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

P/SA RESEARCH

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).

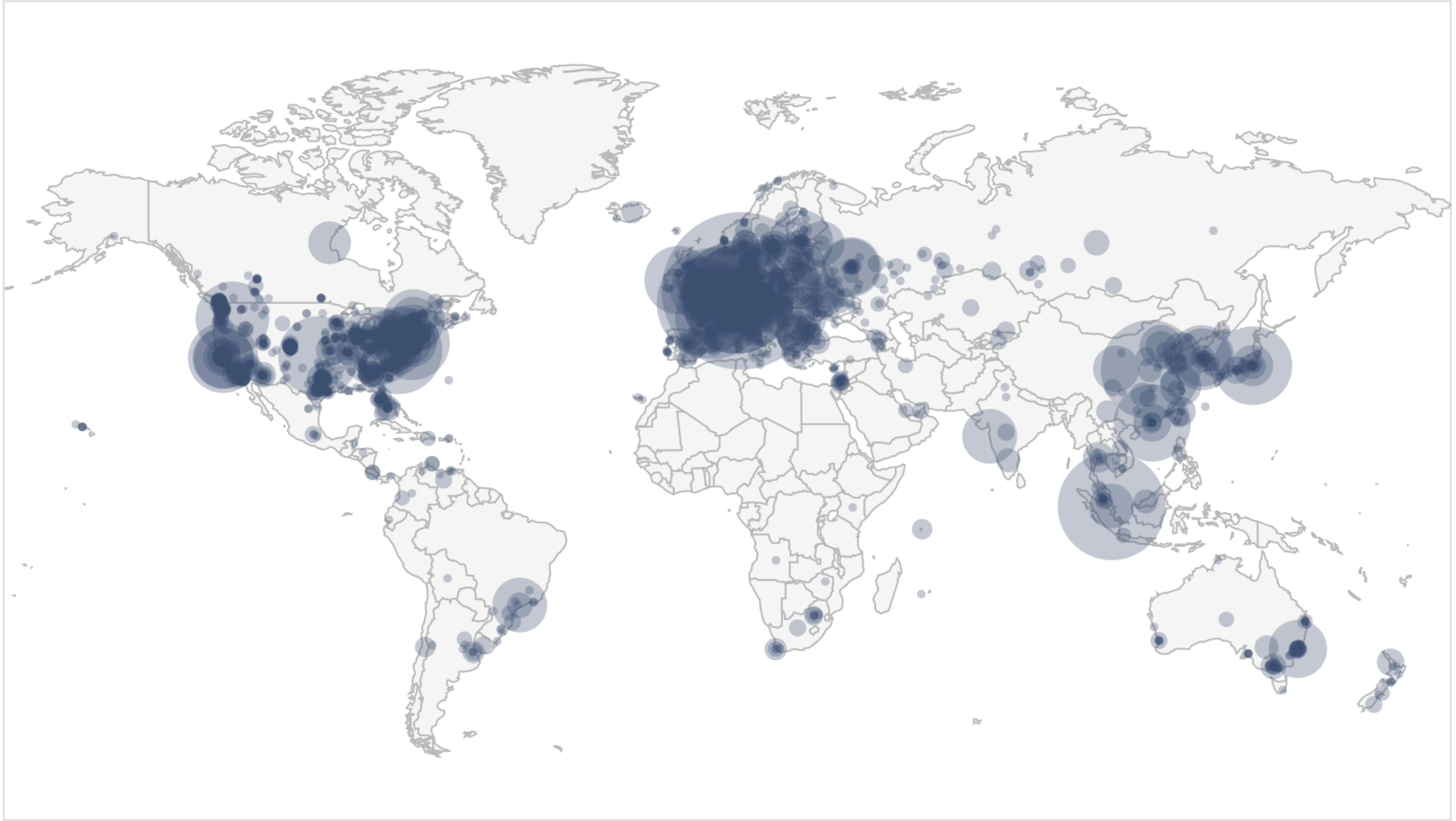
10365 NODES

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) »



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).

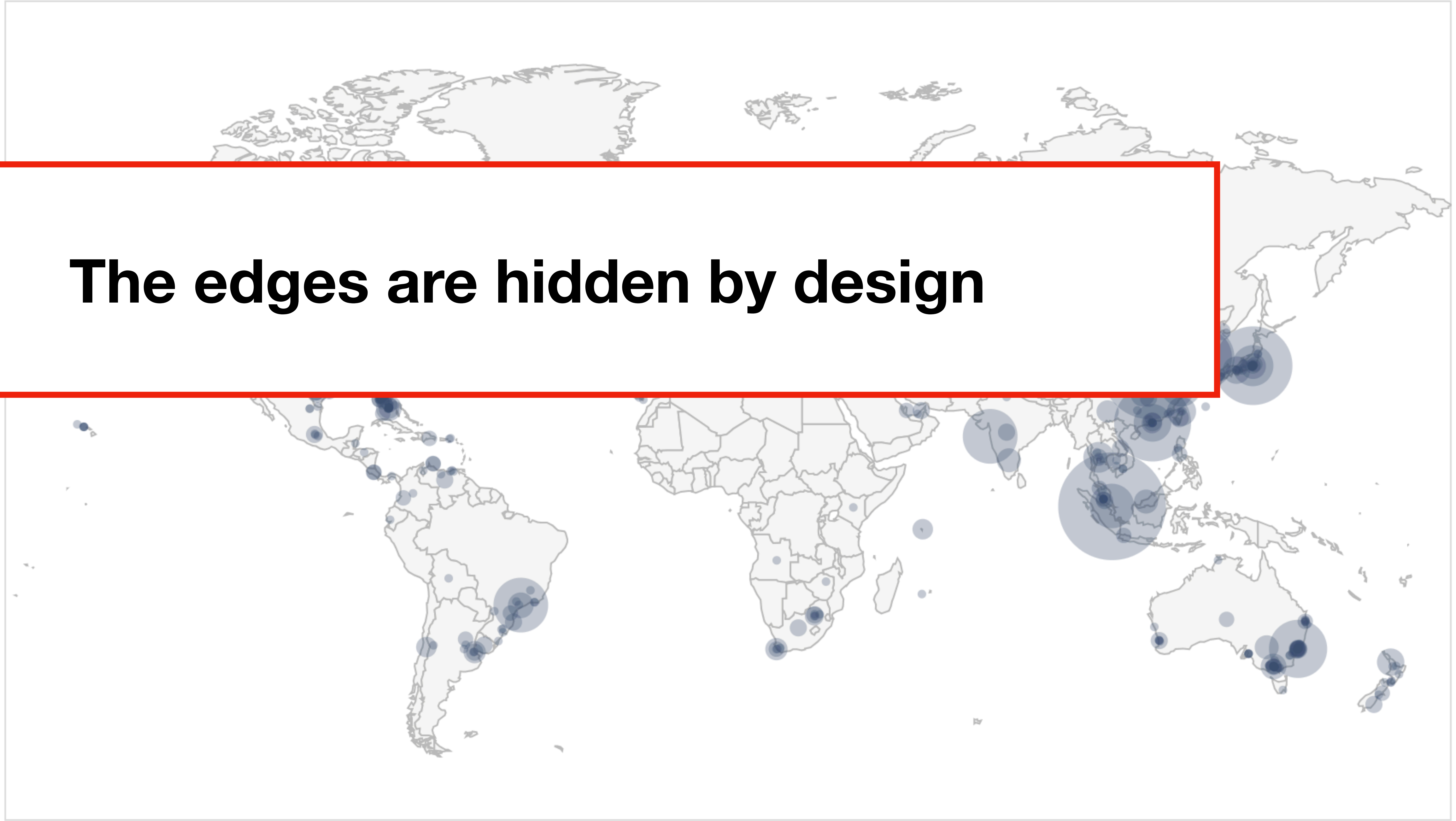
10365 NODES

24-hour charts »

Top 10 countries with their respecti
reachable nodes are as follow.

RANK	COUNTRY	NOD
1	United States	257
2	Germany	196
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 edges are hidden by design

WHY HAVE A HIDDEN TOPOLOGY?

An open topology **could ease** different types of attacks:

- Transaction deanonymization
- Network based attacks (e.g: Eclipse attacks)

The **current approach** of the Bitcoin Core is to **keep it hidden**

WHY HAVE AN OPEN TOPOLOGY?

We know nothings about how the network really is:

- Is the network decentralised?
- Are there supernodes controlling the network traffic?
    - Information withholding
    - Censorship
- Are there weak spots in the network that can be easily isolated?

Security by obscurity does not seem the proper way to go

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

BACKGROUND

Our inferring technique is based on **transaction propagation**

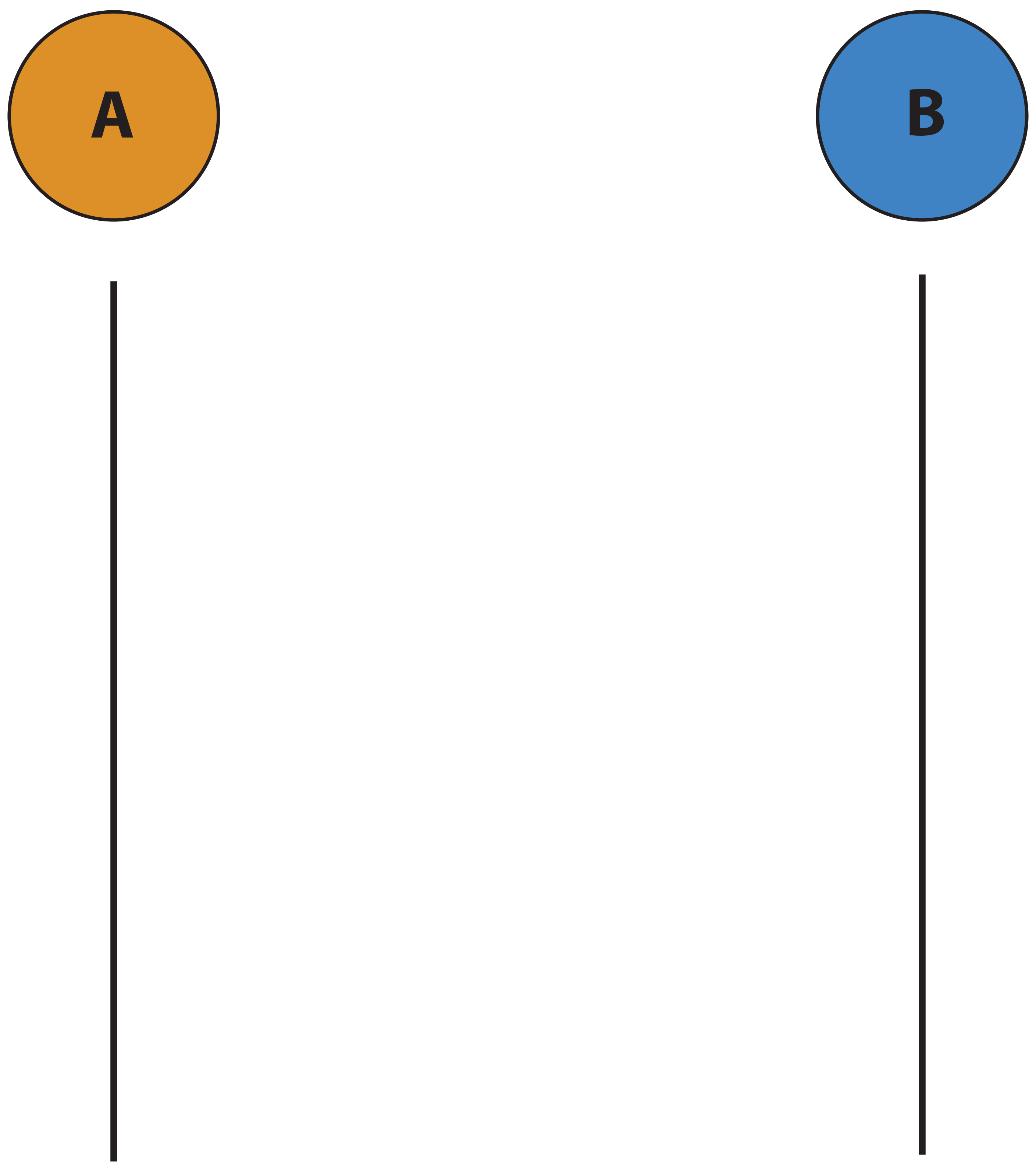
We take advantage of how transactions are handled by nodes:

- **orphans transactions**
- **double-spending transactions**

TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

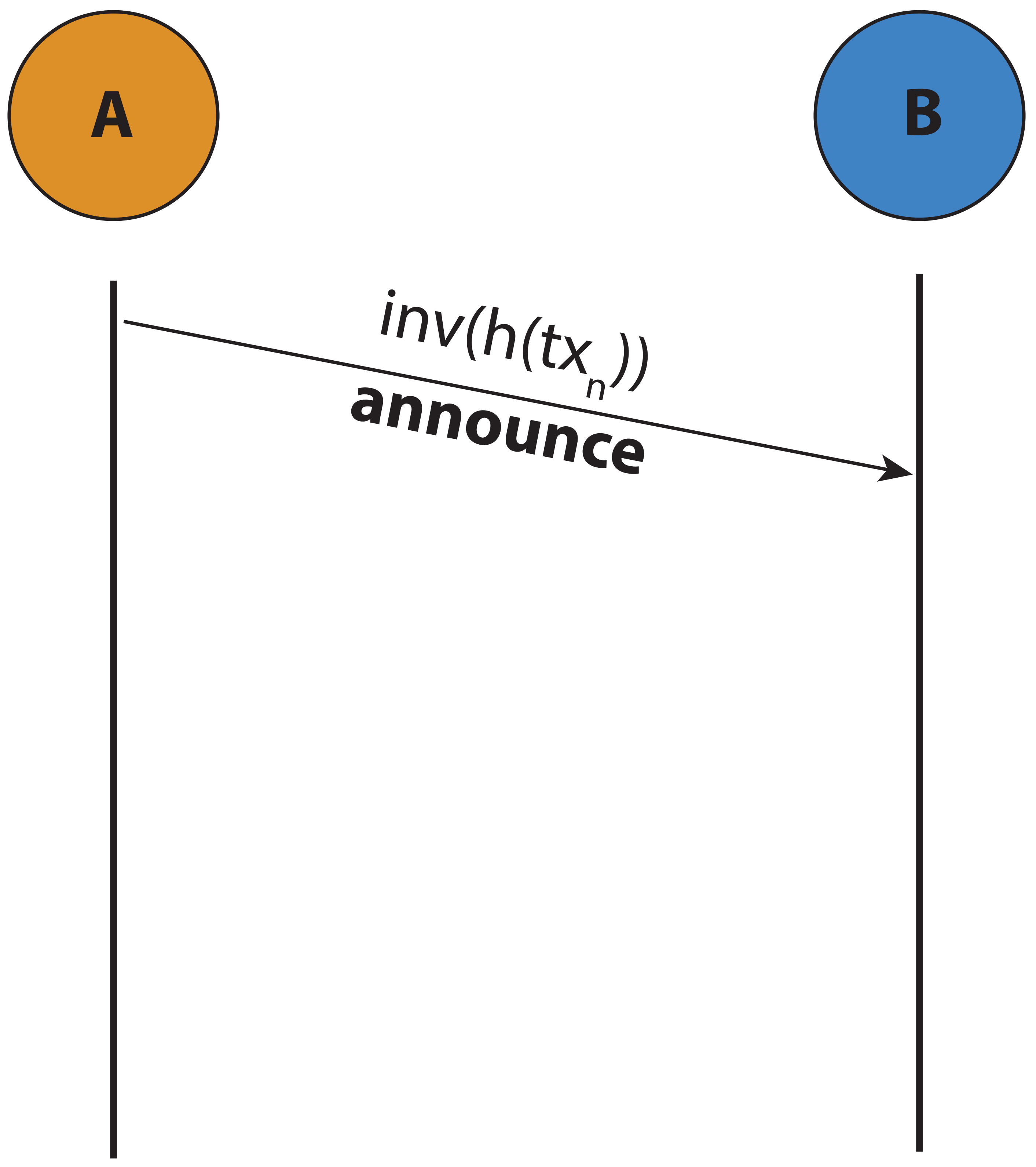
Transaction in mempool are **eventually propagated** throughout the node neighborhood



TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

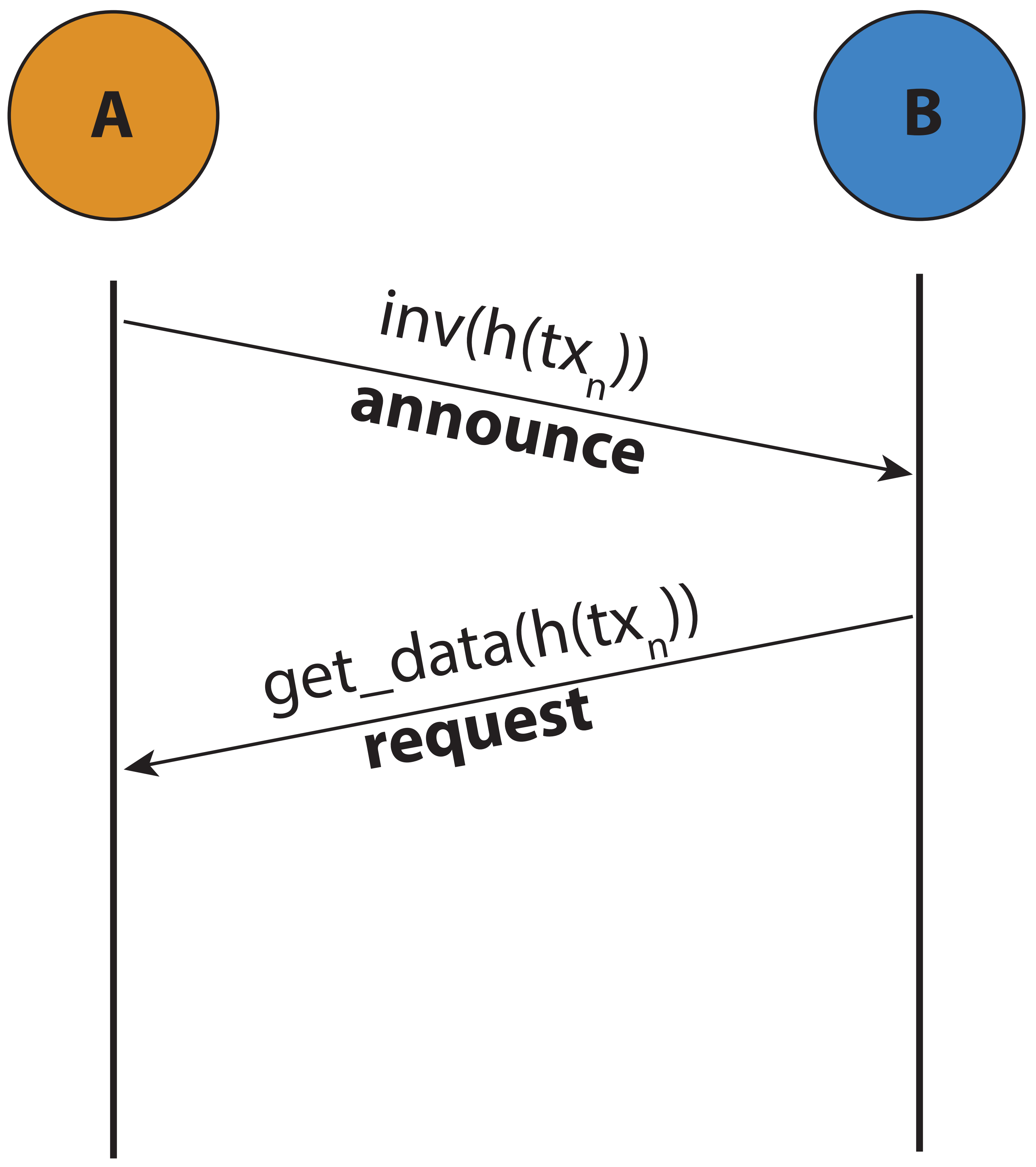
Transaction in mempool are **eventually propagated** throughout the node neighborhood



TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

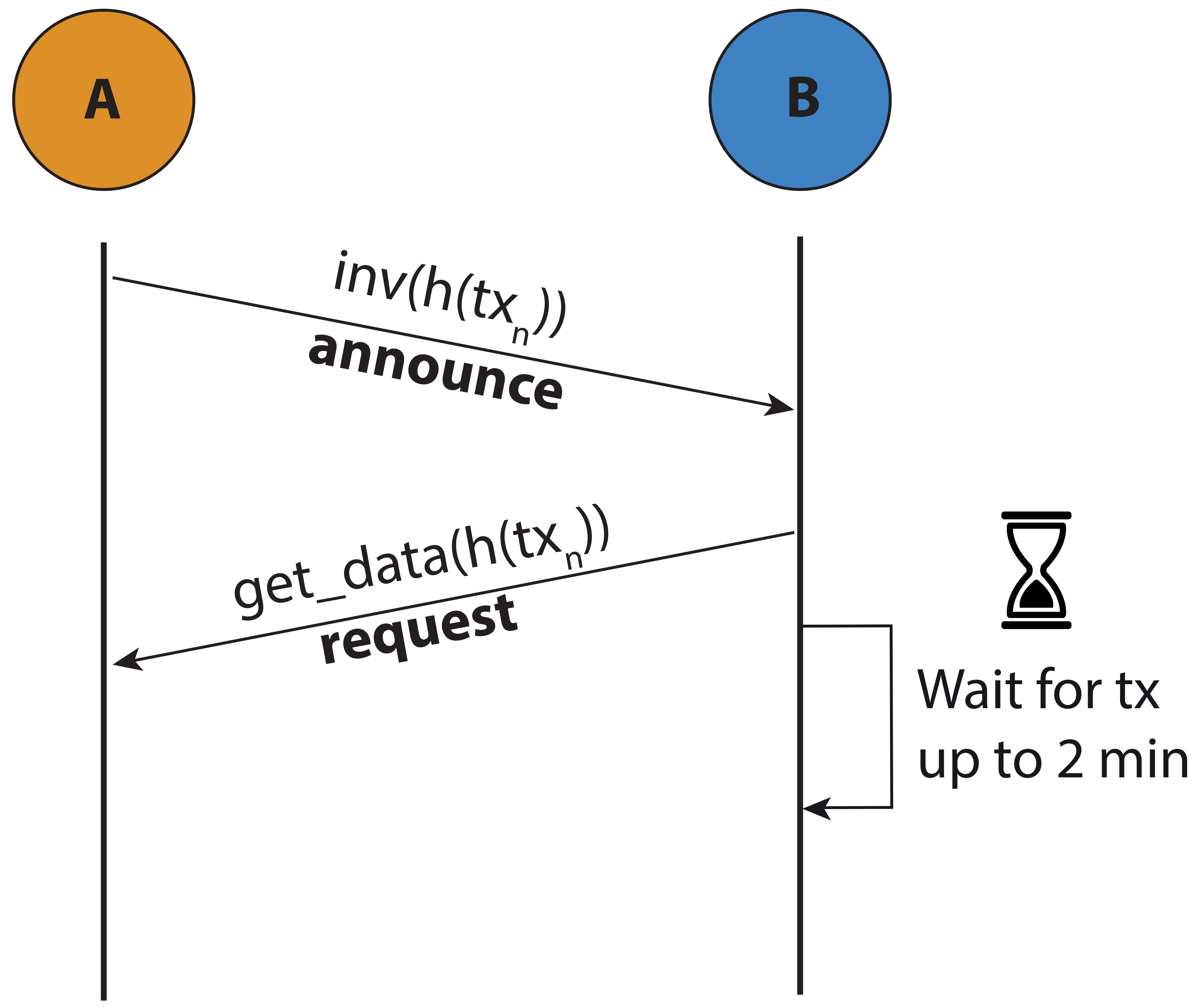
Transaction in mempool are **eventually propagated** throughout the node neighborhood



TRANSACTION PROPAGATION IN BITCOIN

Valid transaction are stored in **mempool**

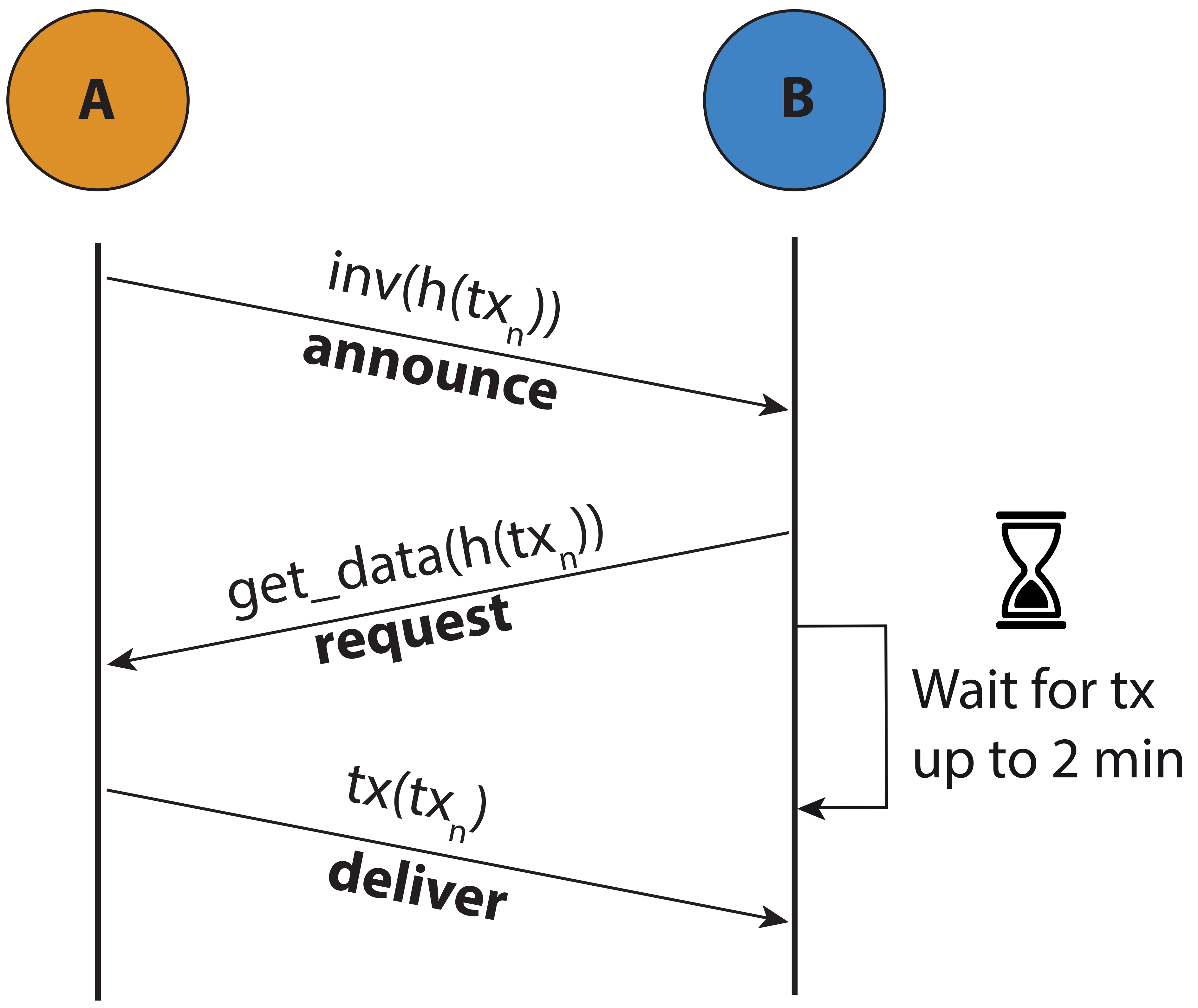
Transaction in mempool are **eventually propagated** throughout the node neighborhood



TRANSACTION PROPAGATION IN BITCOIN

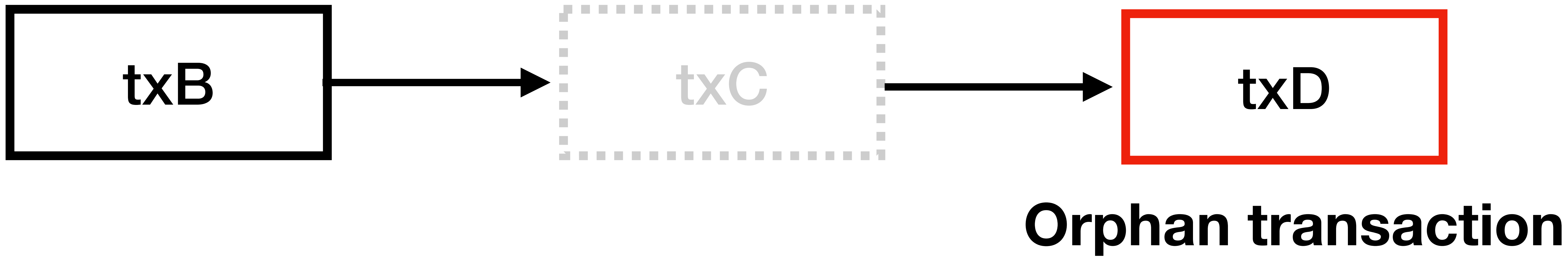
Valid transaction are stored in **mempool**

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



ORPHAN TRANSACTIONS

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

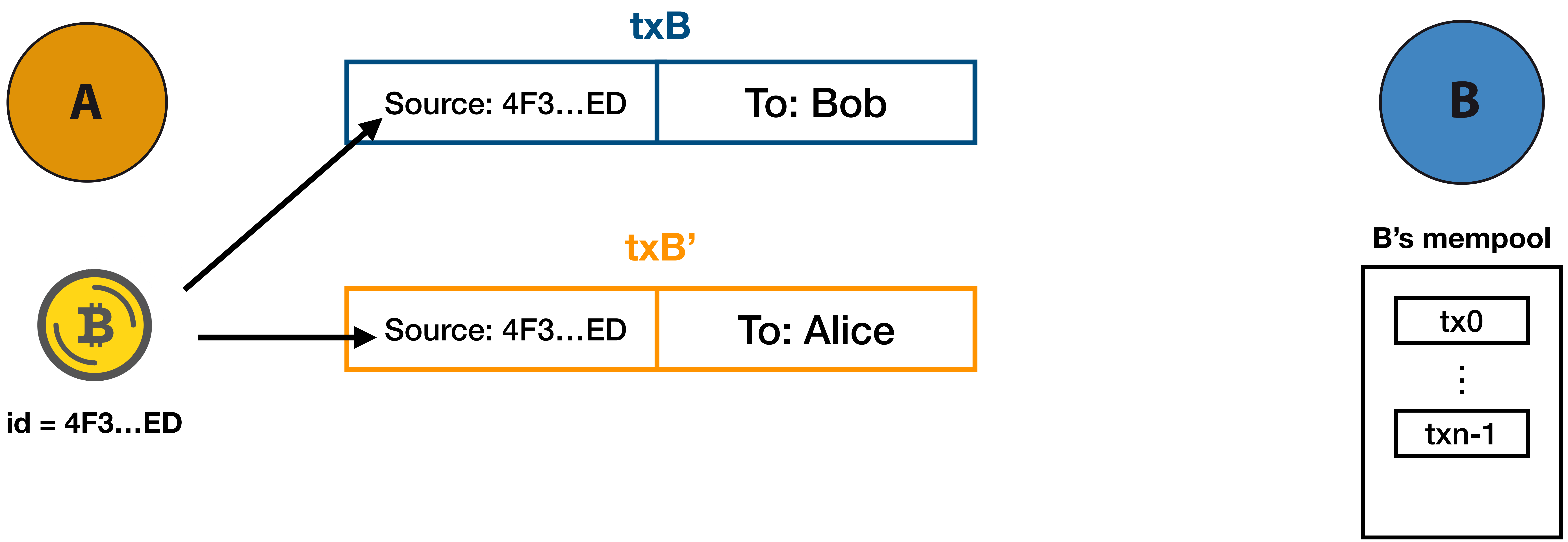


They can not be validated, so they are stored in a separated data structure known as **MapOrphanTransactions (or OrphanPool for short)**

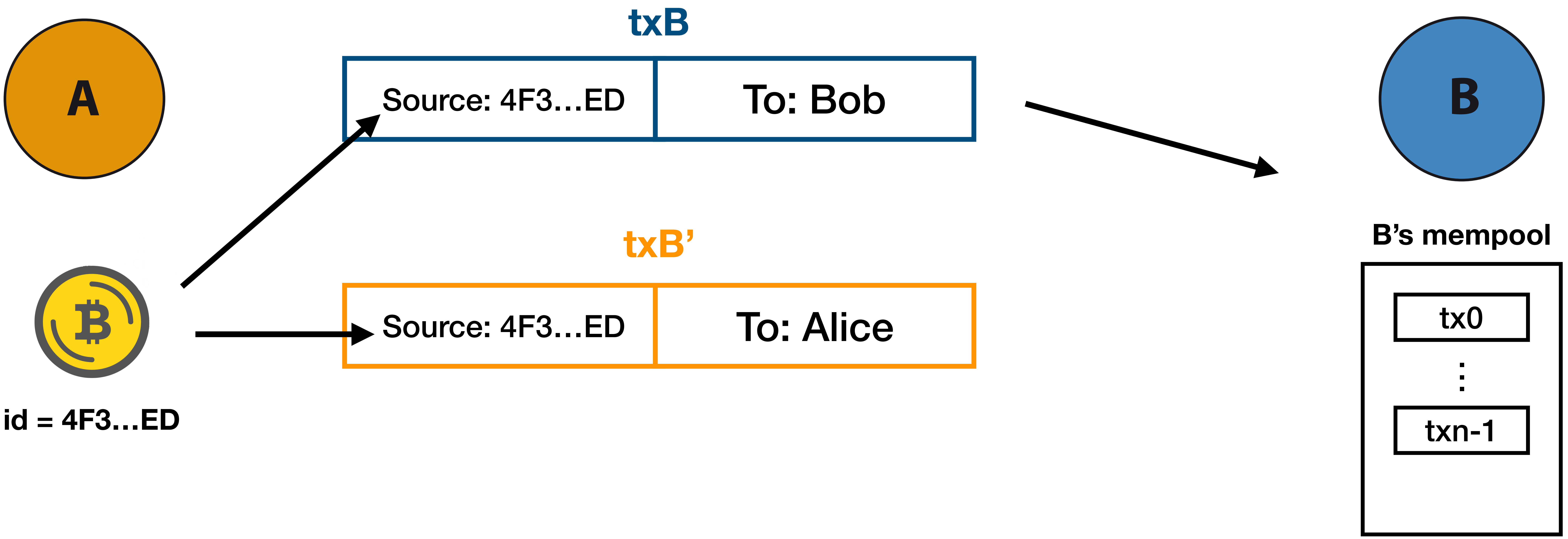
Transactions in MapOrphanTransactions are **NOT forwarded to any node**

If the same transactions is offered again to the node **(inv message)**, it will not requested back **(getaddr)**

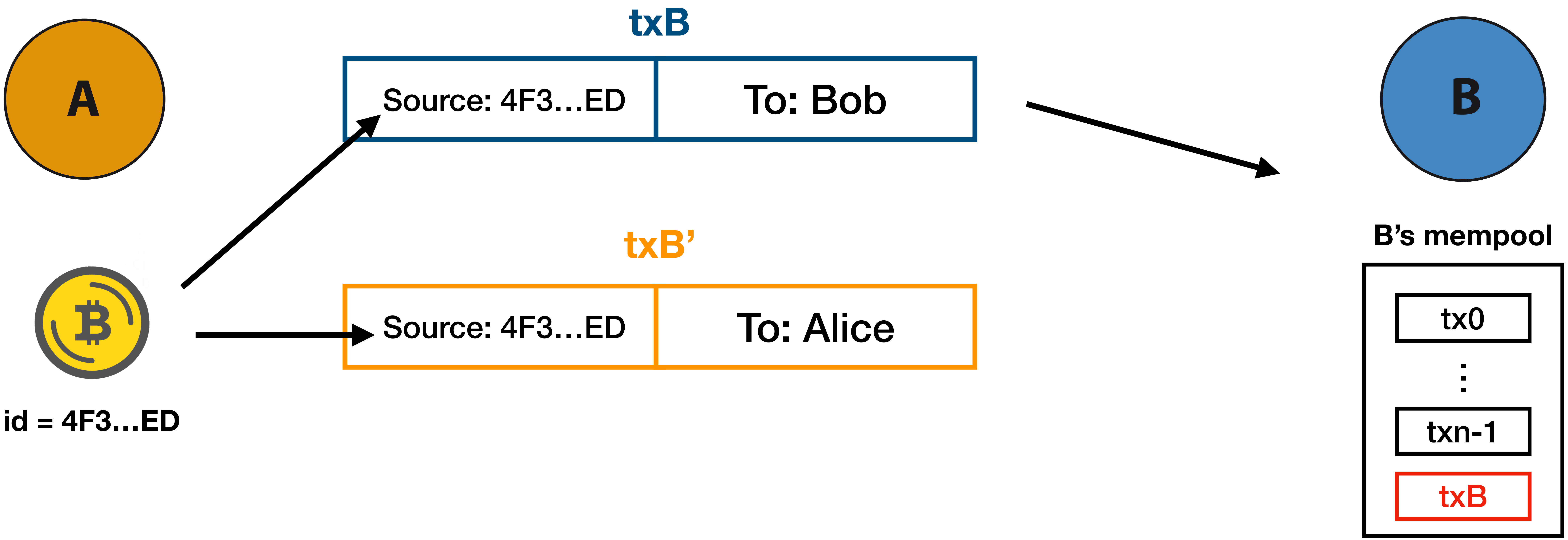
DOUBLE-SPENDING TRANSACTIONS



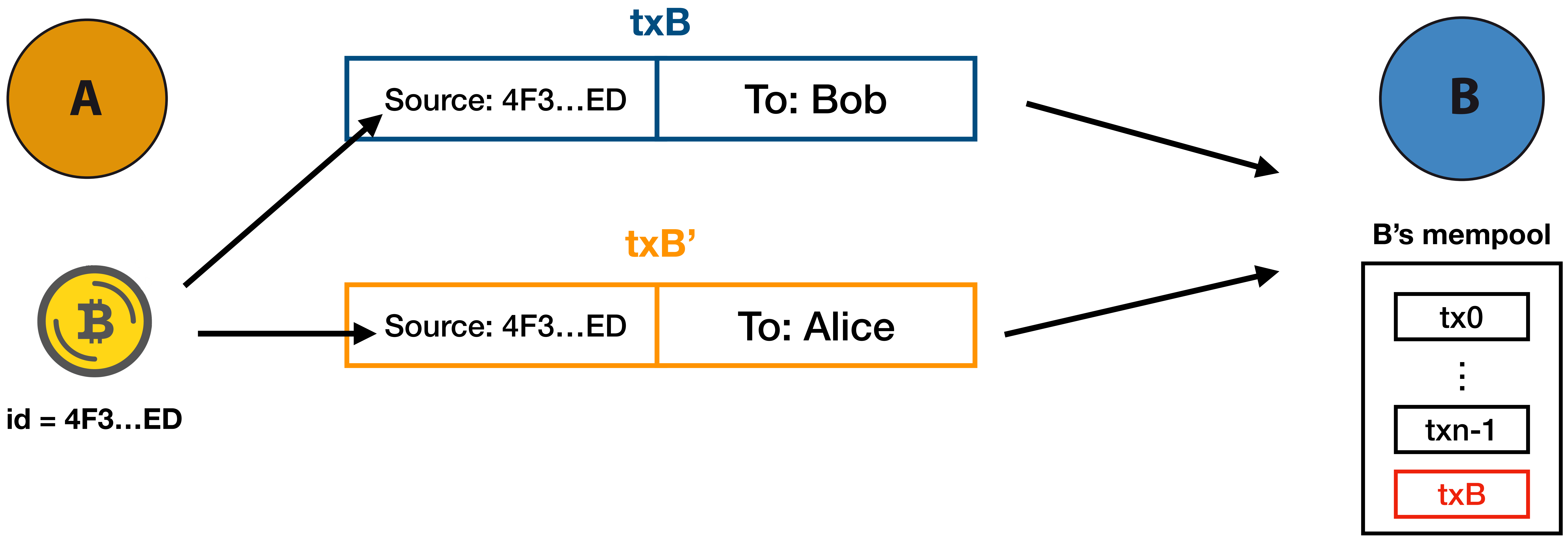
DOUBLE-SPENDING TRANSACTIONS



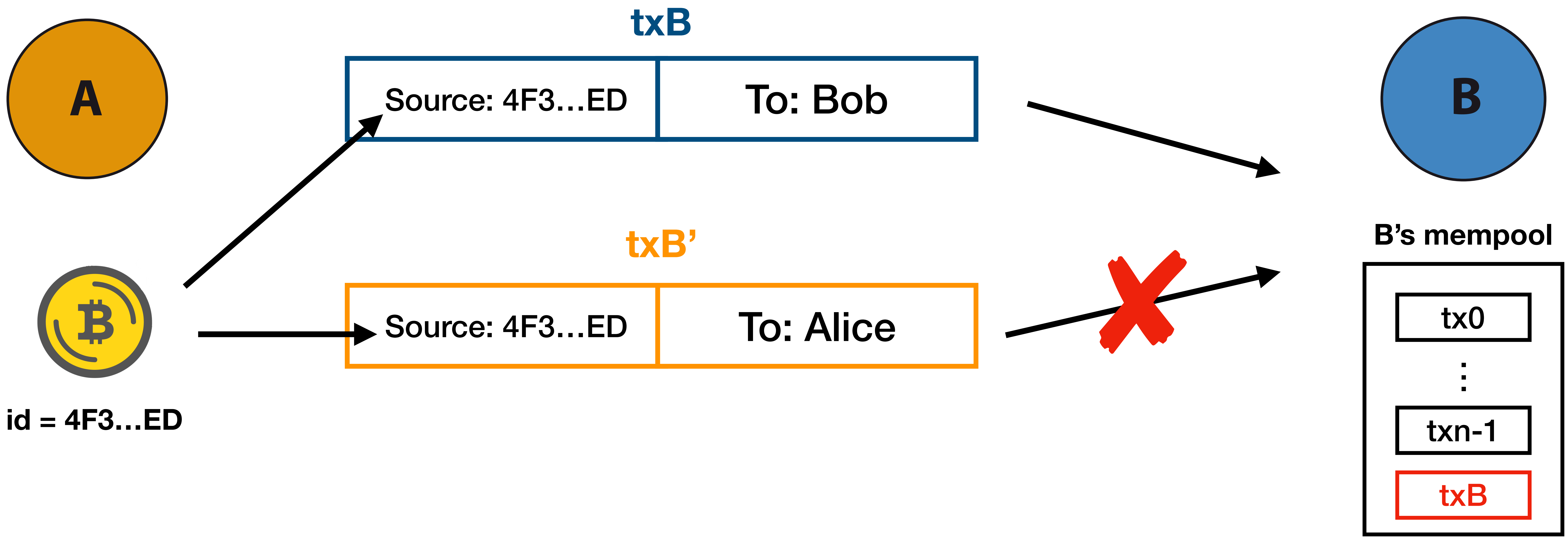
DOUBLE-SPENDING TRANSACTIONS



DOUBLE-SPENDING TRANSACTIONS



DOUBLE-SPENDING TRANSACTIONS



A BASIC TOPOLOGY INFERRING TECHNIQUE

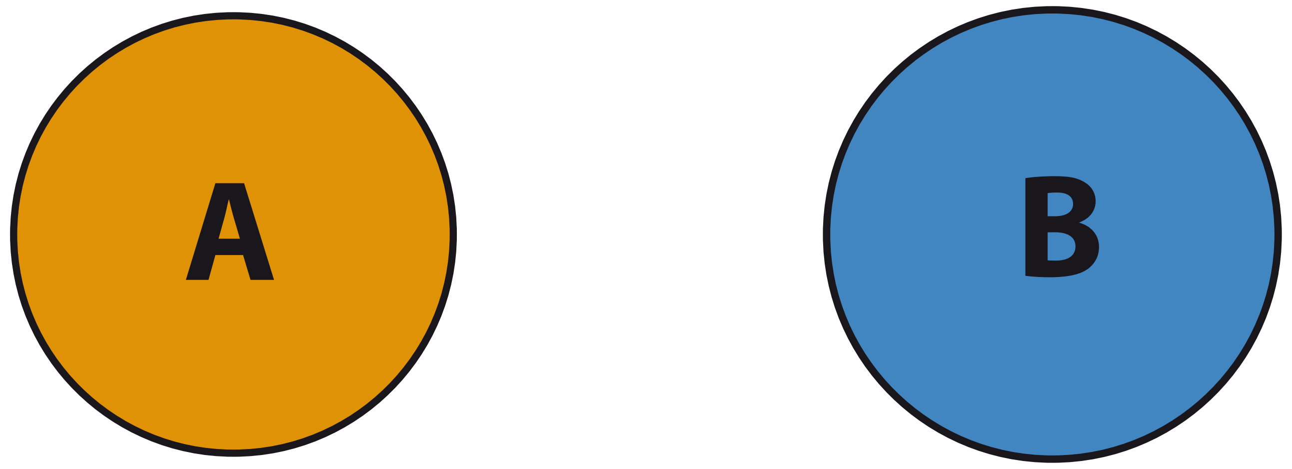
Two nodes

Three transactions

Observation tool (like coinscope)

A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes



Observation tool (like coinscope)

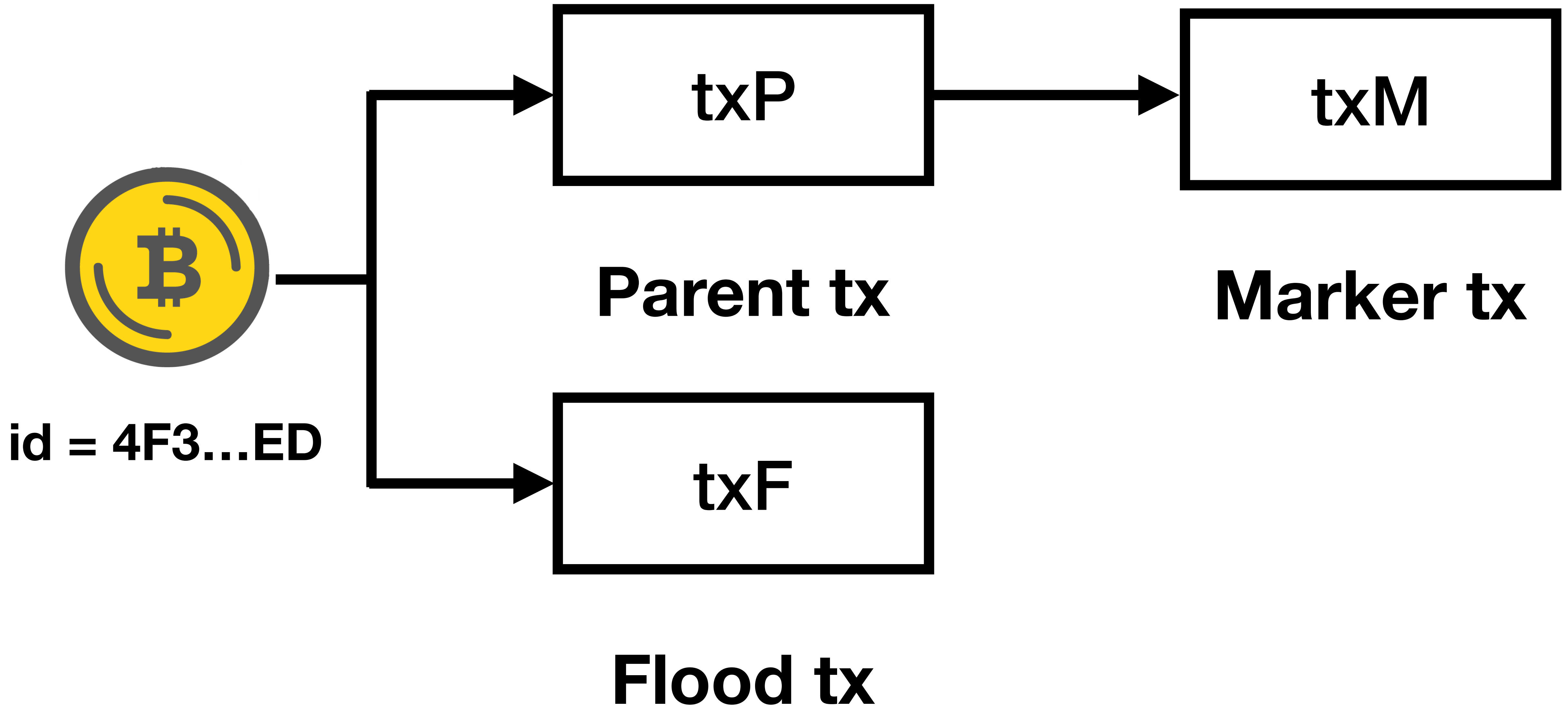
Three transactions

A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

**Observation tool
(like coinscope)**

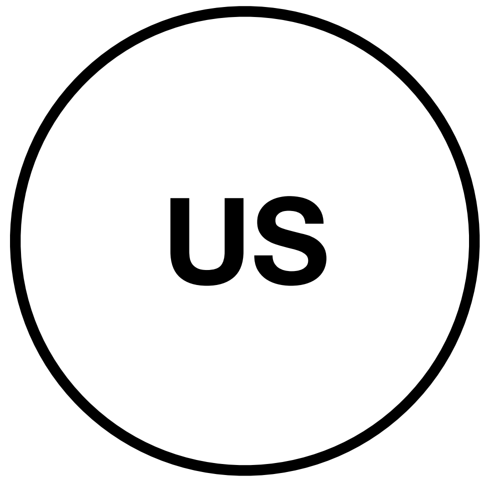
Three transactions



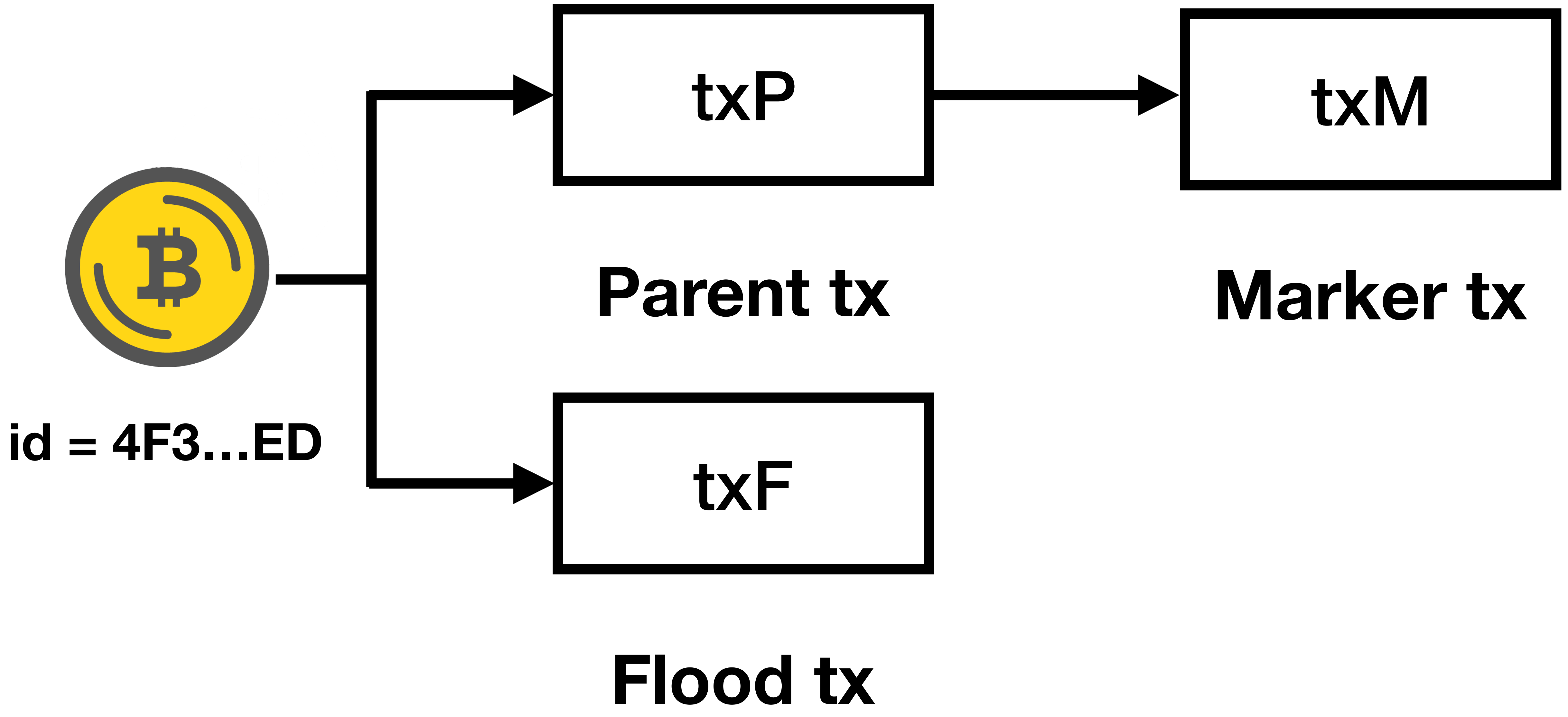
A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

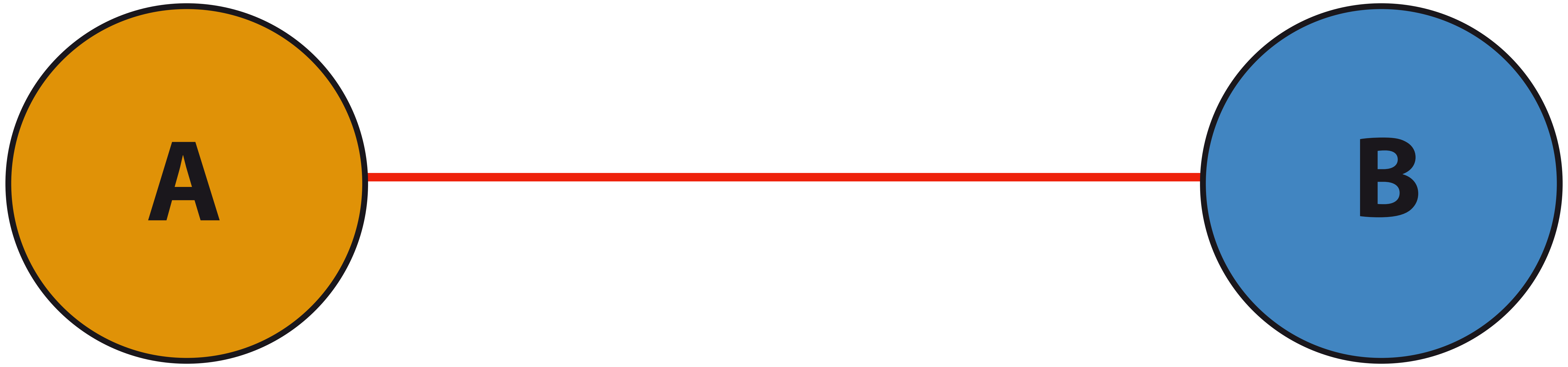
Observation tool (like coinscope)



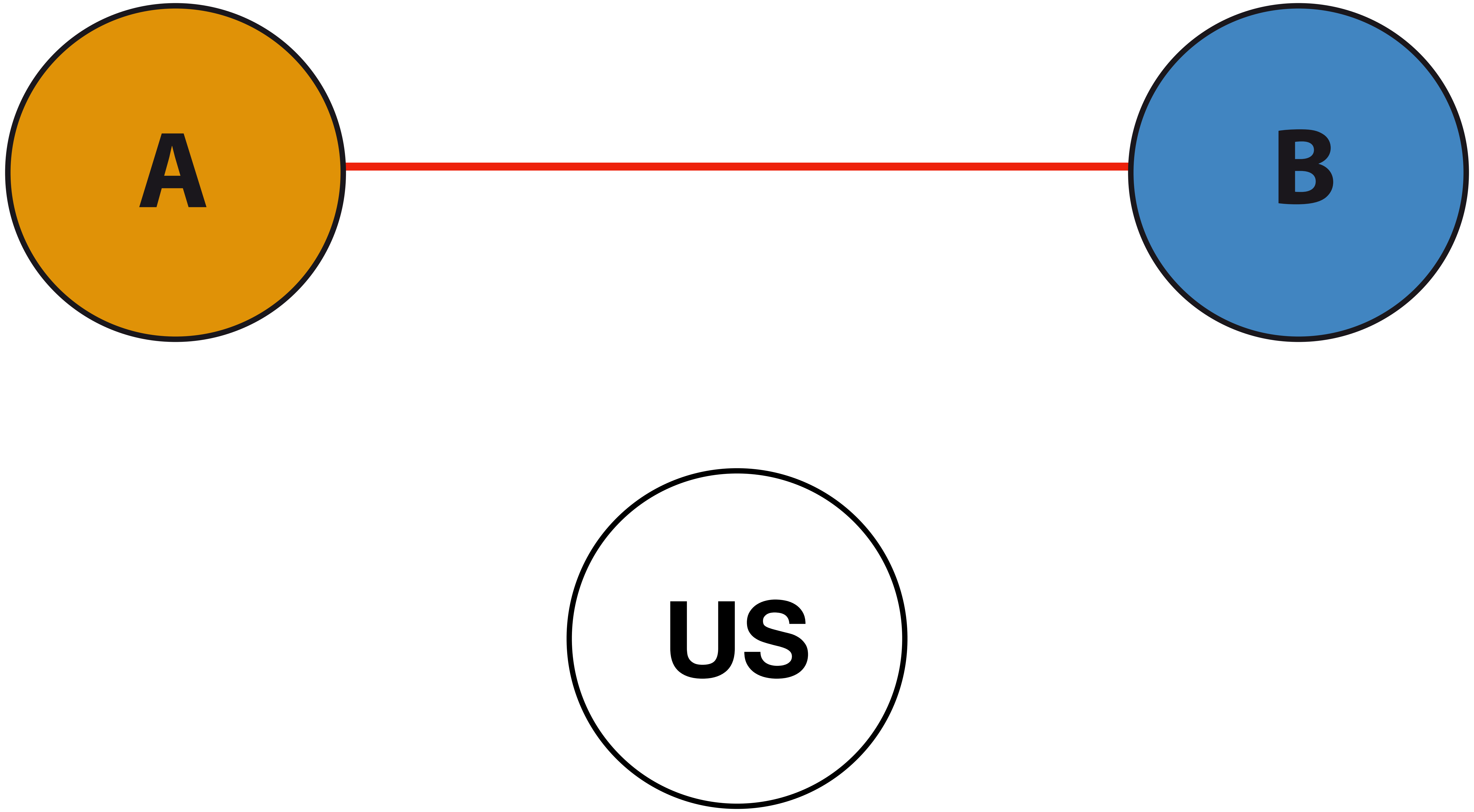
Three transactions



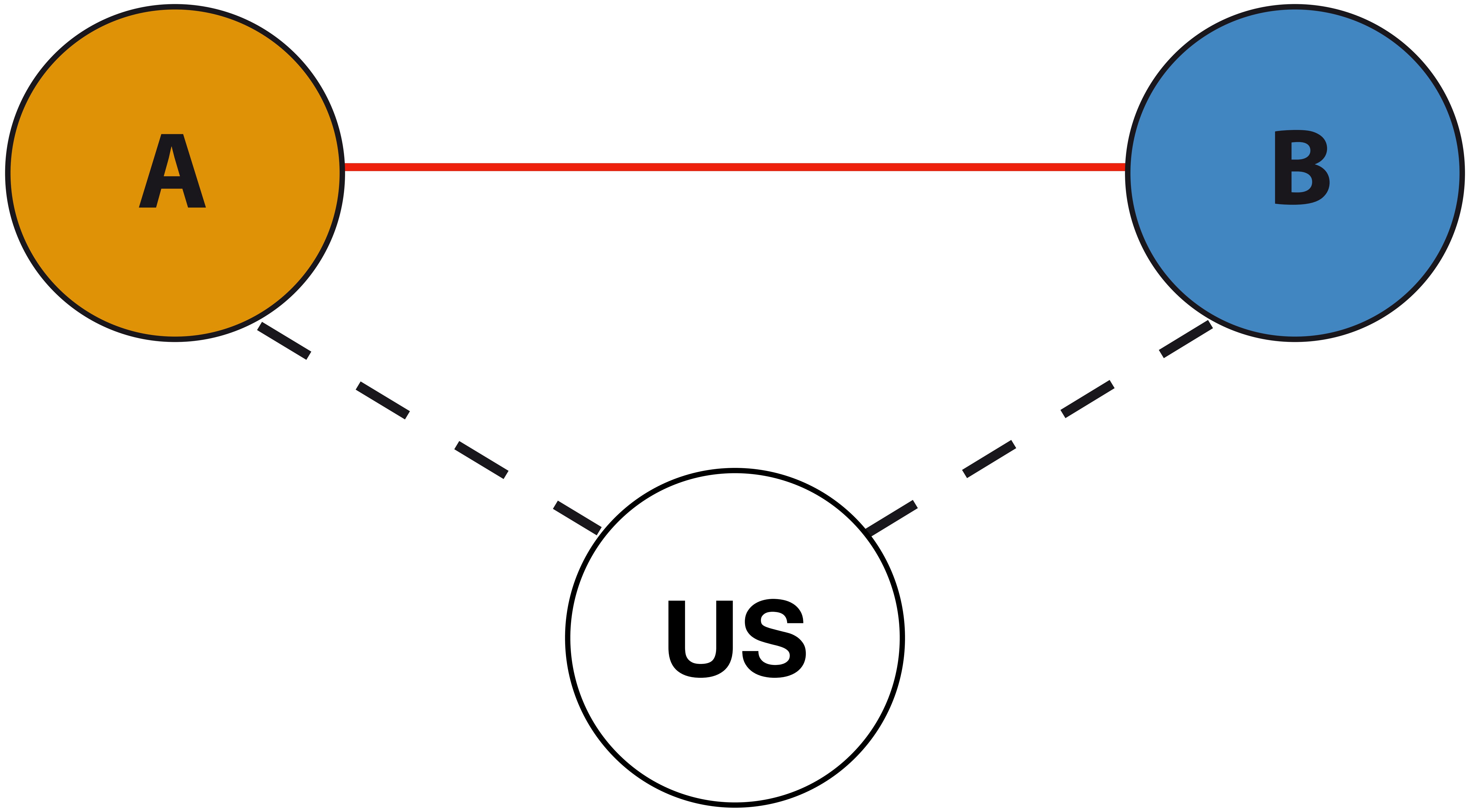
POSITIVE INFERRING TECHNIQUE



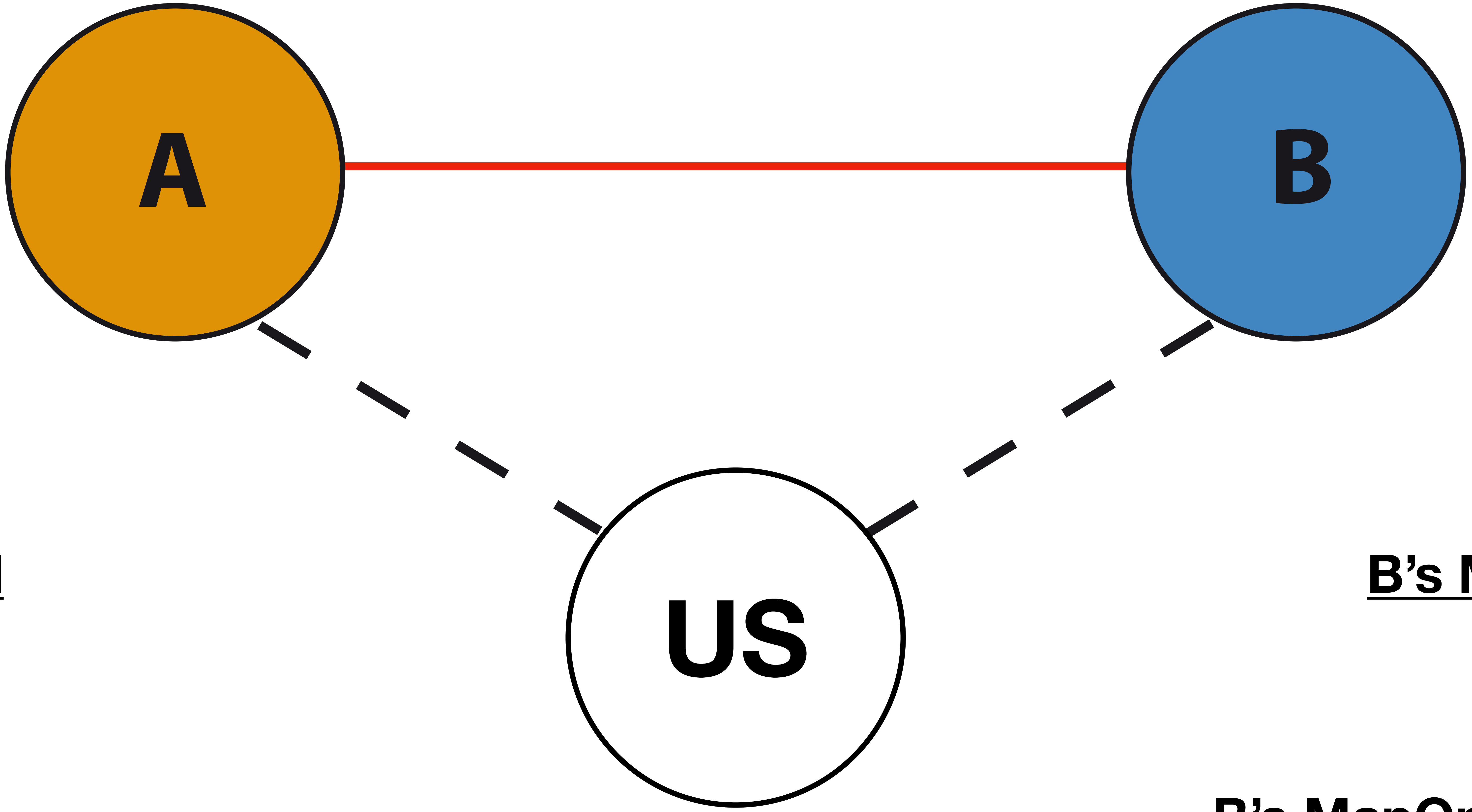
POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



A's Mempool

∅

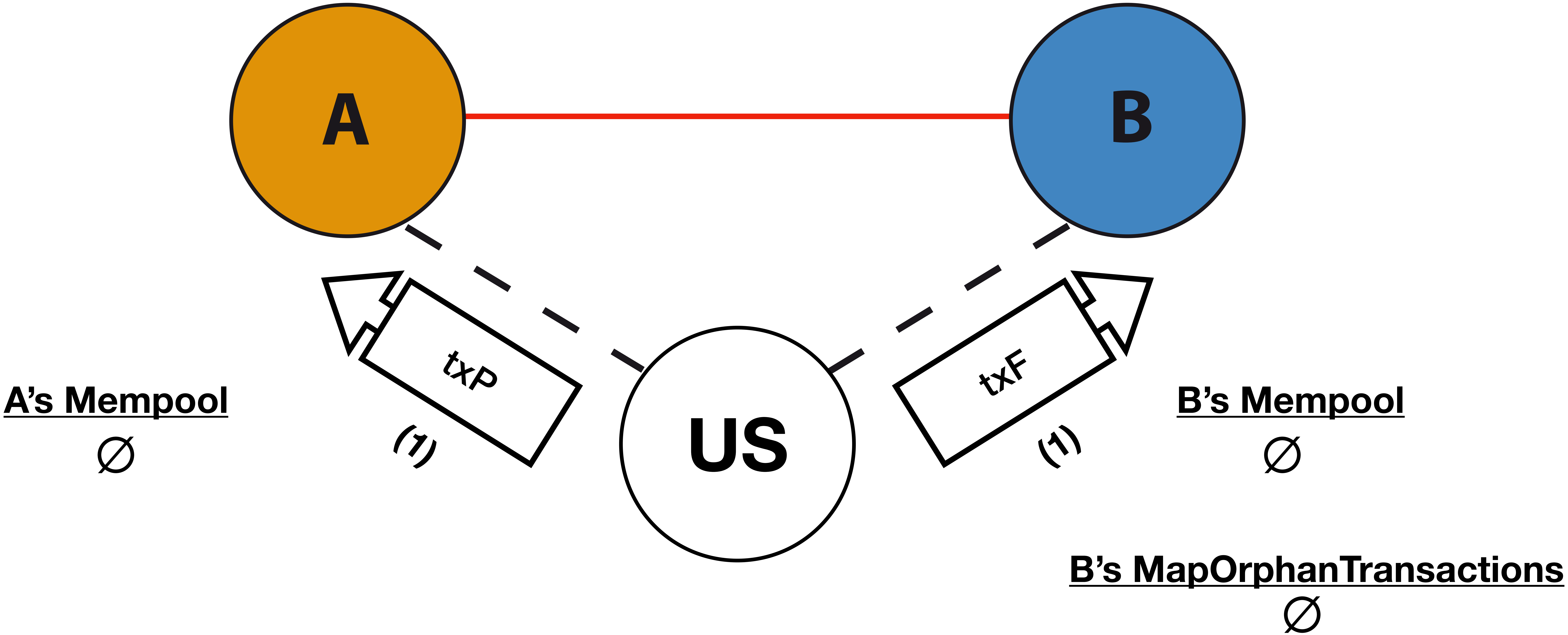
B's Mempool

∅

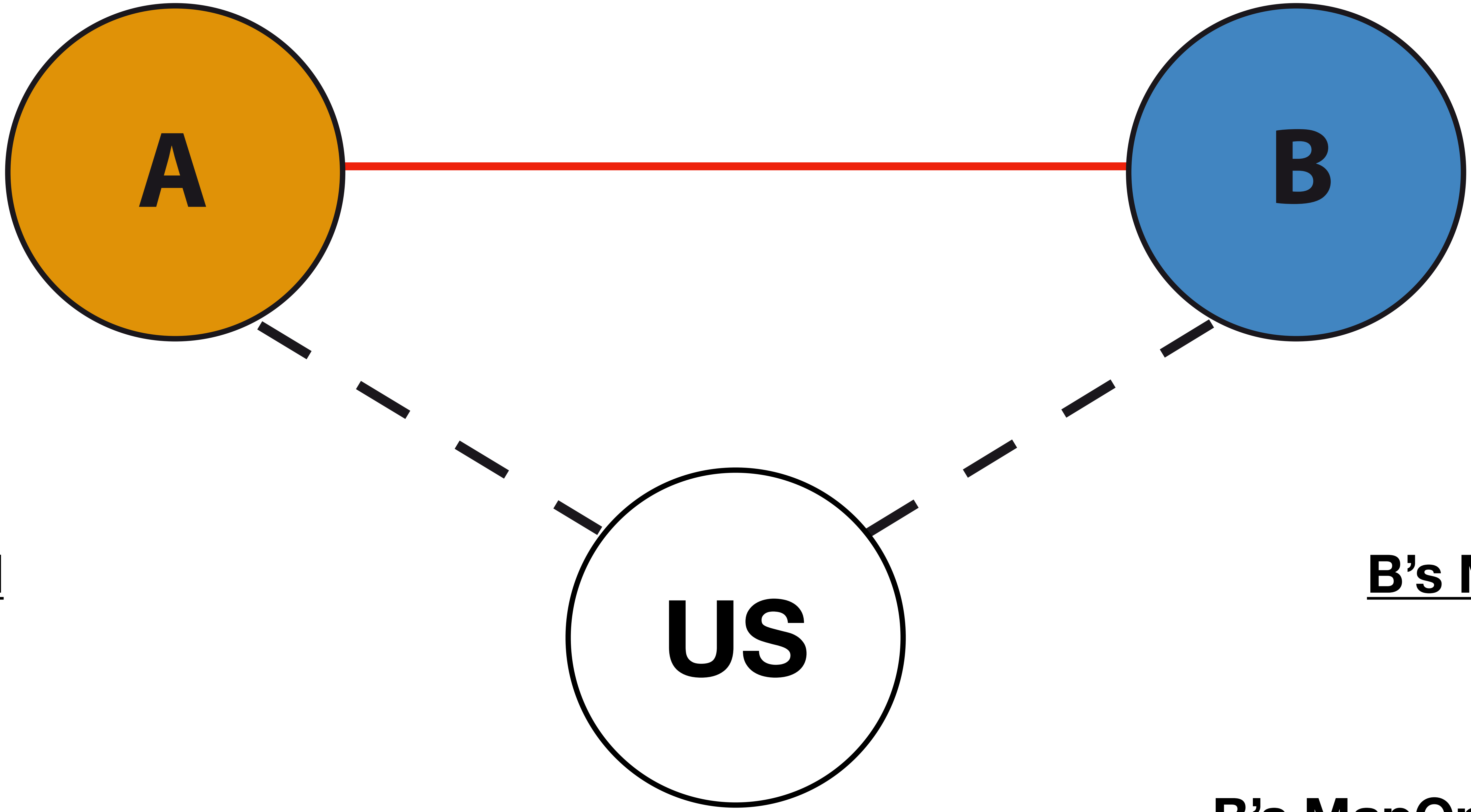
B's MapOrphanTransactions

∅

POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



A's Mempool

∅

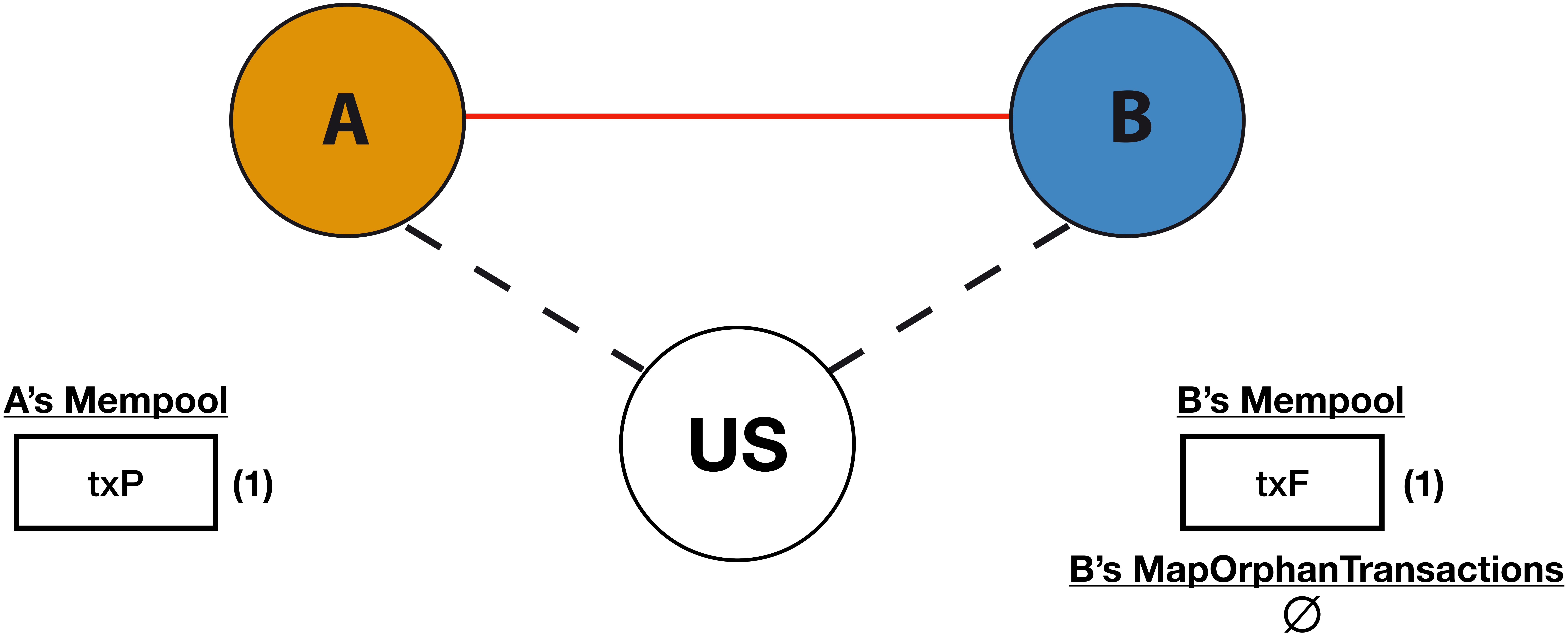
B's Mempool

∅

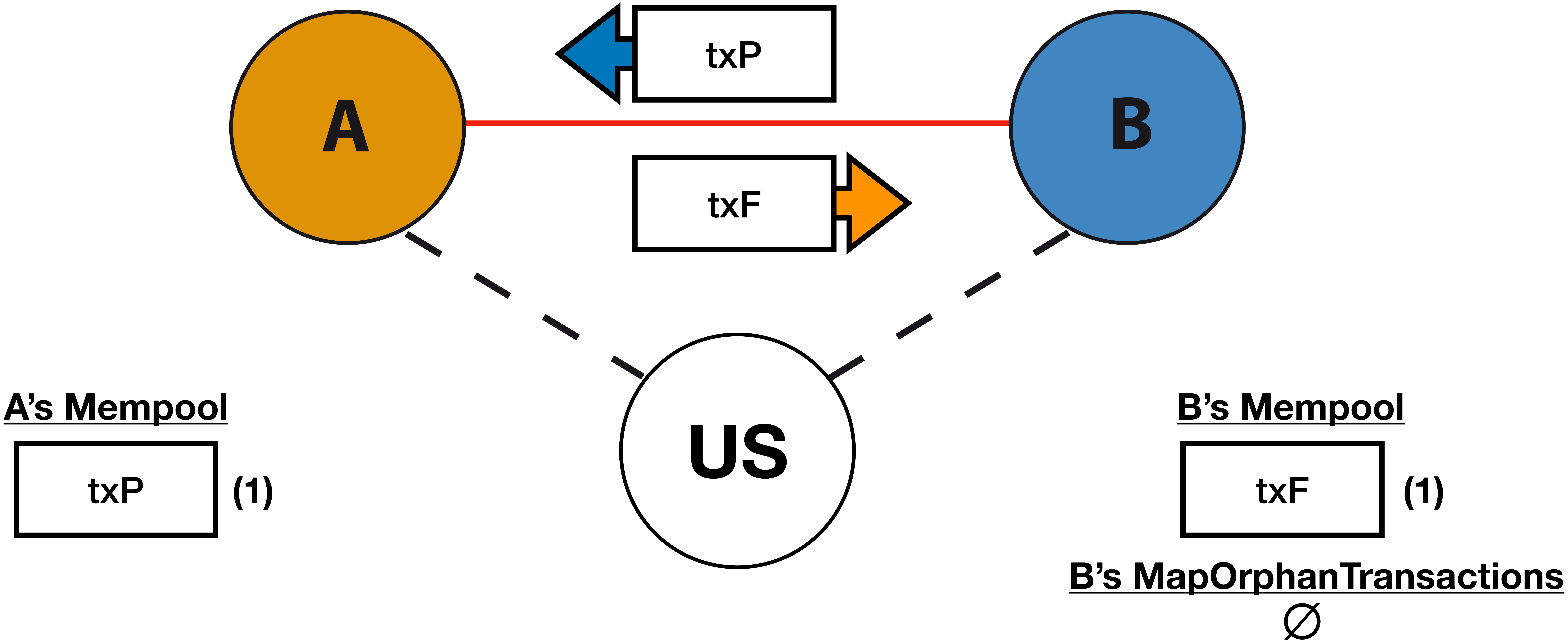
B's MapOrphanTransactions

∅

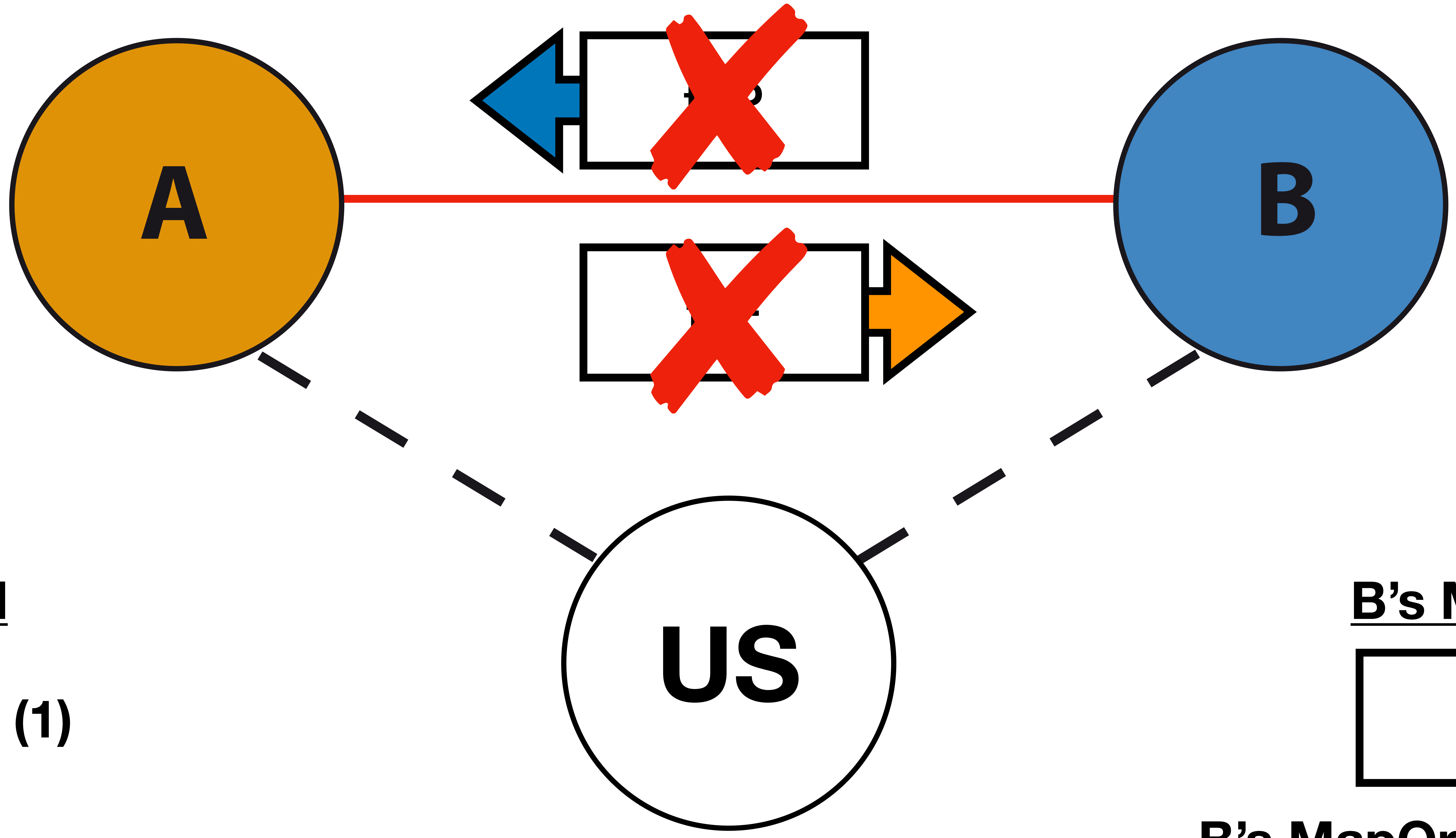
POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



POSITIVE INFERRING TECHNIQUE



A's Mempool

txP (1)

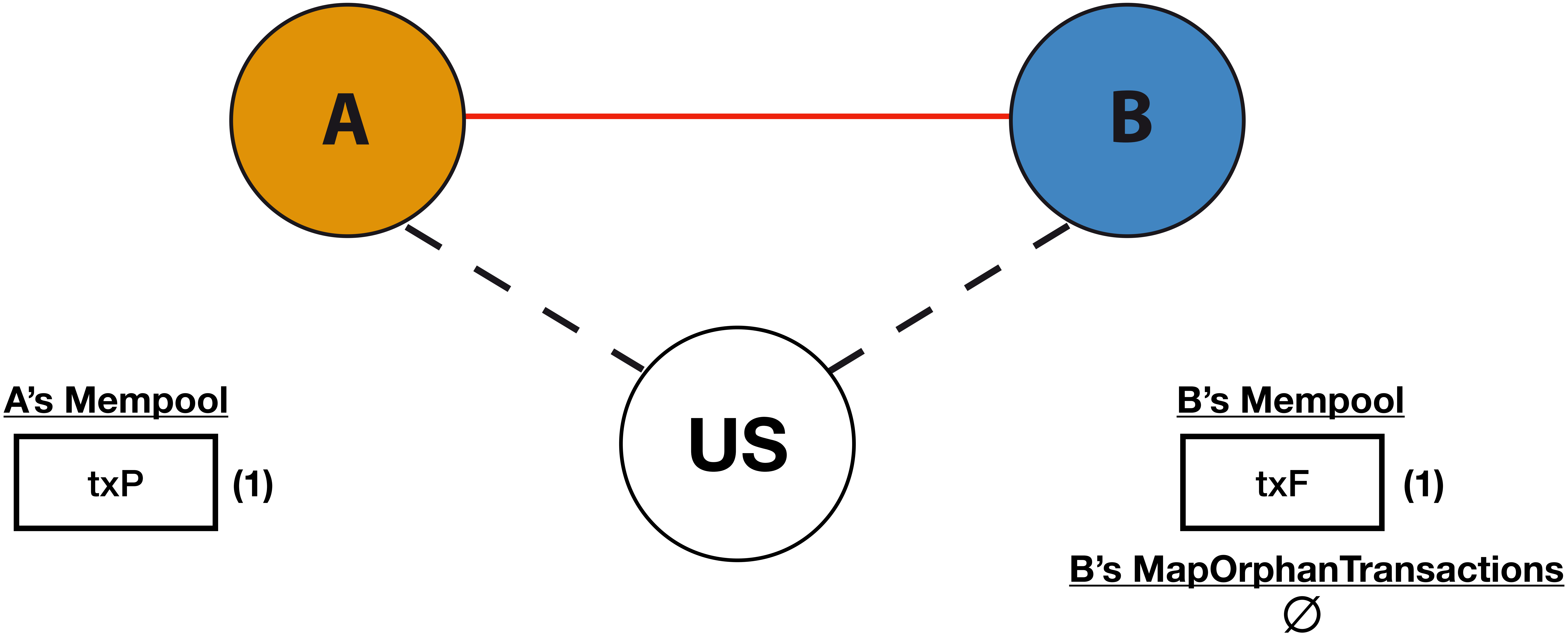
B's Mempool

txF (1)

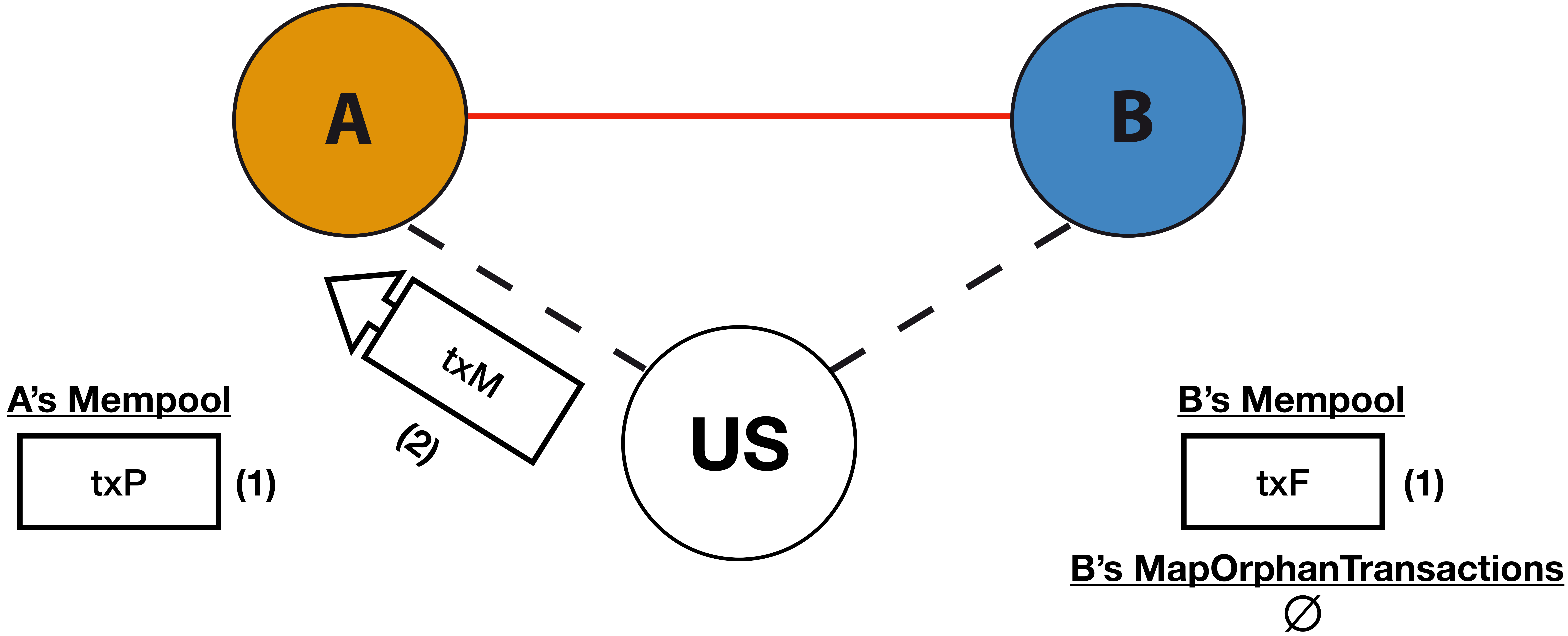
B's MapOrphanTransactions

∅

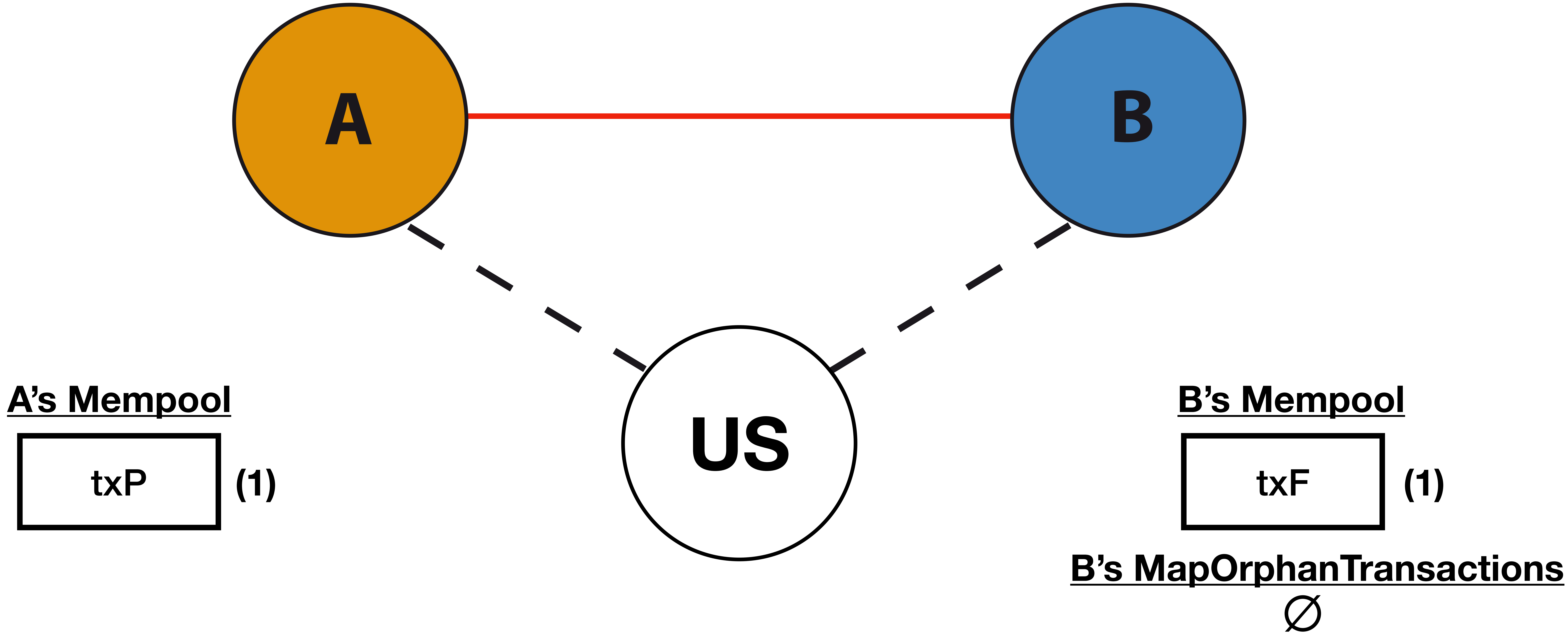
POSITIVE INFERRING TECHNIQUE



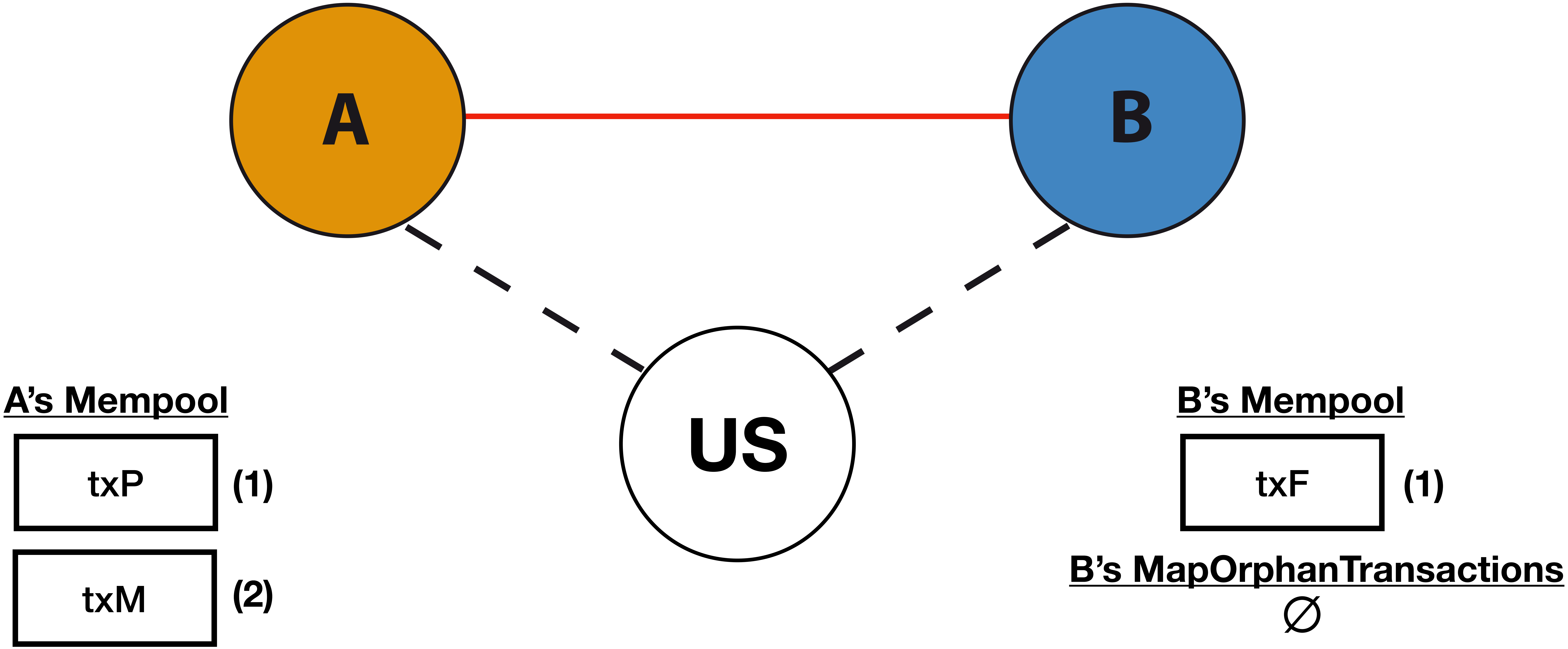
POSITIVE INFERRING TECHNIQUE



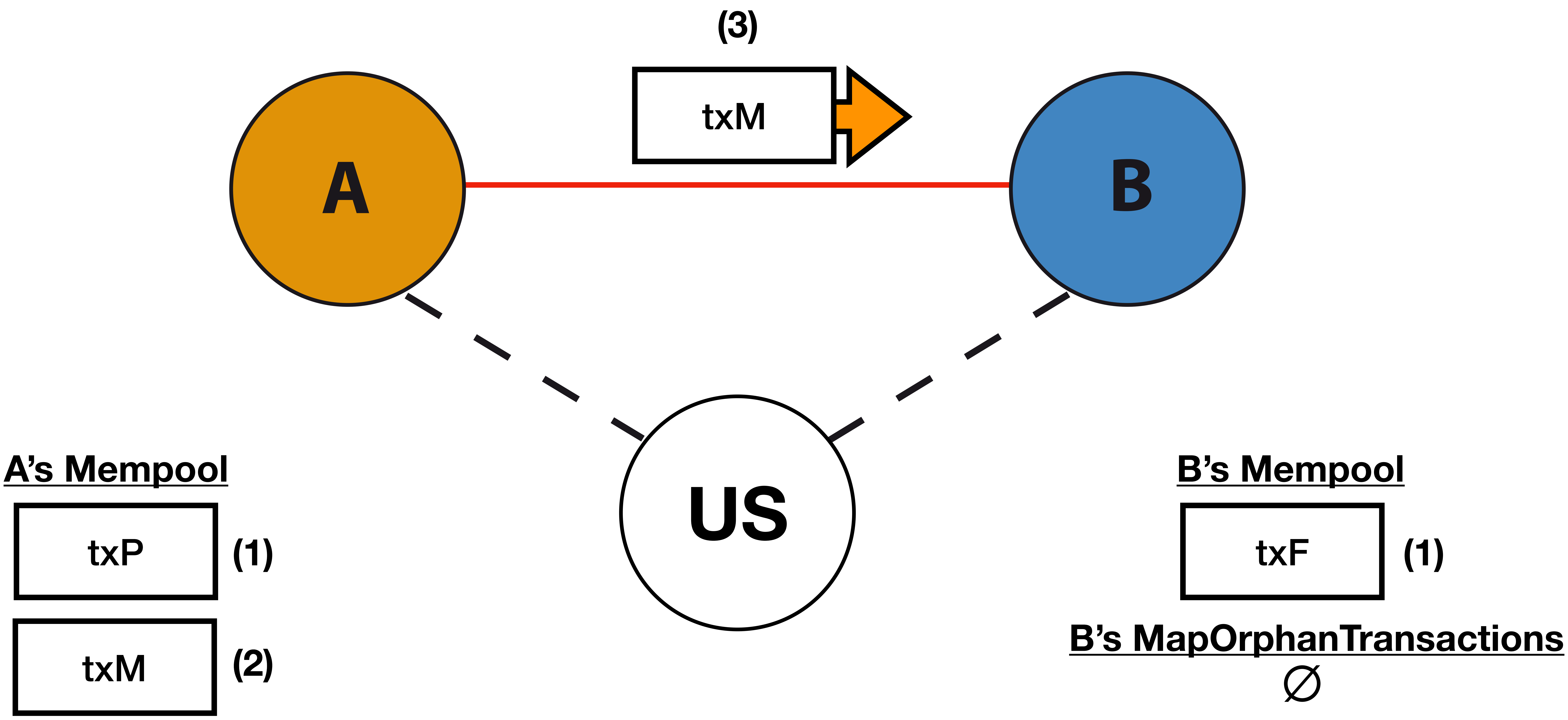
POSITIVE INFERRING TECHNIQUE



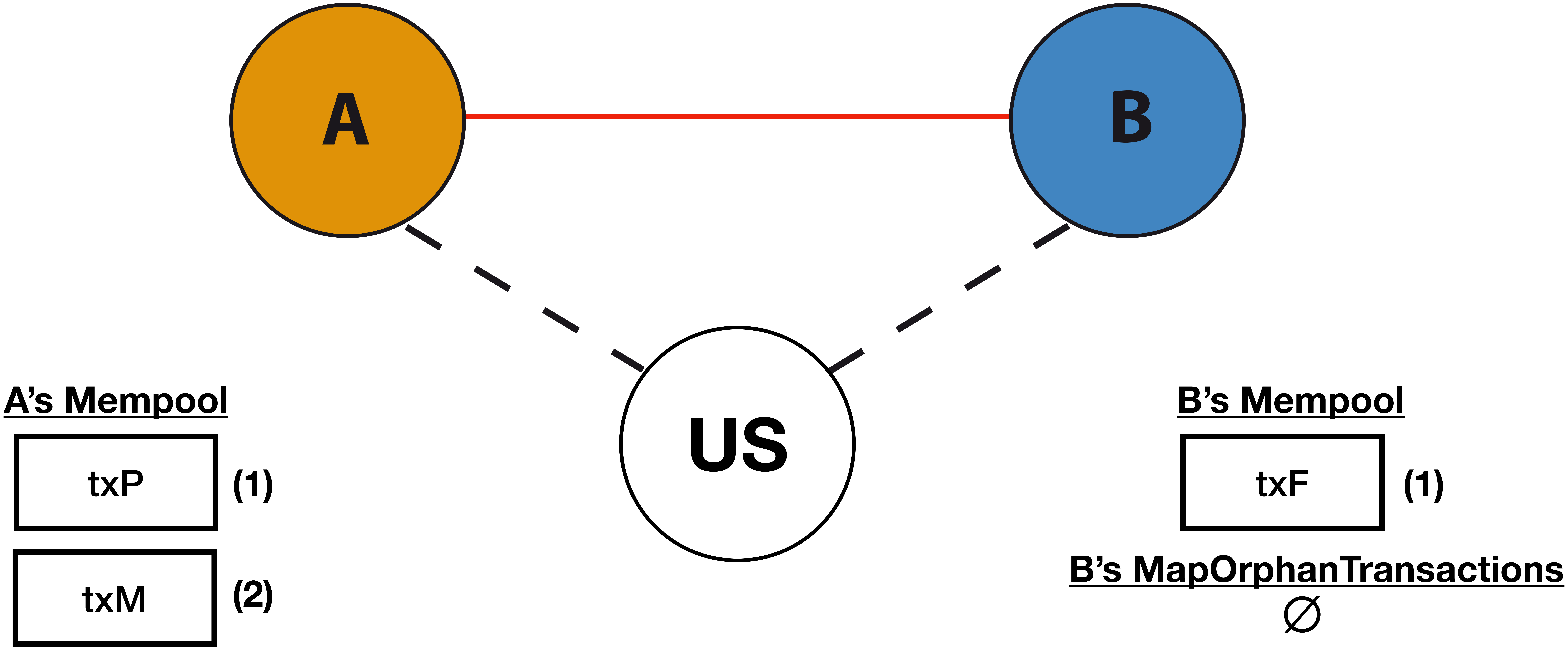
POSITIVE INFERRING TECHNIQUE



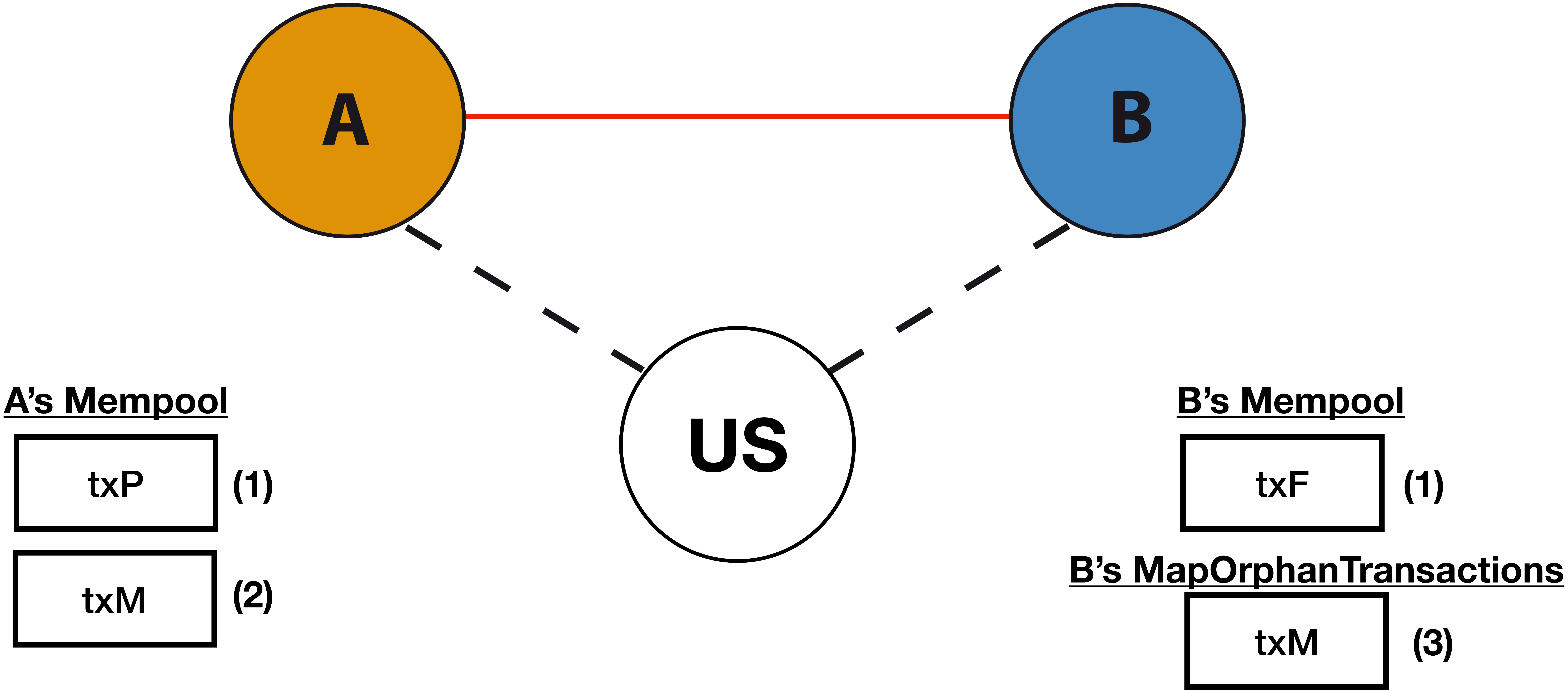
POSITIVE INFERRING TECHNIQUE



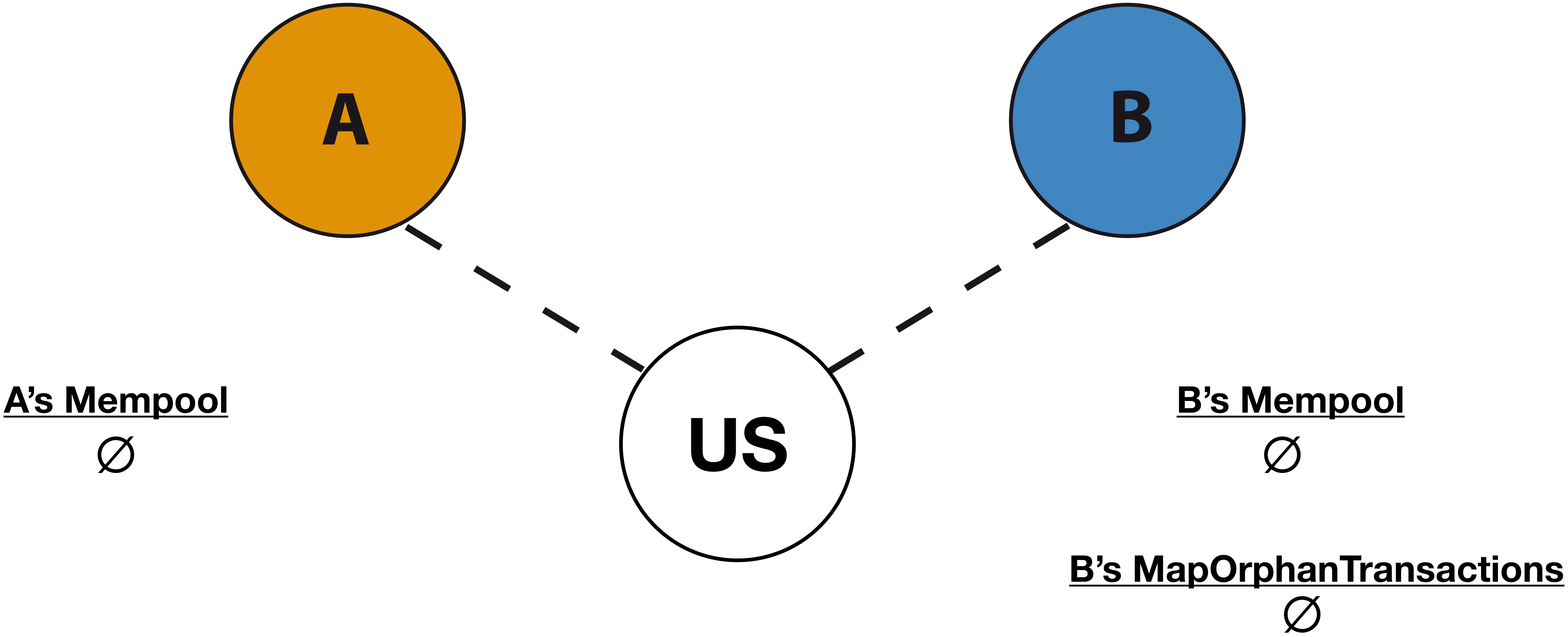
POSITIVE INFERRING TECHNIQUE



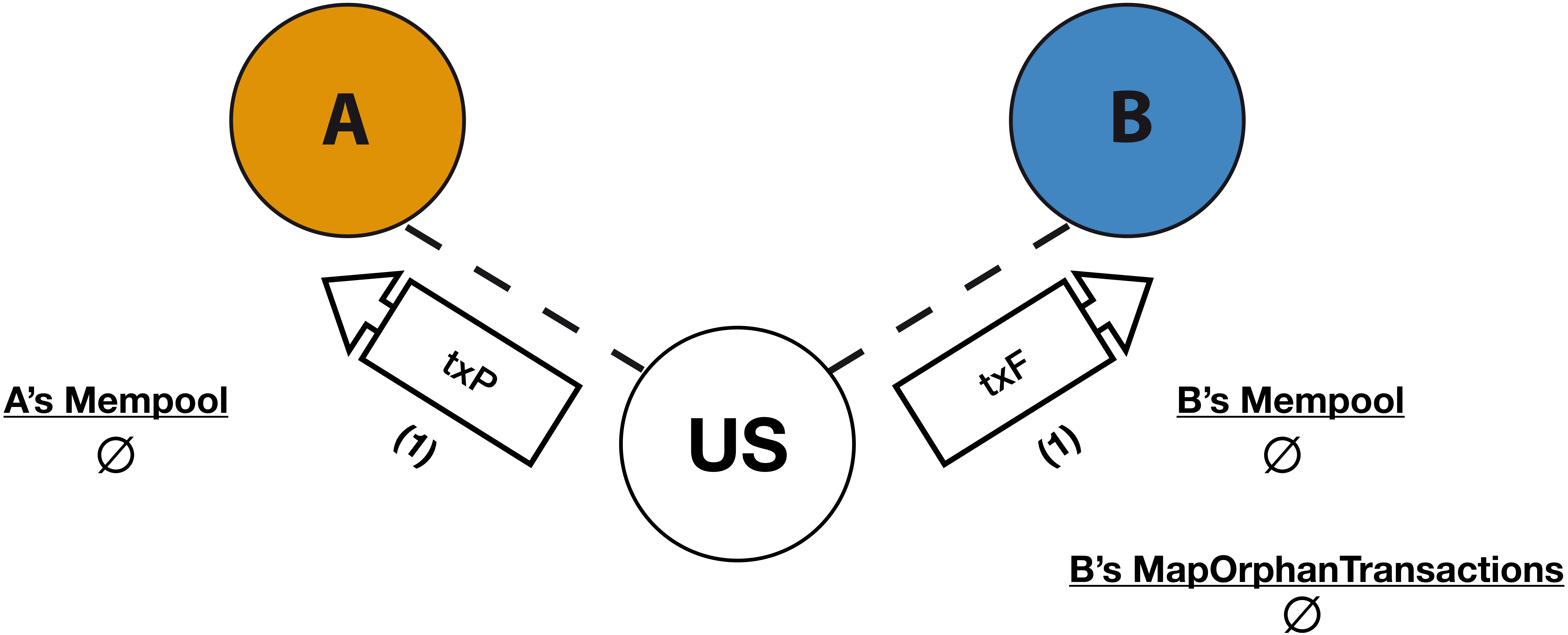
POSITIVE INFERRING TECHNIQUE



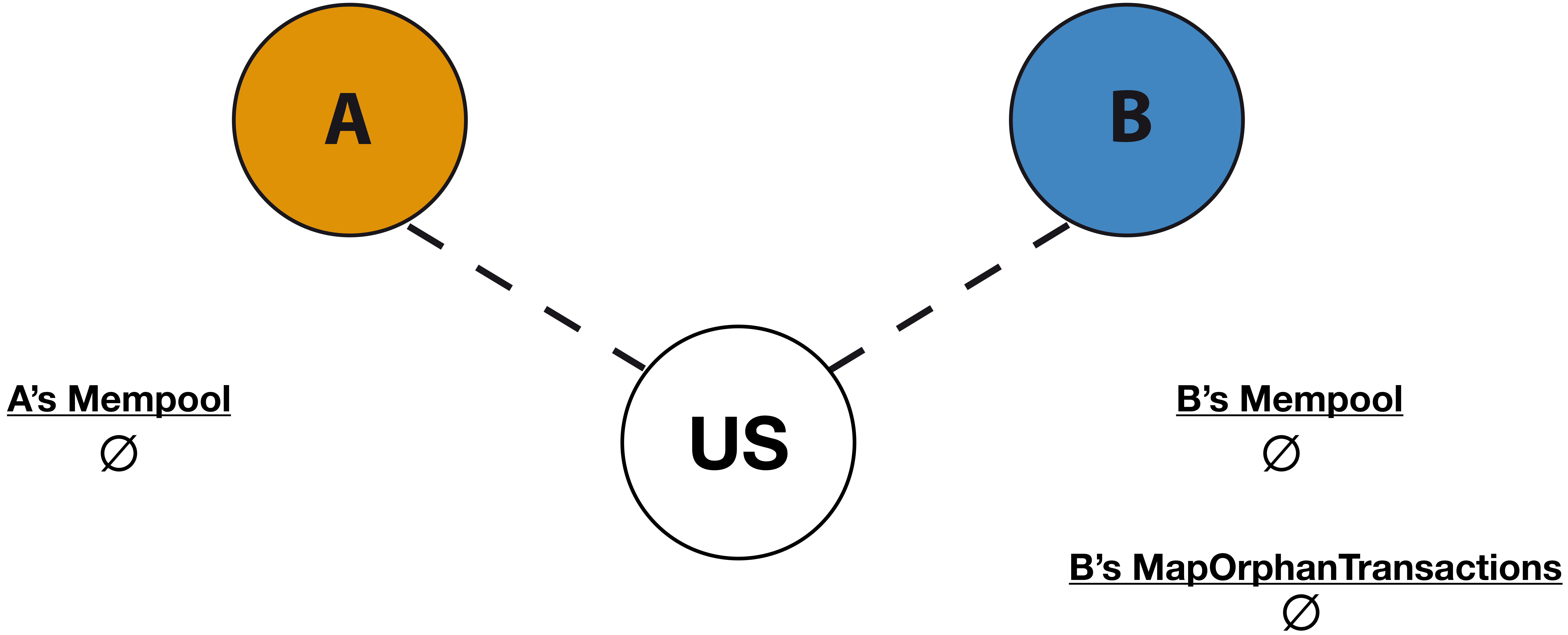
NEGATIVE INFERRING TECHNIQUE



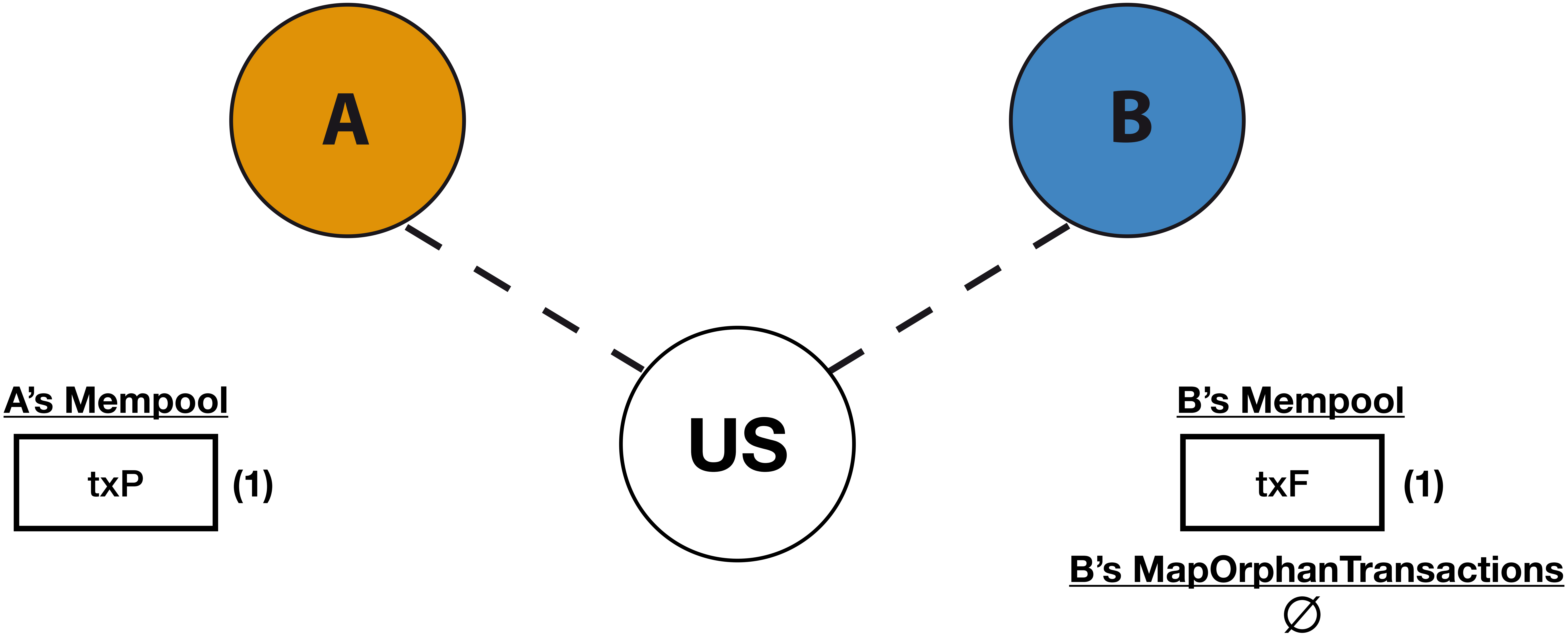
NEGATIVE INFERRING TECHNIQUE



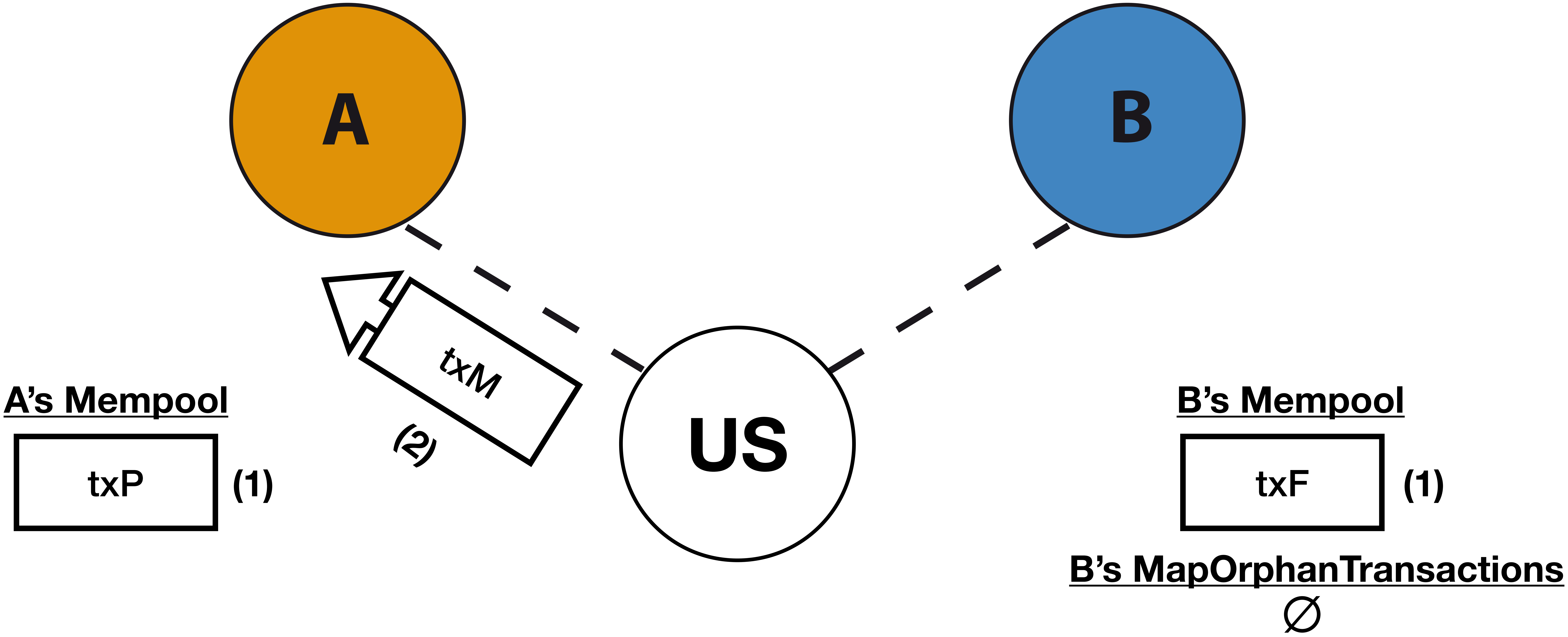
NEGATIVE INFERRING TECHNIQUE



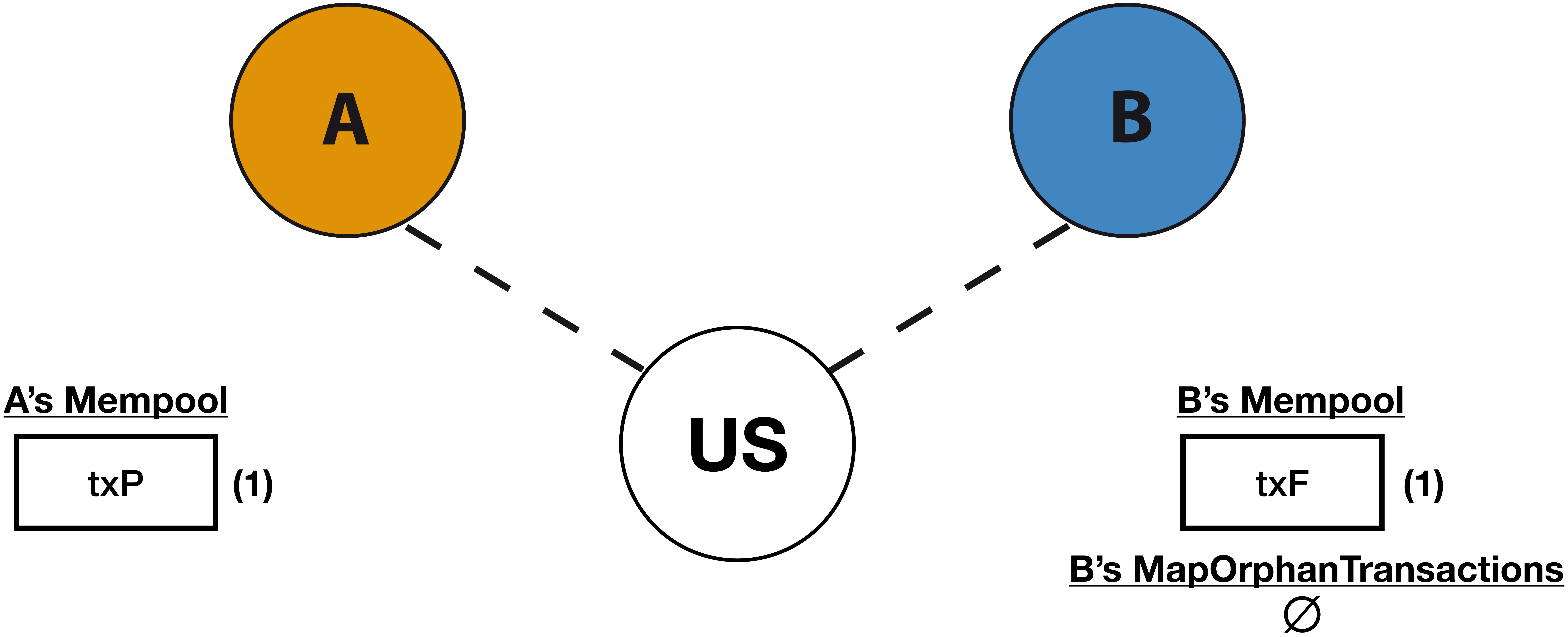
NEGATIVE INFERRING TECHNIQUE



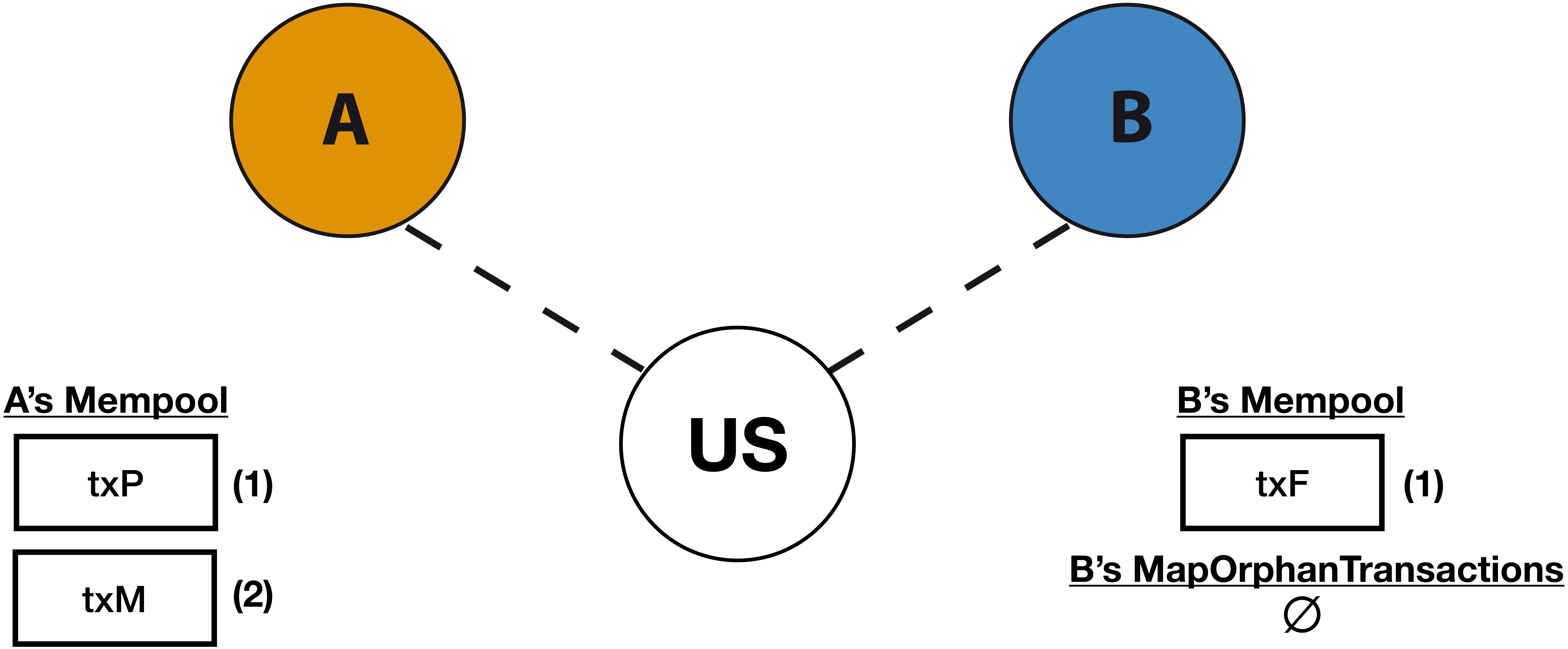
NEGATIVE INFERRING TECHNIQUE



NEGATIVE INFERRING TECHNIQUE

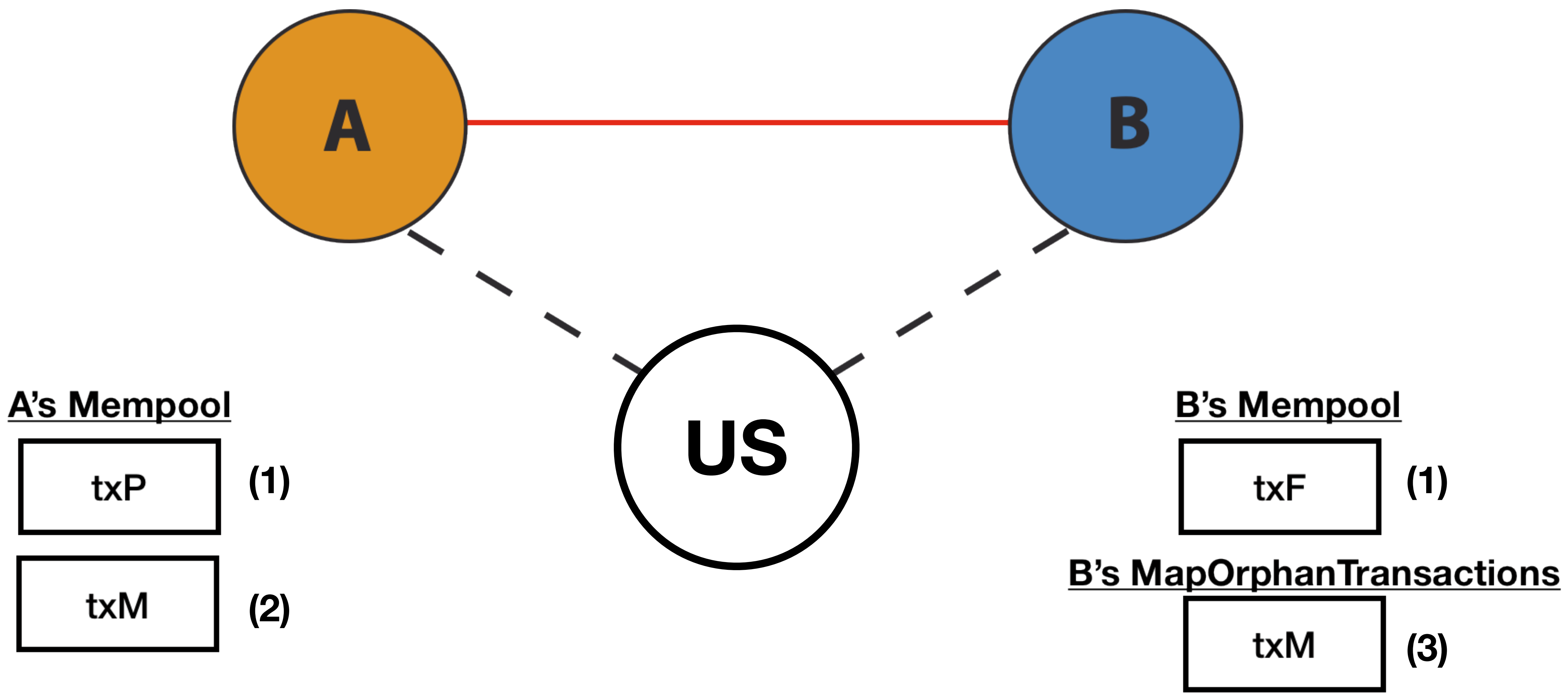


NEGATIVE INFERRING TECHNIQUE

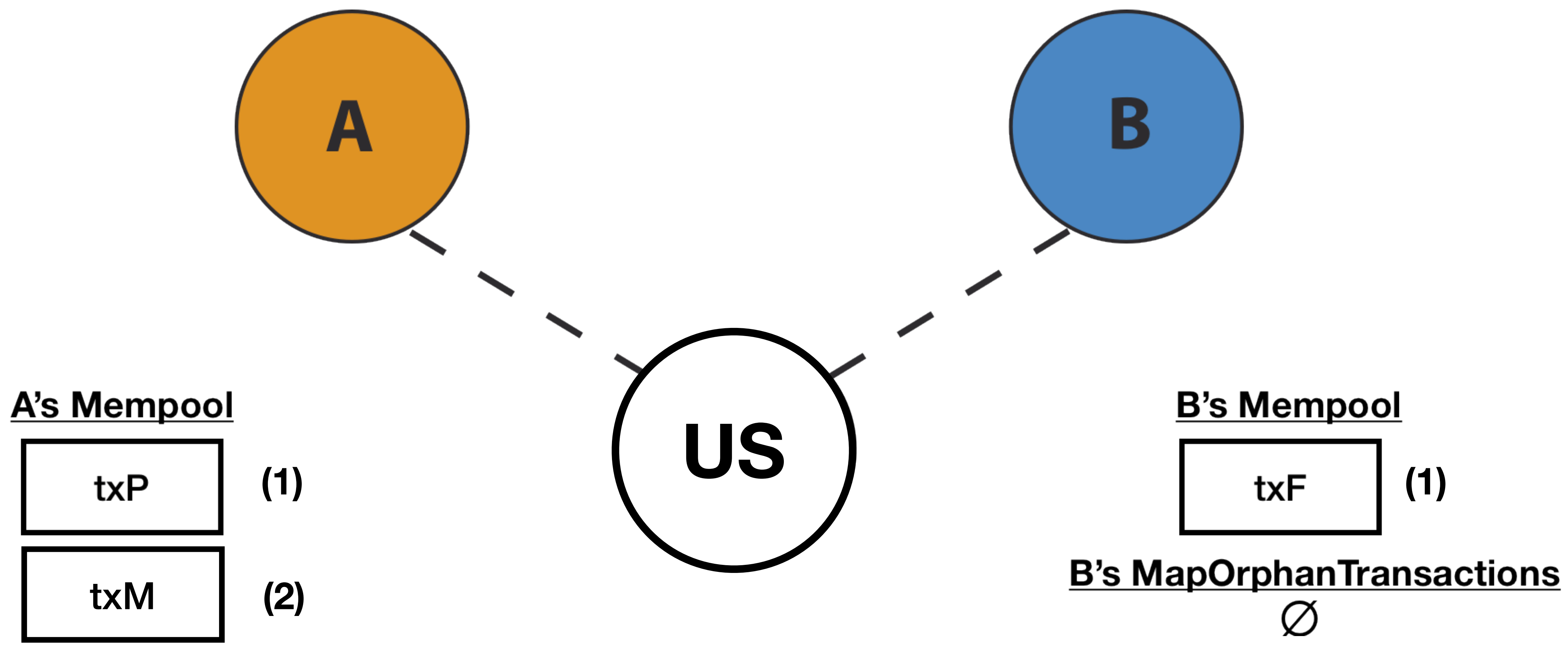


A BASIC TOPOLOGY INFERRING TECHNIQUE

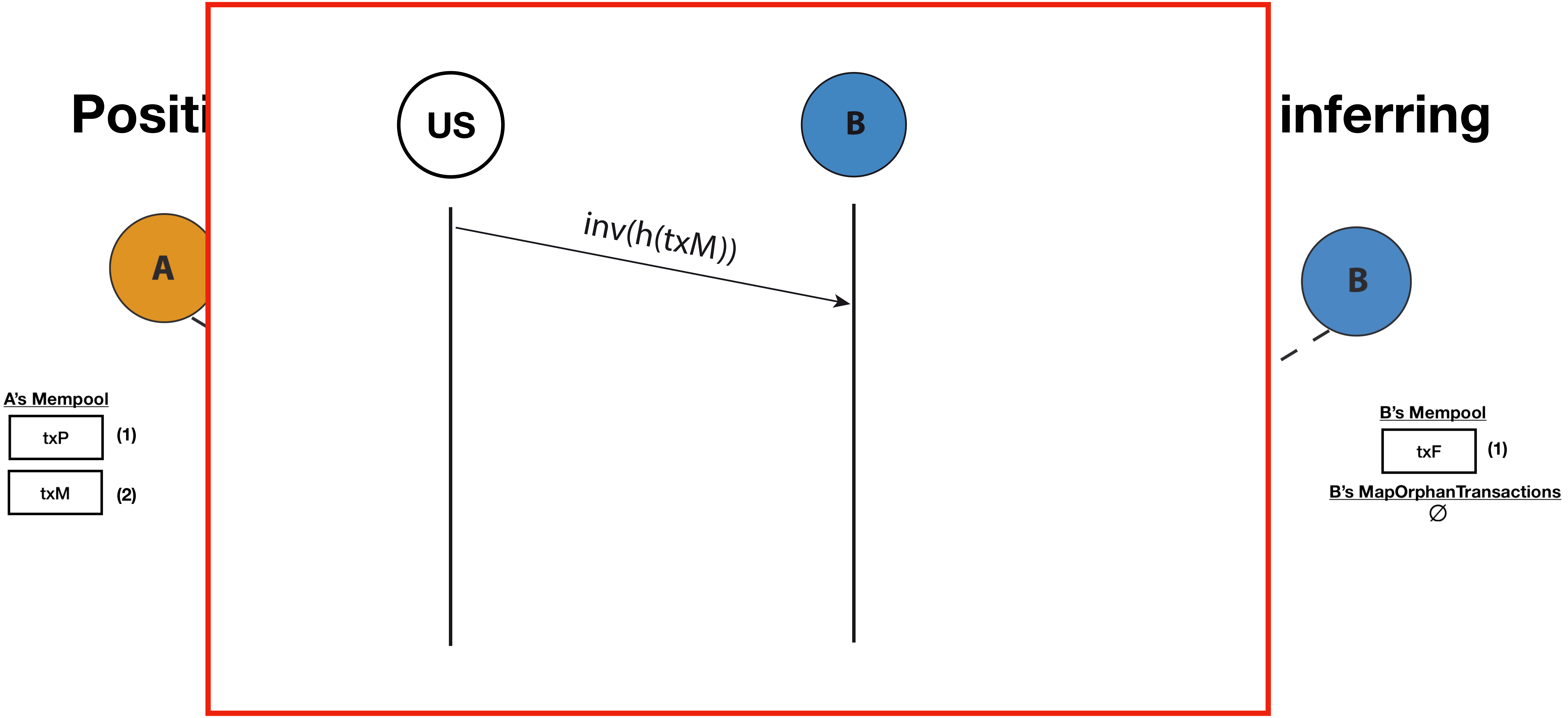
Positive edge inferring



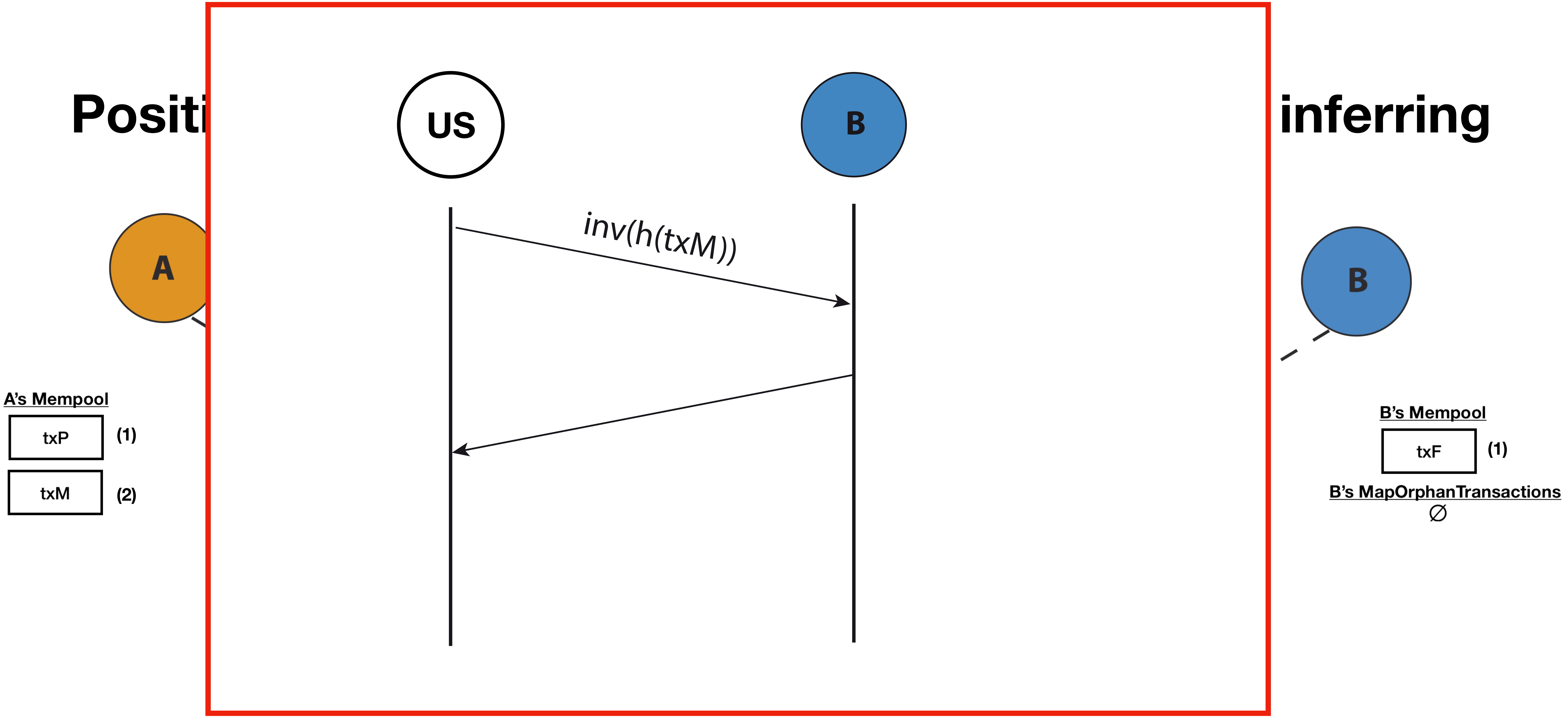
Negative edge inferring



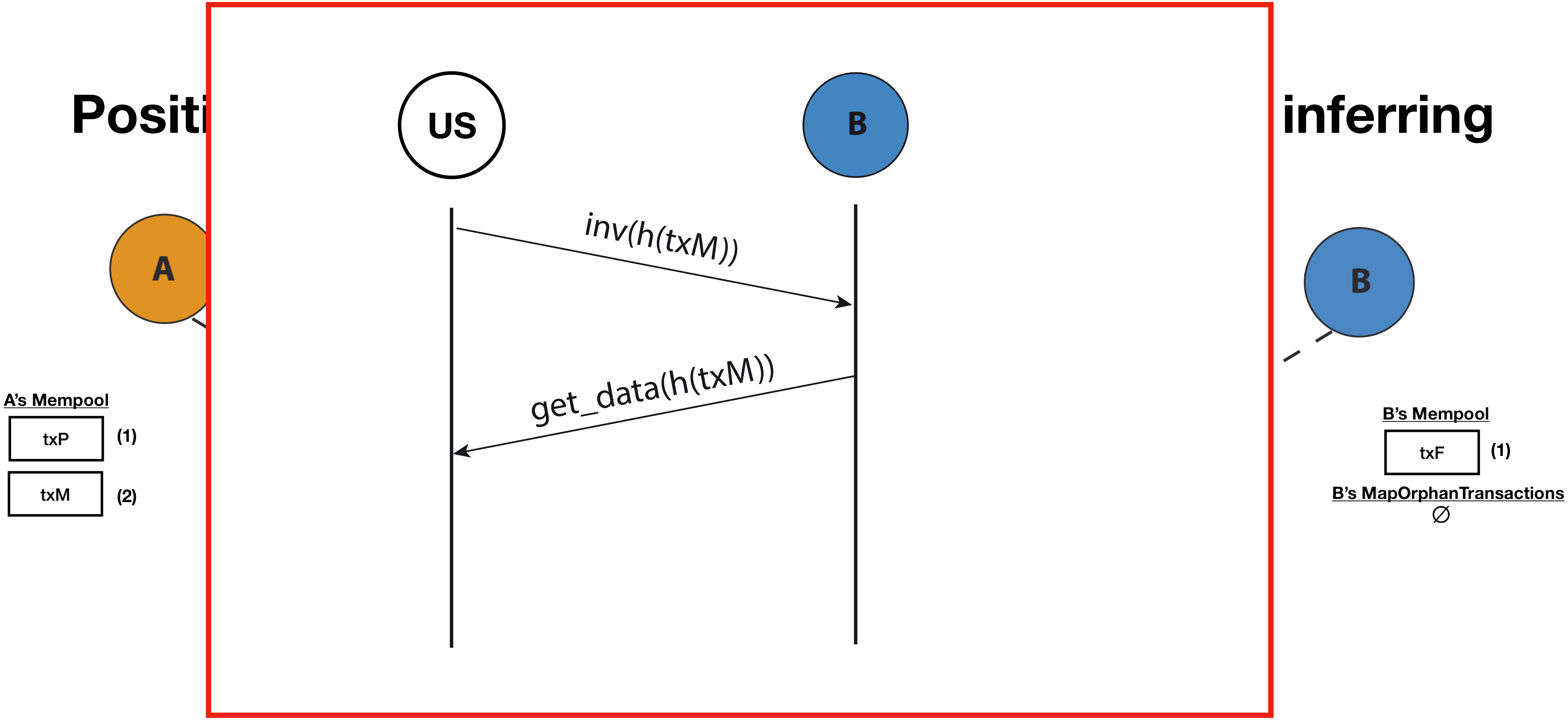
A BASIC TOPOLOGY INFERRING TECHNIQUE



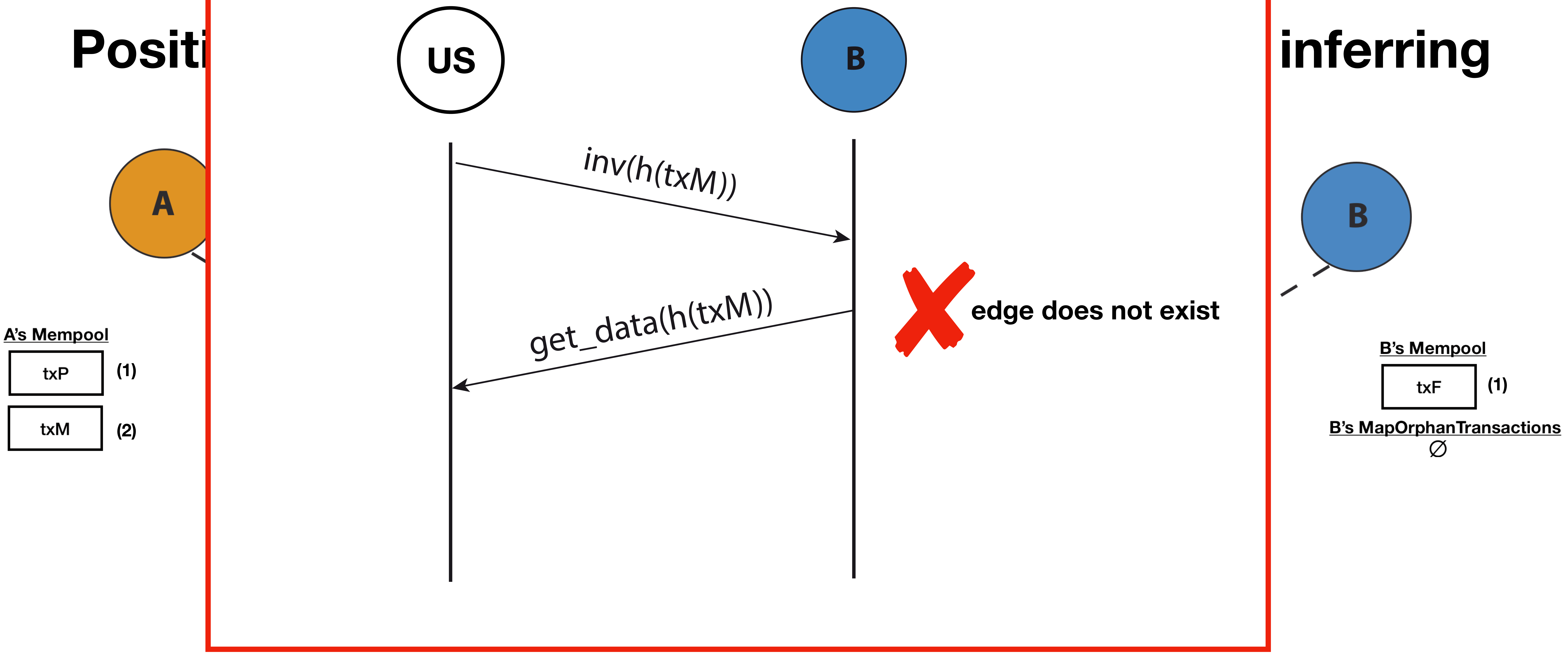
A BASIC TOPOLOGY INFERRING TECHNIQUE



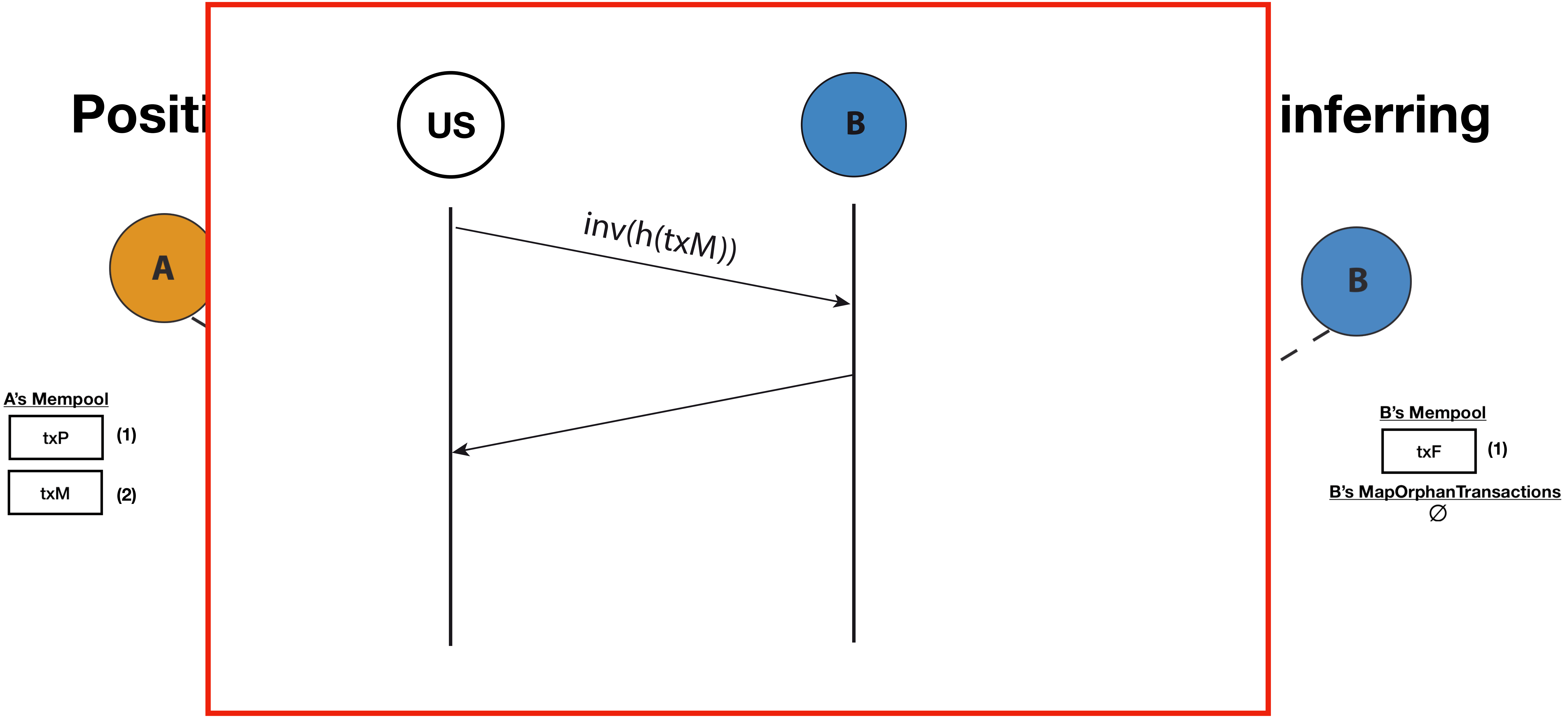
A BASIC TOPOLOGY INFERRING TECHNIQUE



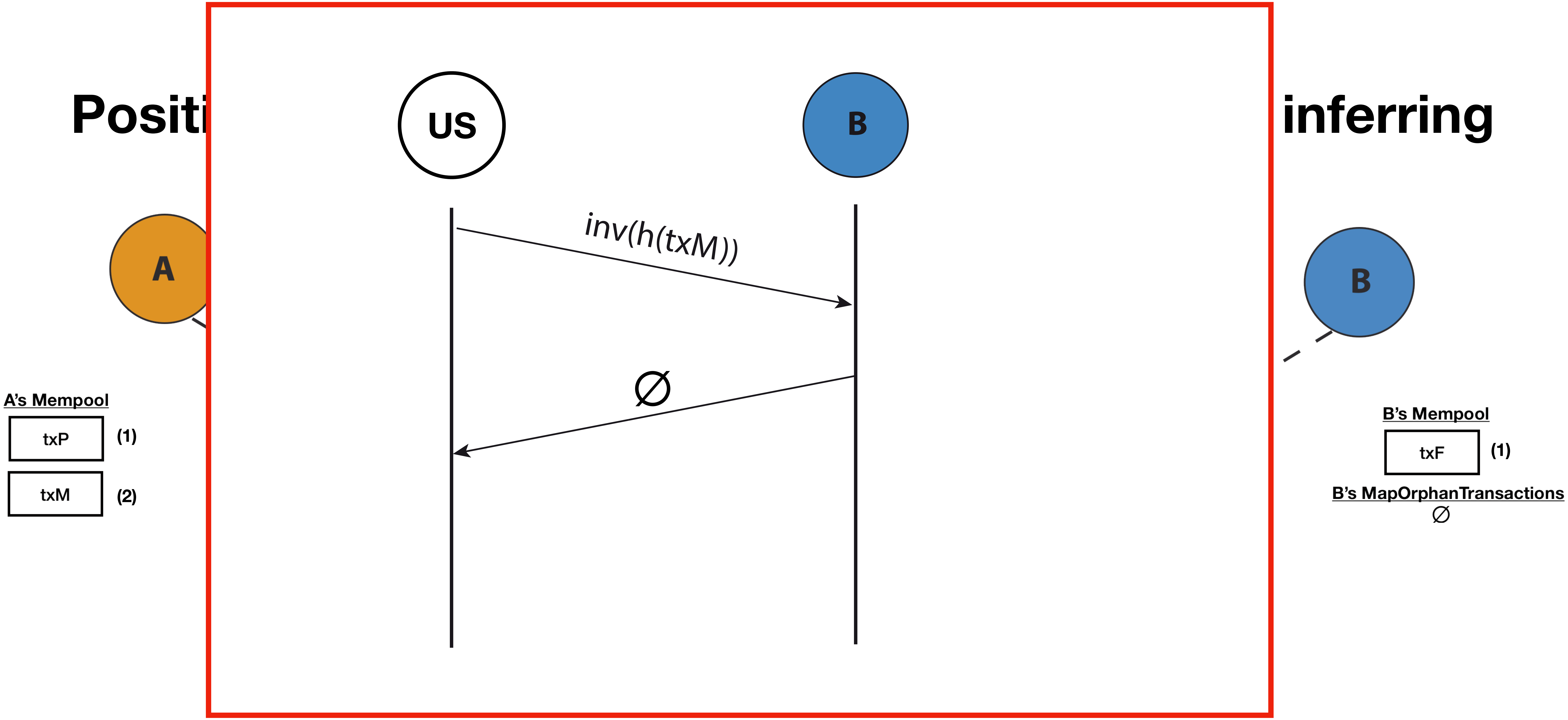
A BASIC TOPOLOGY INFERRING TECHNIQUE



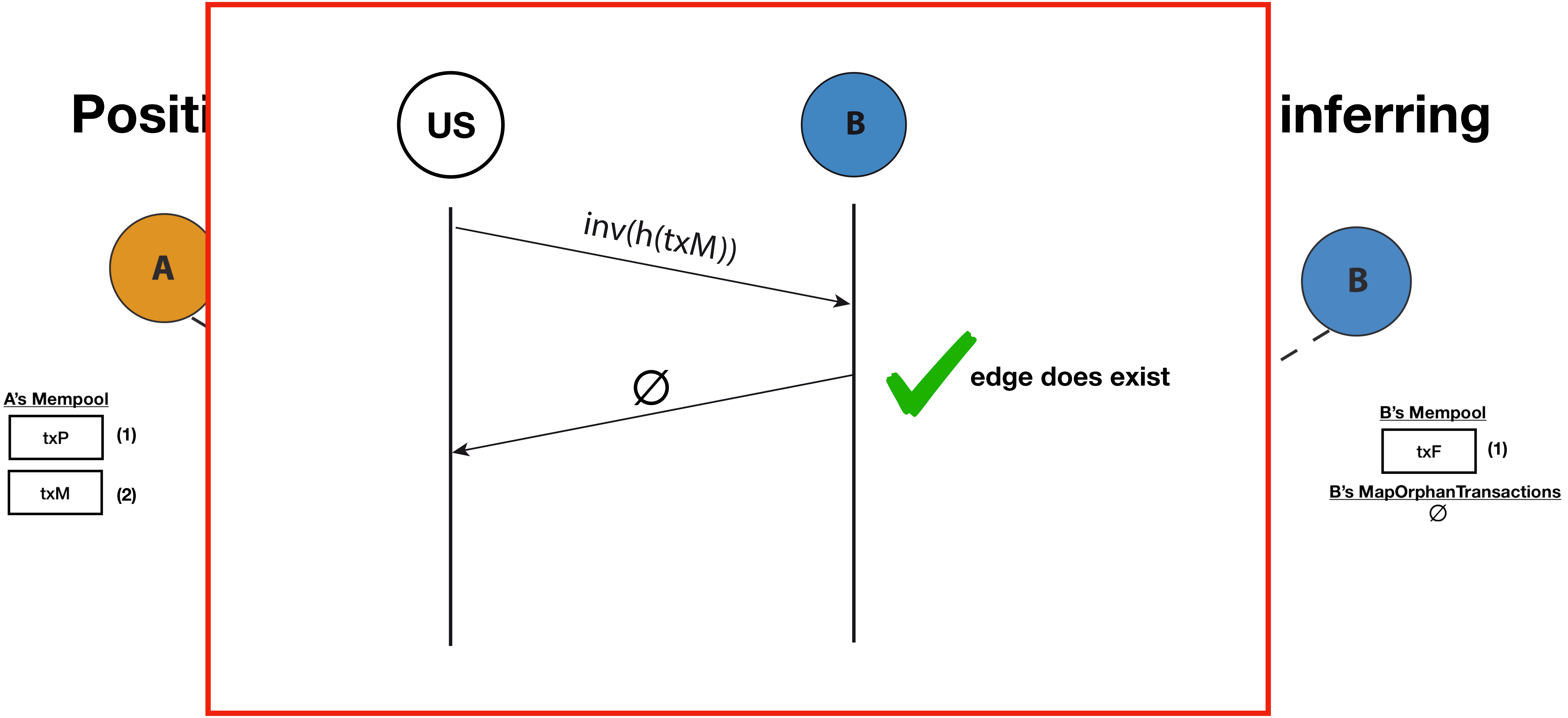
A BASIC TOPOLOGY INFERRING TECHNIQUE



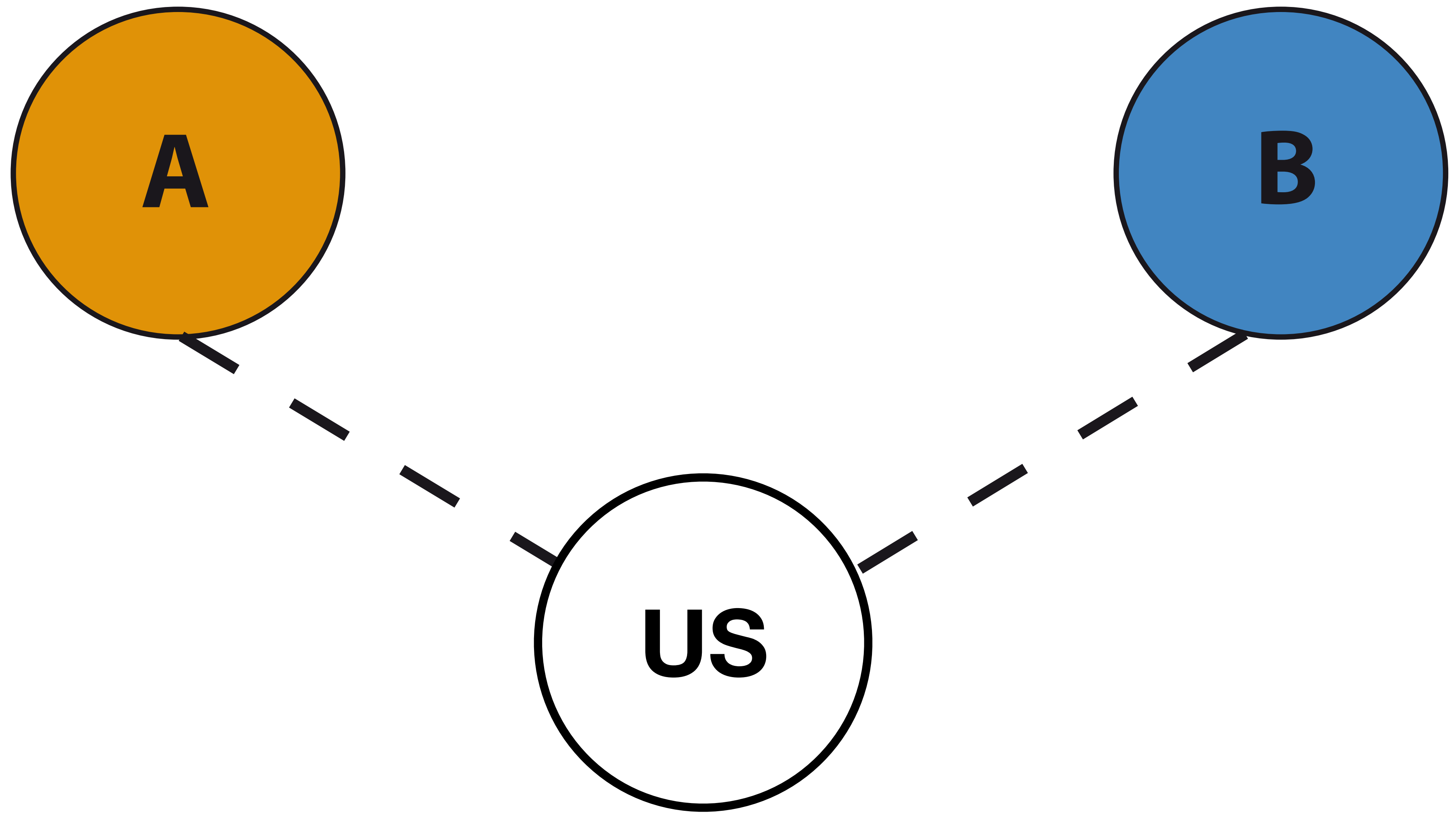
A BASIC TOPOLOGY INFERRING TECHNIQUE



A BASIC TOPOLOGY INFERRING TECHNIQUE

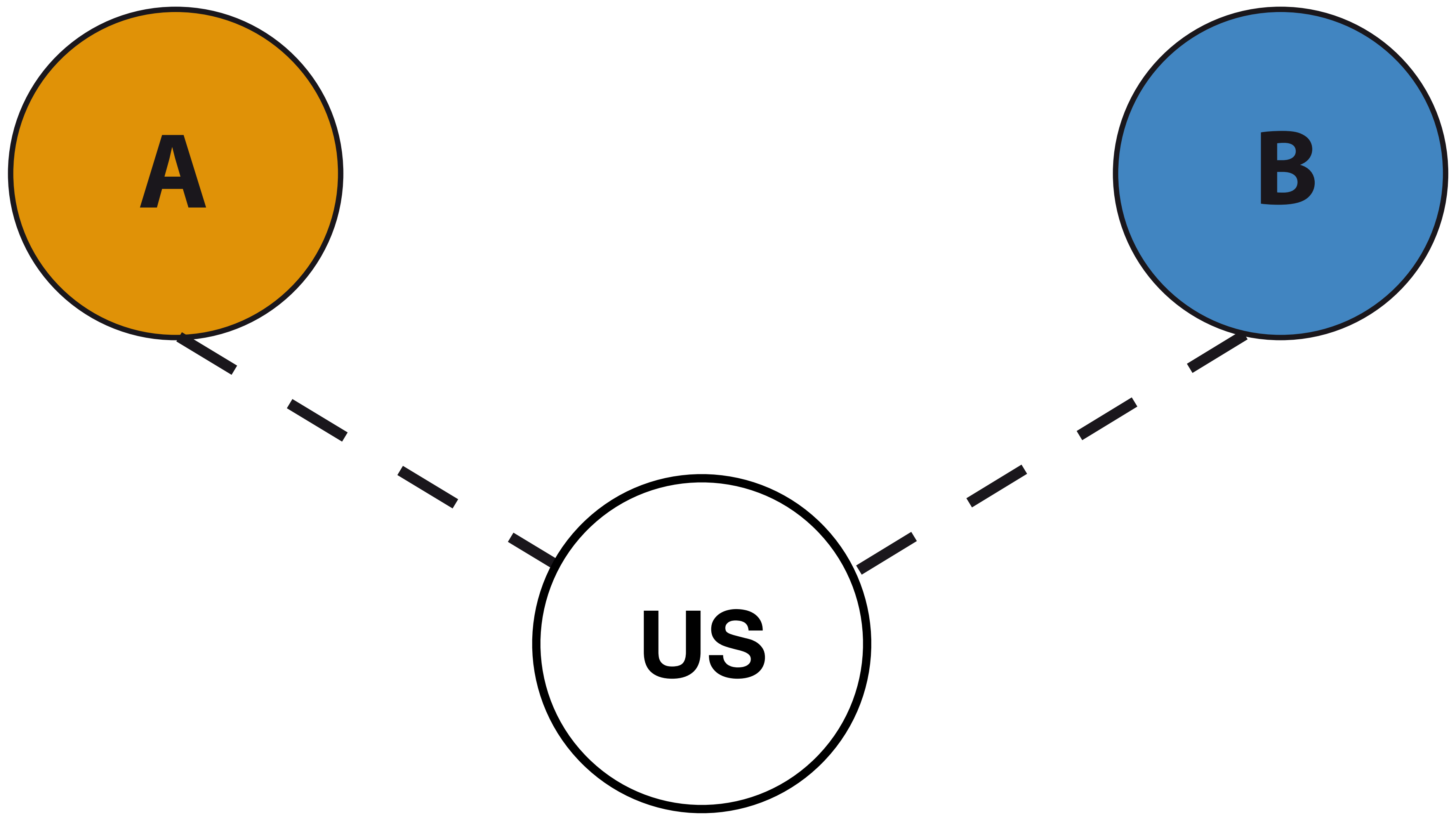


ITS NOT THAT EASY



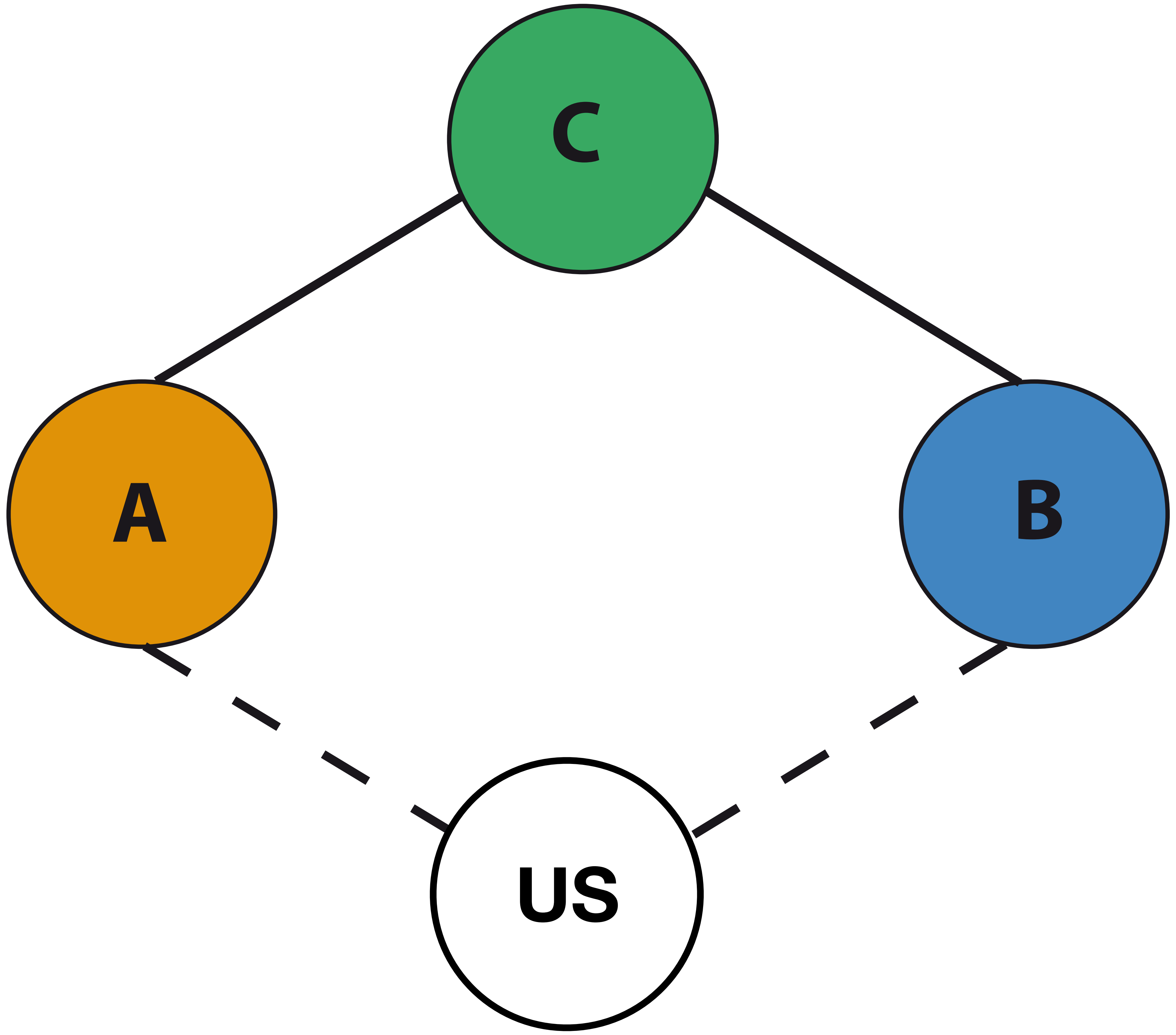
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



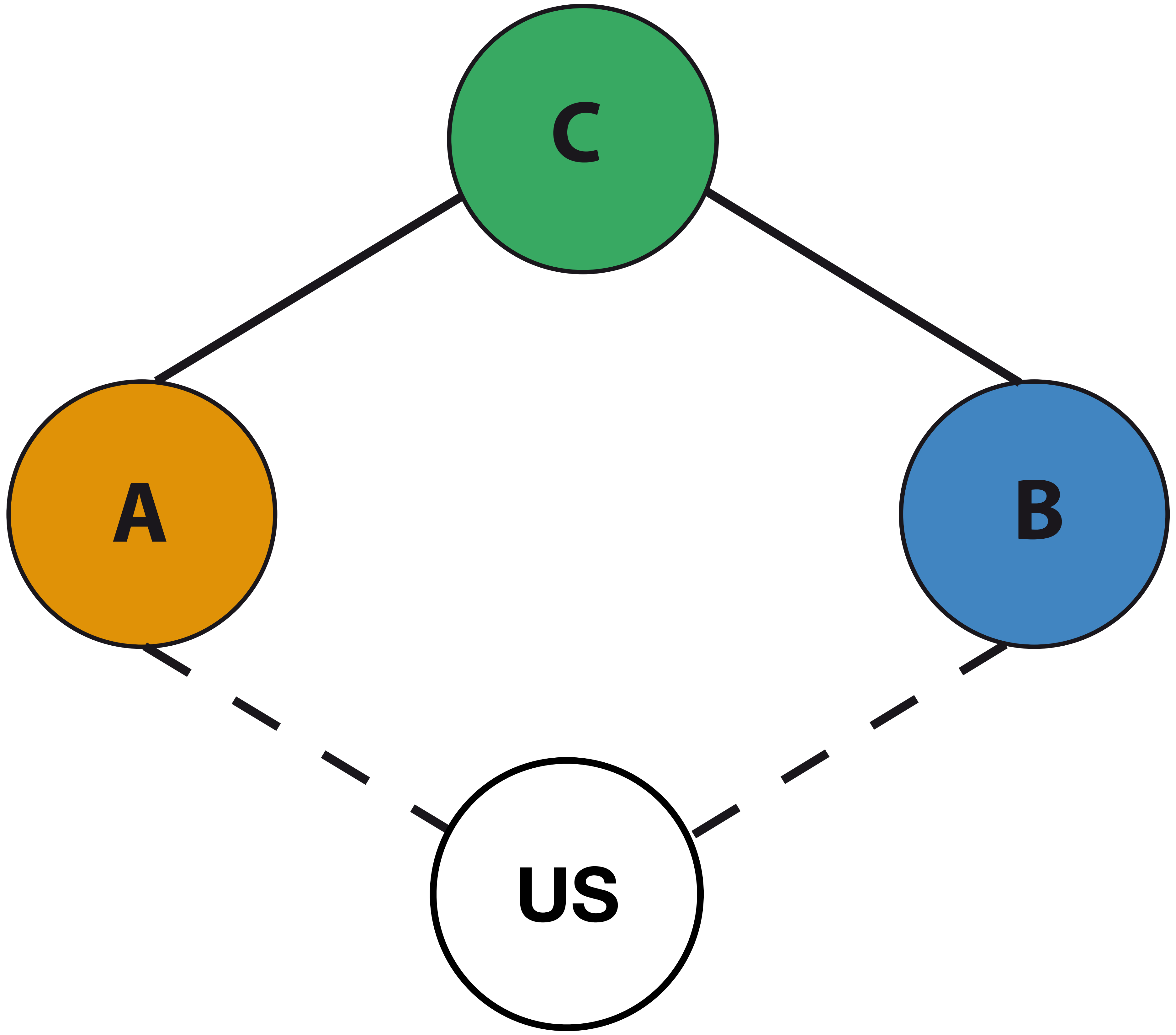
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



A's Mempool

∅

C's Mempool

∅

B's Mempool

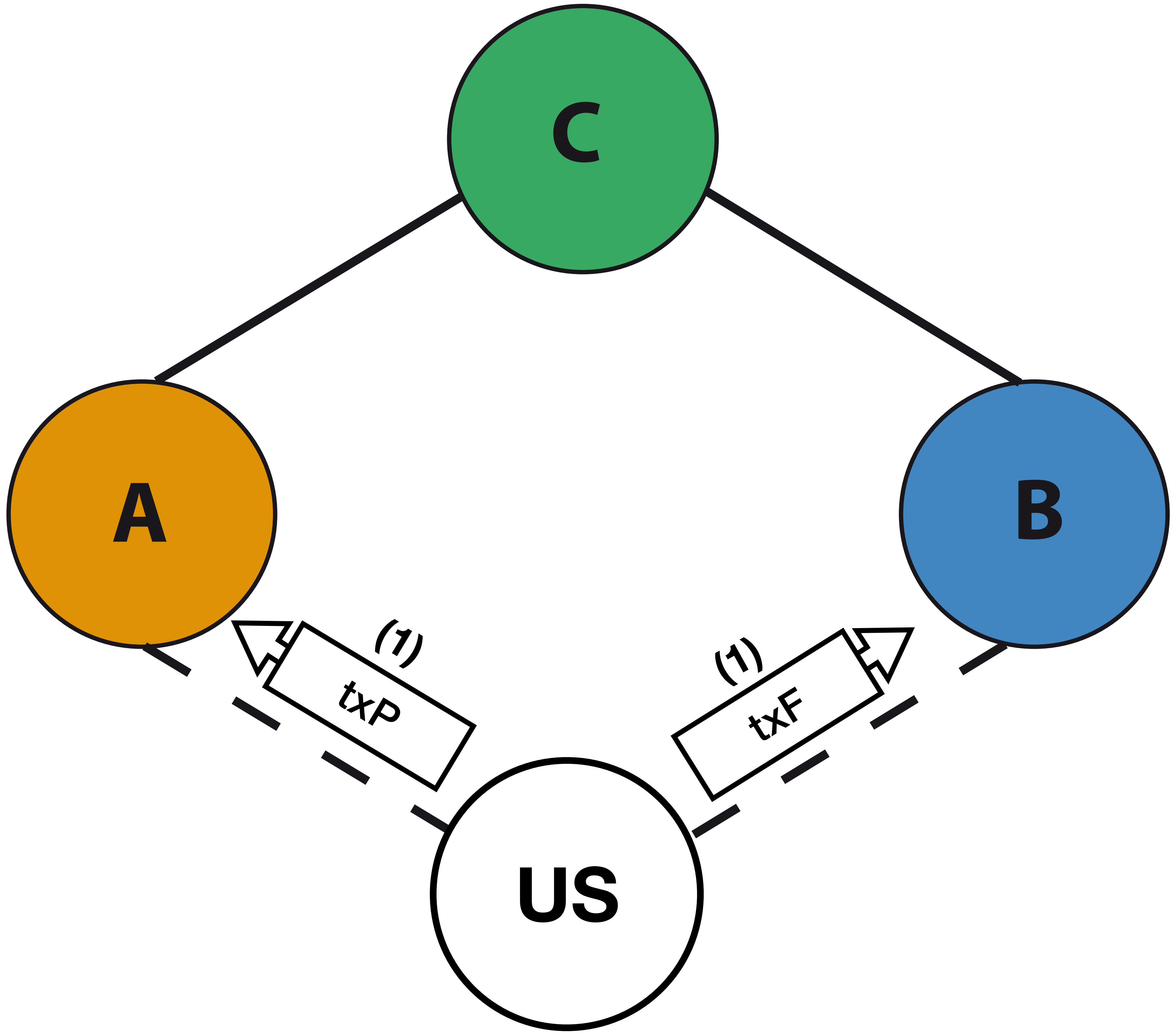
∅

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



A's Mempool

∅

C's Mempool

∅

B's Mempool

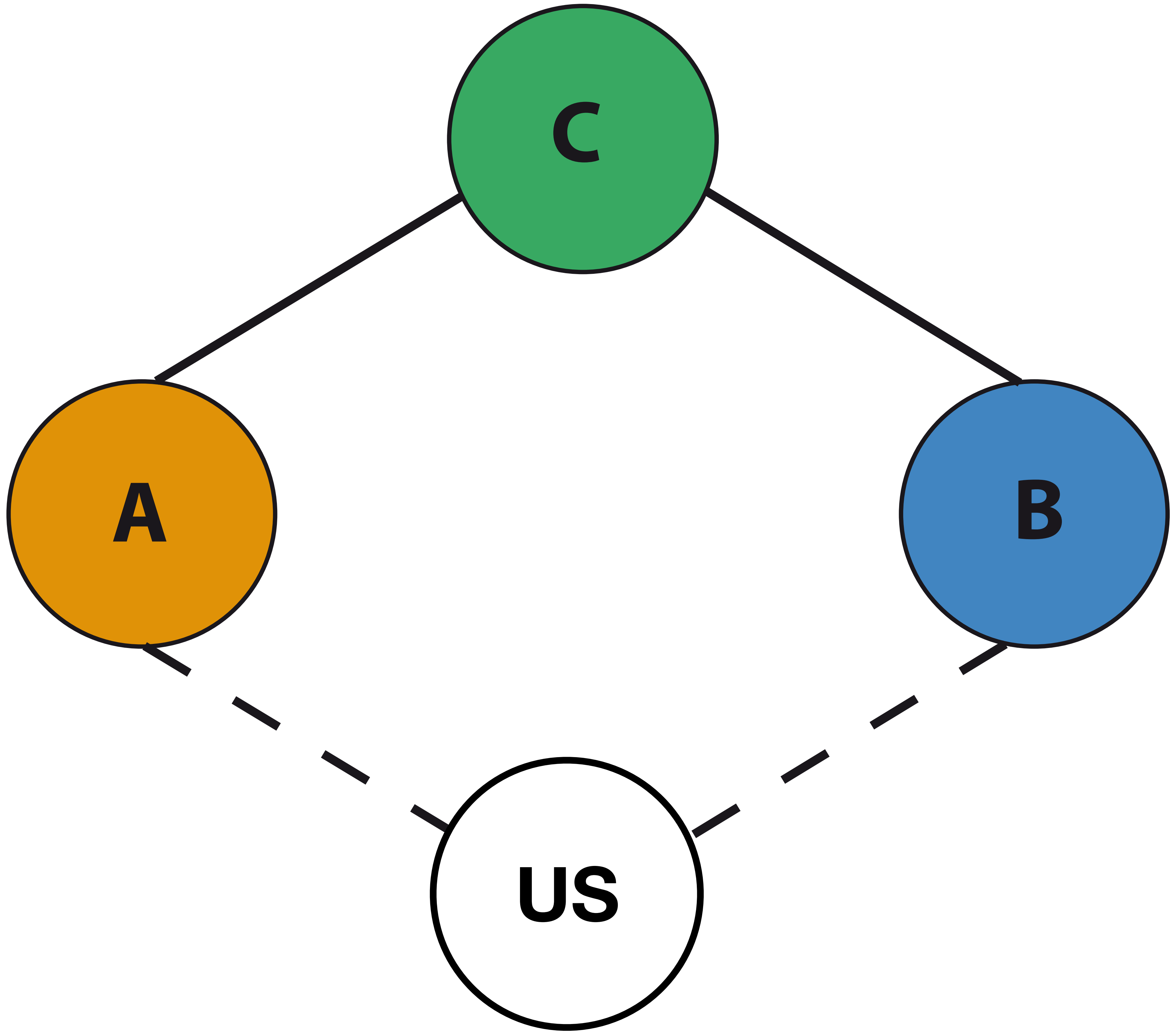
∅

B's MapOrphanTransactions

∅

ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



<u>A's Mempool</u>

∅

<u>C's Mempool</u>

∅

<u>B's Mempool</u>

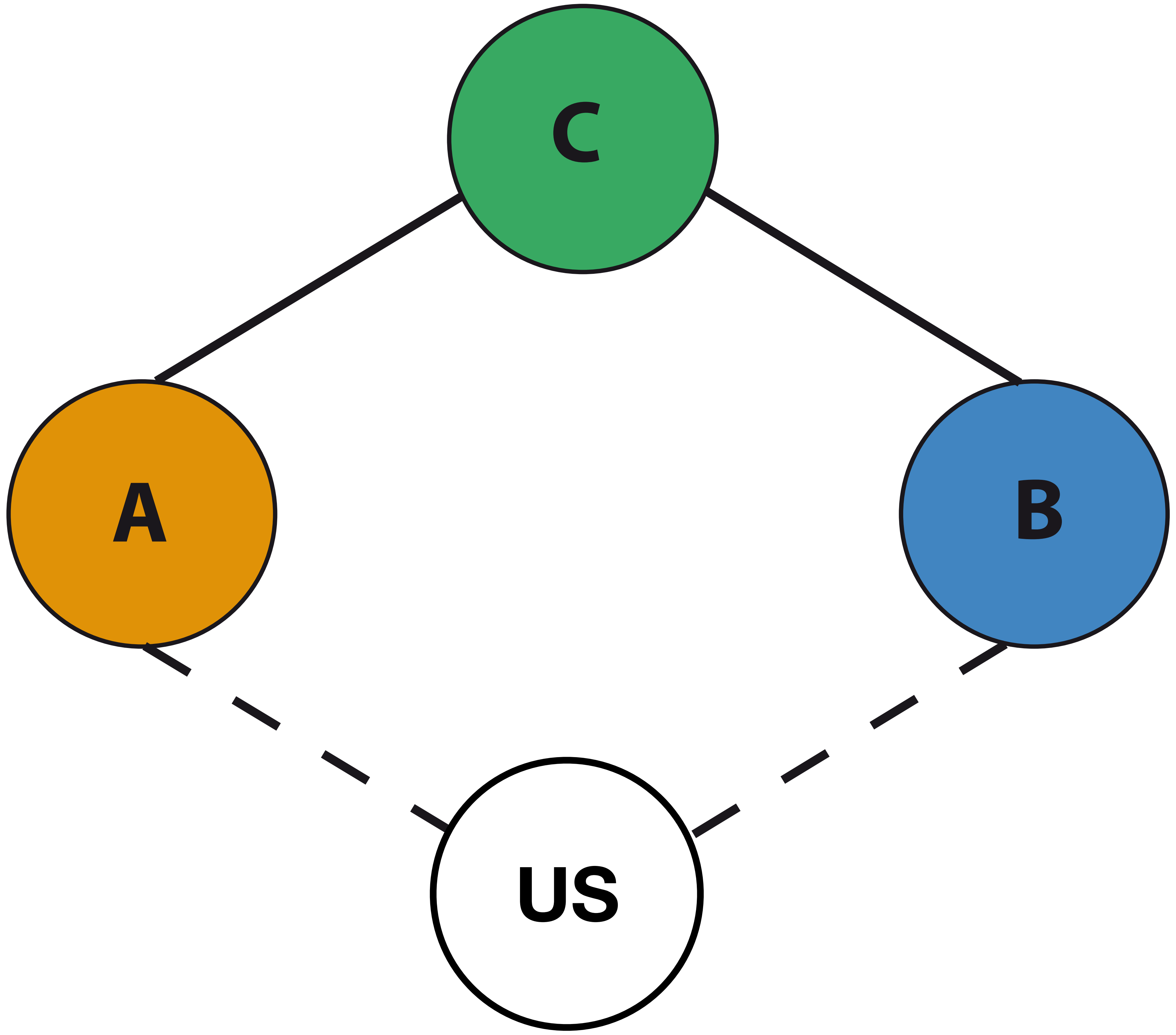
∅

<u>B's MapOrphanTransactions</u>

∅

ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



A's Mempool

| txP | (1) |

C's Mempool

∅

B's Mempool

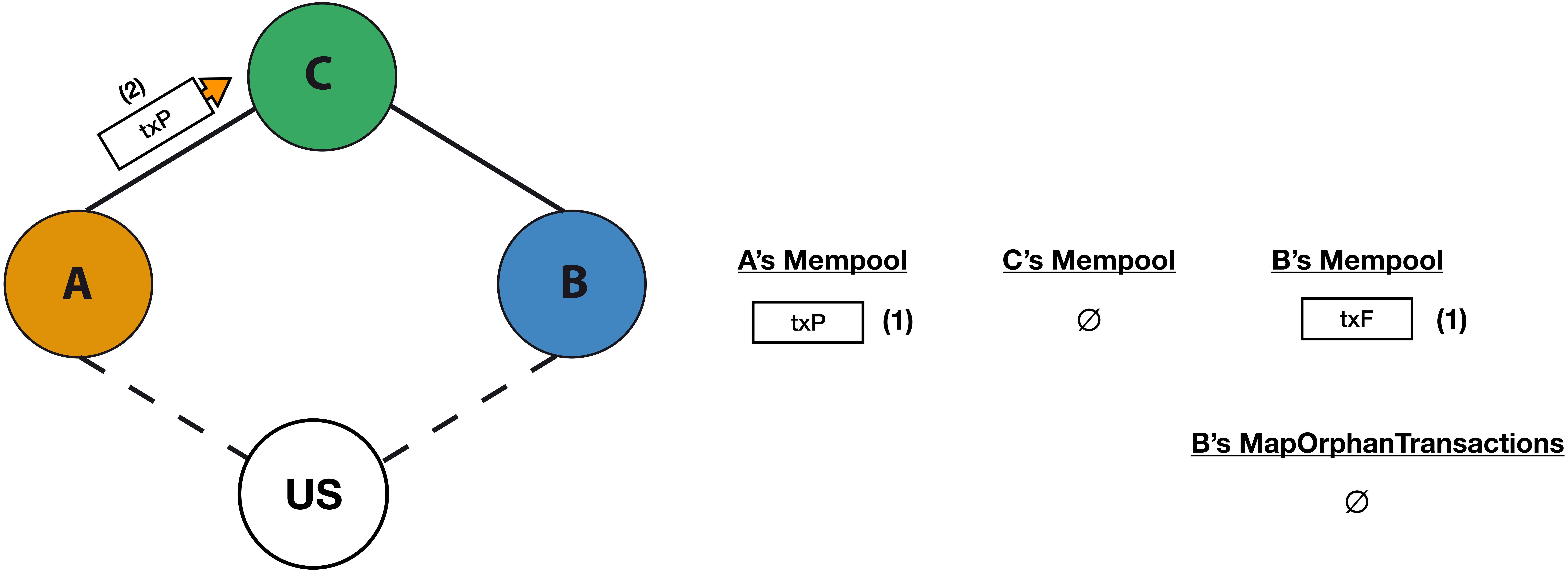
| txF | (1) |

B's MapOrphanTransactions

∅

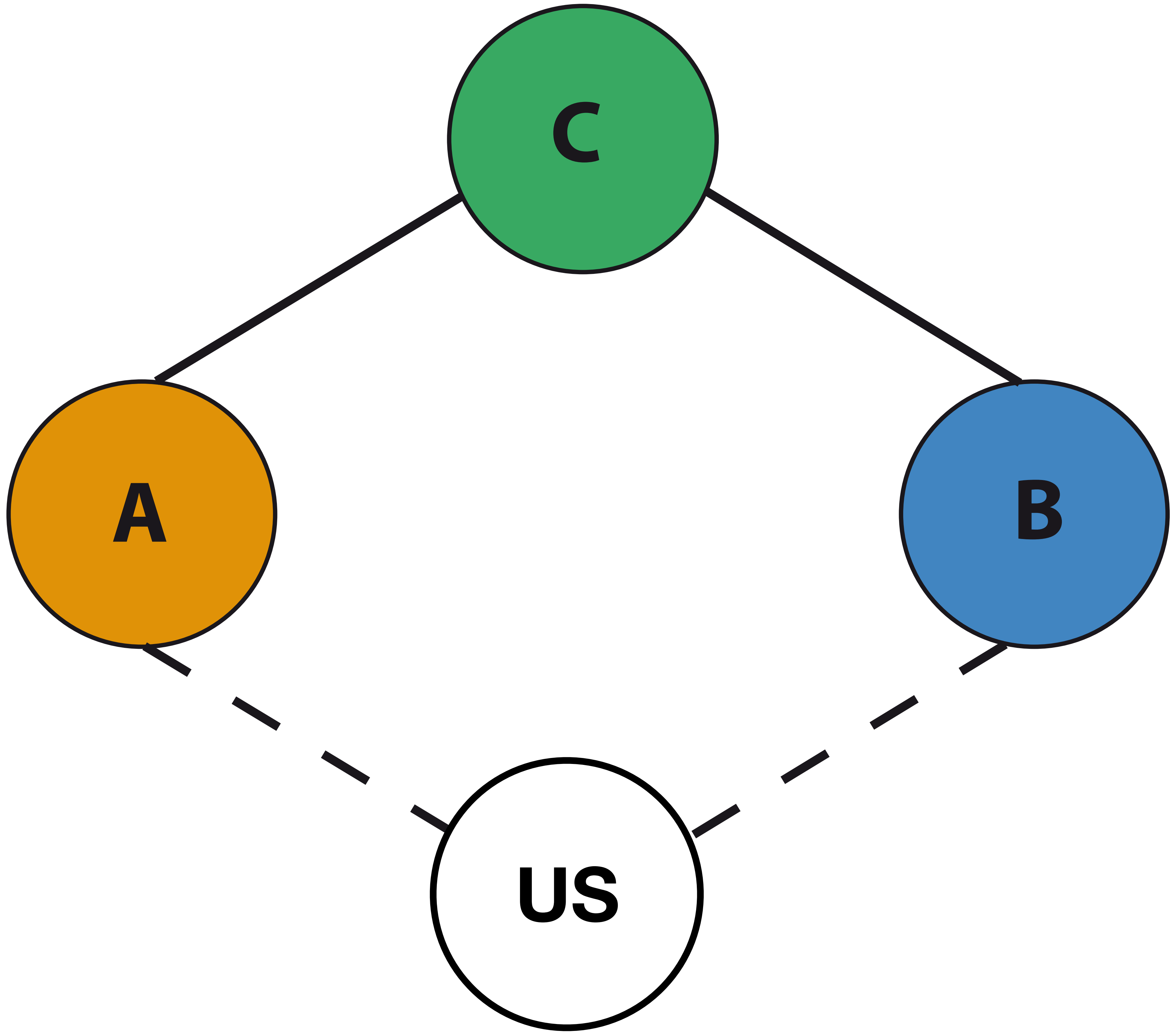
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



A's Mempool

txP (1)

C's Mempool

∅

B's Mempool

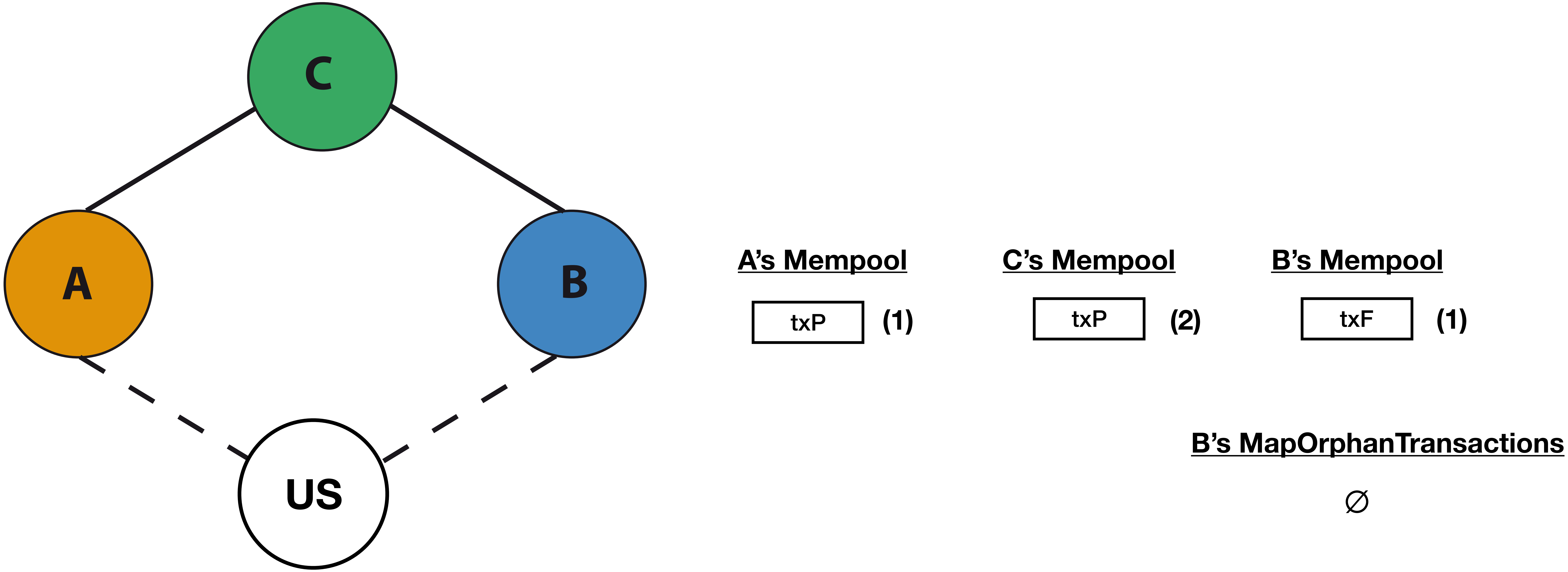
txF (1)

B's MapOrphanTransactions

∅

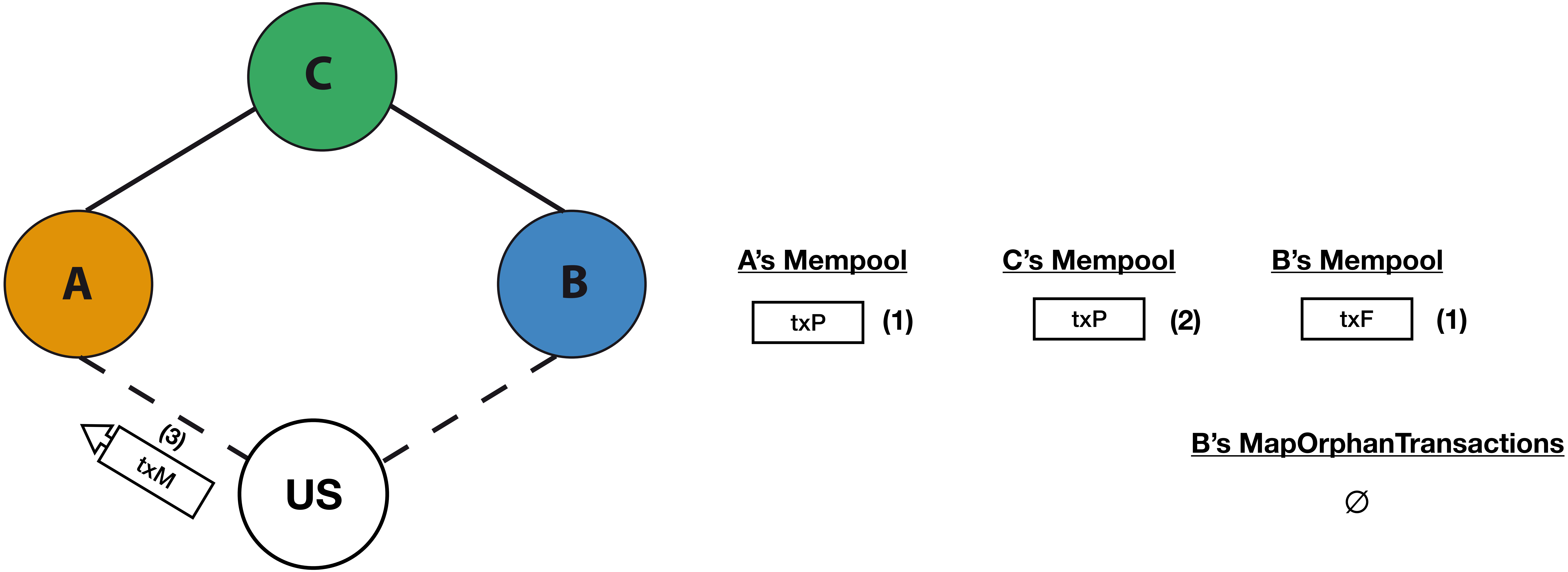
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



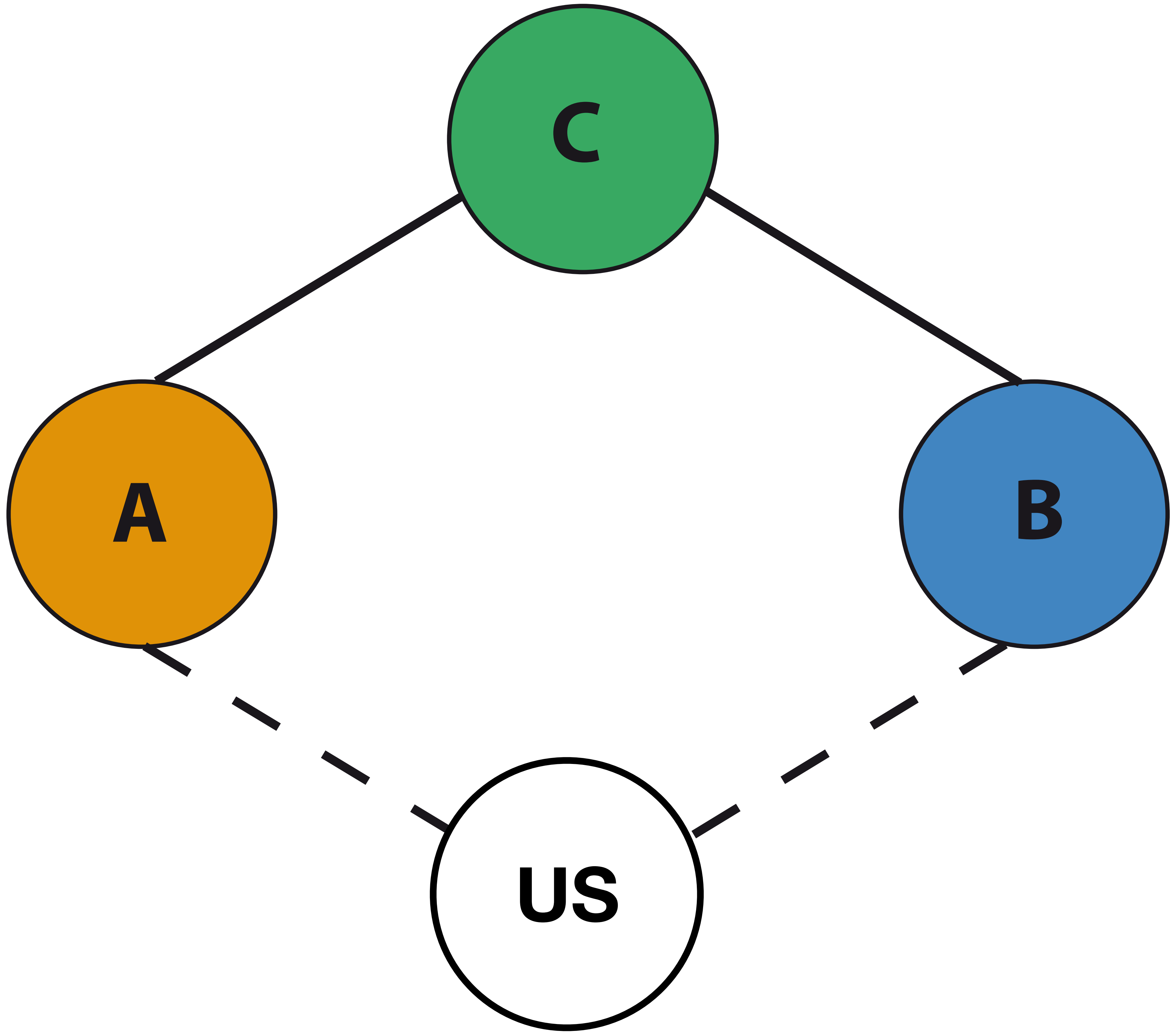
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



A's Mempool

txP (1)

C's Mempool

txP (2)

B's Mempool

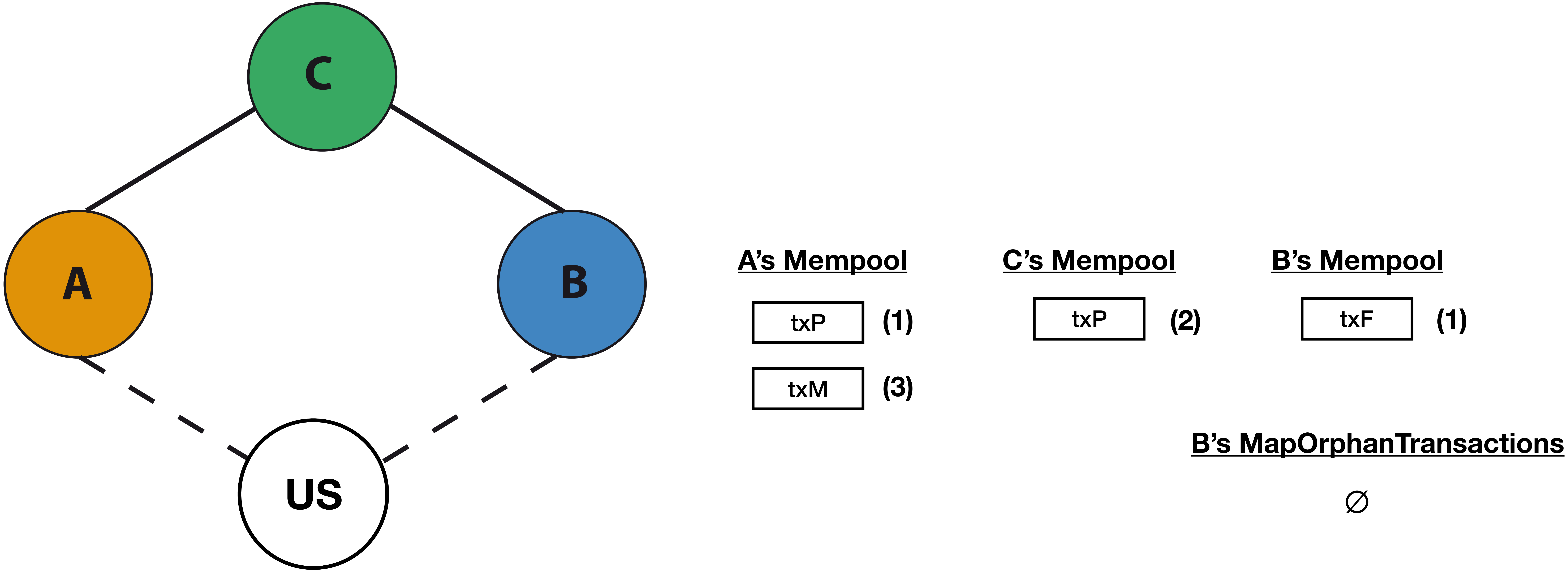
txF (1)

B's MapOrphanTransactions

∅

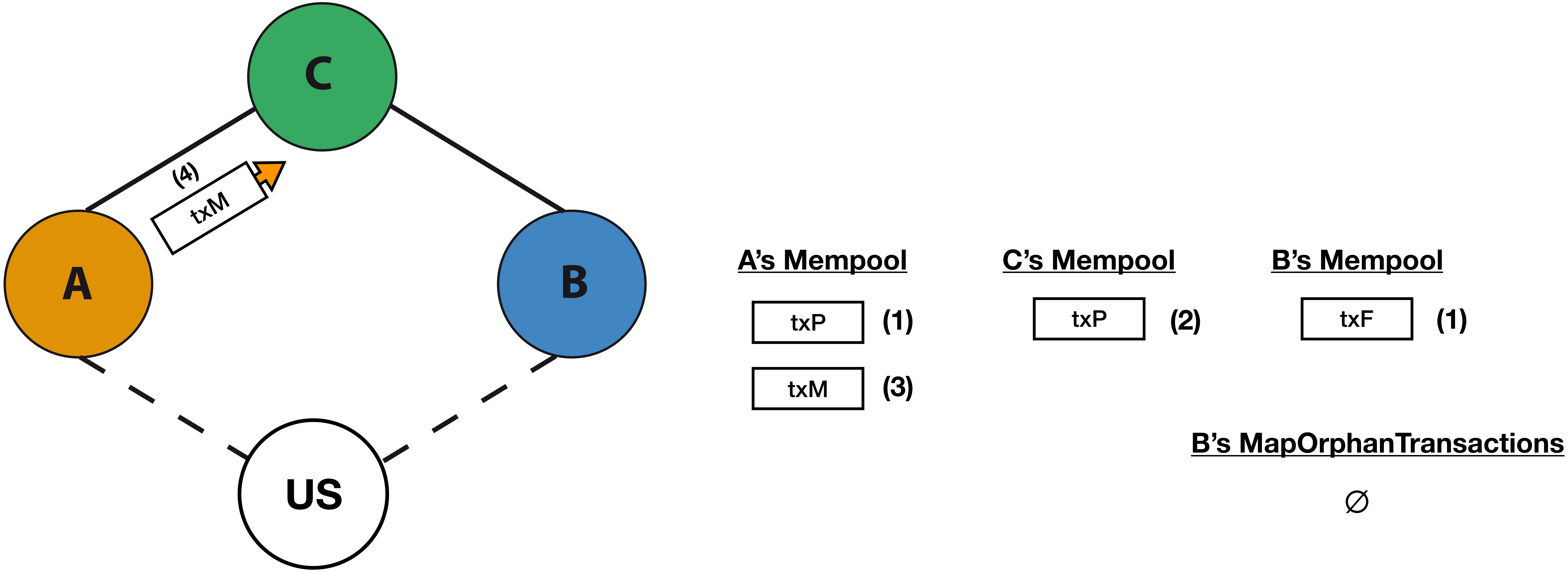
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



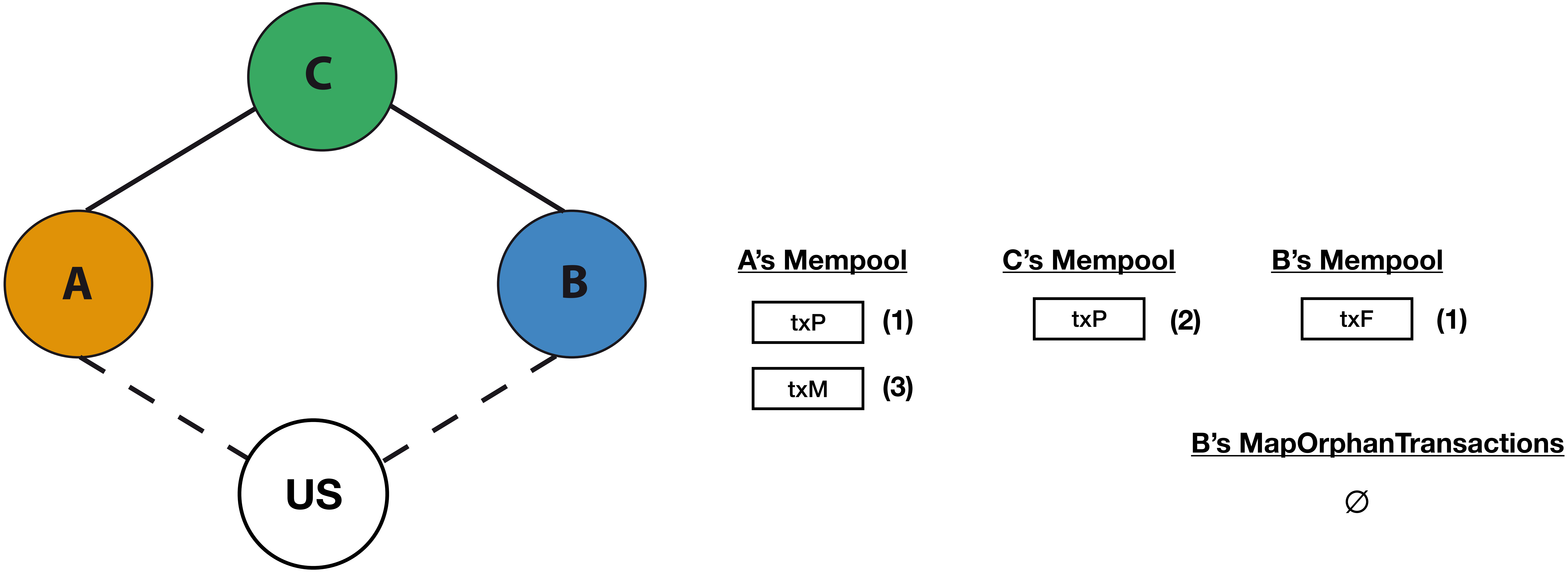
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



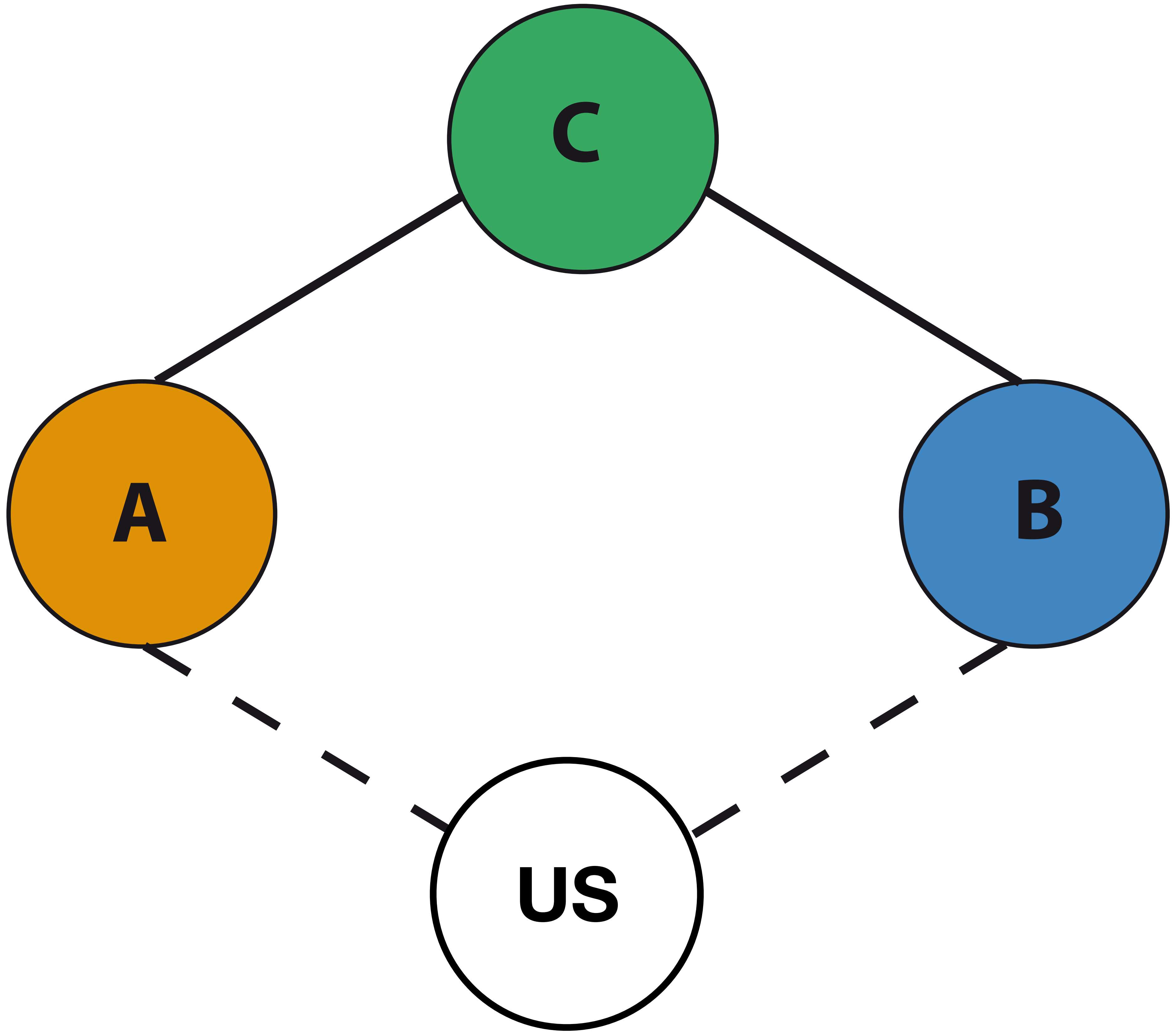
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



A's Mempool

txP (1)

txM (3)

C's Mempool

txP (2)

txM (4)

B's Mempool

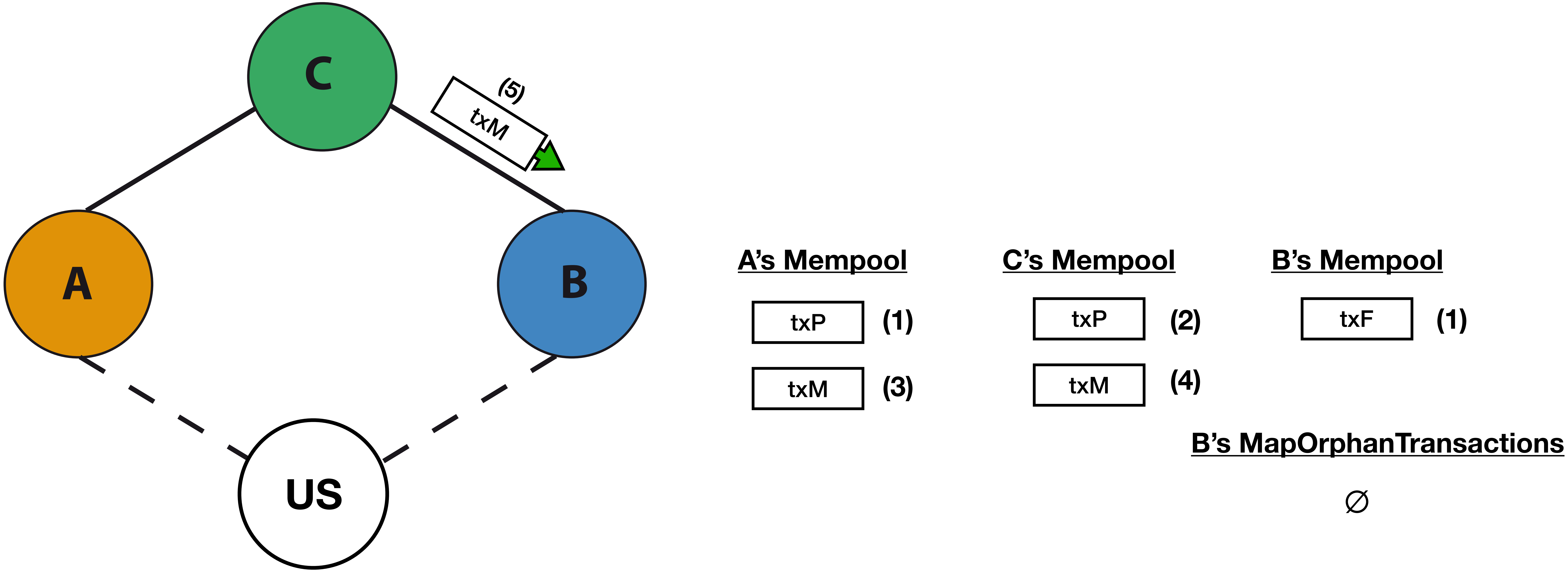
txF (1)

B's MapOrphanTransactions

∅

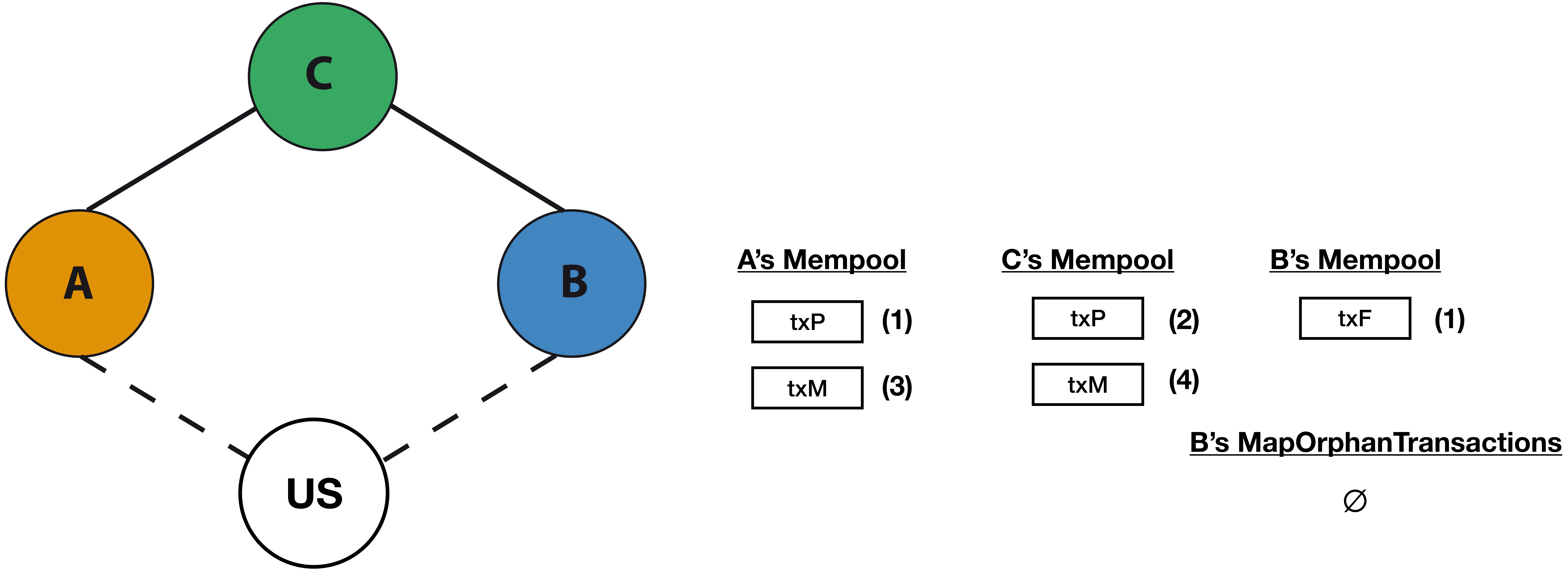
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture



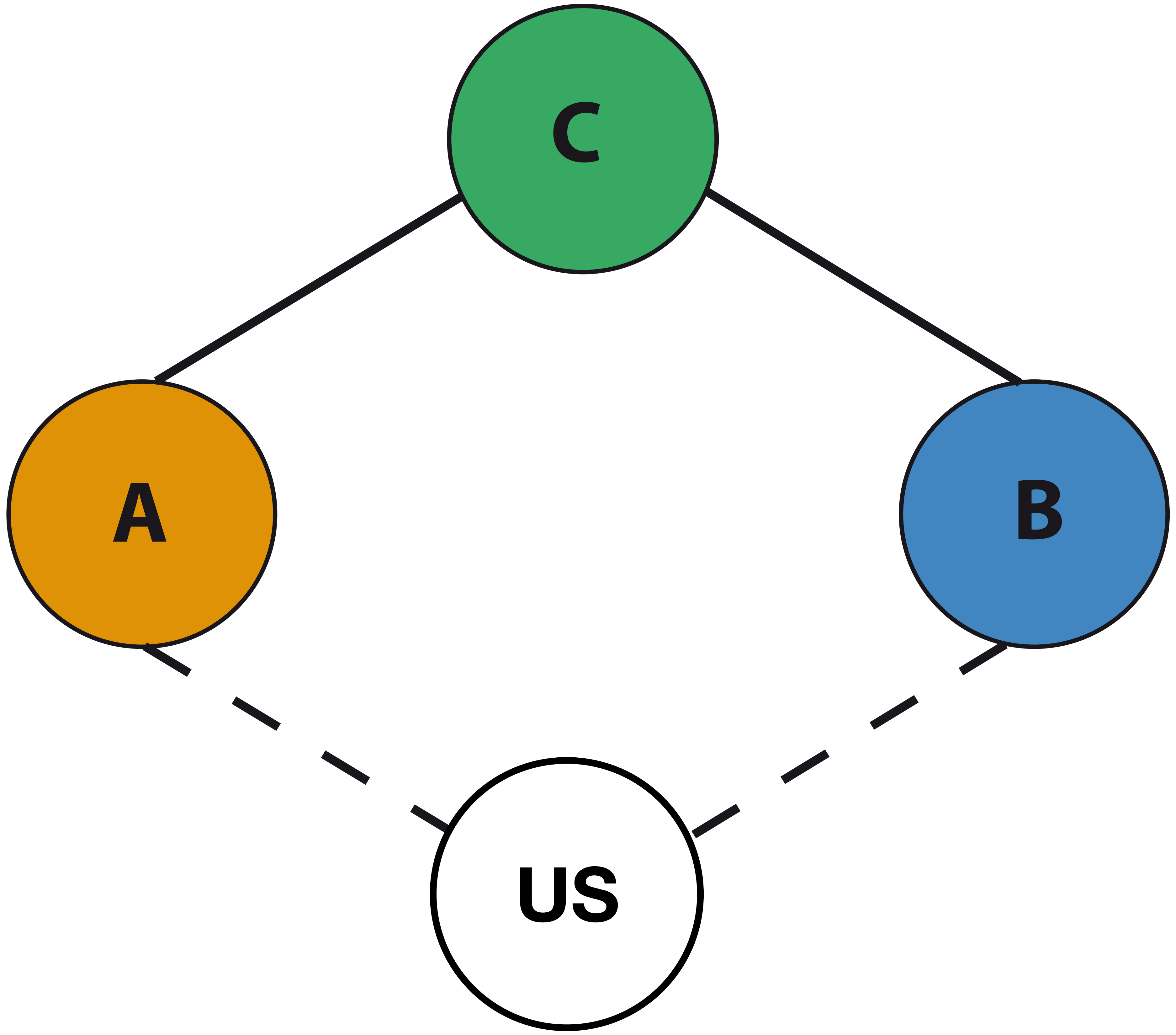
ITS NOT THAT EASY

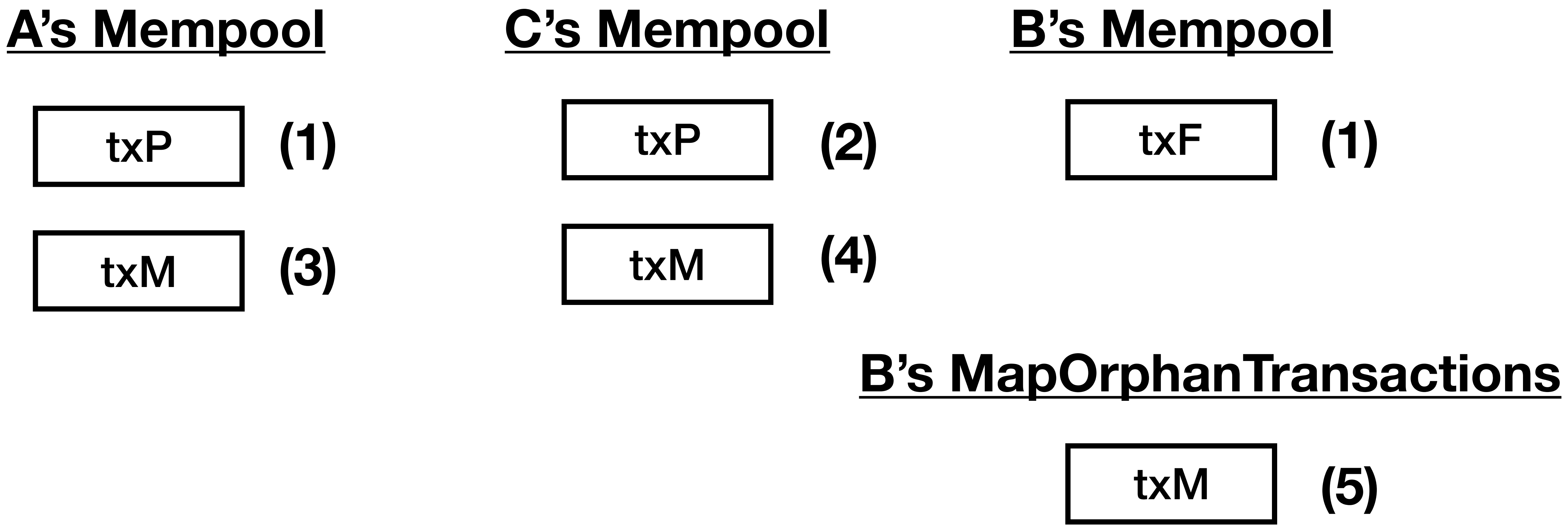
Long story short, the technique will fail if we add an additional node to the picture



ITS NOT THAT EASY

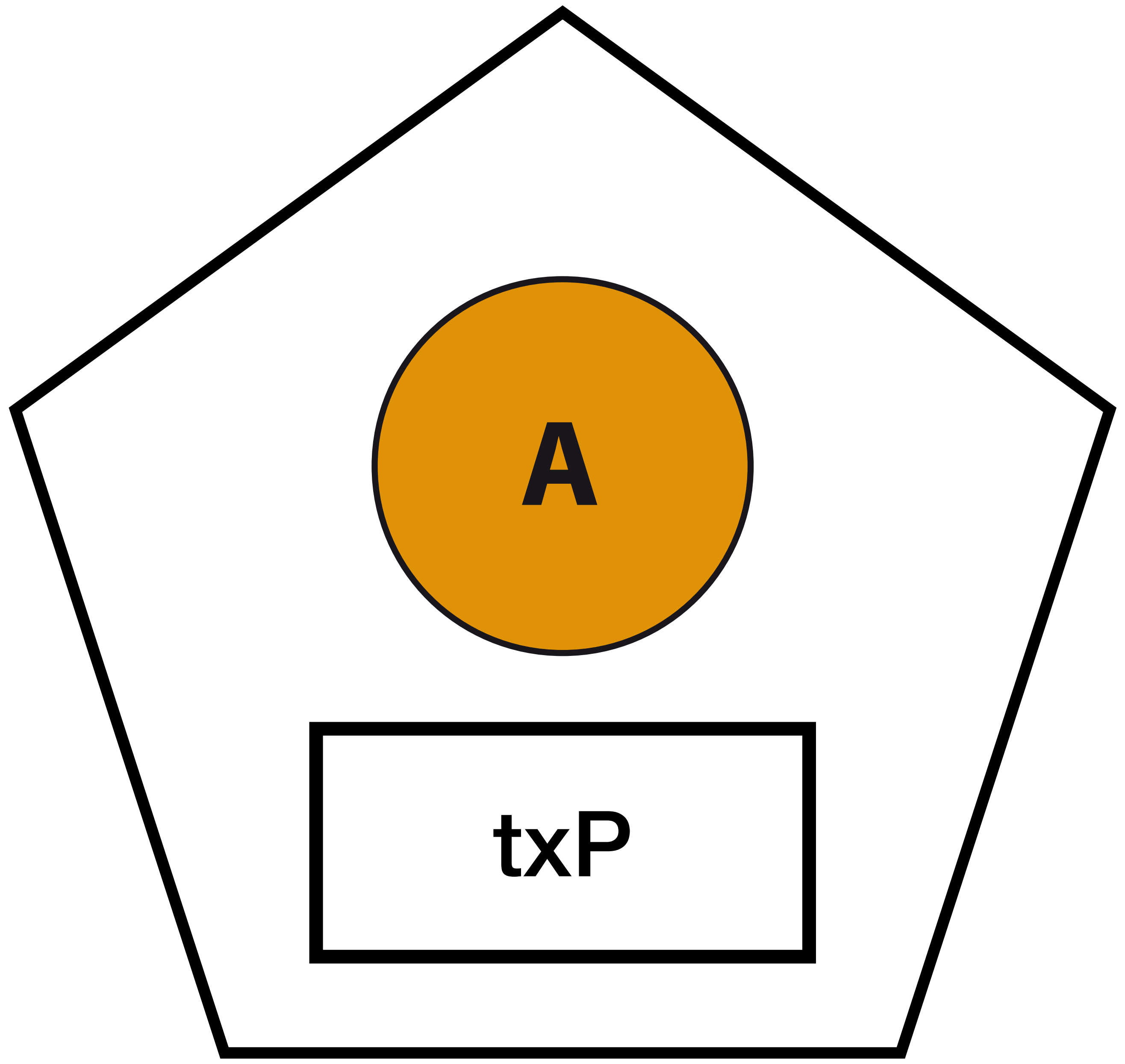
Long story short, the technique will fail if we add an additional node to the picture



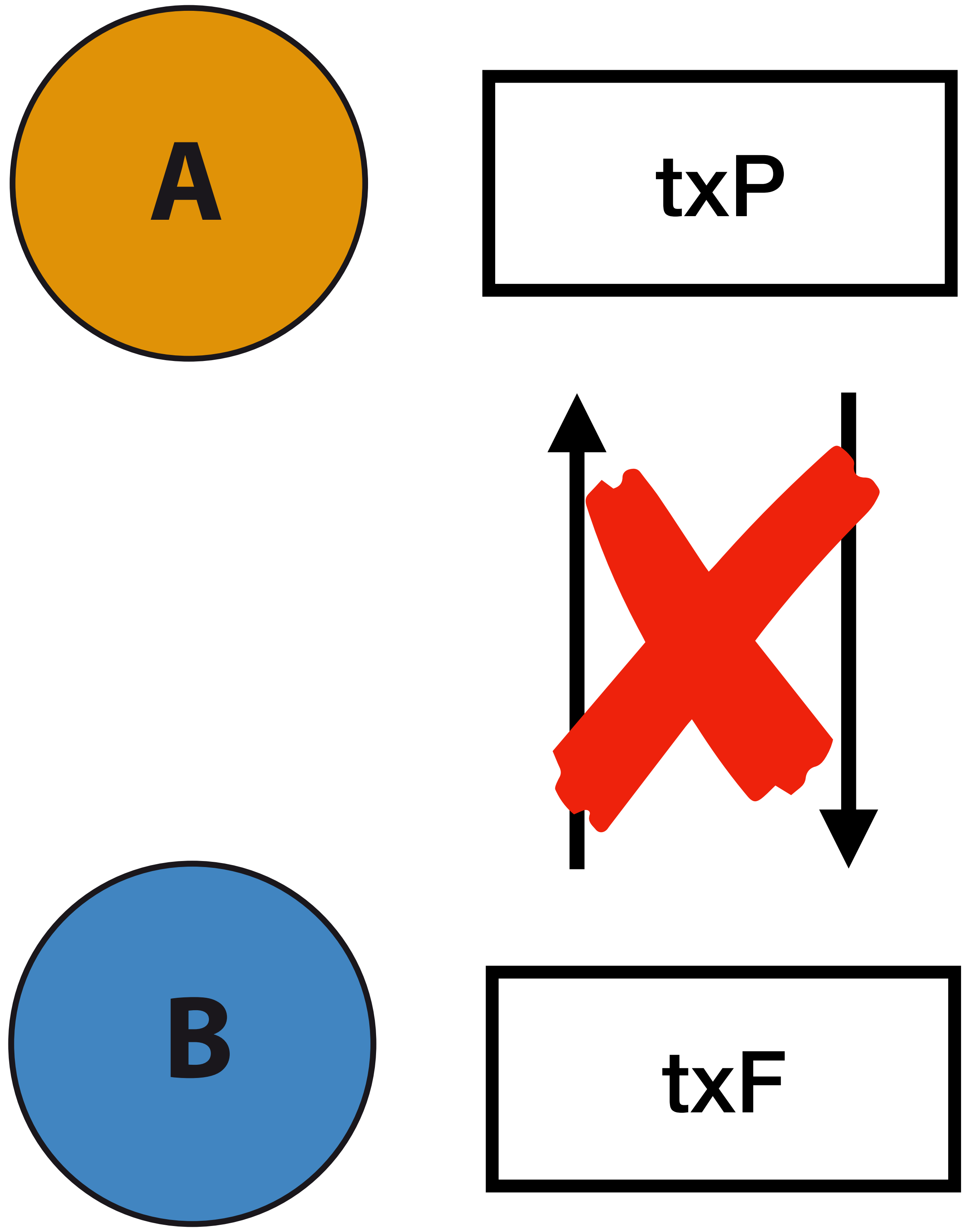


MAKE THIS WORK IN A REAL NETWORK

Isolation



Synchrony



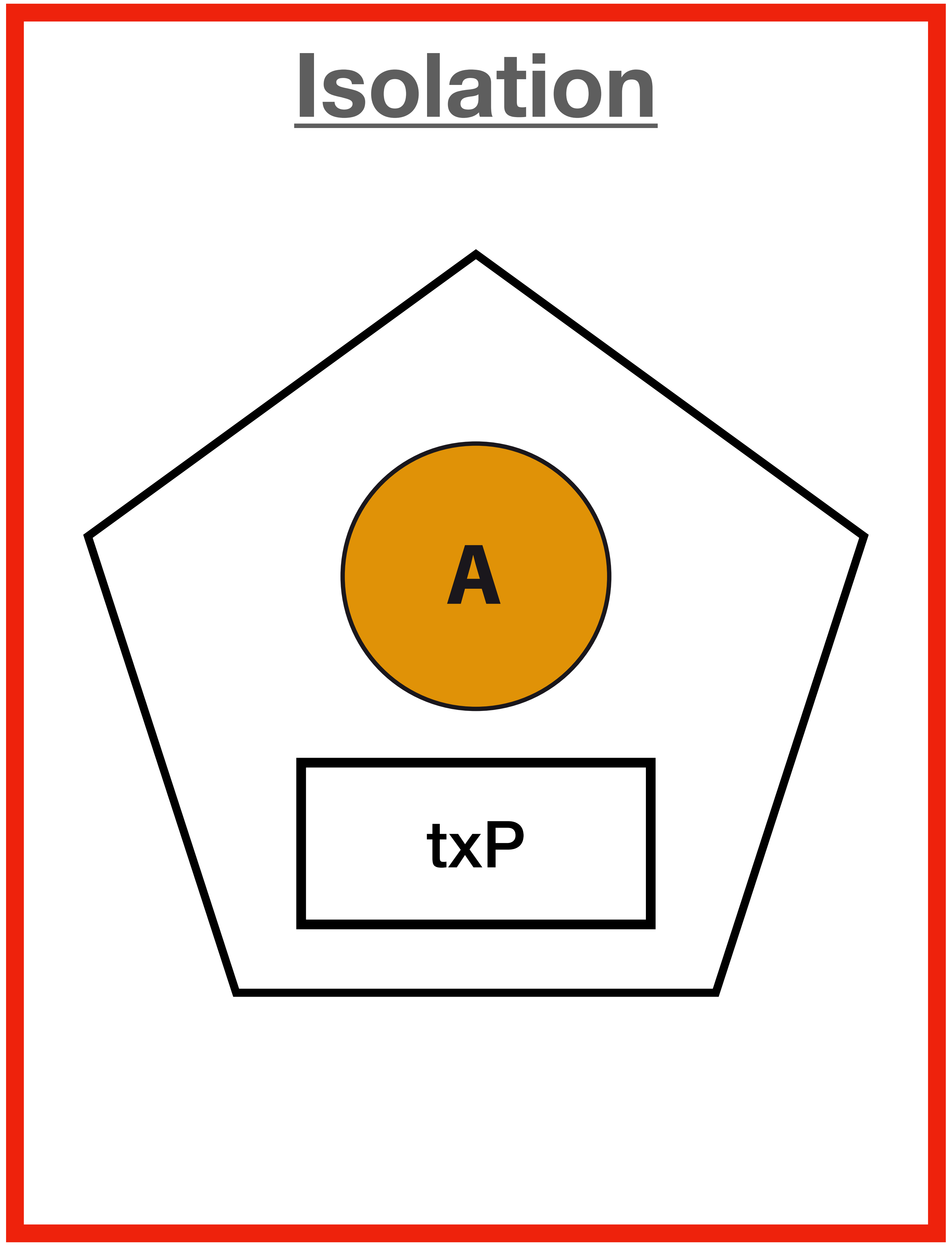
Efficiency

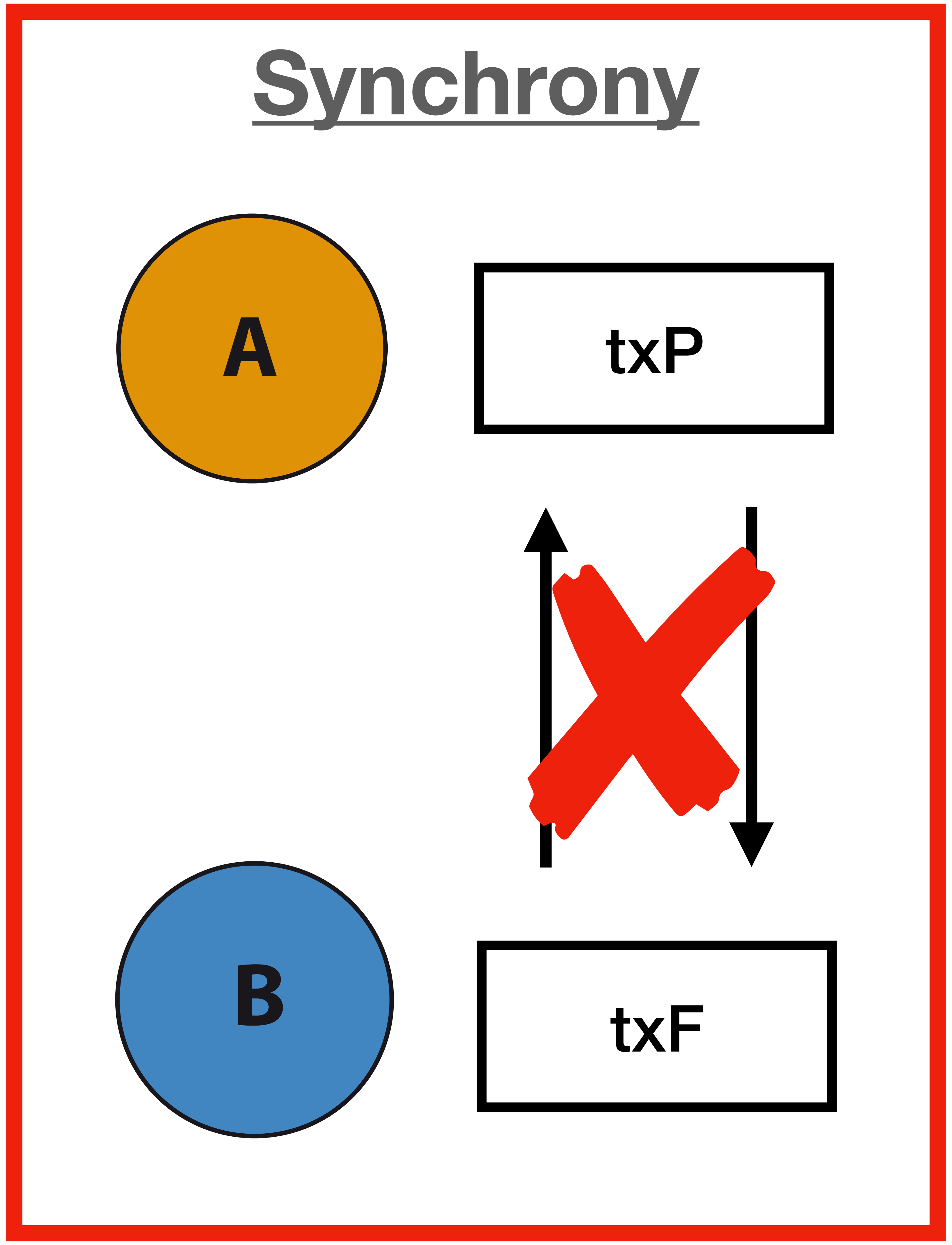
$$\approx O(n)$$

$$\approx O(\sqrt{n})$$

$$n = \#nodes$$

MAKE THIS WORK IN A REAL NETWORK





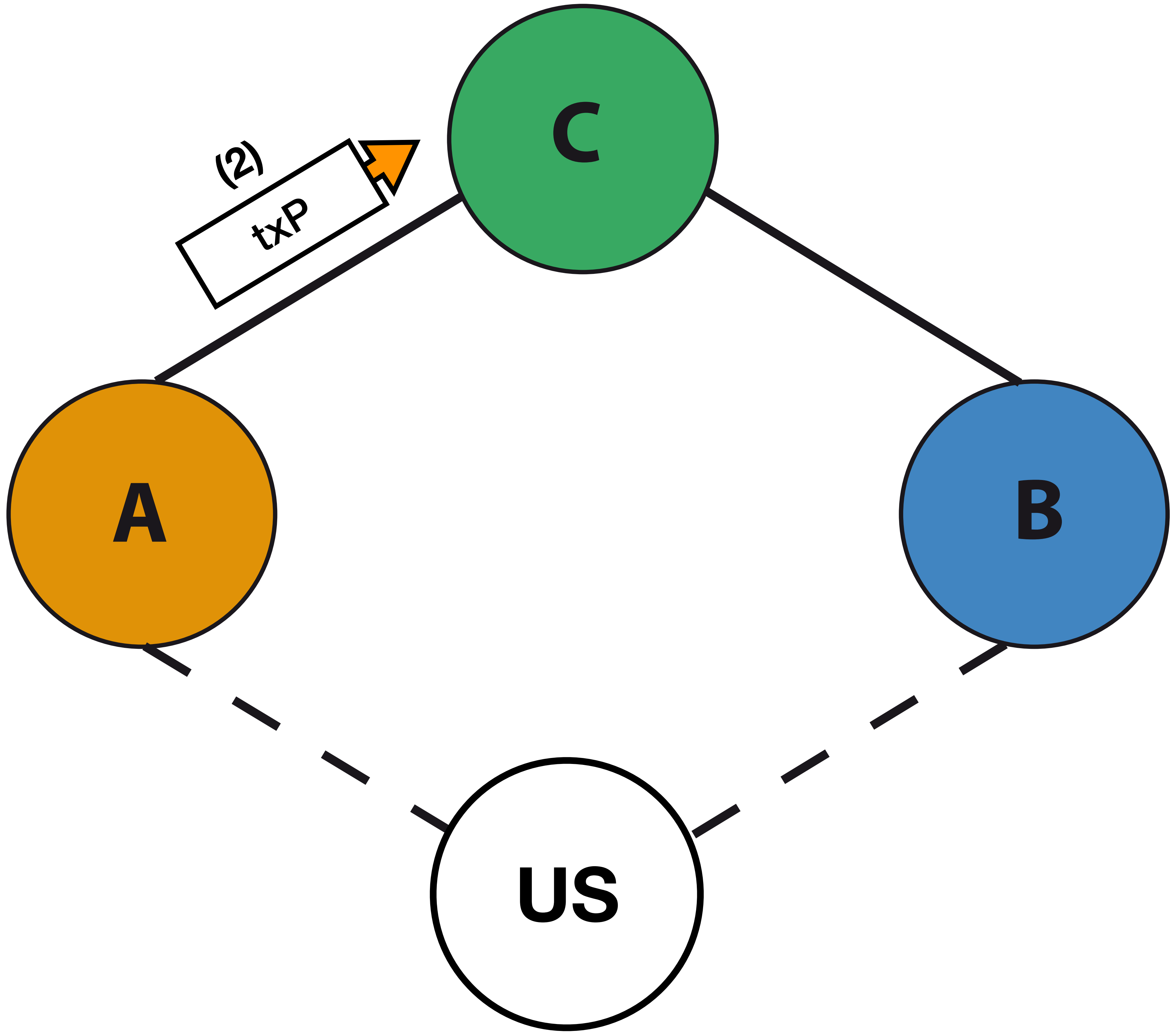
Efficiency

$\approx O(n)$

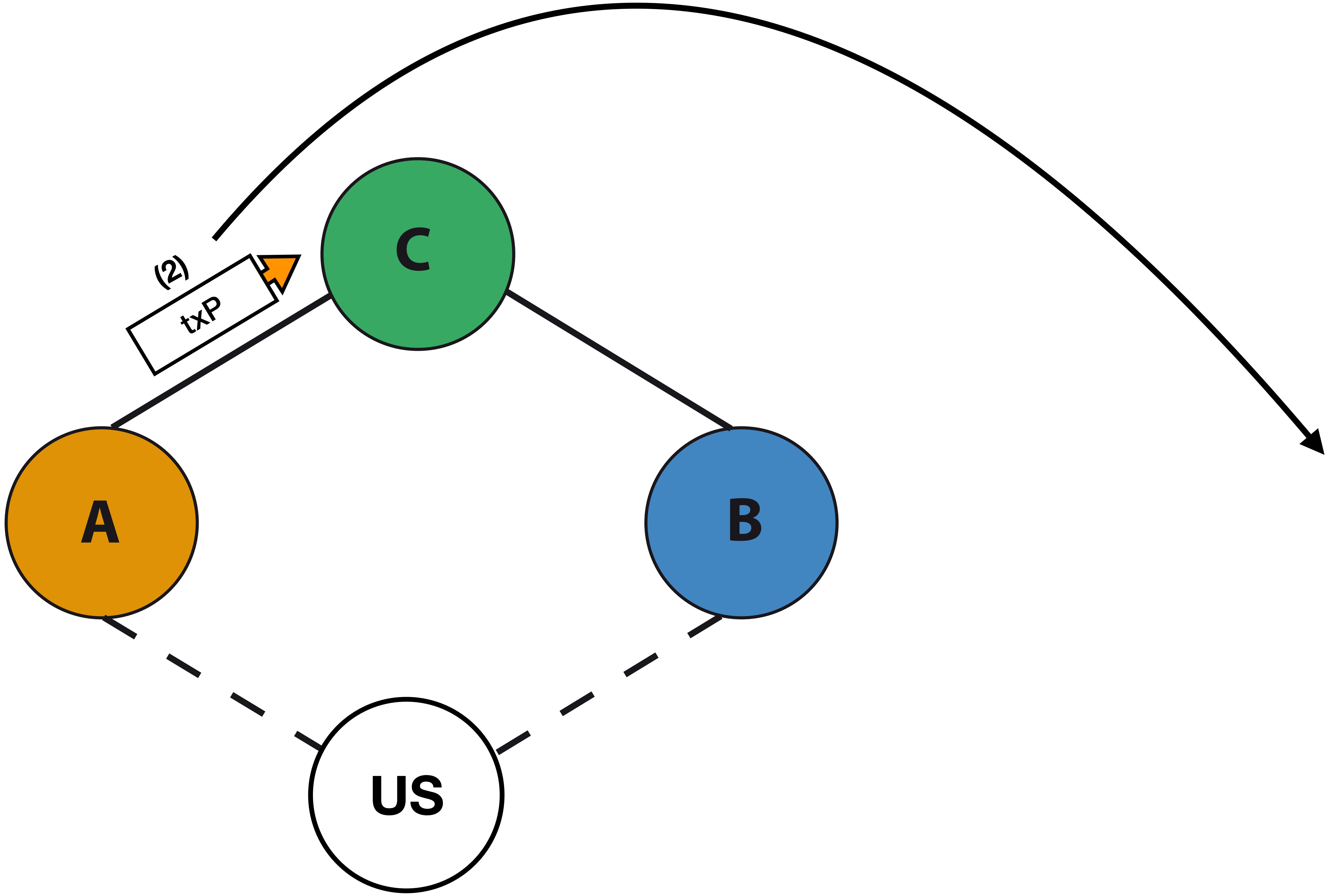
$\approx O(\sqrt{n})$

$n = \#nodes$

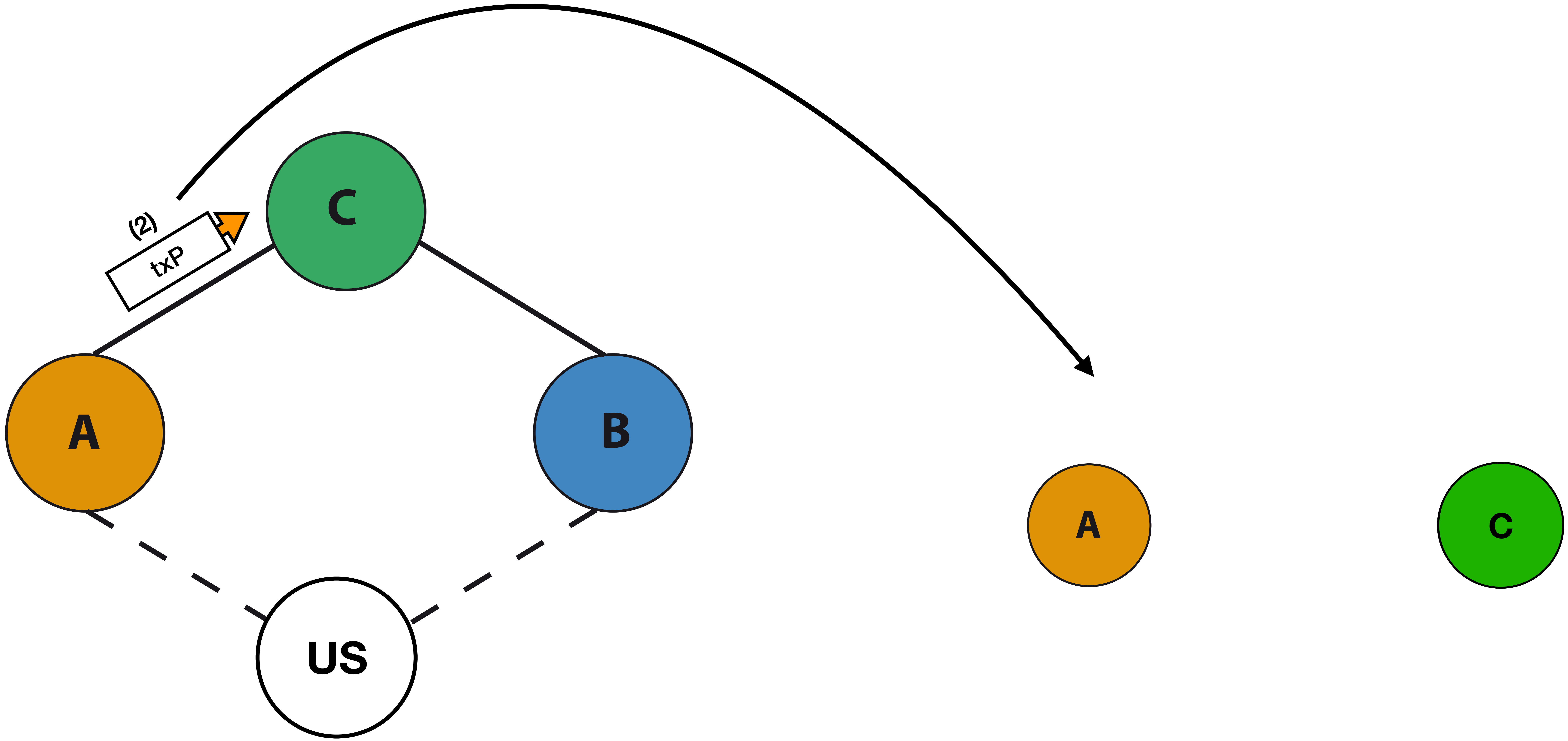
ISOLATION



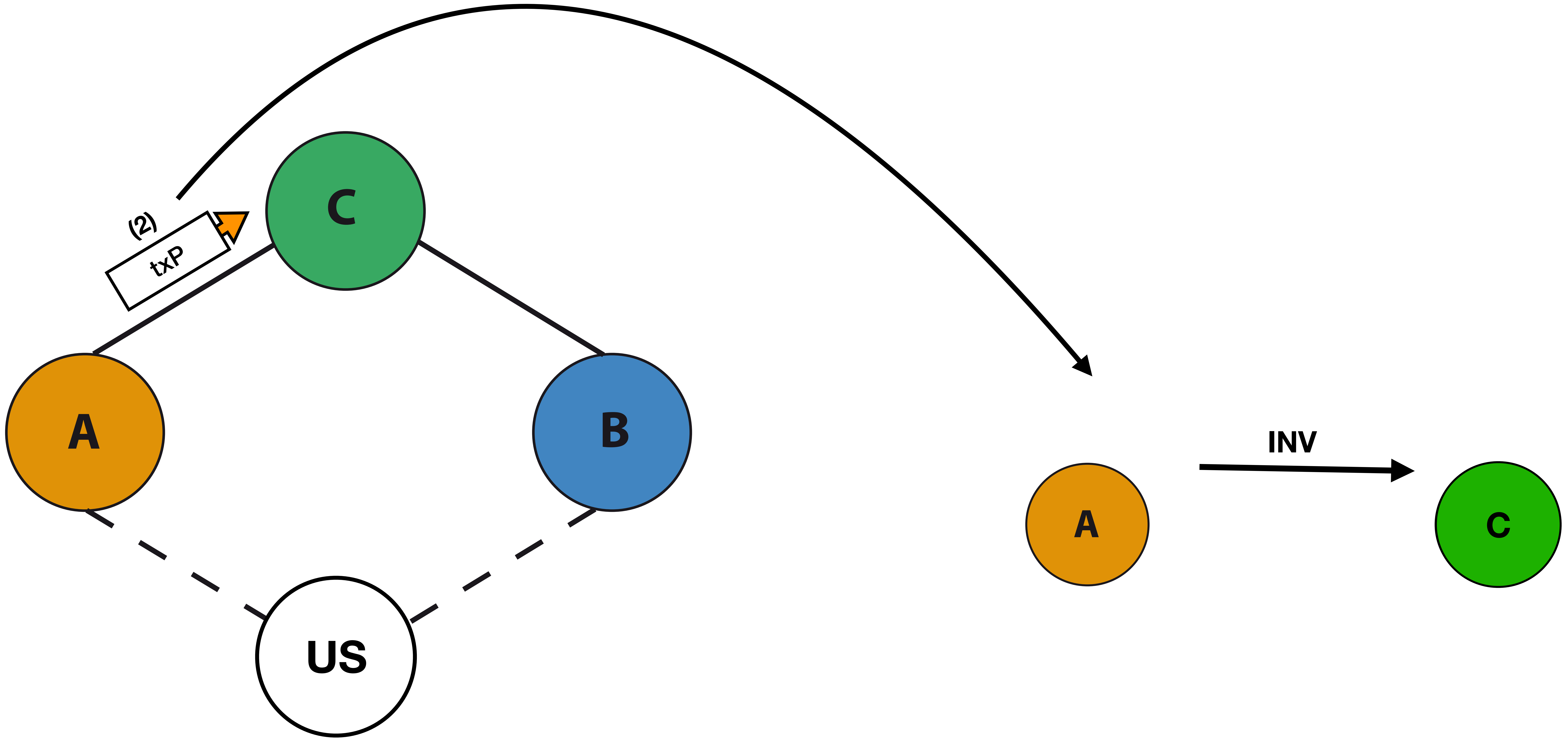
ISOLATION



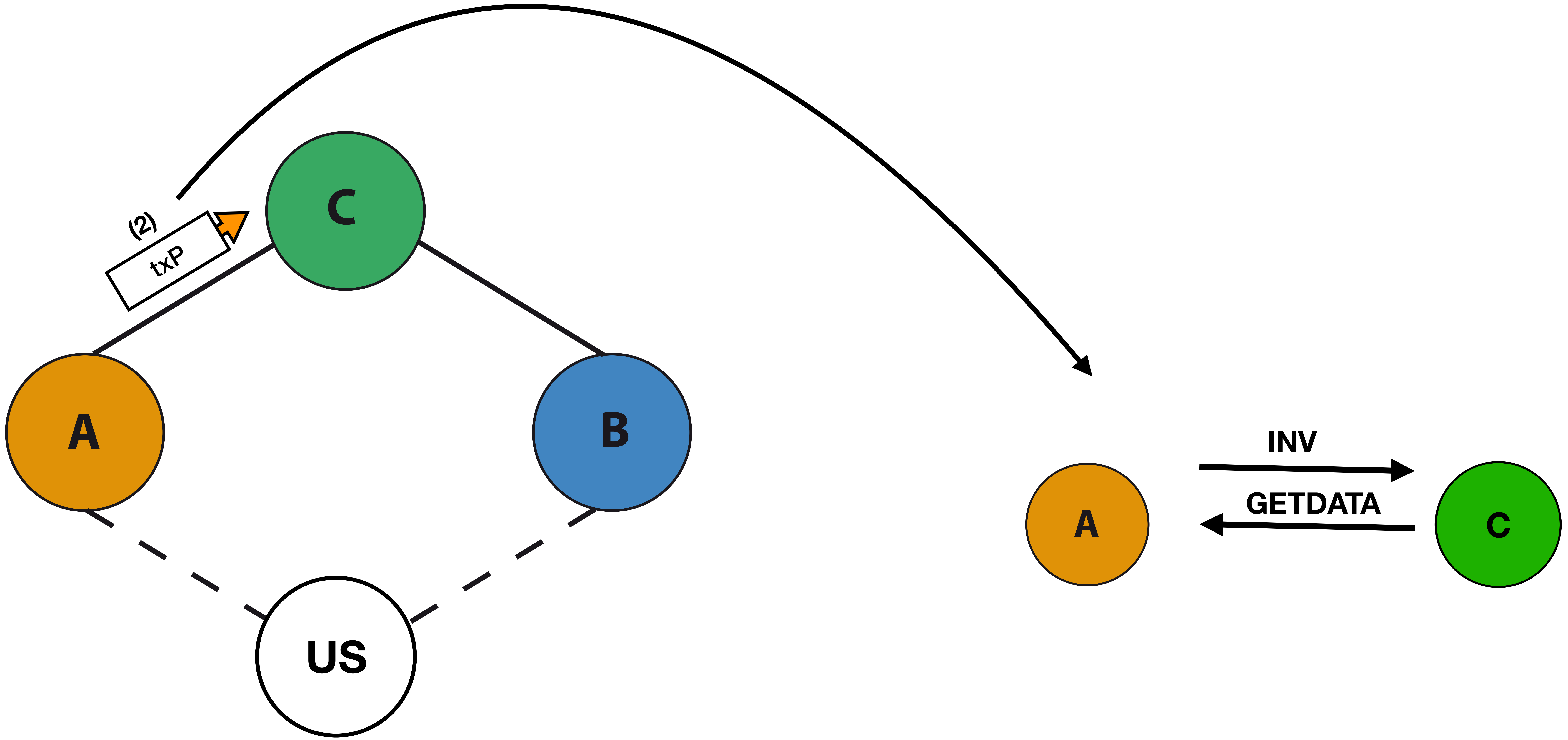
ISOLATION



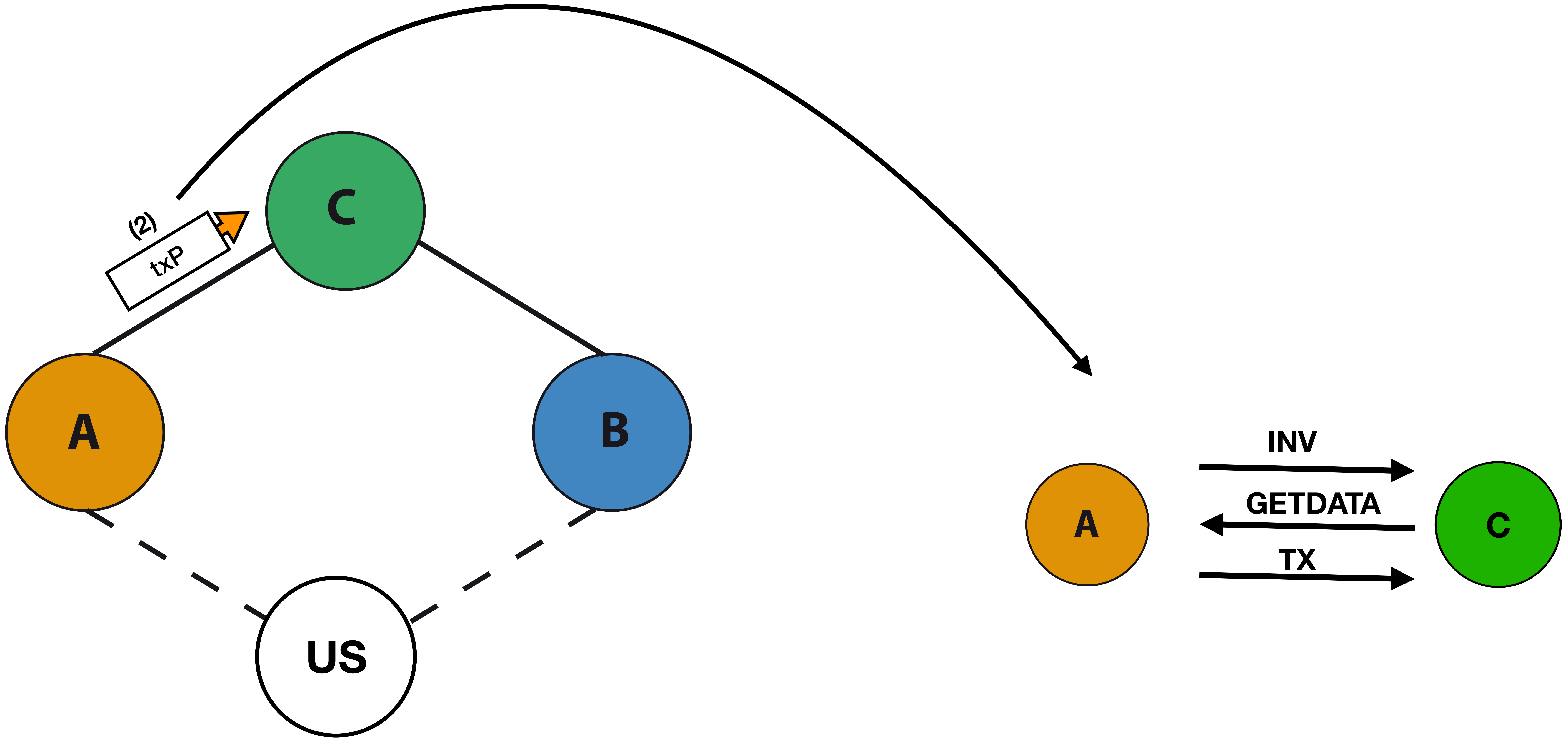
ISOLATION



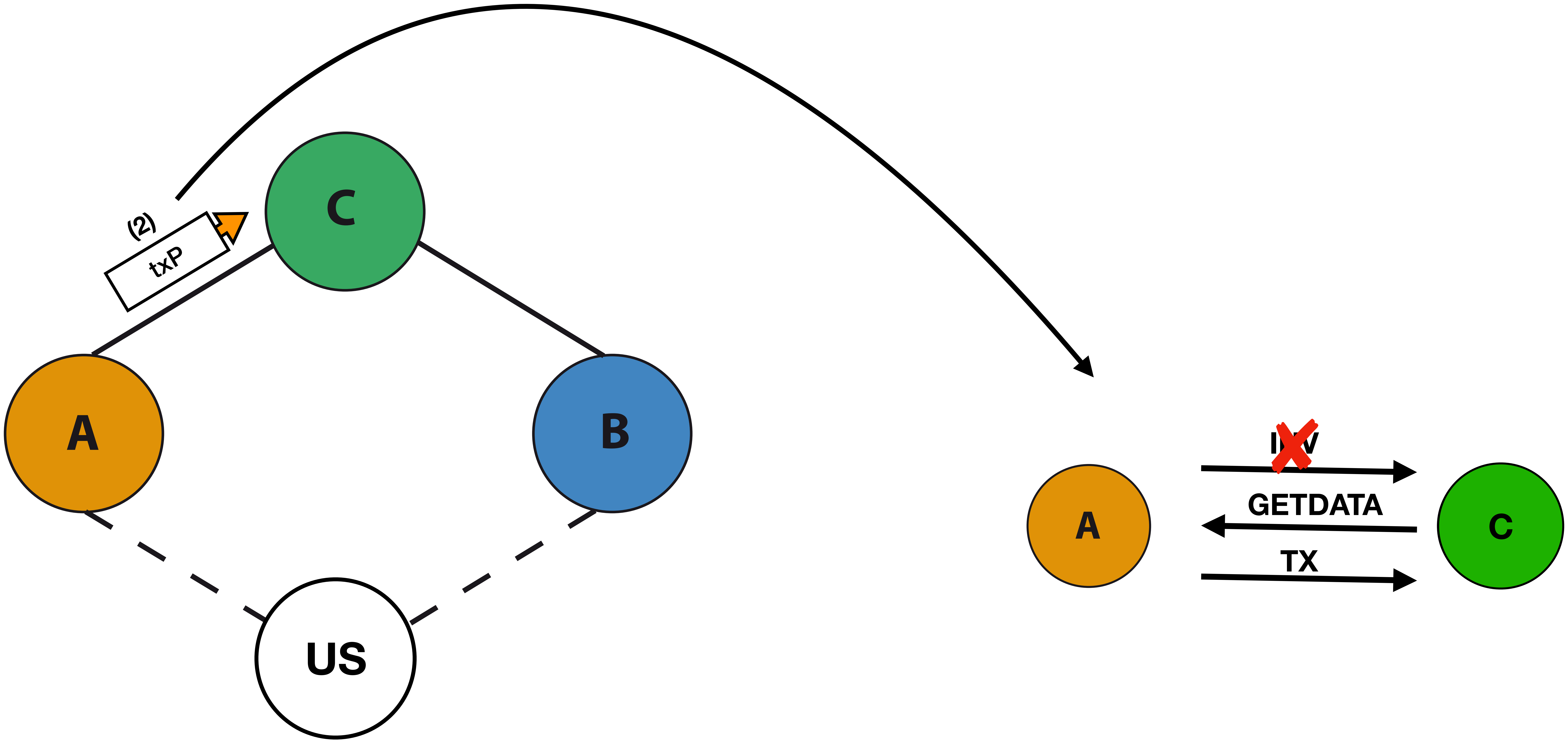
ISOLATION



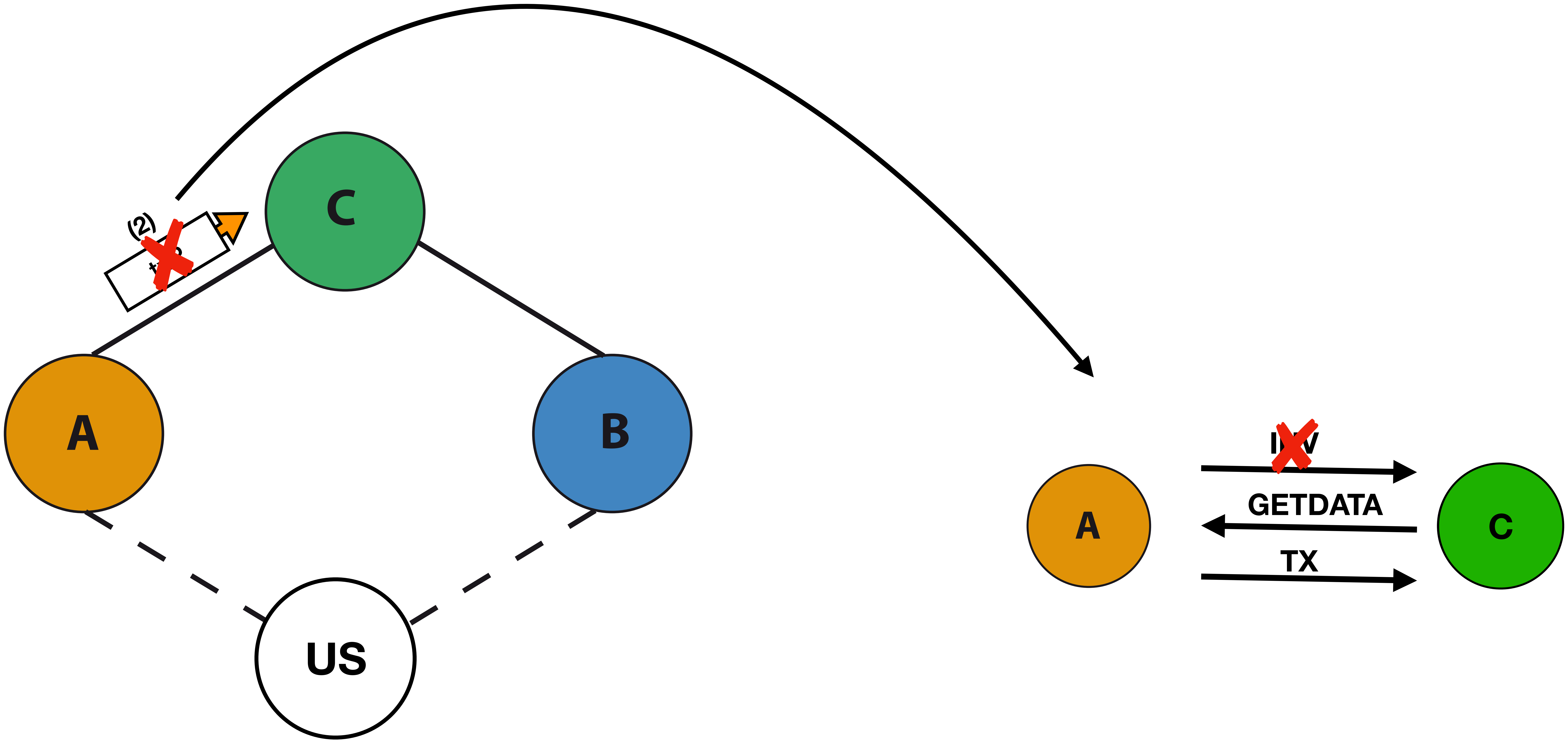
ISOLATION



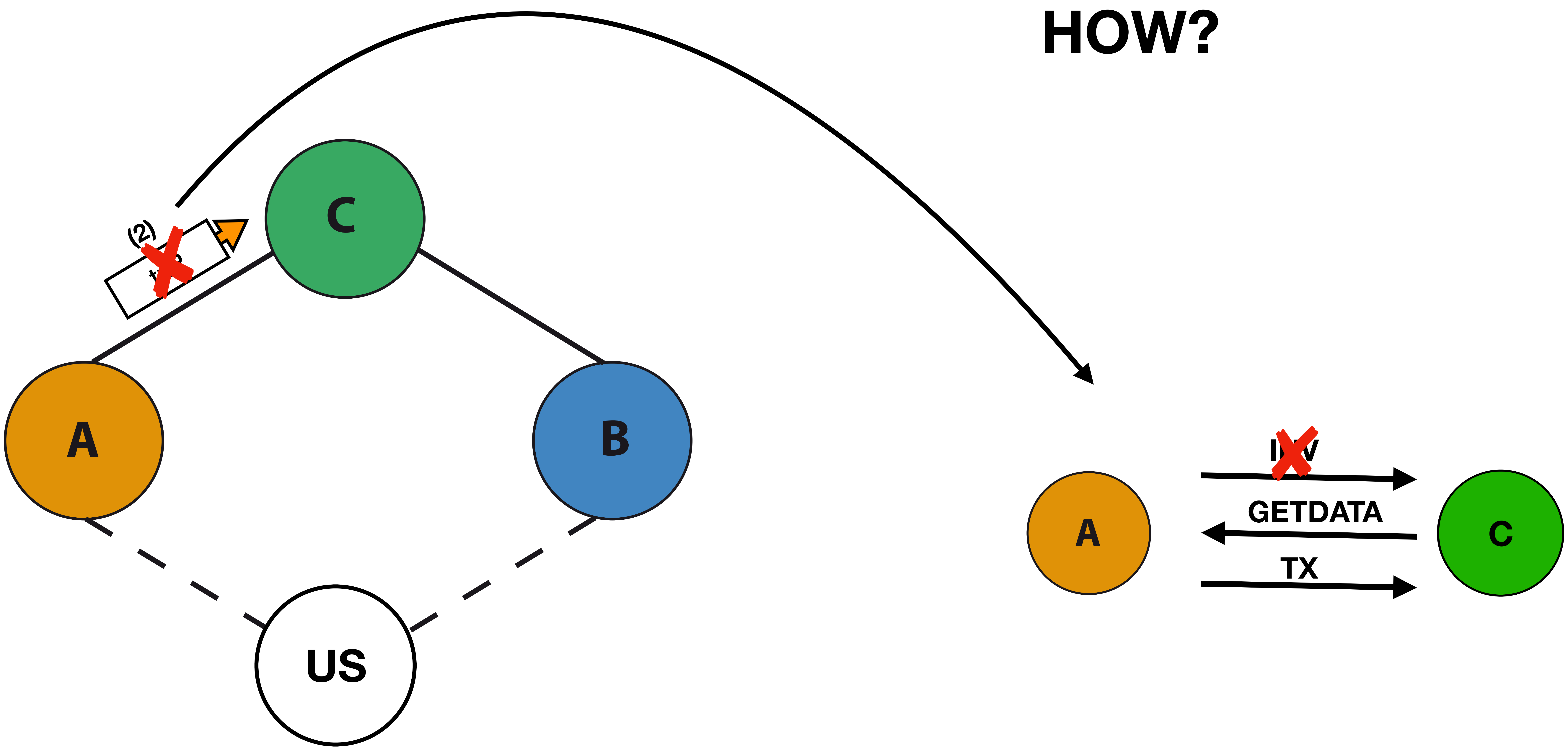
ISOLATION



ISOLATION

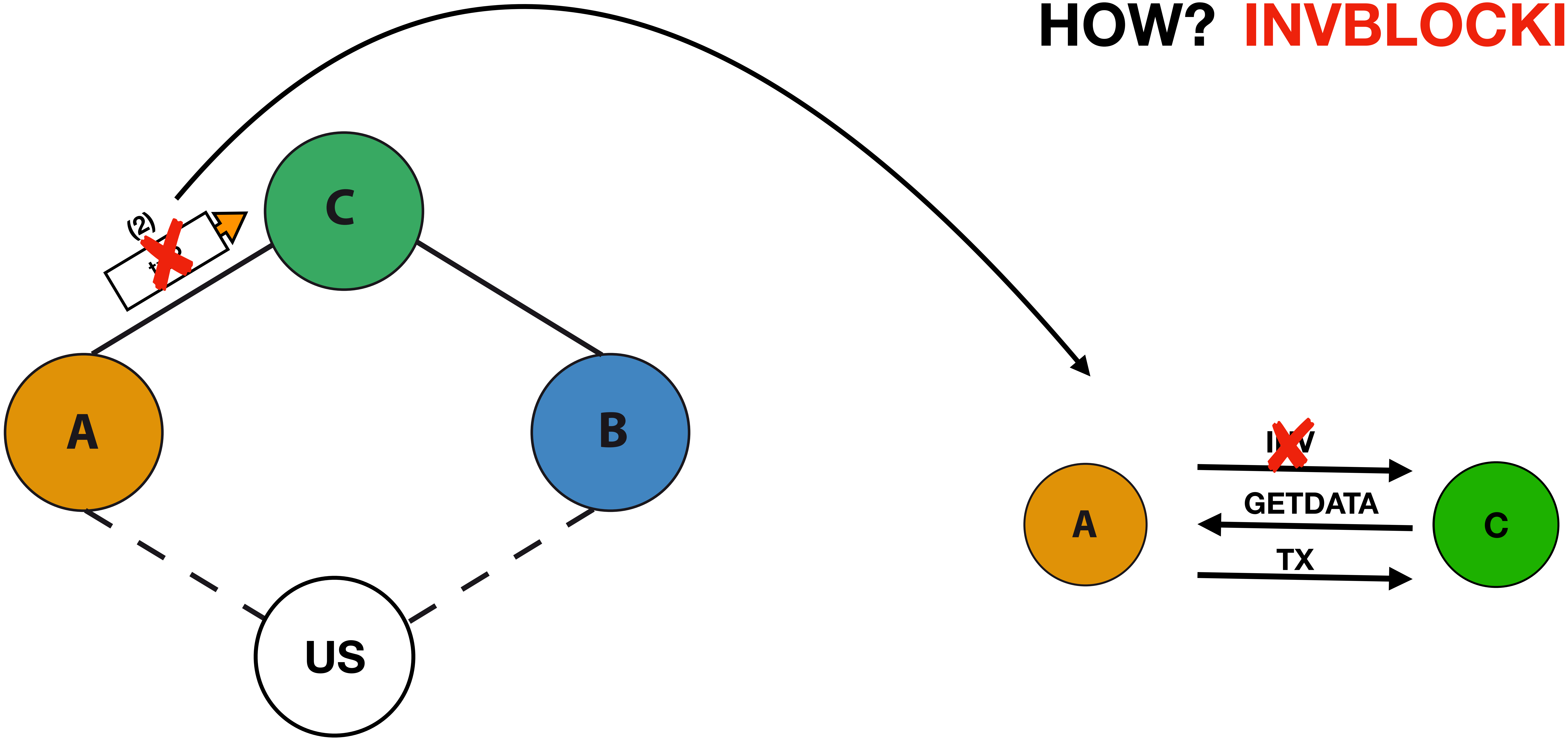


ISOLATION

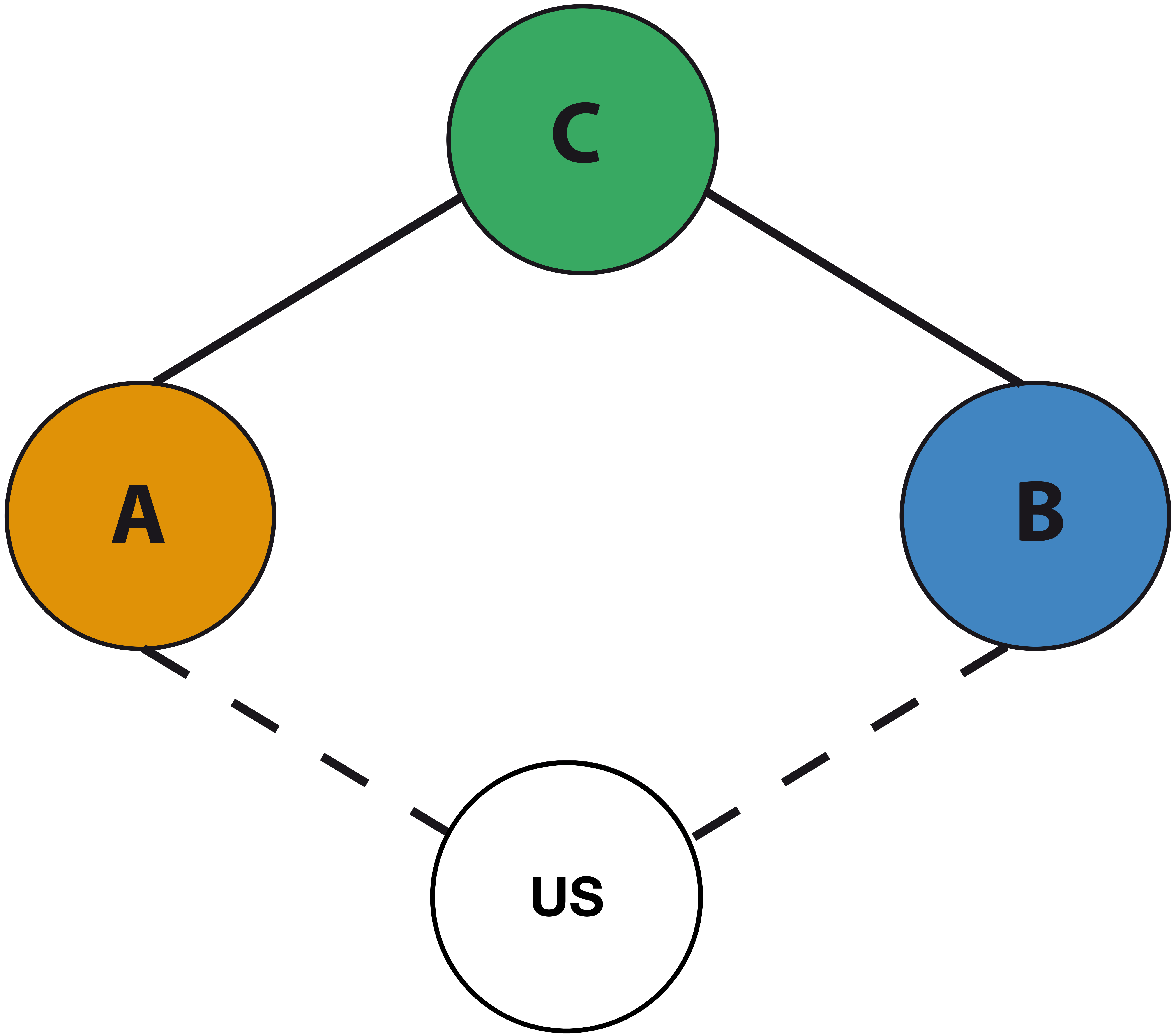


ISOLATION

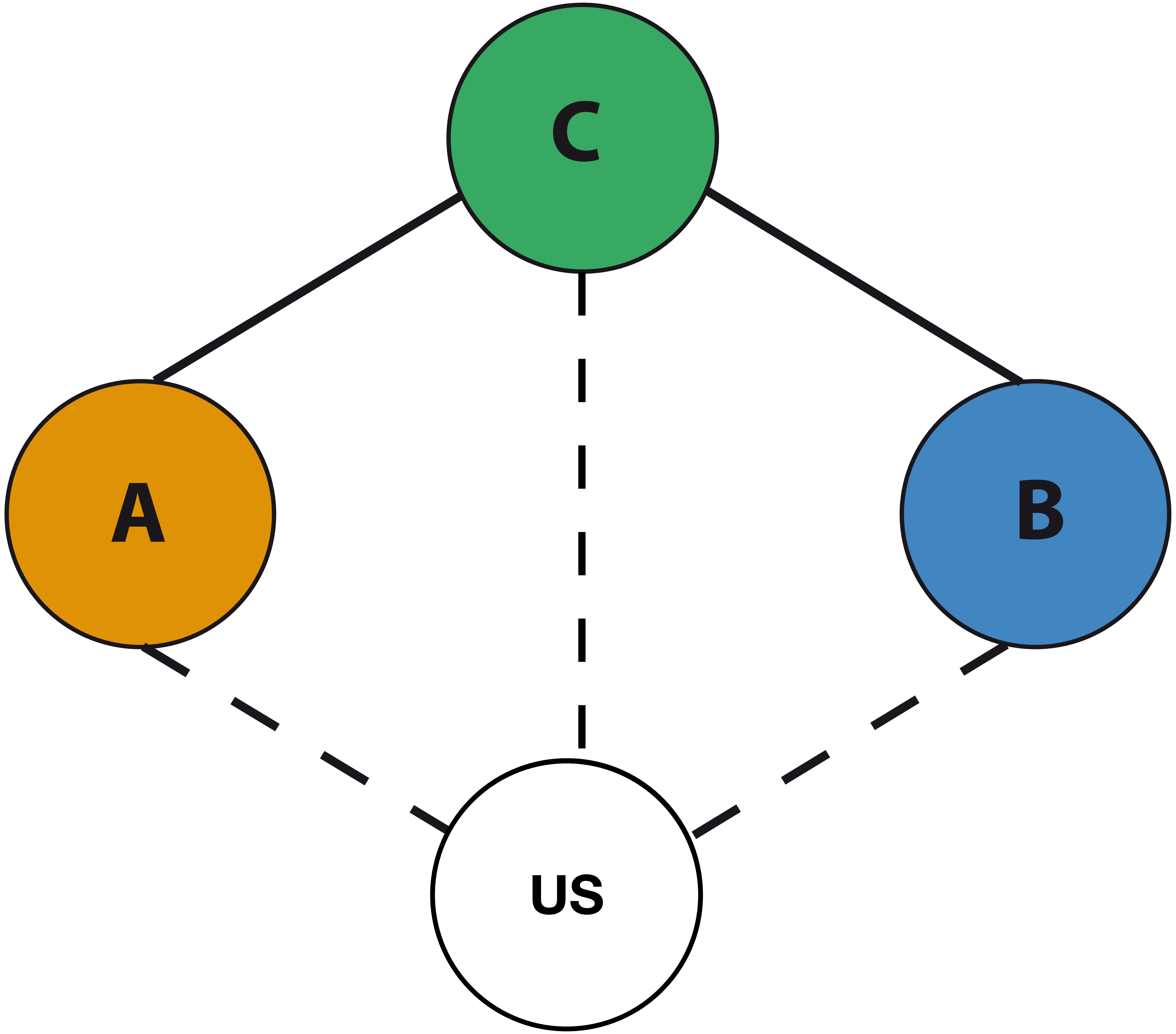
HOW? INVBLOCKING



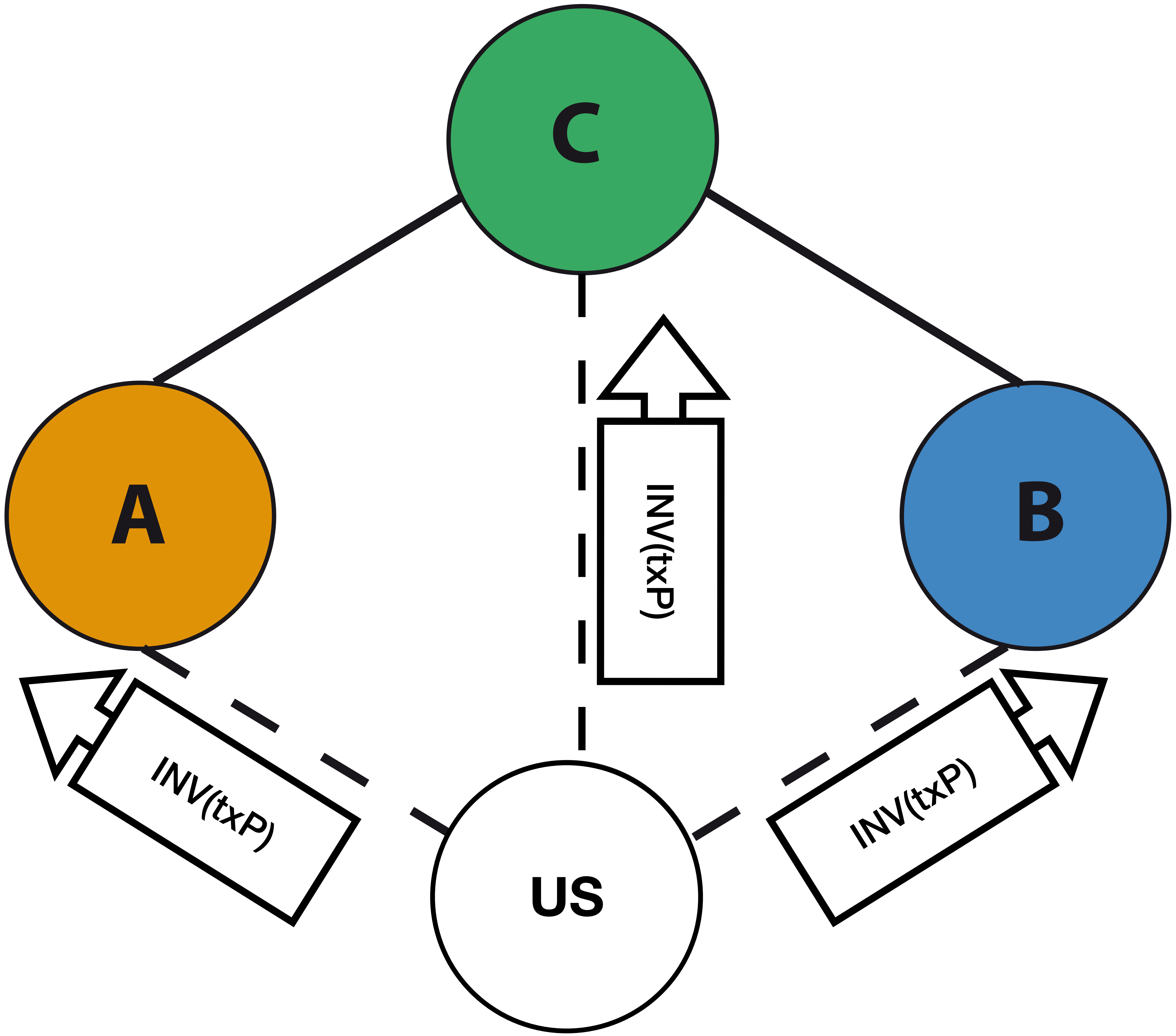
INVBLOCKING



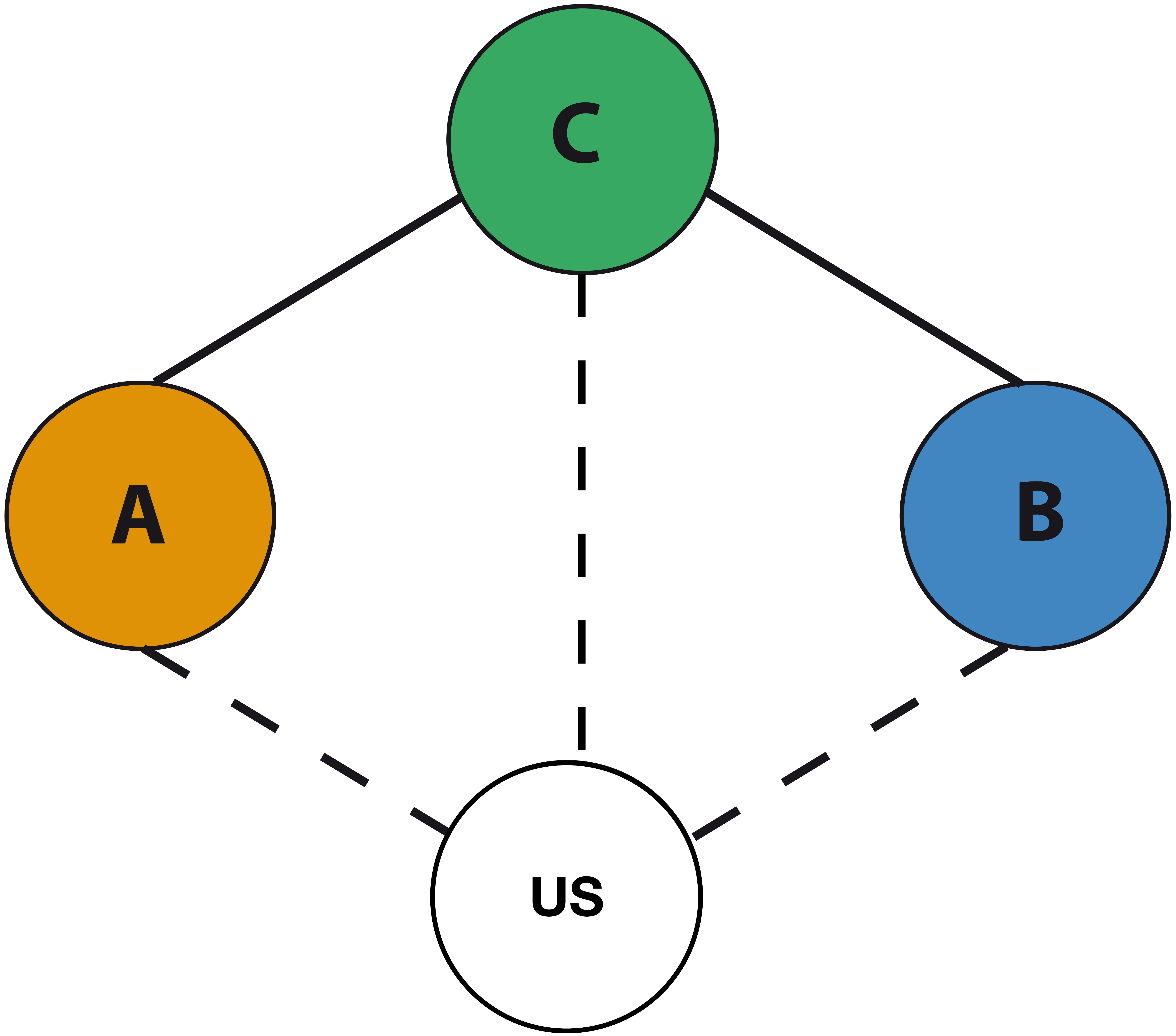
INVBLOCKING



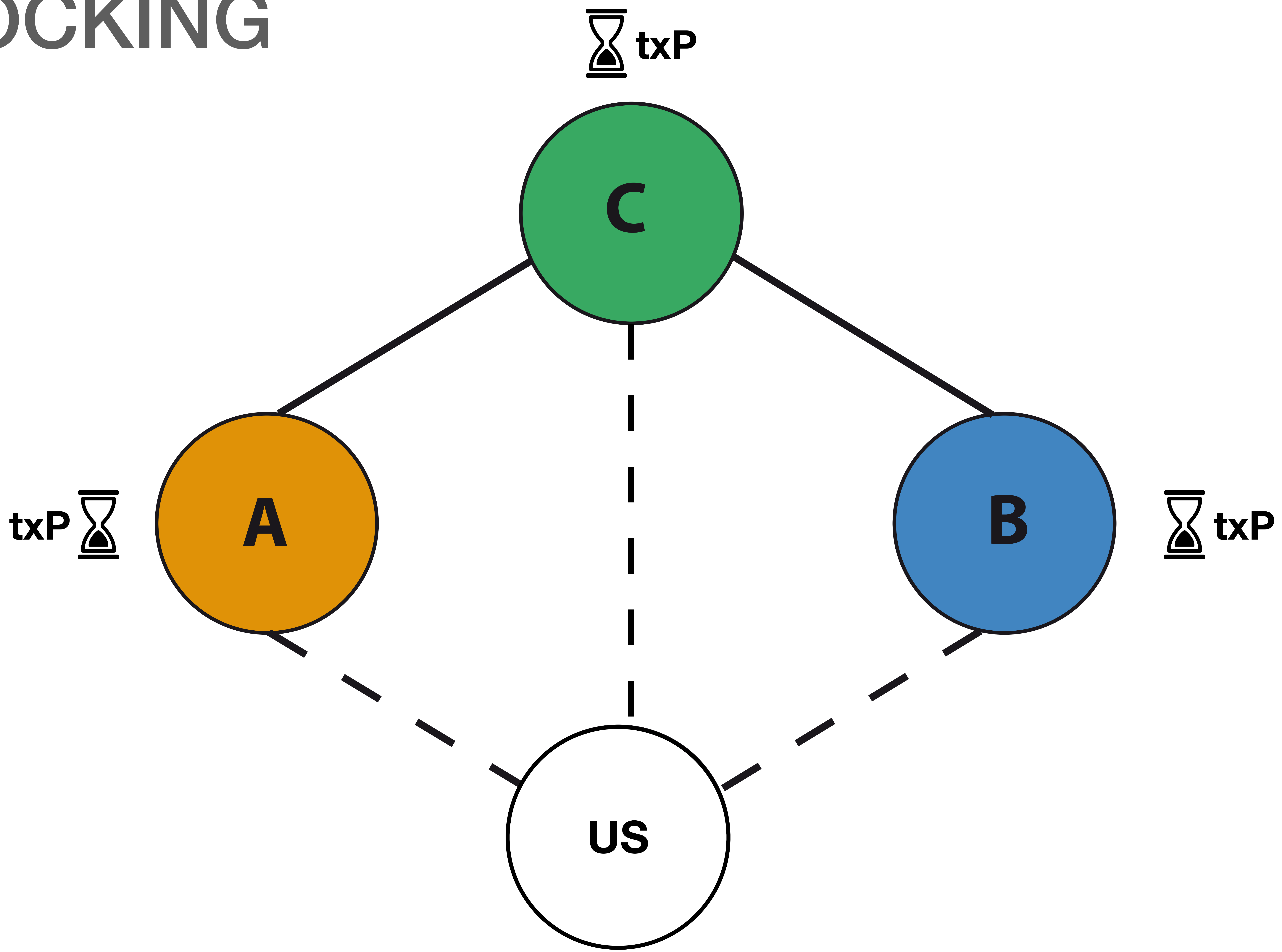
INVBLOCKING



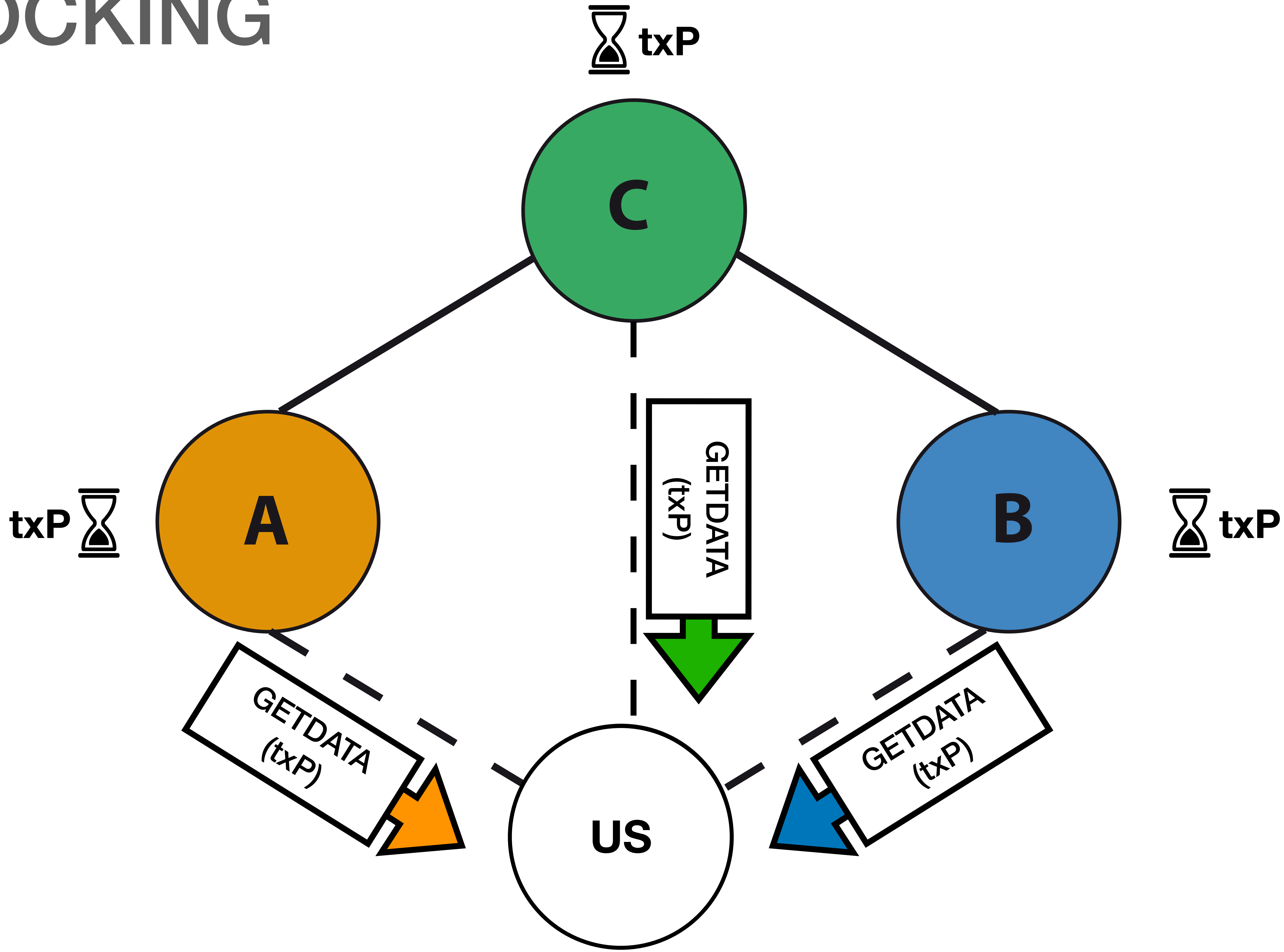
INVBLOCKING



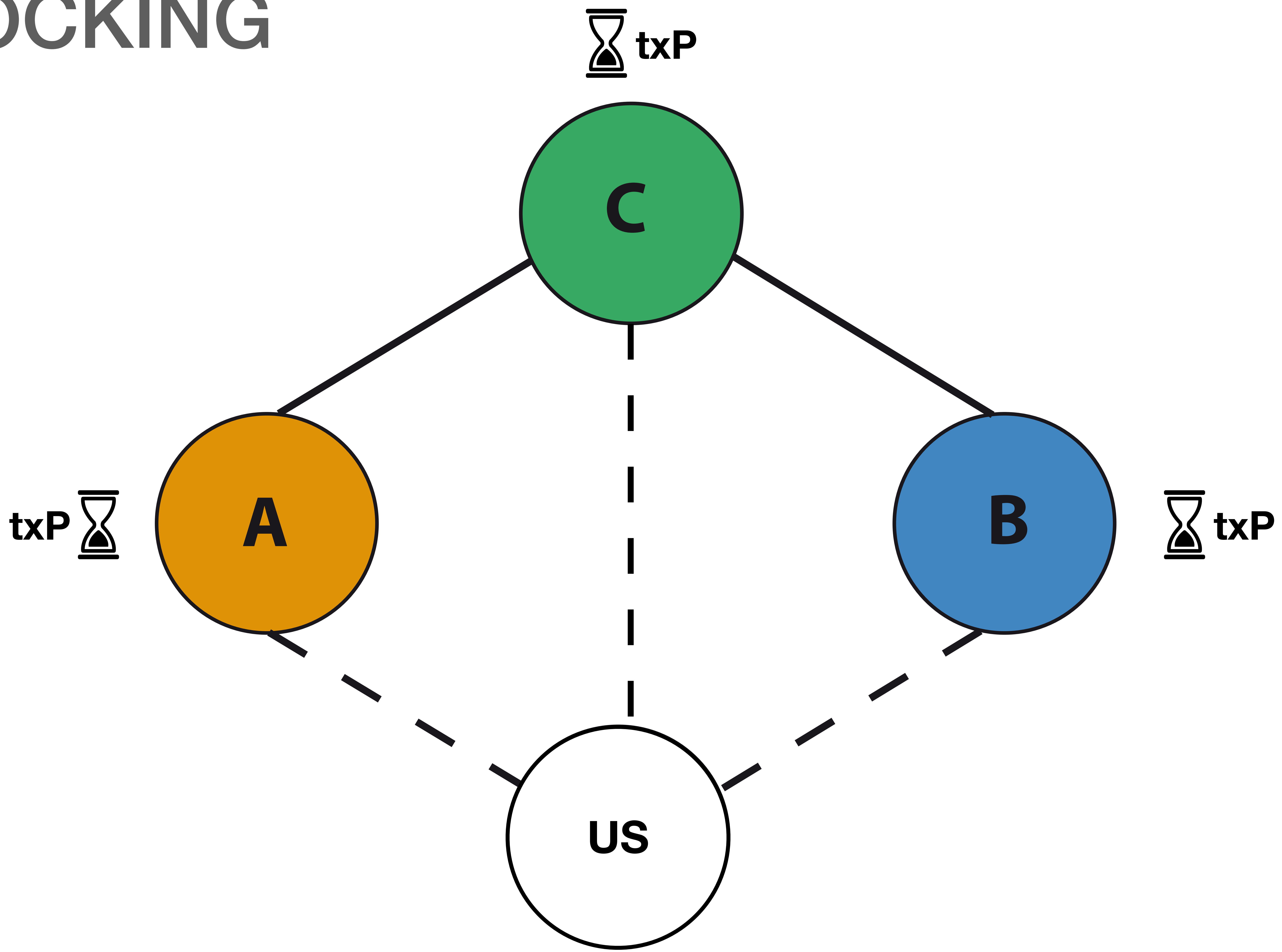
INVBLOCKING



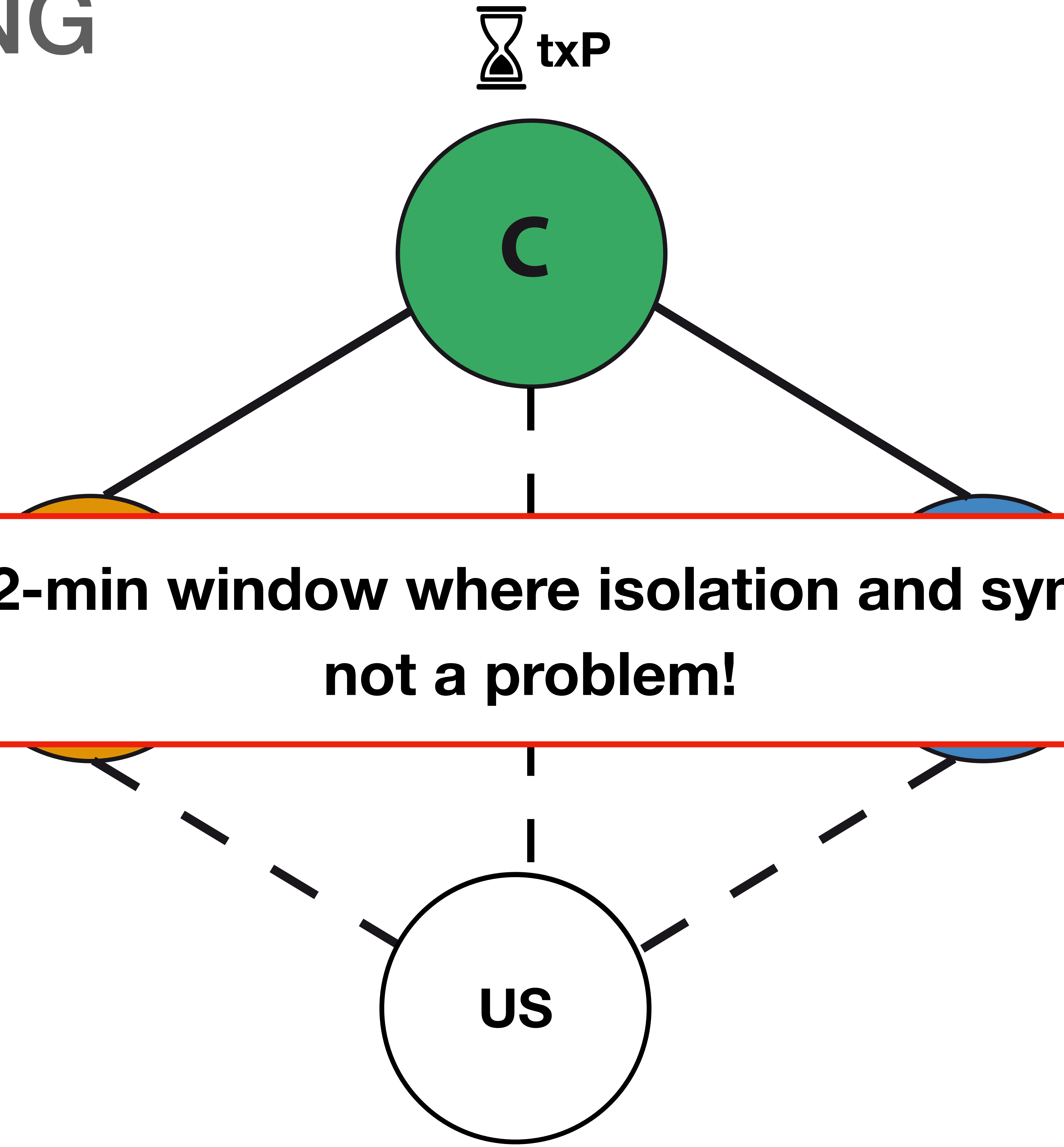
INVBLOCKING



INVBLOCKING

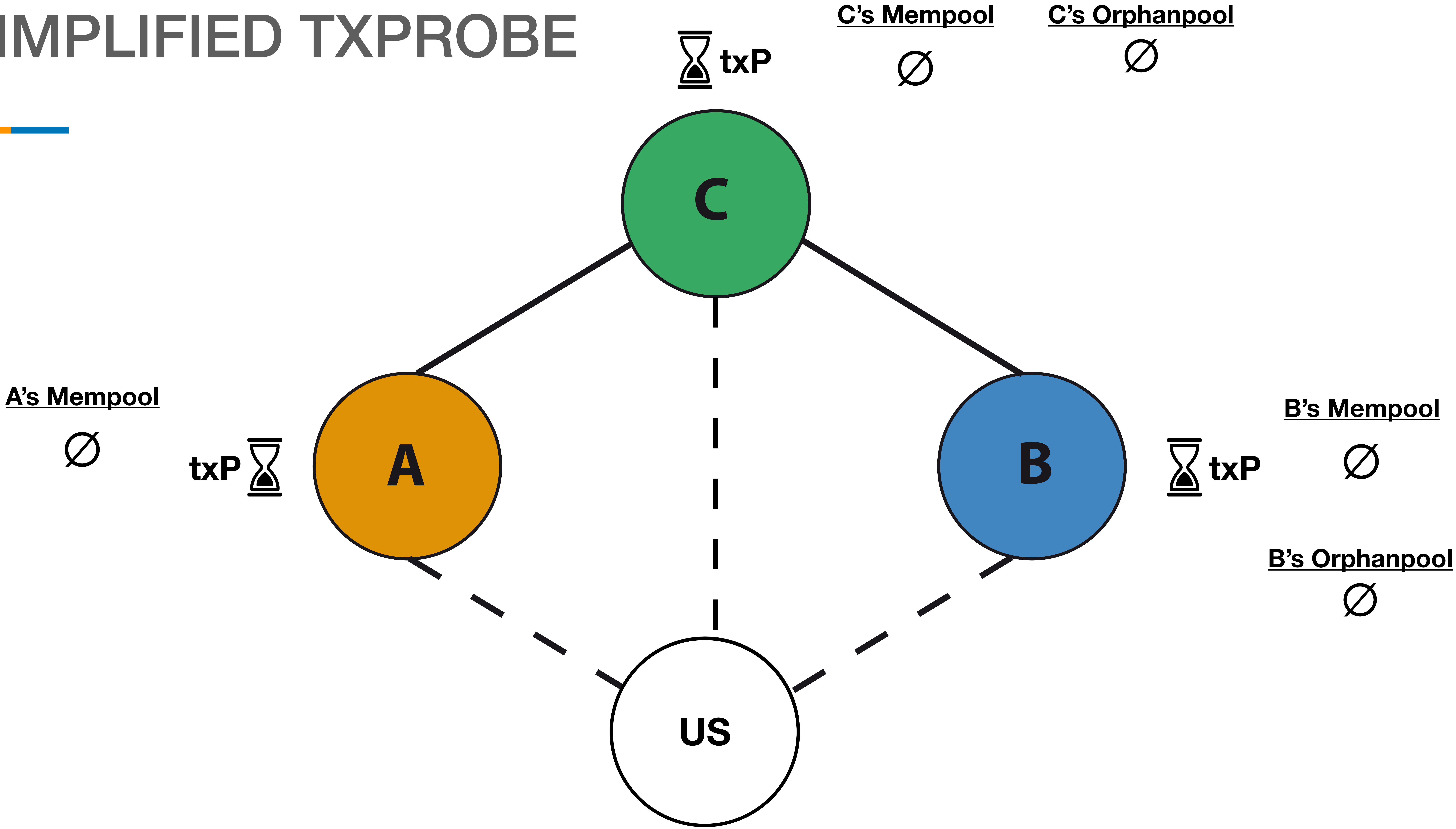


INVBLOCKING

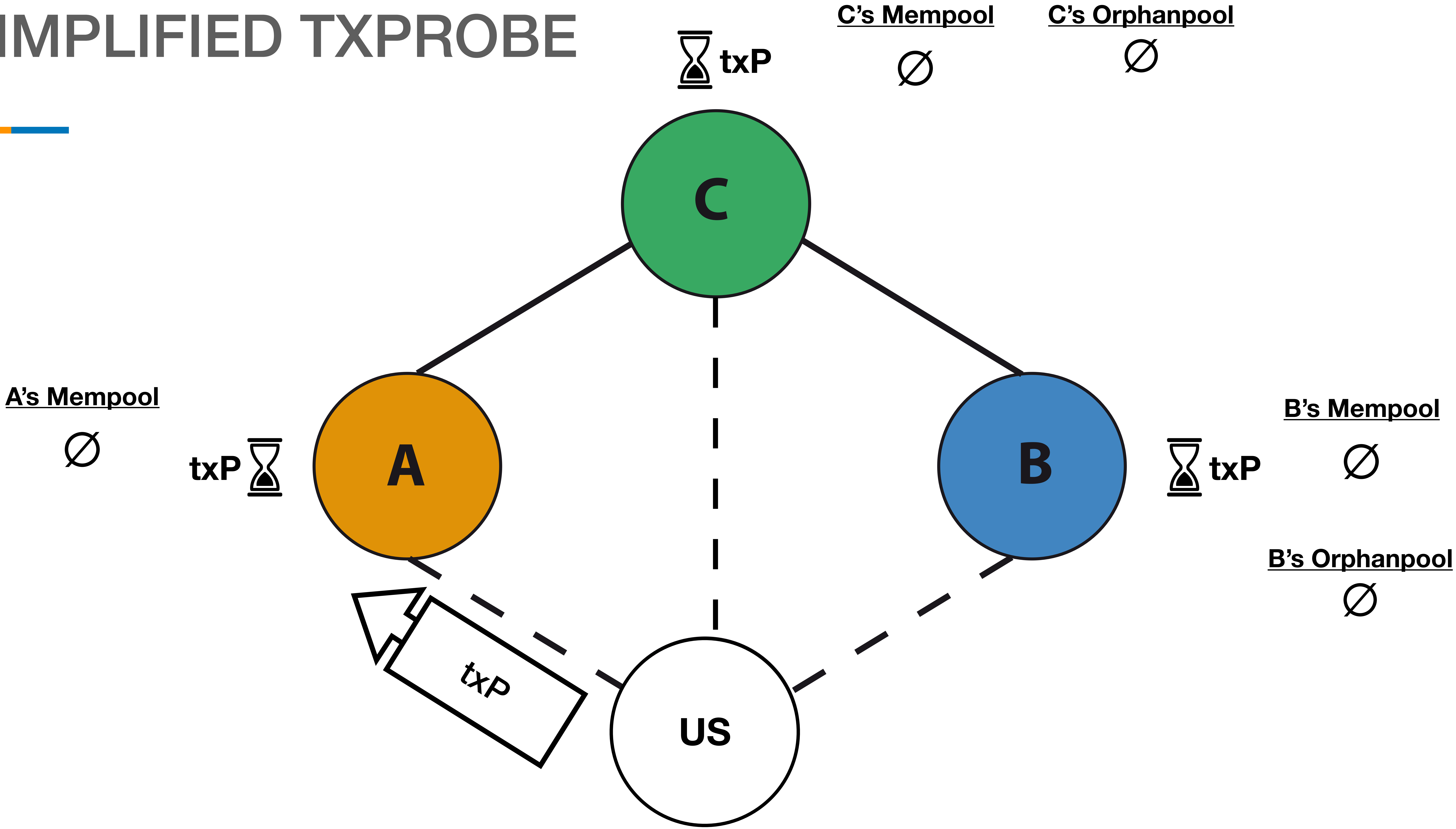


We have a 2-min window where isolation and synchrony are not a problem!

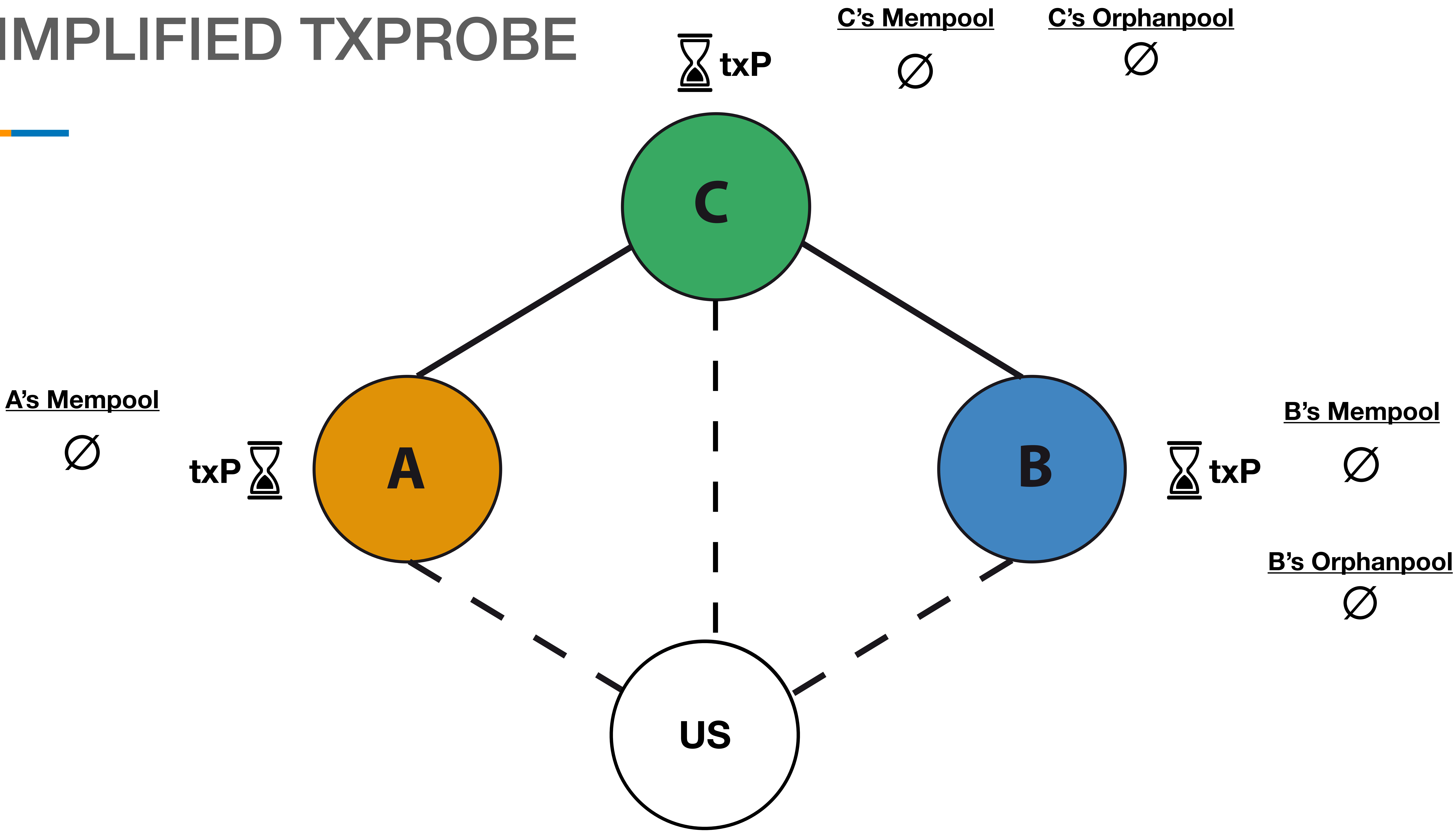
SIMPLIFIED TXPROBE



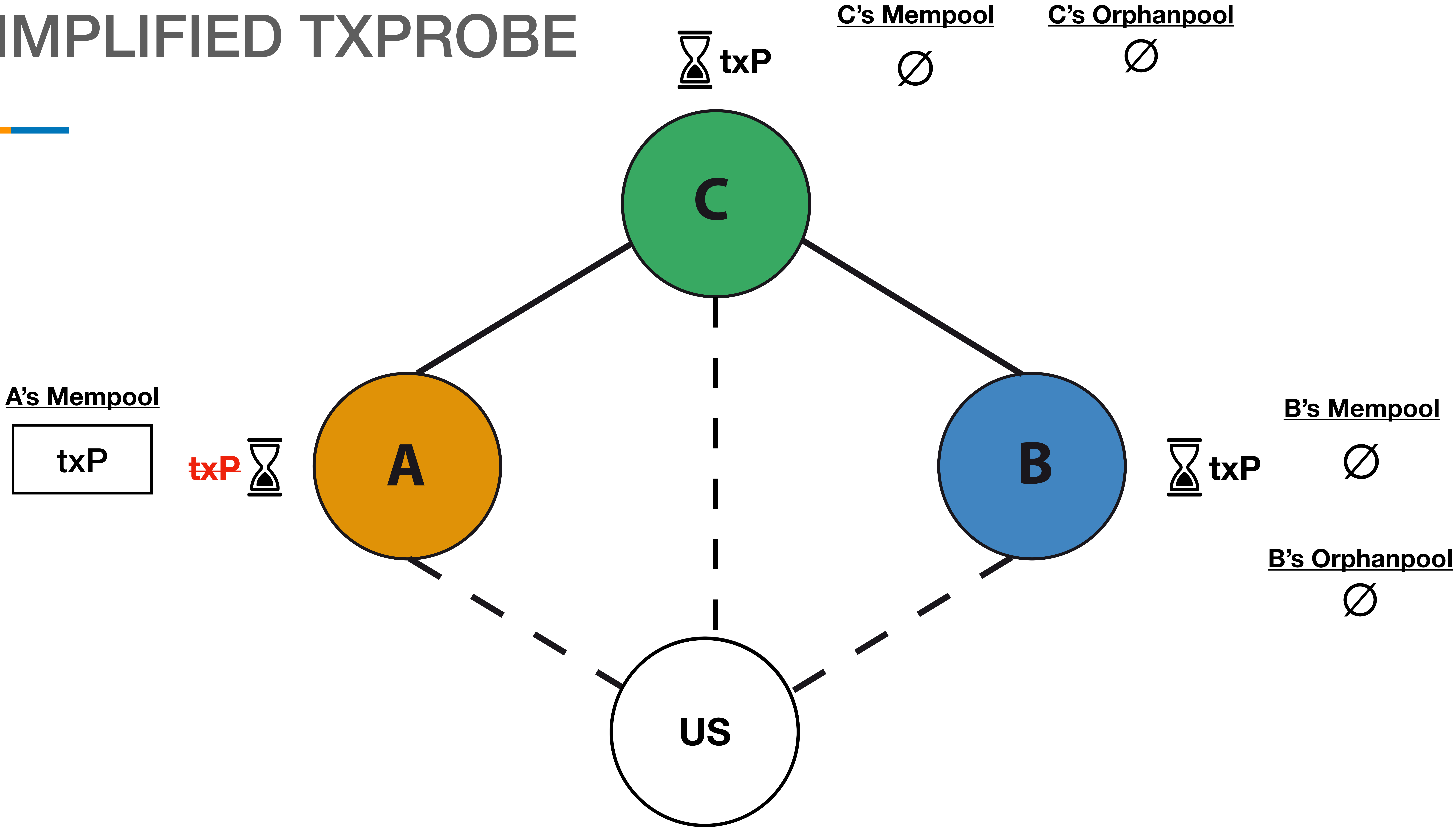
SIMPLIFIED TXPROBE



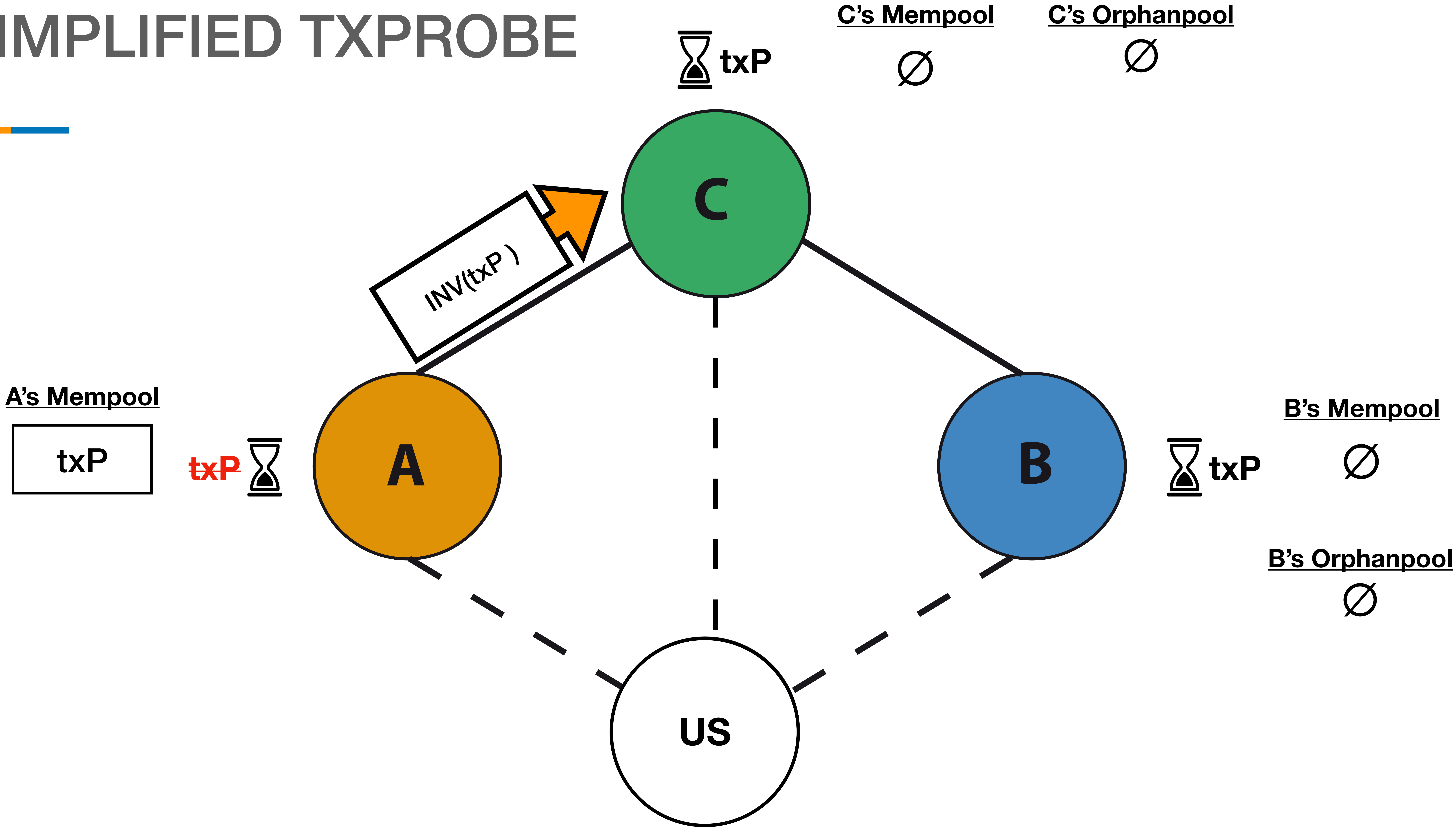
SIMPLIFIED TXPROBE



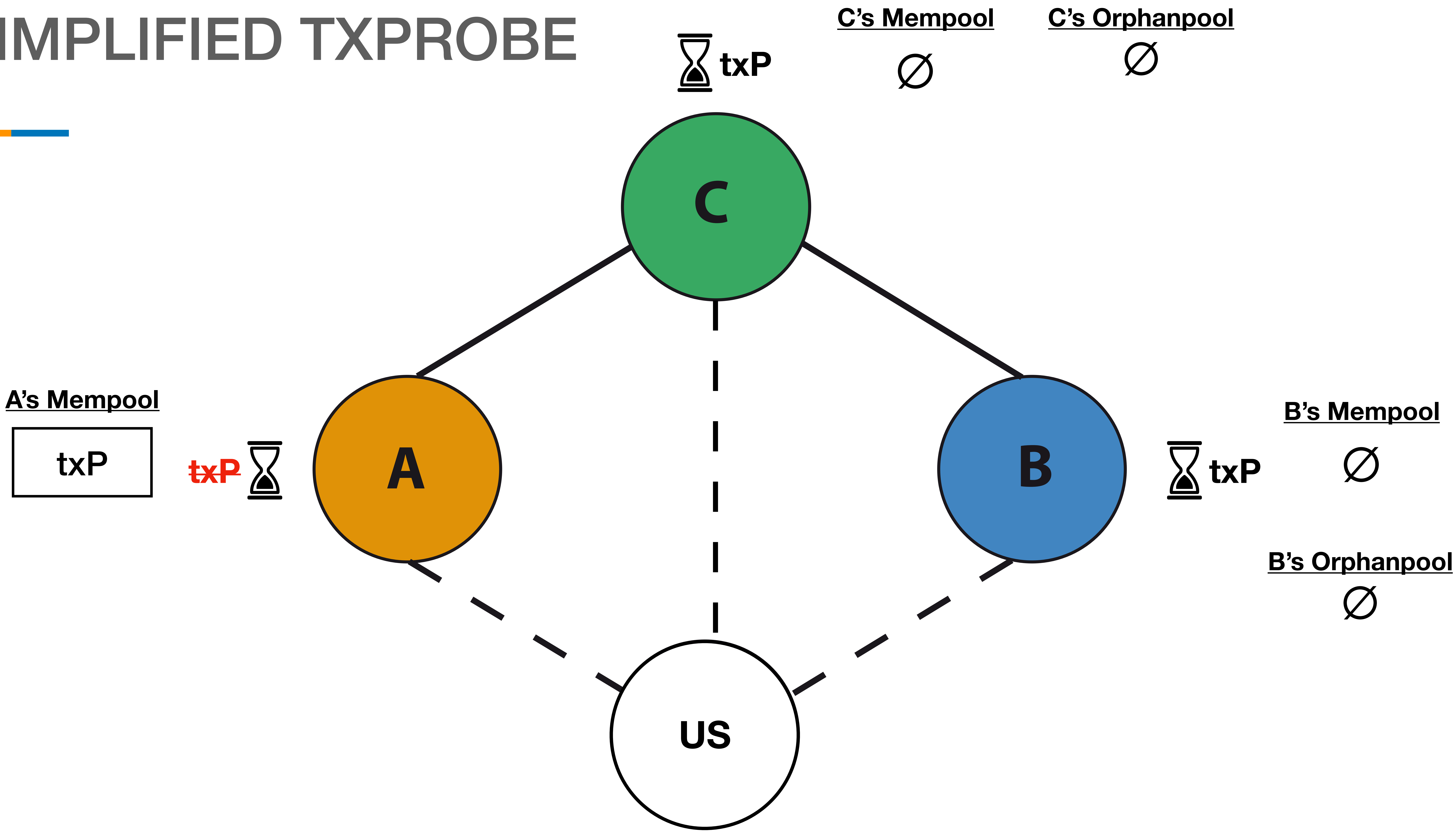
SIMPLIFIED TXPROBE



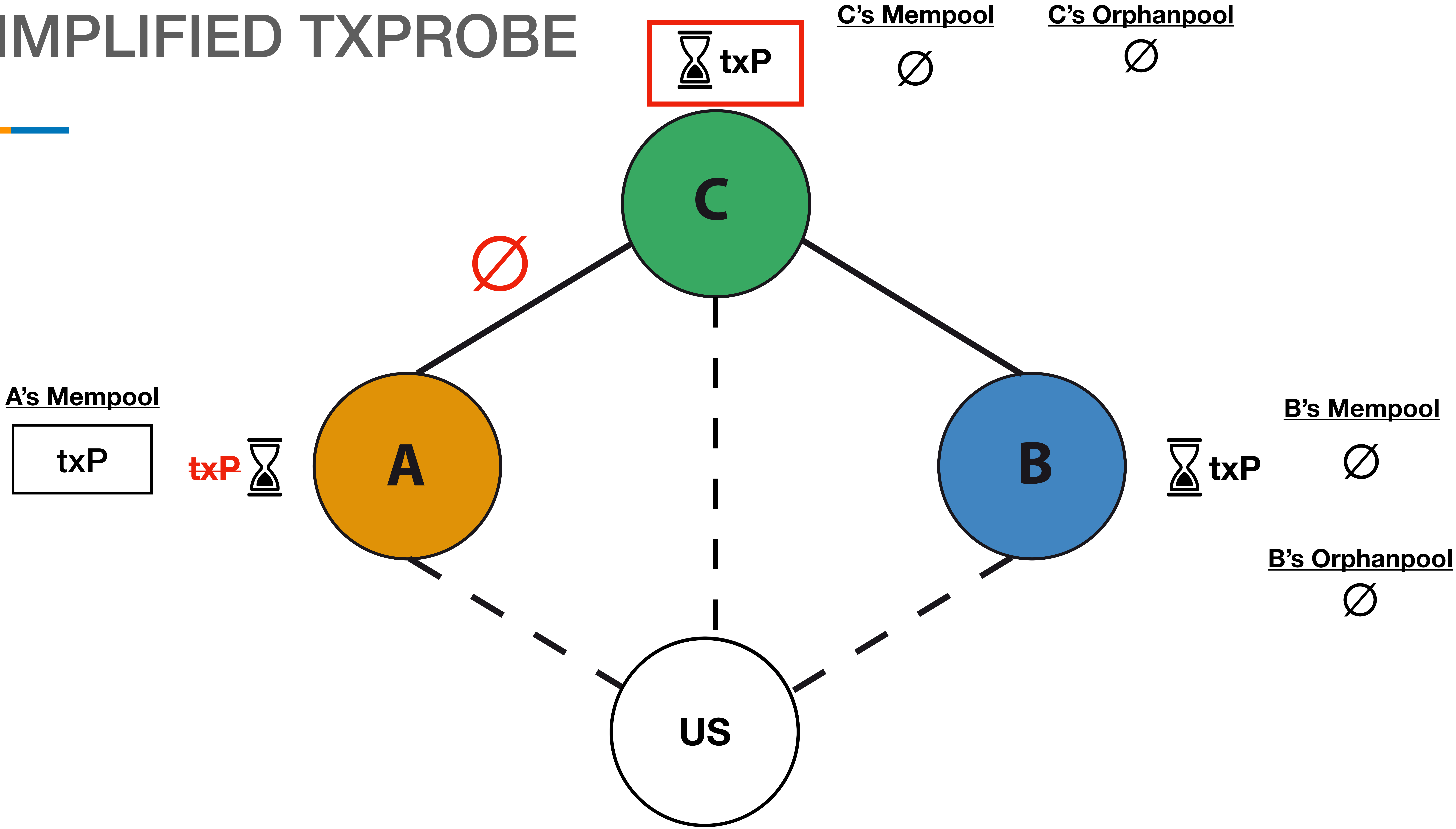
SIMPLIFIED TXPROBE



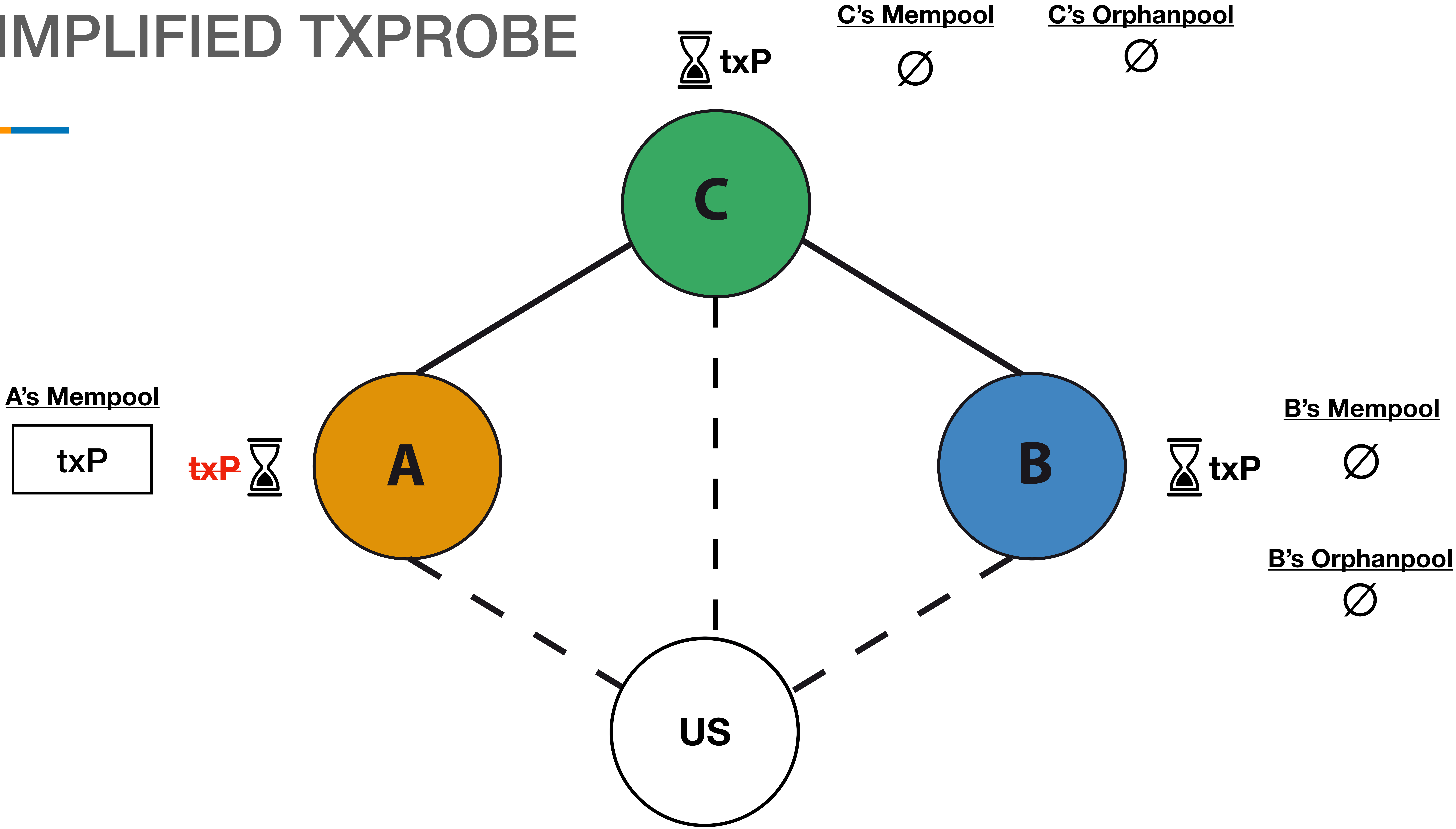
SIMPLIFIED TXPROBE



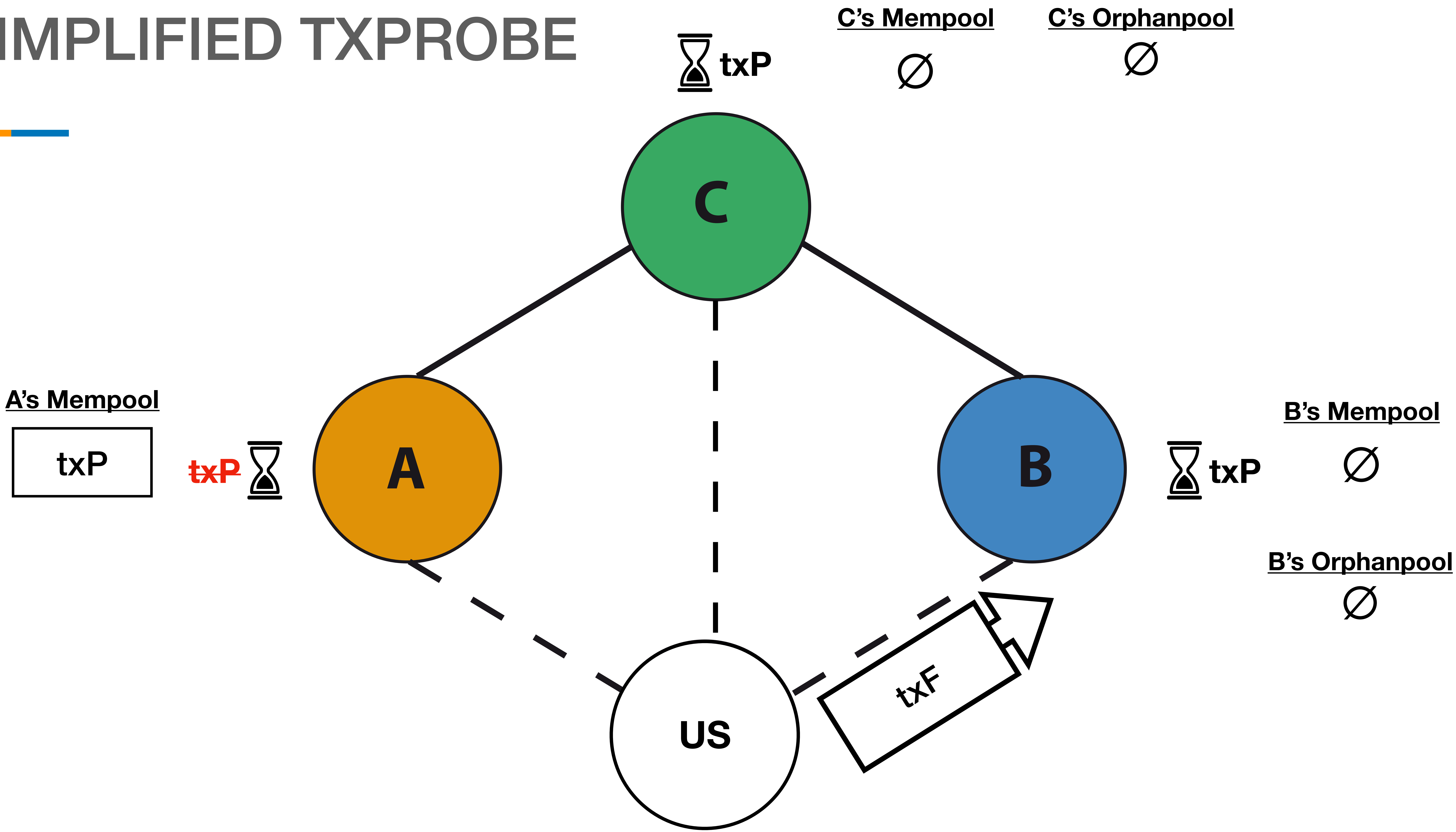
SIMPLIFIED TXPROBE



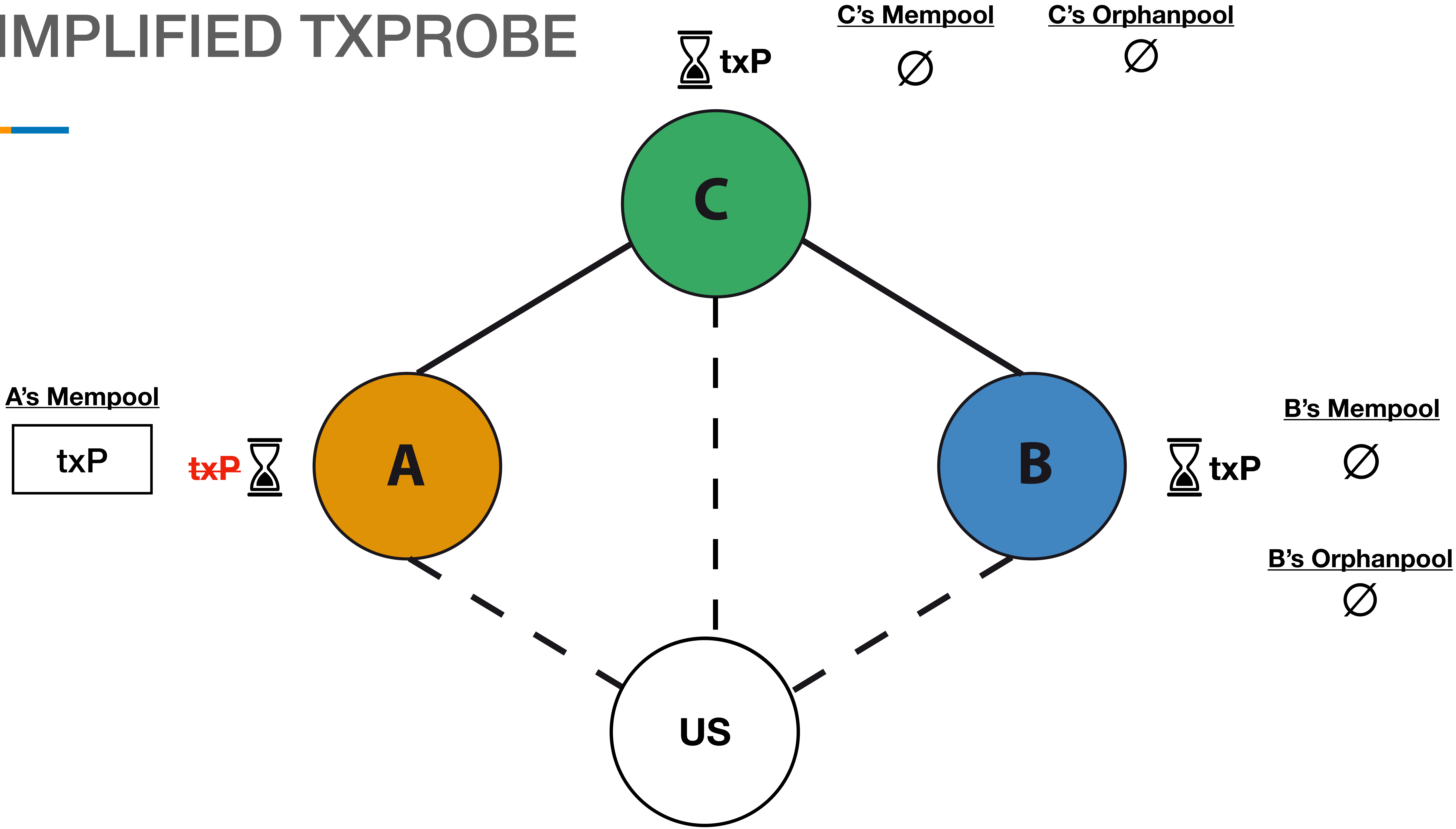
SIMPLIFIED TXPROBE



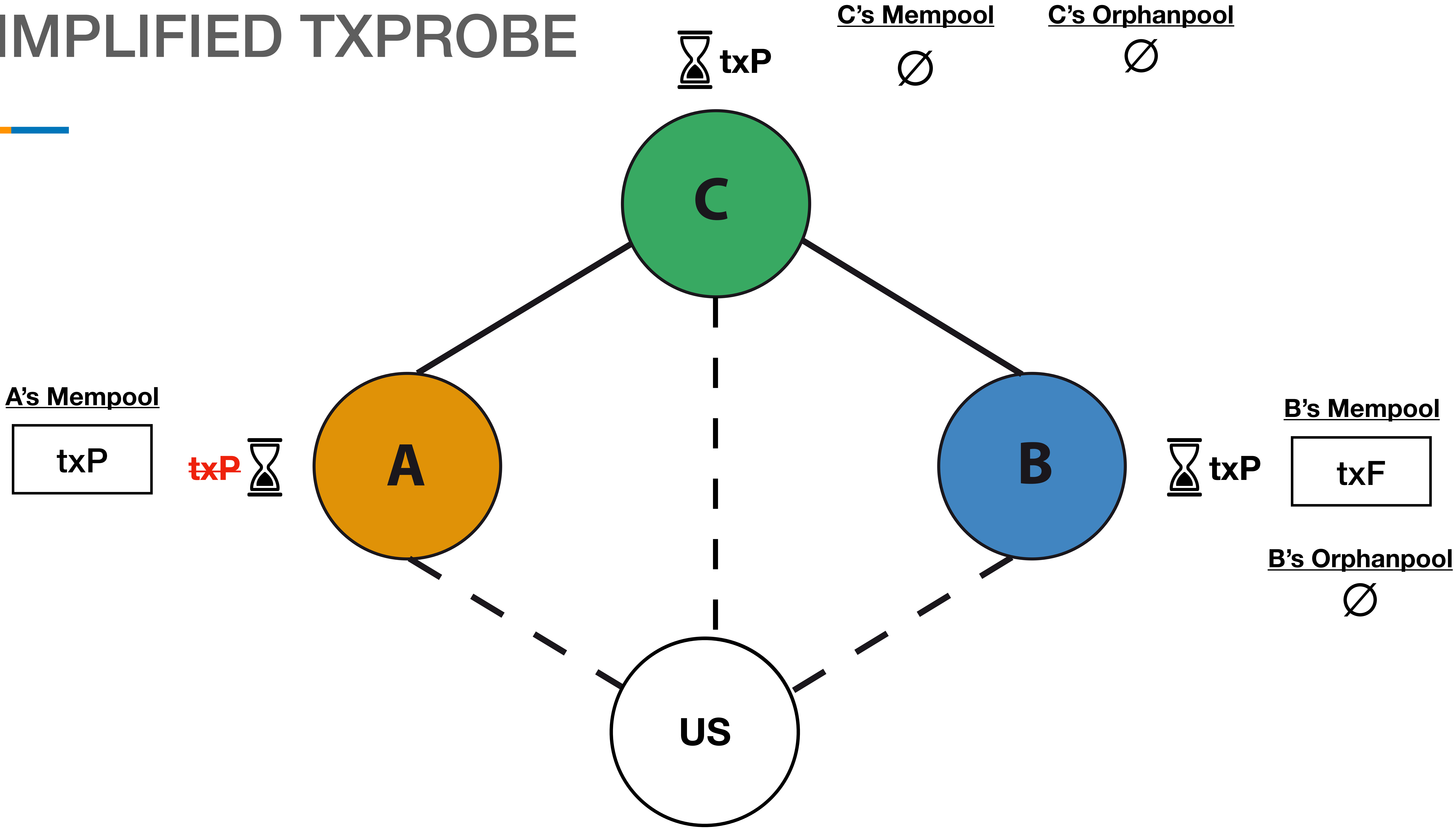
SIMPLIFIED TXPROBE



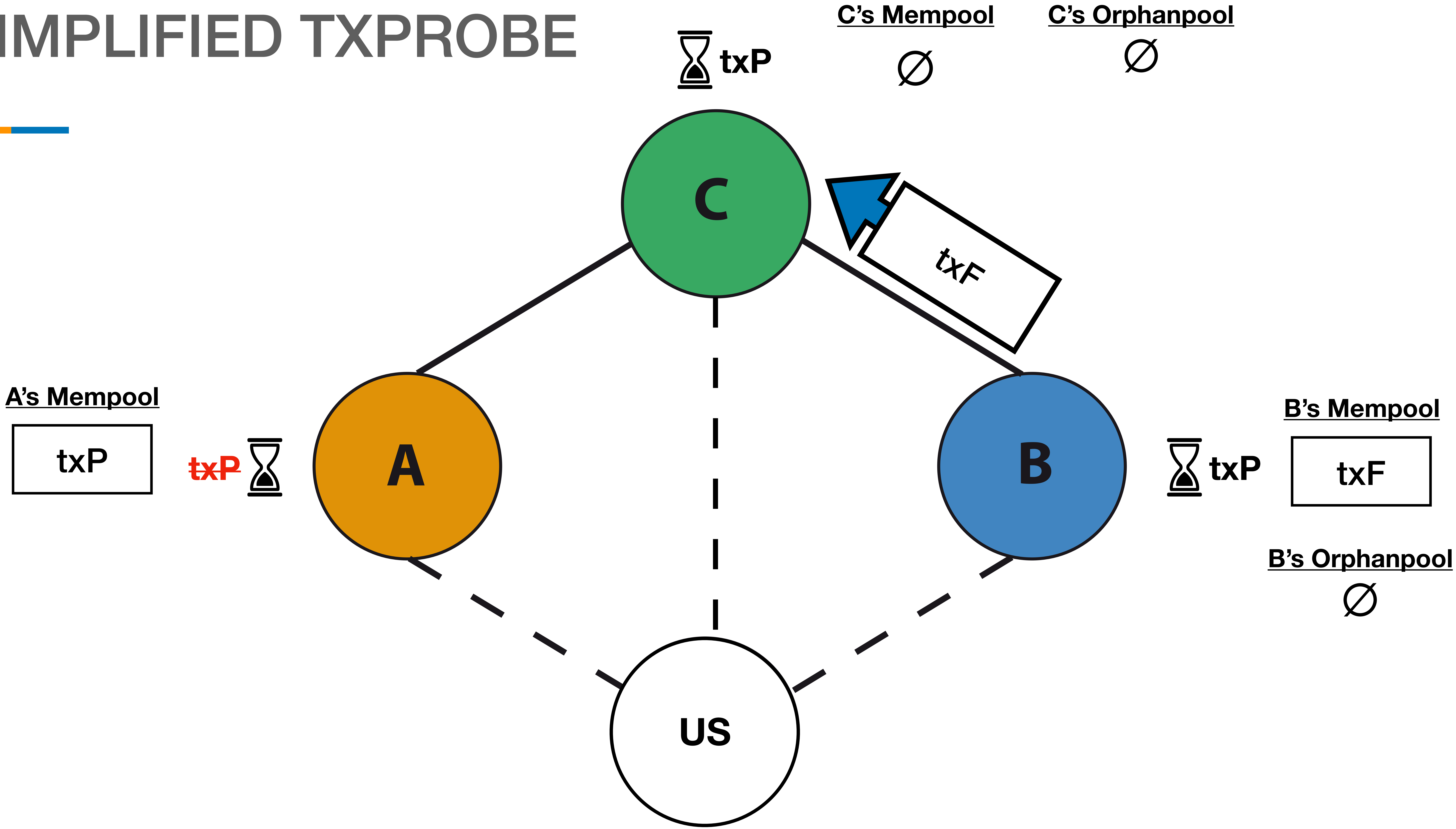
SIMPLIFIED TXPROBE



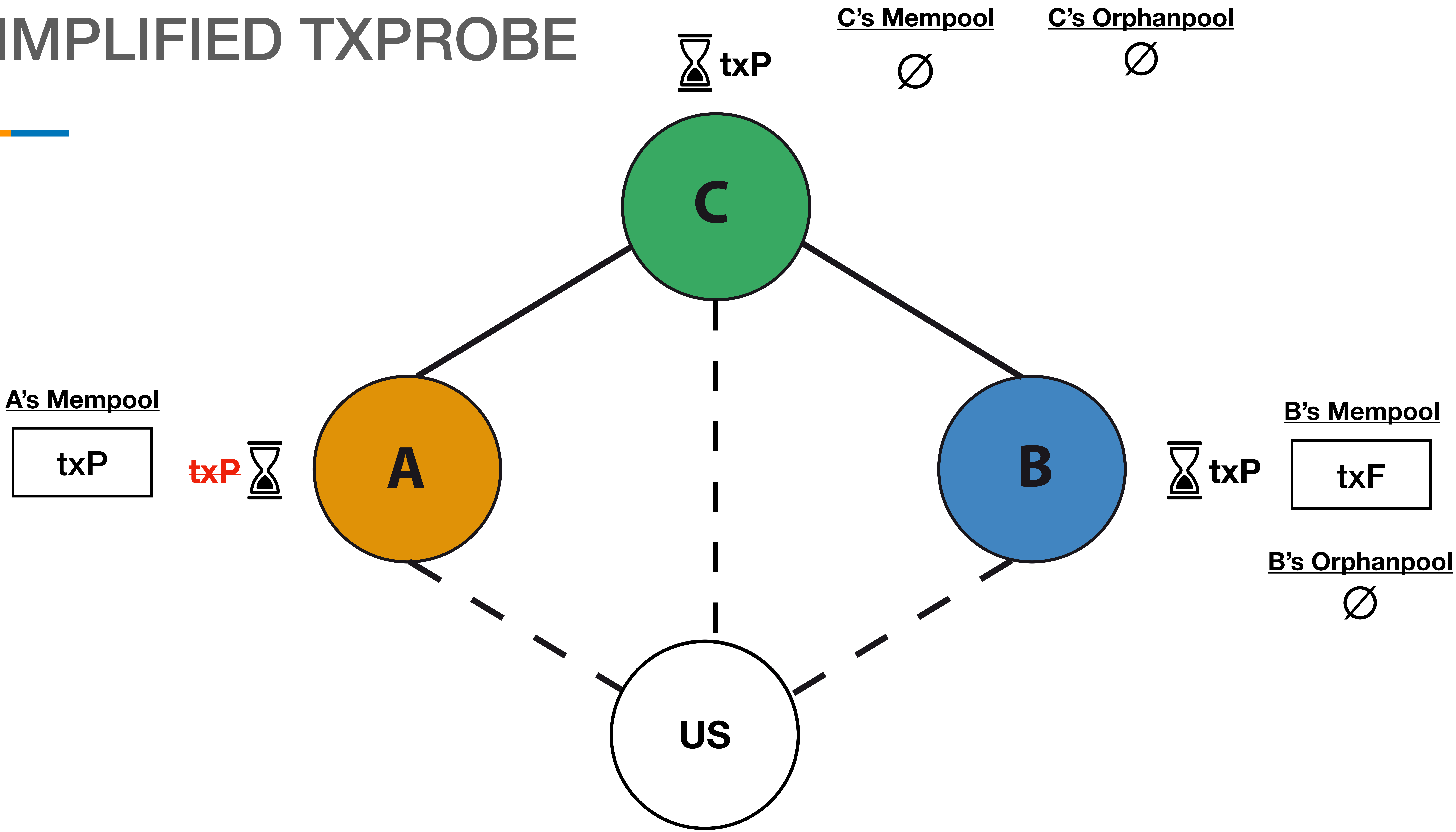
SIMPLIFIED TXPROBE



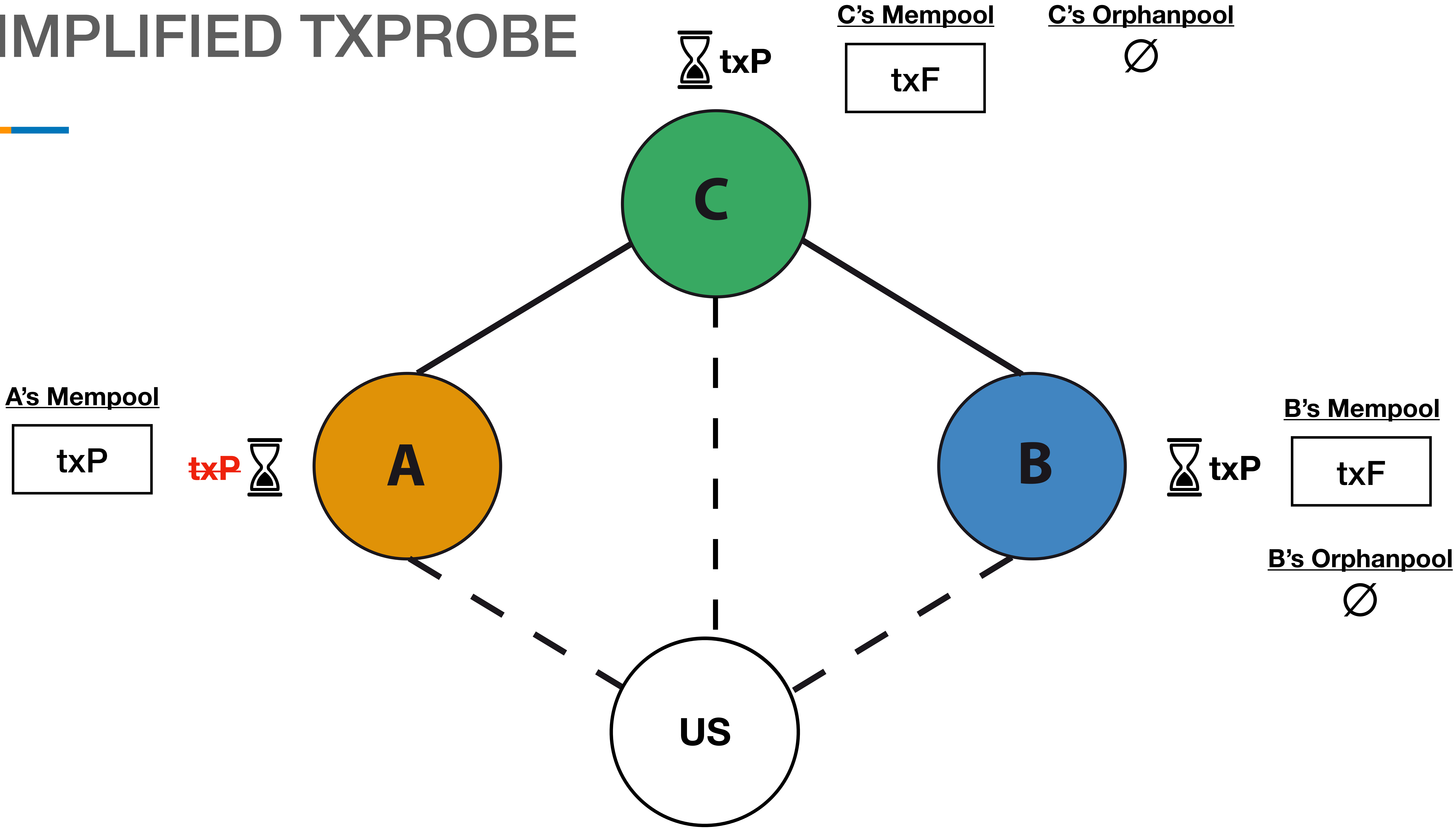
SIMPLIFIED TXPROBE



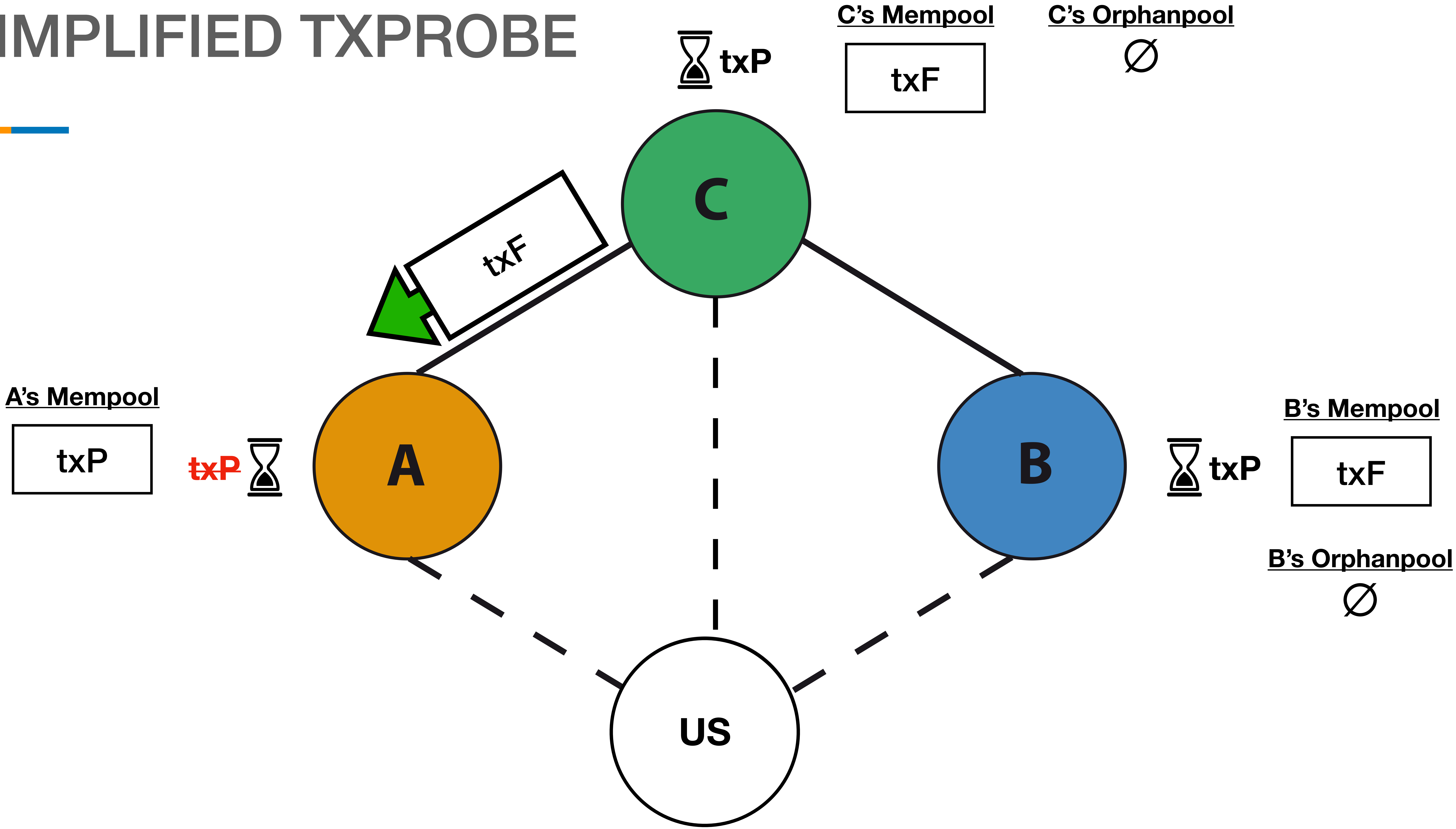
SIMPLIFIED TXPROBE



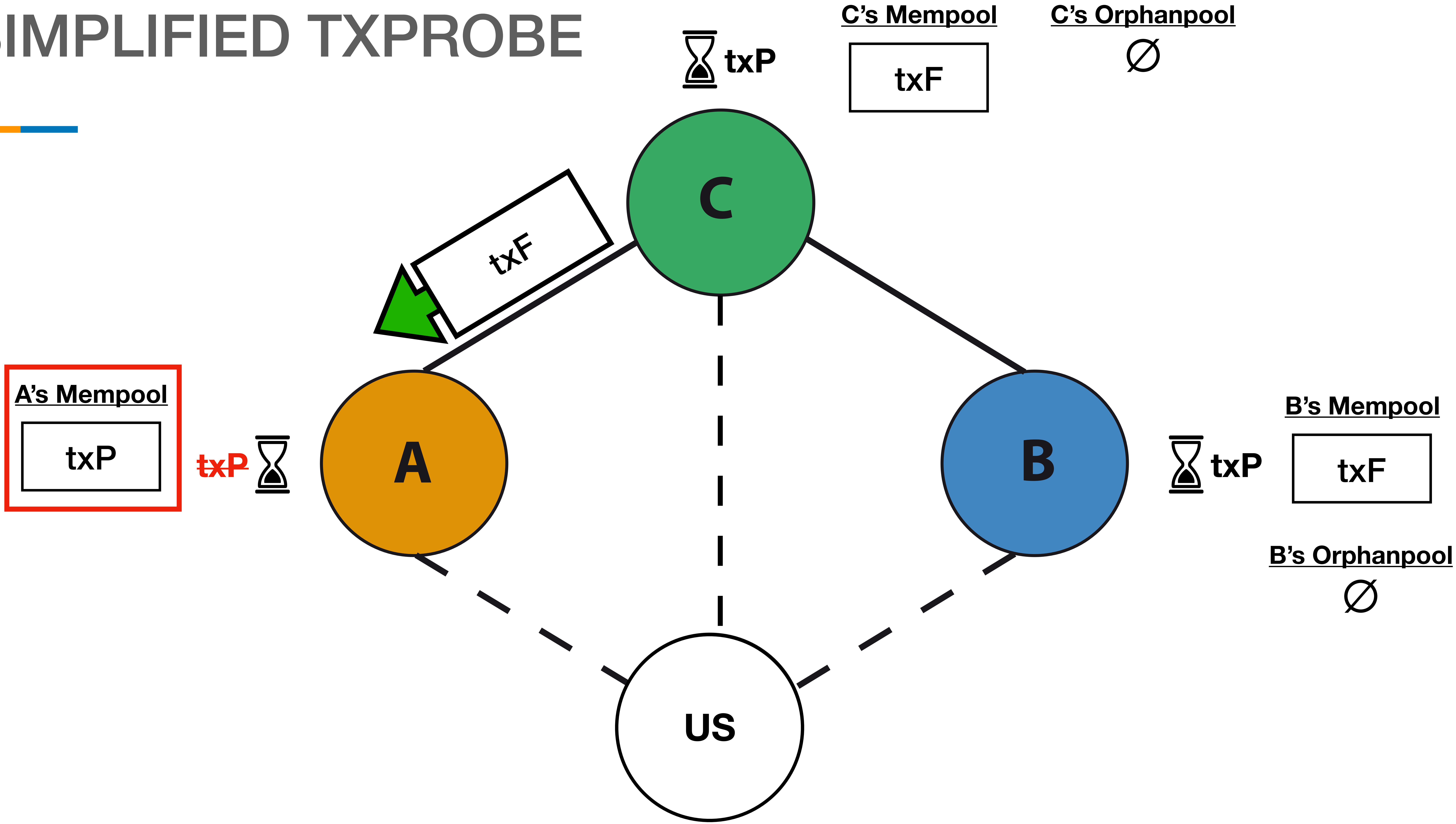
SIMPLIFIED TXPROBE



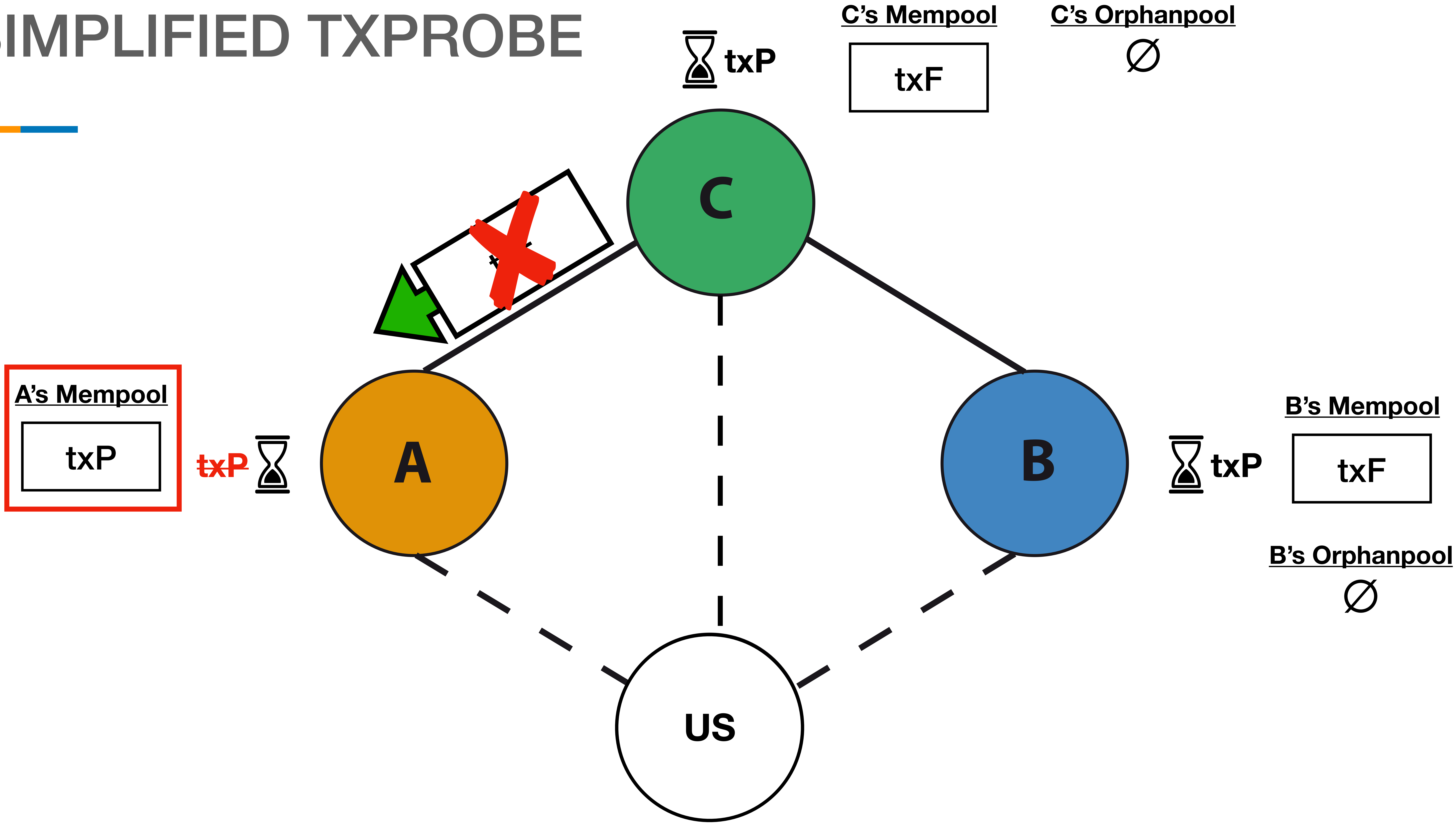
SIMPLIFIED TXPROBE



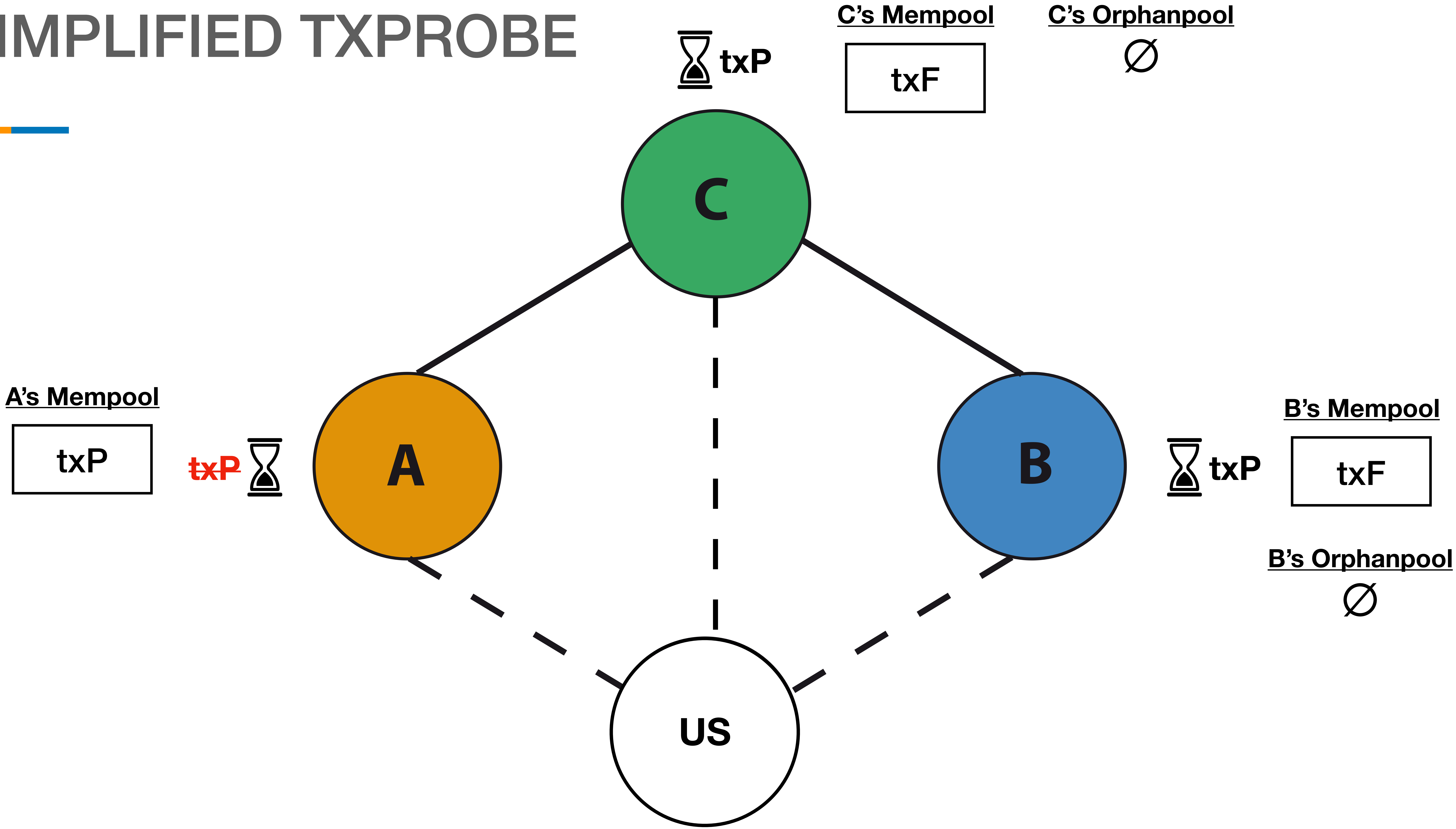
SIMPLIFIED TXPROBE



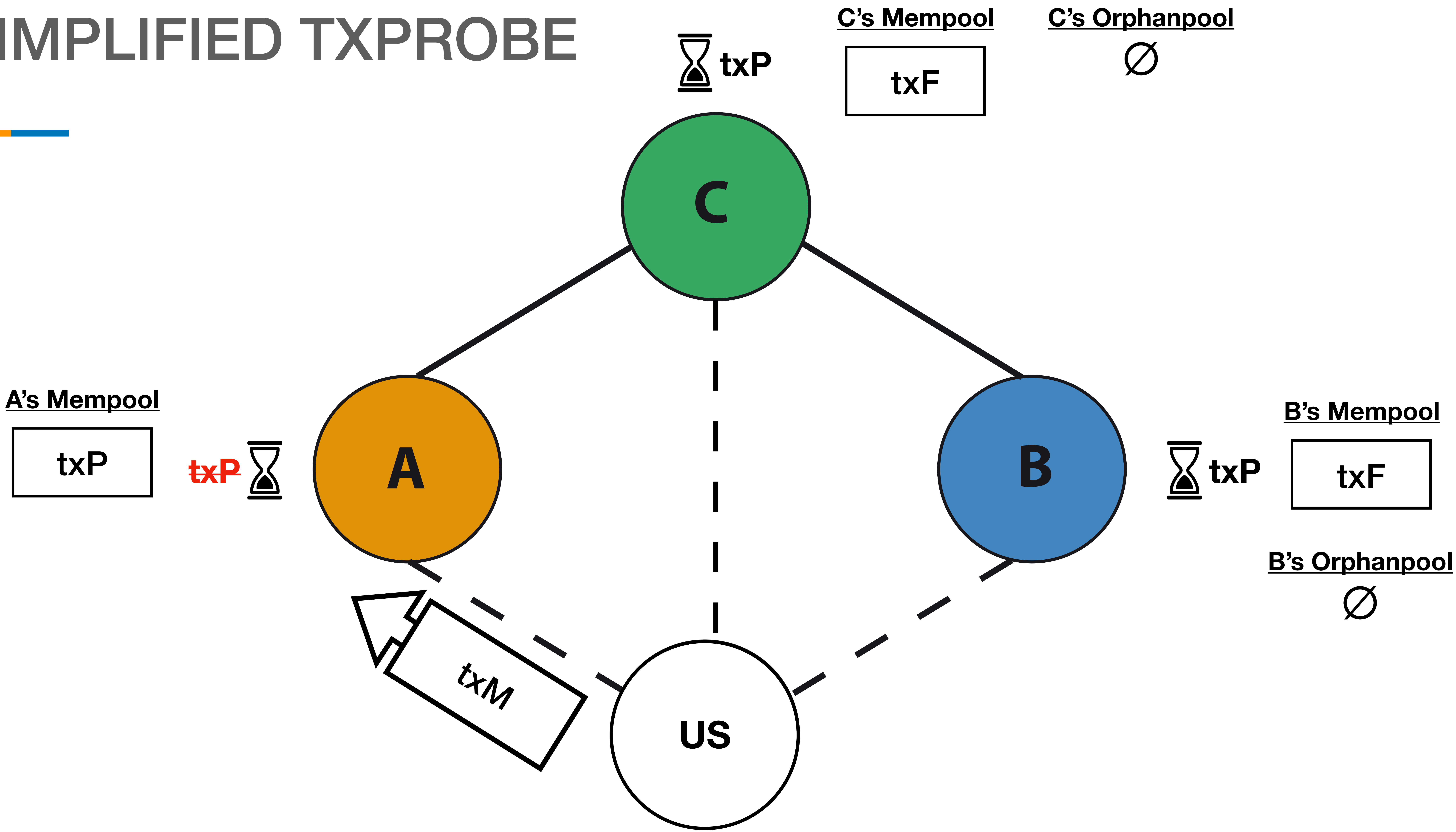
SIMPLIFIED TXPROBE



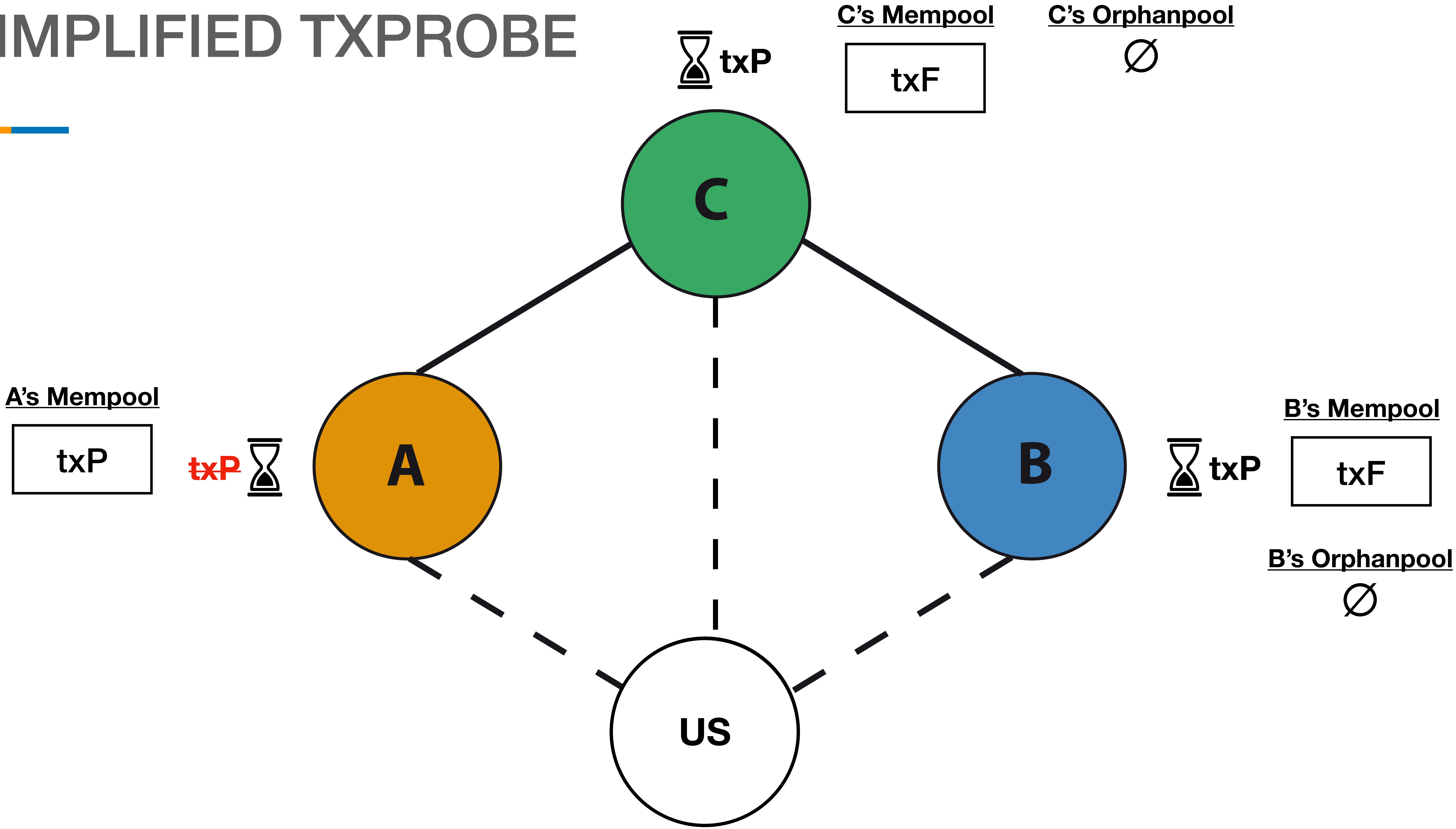
SIMPLIFIED TXPROBE



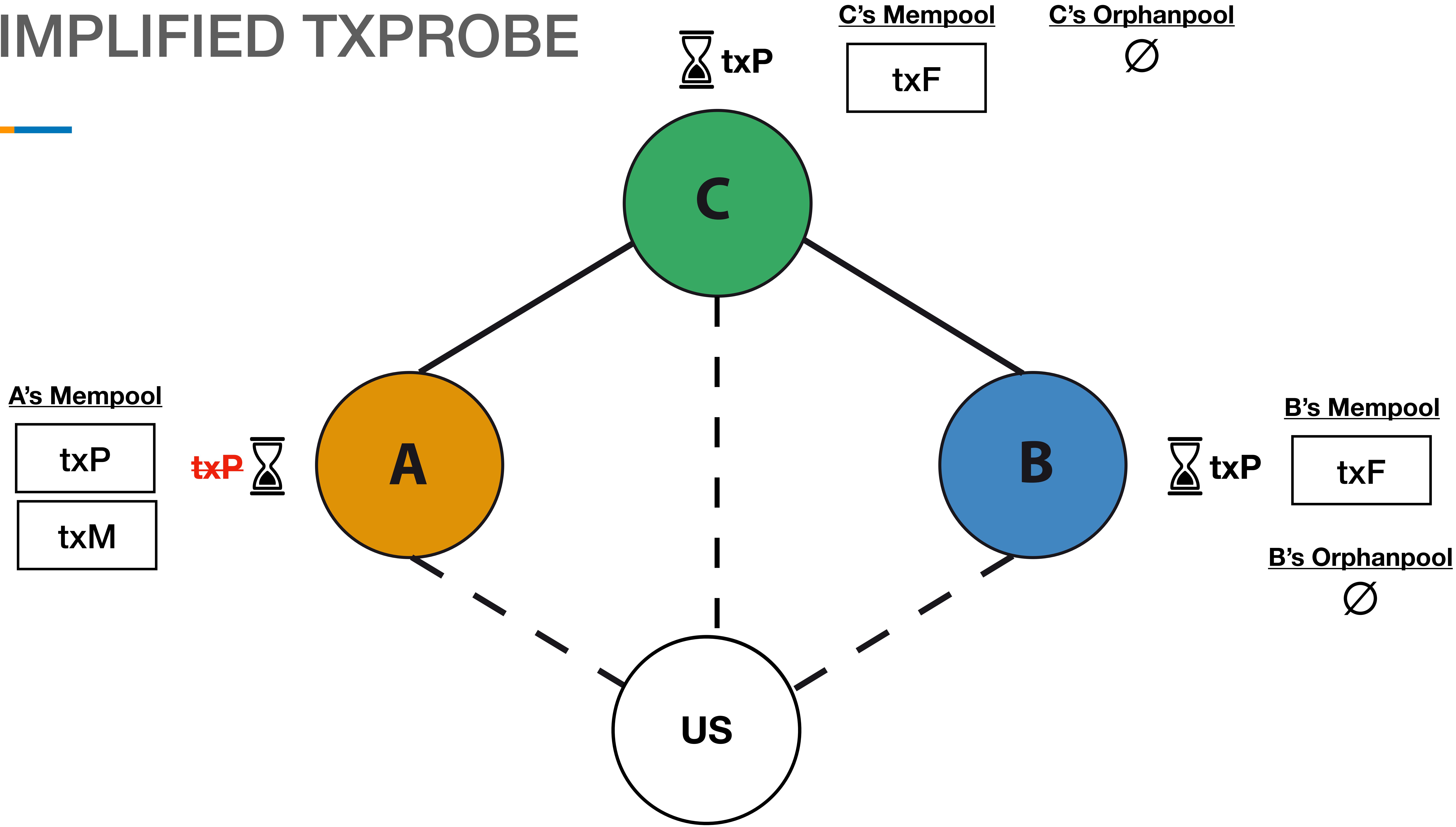
SIMPLIFIED TXPROBE



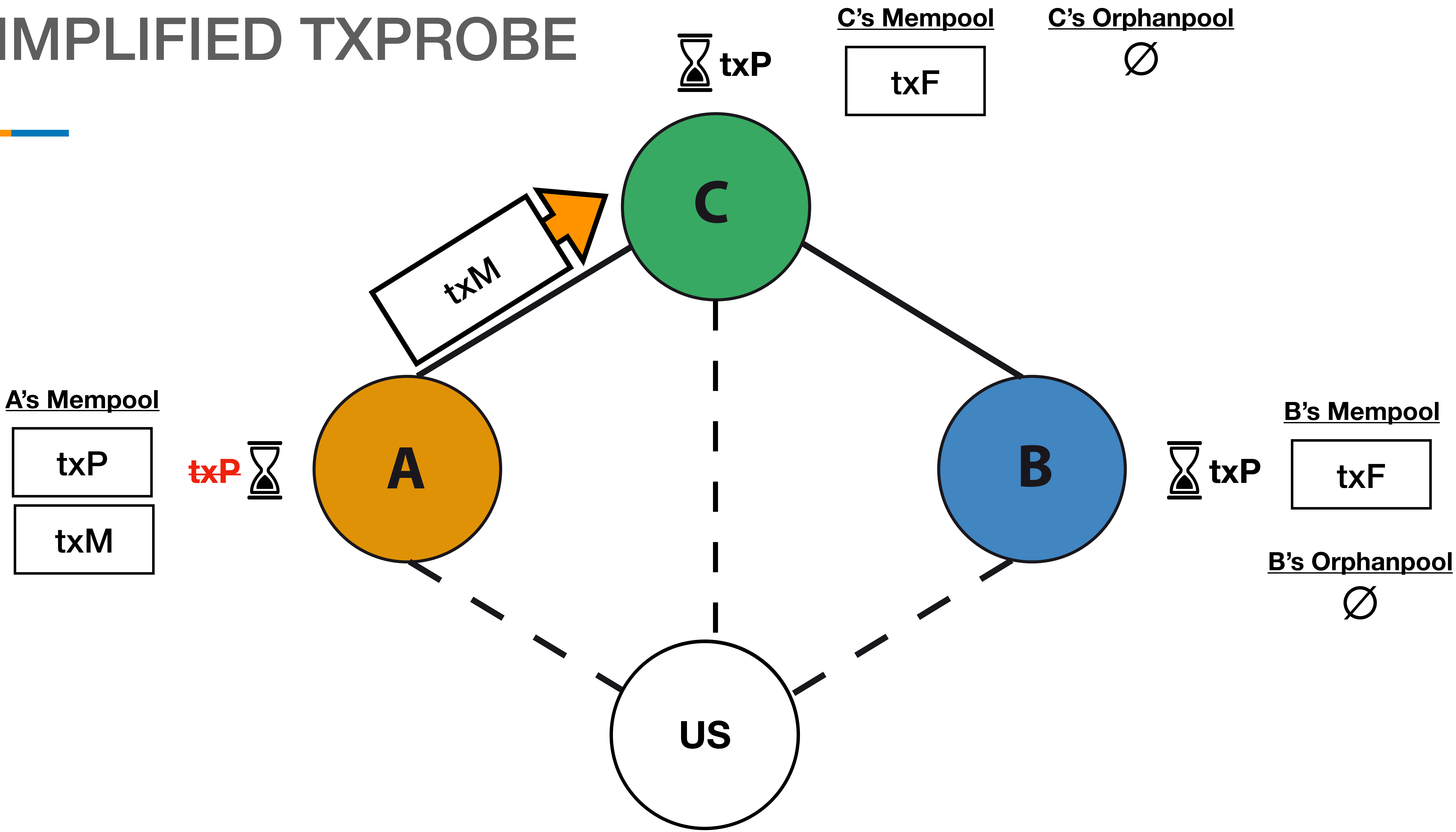
SIMPLIFIED TXPROBE



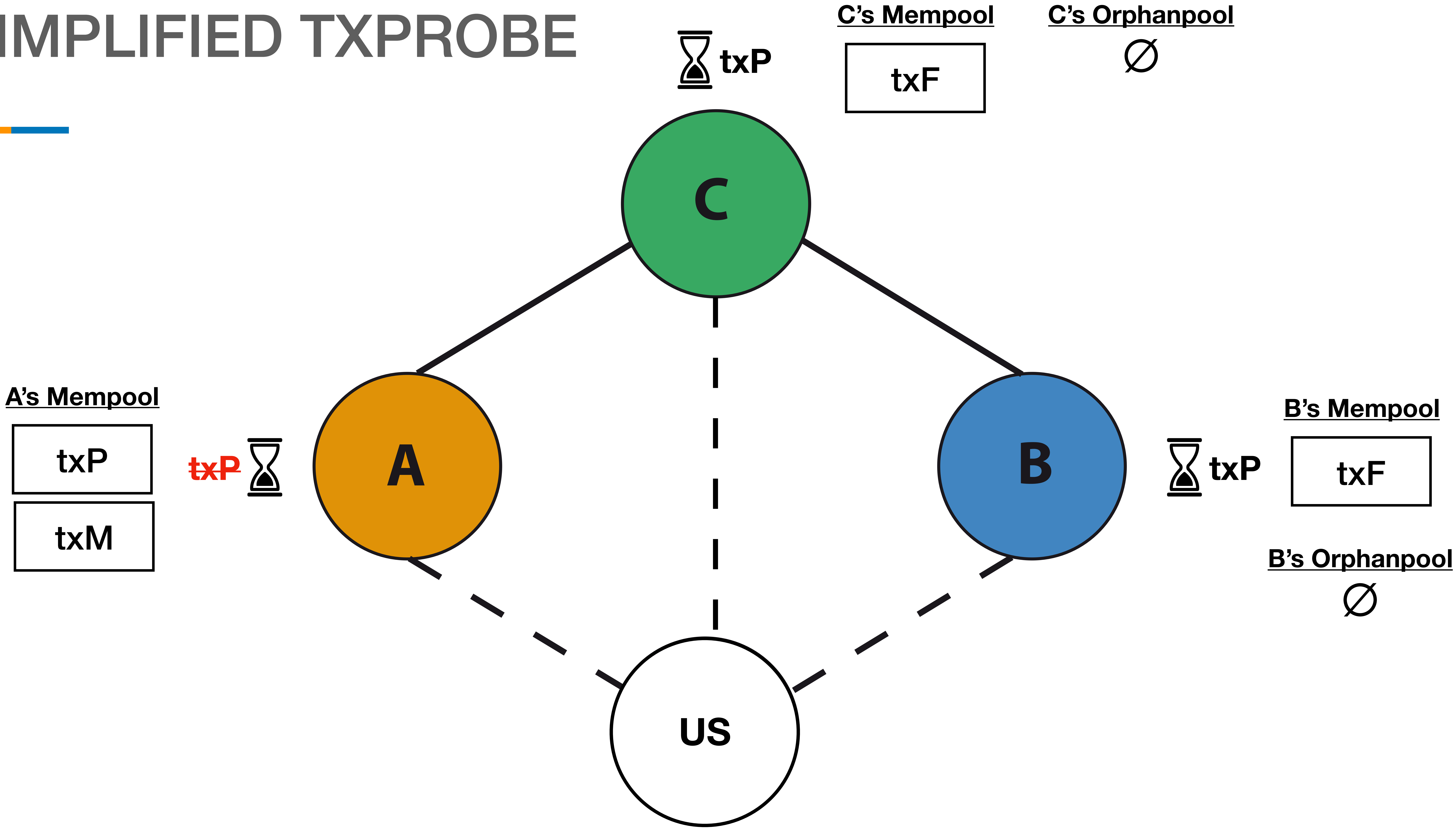
SIMPLIFIED TXPROBE



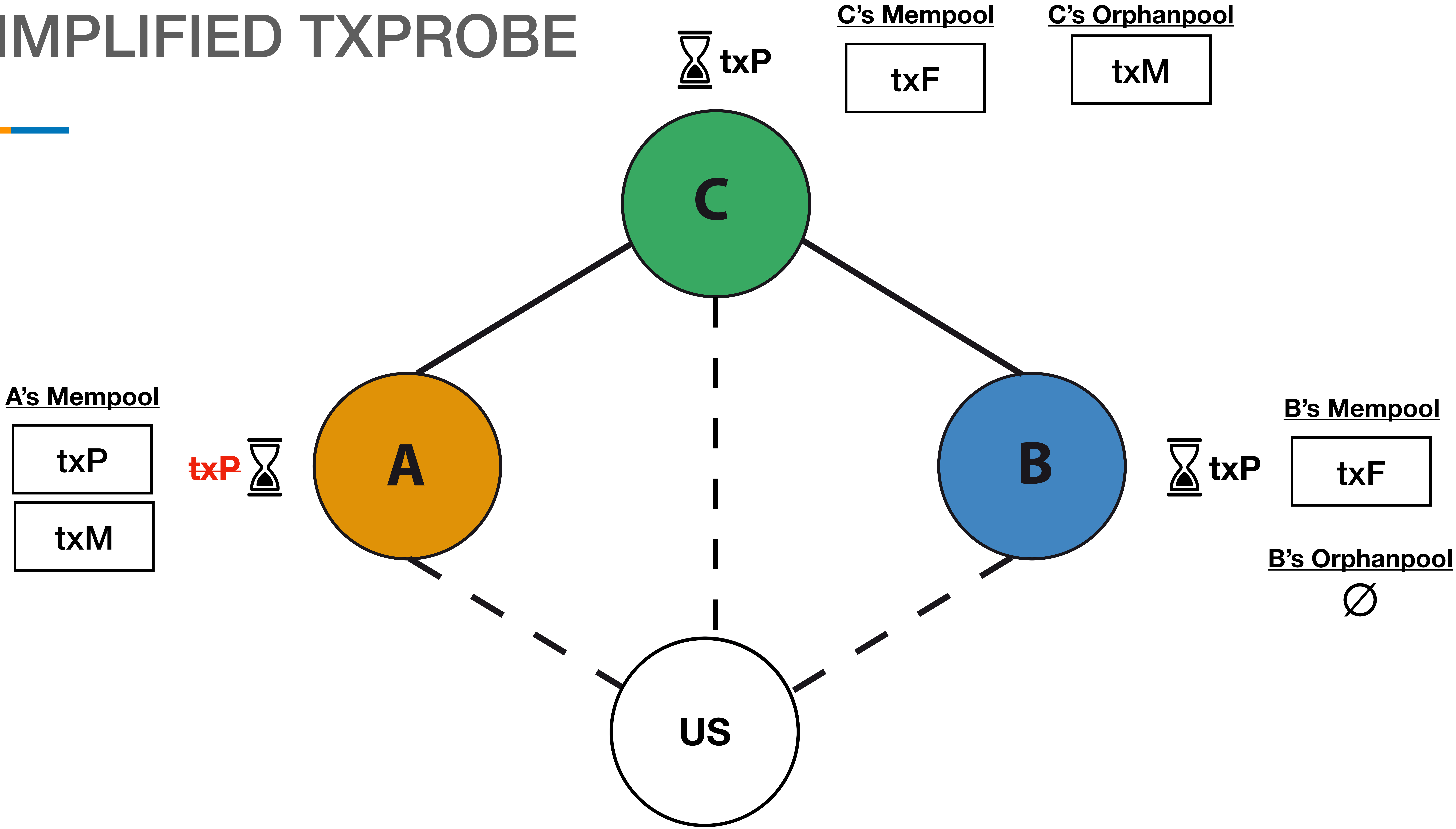
SIMPLIFIED TXPROBE



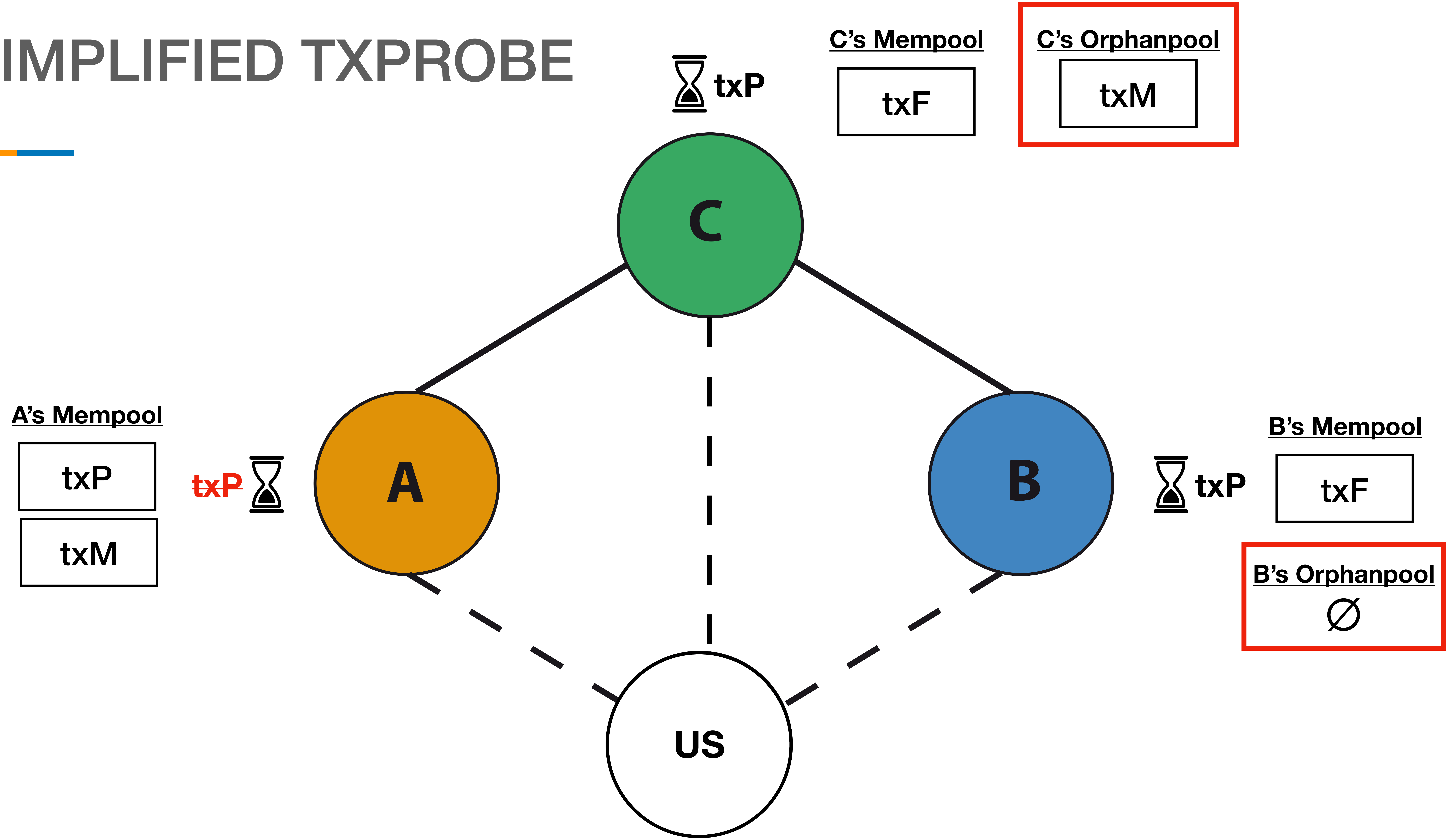
SIMPLIFIED TXPROBE



SIMPLIFIED TXPROBE



SIMPLIFIED TXPROBE



TXPROBE - PROTOCOL OVERVIEW

- **Choose** a target node
- **Create** Parent, Marker and Flood transactions
- **INVBLOCK** the network
- **Send** Flood to all connected nodes but target
- **Let** Flood propagate
- **Send** Parent to target
- **Send** Marker to target
- **Let** Marker propagate
- **Request** marker back from all nodes but target

TXPROBE - PROTOCOL OVERVIEW

- **Choose** a target node
- **Create** Parent, Marker and Flood transactions
- **INVBLOCK** the network
- **Send** Flood to all connected nodes but target
- **Let** Flood propagate
- **Send** Parent to target
- **Send** Marker to target
- **Let** Marker propagate
- **Request** marker back from all nodes but target

For every node in the network

TXPROBE - COSTS ESTIMATION

For a network like **Bitcoin mainnet:**

nodes ≈ 10000

time ≈ 8.25 hours

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

TXPROBE - DATA VALIDATION (TESTNET)

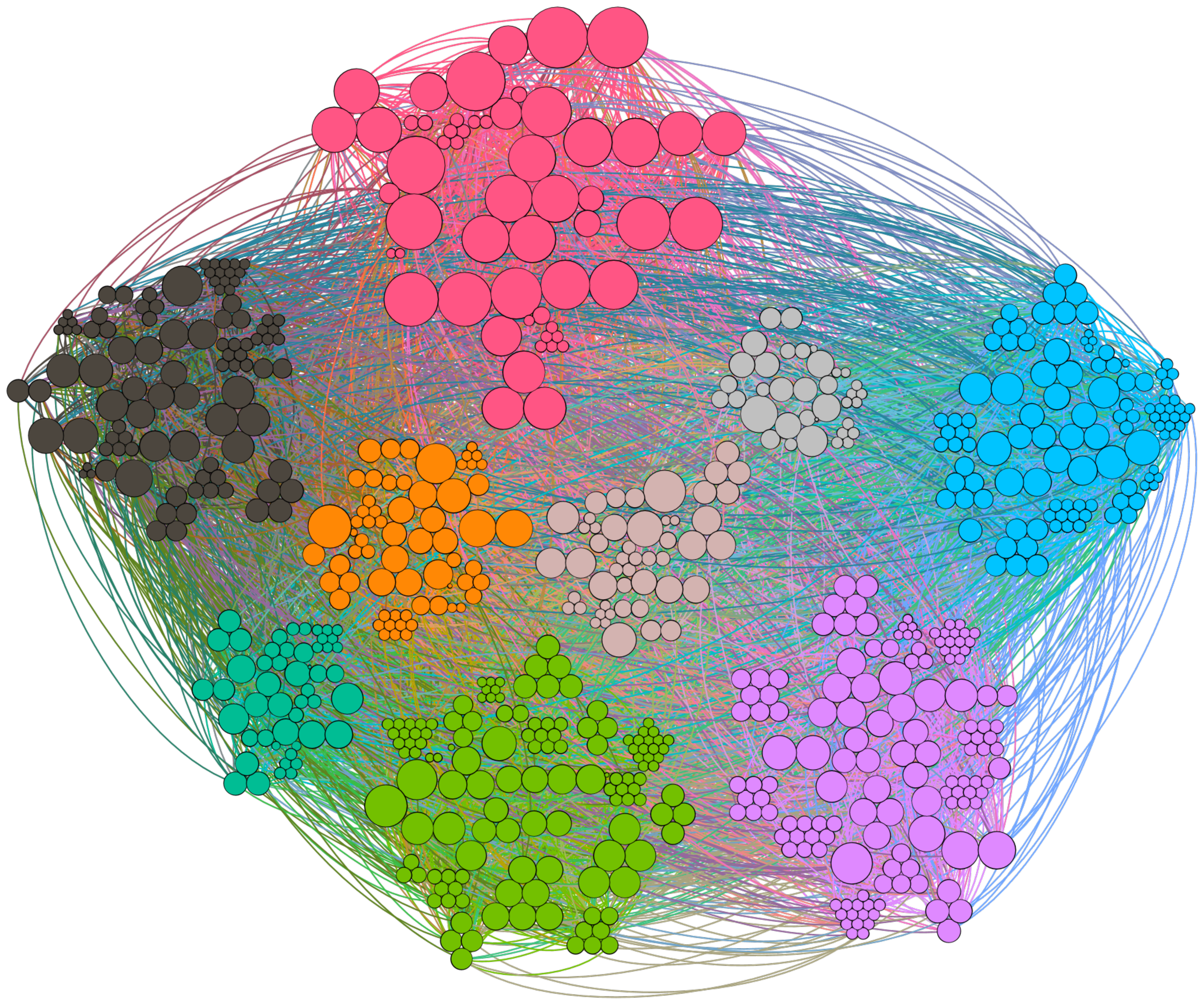
We run 5 Bitcoin Core nodes as **ground truth**

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

Over 40 trials and with 95% confidence:

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

TXPROBE - TESTNET TOPOLOGY



precision = 100%

recall = 97.40%

size ⟶ degree

color ⟶ Community unfolding

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

CONCLUSIONS

CONCLUSIONS

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.

CONCLUSIONS

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

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

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 c
(lexicographic

Is topology hiding a design goal, or is it a **mean to achieve other goals (e.g: Transaction privacy)?**

randomize G
toward outbo

Merged sipa merged

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

QUESTIONS

sr_gi

P/SA RESEARCH

BONUS TRACK

Testnet vs Mainnet

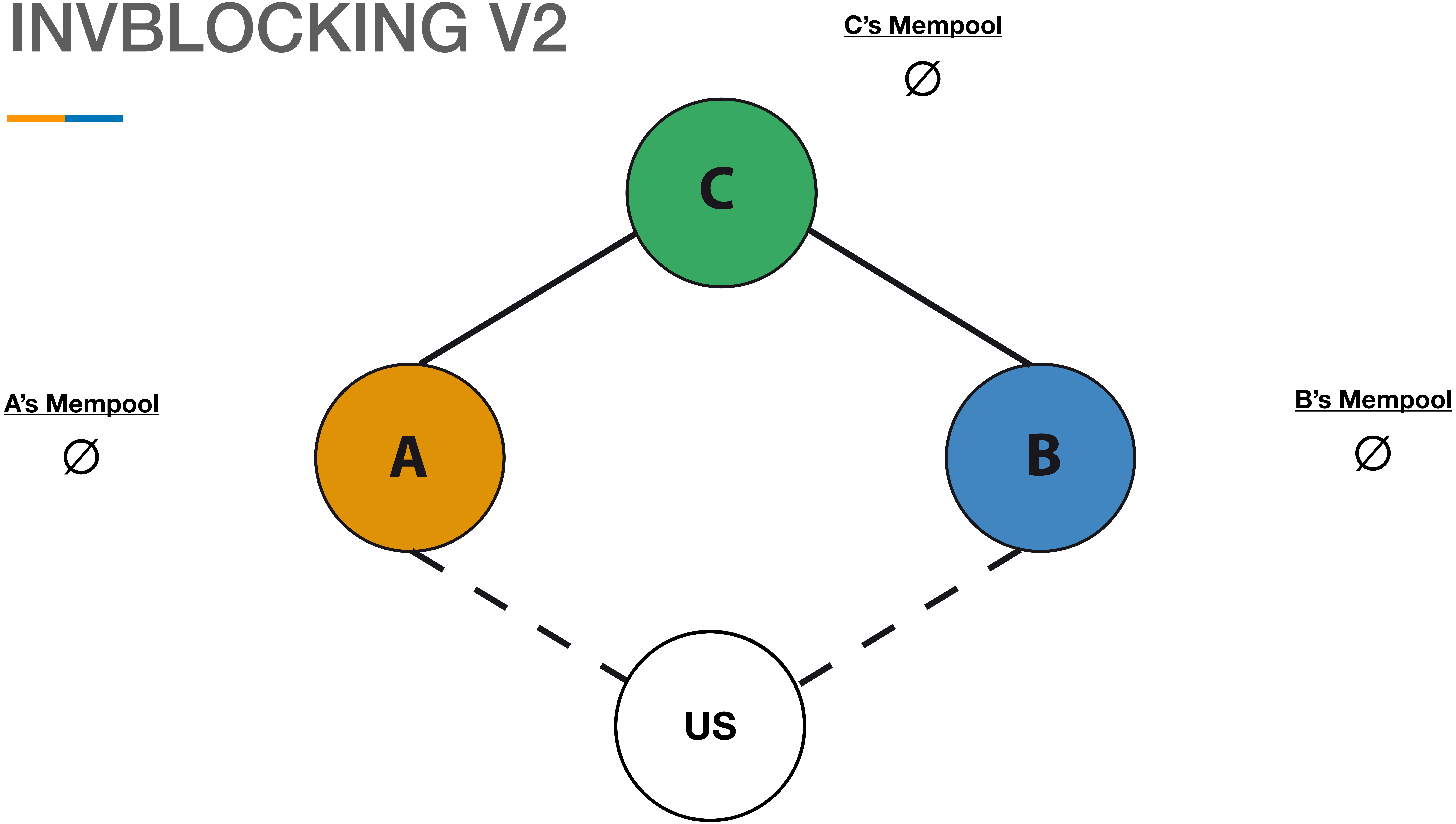
INVBLOCKING (no-link)

Efficiency / Orphan pool eviction

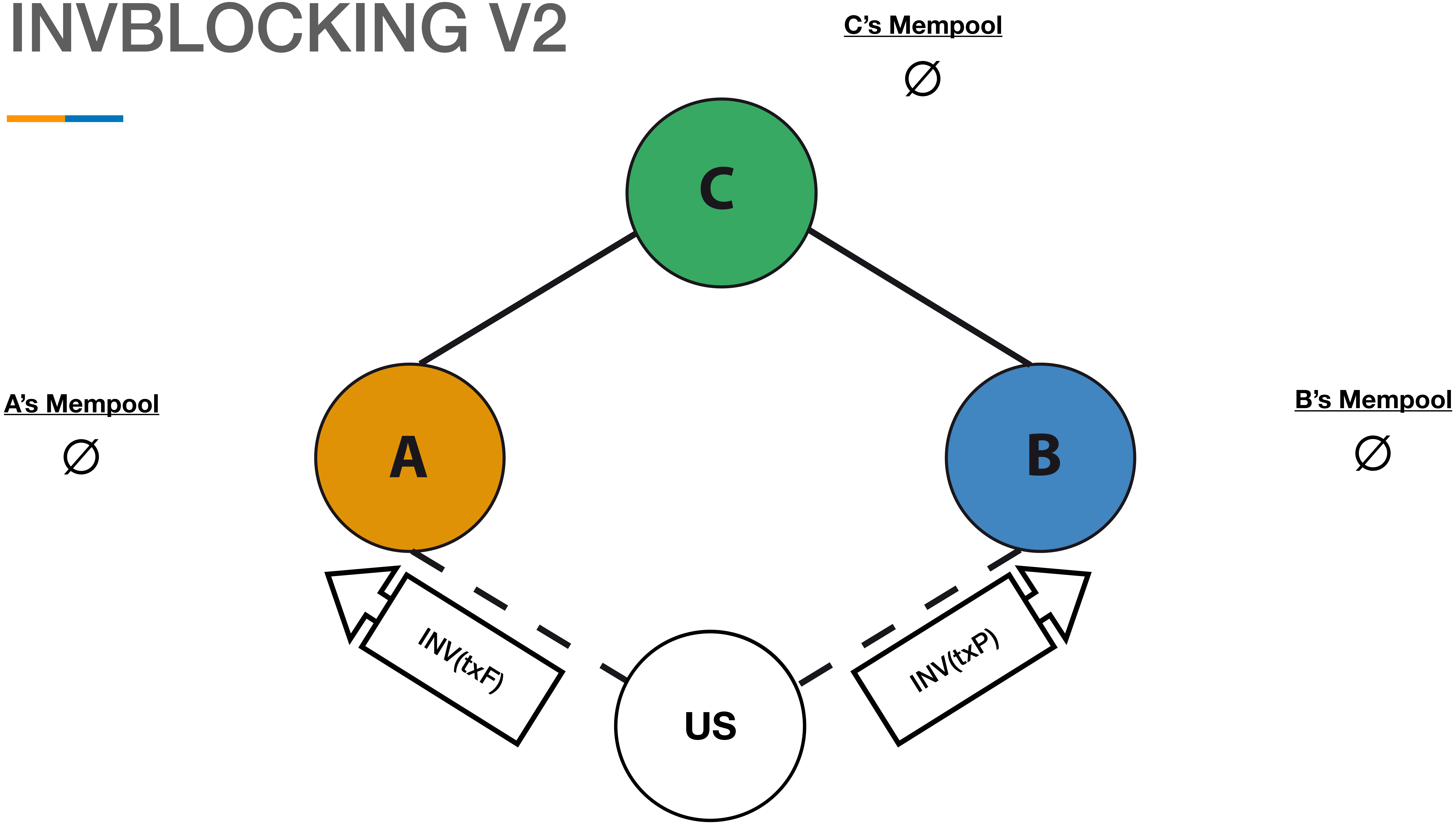
WHY TESTNET AND NO MAINNET?

- 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**

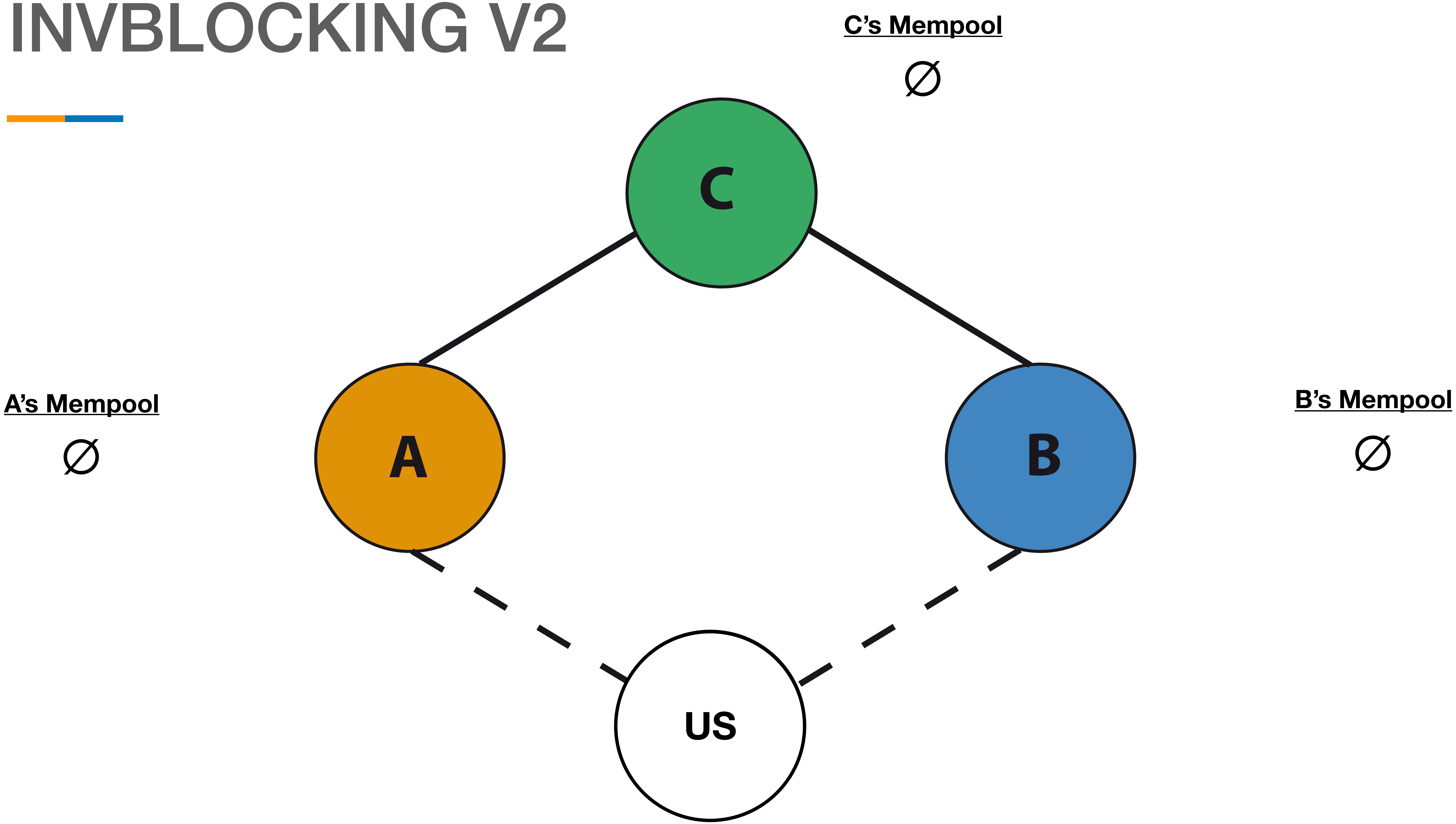
INVBLOCKING V2



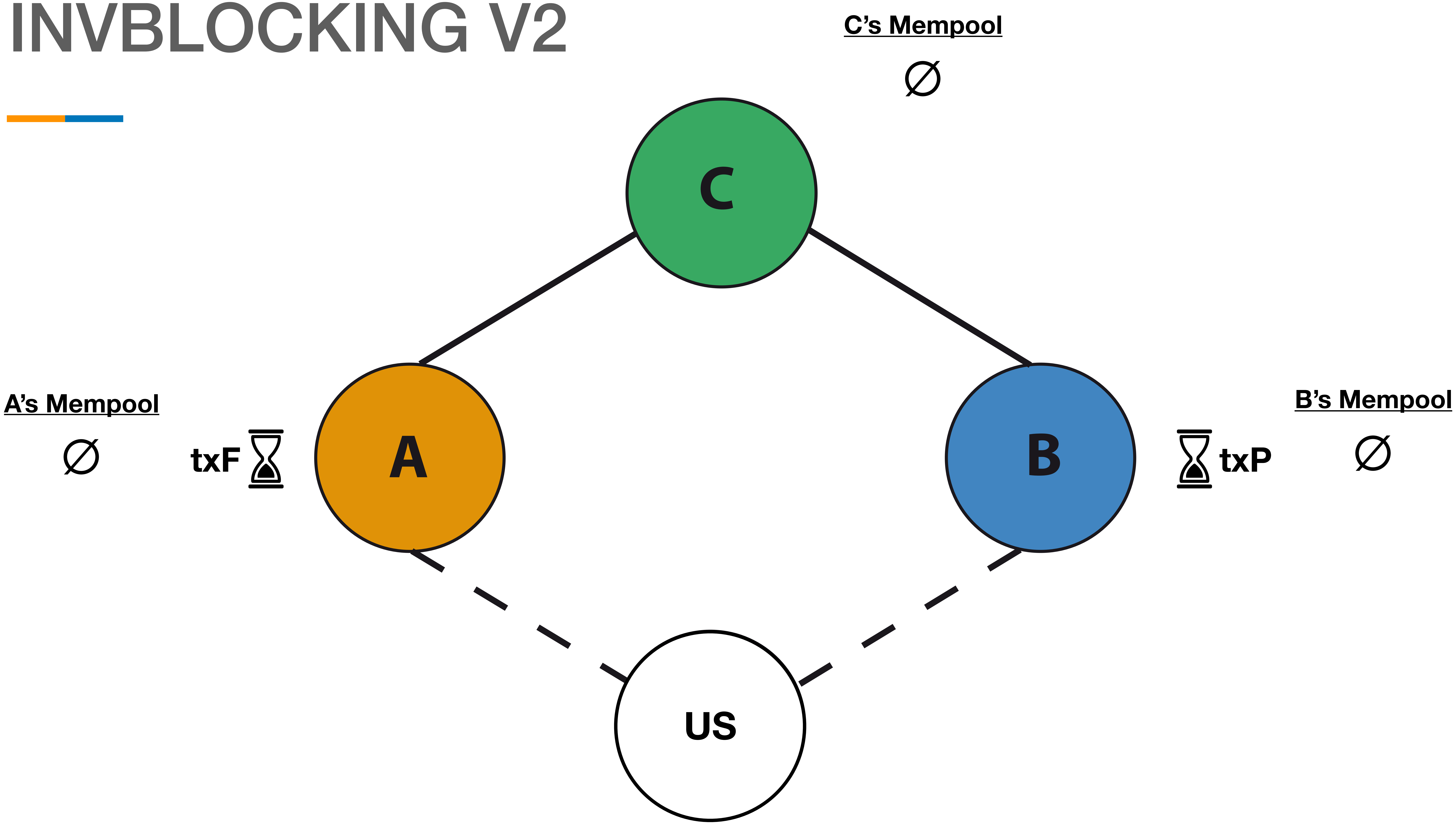
INVBLOCKING V2



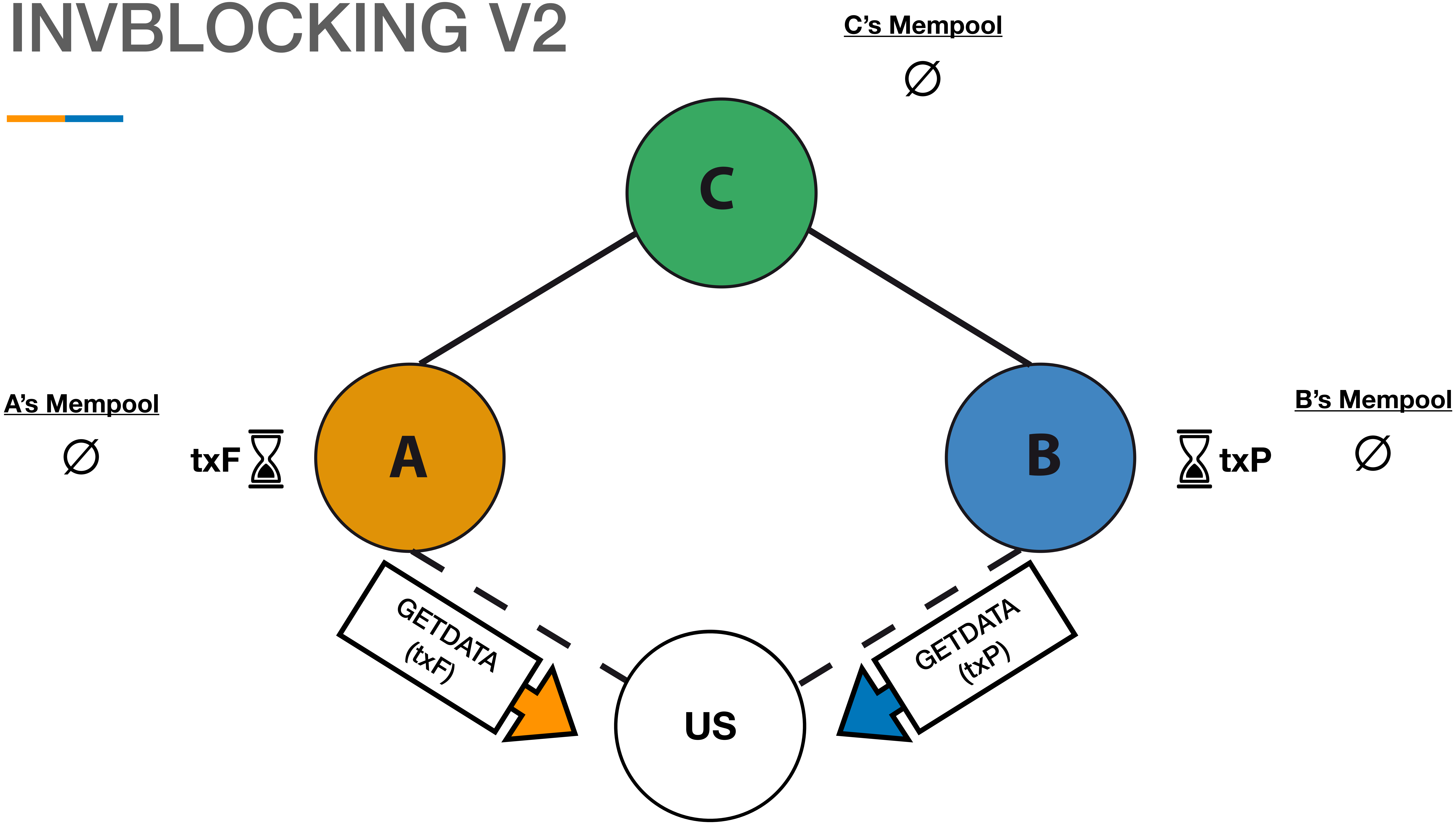
INVBLOCKING V2



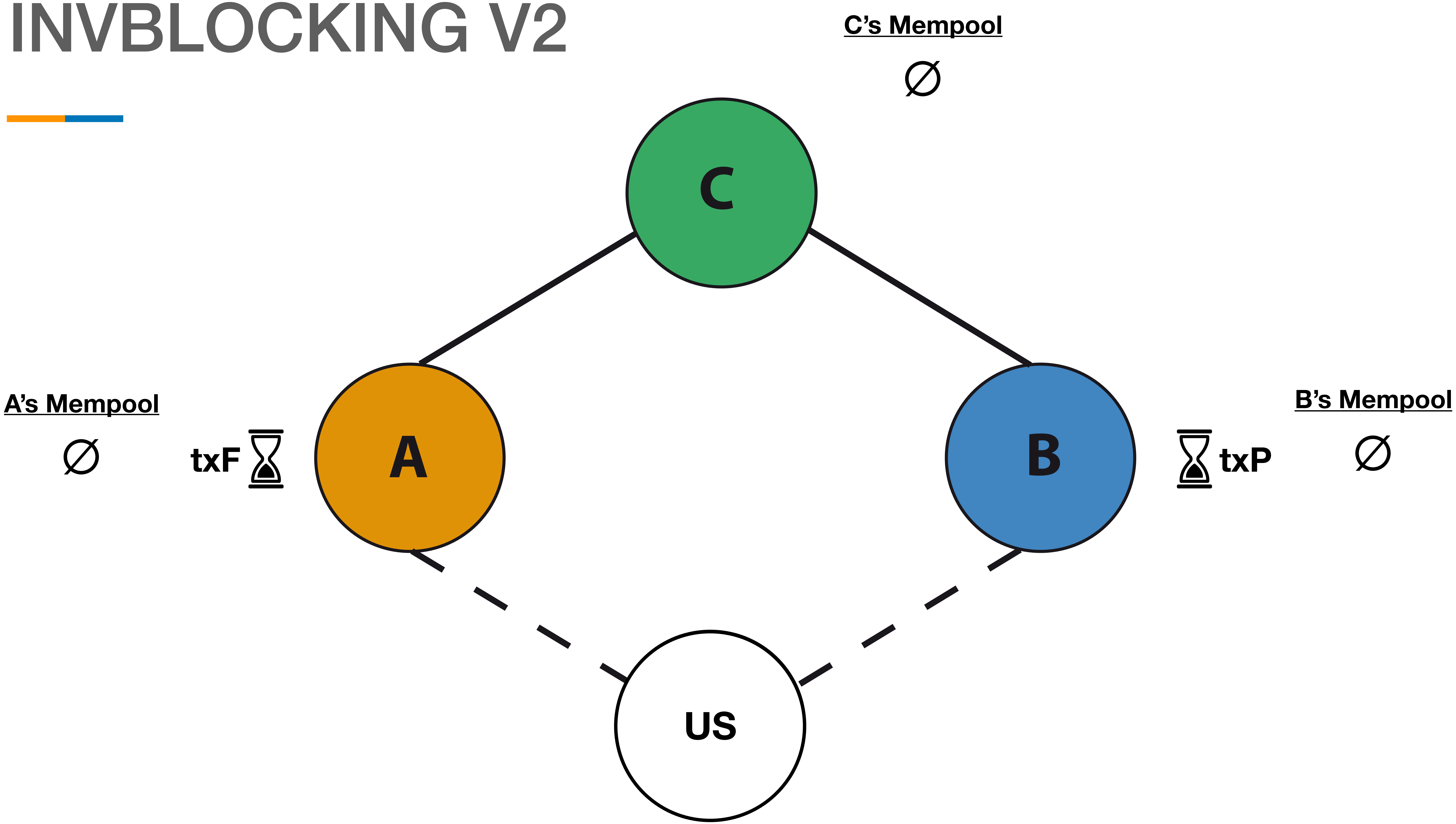
INVBLOCKING V2



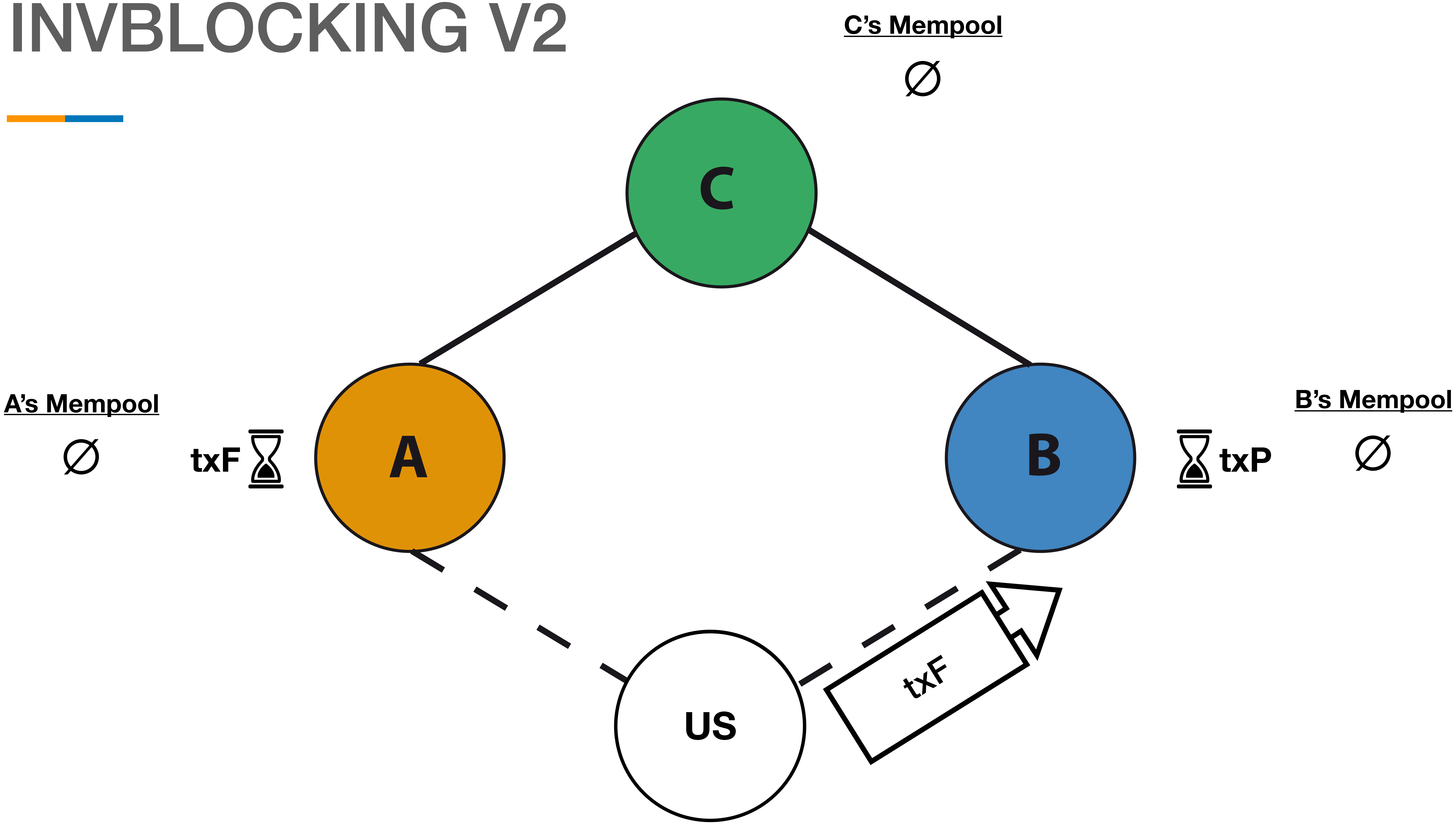
INVBLOCKING V2



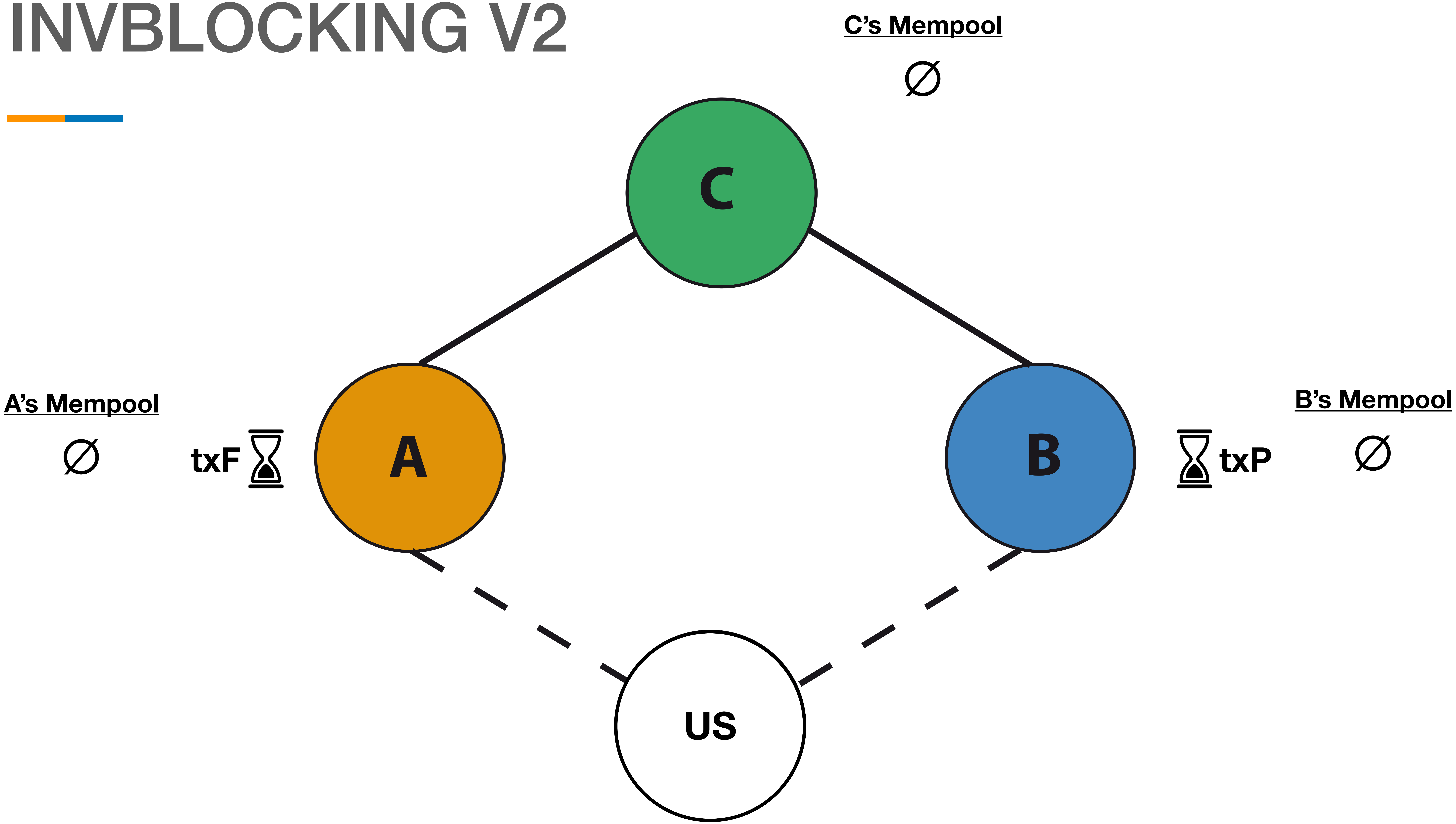
INVBLOCKING V2



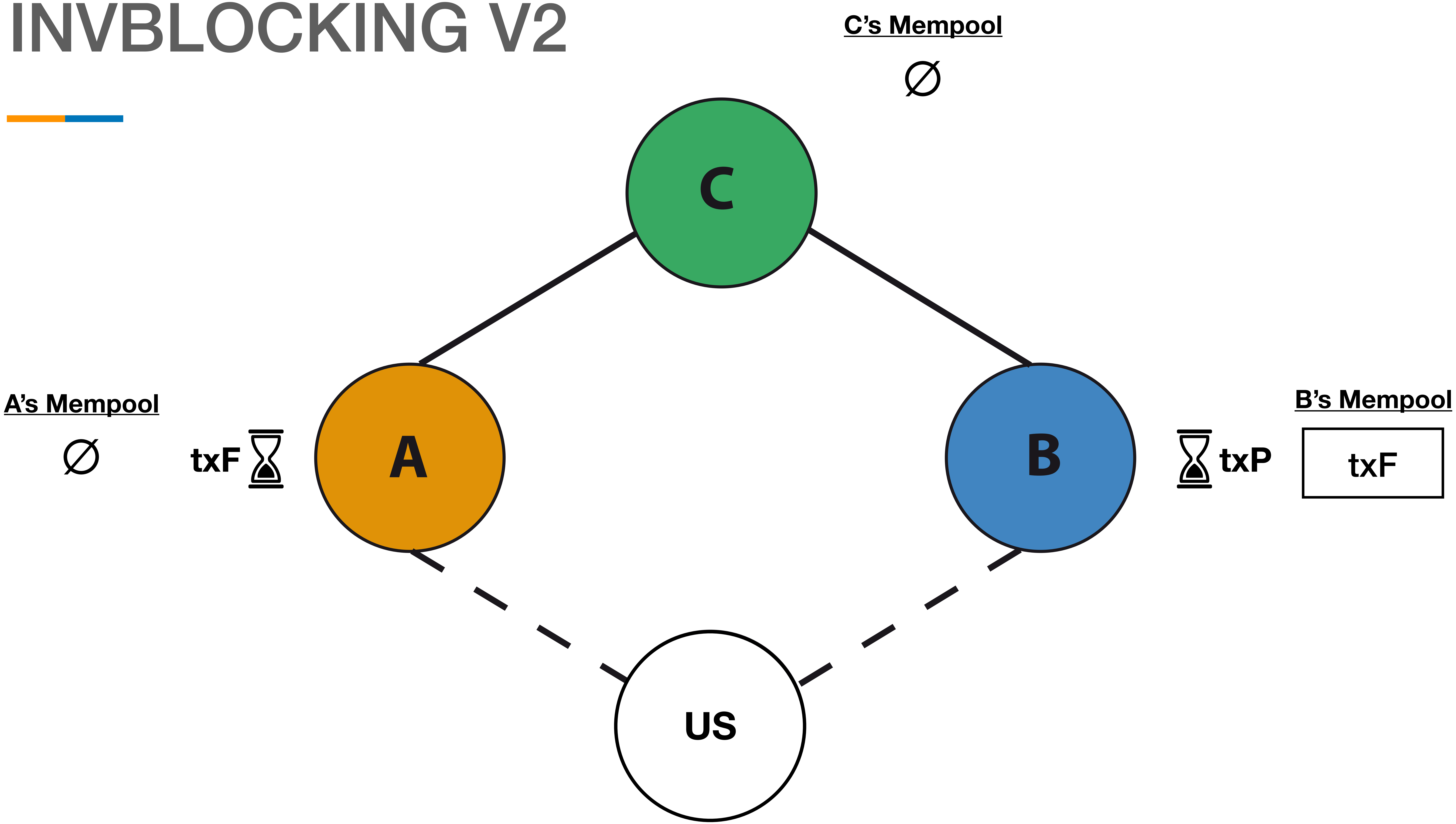
INVBLOCKING V2



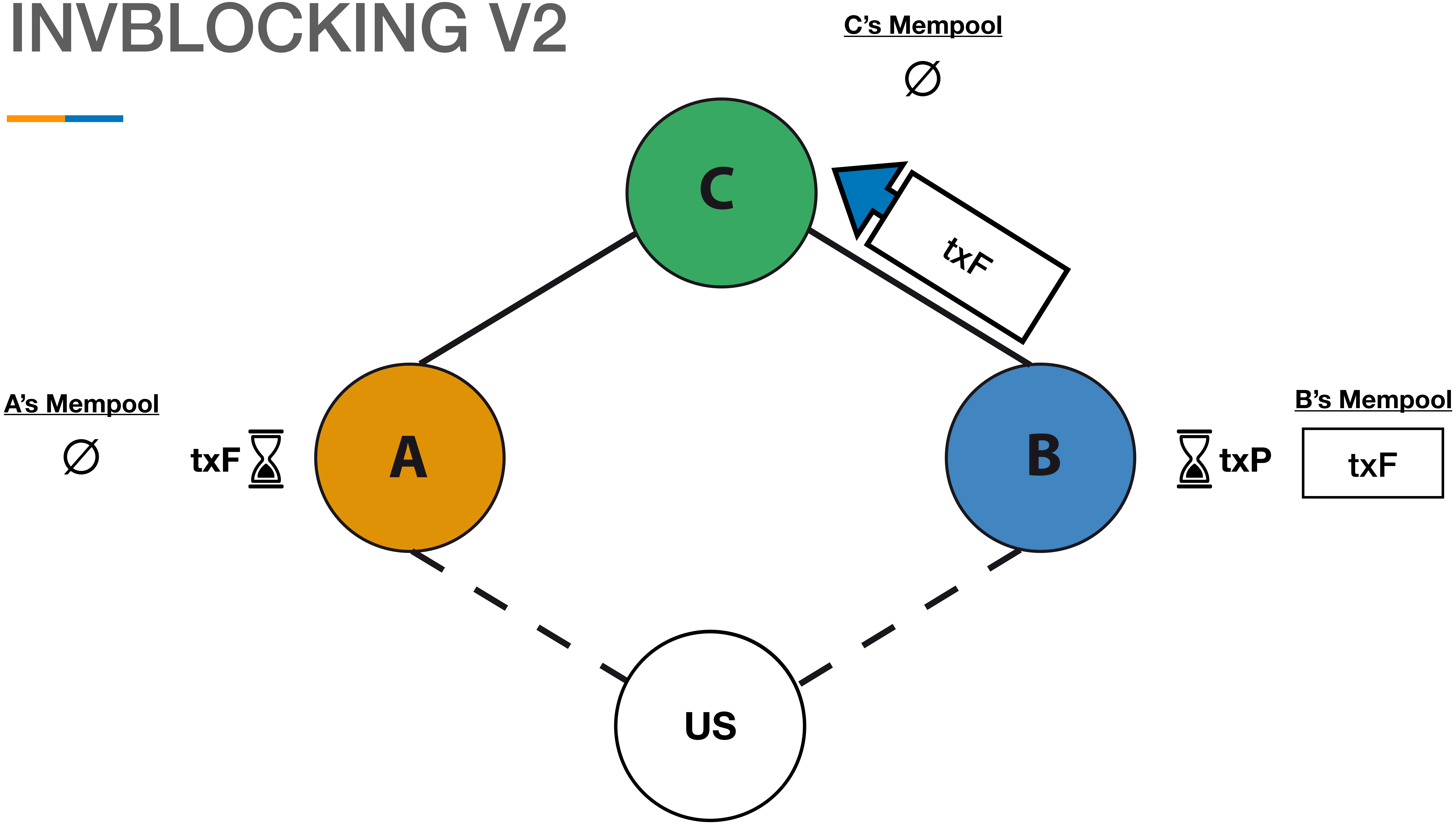
INVBLOCKING V2



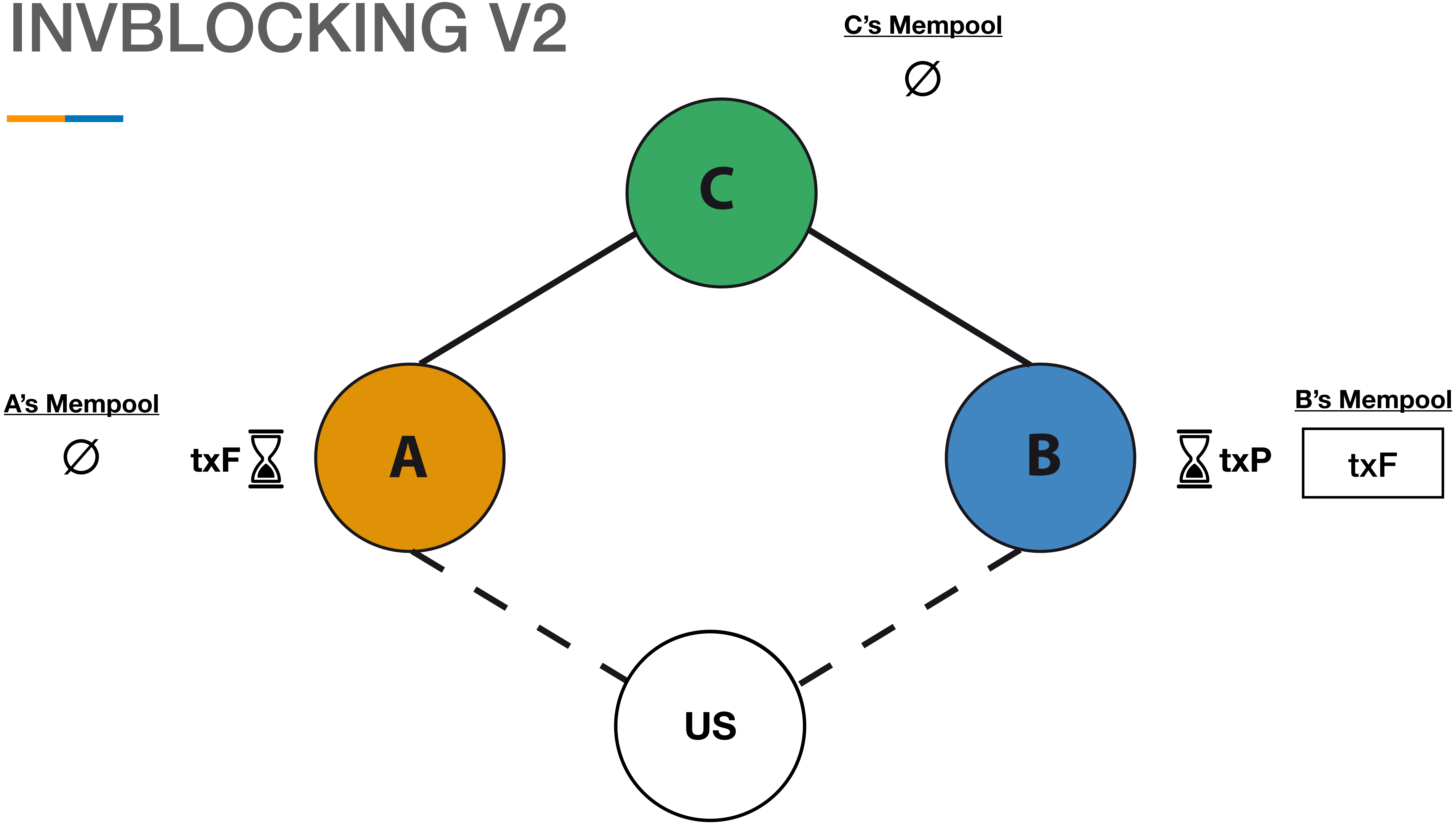
INVBLOCKING V2



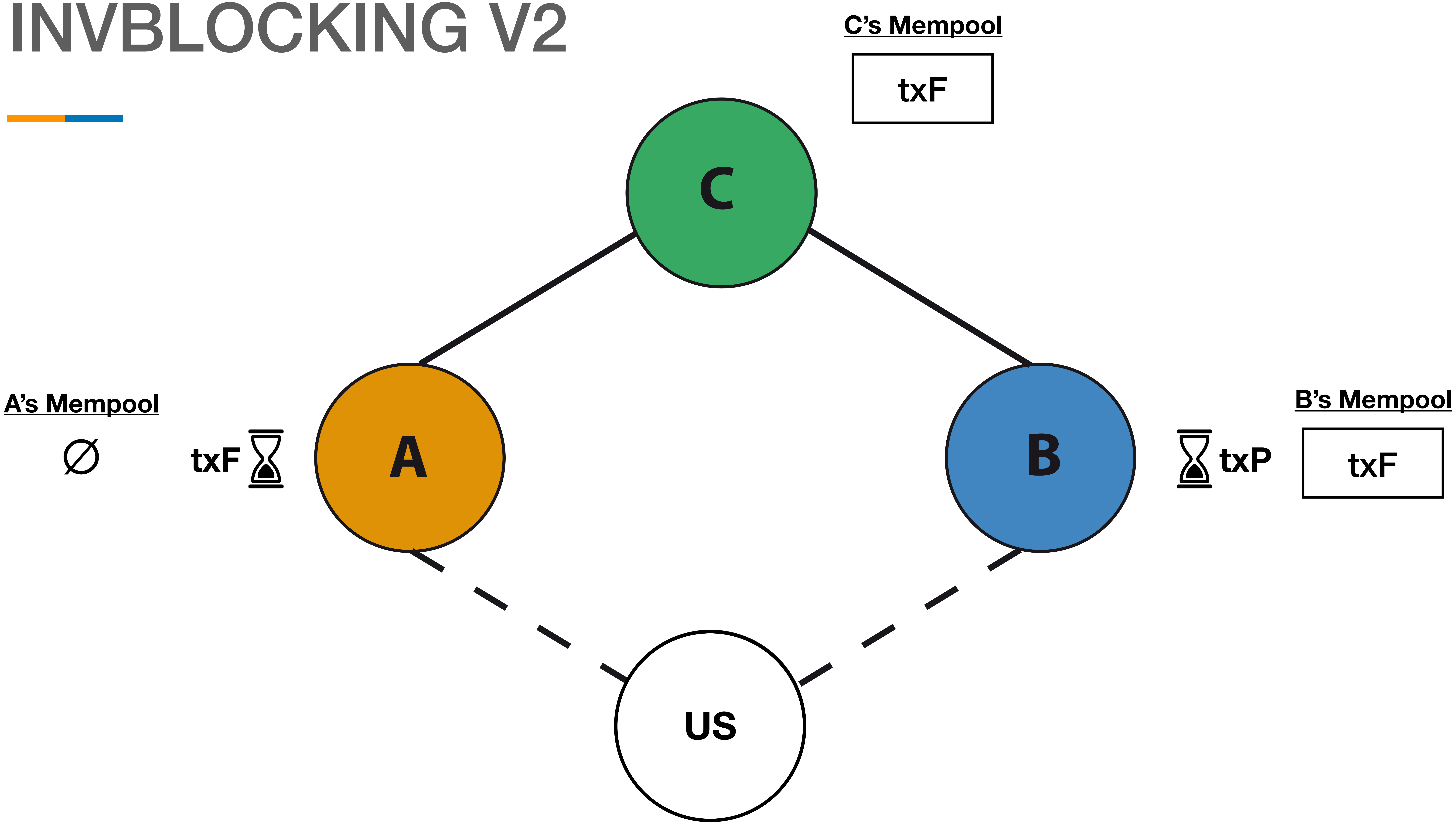
INVBLOCKING V2



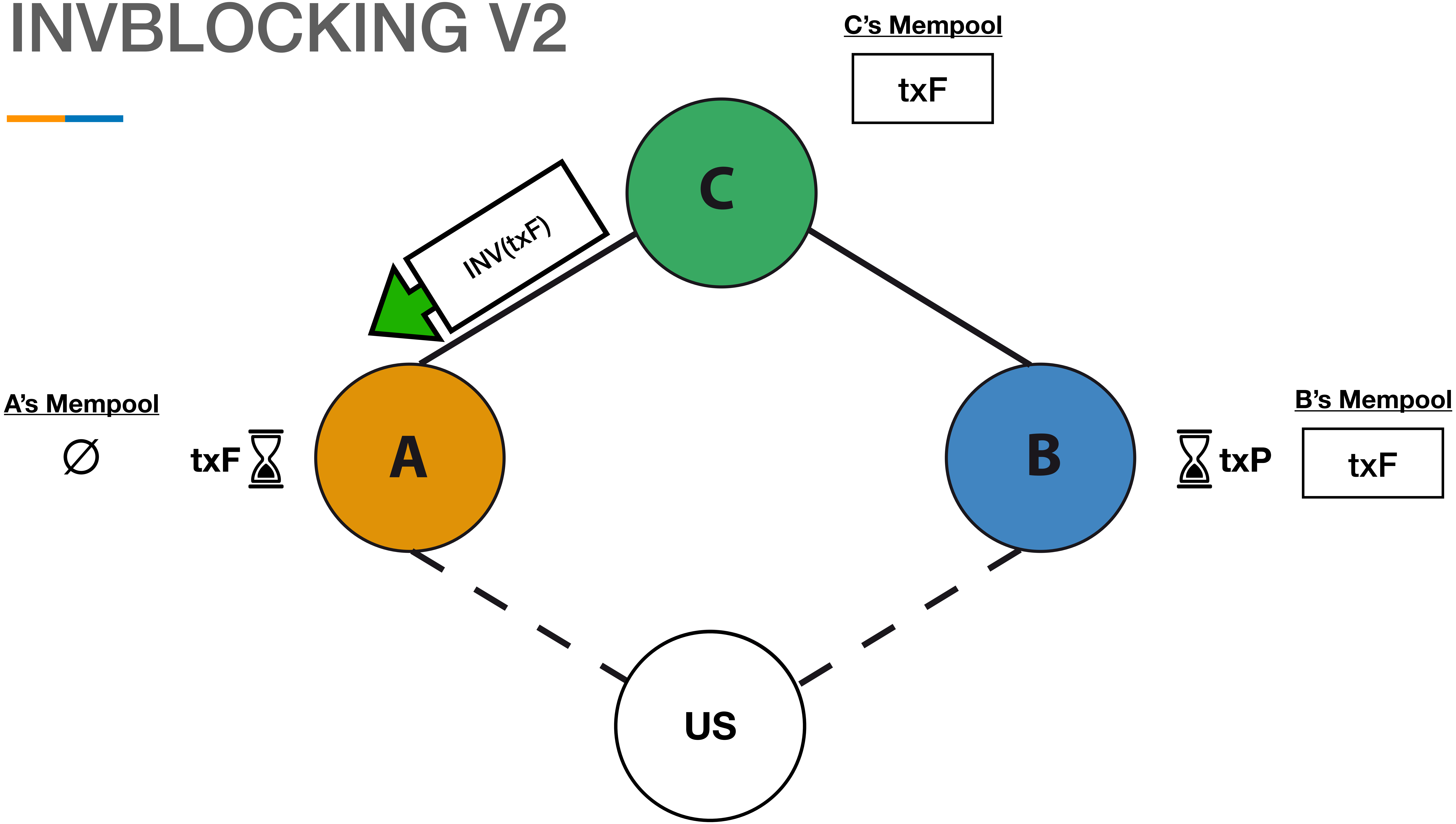
INVBLOCKING V2



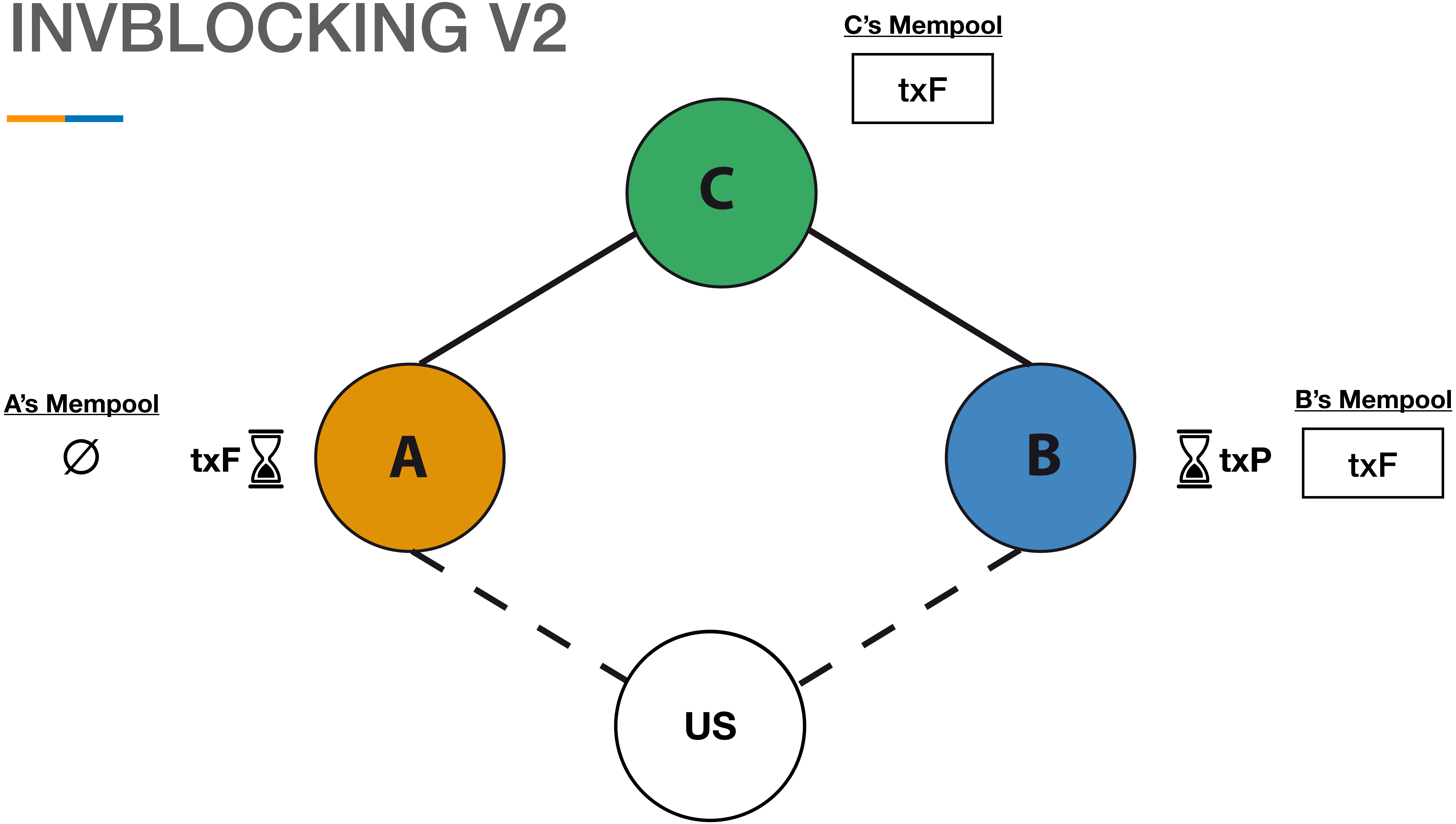
INVBLOCKING V2



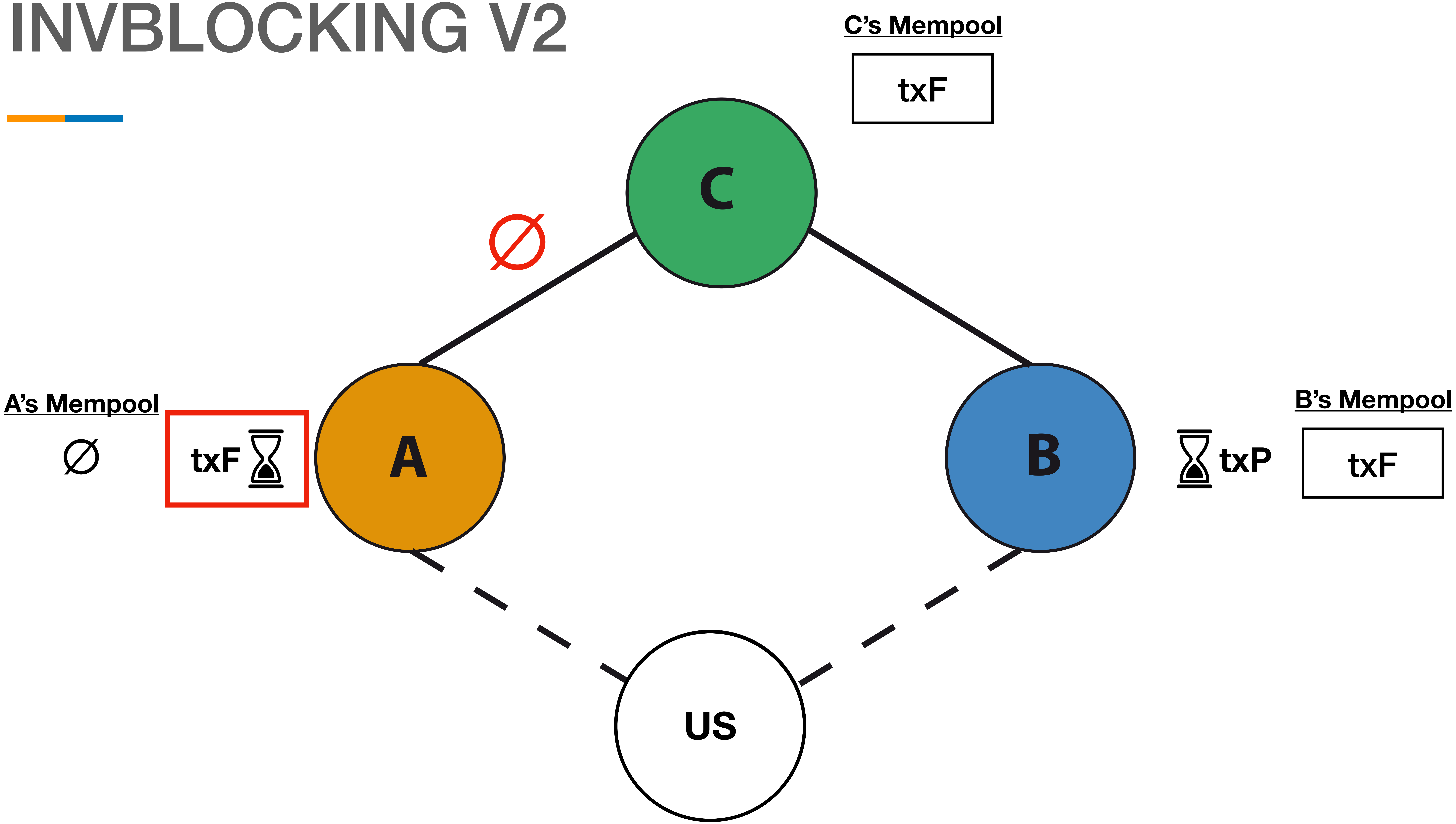
INVBLOCKING V2



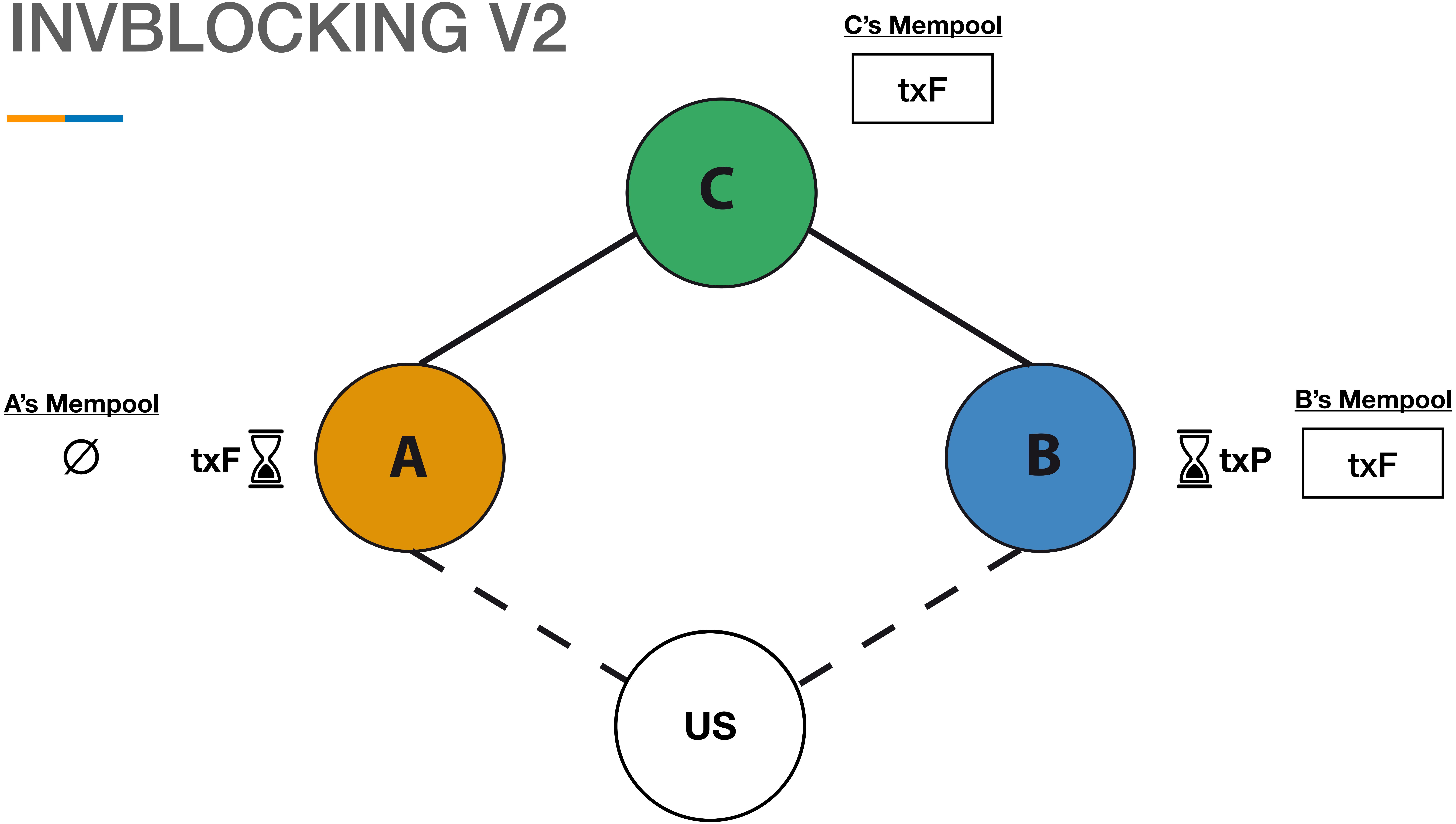
INVBLOCKING V2



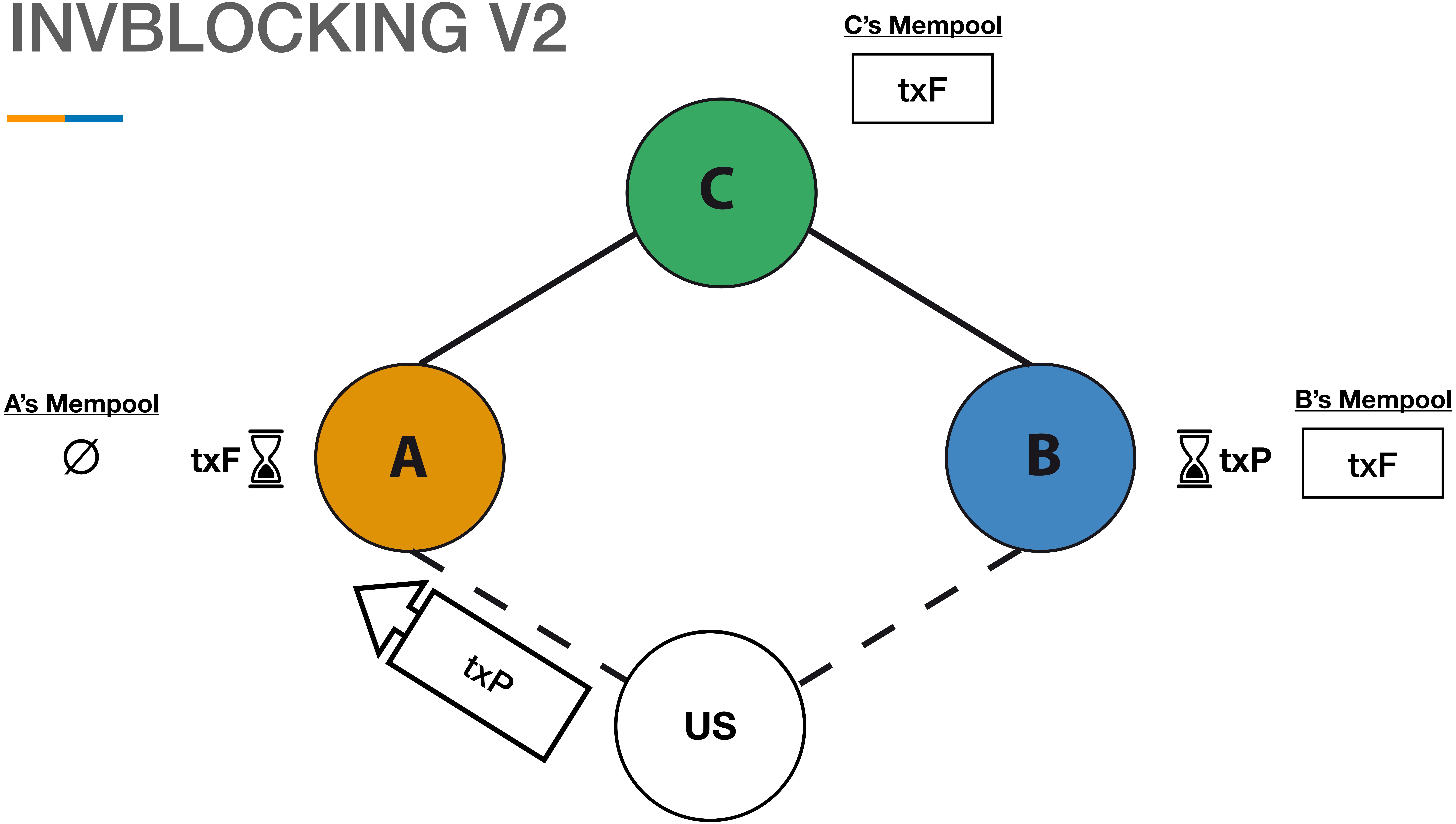
INVBLOCKING V2



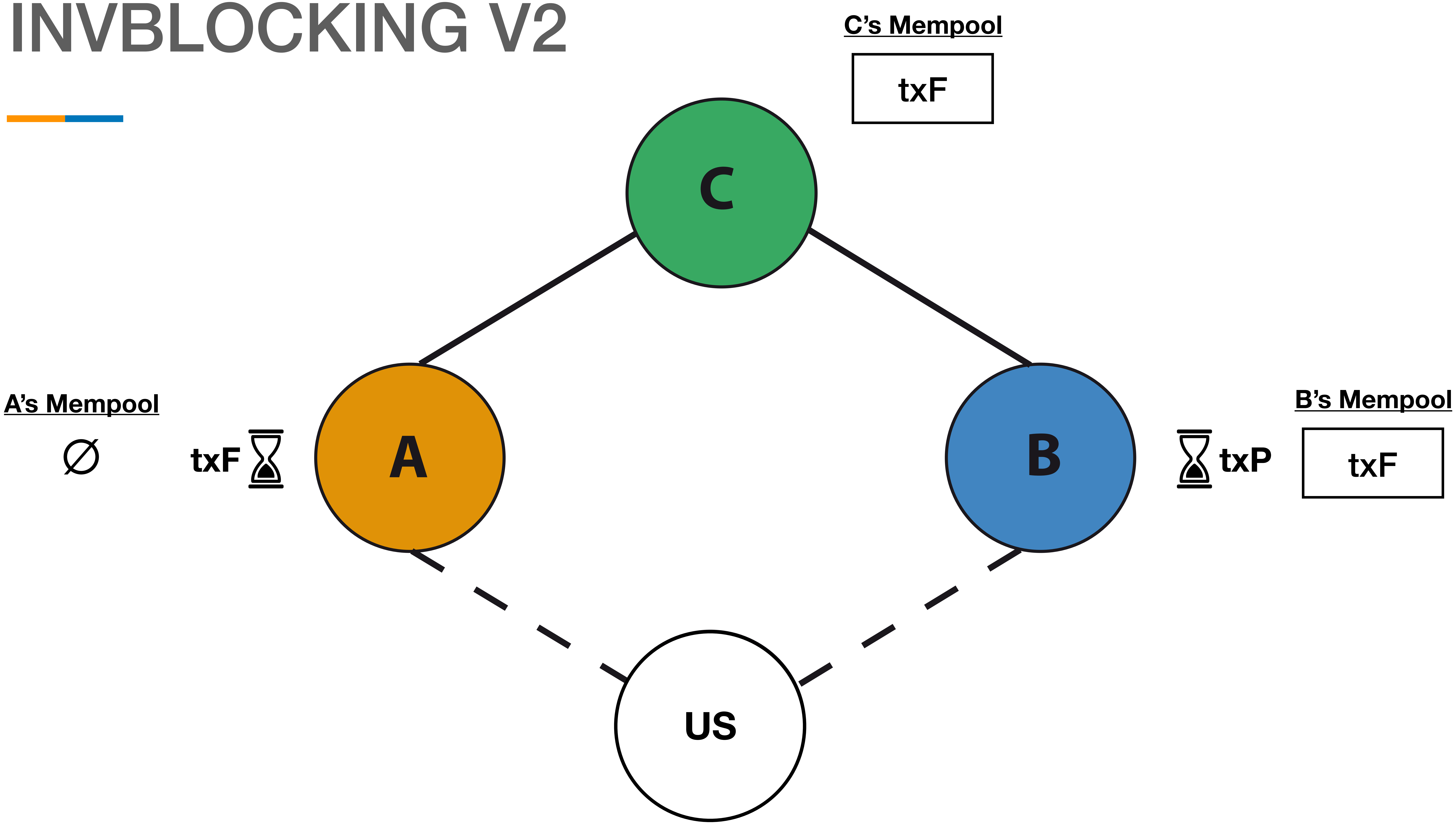
INVBLOCKING V2



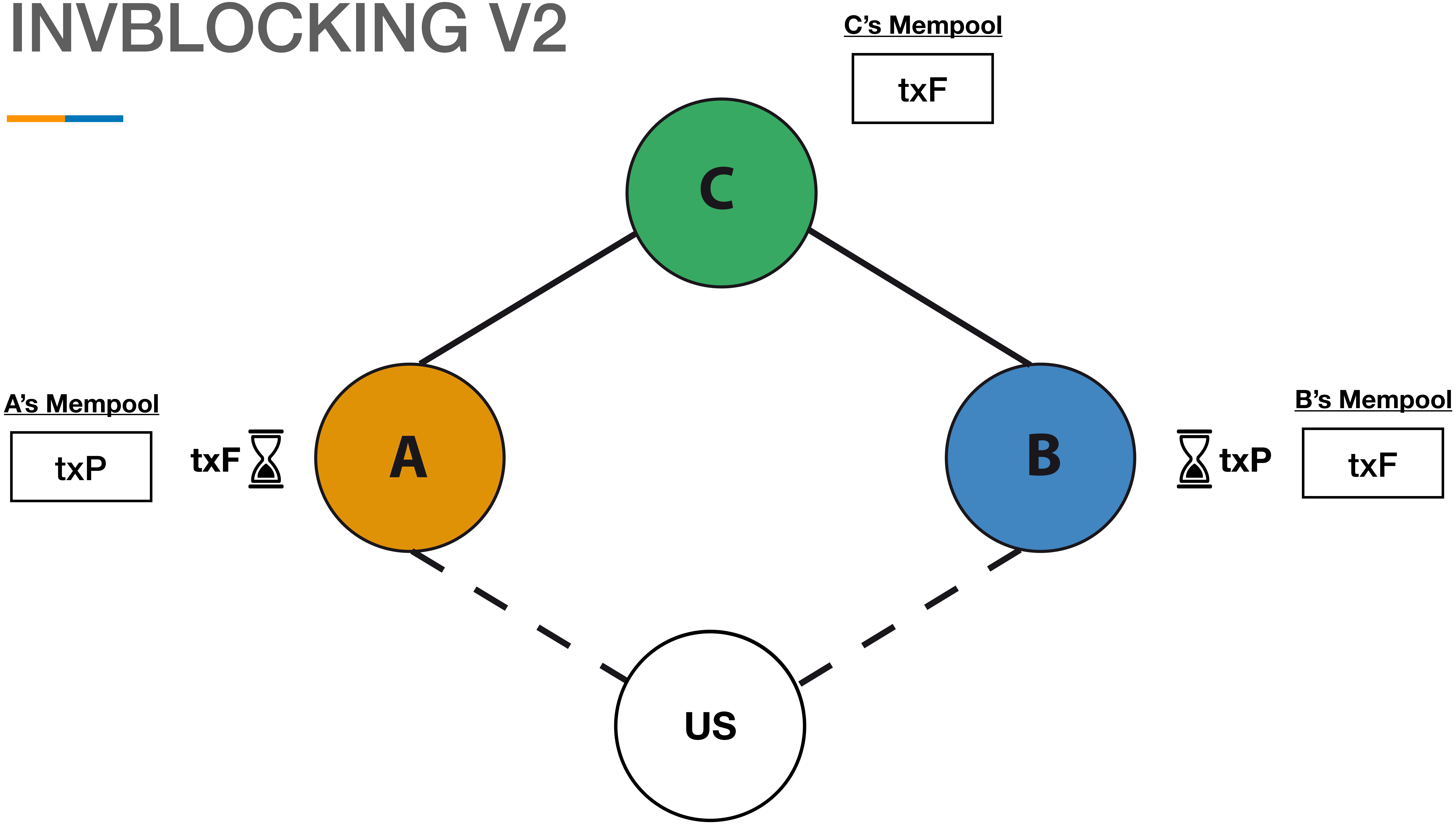
INVBLOCKING V2



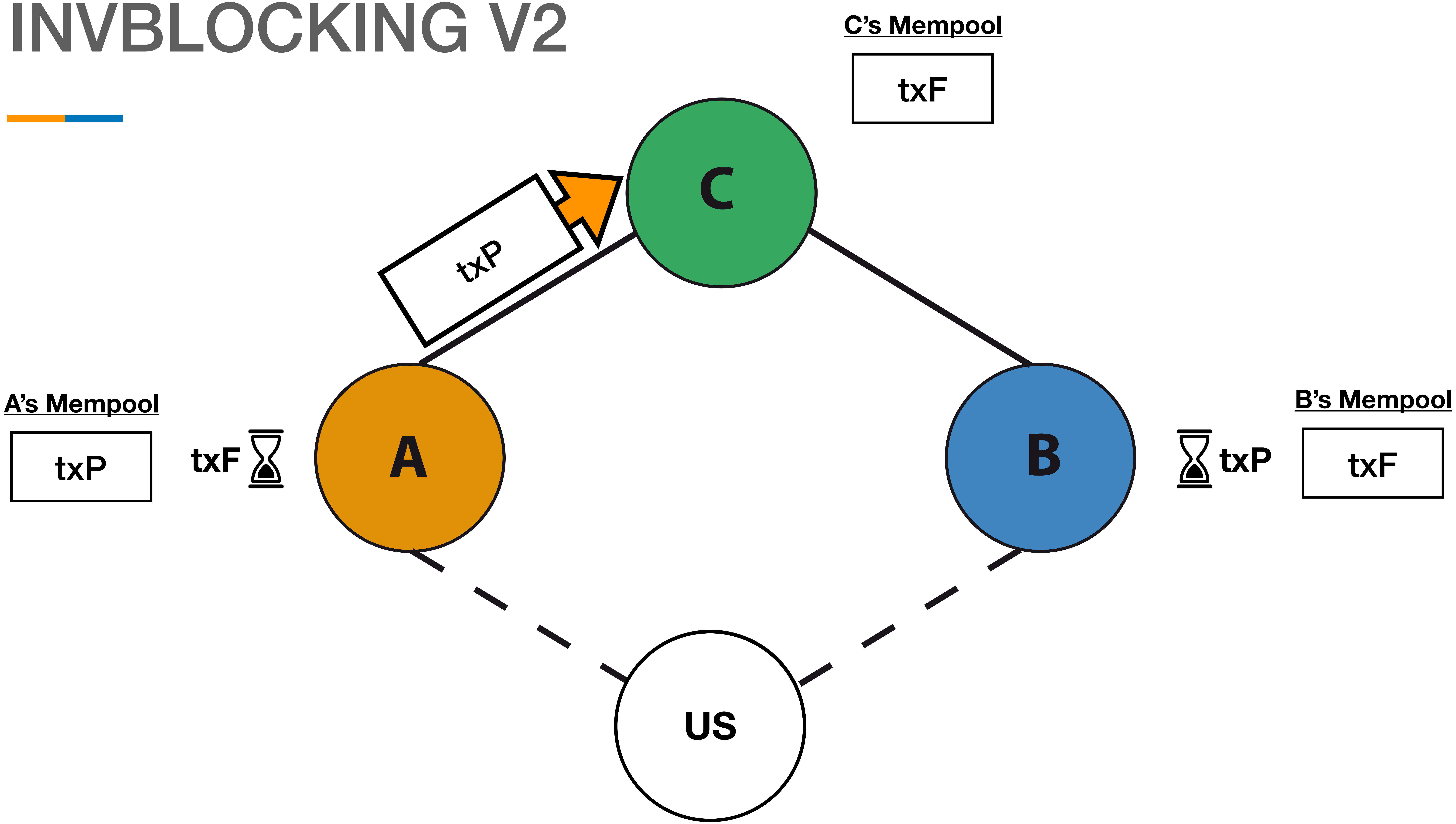
INVBLOCKING V2



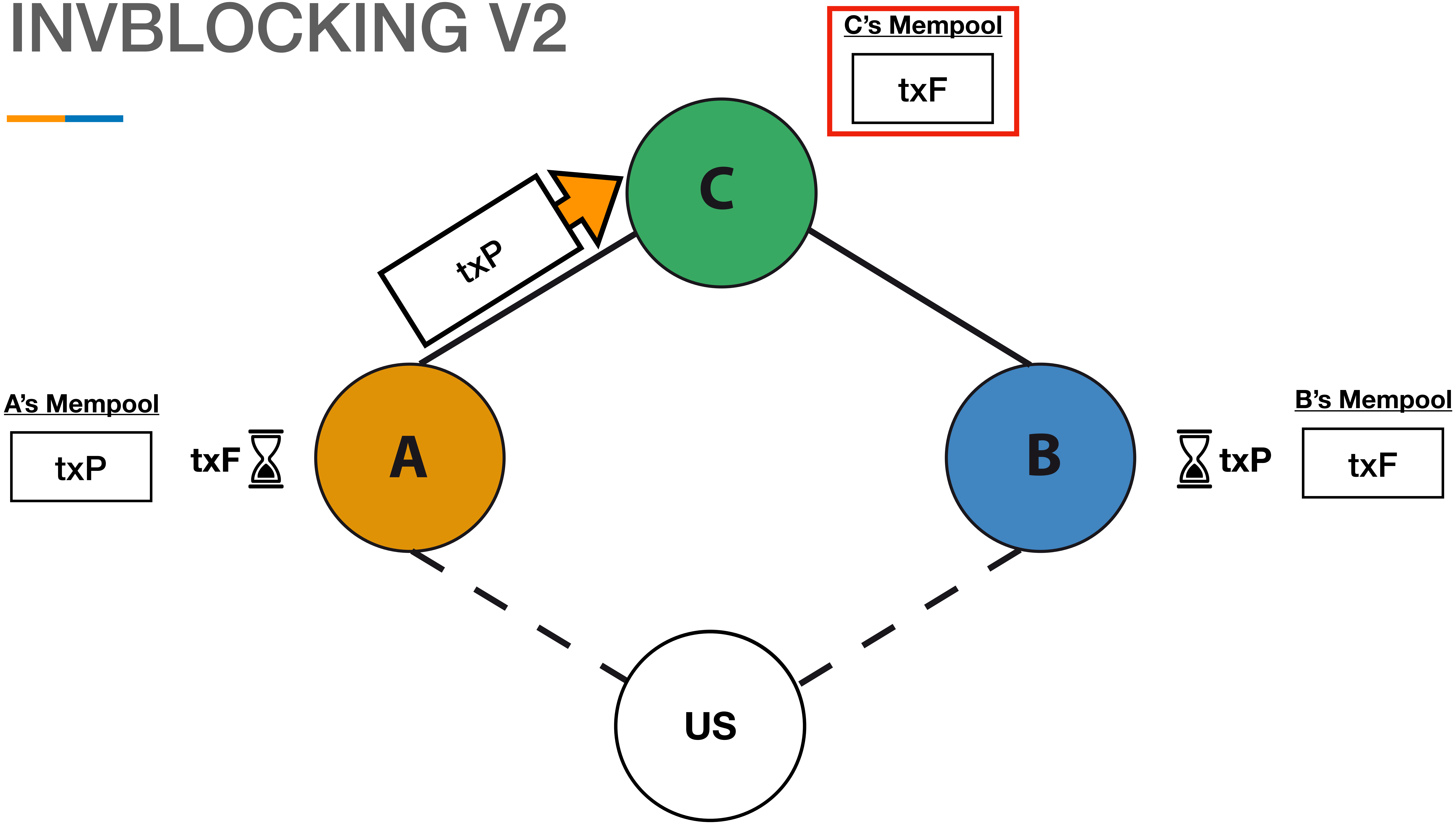
INVBLOCKING V2



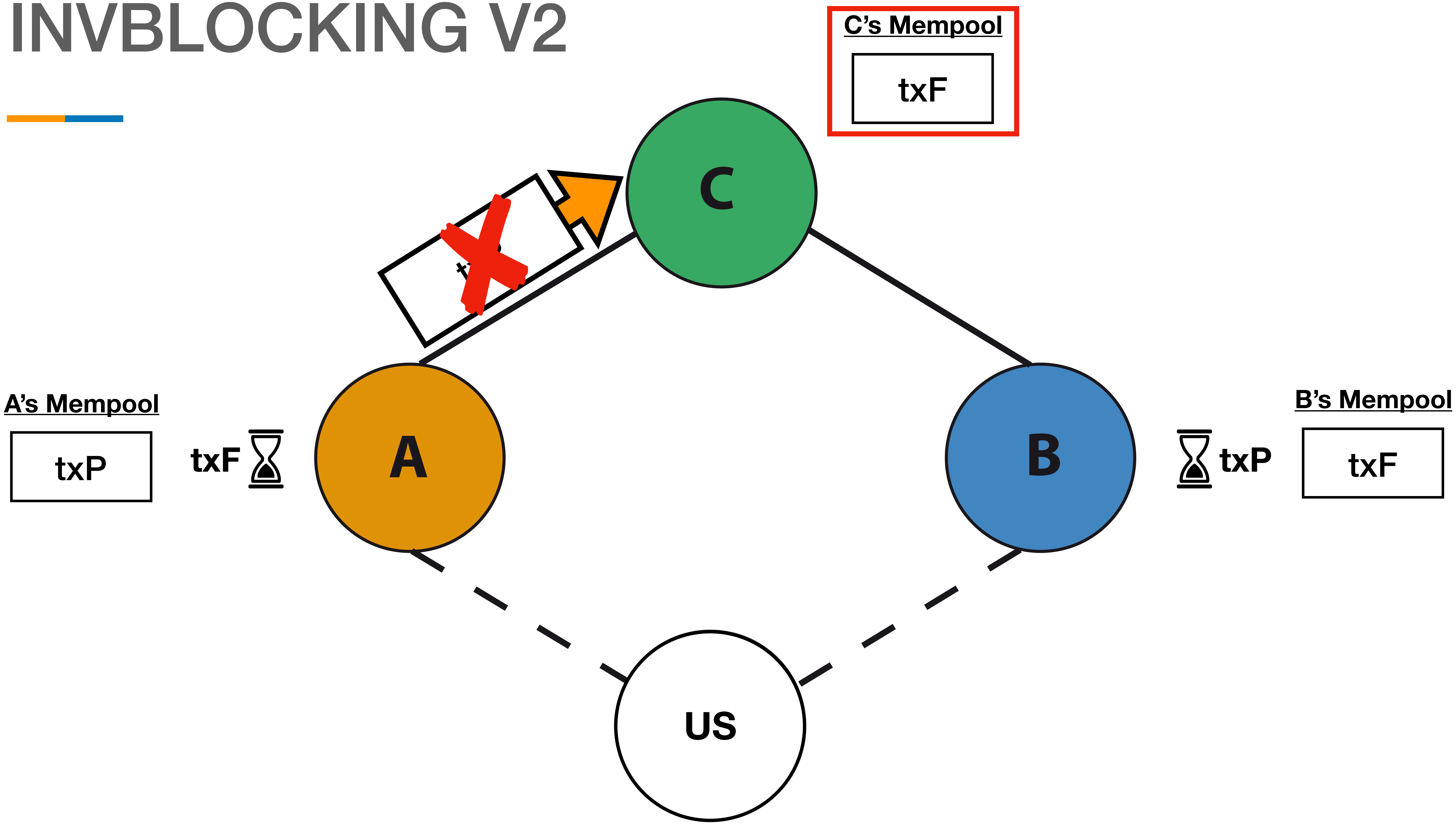
INVBLOCKING V2



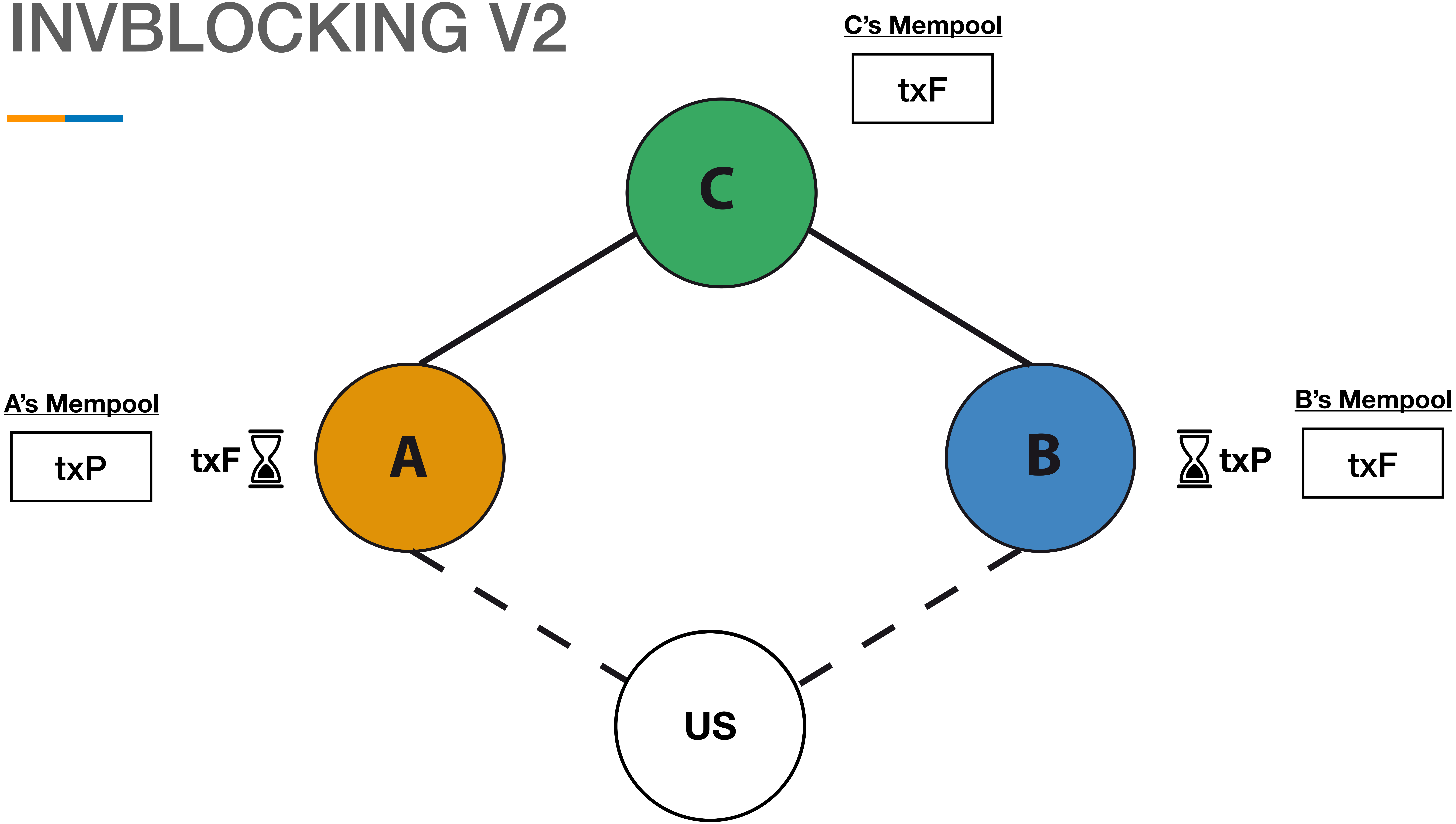
INVBLOCKING V2



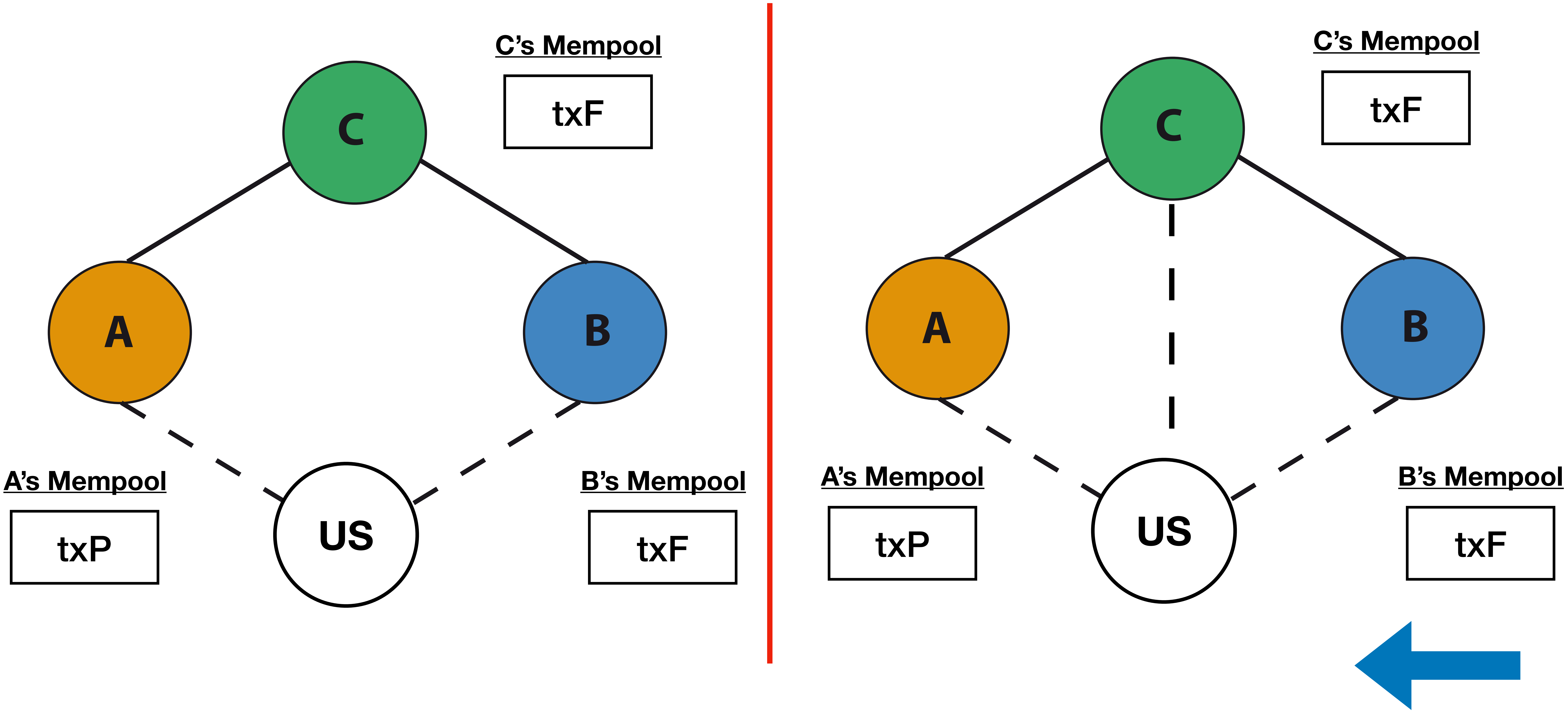
INVBLOCKING V2



INVBLOCKING V2



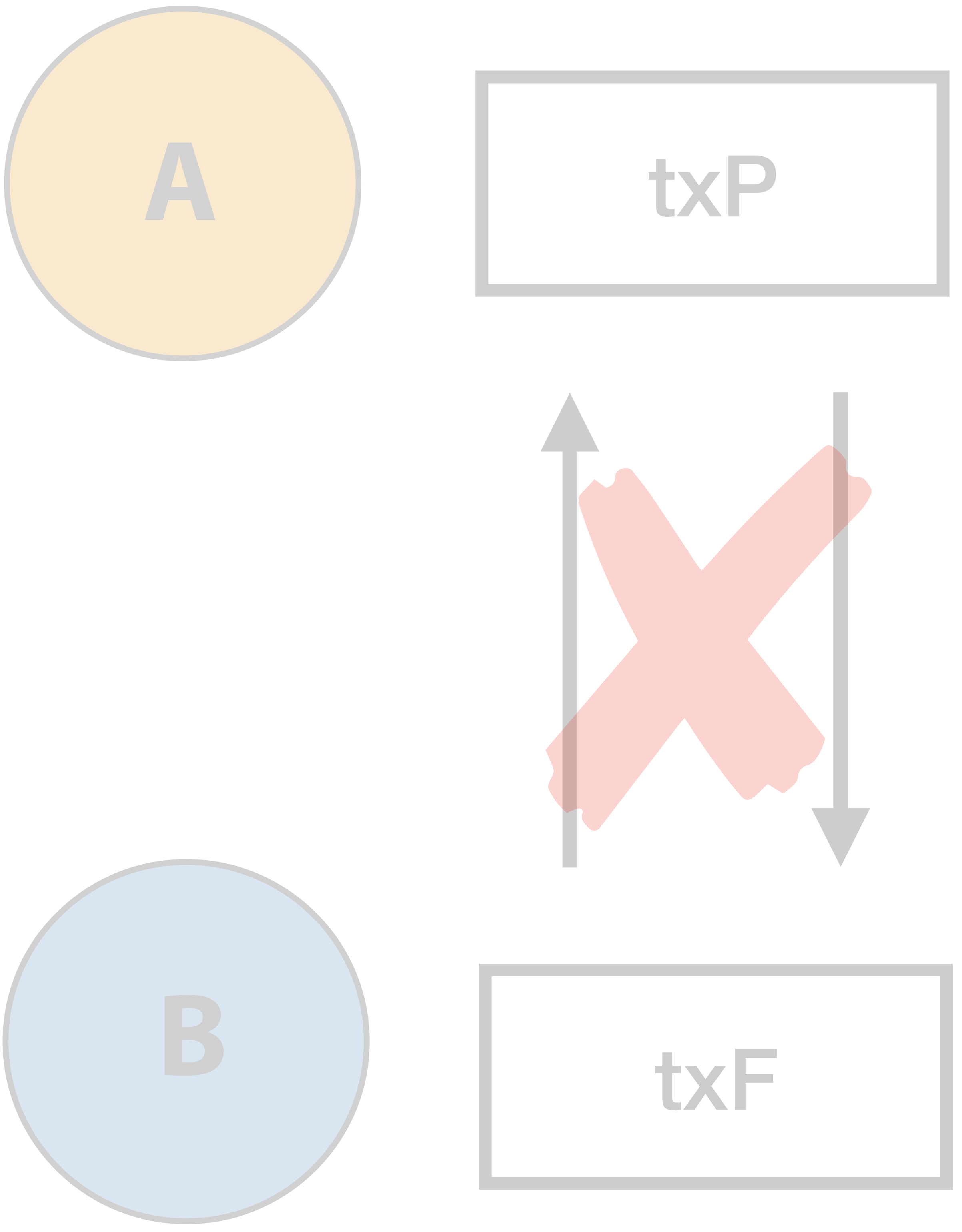
INVBLOCKING COMPARISON



MAKE THIS WORK IN A REAL NETWORK

Isolation

Synchrony



Efficiency

$$\approx O(n)$$

$$\downarrow$$

$$\approx O(\sqrt{n})$$

$$n = \#nodes$$

ORPHANPOOL EVICTION (BEFORE v0.18)

```
while (mapOrphanTransactions.size() > nMaxOrphans)
{
    // Evict a random orphan:
    uint256 randomhash = rng.rand256();
    std::map<uint256, COrphanTx>::iterator it = mapOrphanTransactions.lower_bound(randomhash);
    if (it == mapOrphanTransactions.end())
        it = mapOrphanTransactions.begin();
    EraseOrphanTx(it->first);
    ++nEvicted;
}
```

source: https://github.com/bitcoin/bitcoin/blob/273d025/src/net_processing.cpp#L783-L791

ORPHANPOOL EVICTION (BEFORE v0.18)

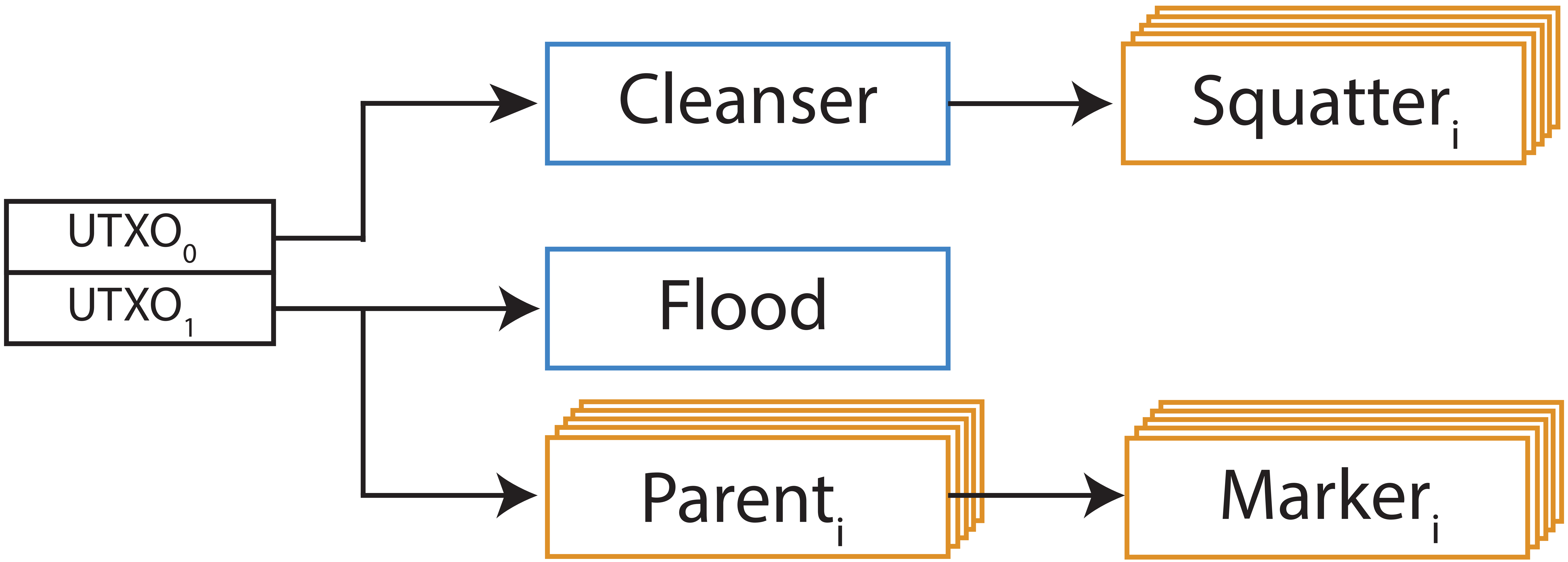
```
while (mapOrphanTransactions.size() > nMaxOrphans)
{
```

- Pick a random 256-bit value **R**
- Get the orphan transaction (**O**) with hash **closer to, but greater than, R**
- **Evict O**
- Repeat until mapOrphanTransaction **is not full (default: 100)**
- **Double-spends are not checked for orphans**

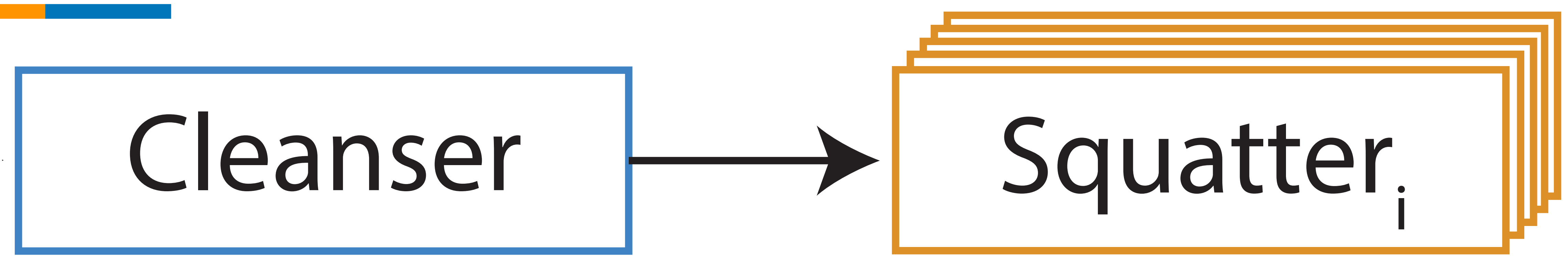
```
}
```

source: https://github.com/bitcoin/bitcoin/blob/273d025/src/net_processing.cpp#L783-L791

TXPROBE TRANSACTIONS OVERVIEW



MAKE ROOM IN THE ORPHANPOOL



- Create the cleanser (regular transaction) and **100 squatters (double-spends between each other)**
- Every squatter is created in a POW-ish way (e.g. re-sign until its hash falls bellow a certain threshold)
- All squatters are sent to the **flood set nodes** to replace any existing orphan.
- Finally, the cleanser is sent to empty the orphanpool

QUESTIONS