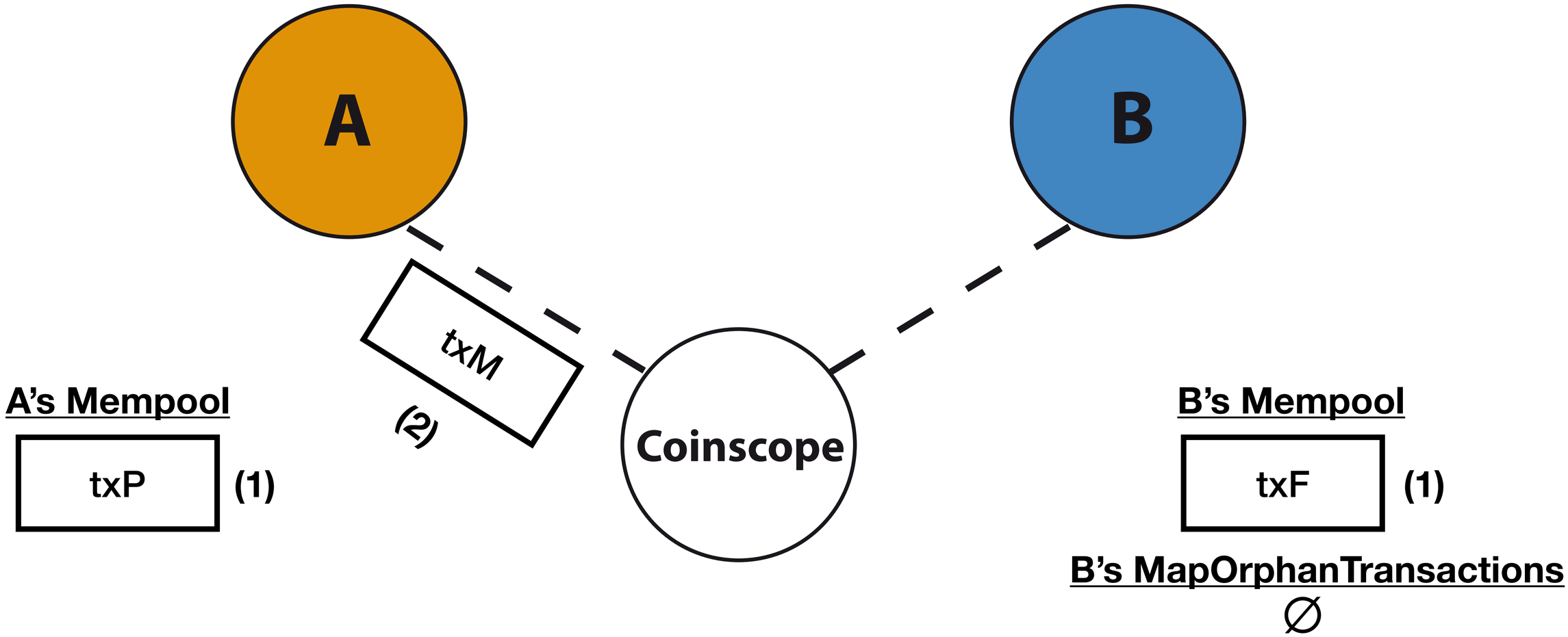
B's MapOrphanTransactions

∅

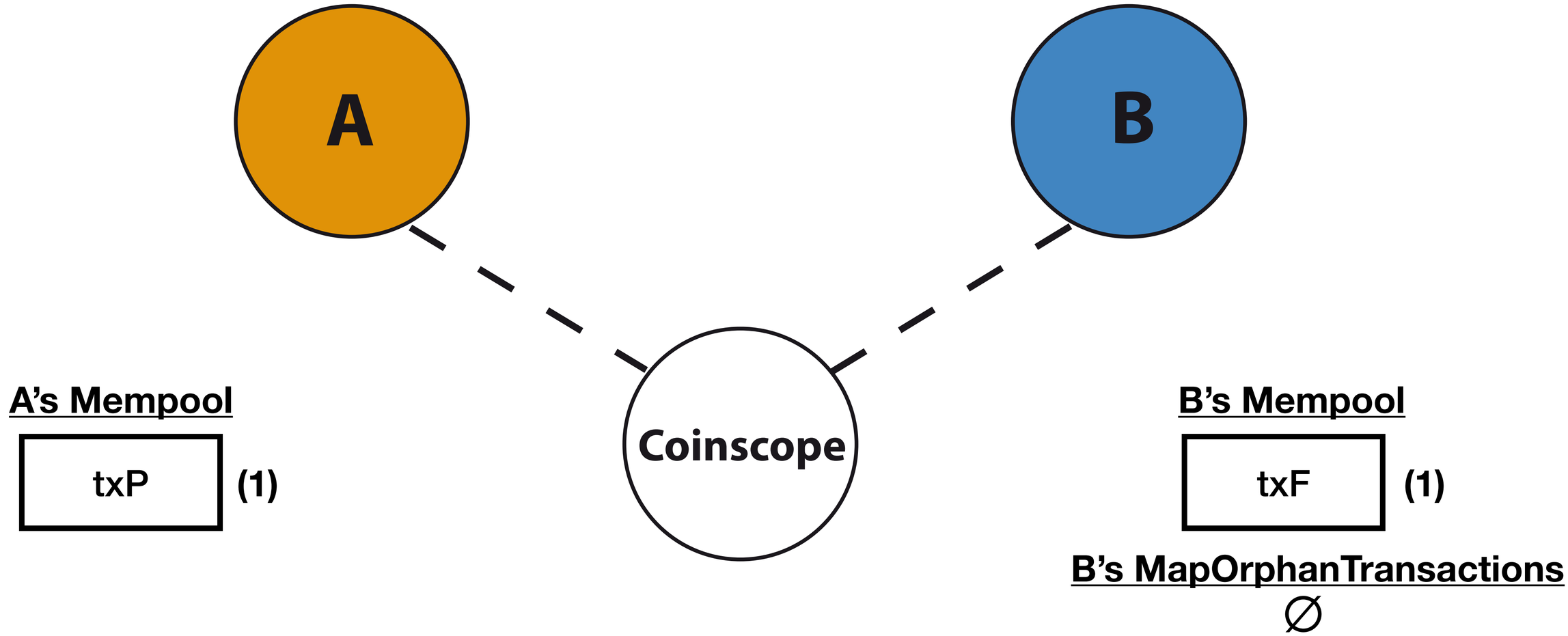
NEGATIVE INFERRING TECHNIQUE



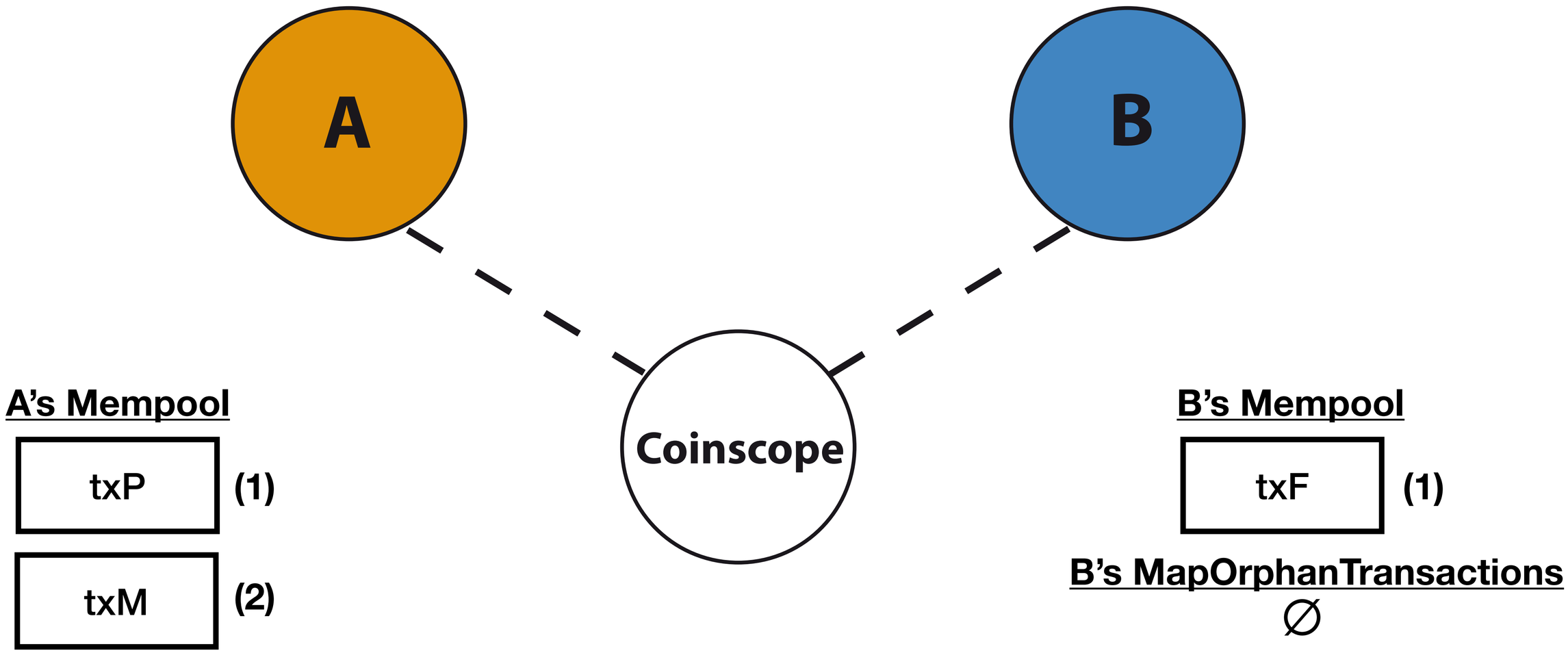
NEGATIVE INFERRING TECHNIQUE



NEGATIVE INFERRING TECHNIQUE



NEGATIVE INFERRING TECHNIQUE



NEGATIVE INFERRING TECHNIQUE

A's Mempool

txP	(1)
txM	(2)

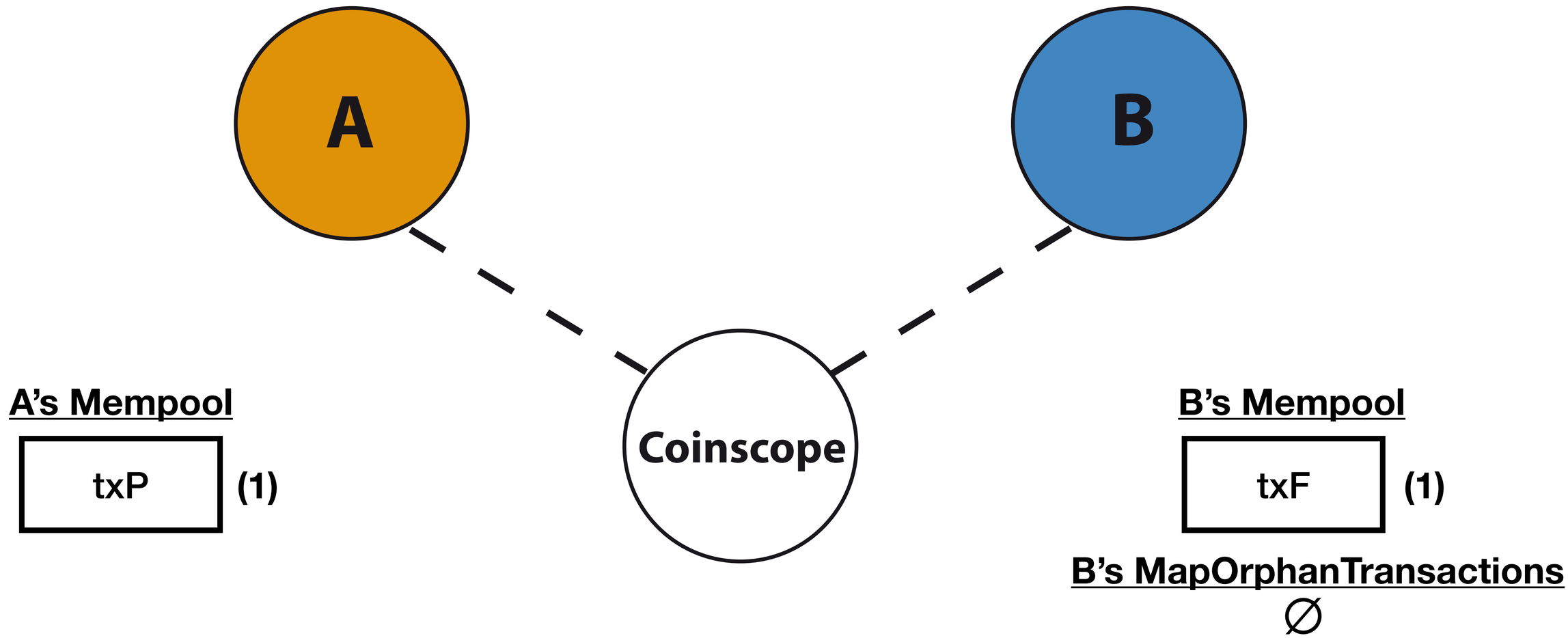
B's Mempool

txF	(1)

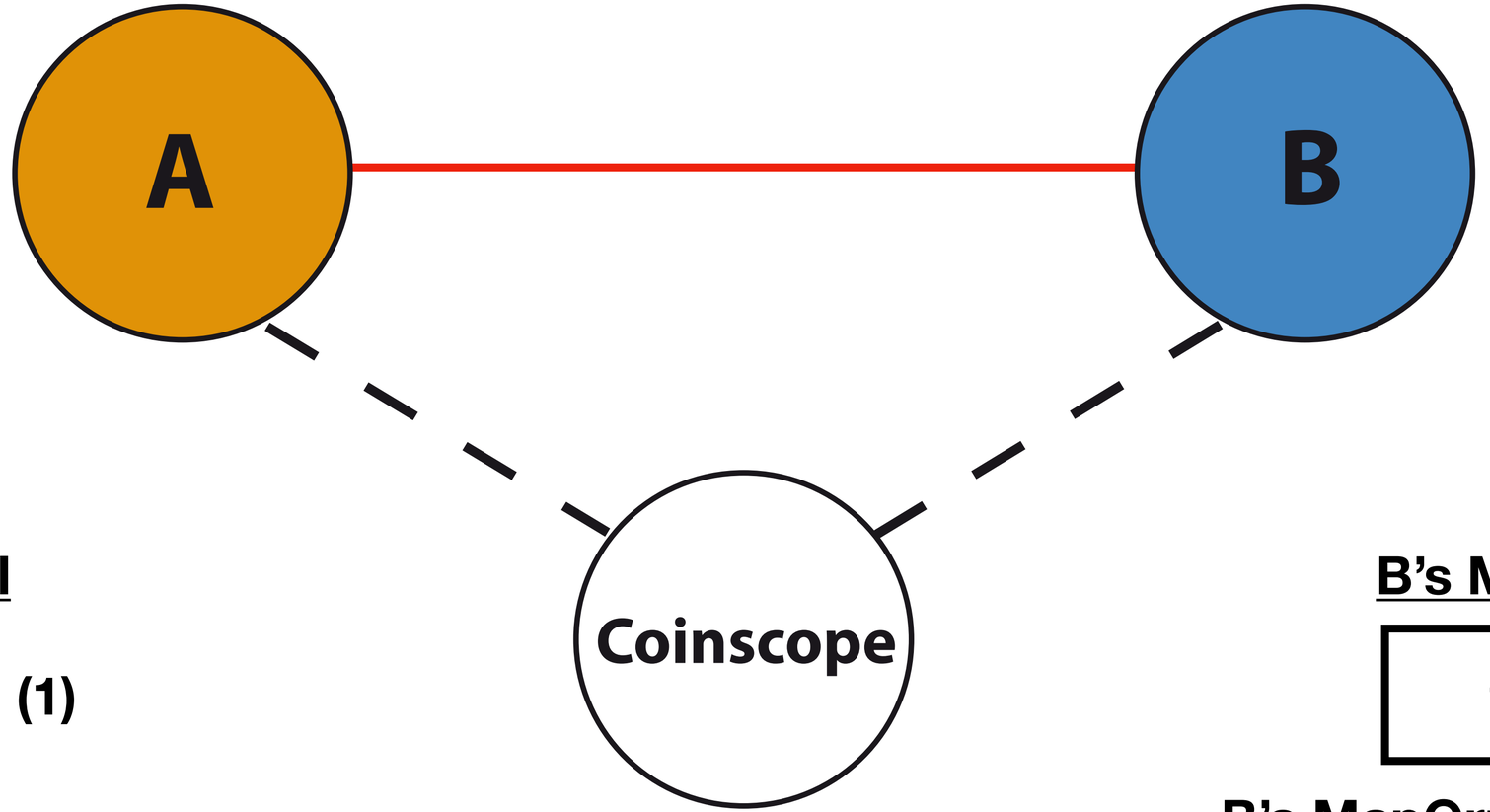
B's MapOrphanTransactions

∅

POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



A's Mempool

txP (1)

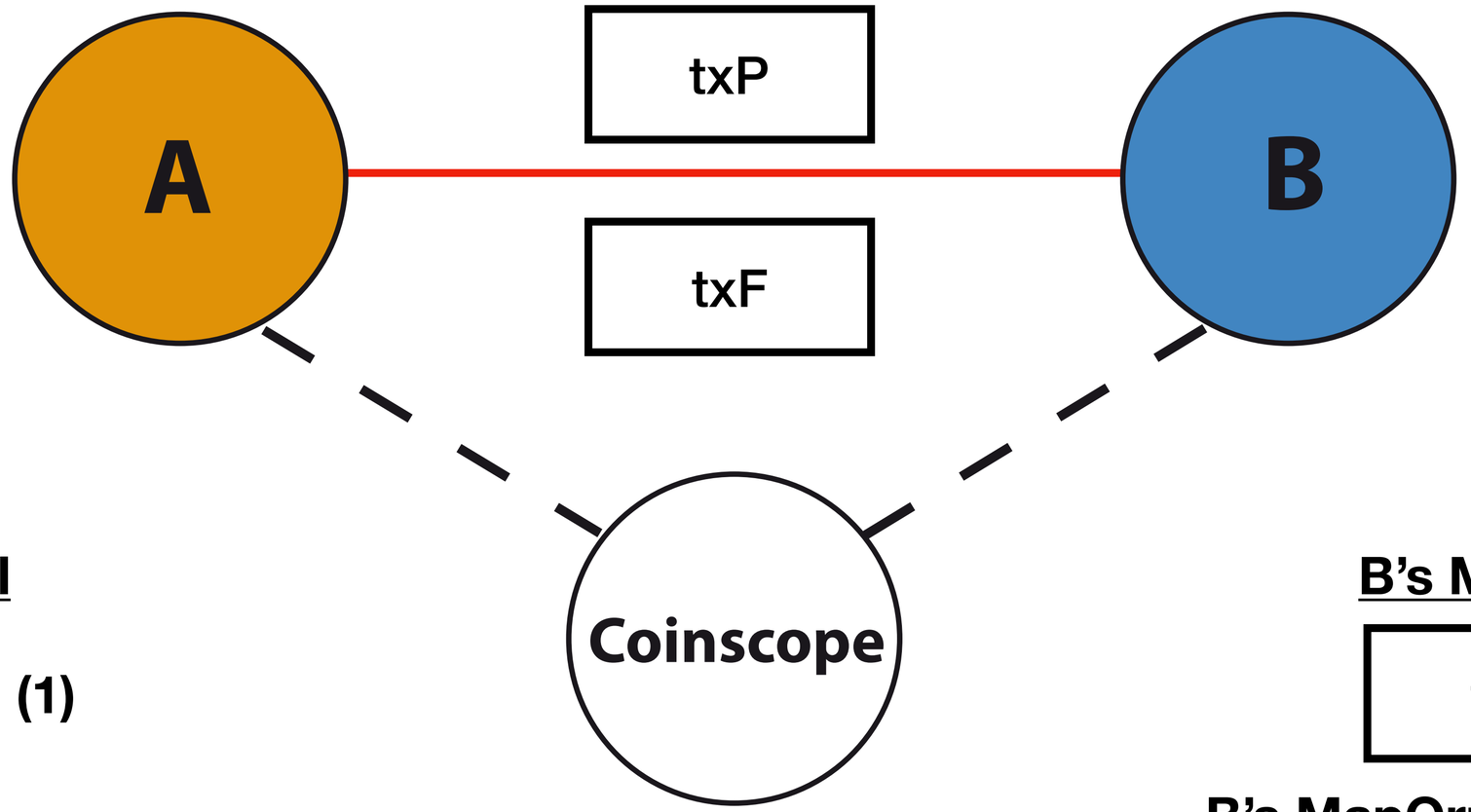
B's Mempool

txF (1)

B's MapOrphanTransactions

∅

POSITIVE INFERRING TECHNIQUE



A's Mempool

txP (1)

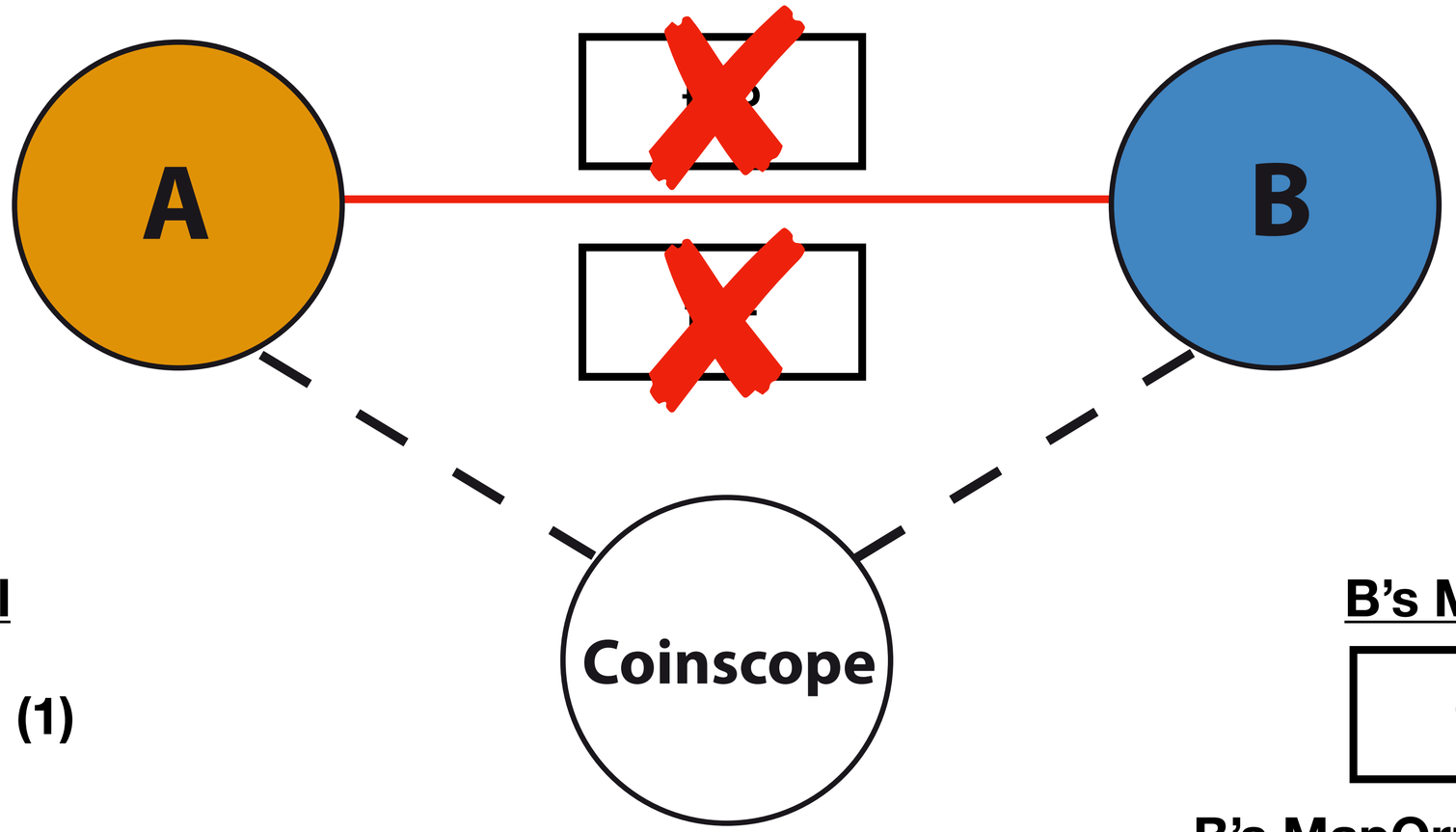
B's Mempool

txF (1)

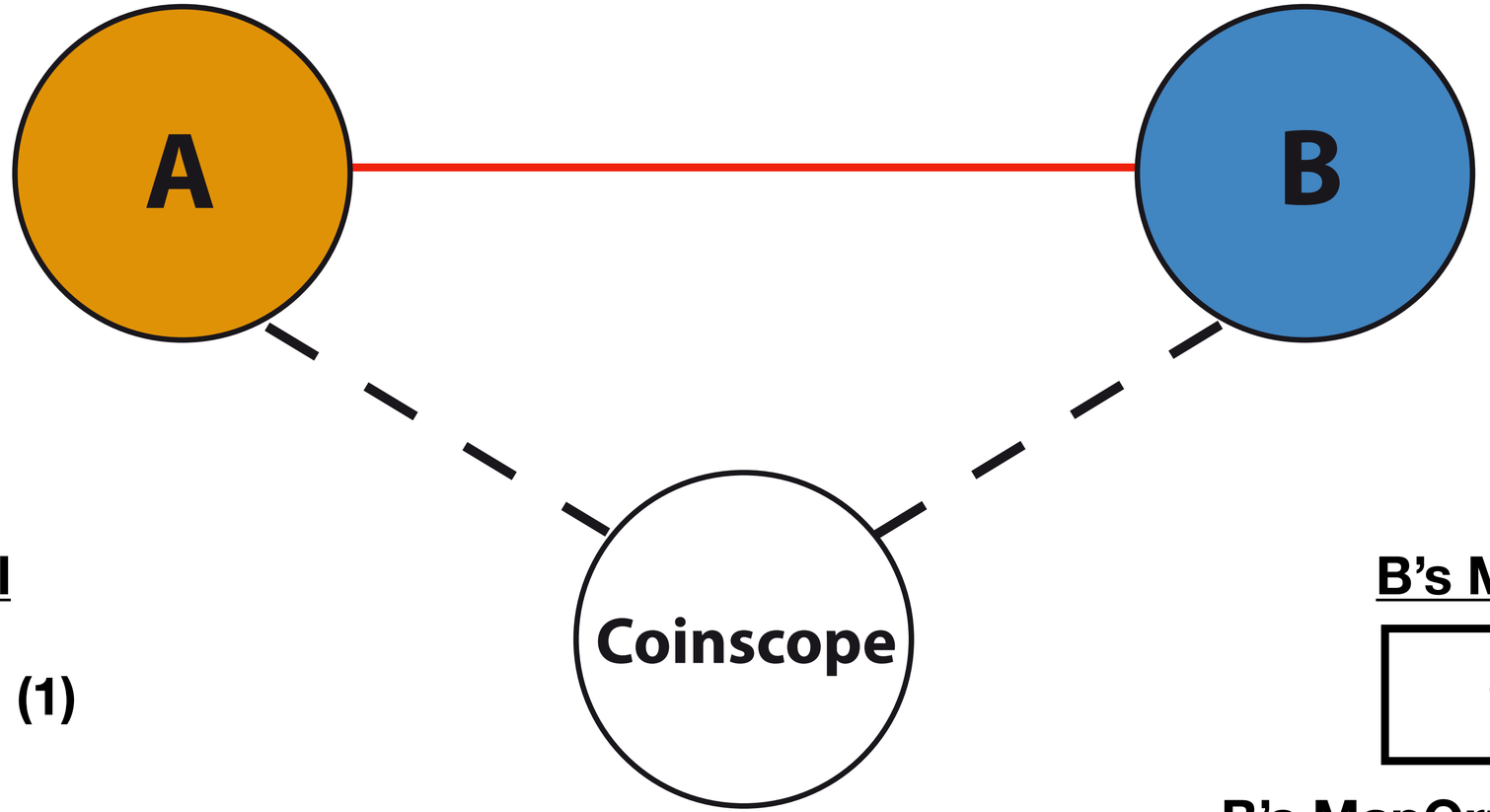
B's MapOrphanTransactions

∅

POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



A's Mempool

txP (1)

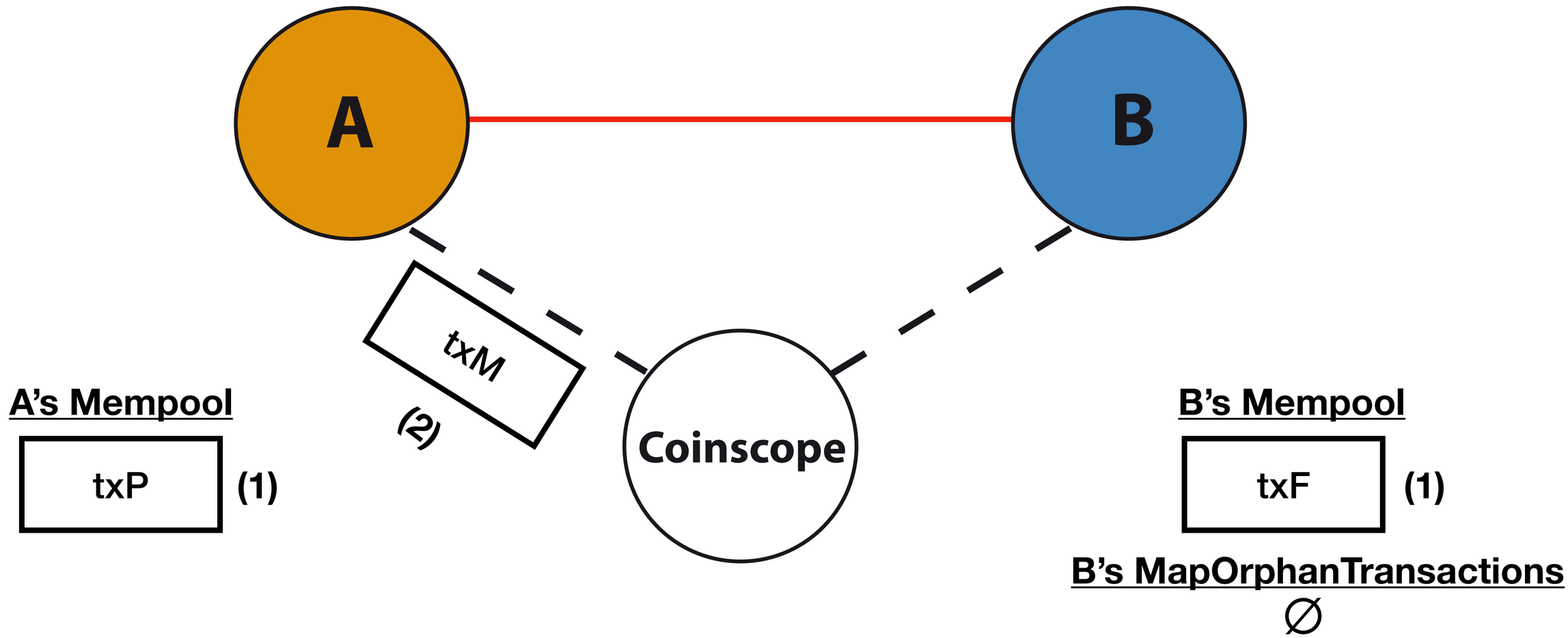
B's Mempool

txF (1)

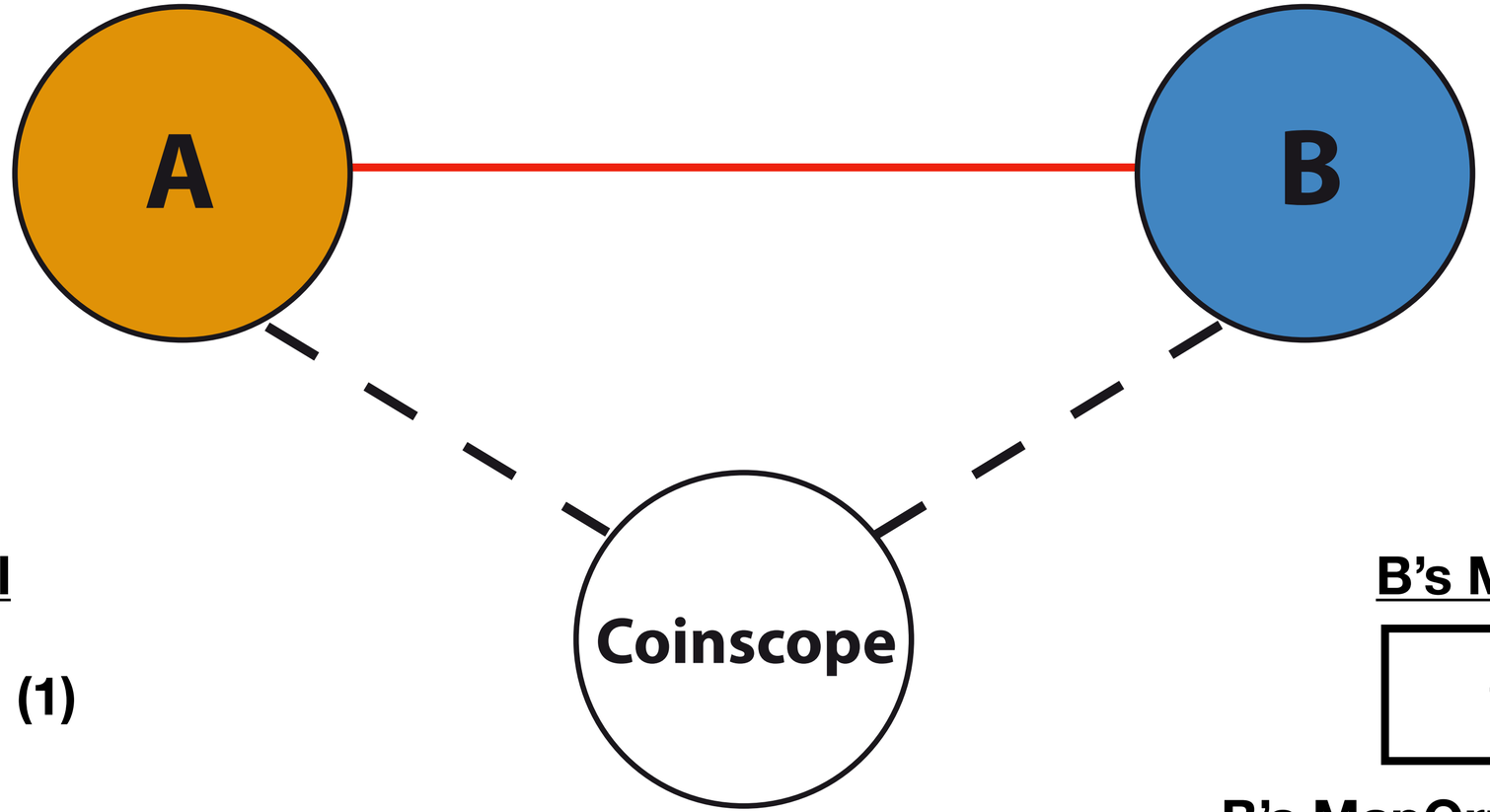
B's MapOrphanTransactions

∅

POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



A's Mempool

txP (1)

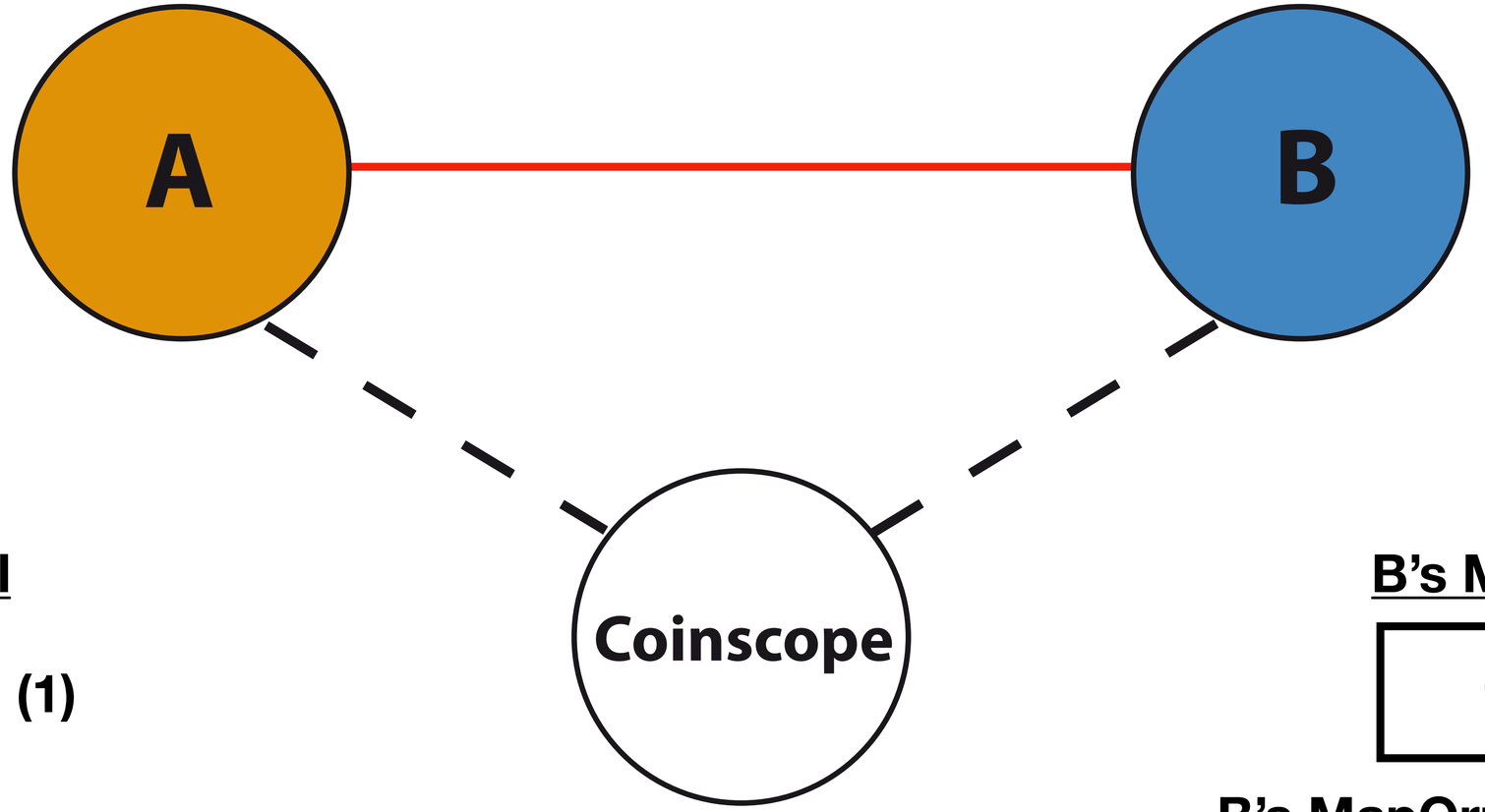
B's Mempool

txF (1)

B's MapOrphanTransactions

∅

POSITIVE INFERRING TECHNIQUE



A's Mempool

txP (1)

txM (2)

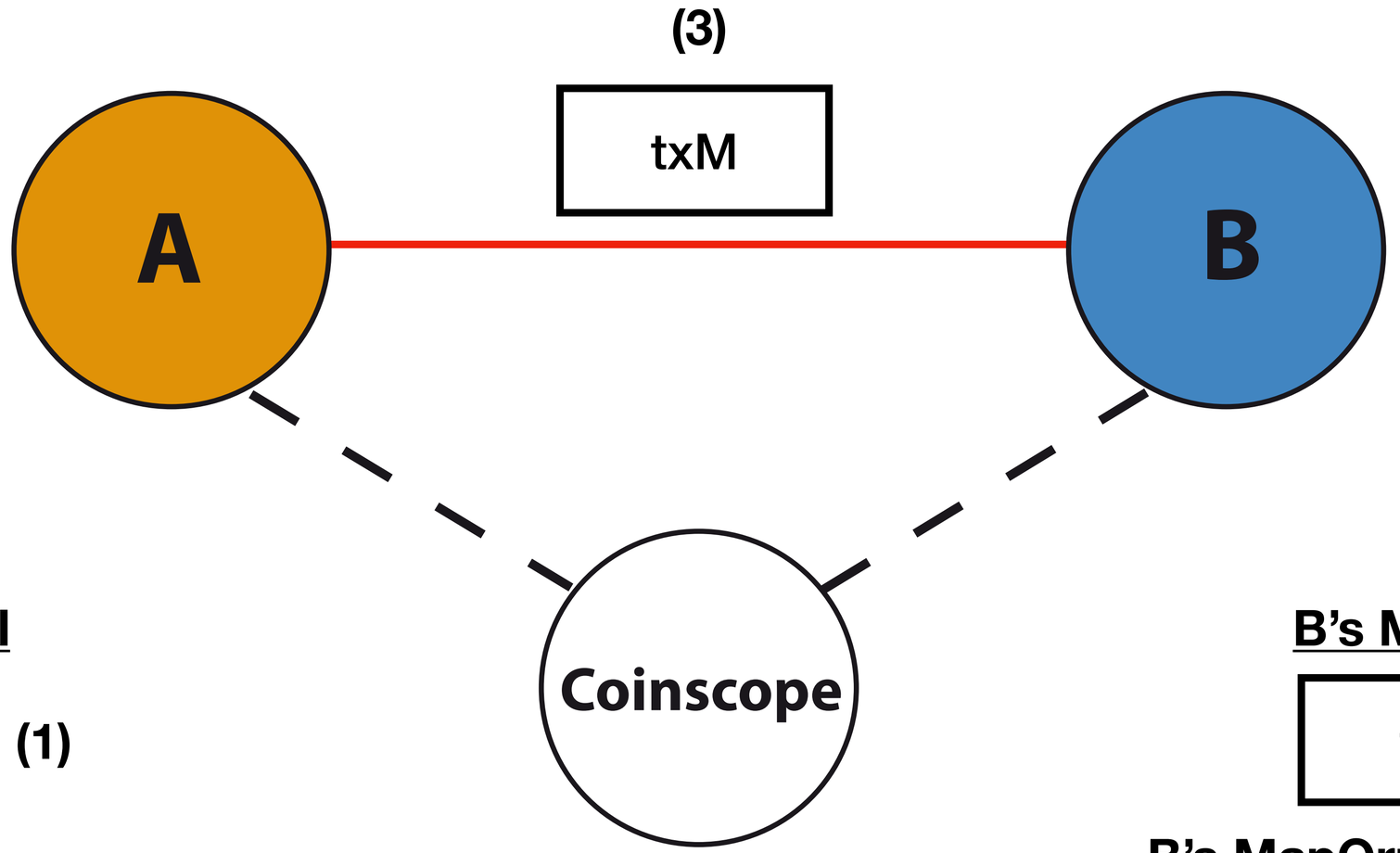
B's Mempool

txF (1)

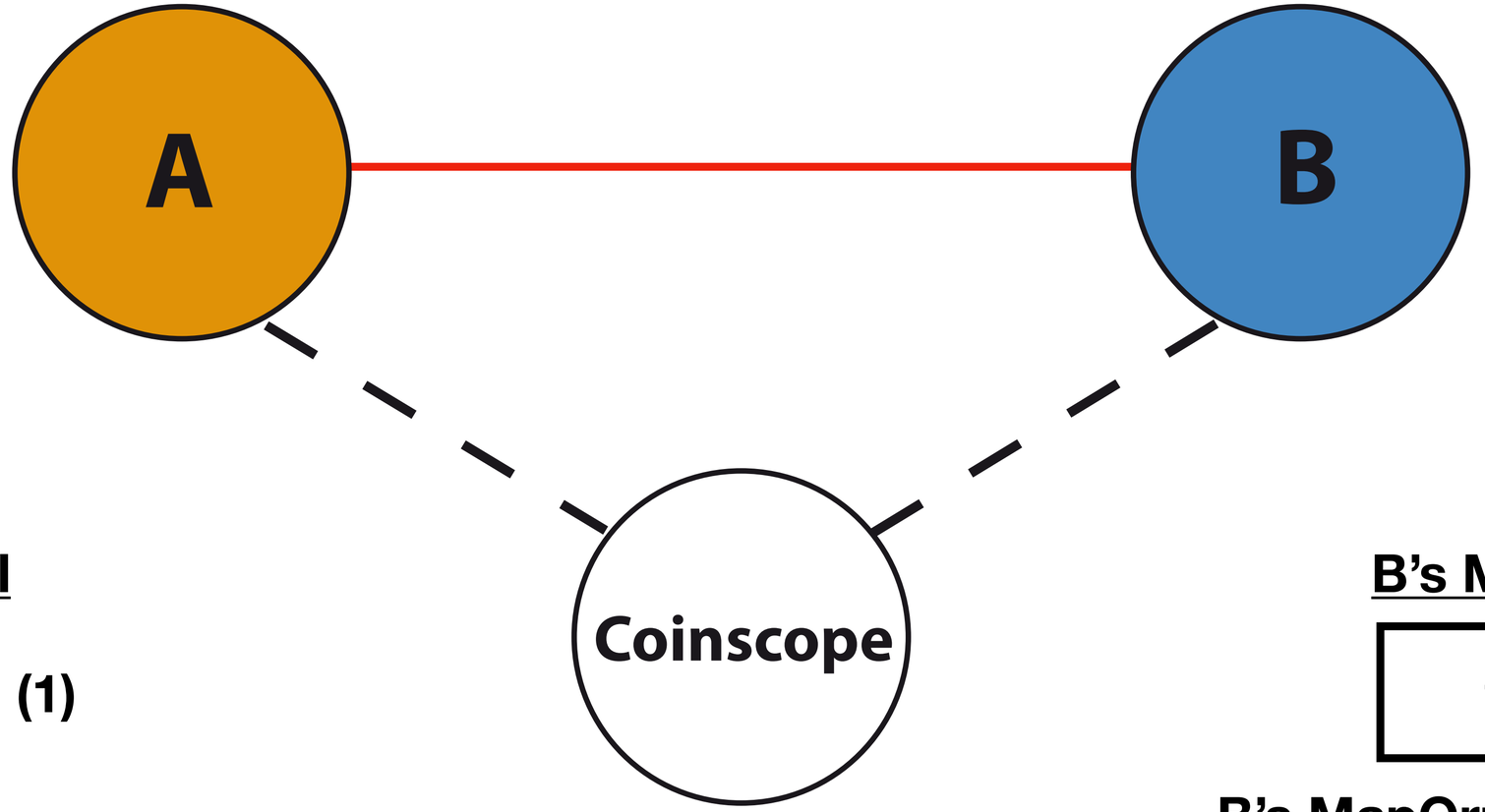
B's MapOrphanTransactions

∅

POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



A's Mempool

txP (1)

txM (2)

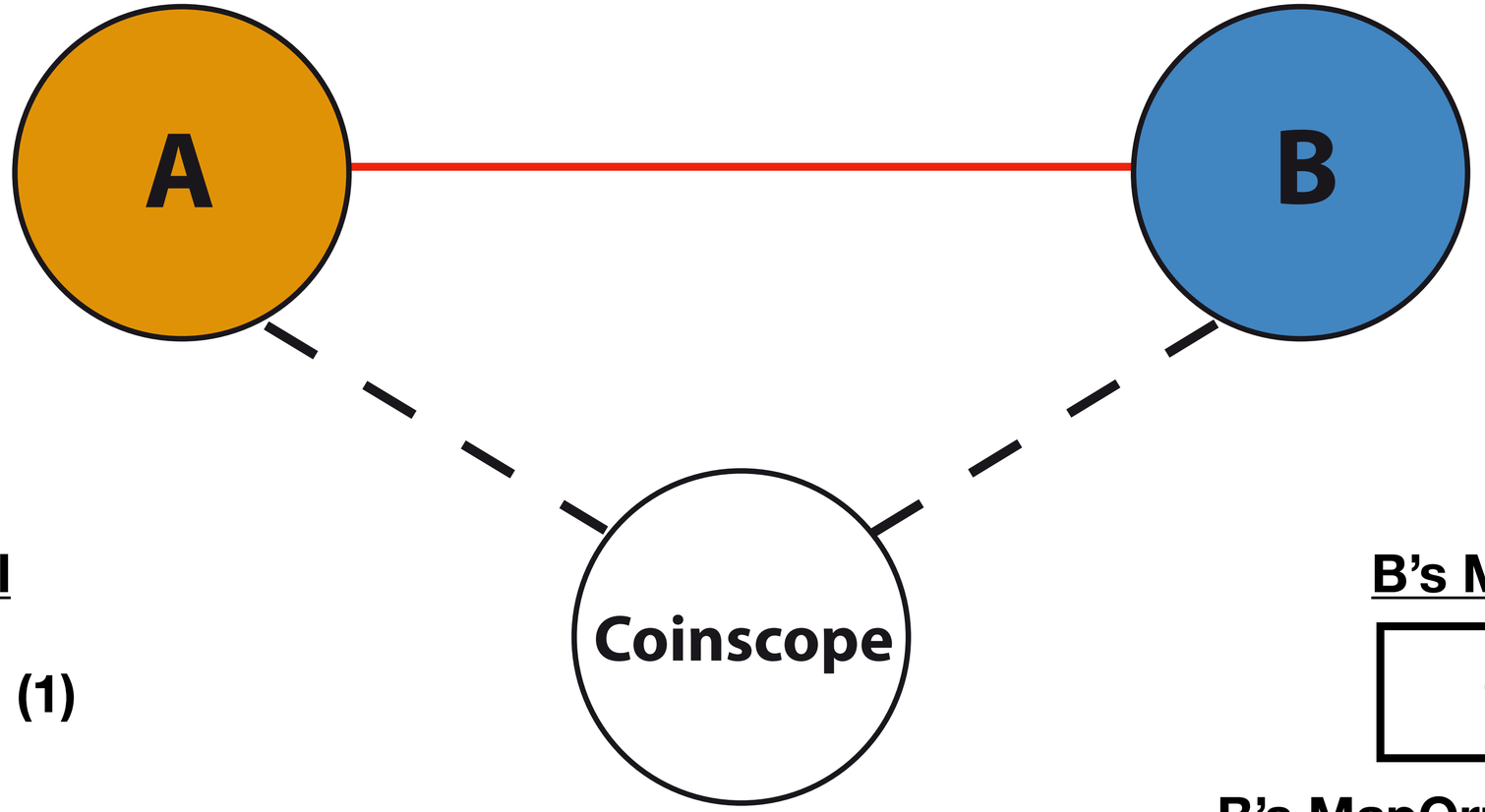
B's Mempool

txF (1)

B's MapOrphanTransactions

∅

POSITIVE INFERRING TECHNIQUE



A's Mempool

txP	(1)
txM	(2)

B's Mempool

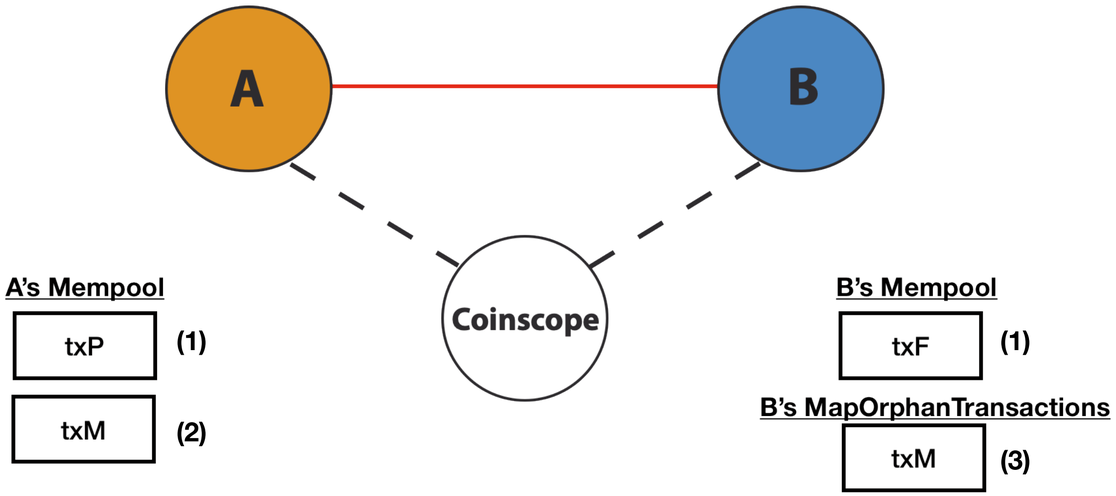
txF	(1)

B's MapOrphanTransactions

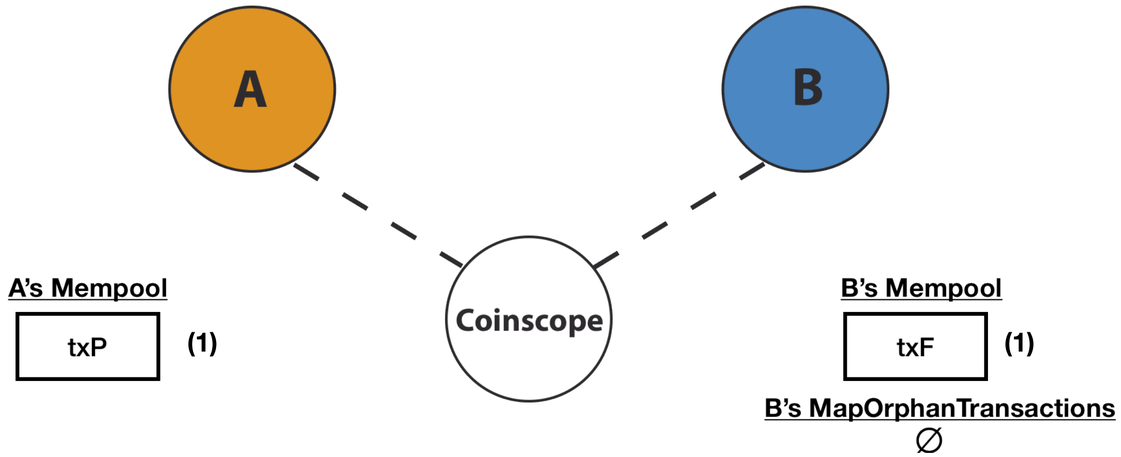
txM	(3)

A BASIC TOPOLOGY INFERRING TECHNIQUE

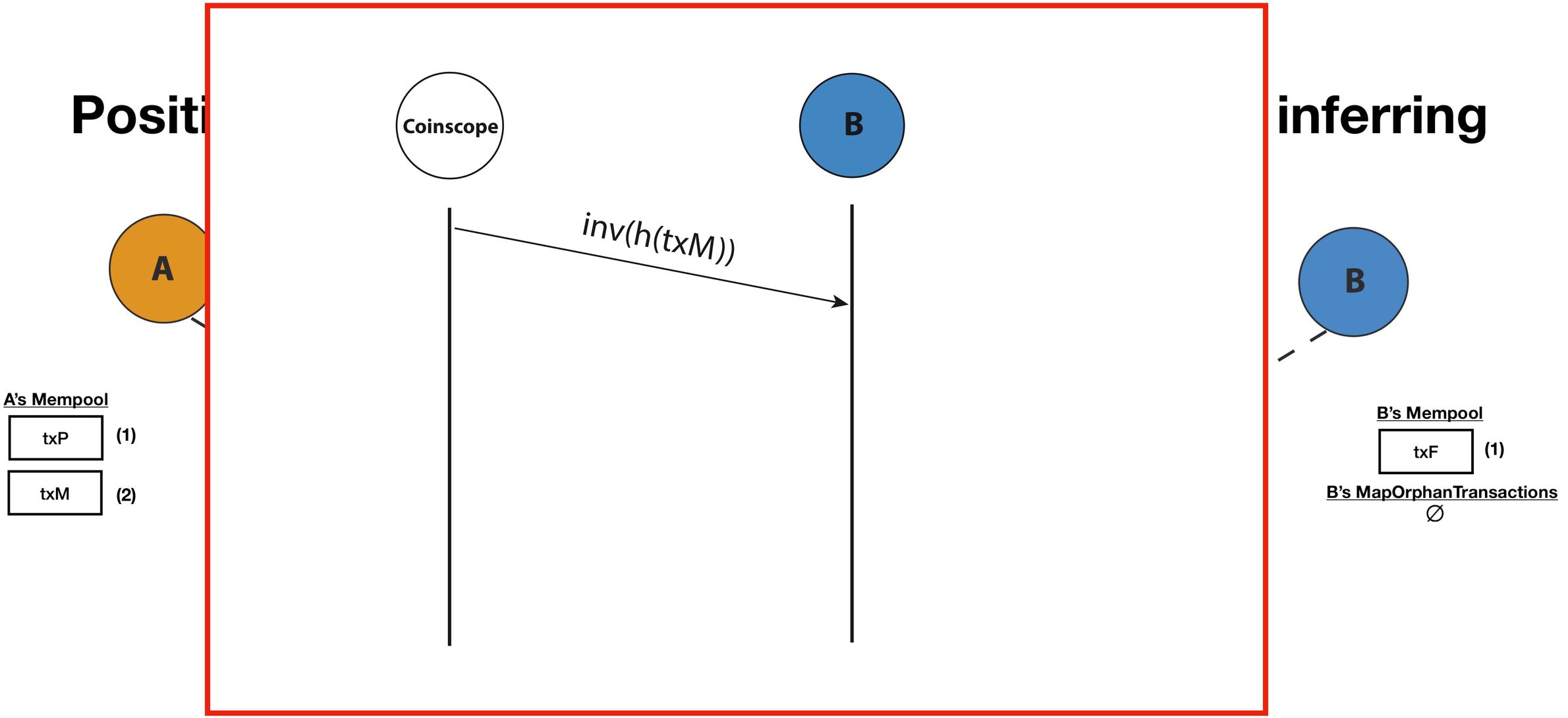
Positive edge inferring



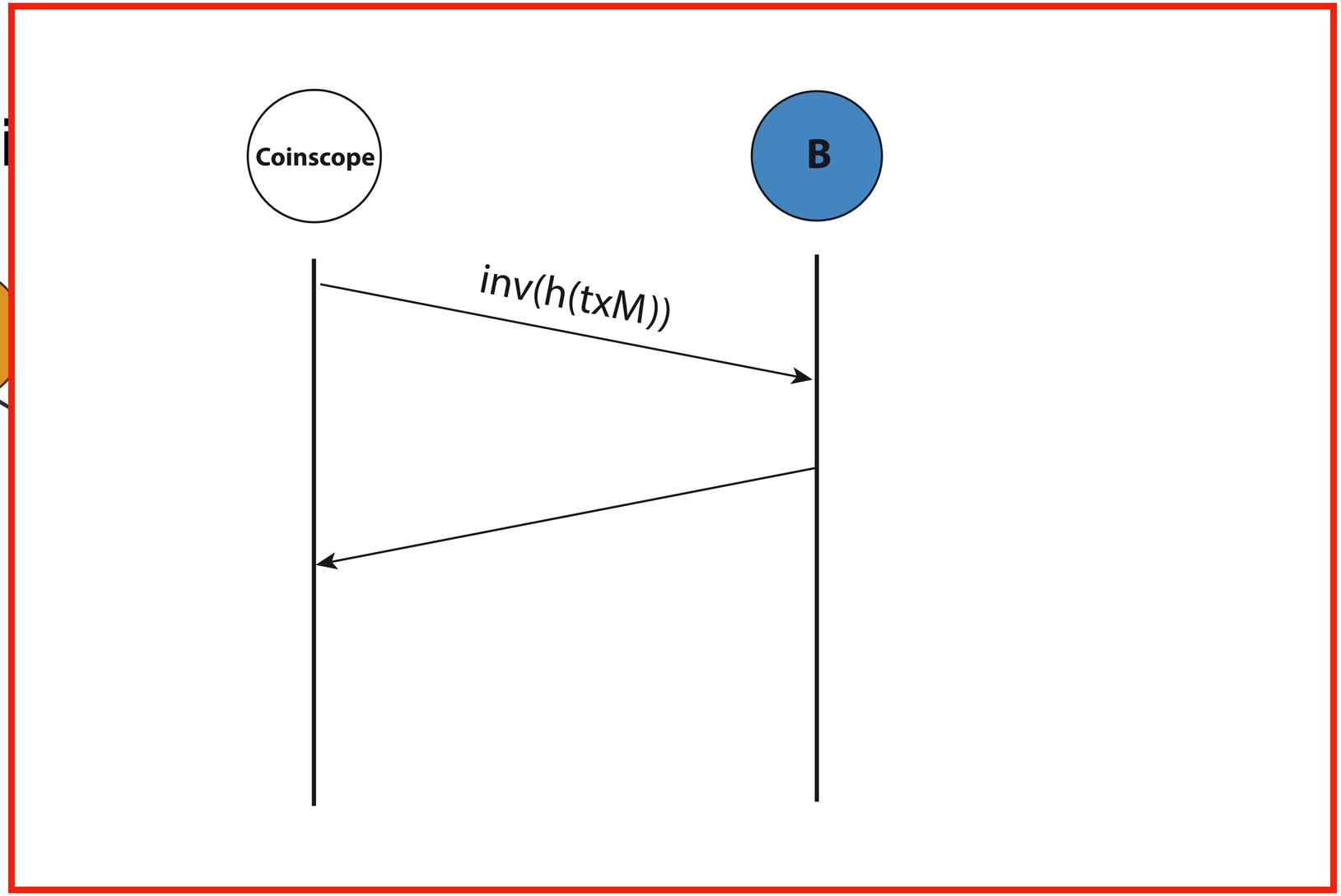
Negative edge inferring



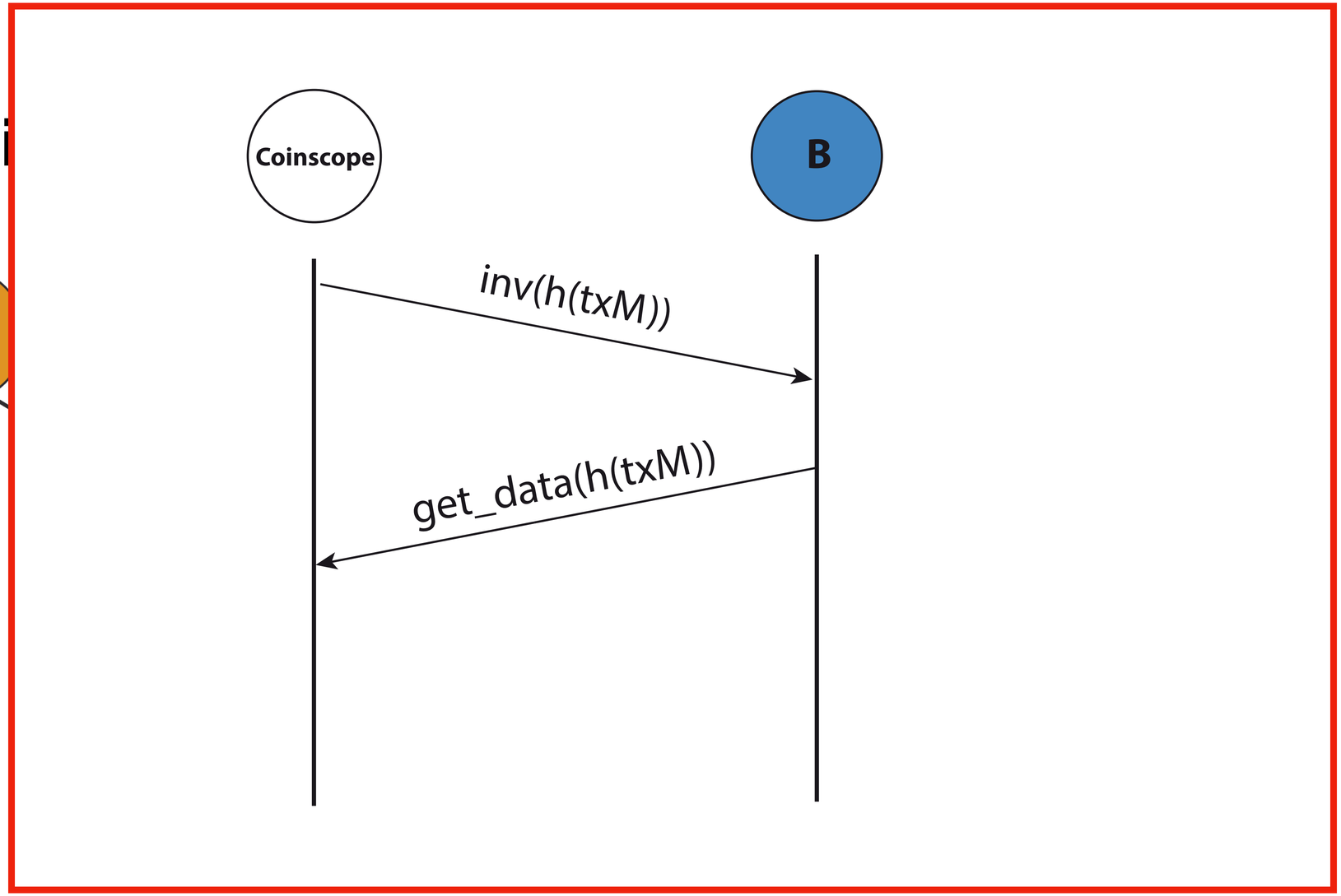
A BASIC TOPOLOGY INFERRING TECHNIQUE



A BASIC TOPOLOGY INFERRING TECHNIQUE



A BASIC TOPOLOGY INFERRING TECHNIQUE



Positi

inferring

A

A's Mempool

txP (1)

txM (2)

B

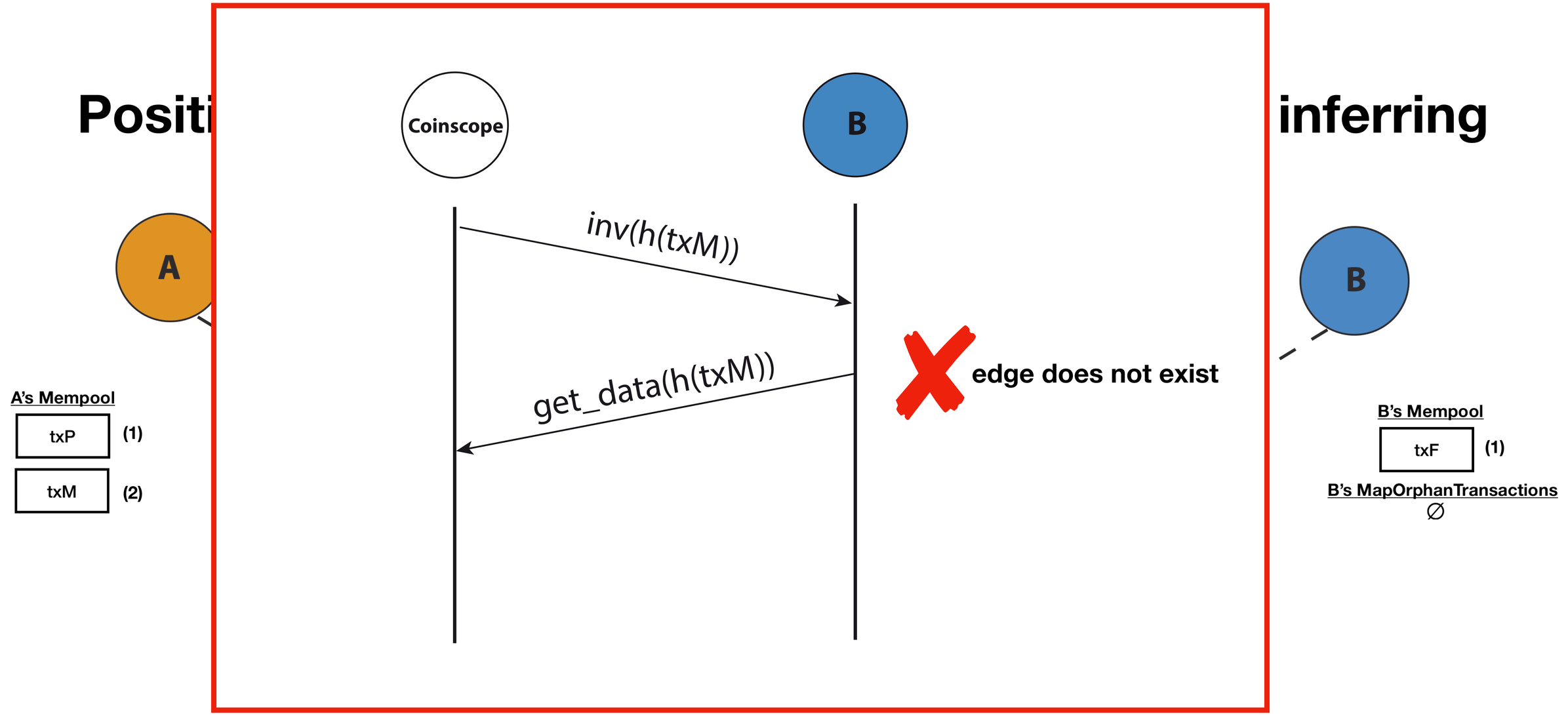
B's Mempool

txF (1)

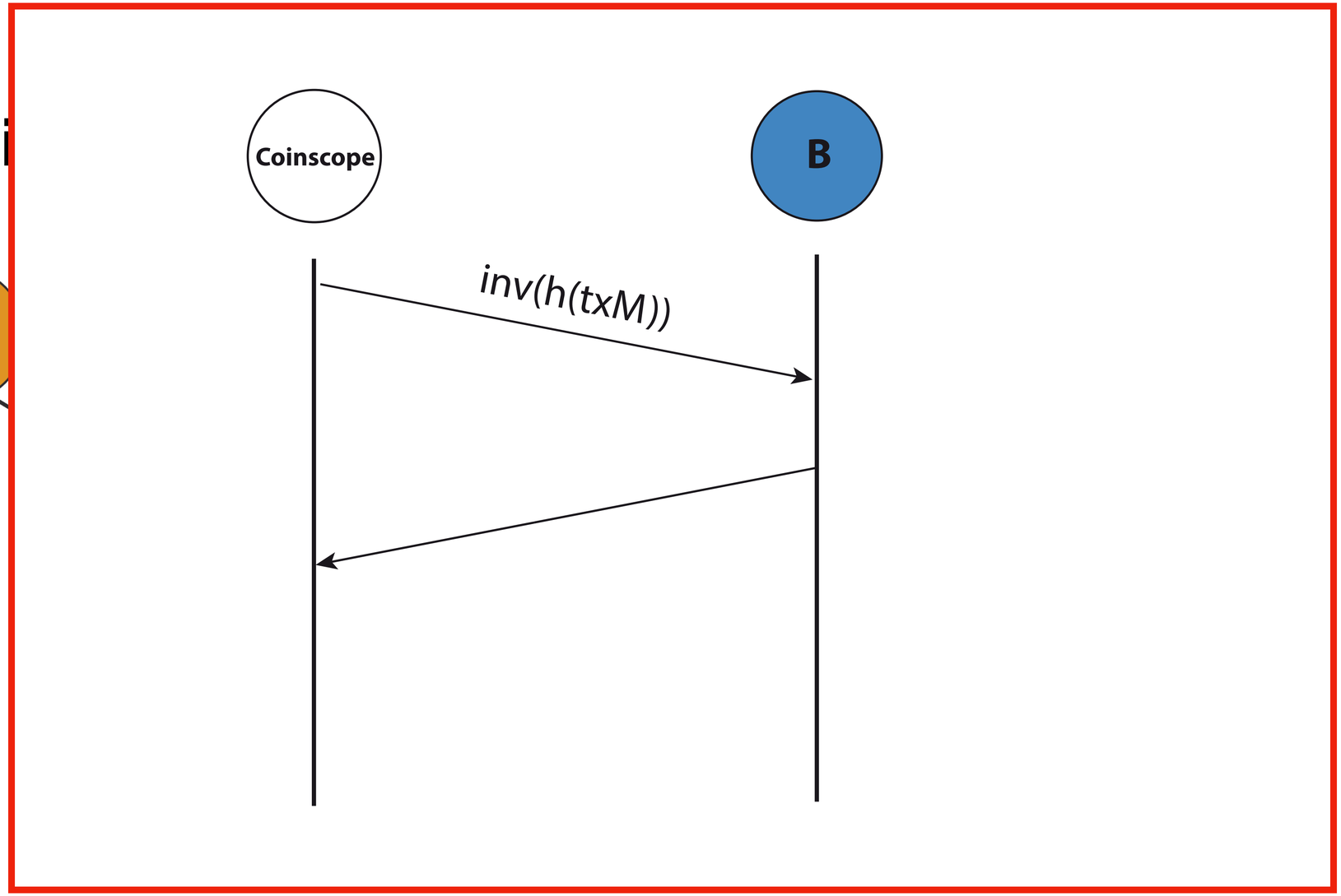
B's MapOrphanTransactions

∅

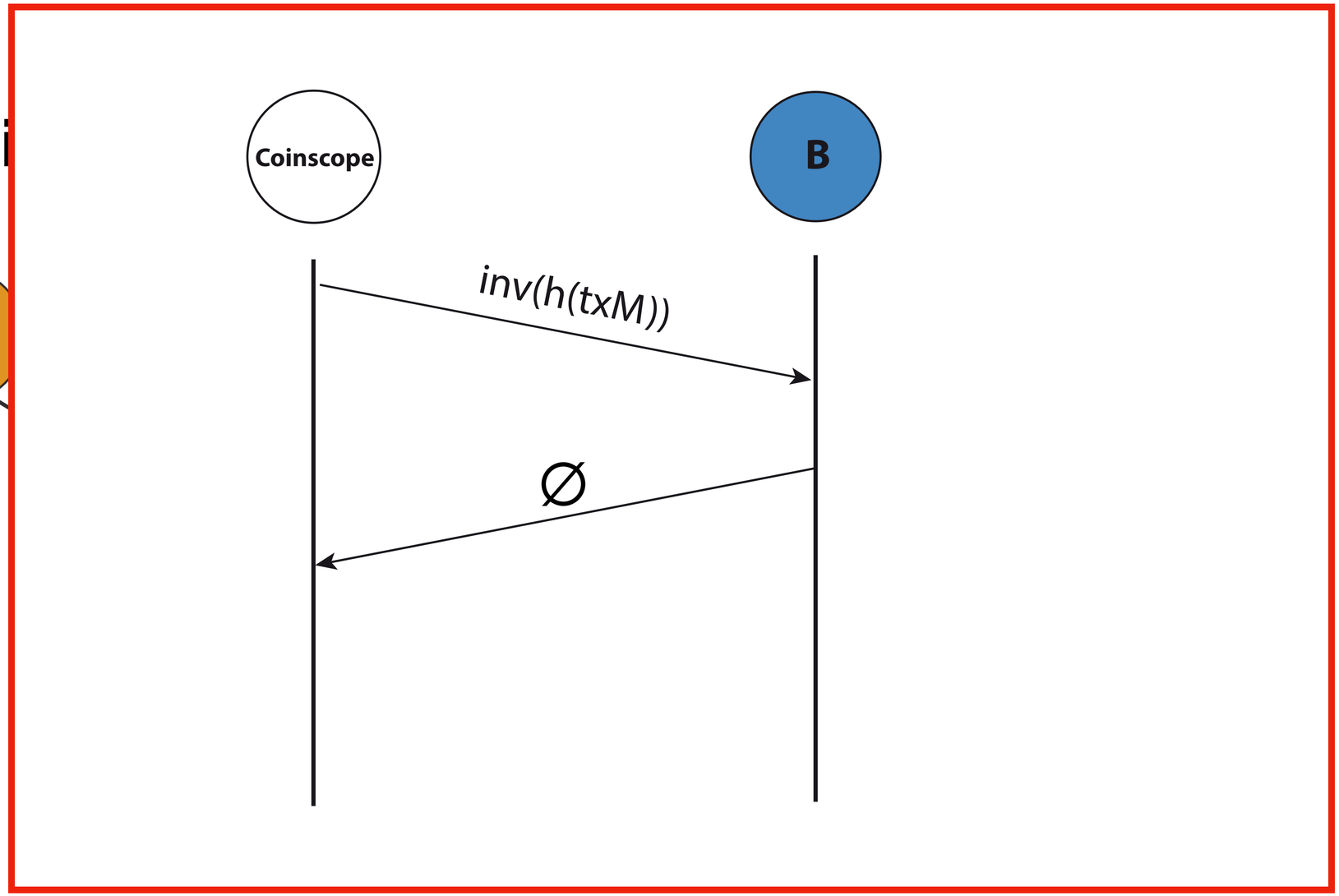
A BASIC TOPOLOGY INFERRING TECHNIQUE



A BASIC TOPOLOGY INFERRING TECHNIQUE



A BASIC TOPOLOGY INFERRING TECHNIQUE



Positi

inferring

A

A's Mempool

txP (1)

txM (2)

B

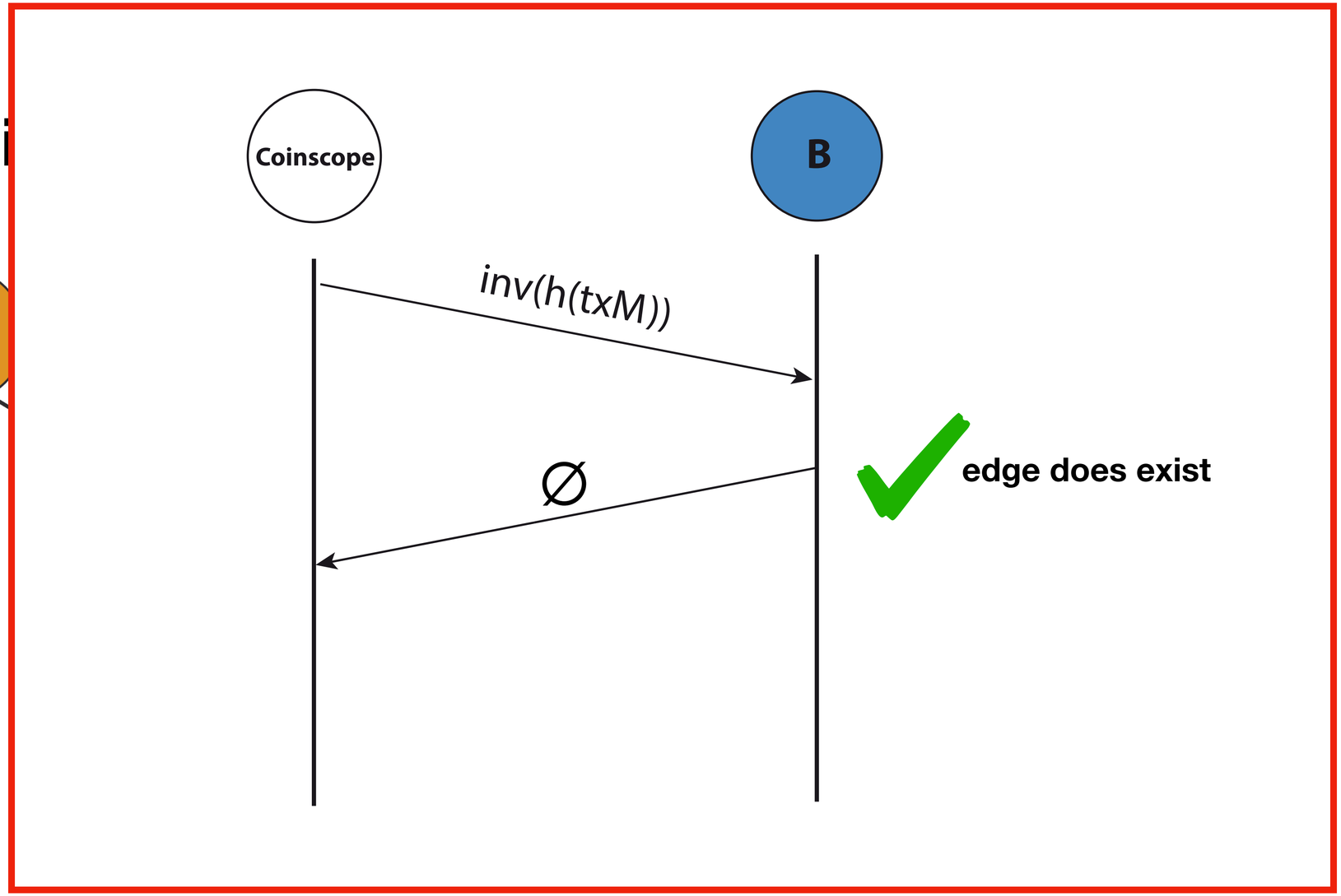
B's Mempool

txF (1)

B's MapOrphanTransactions

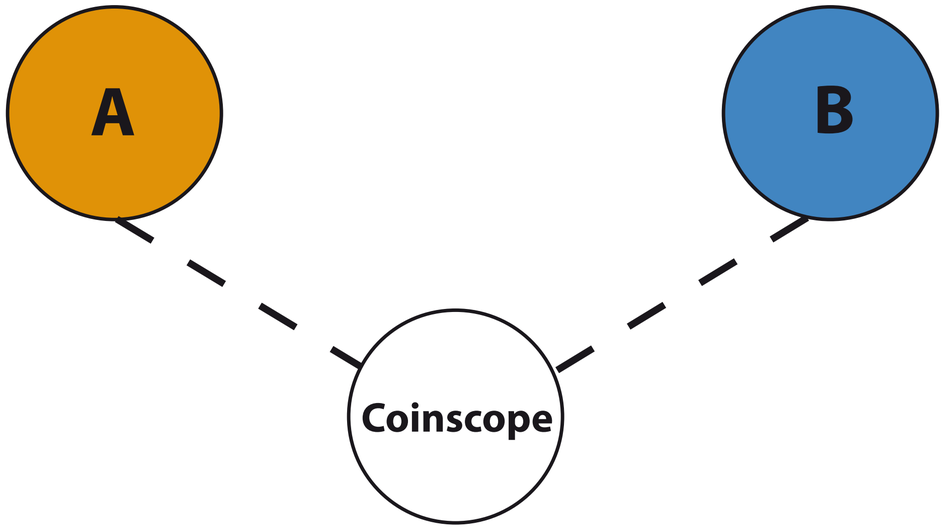
∅

A BASIC TOPOLOGY INFERRING TECHNIQUE



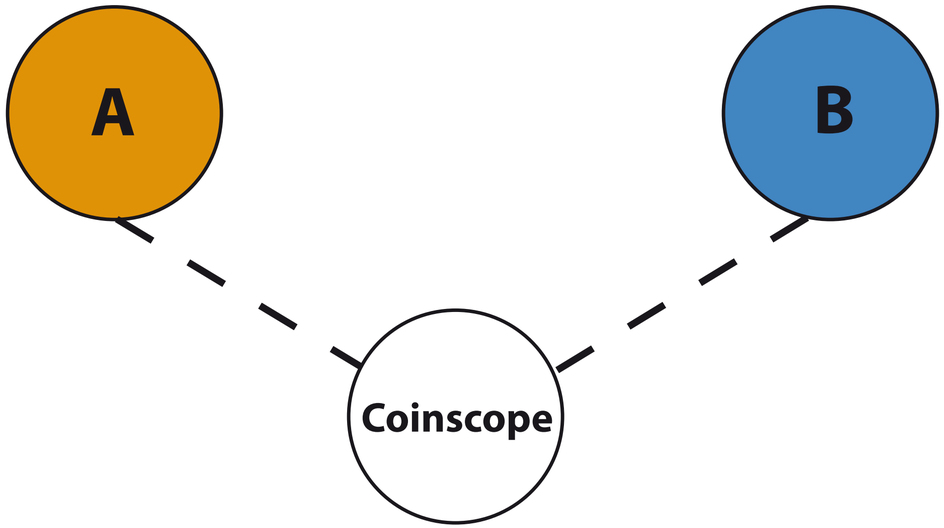
IT IS NOT THAT EASY

ITS NOT THAT EASY



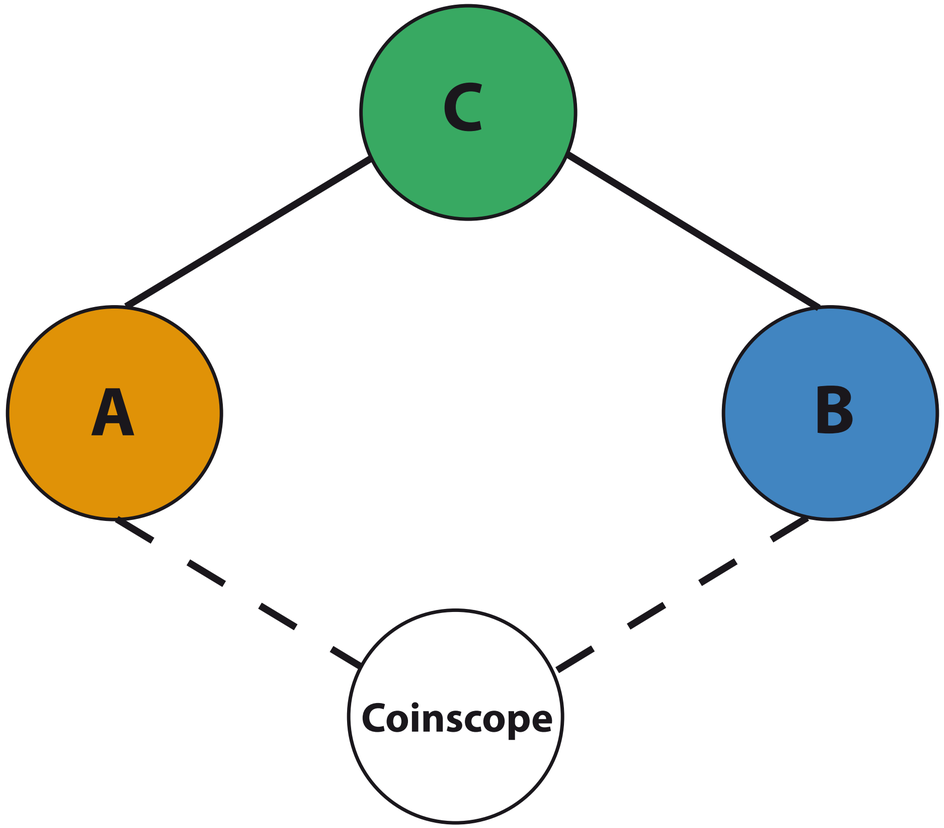
ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



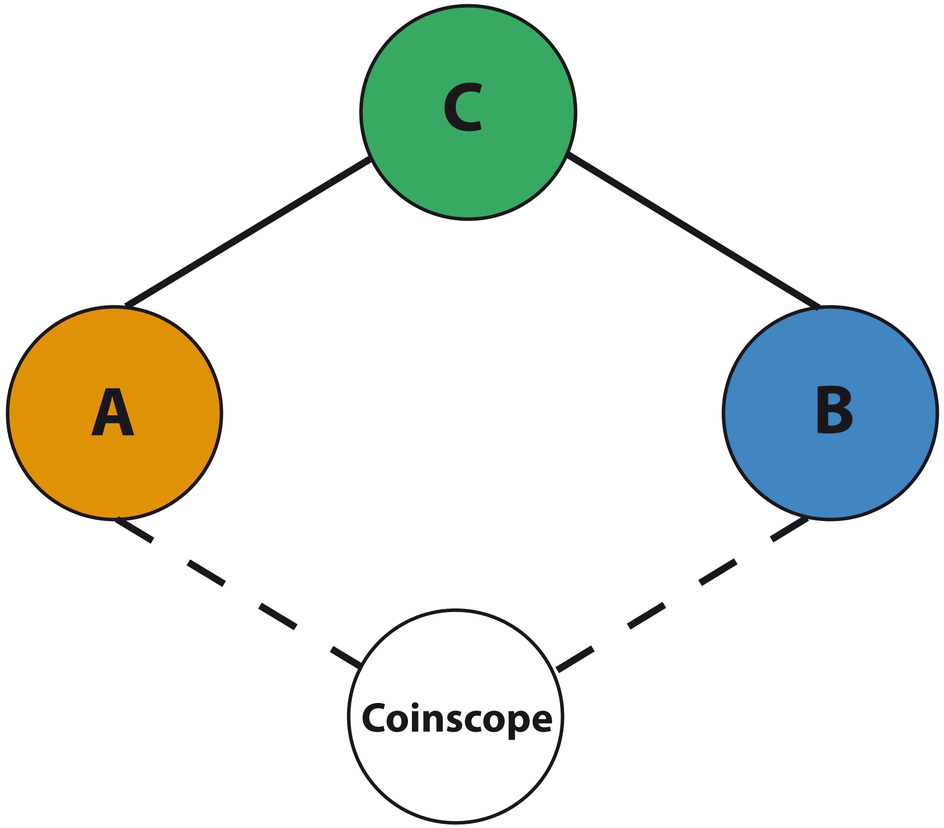
ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

∅

C's Mempool

∅

B's Mempool

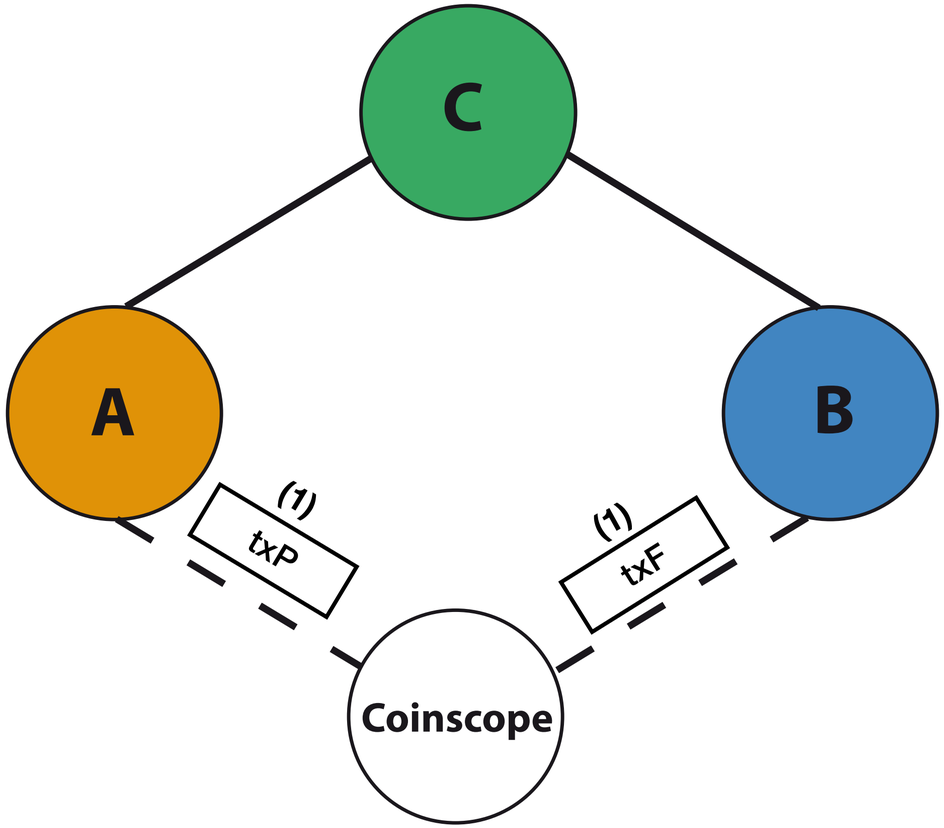
∅

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

∅

C's Mempool

∅

B's Mempool

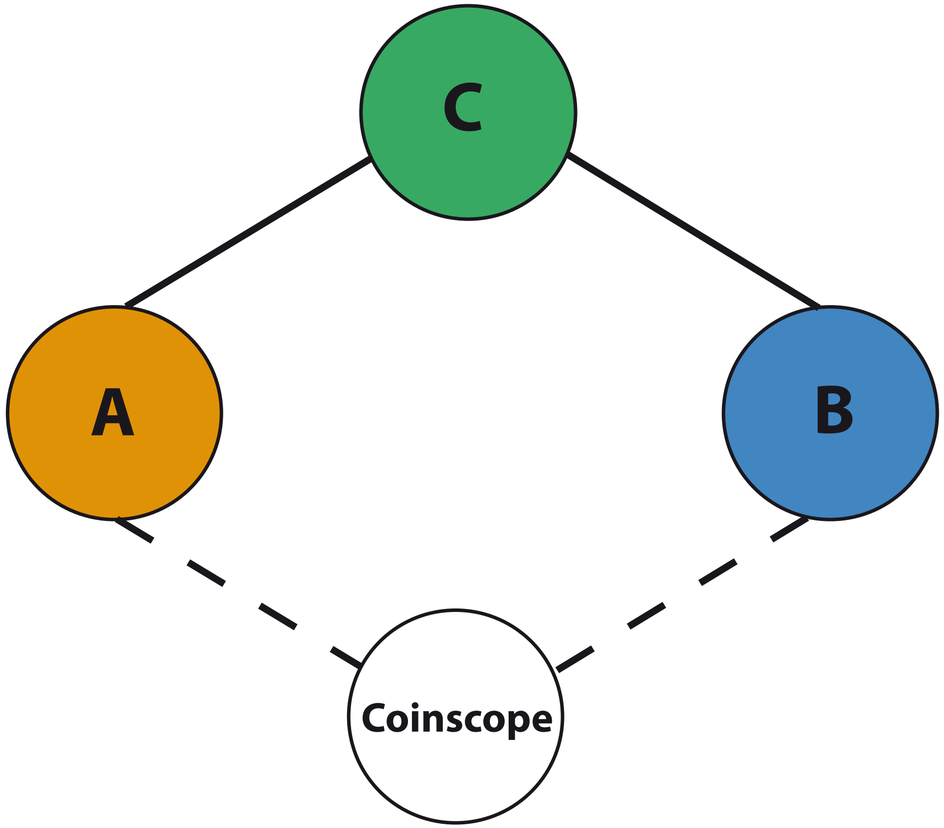
∅

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

∅

C's Mempool

∅

B's Mempool

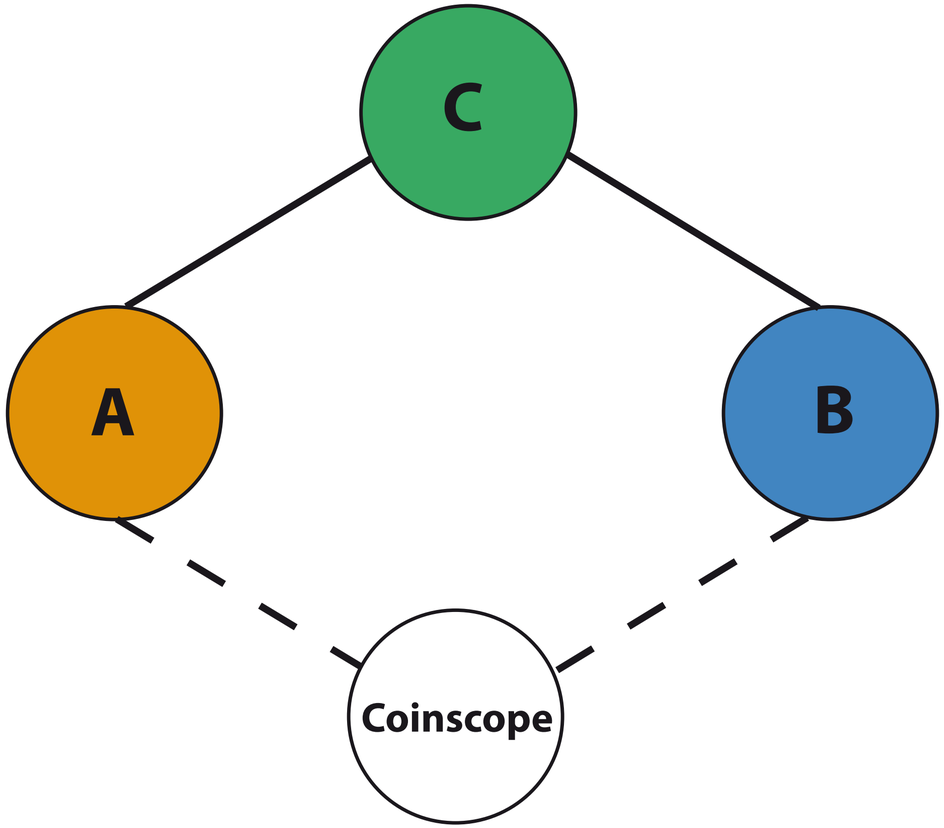
∅

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

C's Mempool

∅

B's Mempool

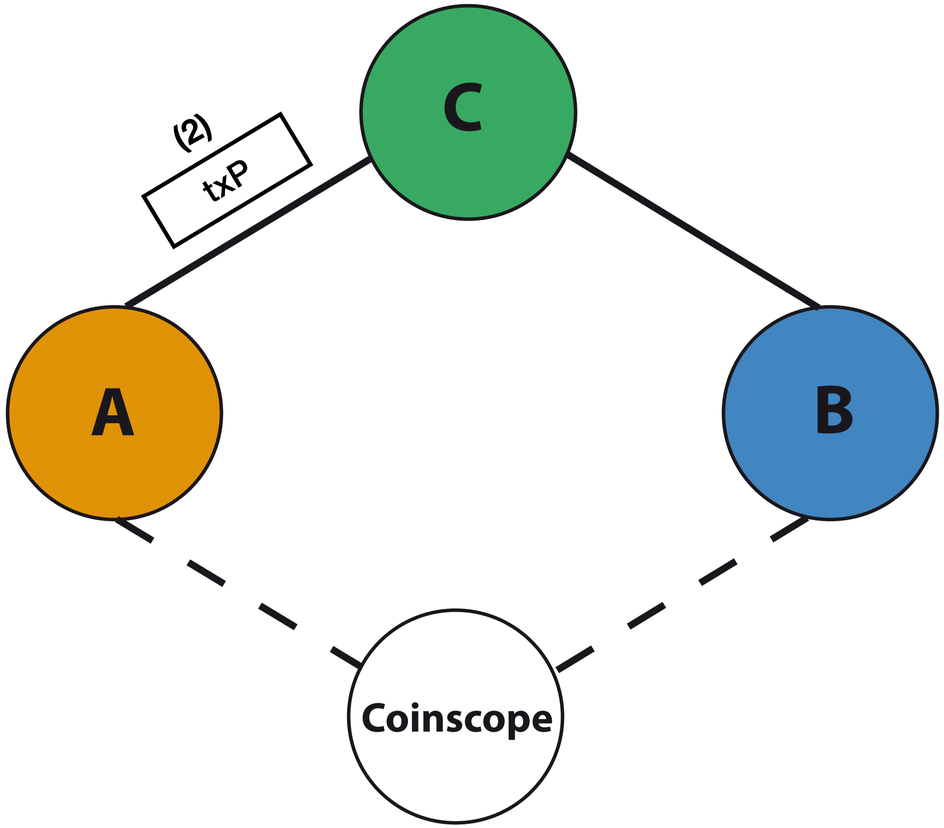
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

C's Mempool

∅

B's Mempool

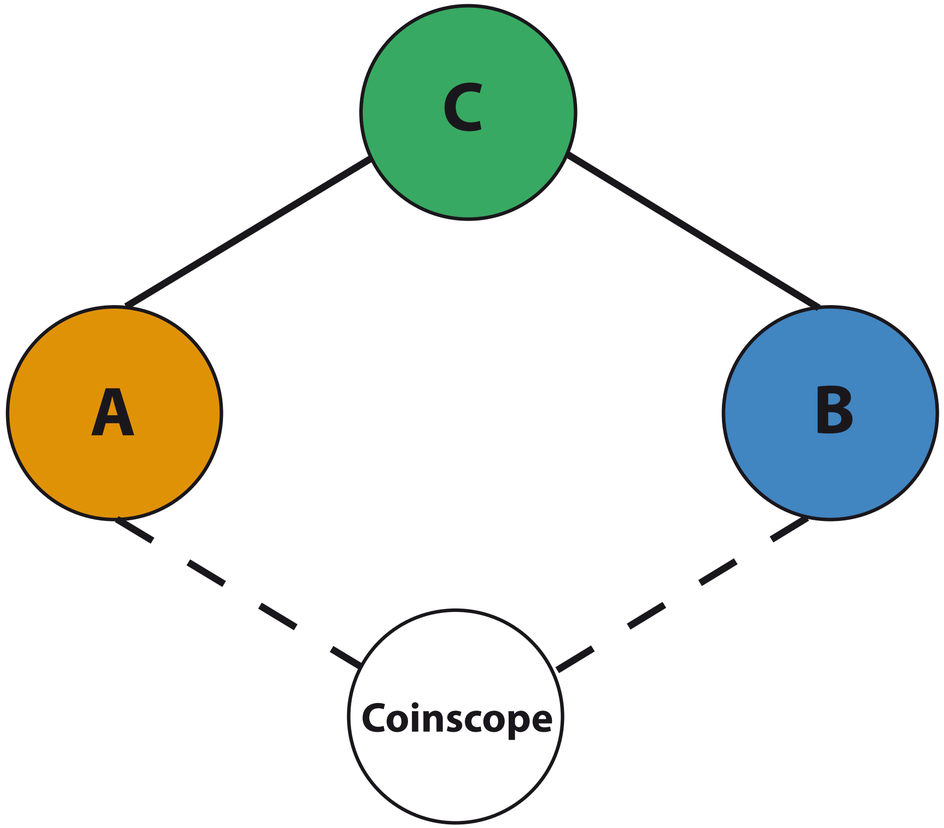
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

C's Mempool

∅

B's Mempool

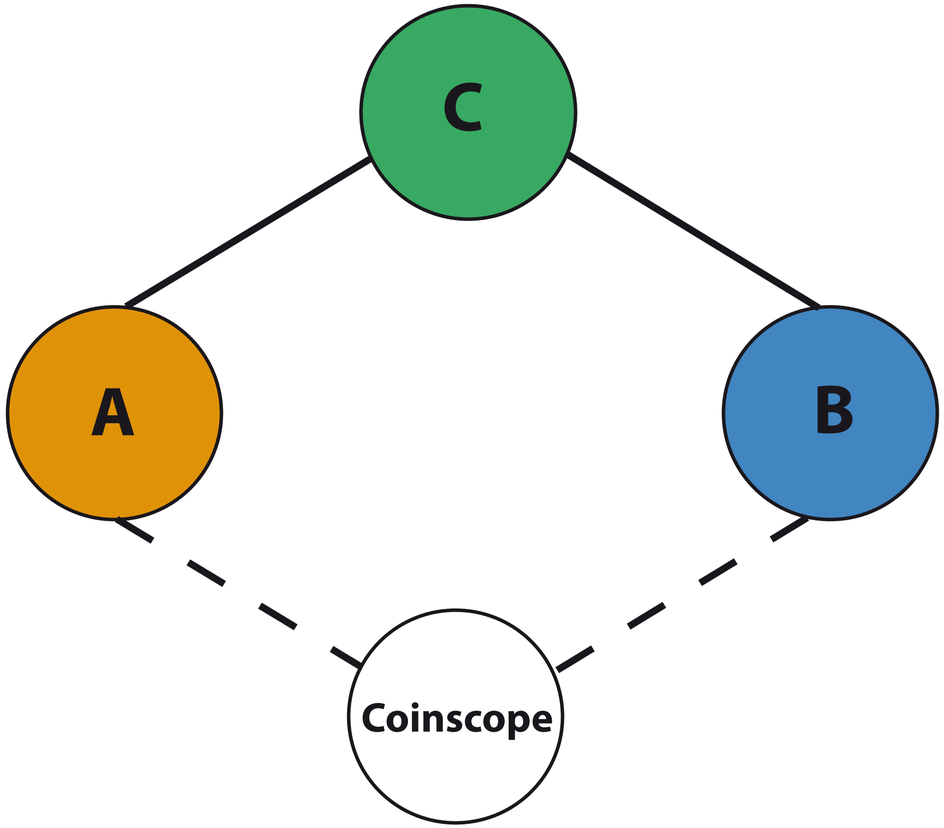
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

C's Mempool

txP (2)

B's Mempool

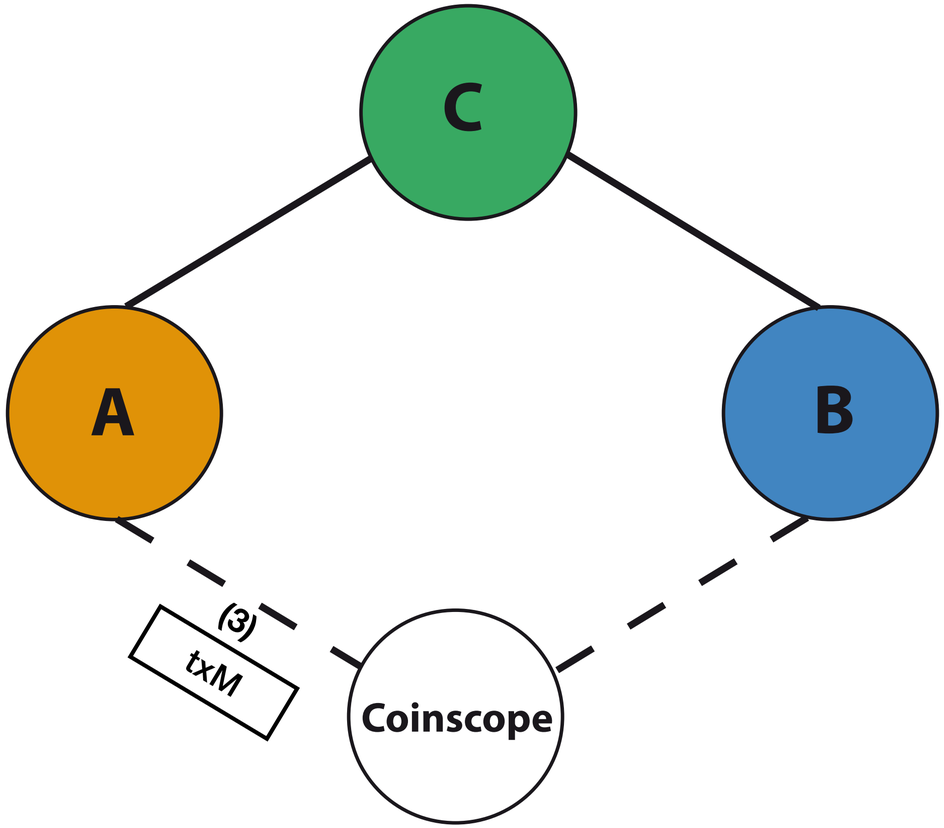
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

C's Mempool

txP (2)

B's Mempool

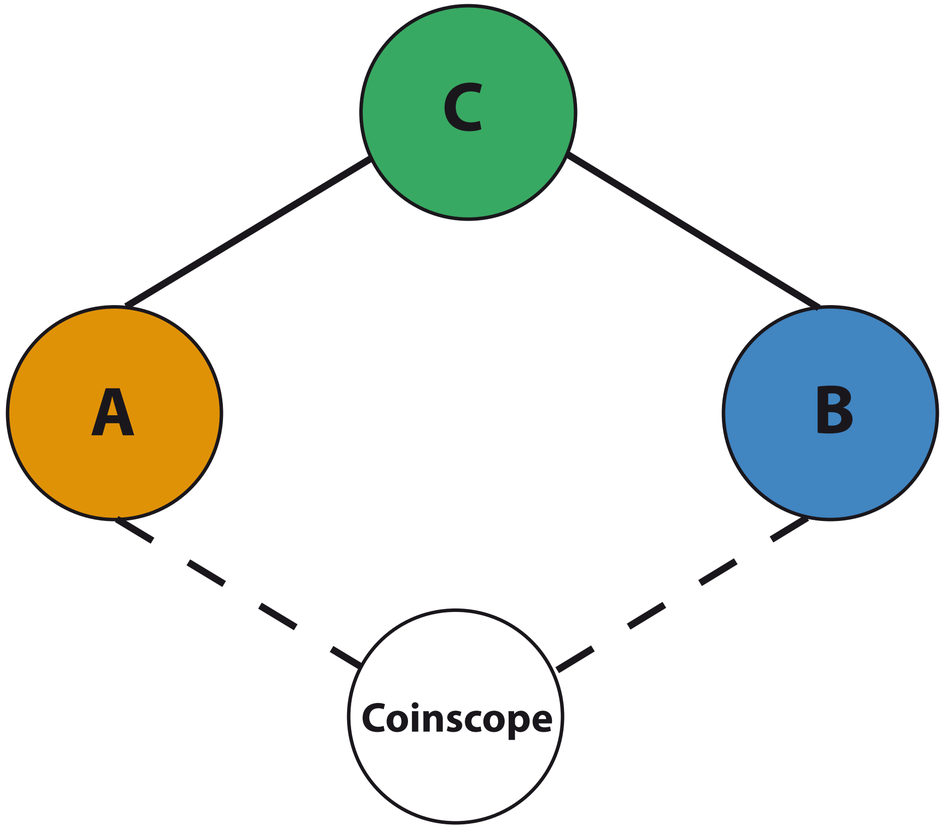
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

C's Mempool

txP (2)

B's Mempool

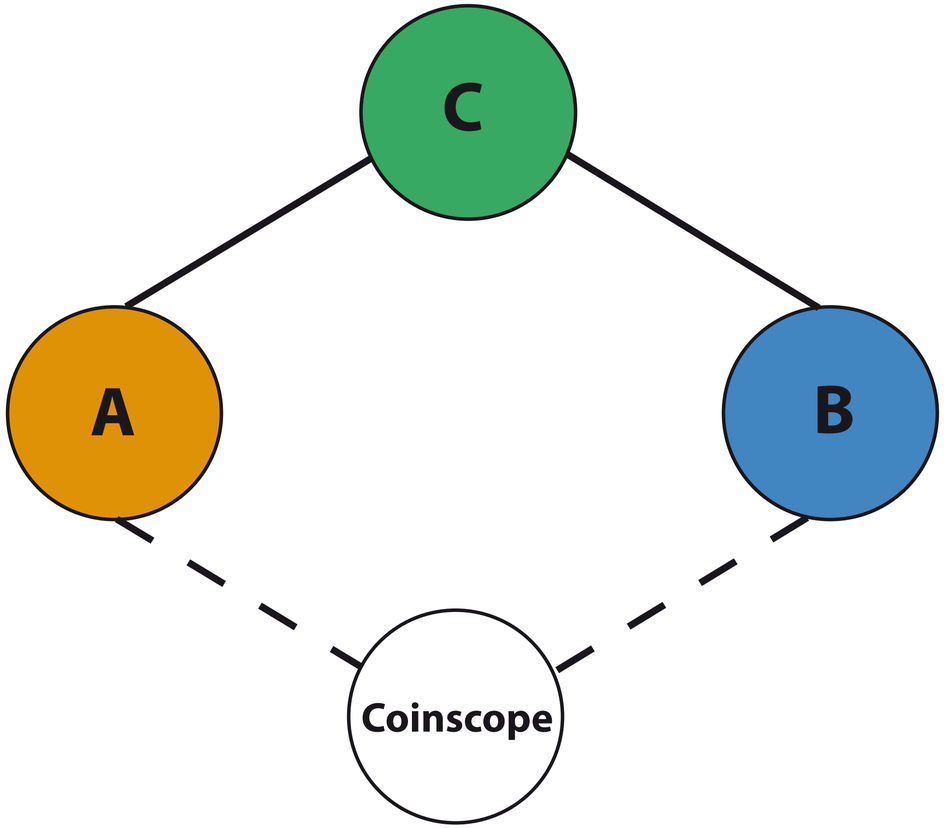
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

txM (3)

C's Mempool

txP (2)

B's Mempool

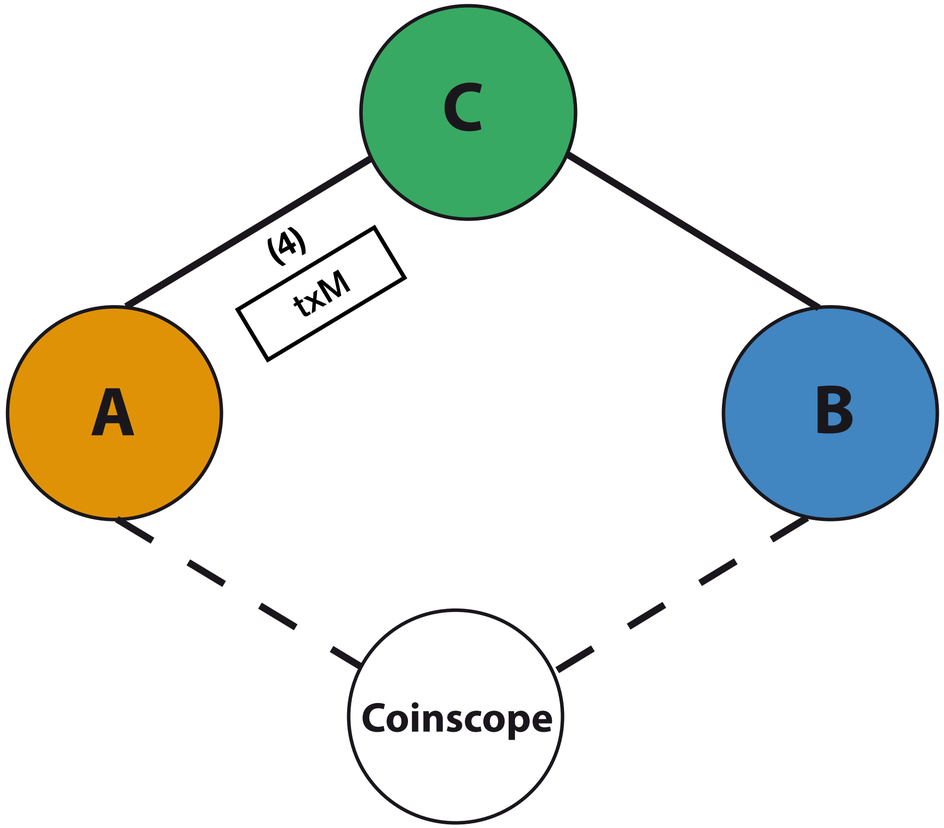
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

txM (3)

C's Mempool

txP (2)

B's Mempool

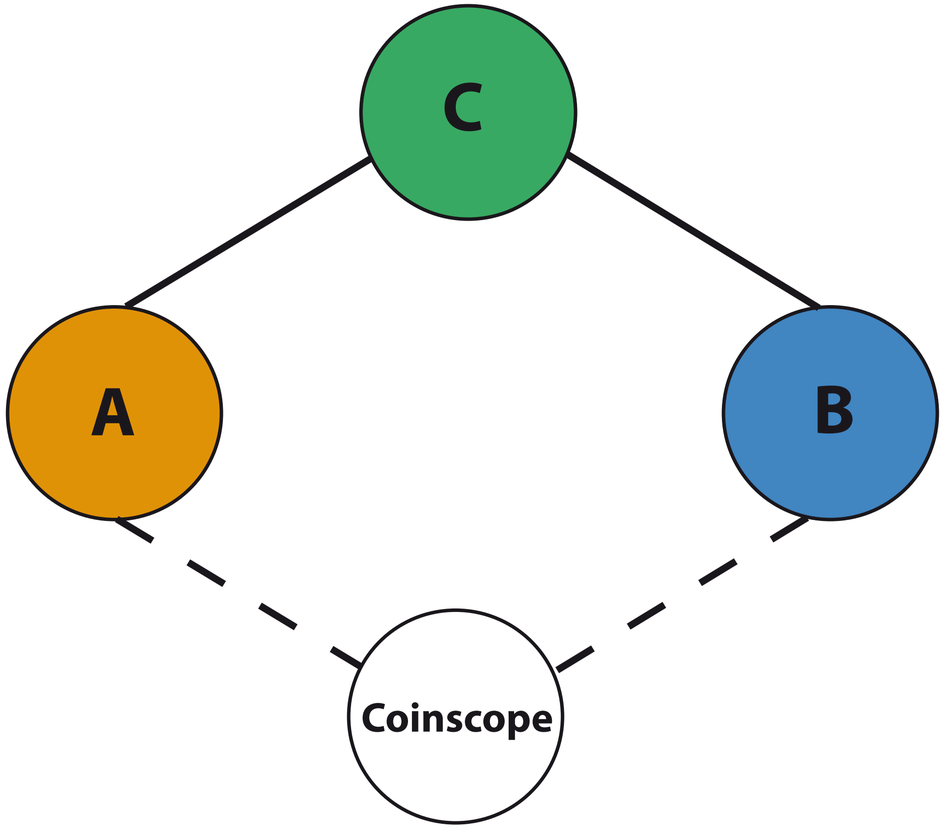
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

txM (3)

C's Mempool

txP (2)

B's Mempool

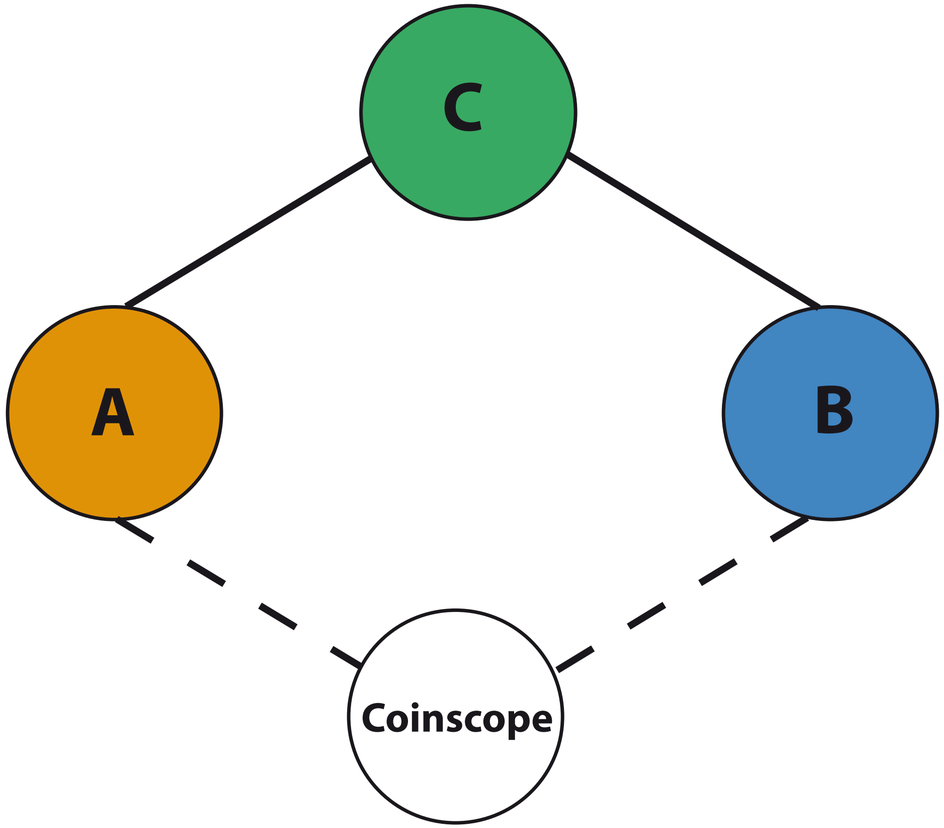
txF (1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP	(1)
txM	(3)

C's Mempool

txP	(2)
txM	(4)

B's Mempool

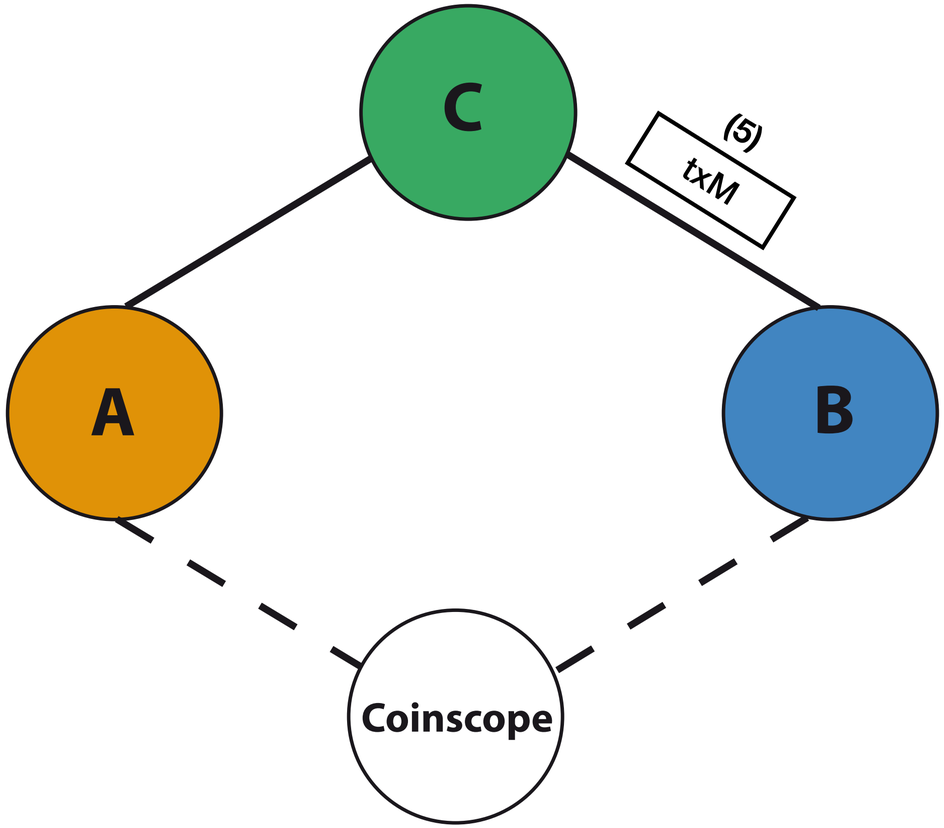
txF	(1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP	(1)
txM	(3)

C's Mempool

txP	(2)
txM	(4)

B's Mempool

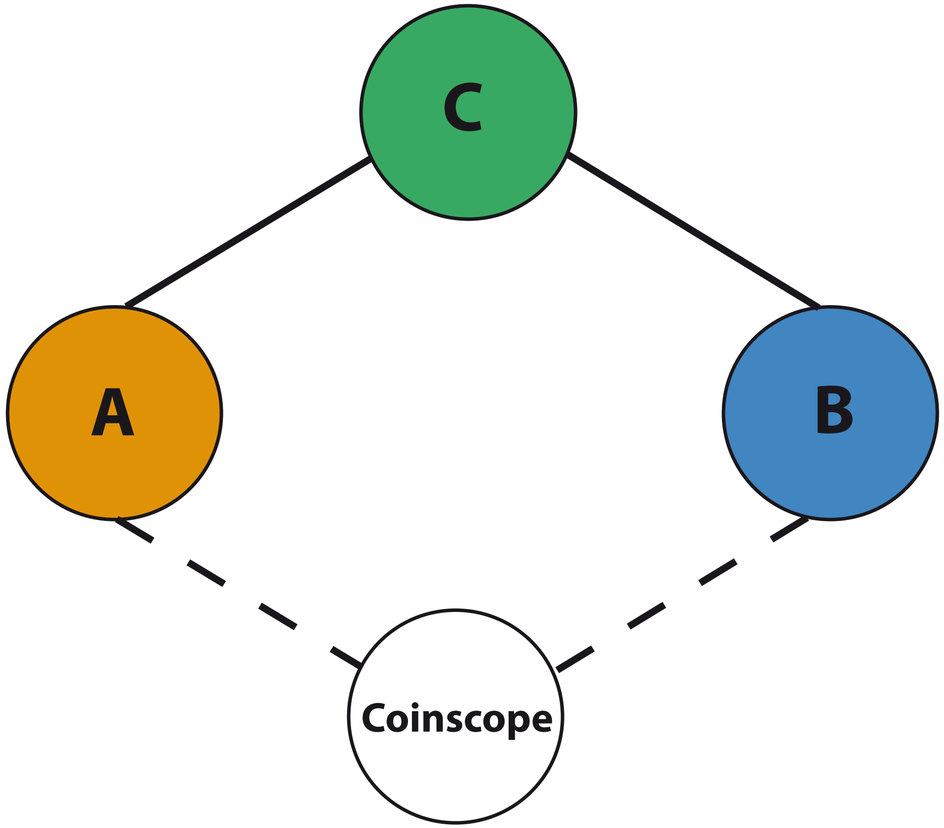
txF	(1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP	(1)
txM	(3)

C's Mempool

txP	(2)
txM	(4)

B's Mempool

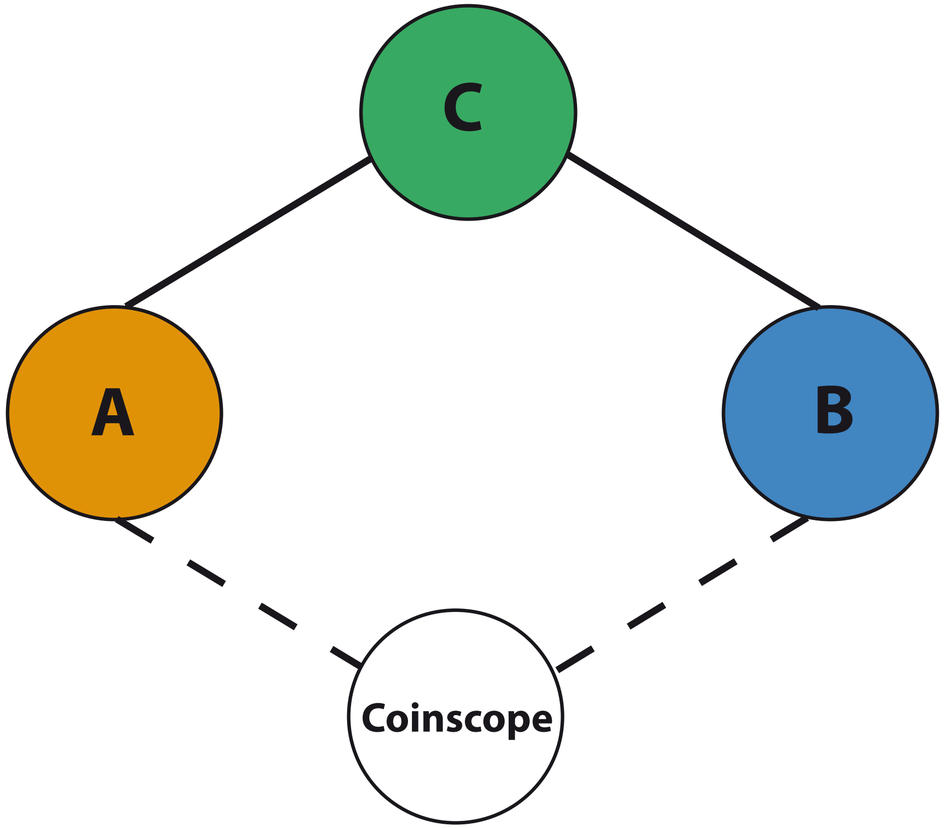
txF	(1)

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, if you add an additional node to the equation it will fail



A's Mempool

txP (1)

txM (3)

C's Mempool

txP (2)

txM (4)

B's Mempool

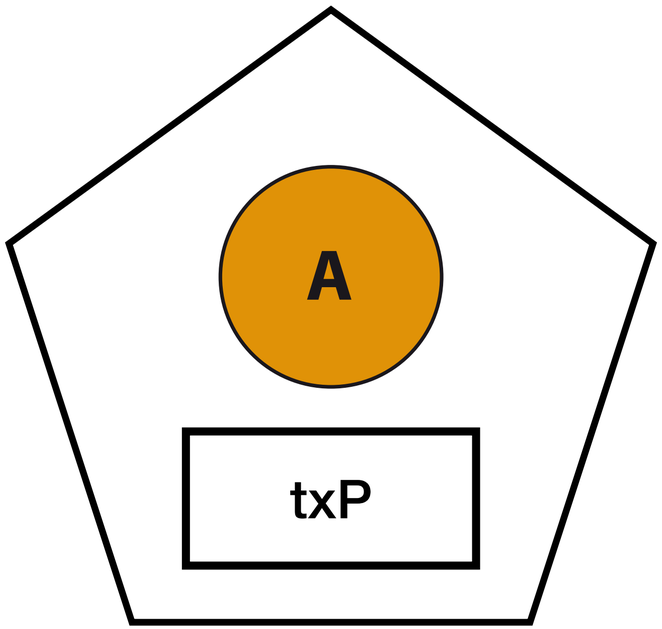
txF (1)

B's MapOrphanTransactions

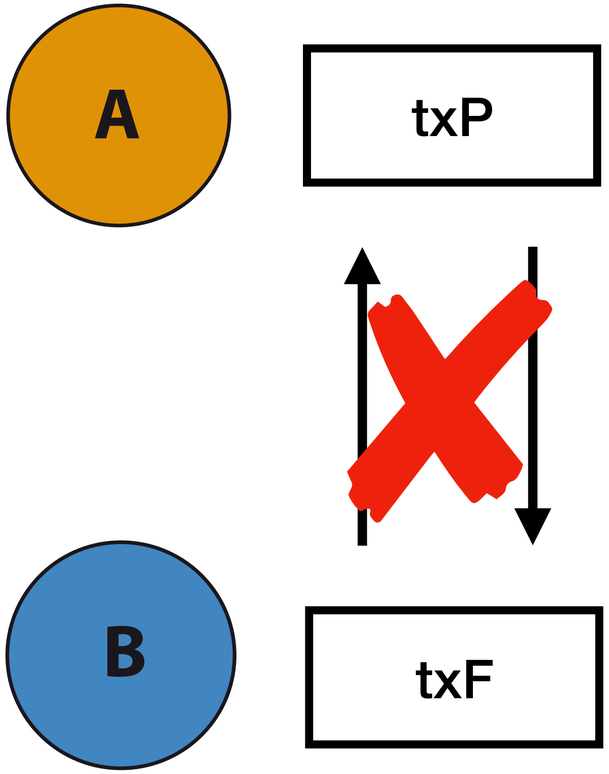
txM (5)

MAKE THIS WORK IN A REAL NETWORK

Isolation



Synchrony



Efficiency

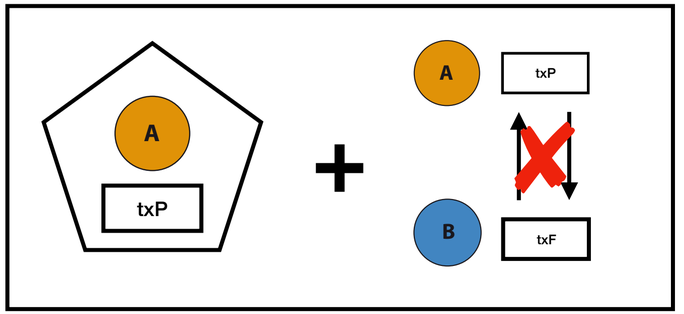
$$\approx 3n\ txs$$

$$\approx 2\sqrt{n}\ txs$$

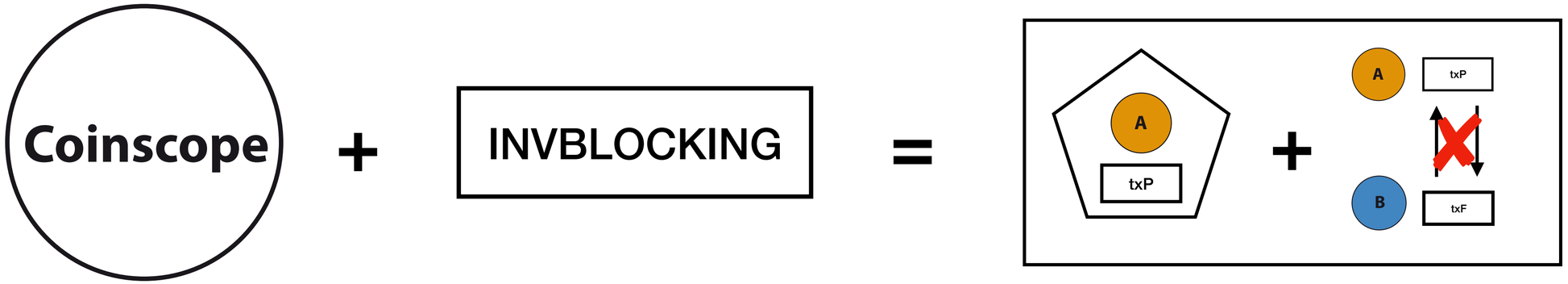
$$n = \#nodes$$

ACHIEVING ISOLATION AND SYNCHRONY

Coinscope + INVBLOCKING =

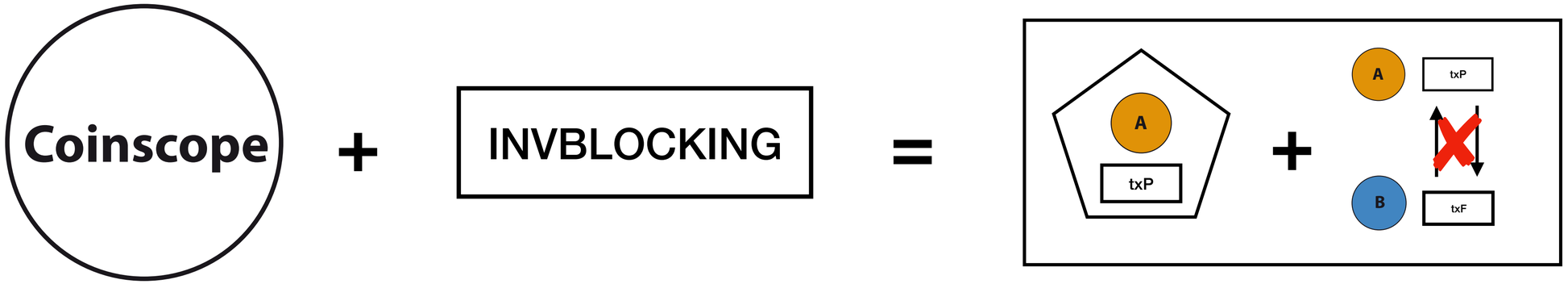


ACHIEVING ISOLATION AND SYNCHRONY



INVBLOCKING

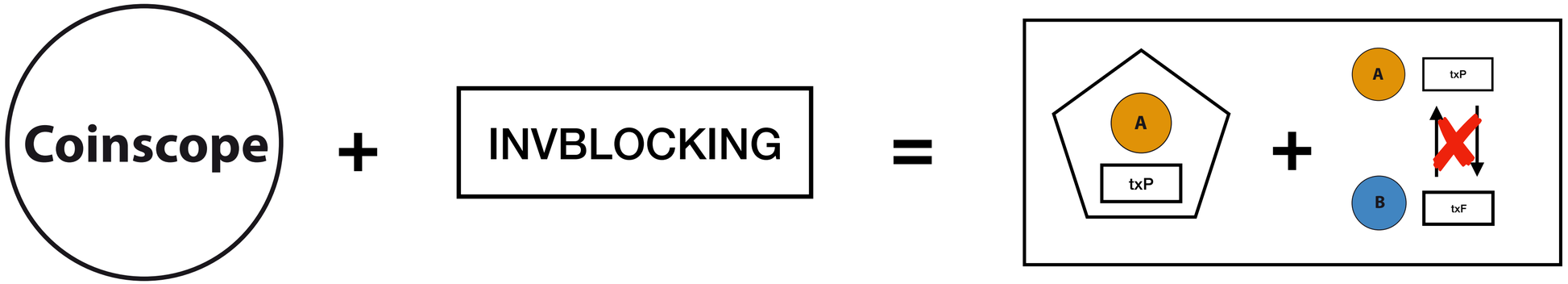
ACHIEVING ISOLATION AND SYNCHRONY



INVBLOCKING

- Send **INV** messages with **txP** and **txF** to the whole network

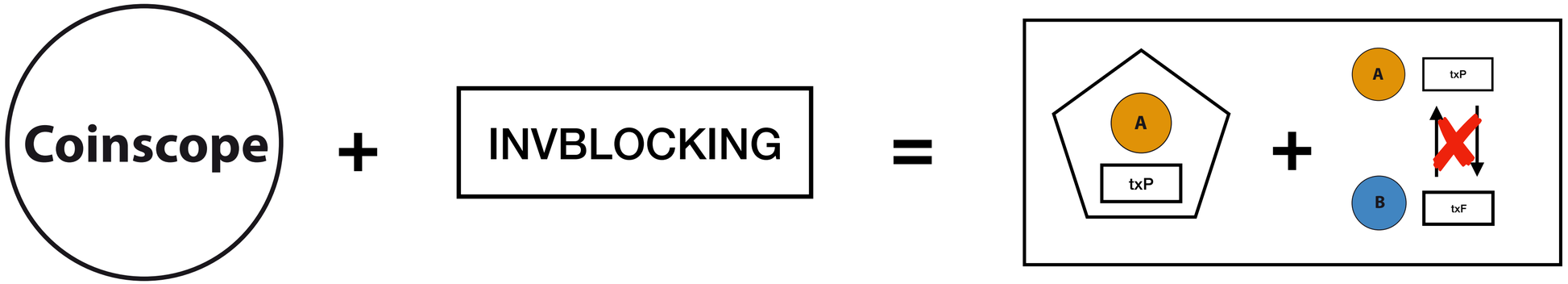
ACHIEVING ISOLATION AND SYNCHRONY



INVBLOCKING

- Send **INV** messages with **txP** and **txF** to the whole network
- Nodes will **ask us** about **txP** and **txF**

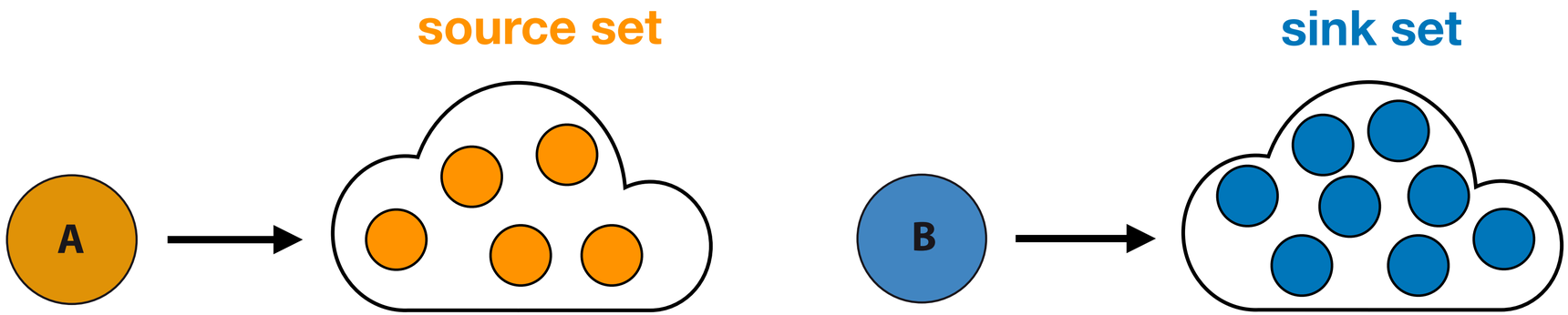
ACHIEVING ISOLATION AND SYNCHRONY



INVBLOCKING

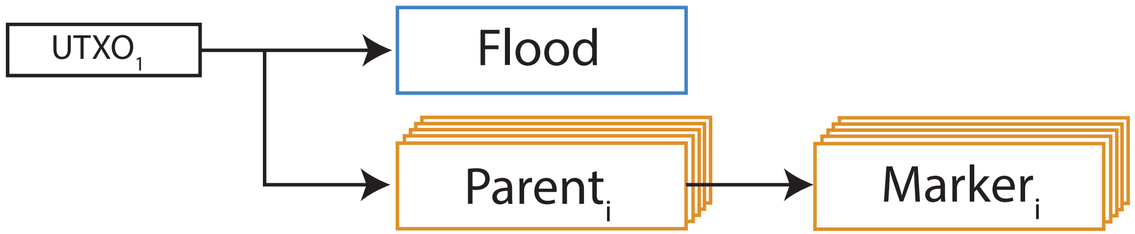
- Send **INV** messages with **txP** and **txF** to the whole network
- Nodes will **ask us** about **txP** and **txF**
- We **withhold** the information effectively **blocking the propagation of txP and txF**

ACHIEVING EFFICIENCY



$$\approx 3n\ txs$$
$$\downarrow$$
$$\approx 2\sqrt{n}\ txs$$

n = source set + sink set



edges are inferred by checking the **propagation of markers** from the **source set to the sink set**

ACHIEVING EFFICIENCY (CONT)

ACHIEVING EFFICIENCY (CONT)

To satisfy the **efficiency property**, we are inferring several edges at the same time

ACHIEVING EFFICIENCY (CONT)

To satisfy the **efficiency property**, we are inferring several edges at the same time

We need to make sure that the MapOrphanTransactions pool **have enough room** to store all our orphans

ACHIEVING EFFICIENCY (CONT)

To satisfy the **efficiency property**, we are inferring several edges at the same time

We need to make sure that the MapOrphanTransactions pool **have enough room** to store all our orphans

Otherwise **false negatives** could occur

ACHIEVING EFFICIENCY (CONT)

To satisfy the **efficiency property**, we are inferring several edges at the same time

We need to make sure that the MapOrphanTransactions pool **have enough room** to store all our orphans

Otherwise **false negatives** could occur

Solution: Inject our own orphans to clear the pool of all network nodes

ACHIEVING EFFICIENCY (CONT)

To satisfy the **efficiency property**, we are inferring several edges at the same time

We need to make sure that the MapOrphanTransactions pool **have enough room** to store all our orphans

Otherwise **false negatives** could occur

Solution: Inject our own orphans to clear the pool of all network nodes

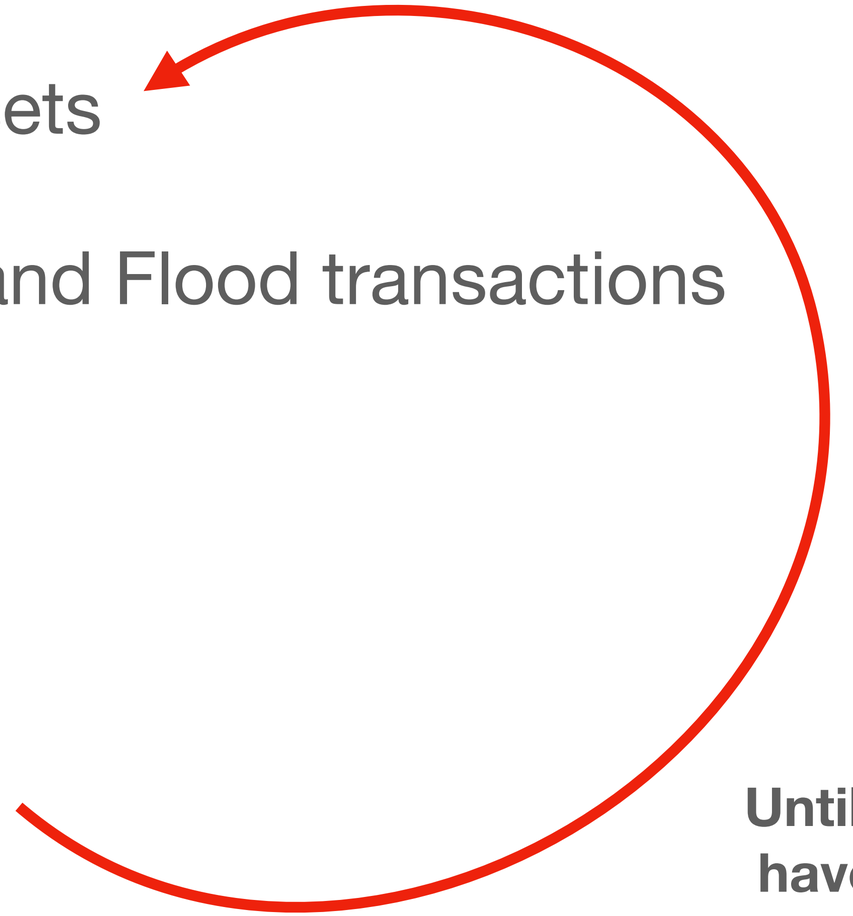
Refer to the paper for details about the orphan pool handling

TXPROBE - PROTOCOL OVERVIEW

- **Choose** source and sink sets
- **Create** Parents, Markers and Flood transactions
- **INVBLOCK** the network
- **Send** transactions
- **Request** markers back

TXPROBE - PROTOCOL OVERVIEW

- **Choose** source and sink sets
- **Create** Parents, Markers and Flood transactions
- **INVBLOCK** the network
- **Send** transactions
- **Request** markers back



Until every pair of nodes have been in a different set al least once

TXPROBE - COSTS OVERALL

For a network like **Bitcoin mainnet:**

nodes ≈ 10000

time ≈ 8.25 hours

cost = 573210-764280 satoshi (5 sat/byte) ≈ **$(20-30)**

TXPROBE - DATA VALIDATION (40 TRIALS)

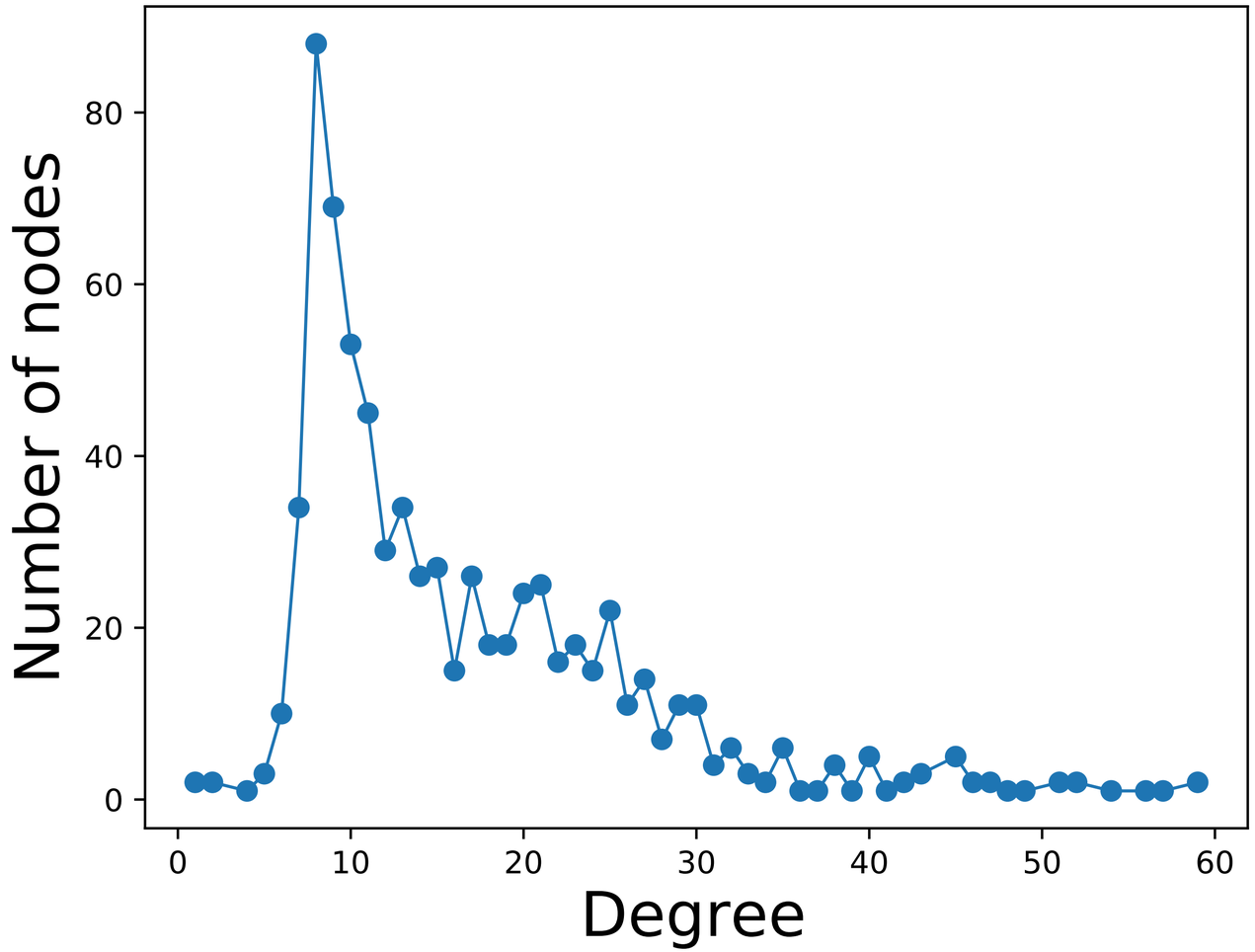
We run 5 Bitcoin Core nodes as **ground truth**

We define our precision / recall by checking how well can we infer the ground truth nodes connections

With 95% confidence:

- **Precision** = 100%
- **Recall** = 93.86% - 95.45%

TXPROBE - TESTNET TOPOLOGY



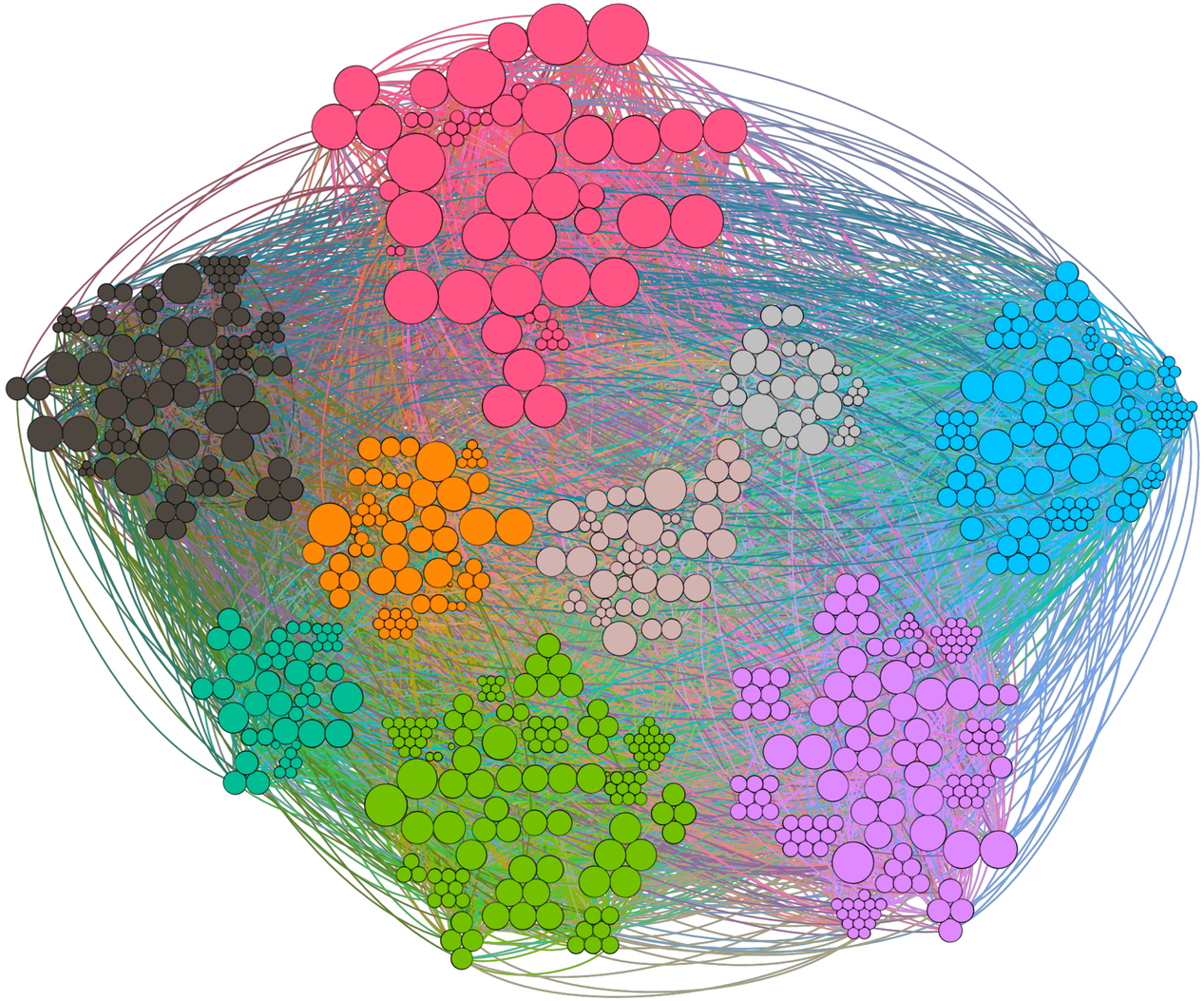
nodes = 733

edges = 6090

avg. degree ≈ 16.6

most common degree = 8
(12% of the nodes)

TXPROBE - TESTNET TOPOLOGY (CONT)



precision = 100%

recall = 97.40%

size ⟶ degree

color ⟶ Louvain community unfolding

Higher **community structure** and **modularity** than random graph

The snapshot contains a **clique of 24 nodes**

WHY TESTNET AND NO MAINNET?

- TxProbe is rather invasive: it empties the **MapOrphanTransactions pool** of all nodes in the network every round

- We could not measure the implication that such behavior may have had on the **propagation of regular transactions**

LIMITATIONS OF TOPOLOGY MEASUREMENT

We can only target the **reachable network (≈10% of the total nodes)**

The “vanilla P2P network” is not the only mechanism used:

- Private miner peering
- Optimized relay networks (FIBRE, Falcon, …)

TxProbe relies in **implementation specific features**

TACIT MOUSE AND CAT GAME

Select orphan transaction uniformly for eviction #14626

Merged **MarcoFalke** merged 1 commit into `bitcoin:master` from `sipa:201810_uniform_orphan_eviction` 3 days ago

Conversation 20 · Commits 1 · Checks 0 · Files changed 1

sipa commented on 31 Oct 2018 — Member

The previous code was biased towards evicting transactions whose txid has a larger gap (lexicographically) with the previous txid in the orphan pool.

randomize GETDATA(tx) request order and introduce bias toward outbound #14897

Merged **sipa** merged 1 commit into `bitcoin:master` from `naumenkogs:master` 10 days ago

Conversation 115 · Commits 1 · Checks 0 · Files changed 6

naumenkogs commented on 8 Dec 2018 • edited by MarcoFalke — Contributor

This code makes executing two particular (and potentially other) attacks harder.

InvBlock

TACIT MOUSE AND CAT GAME

Select orphan transaction uniformly for eviction #14626

Merged **MarcoFalke** merged 1 commit into `bitcoin:master` from `sipa:201810_uniform_orphan_eviction` 3 days ago

Conversation 20 · Commits 1 · Checks 0 · Files changed 1

sipa commented on 31 Oct 2018 — Member

The previous code was biased towards evicting transactions whose txid has a larger gap (lexicographically) with the previous txid in the orphan pool.

randomize GETDATA(tx) request order and introduce bias toward outbound #14897

Merged **sipa** merged 1 commit into `bitcoin:master` from `naumenkogs:master` 10 days ago

Conversation 115 · Commits 1 · Checks 0 · Files changed 6

naumenkogs commented on 8 Dec 2018 • edited by MarcoFalke — Contributor

This code makes executing two particular (and potentially other) attacks harder.

InvBlock

CONCLUSIONS

TxProbe can be use to reconstruct the topology of Bitcoin (and its forks) with reasonably low cost

The technique relies on implementation specific behavior (Core client)

Is topology hiding a design goal? Focus on achieving it robustly

Is it a mean to achieve other goals (e.g: stronger privacy)? Deploy built-in measurement-supporting mechanisms (e.g: Tor project)

QUESTIONS