Network Layer and Information Propagation

Sergi Delgado Segura

What are we going to cover

- How a new node joins the network (how it learns about the network and how others learn about it)
- How data is propagated throughout the network
- What can go wrong
- How the network looks like

Before we start

We will use Bitcoin as example when explaining how certain parts of the network work. However, the same mechanisms apply to most of the existing cryptocurrencies with slight modifications (some times even without any).

Also keep in mind that for most of things within cryptocurrencies there is no formal specification but the live code. Therefore some details may change in the near future.

Introduction

Client-Server paradigm (1/2)

- Classic paradigm
- Actors are split in two types:
  - Servers:
    - serve specific resources upon request
    - can also provide different types of services

Client-Server paradigm (2/2)

- Clients:
  - resource/service requesters
  - do not share resources nor provide any service
- Clients initiate the communication and need to know the server endpoint
- Classical examples: WWW, DNS, Email, etc

Peer-to-peer (P2P) paradigm

- All actors (**peers**) are equal and have both client and server capabilities
- Services / resources can be shared between several peers or found in a single location
- Each peer can choose what to serve/request
- Quite usual paradigm for distributed file sharing (e.g: BitTorrent)
- **Usual problems:** Bootstrapping and file searching

P2P bootstrapping

P2P bootstrapping

- How do you find peers when you run a new node in the network?

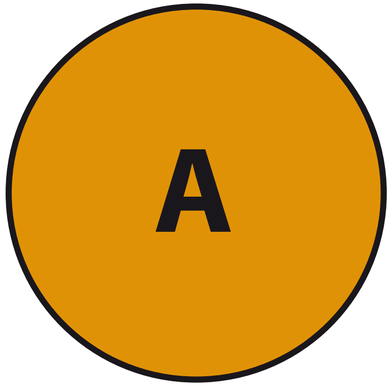
P2P bootstrapping

- How do you find peers when you run a new node in the network?

- How peers announce its presence in the network?

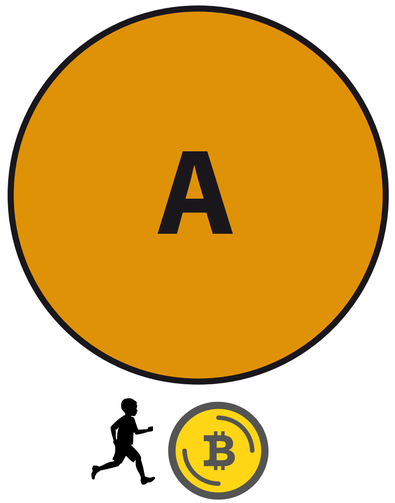
Peer discovery?

Peer discovery?

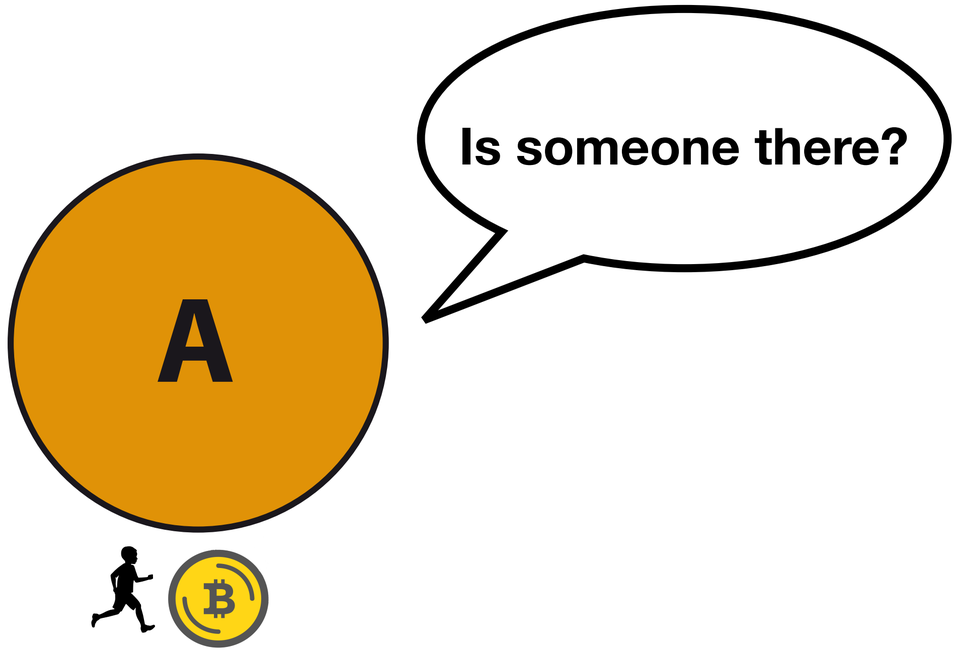


Peer discovery?

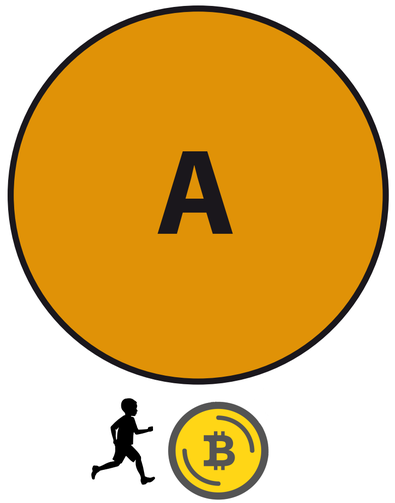
Peer discovery?



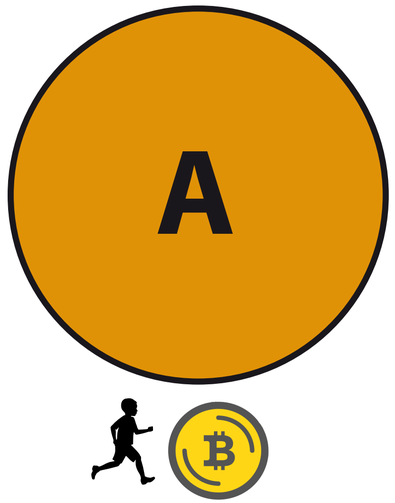
Peer discovery?



Peer discovery?

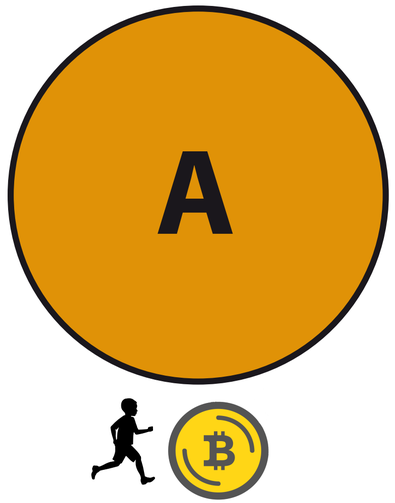


Peer discovery?



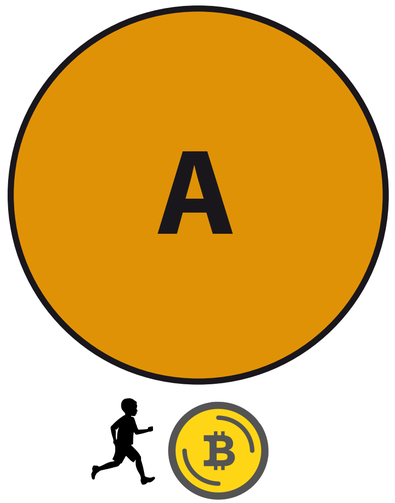
tumbleweed

Peer discovery?



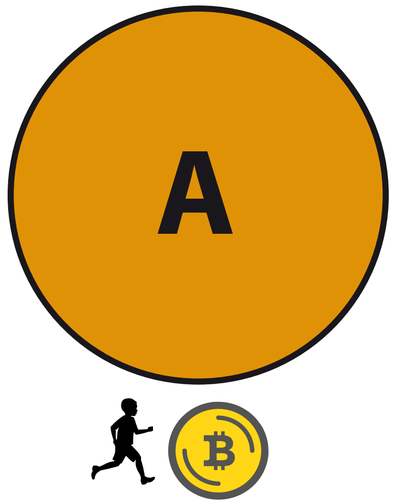
tumbleweed

Peer discovery?



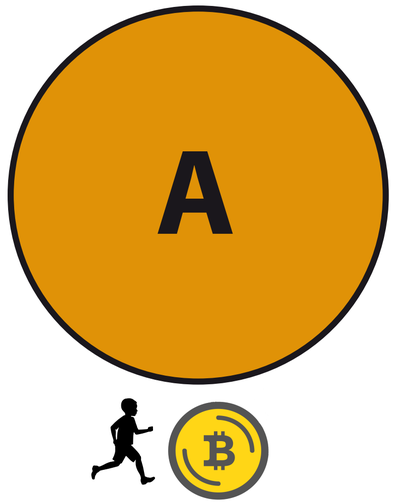
tumbleweed

Peer discovery?



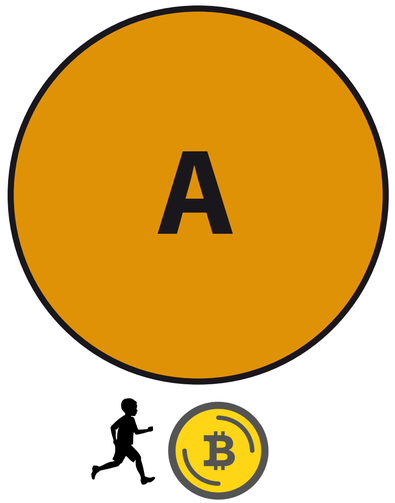
tumbleweed

Peer discovery?

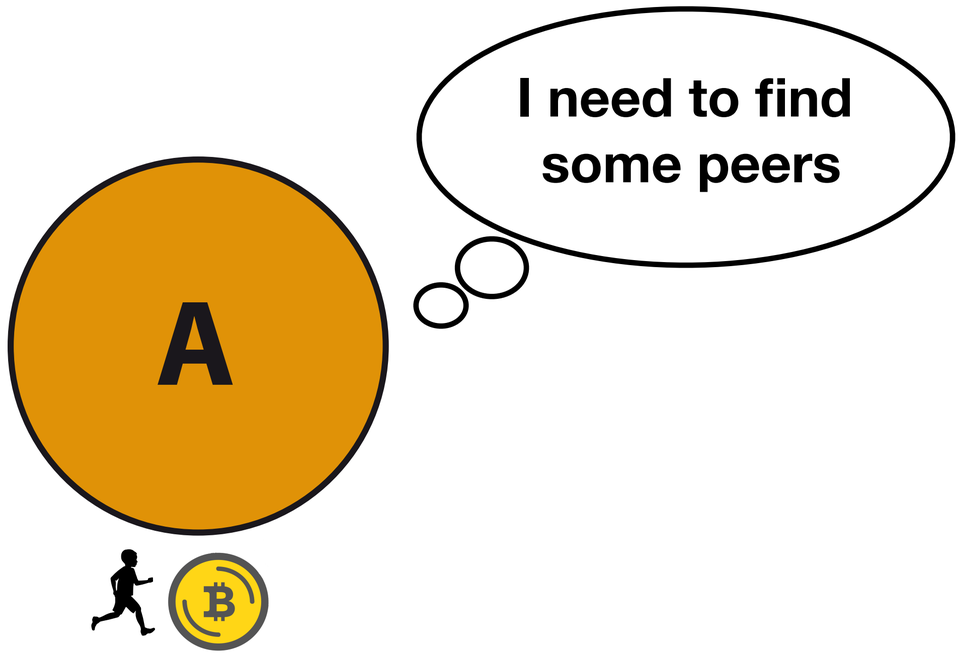


tumbleweed

Peer discovery?



Peer discovery?



P2P bootstrapping

- How do you find peers when you run a new node in the network?

- How peers announce its presence in the network?

P2P bootstrapping

- How do you find peers when you run a new node in the network?
- How peers announce its presence in the network?
- **Hardcoded trusted addresses / IRC bootstrapping / Trusted DNS seeds / etc**

P2P file sharing (1/2)

- How to identify what other nodes are sharing (who knows what)?

- How files are served?

P2P file sharing (1/2)

- How to identify what other nodes are sharing (who knows what)?
- How files are served?
- **Announce / Request**

P2P file sharing (2/2)

- **Request paradigm:** Files are requested by peers, so the network needs a lookup protocol to identify who knows what (e.g: DHT, trackers, etc)

- **Announce paradigm:** Files is announced to peers, which will decide whether they would like a copy or not. No lookup protocol is required (e.g: gossip protocols)

P2P file sharing (2/2)

- **Request paradigm:** Files are requested by peers, so the network needs a lookup protocol to identify who knows what (e.g: DHT, trackers, etc)

- **Announce paradigm:** Files is announced to peers, which will decide whether they would like a copy or not. No lookup protocol is required (e.g: gossip protocols)

- **What paradigm you do cryptocurrency networks follow?**

P2P file sharing (2/2)

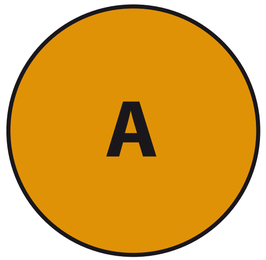
- **Request paradigm:** Files are requested by peers, so the network needs a lookup protocol to identify who knows what (e.g: DHT, trackers, etc)

- **Announce paradigm:** Files is announced to peers, which will decide whether they would like a copy or not. No lookup protocol is required (e.g: gossip protocols)

- **What paradigm you do cryptocurrency networks follow?** **Announce**

P2P file sharing: Request paradigm (BitTorrent)

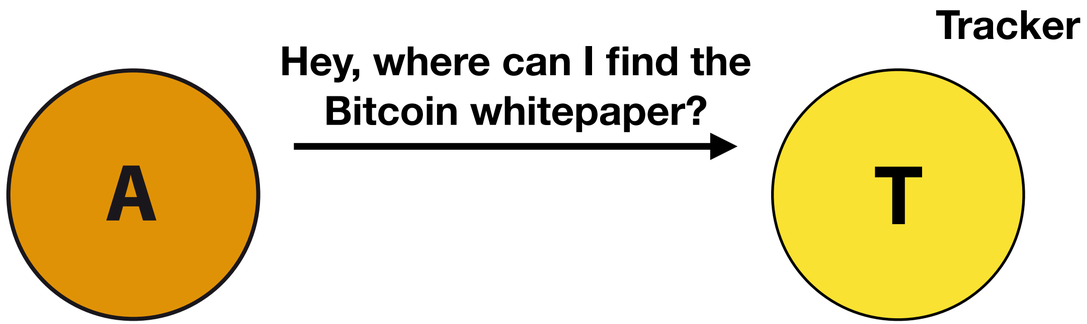
P2P file sharing: Request paradigm (BitTorrent)



P2P file sharing: Request paradigm (BitTorrent)

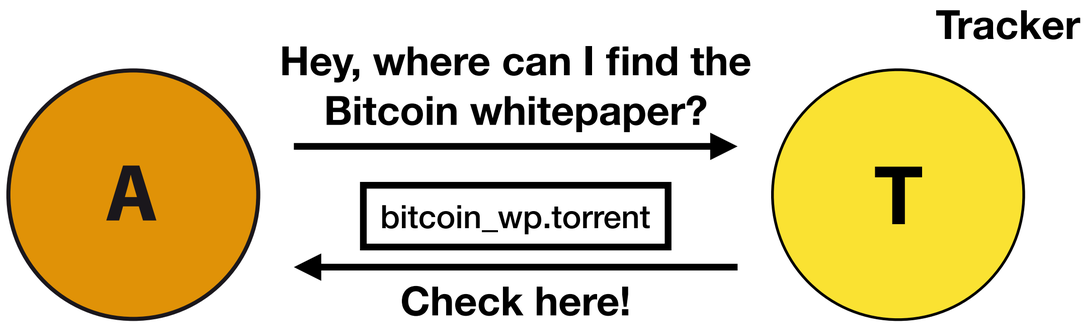
P2P file sharing: Request paradigm (BitTorrent)

- Get file information from a tracker



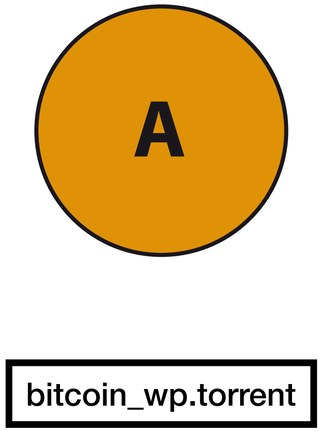
P2P file sharing: Request paradigm (BitTorrent)

- Get file information from a tracker



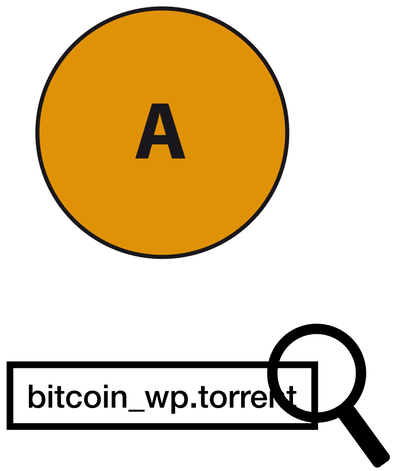
P2P file sharing: Request paradigm (BitTorrent)

- Get file information from a tracker

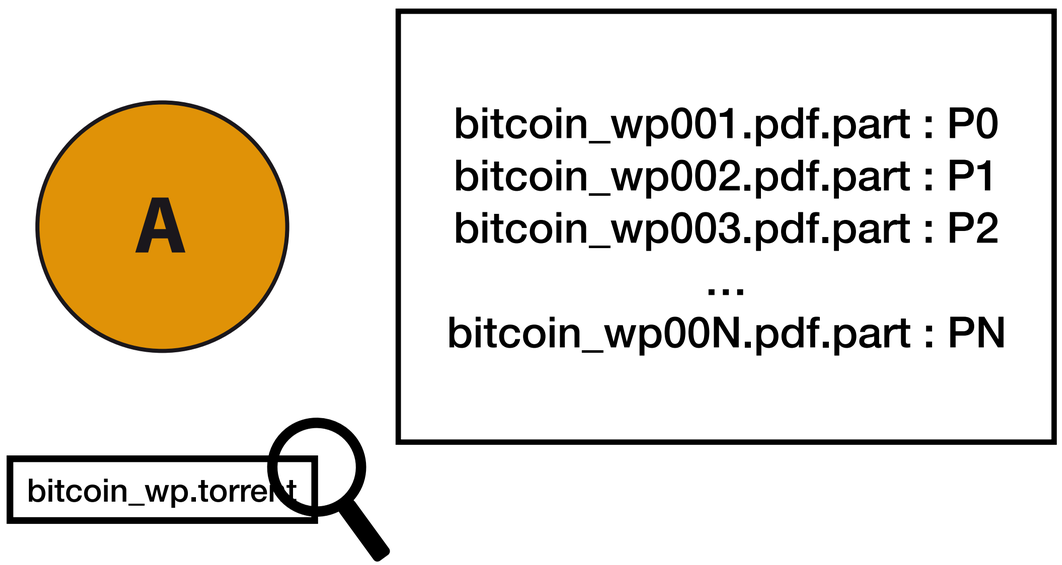


P2P file sharing: Request paradigm (BitTorrent)

- Get file information from a tracker
- Check the .torrent file

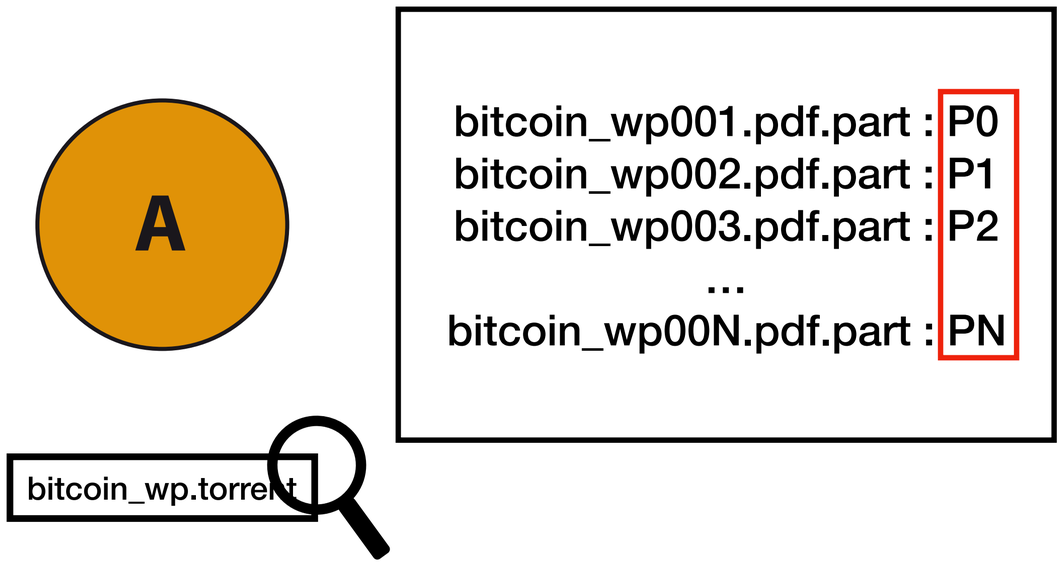


P2P file sharing: Request paradigm (BitTorrent)



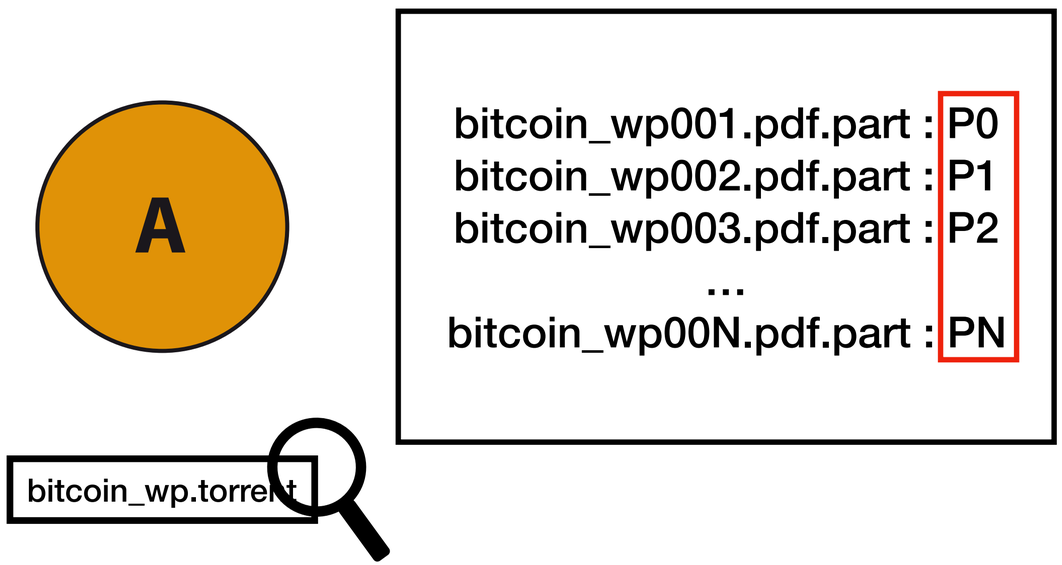
- Get file information from a tracker
- Check the .torrent file

P2P file sharing: Request paradigm (BitTorrent)



- Get file information from a tracker
- Check the .torrent file
- Connect to peers and retrieve the file parts

P2P file sharing: Request paradigm (BitTorrent)



- Get file information from a tracker
- Check the .torrent file
- Connect to peers and retrieve the file parts

We will cover the announce paradigm later on!

Announce vs request

Announce vs request

- Why a request paradigm (like the one we just saw) would not work for cryptocurrency networks?

Announce vs request

- Why a request paradigm (like the one we just saw) would not work for cryptocurrency networks?

  - **New items (transactions and blocks) can be created by others, so we can’t know about them if they are not offered**

Announce vs request

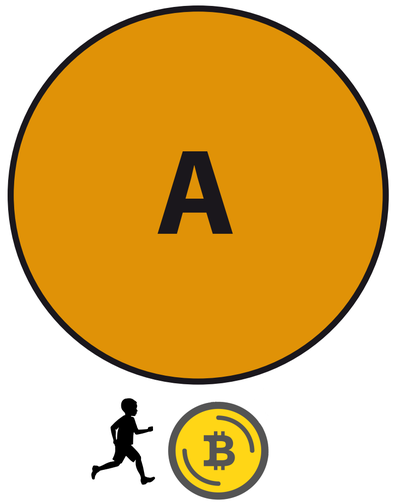
- Why a request paradigm (like the one we just saw) would not work for cryptocurrency networks?
  - **New items (transactions and blocks) can be created by others, so we can’t know about them if they are not offered**
- What information should a node know about the system?

Announce vs request

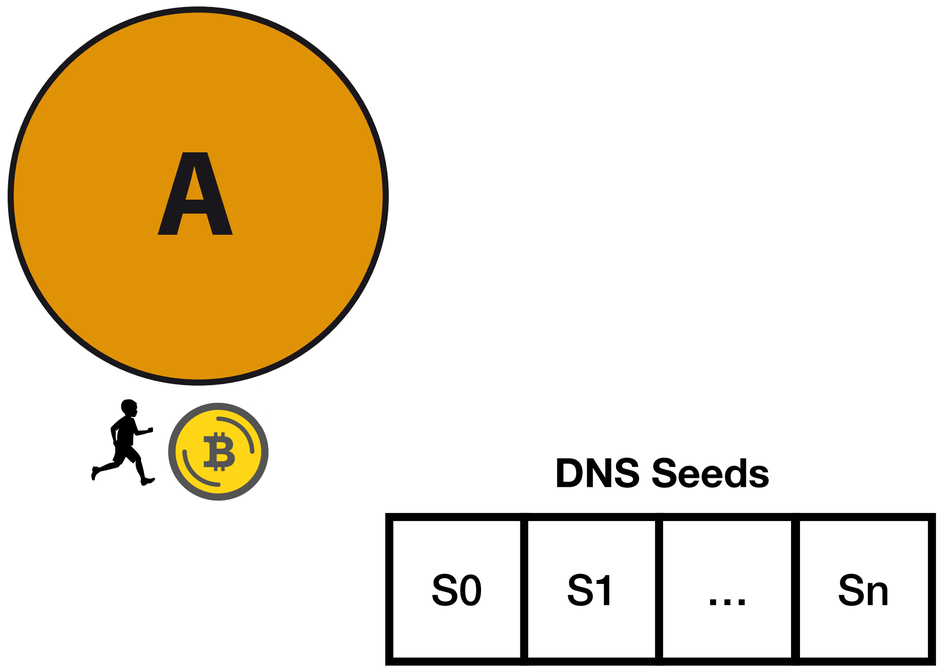
- Why a request paradigm (like the one we just saw) would not work for cryptocurrency networks?
  - **New items (transactions and blocks) can be created by others, so we can’t know about them if they are not offered**
- What information should a node know about the system?
  - **A (full) node needs all the information in order validate new items**

Node bootstrapping and peer discovery

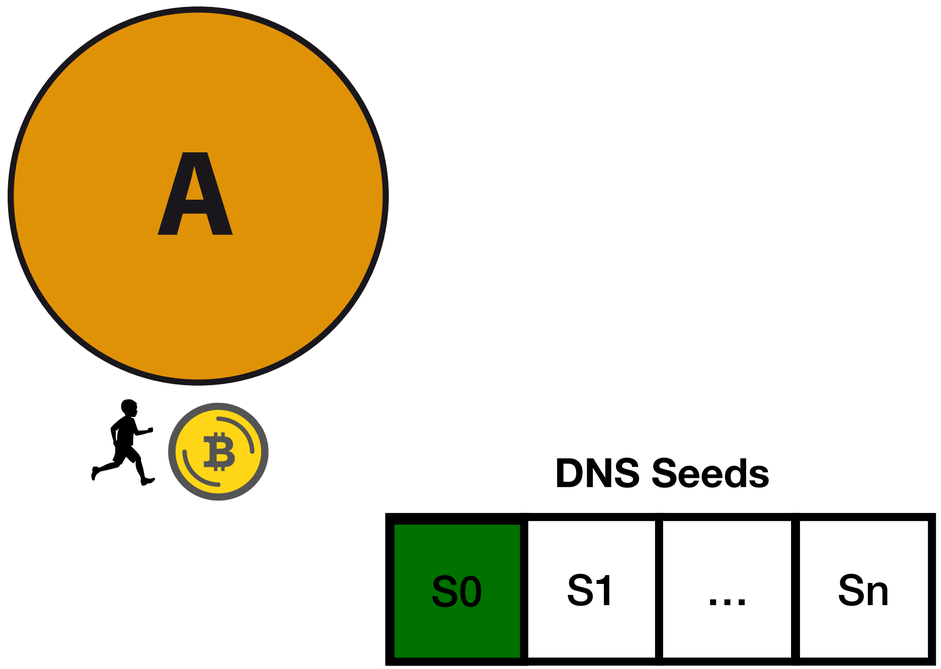
Bitcoin P2P bootstrapping (peer discovery)



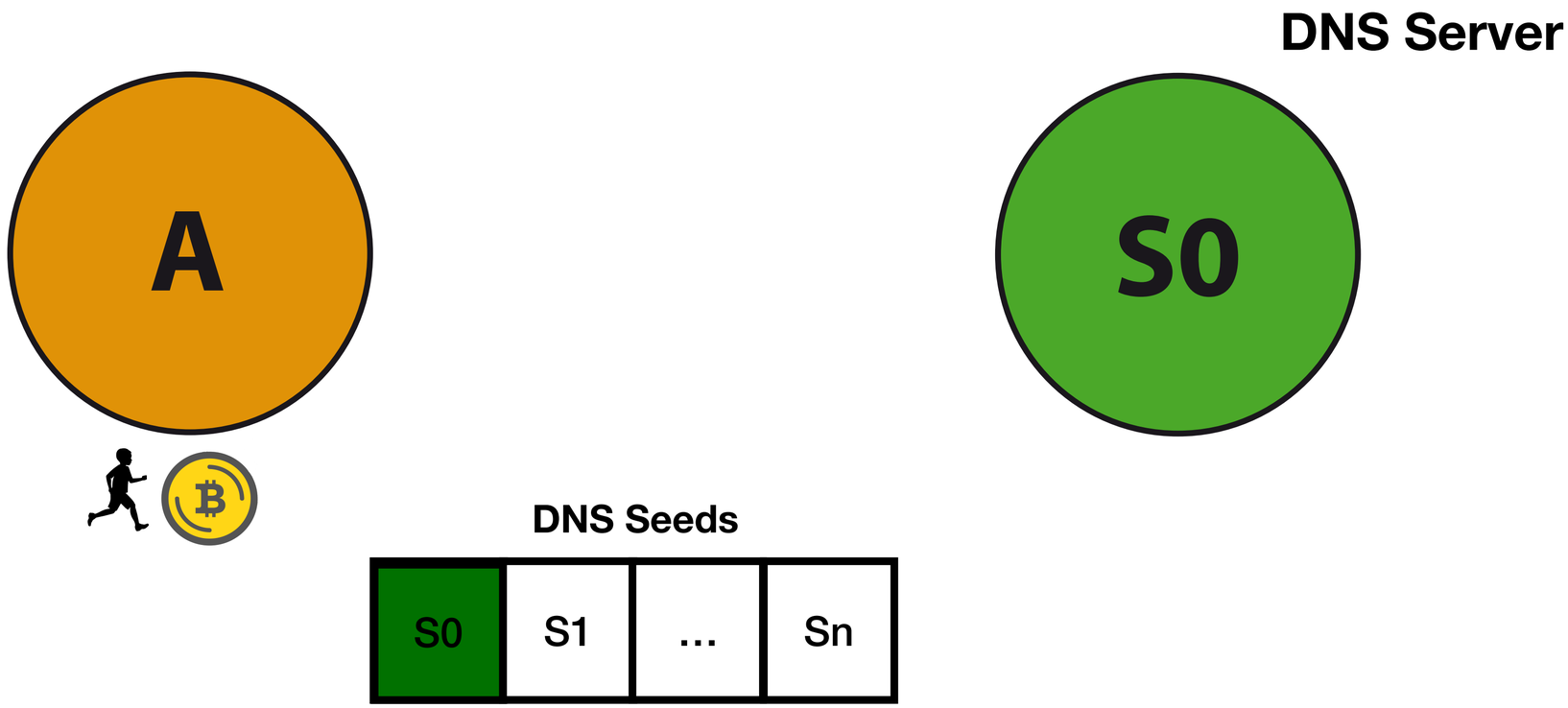
Bitcoin P2P bootstrapping (peer discovery)



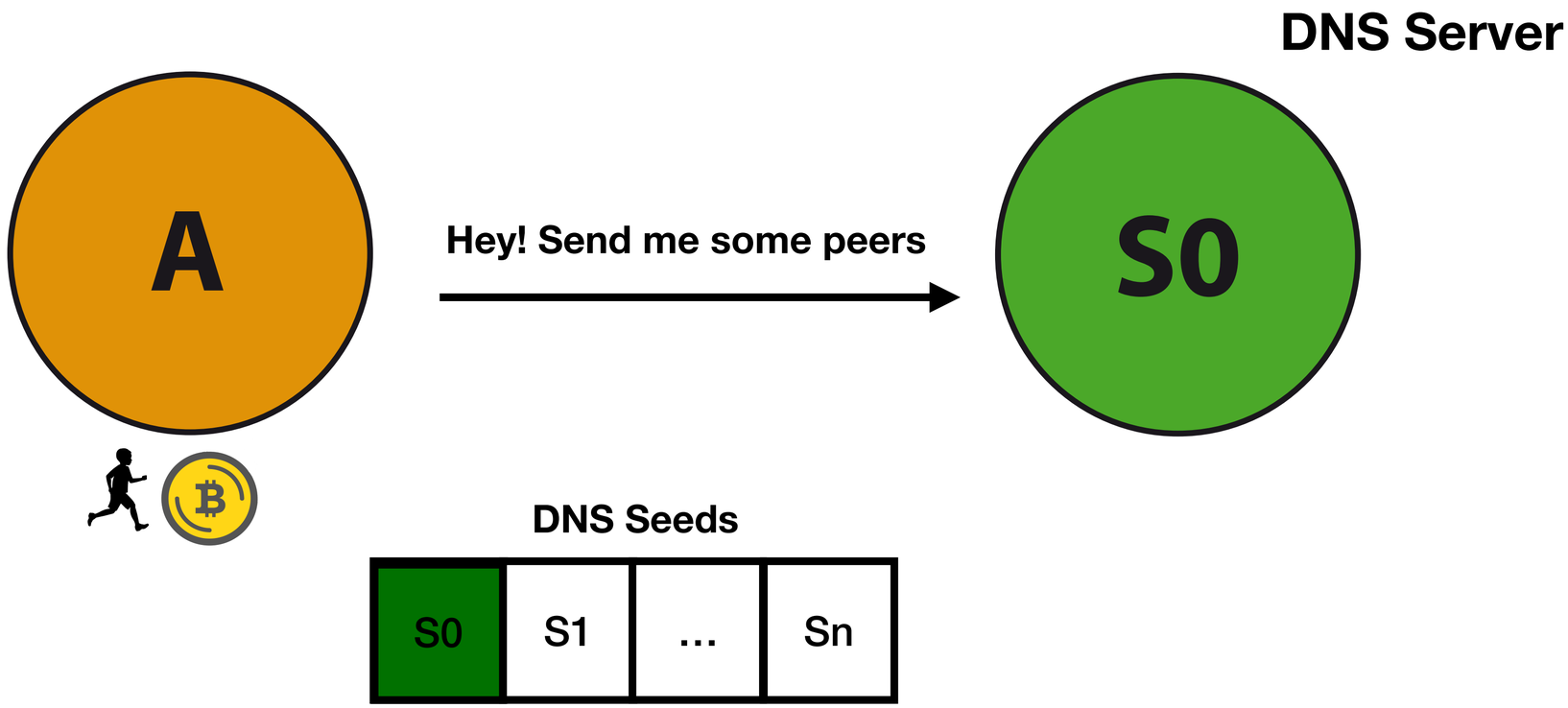
Bitcoin P2P bootstrapping (peer discovery)



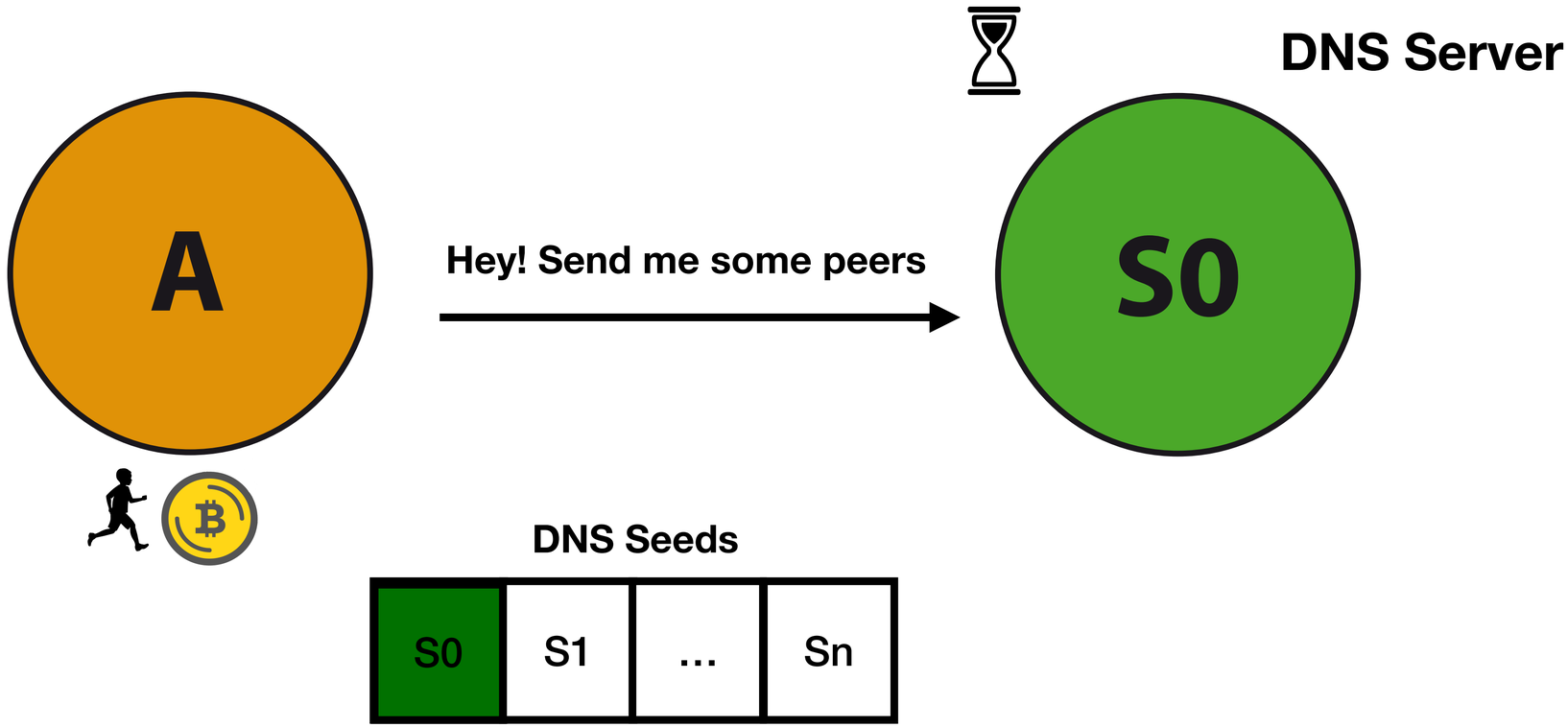
Bitcoin P2P bootstrapping (peer discovery)



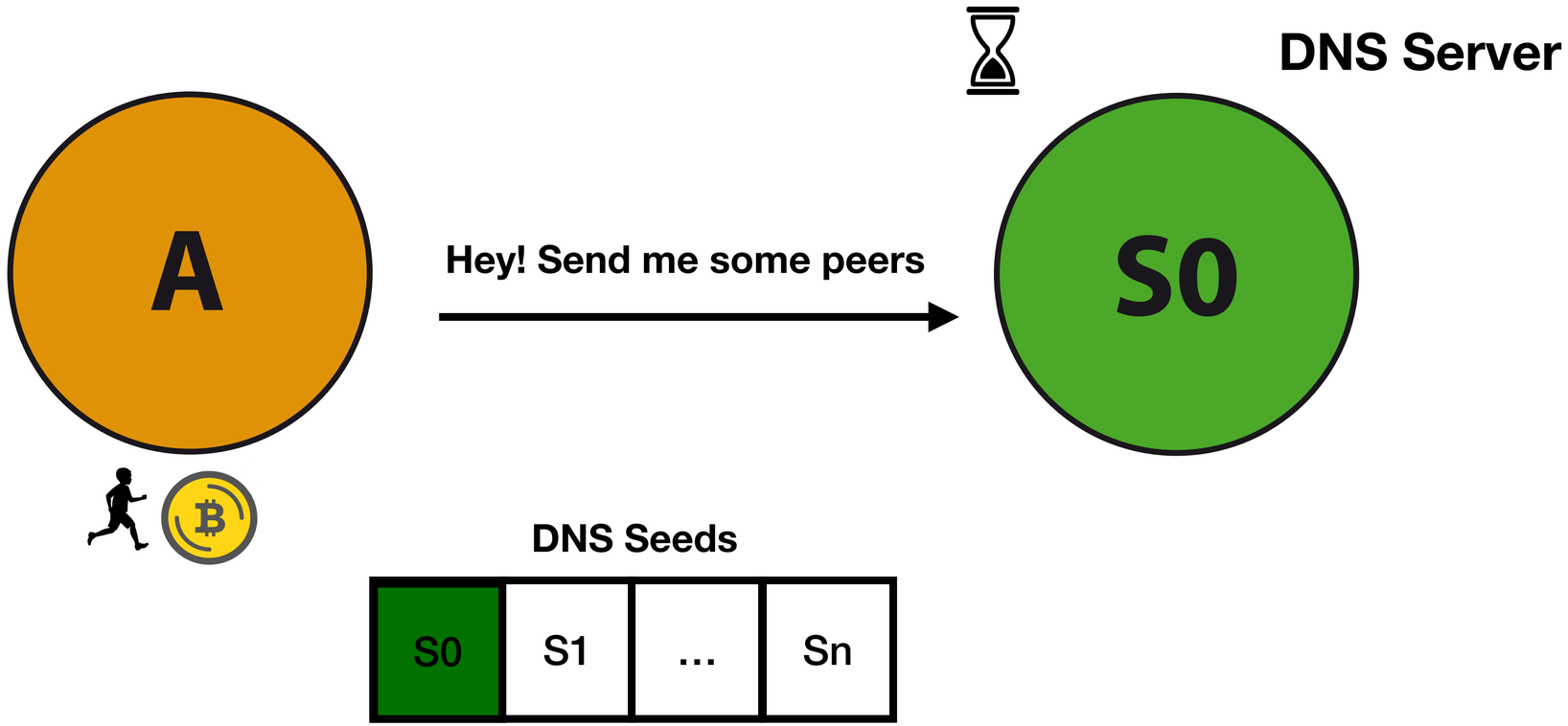
Bitcoin P2P bootstrapping (peer discovery)



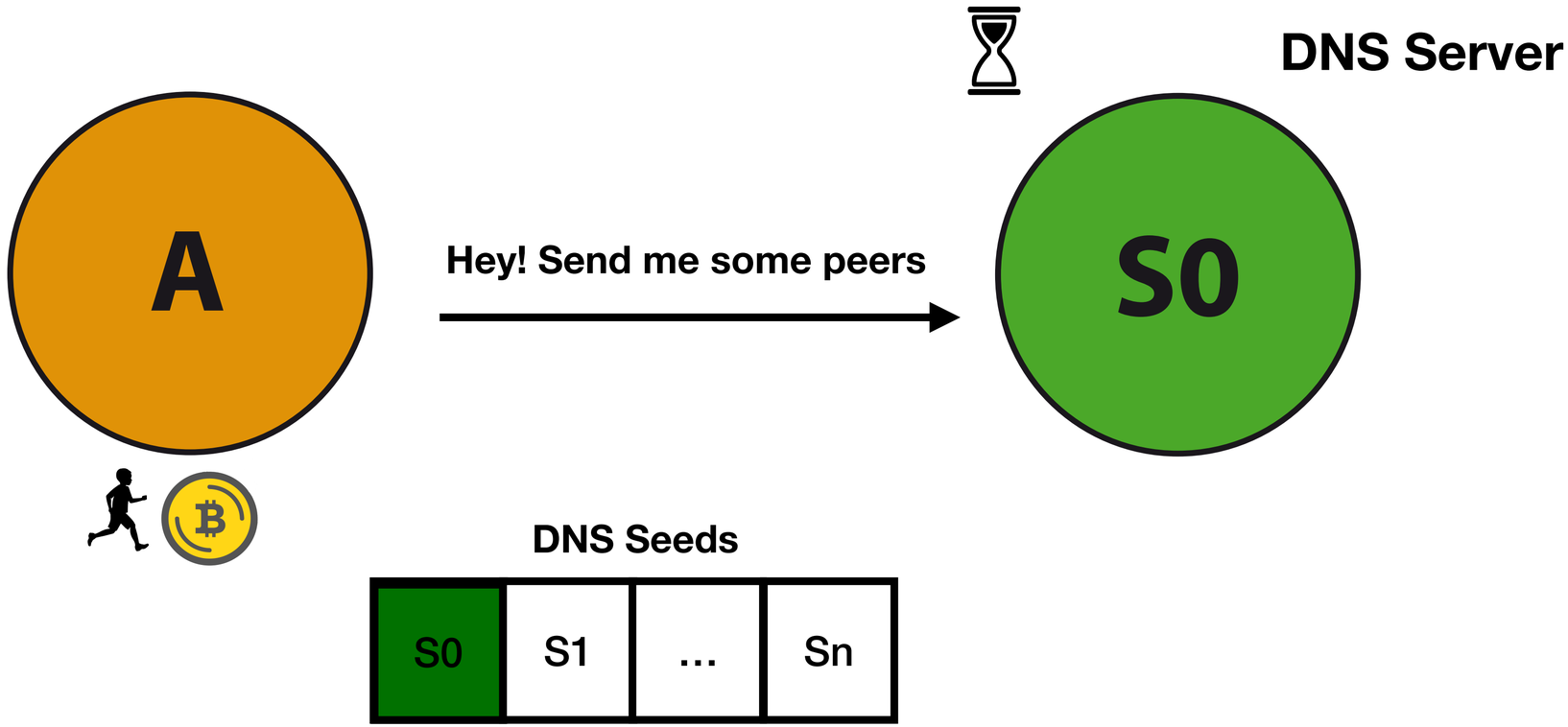
Bitcoin P2P bootstrapping (peer discovery)



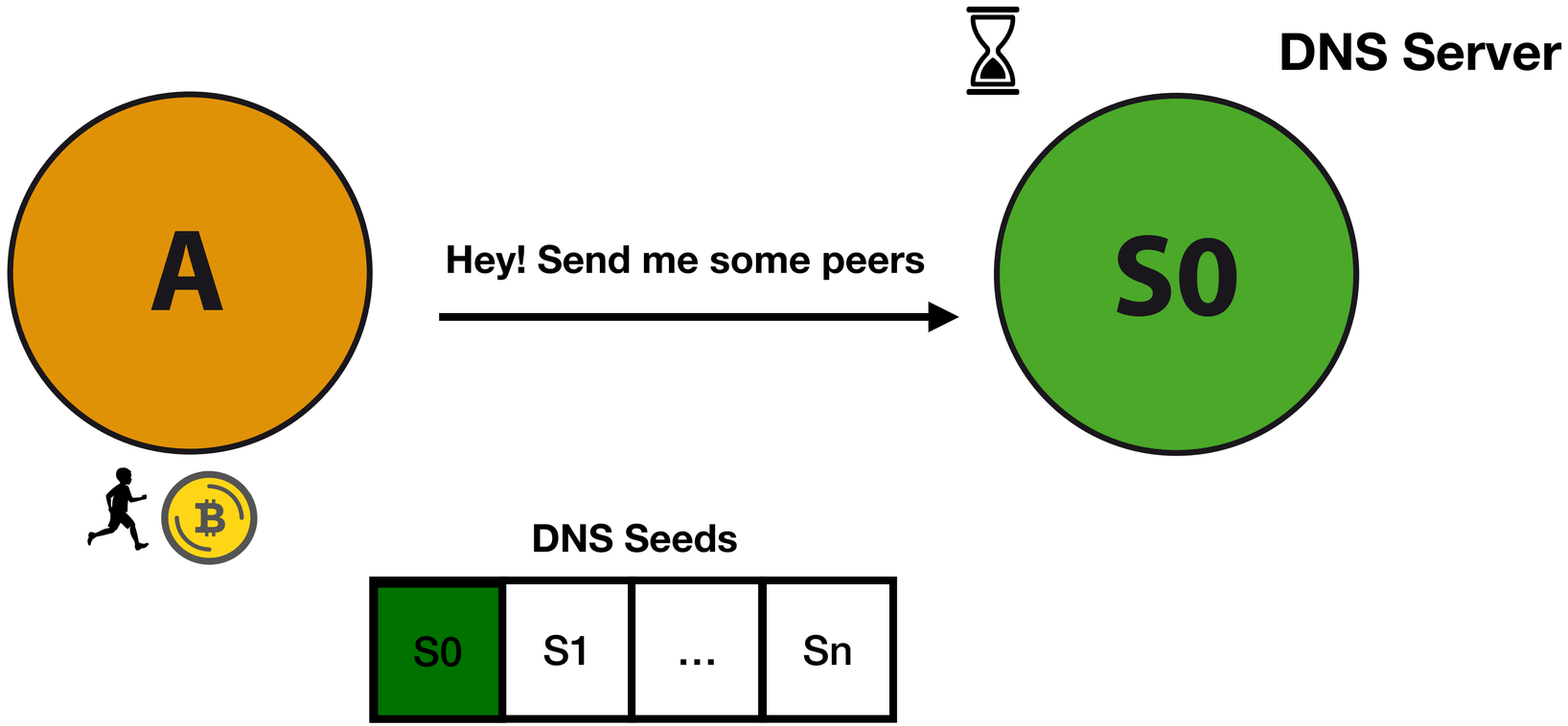
Bitcoin P2P bootstrapping (peer discovery)



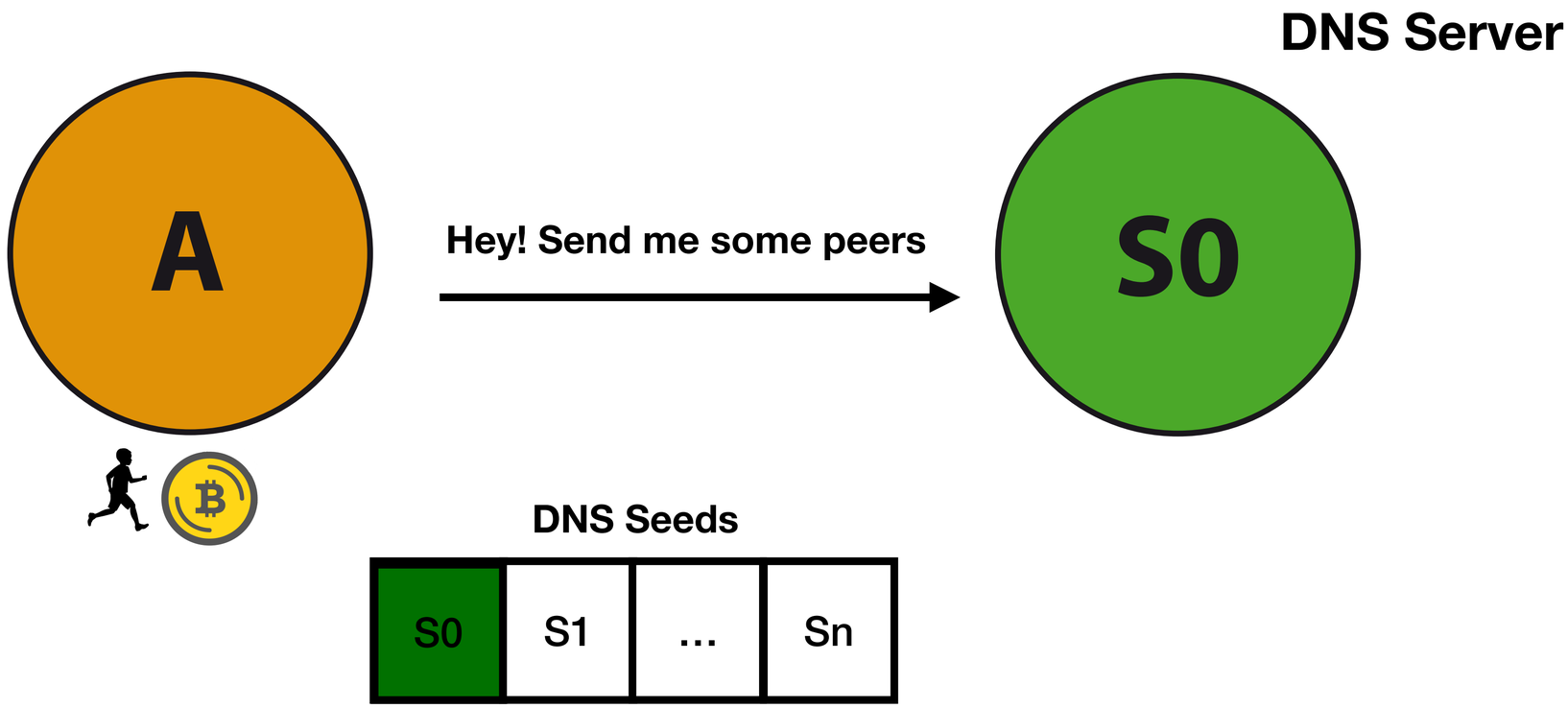
Bitcoin P2P bootstrapping (peer discovery)



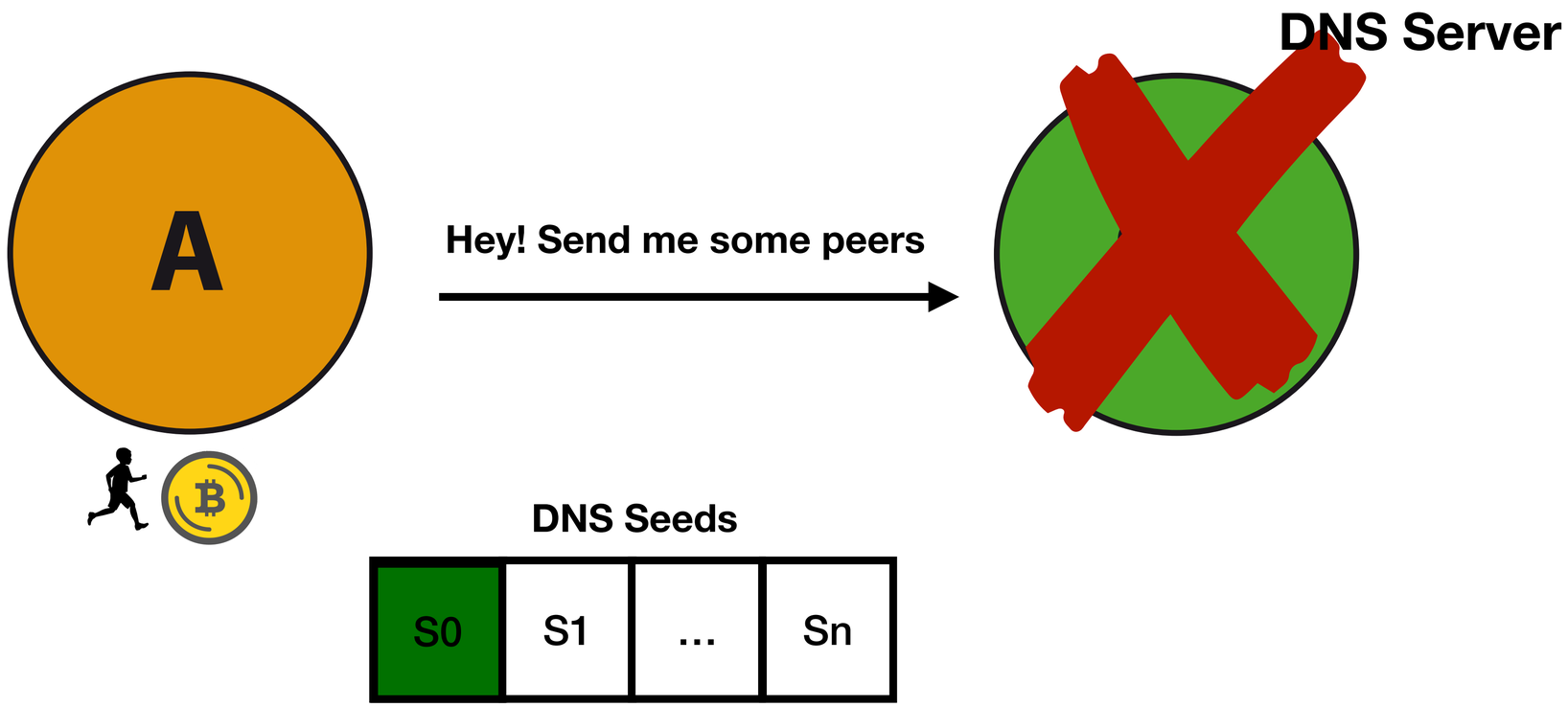
Bitcoin P2P bootstrapping (peer discovery)



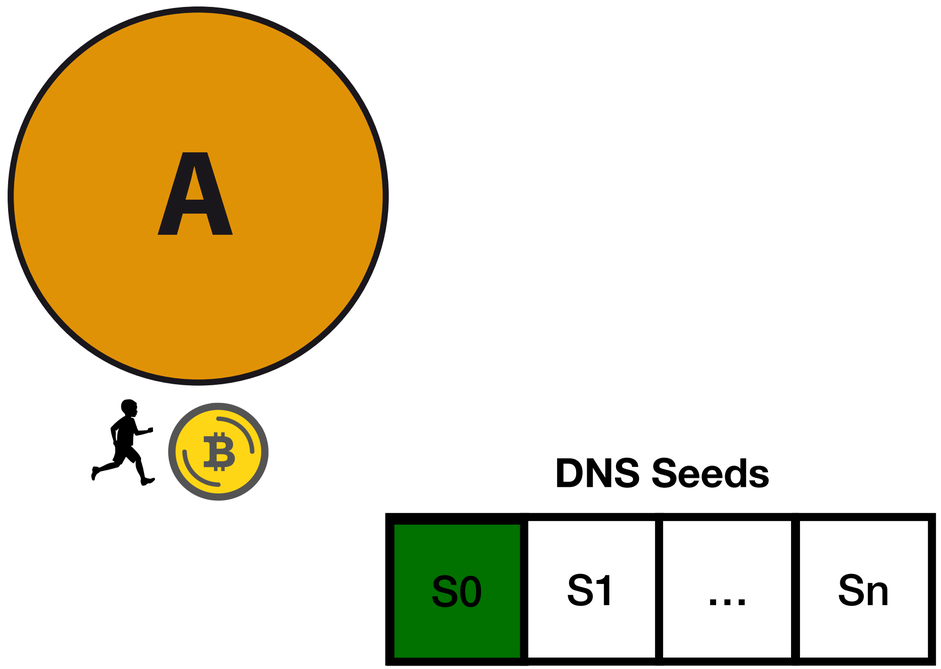
Bitcoin P2P bootstrapping (peer discovery)



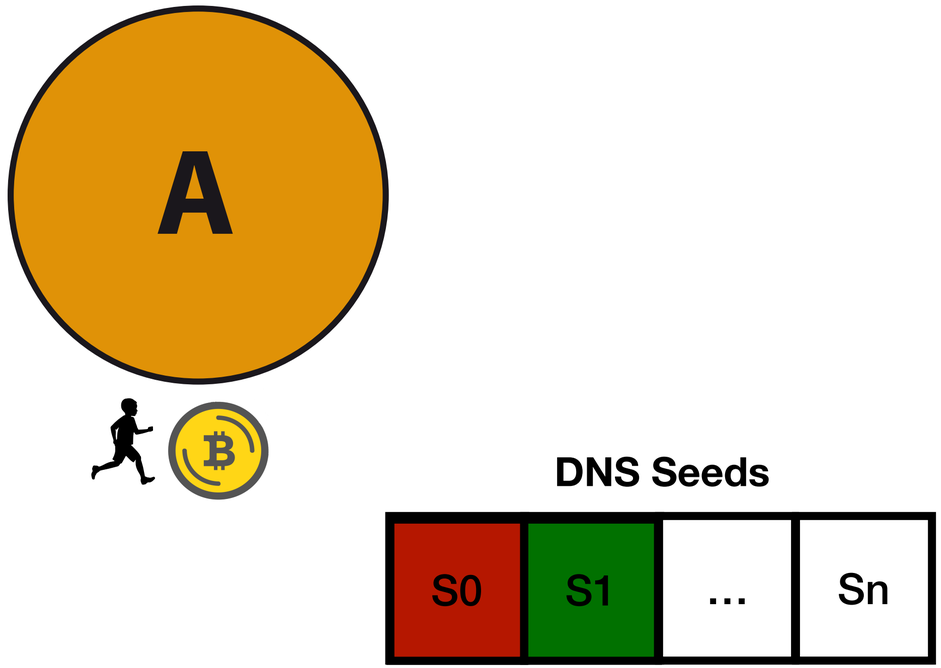
Bitcoin P2P bootstrapping (peer discovery)



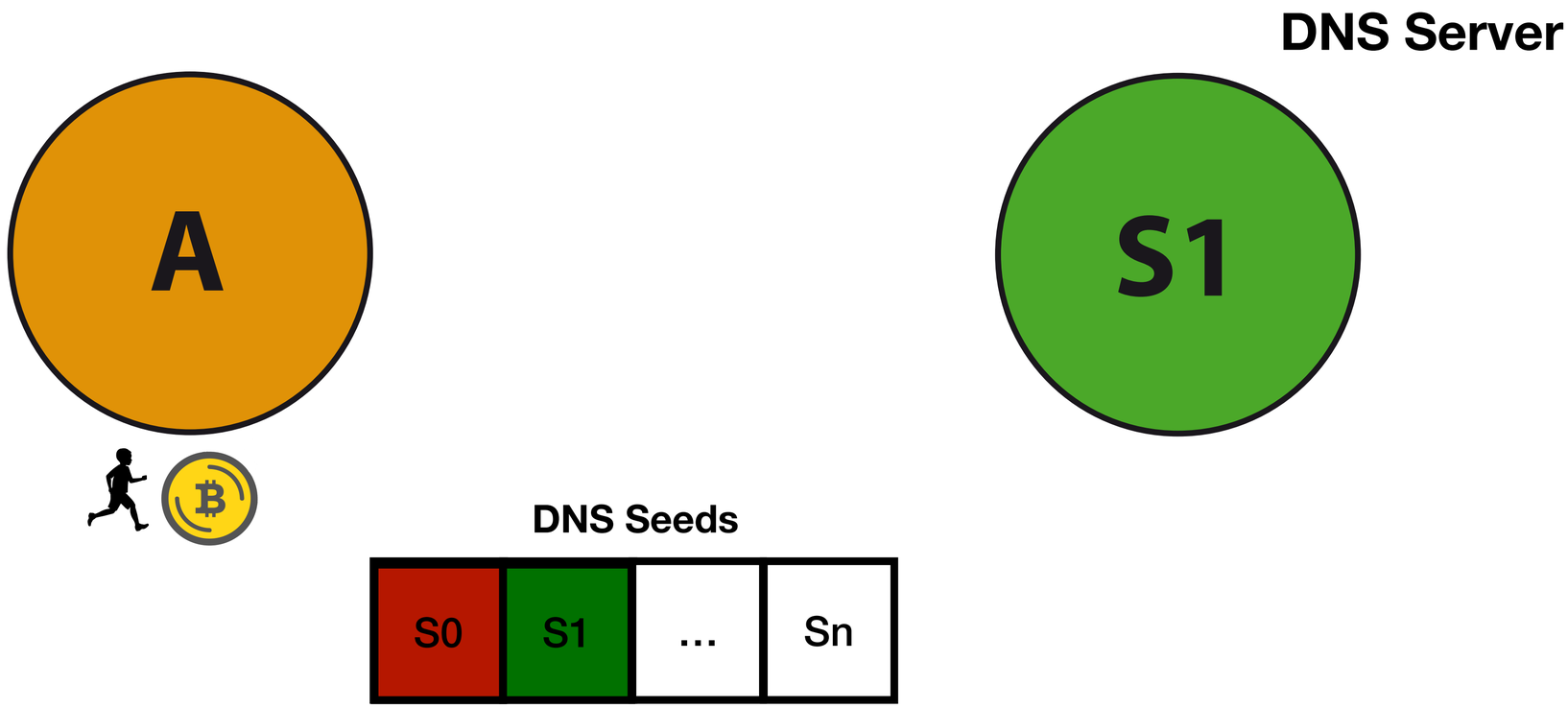
Bitcoin P2P bootstrapping (peer discovery)



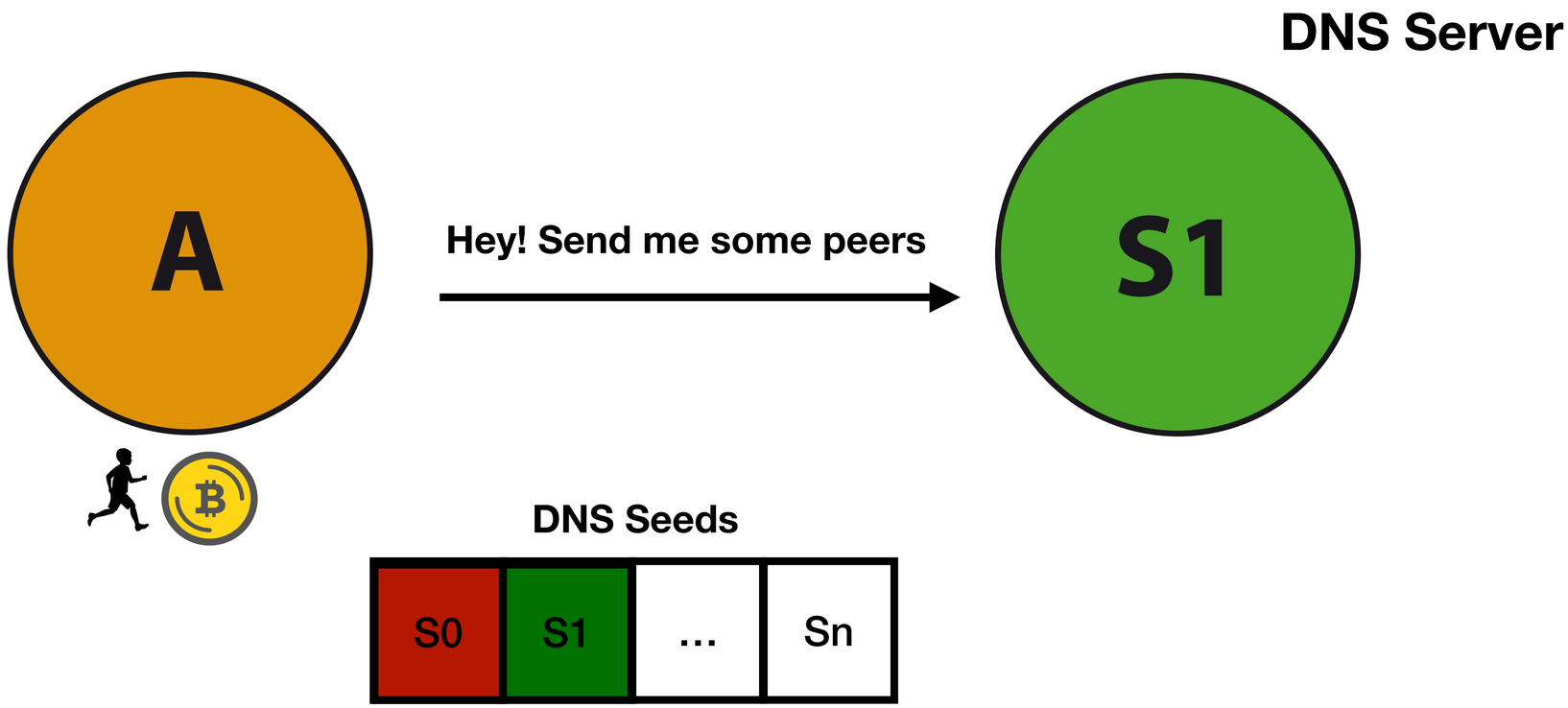
Bitcoin P2P bootstrapping (peer discovery)



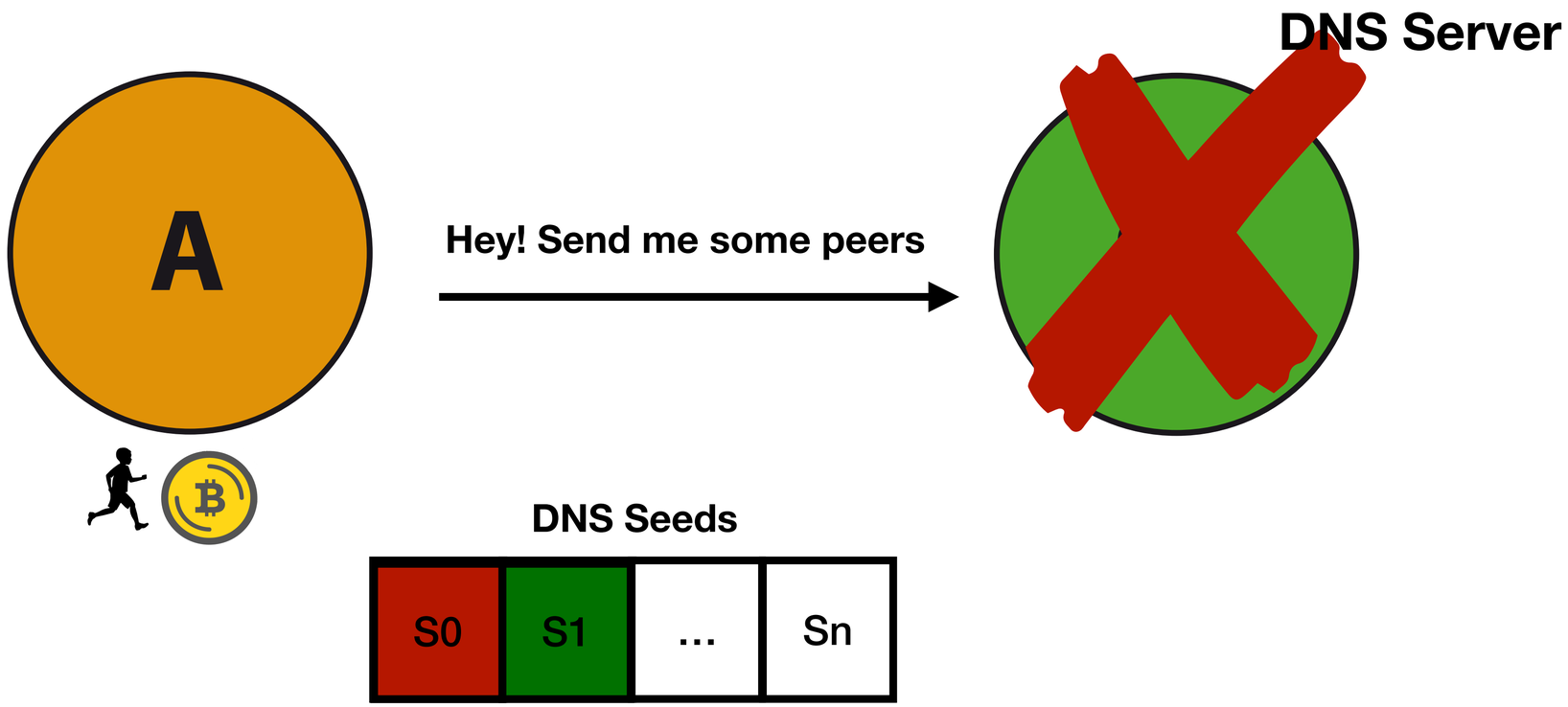
Bitcoin P2P bootstrapping (peer discovery)



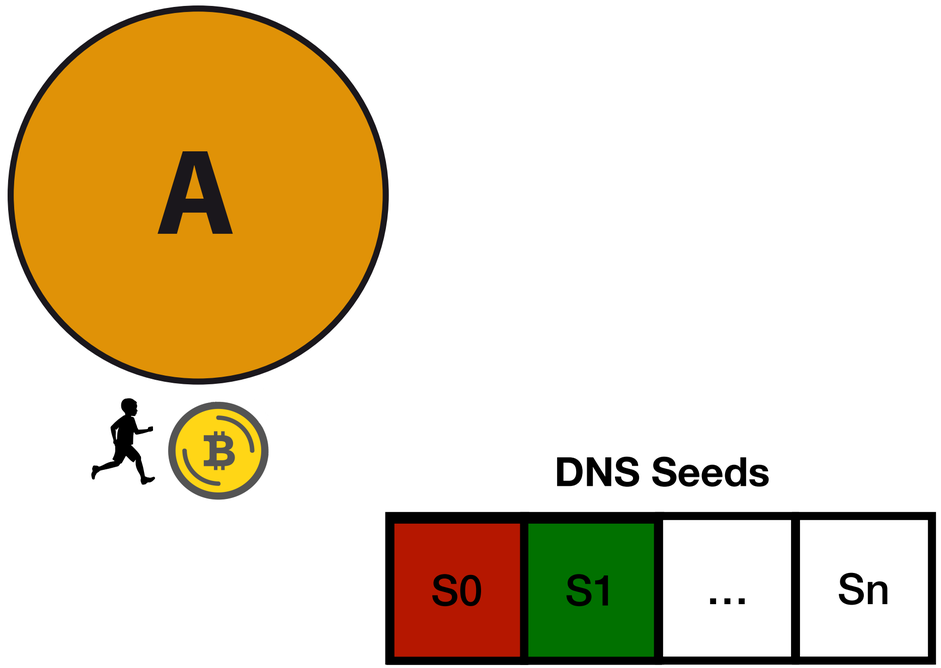
Bitcoin P2P bootstrapping (peer discovery)



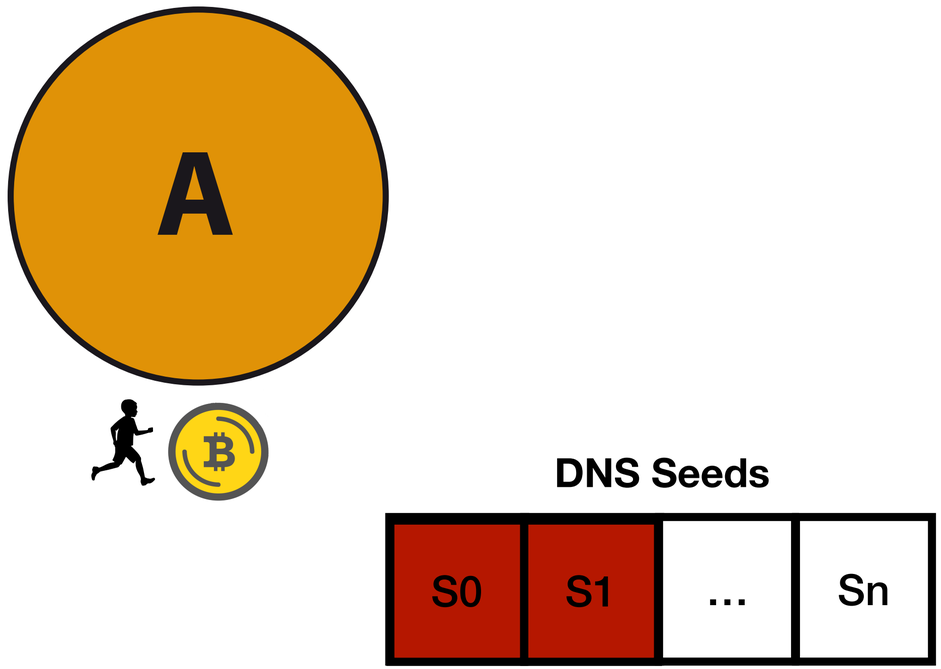
Bitcoin P2P bootstrapping (peer discovery)



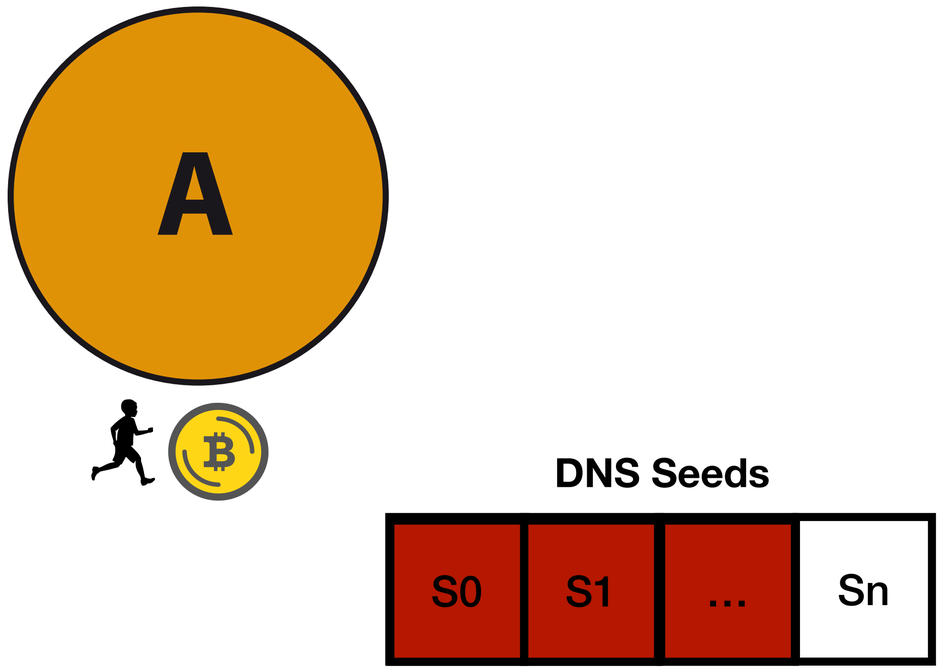
Bitcoin P2P bootstrapping (peer discovery)



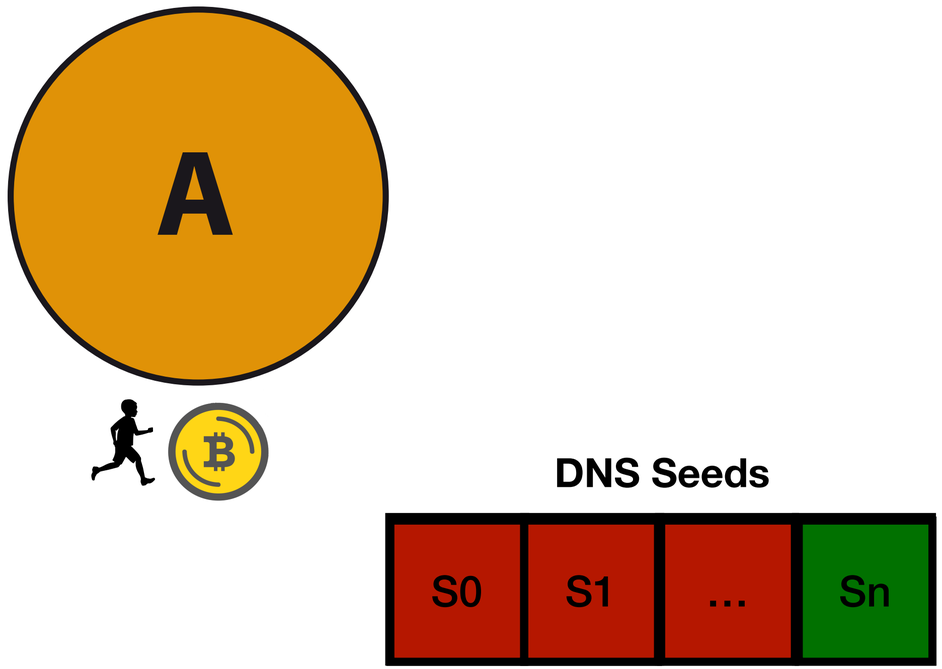
Bitcoin P2P bootstrapping (peer discovery)



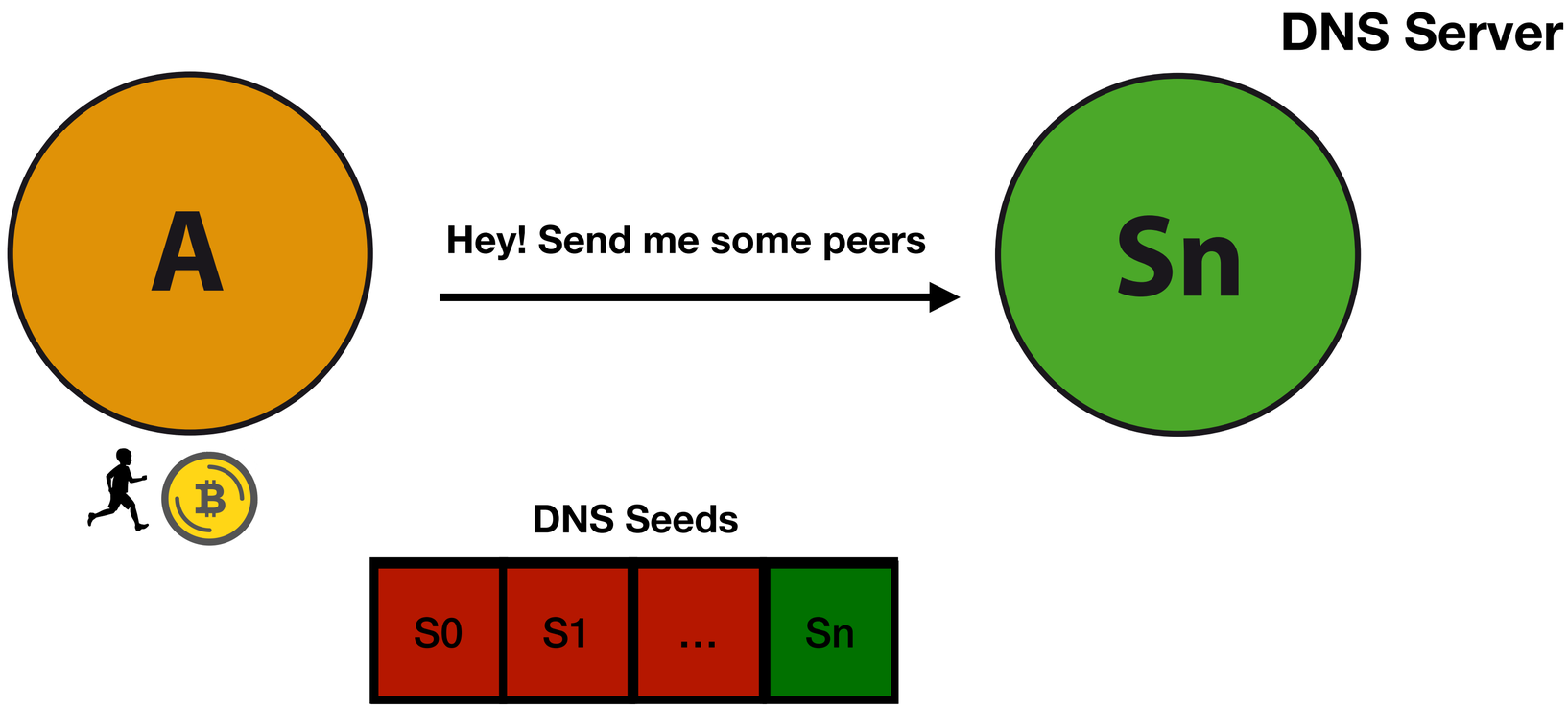
Bitcoin P2P bootstrapping (peer discovery)



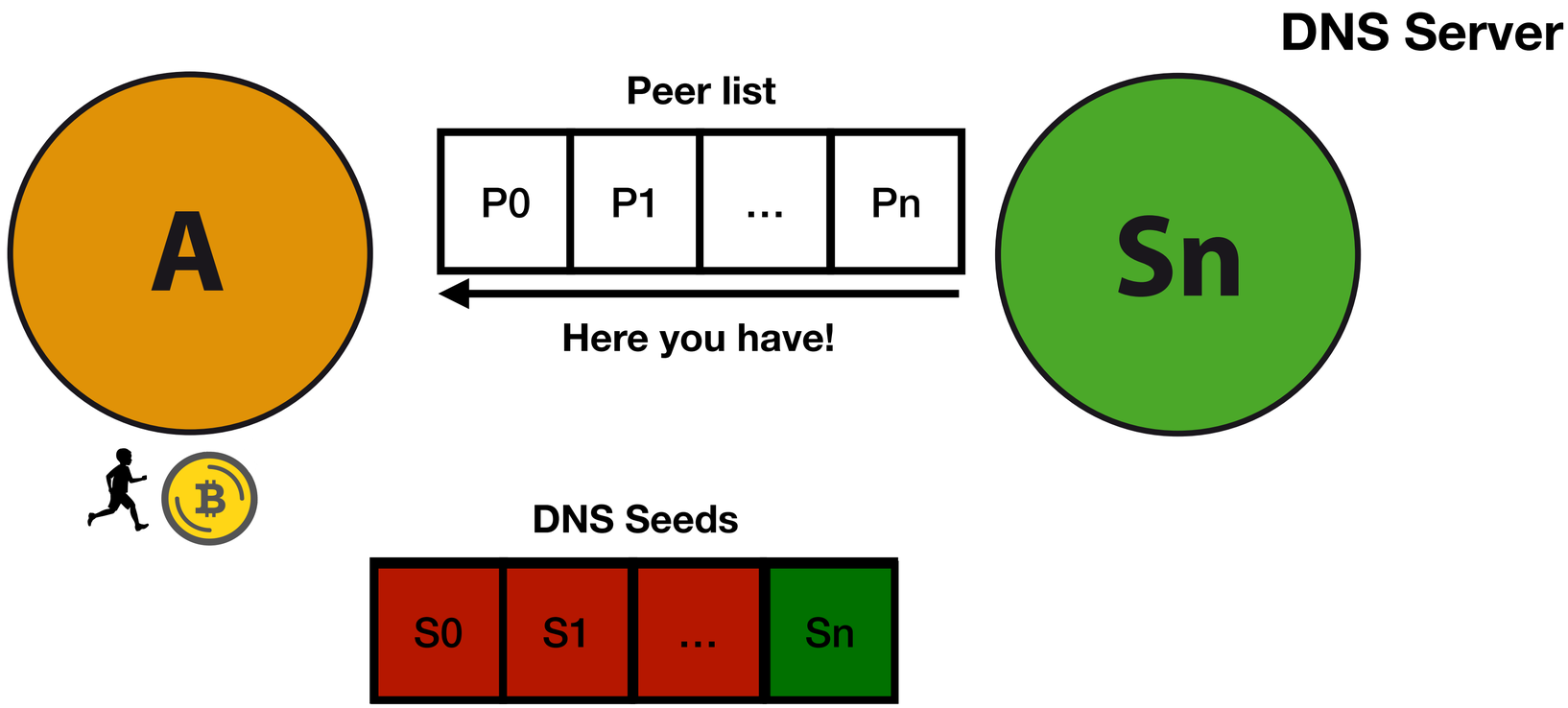
Bitcoin P2P bootstrapping (peer discovery)



Bitcoin P2P bootstrapping (peer discovery)



Bitcoin P2P bootstrapping (peer discovery)

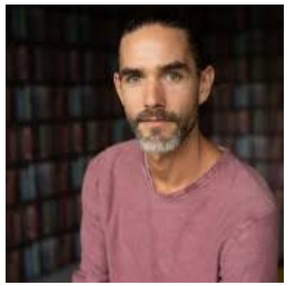


Bitcoin P2P bootstrapping (DNS server hosts)

```
vSeeds.emplace_back("seed.bitcoin.sipa.be"); // Pieter Wuille
vSeeds.emplace_back("dnsseed.bluematt.me"); // Matt Corallo
vSeeds.emplace_back("dnsseed.bitcoin.dashjr.org"); // Luke Dashjr
vSeeds.emplace_back("seed.bitcoinstats.com"); // Christian Decker
vSeeds.emplace_back("seed.bitcoin.jonasschnelli.ch"); // Jonas Schnelli
vSeeds.emplace_back("seed.btc.petertodd.org"); // Peter Todd
vSeeds.emplace_back("seed.bitcoin.sprovoost.nl"); // Sjors Provoost
```





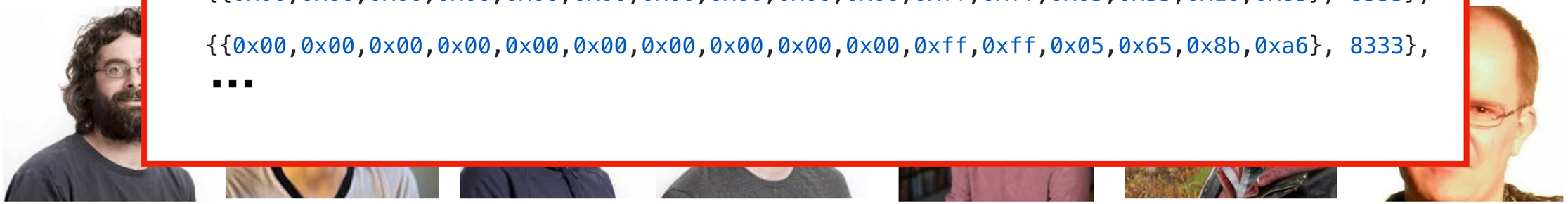


Bitcoin P2P bootstrapping (DNS server hosts)

If DNS seeds do not work, a node will try to connect to a hardcoded list of nodes (fixed seed)

```
vSeeds.emplace_back("seed.bitcoin.sipa.be"); // Pieter Wuille
vSeed
vSeed
vSeed
vSeed
vSeed
vSeed
```

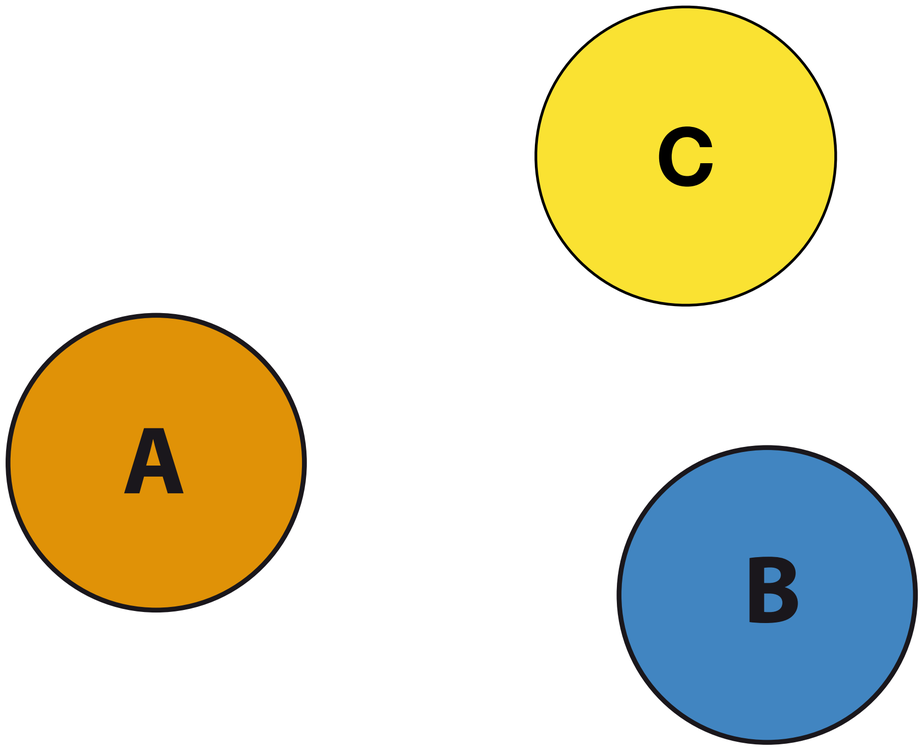
```
static SeedSpec6 pnSeed6_main[] = {
    {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x02,0x84,0x64,0x2f}, 8333},
    {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x01,0x61,0x04}, 8333},
    {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x27,0xae,0x74}, 8333},
    {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x2d,0x4f,0x0e}, 8333},
    {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x35,0x10,0x85}, 8333},
    {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x65,0x8b,0xa6}, 8333},
    ...
```



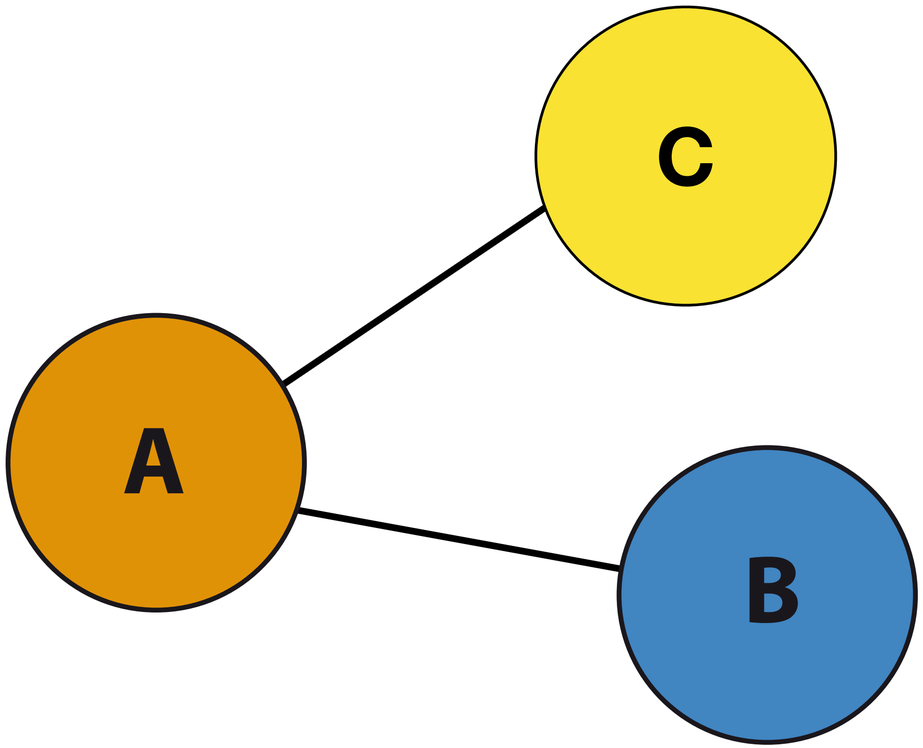
Bitcoin P2P bootstrapping (recap)

- A node bootstraps with no known peers
- First it tries to query a list of well known DNS seeds
- As **last resource** it uses a hardcoded seed

Populating the peers database (addrman)

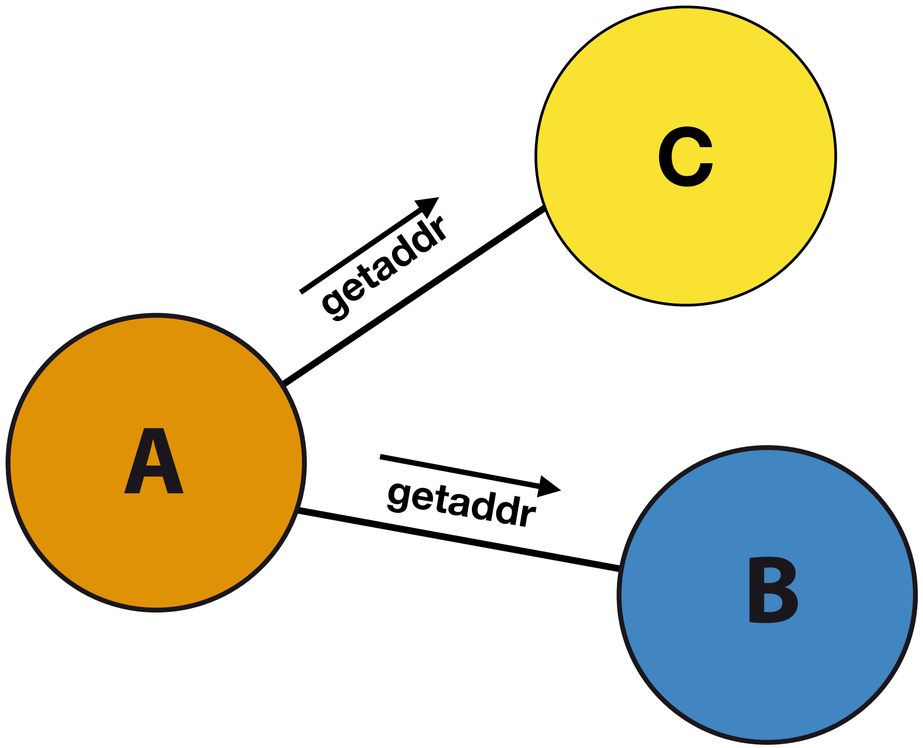


Populating the peers database (addrman)



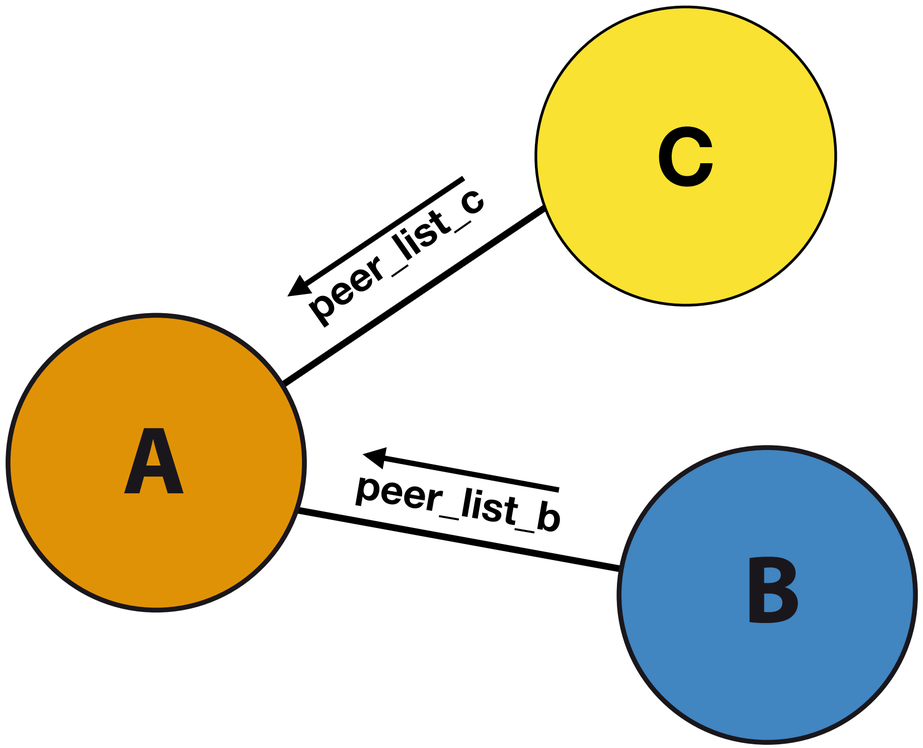
- A connects a subset of peers from the ones learned from the DNS seeds

Populating the peers database (addrman)



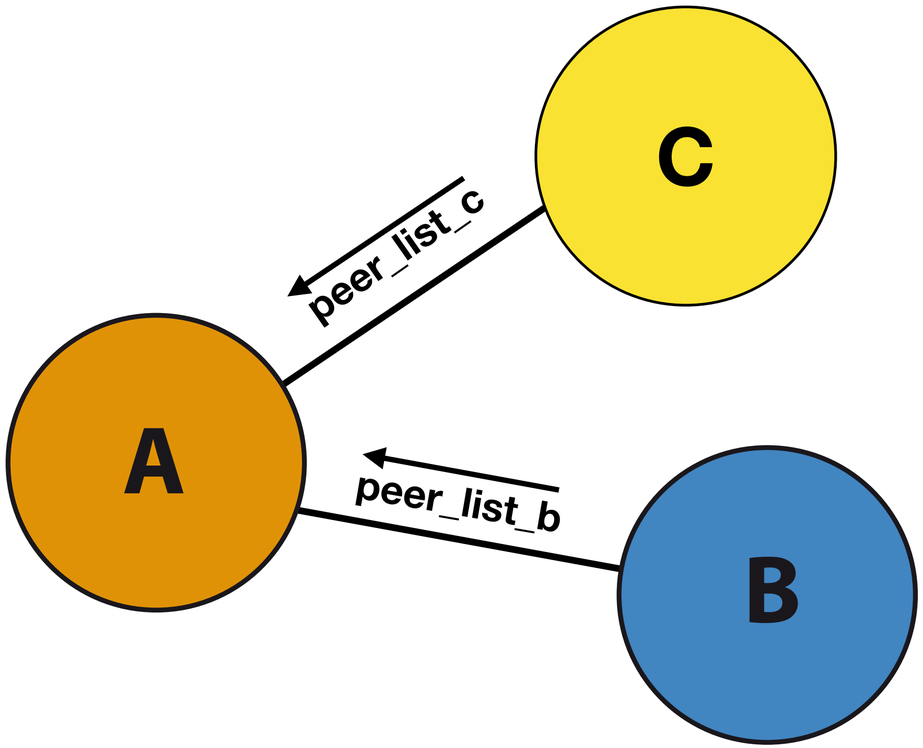
- A connects a subset of peers from the ones learned from the DNS seeds
- A requests more peers to his neighbors (**getaddr**)

Populating the peers database (addrman)



- A connects a subset of peers from the ones learned from the DNS seeds
- A requests more peers to his neighbors (**getaddr**)
- Peers reply with some addresses they known about (**addr, up to 1000 addresses**)

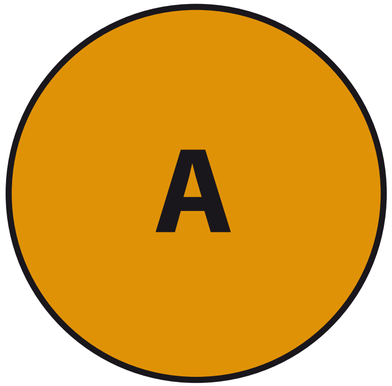
Populating the peers database (addrman)



A's addrman = A's addrman U peer_list_c U peer_list_b

- A connects a subset of peers from the ones learned from the DNS seeds
- A requests more peers to his neighbors (**getaddr**)
- Peers reply with some addresses they known about (**addr, up to 1000 addresses**)
- A adds the new addresses to its peers database (or update the existing ones)

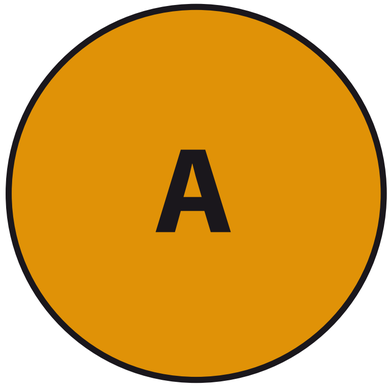
Incoming / outgoing connections



Peer database (addrman)

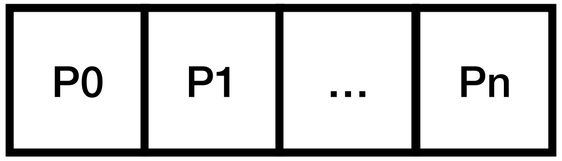
P0	P1	…	Pn

Incoming / outgoing connections

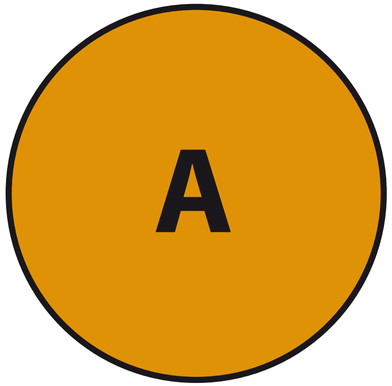


- During bootstrap, a node will start some **outgoing connections** with peers it has learn about (**8 by default**) and try to maintain them

Peer database (addrman)



Incoming / outgoing connections



Peer database (addrman)

P0	P1	…	Pn

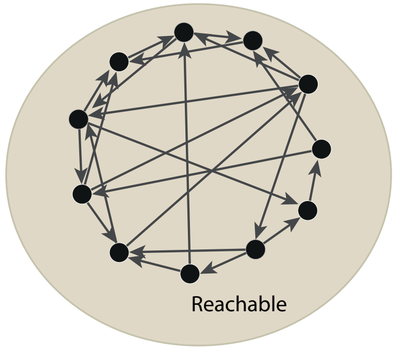
- During bootstrap, a node will start some **outgoing connections** with peers it has learn about (**8 by default**) and try to maintain them

- A node will also accept **some incoming** connections (**117 by default**)

Cryptocurrency networks taxonomy

Cryptocurrency networks taxonomy

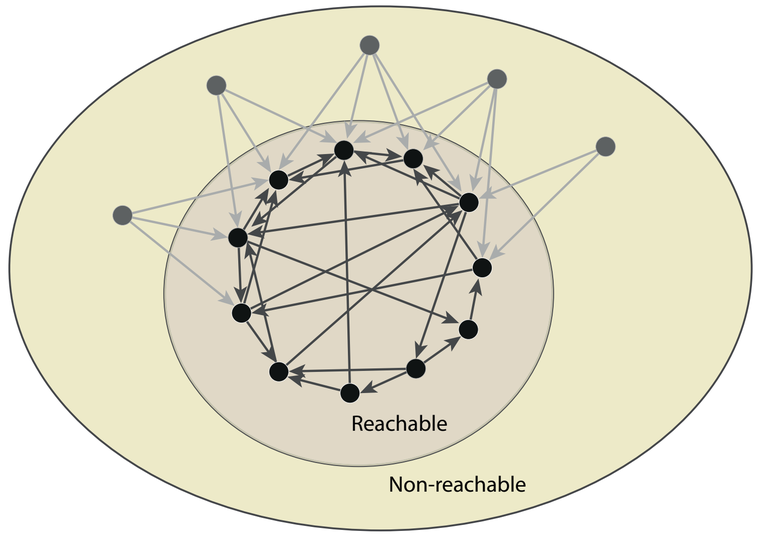
- Reachable network: all nodes accept incoming / outgoing connections



Cryptocurrency networks taxonomy

- Reachable network: all nodes accept incoming / outgoing connections

- Non-reachable: nodes do not accept incoming connections / cannot be reached (NAT/firewalls/…)

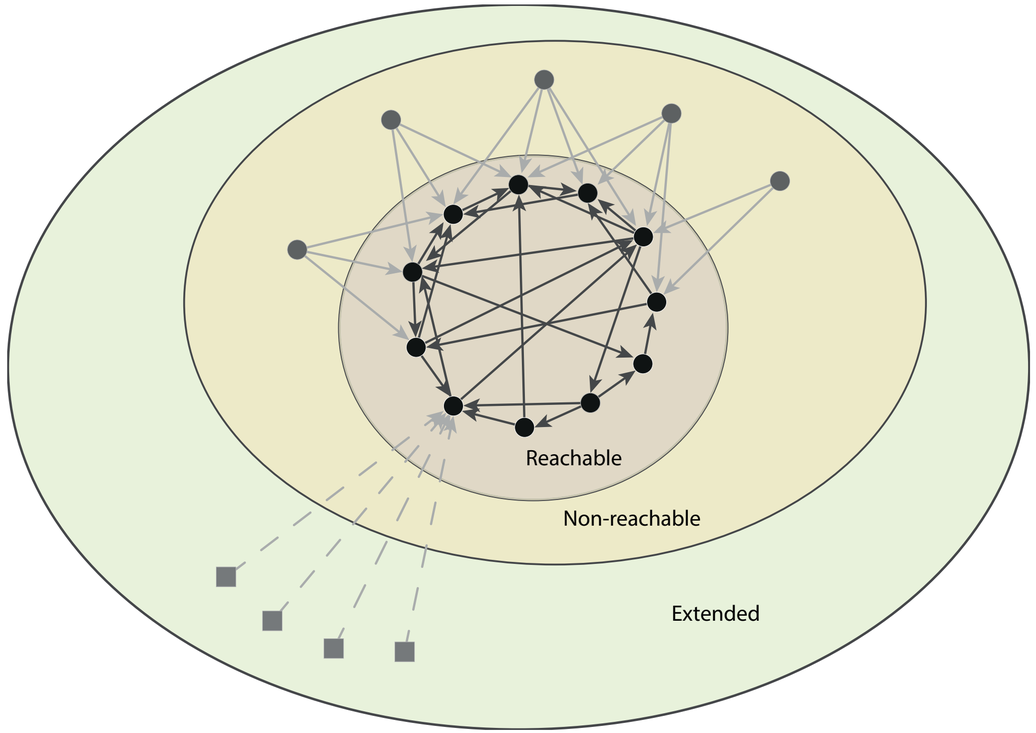


Cryptocurrency networks taxonomy

- Reachable network: all nodes accept incoming / outgoing connections

- Non-reachable: nodes do not accept incoming connections / cannot be reached (NAT/firewalls/…)

- Extended network: nodes use different protocol to communicate (not always P2P)

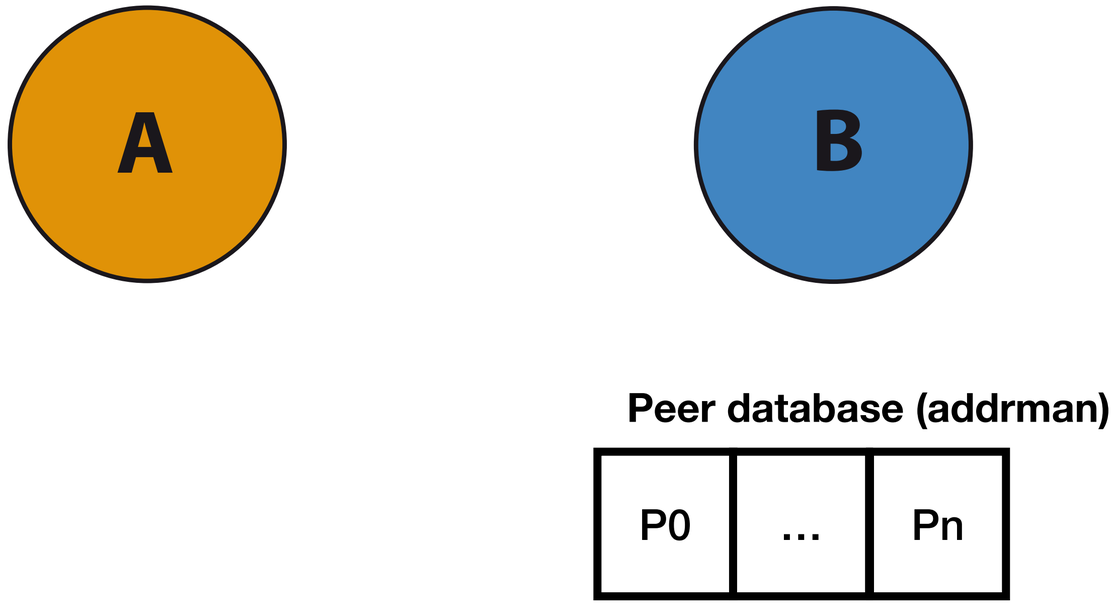


Hello world!

- How does a node announce his presence to the rest of the network?

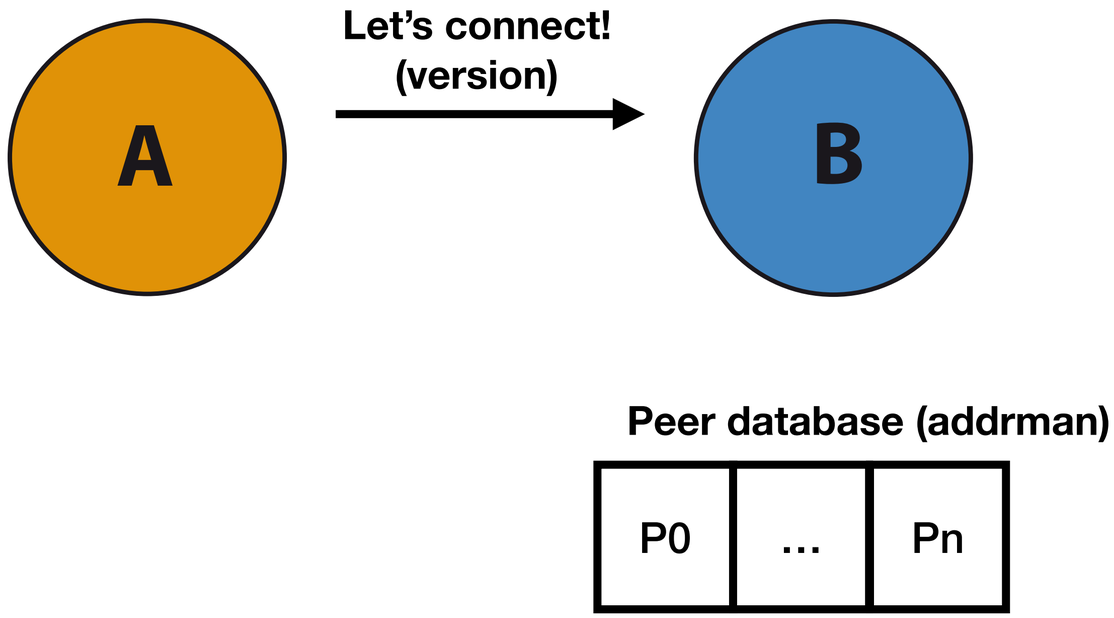
Hello world!

- How does a node announce his presence to the rest of the network?



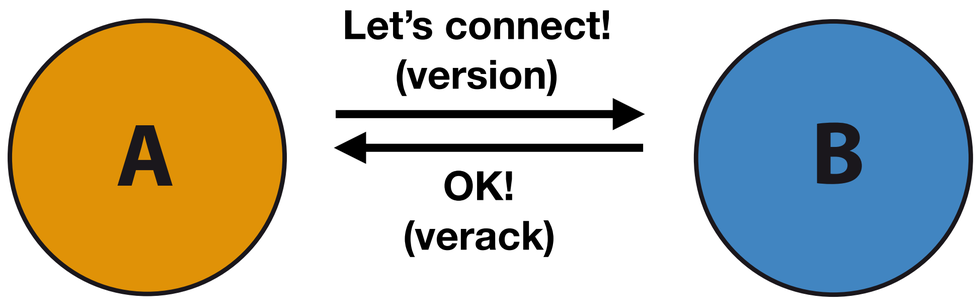
Hello world!

- How does a node announce his presence to the rest of the network?



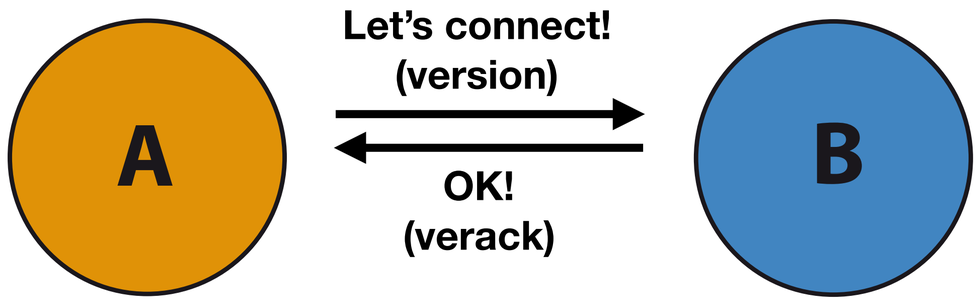
Hello world!

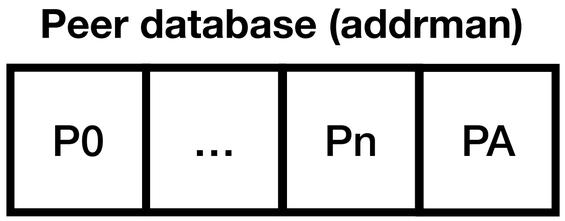
- How does a node announce his presence to the rest of the network?



Hello world!

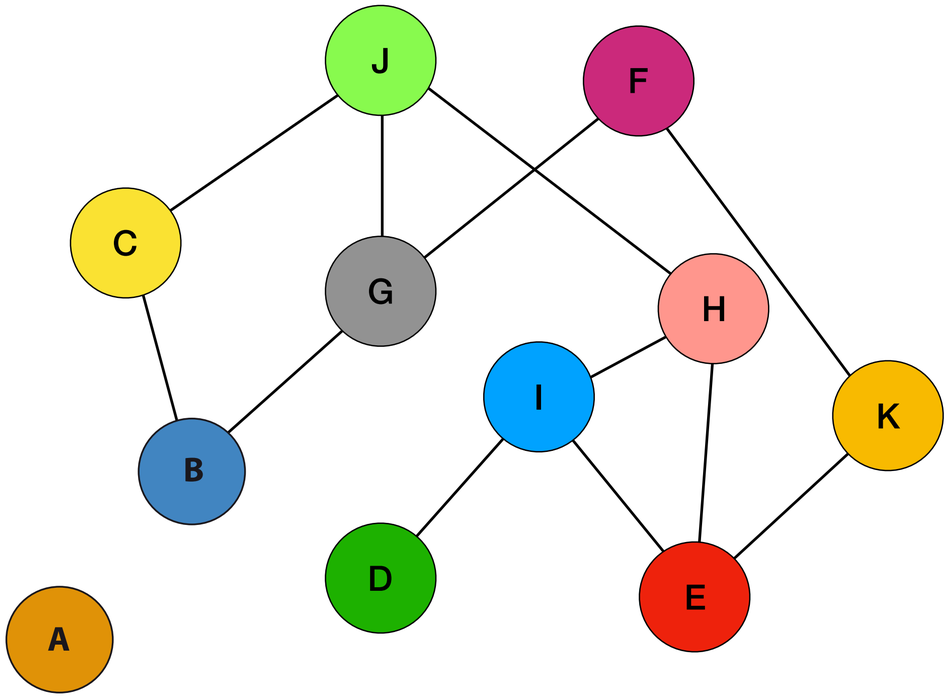
- How does a node announce his presence to the rest of the network?





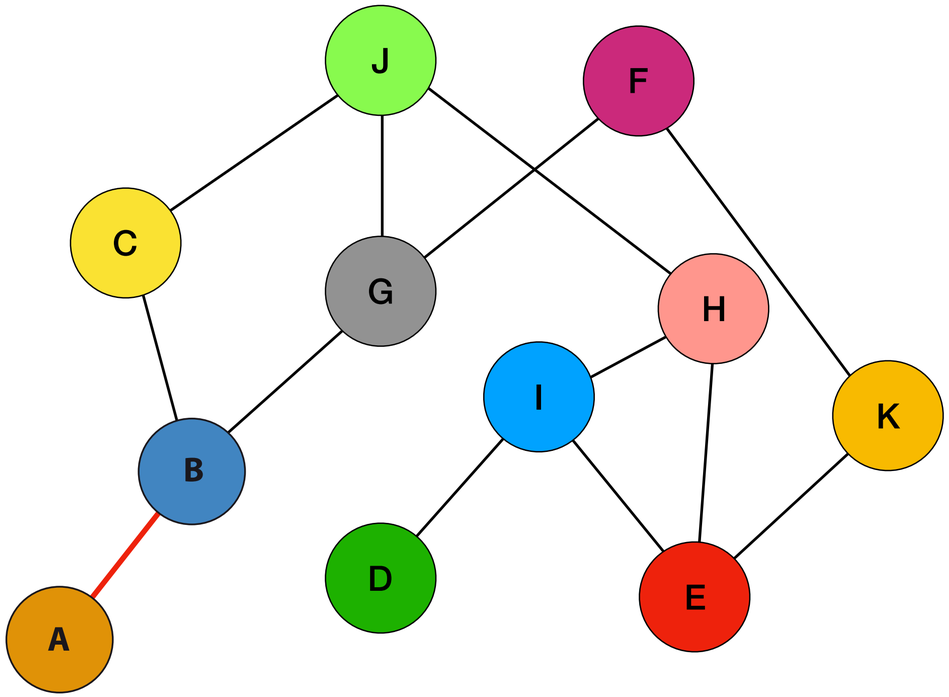
Hello world!

- How does a node announce his presence to the rest of the network?



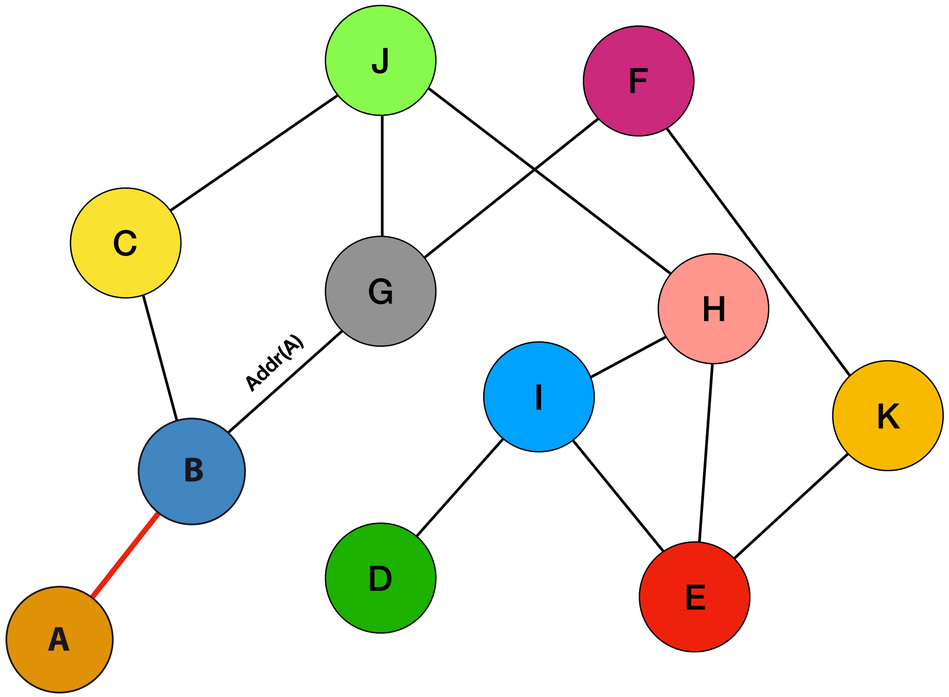
Hello world!

- How does a node announce his presence to the rest of the network?



Hello world!

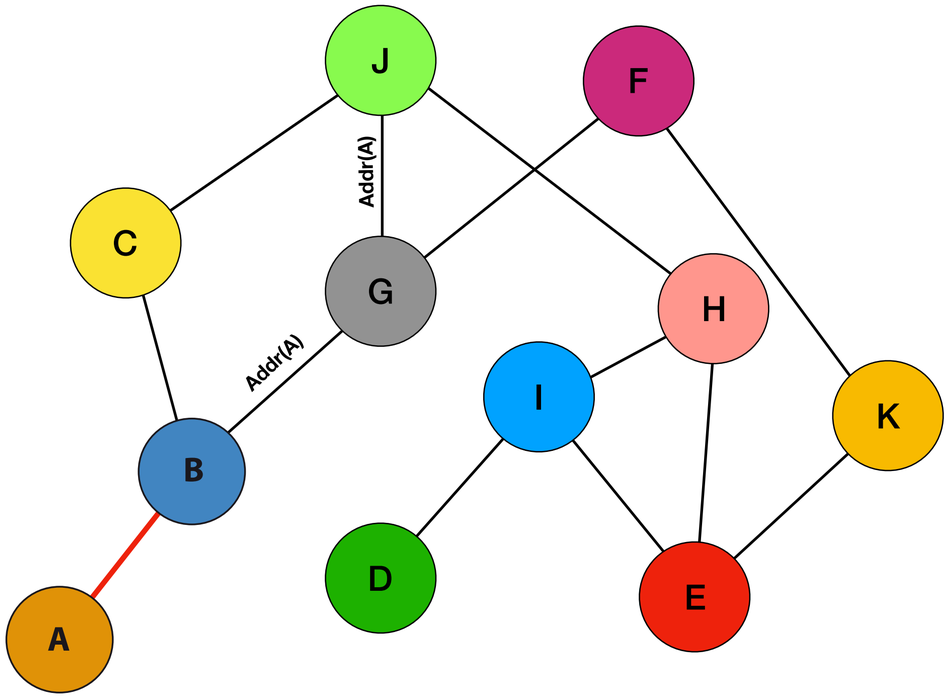
- How does a node announce his presence to the rest of the network?



- B picks a random subset of its neighbors and relays A’s address

Hello world!

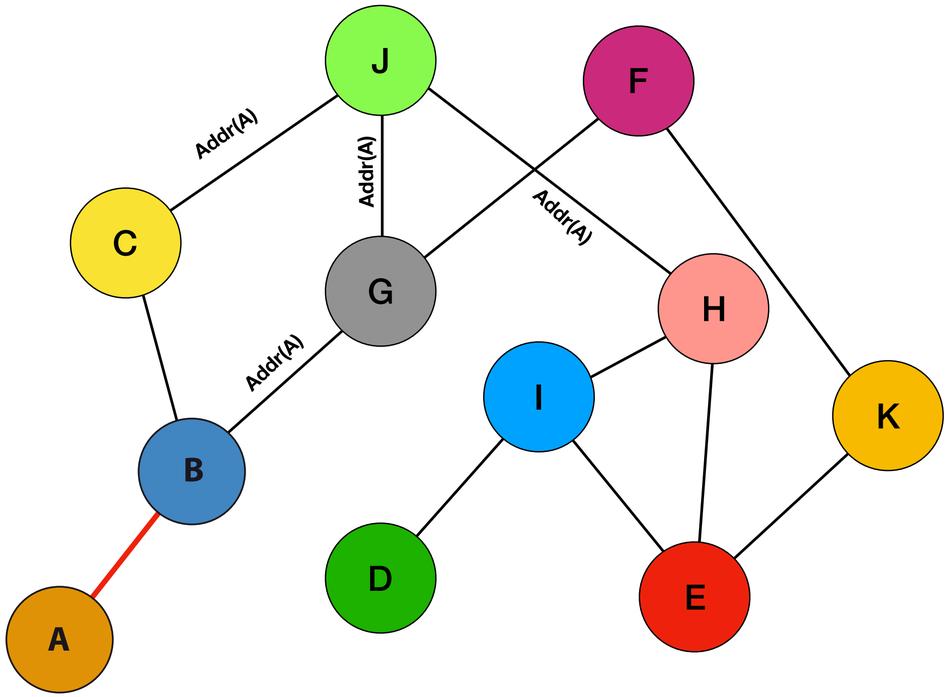
- How does a node announce his presence to the rest of the network?



- B picks a random subset of its neighbors and relays A’s address
- The nodes picked by B pick a random subset of their neighbors and relay A’s address

Hello world!

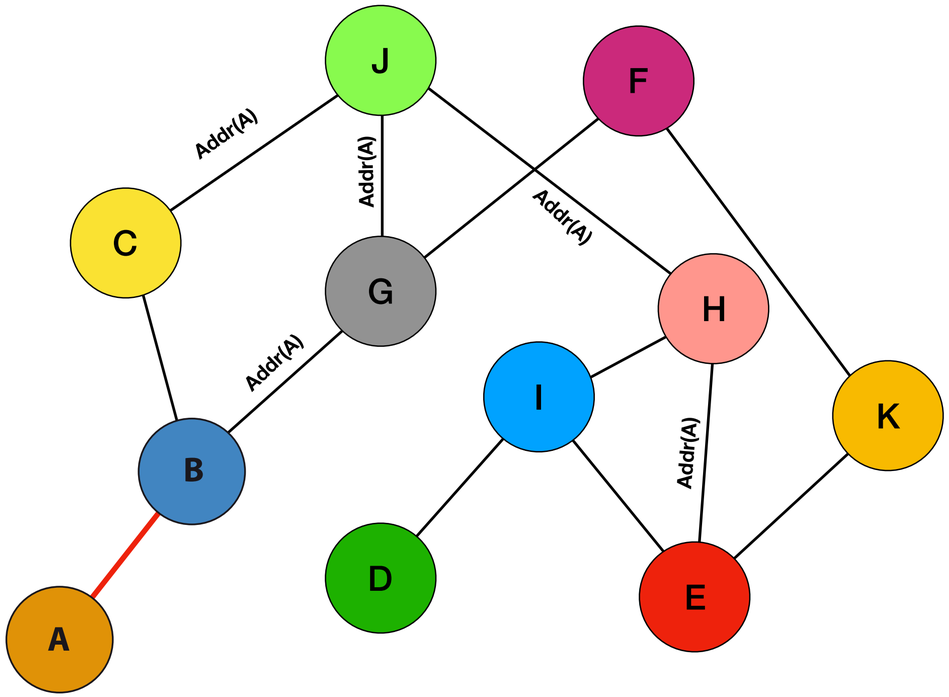
- How does a node announce his presence to the rest of the network?



- B picks a random subset of its neighbors and relays A’s address
- The nodes picked by B pick a random subset of their neighbors and relay A’s address
- And so on and so forth…

Hello world!

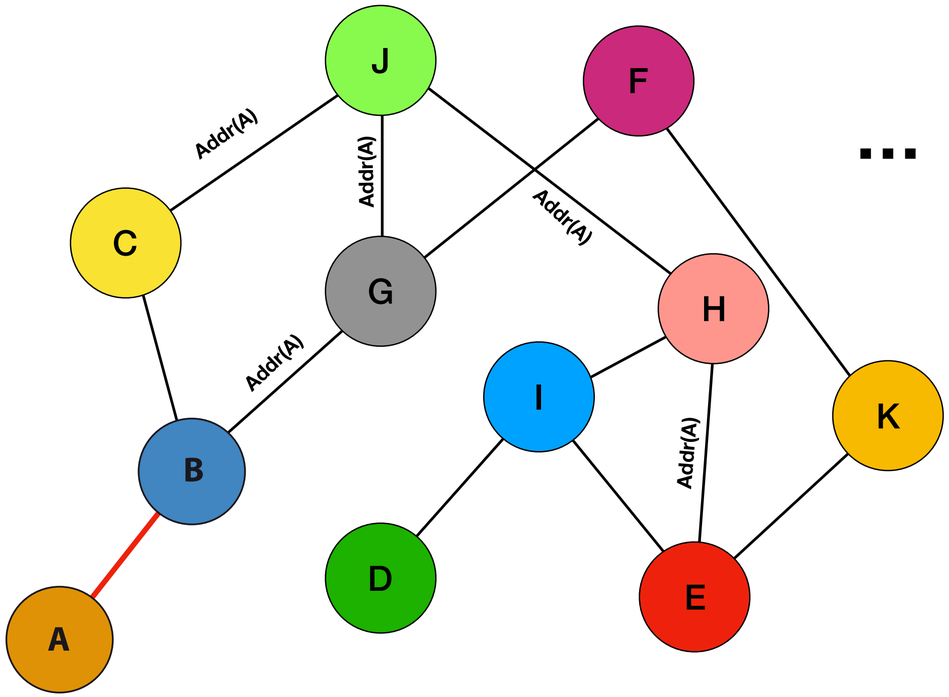
- How does a node announce his presence to the rest of the network?



- B picks a random subset of its neighbors and relays A’s address
- The nodes picked by B pick a random subset of their neighbors and relay A’s address
- And so on and so forth…

Hello world!

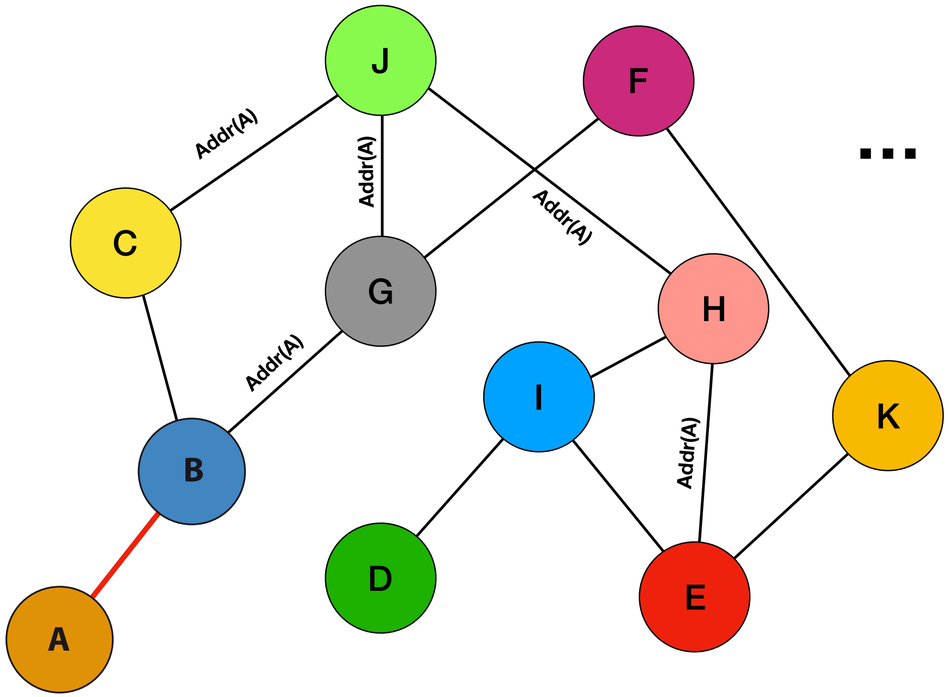
- How does a node announce his presence to the rest of the network?



- B picks a random subset of its neighbors and relays A's address
- The nodes picked by B pick a random subset of their neighbors and relay A's address
- And so on and so forth…

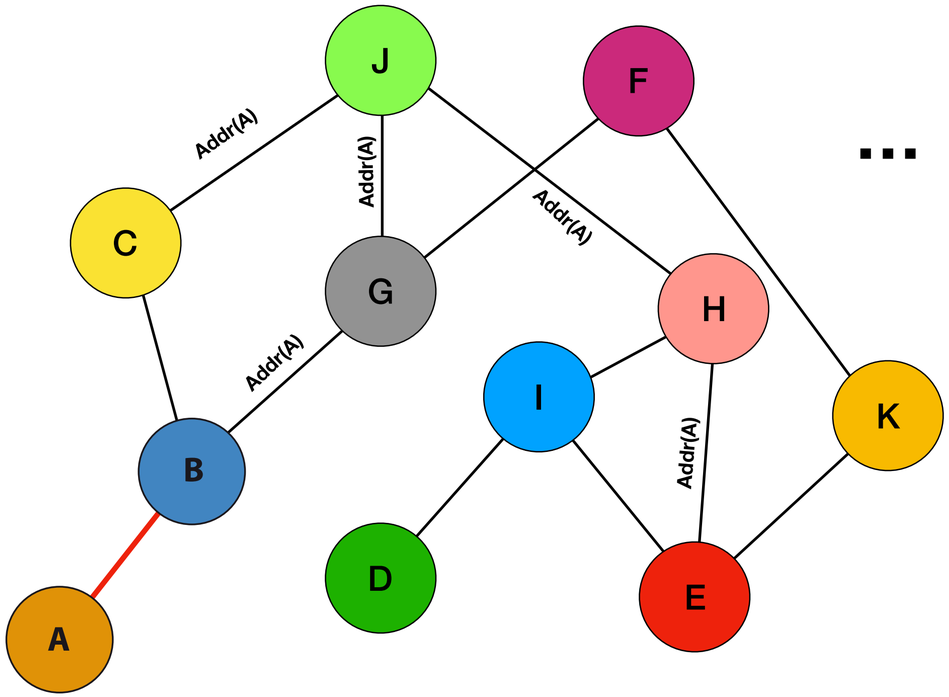
Hello world!

- How does a node announce his presence to the rest of the network?



Hello world!

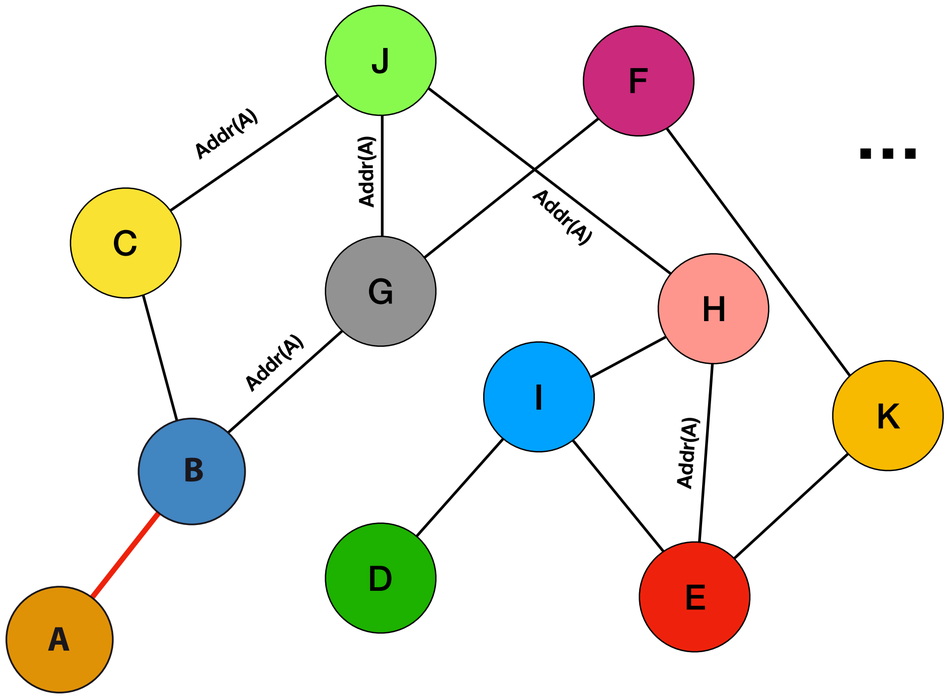
- How does a node announce his presence to the rest of the network?



- The address will eventually be spread throughout the network

Hello world!

- How does a node announce his presence to the rest of the network?



- The address will eventually be spread throughout the network
- Nodes learning about the new peer will add it to their peers database

Connections (recap)

- A node learns about the peers in the network by asking other peers (after an initial bootstrap)

- A node maintains a database of all the peers he has heard of and keeps populating it / updating it

- A node initiates (and maintain) some outgoing connects and also accept some incoming ones

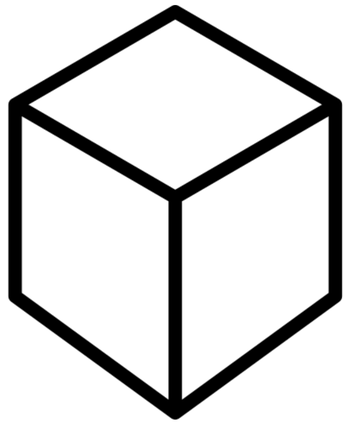
- The address of a new node is propagated thought the network so all peers can know about it

Actors and purpose
(what, who, why, and how)

The data (what?)

- There are two main items that peers share in a cryptocurrency P2P network: **transactions** and **blocks**

From: Ford	To: Arthur	42



The actors (who?) (1/2)

The actors (who?) (1/2)

- There are two main roles followed by nodes: **peers** and **miners**

The actors (who?) (1/2)

- There are two main roles followed by nodes: **peers** and **miners**

- (Normal) **Peers:**

The actors (who?) (1/2)

- There are two main roles followed by nodes: **peers** and **miners**

- (Normal) **Peers:**

  - Can **create transactions** that spend some of their bitcoins

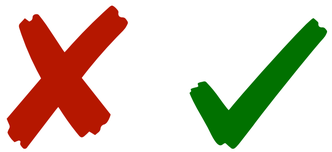
The actors (who?) (1/2)

- There are two main roles followed by nodes: **peers** and **miners**

- (Normal) **Peers:**

  - Can **create transactions** that spend some of their bitcoins

  - Do **verify** the correctness of received **transactions** and **blocks** (from other peers)



The actors (who?) (1/2)

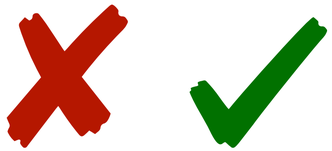
- There are two main roles followed by nodes: **peers** and **miners**

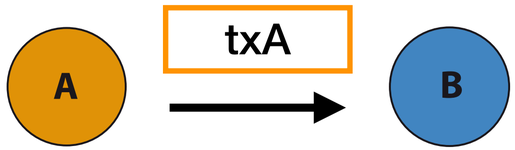
- (Normal) **Peers:**

  - Can **create transactions** that spend some of their bitcoins

  - Do **verify** the correctness of received **transactions** and **blocks** (from other peers)

  - Do **relay** valid **transactions** and **blocks** (created by them or obtained from other peers)





The actors (who?) (2/2)

The actors (who?) (2/2)

- **Miners:**

The actors (who?) (2/2)

- **Miners:**
  - Can do all what a **peer** could do*

The actors (who?) (2/2)

- **Miners:**
  - Can do all what a **peer** could do*
  - Can **generate blocks** through a process known as mining

The actors (who?) (2/2)

- **Miners:**
  - Can do all what a **peer** could do*
  - Can **generate blocks** through a process known as mining

* There are specific purpose miners (ASICS) that only perform mining

The purpose (why?)

- Peers relay transactions in order to reach miners, which will include such transactions in future blocks

- Miners generate blocks to obtain its reward (and also the transactions fees)

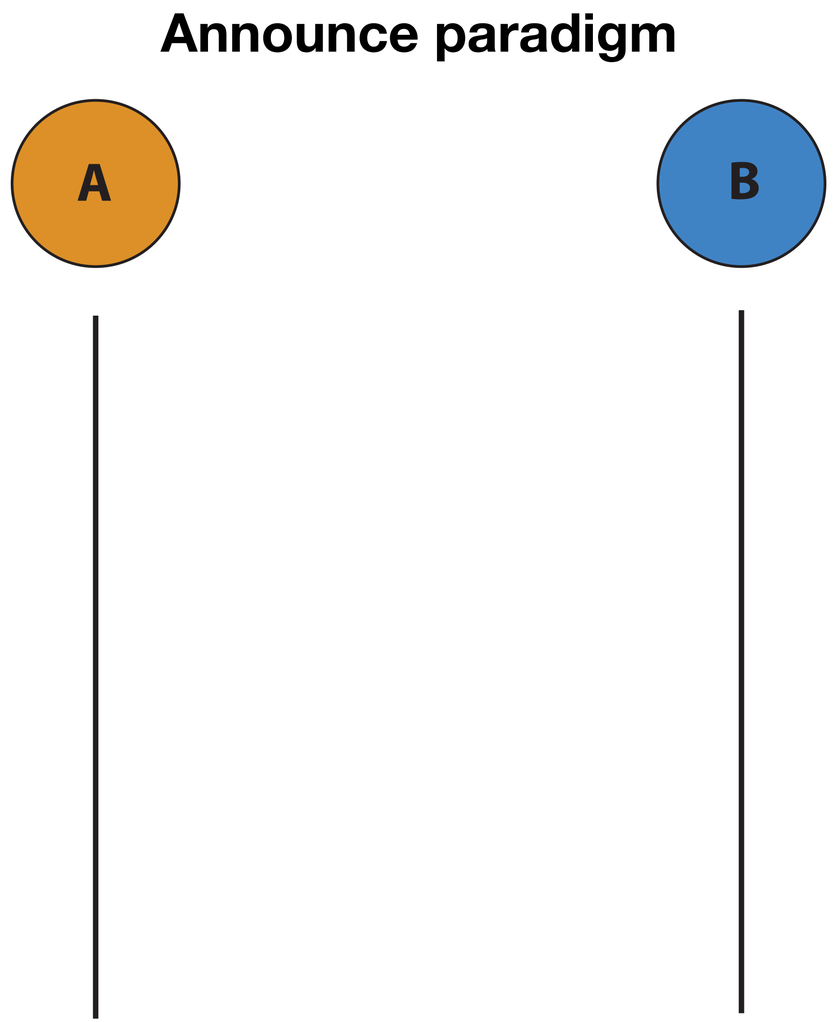
- Blocks are relayed to ultimately achieve a consistent view of the blockchain

- Peers validate transactions and blocks (and relay only the valid ones) in order to avoid cheating (e.g: double-spending, coin forgery, etc)

The gossip protocol (how?)

The gossip protocol (how?)

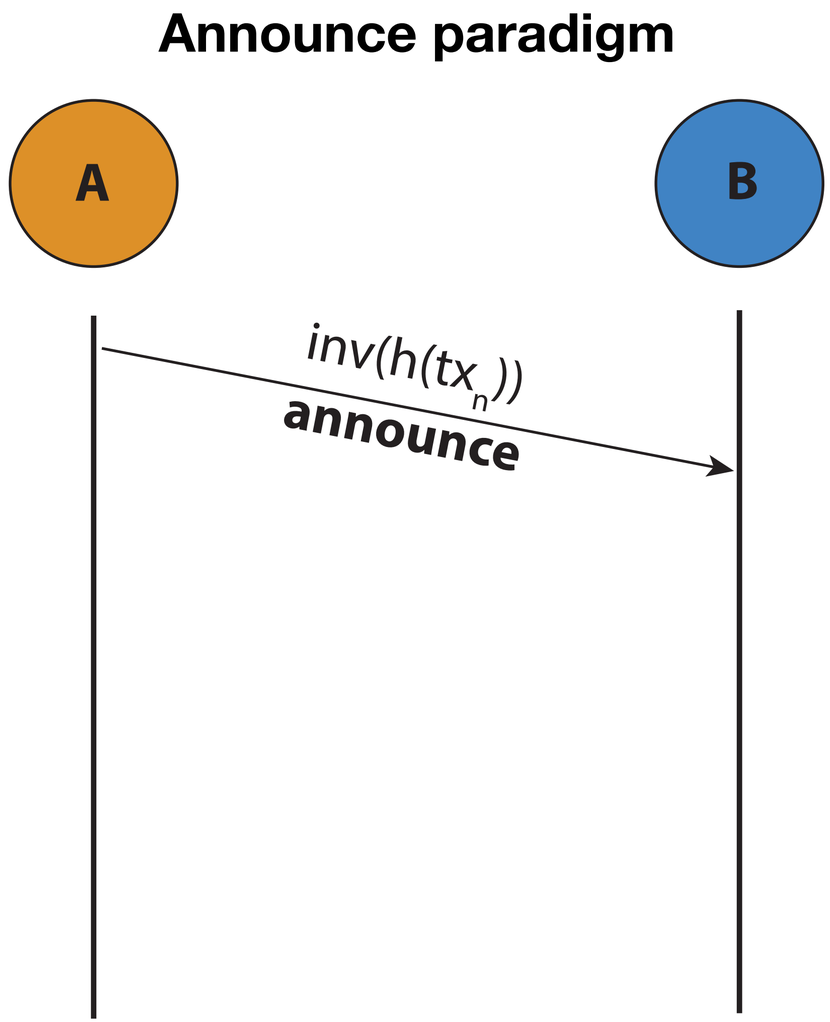
- Items (transactions and blocks) are shared between peers in a push manner



The gossip protocol (how?)

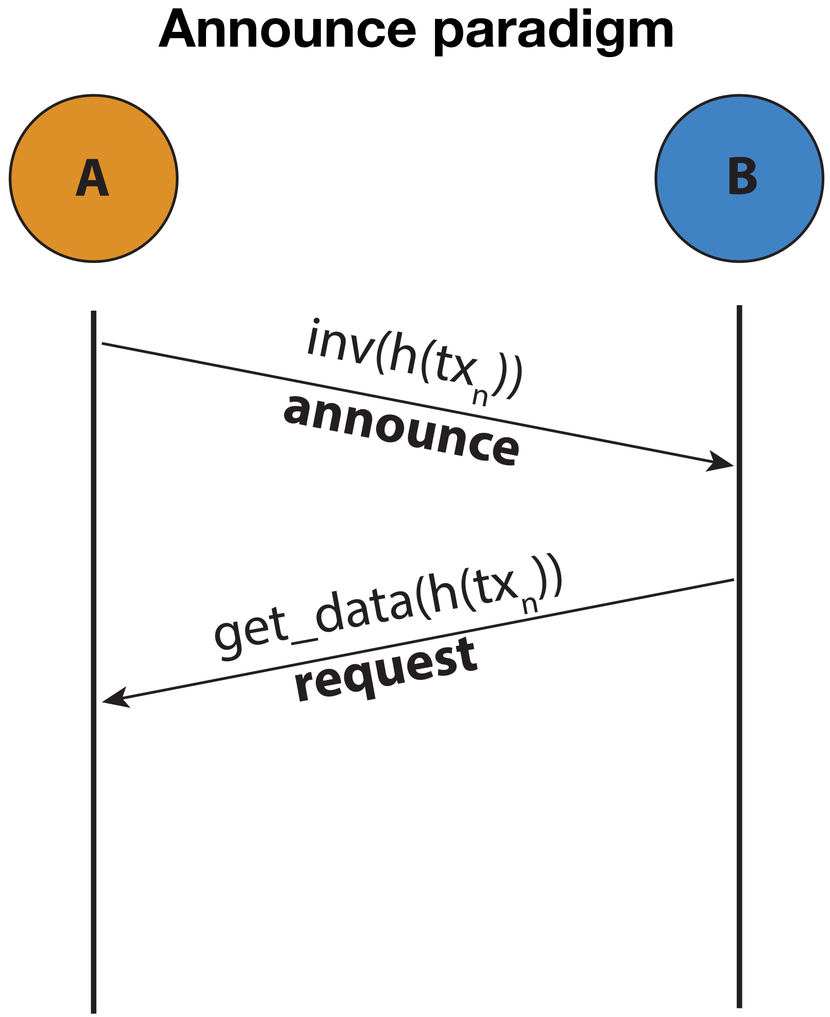
- Items (transactions and blocks) are shared between peers in a push manner

- When a peer receives / generates a new item he announce it to his neighbors (**announce**)



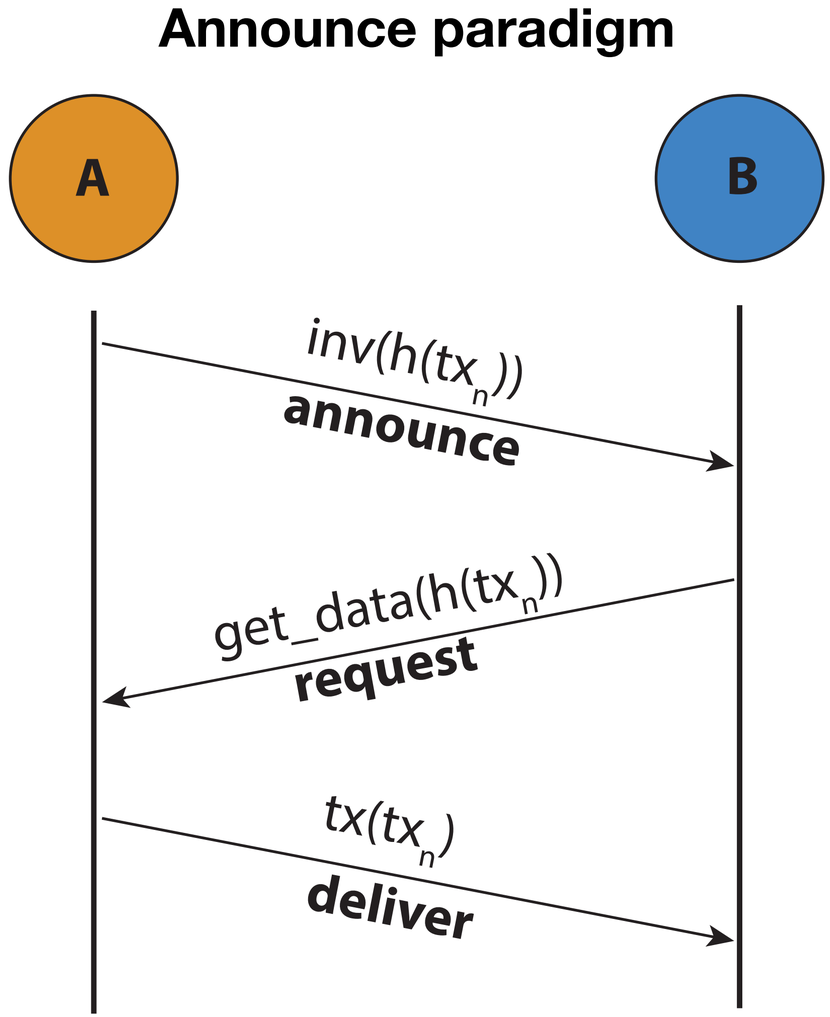
The gossip protocol (how?)

- Items (transactions and blocks) are shared between peers in a push manner
- When a peer receives / generates a new item he announce it to his neighbors (**announce**)
- Upon receiving an announce of an item, a node that does not know about it will request such item back to the announcer (**request**)



The gossip protocol (how?)

- Items (transactions and blocks) are shared between peers in a push manner
- When a peer receives / generates a new item he announce it to his neighbors (**announce**)
- Upon receiving an announce of an item, a node that does not know about it will request such item back to the announcer (**request**)
- Upon receiving a request of a known item, a node will reply back with it (**deliver**)

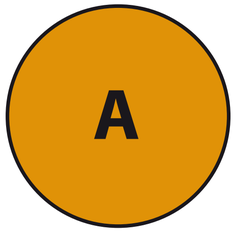



BREAK
TIME

Information propagation

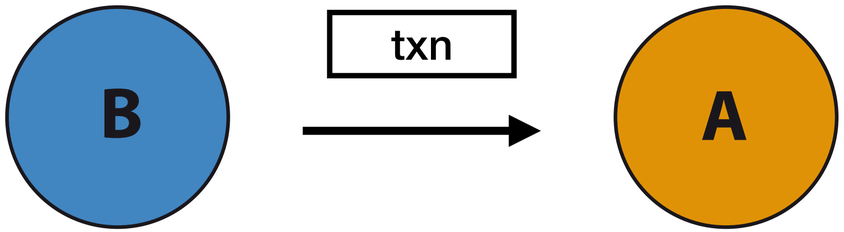
Information propagation (1/2)

Information propagation (1/2)

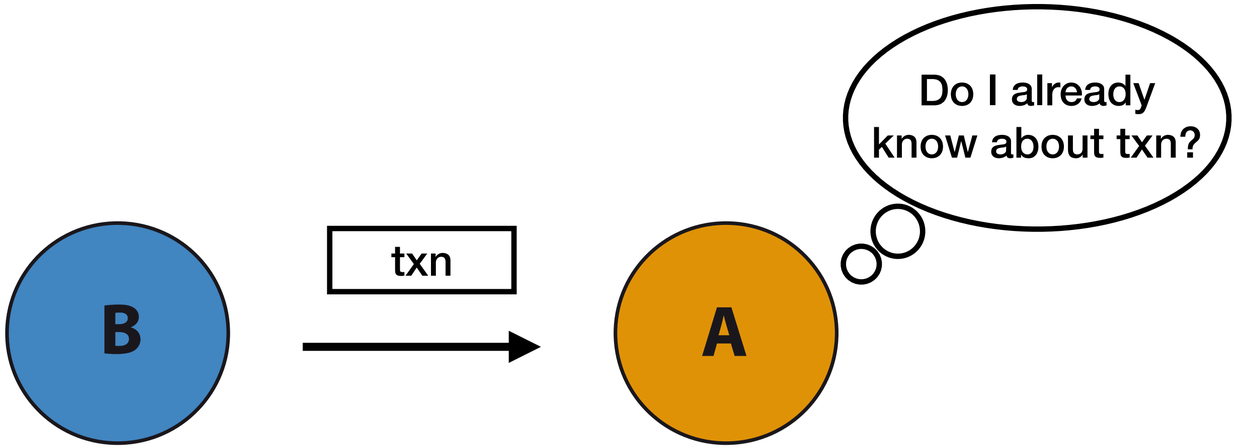


Information propagation (1/2)

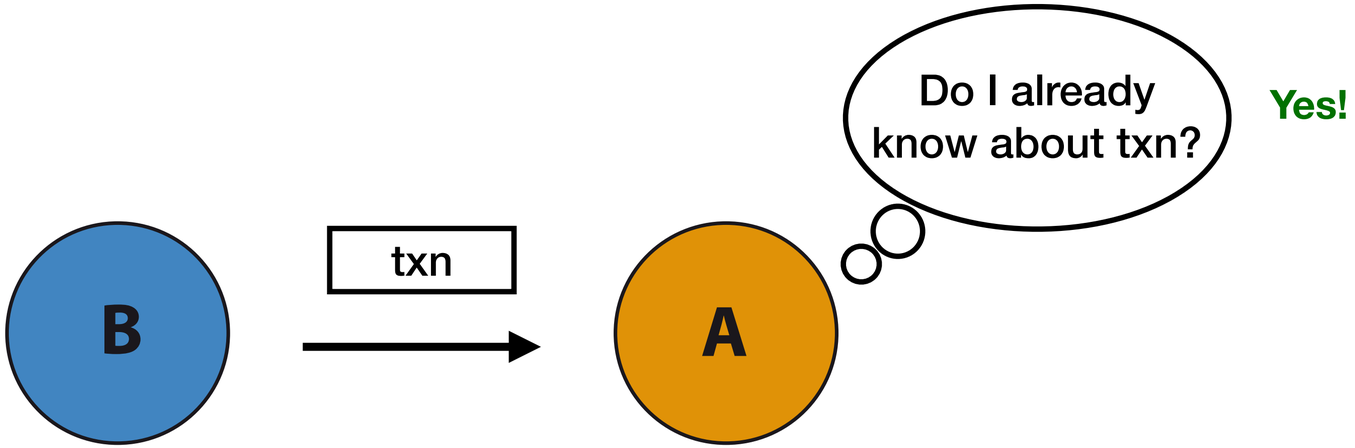
Information propagation (1/2)



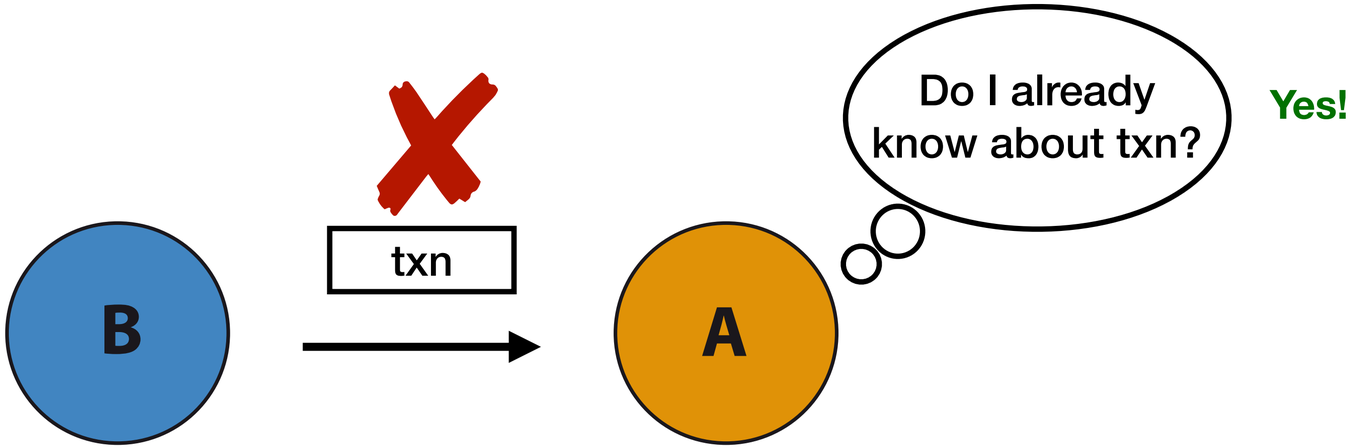
Information propagation (1/2)



Information propagation (1/2)

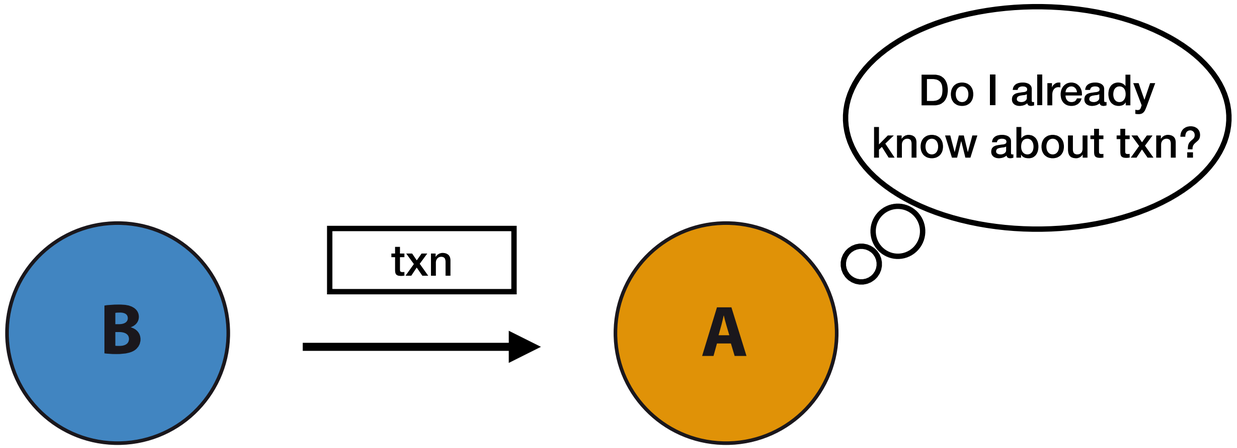


Information propagation (1/2)



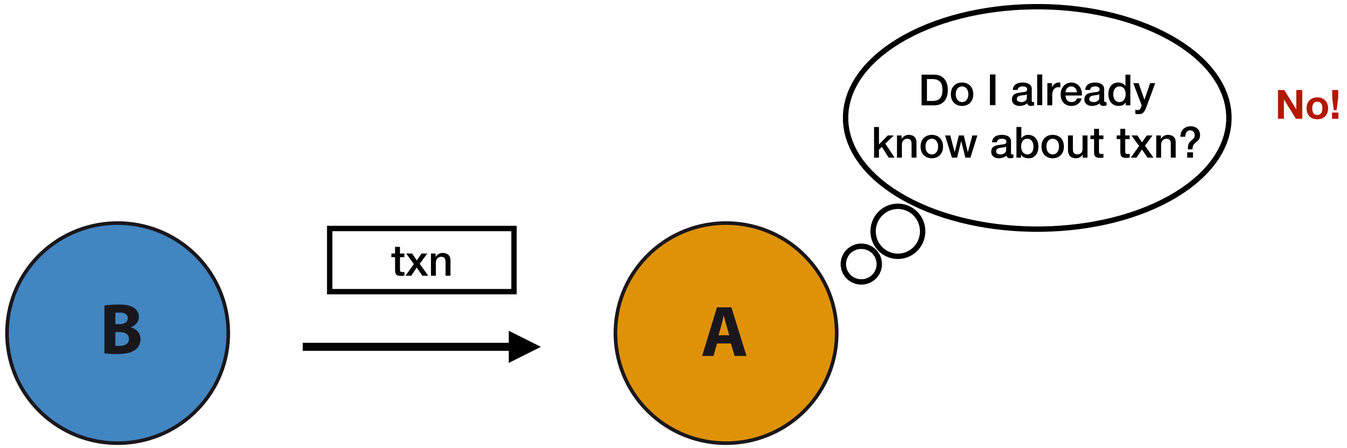
- Known transaction will be rejected

Information propagation (1/2)



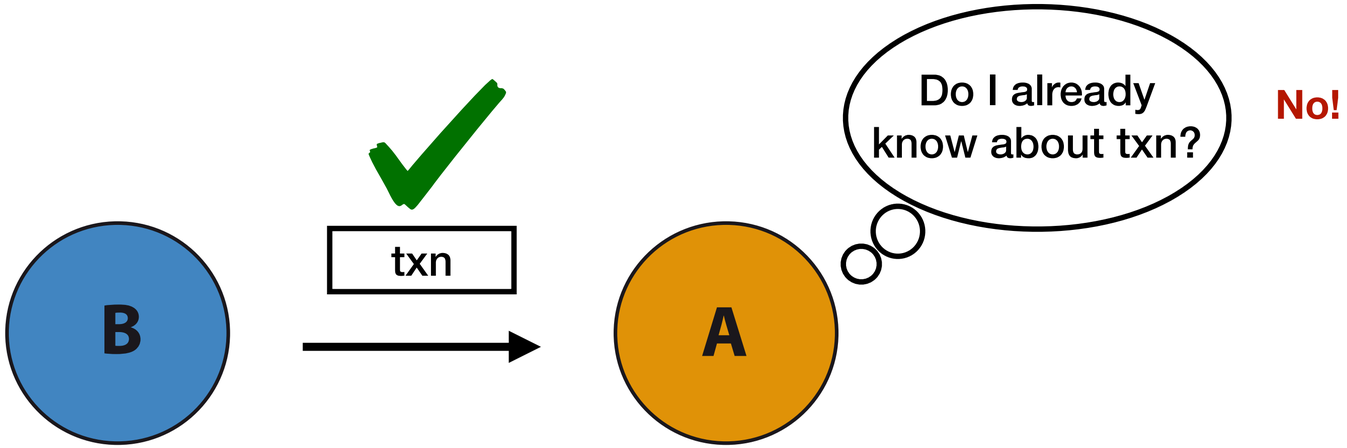
- Known transaction will be rejected

Information propagation (1/2)



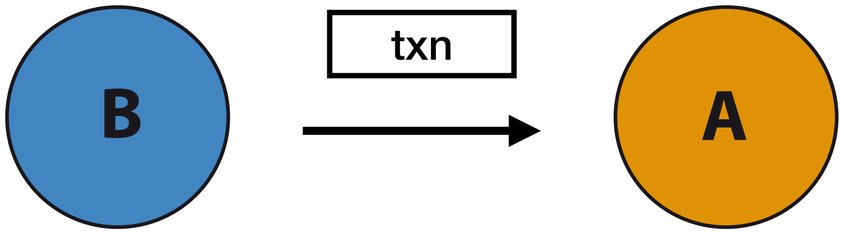
- Known transaction will be rejected

Information propagation (1/2)



- Known transaction will be rejected

Information propagation (1/2)



- Known transaction will be rejected

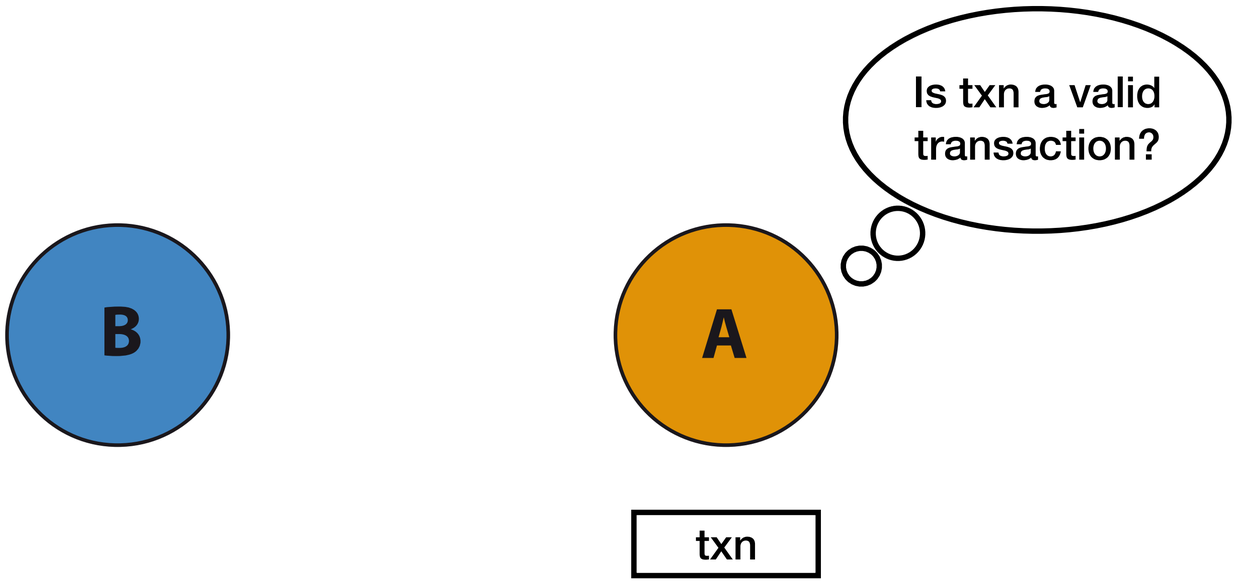
Information propagation (1/2)

- Known transaction will be rejected

Information propagation (1/2)

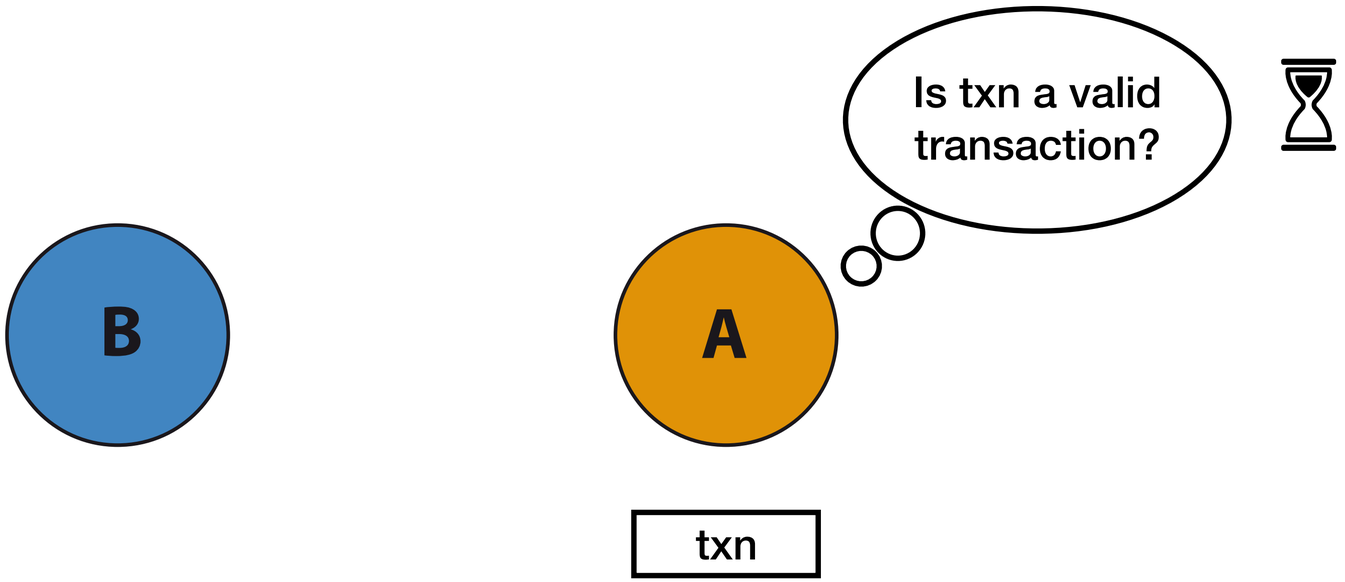
- Known transaction will be rejected

Information propagation (1/2)



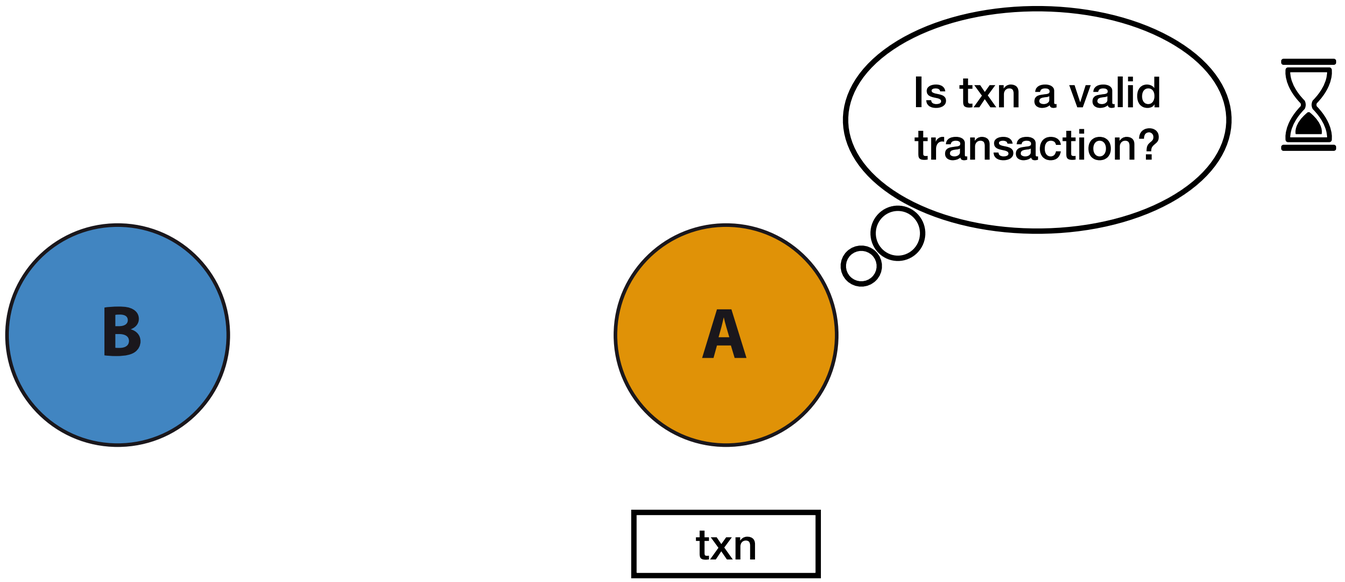
- Known transaction will be rejected

Information propagation (1/2)



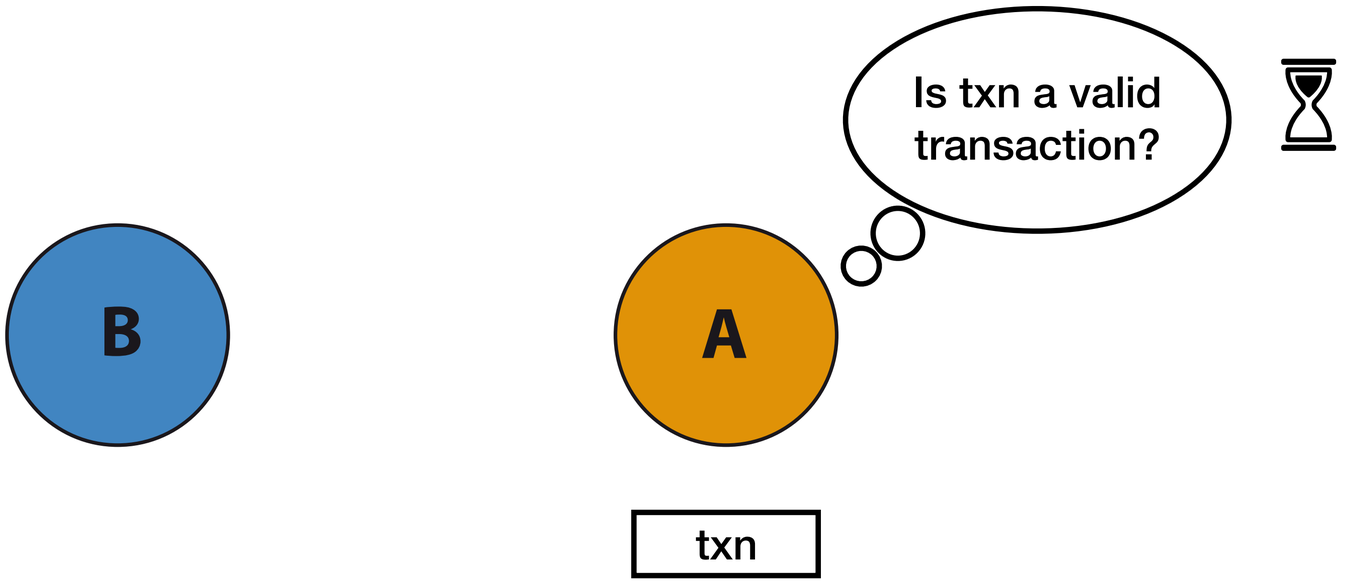
- Known transaction will be rejected

Information propagation (1/2)



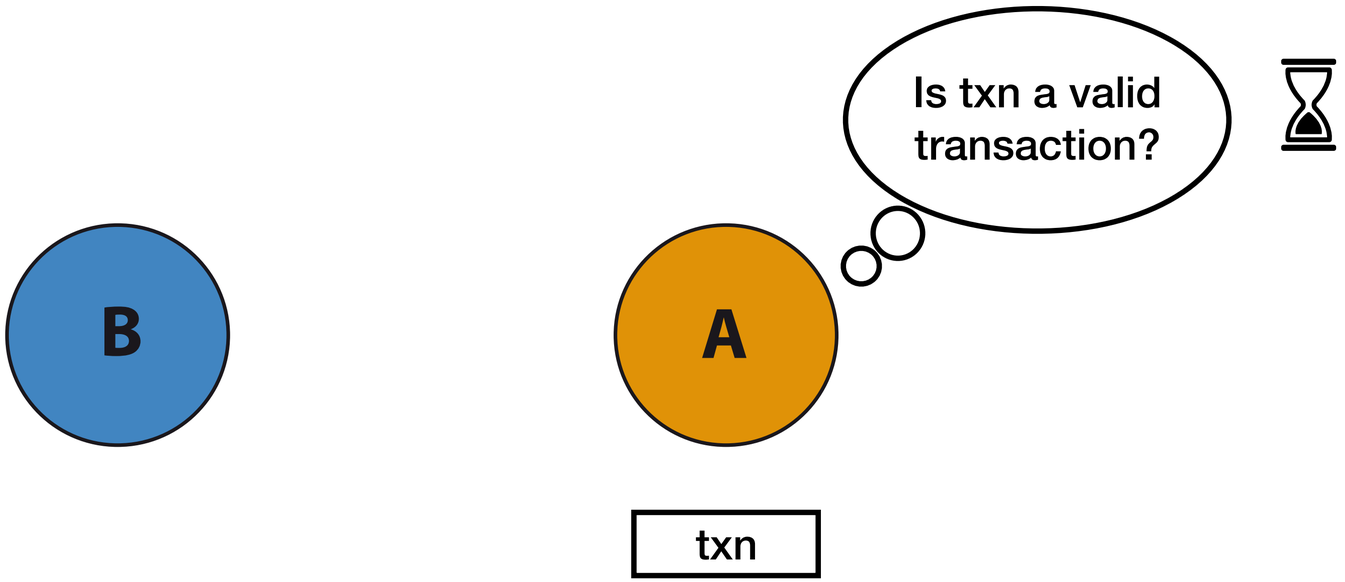
- Known transaction will be rejected

Information propagation (1/2)



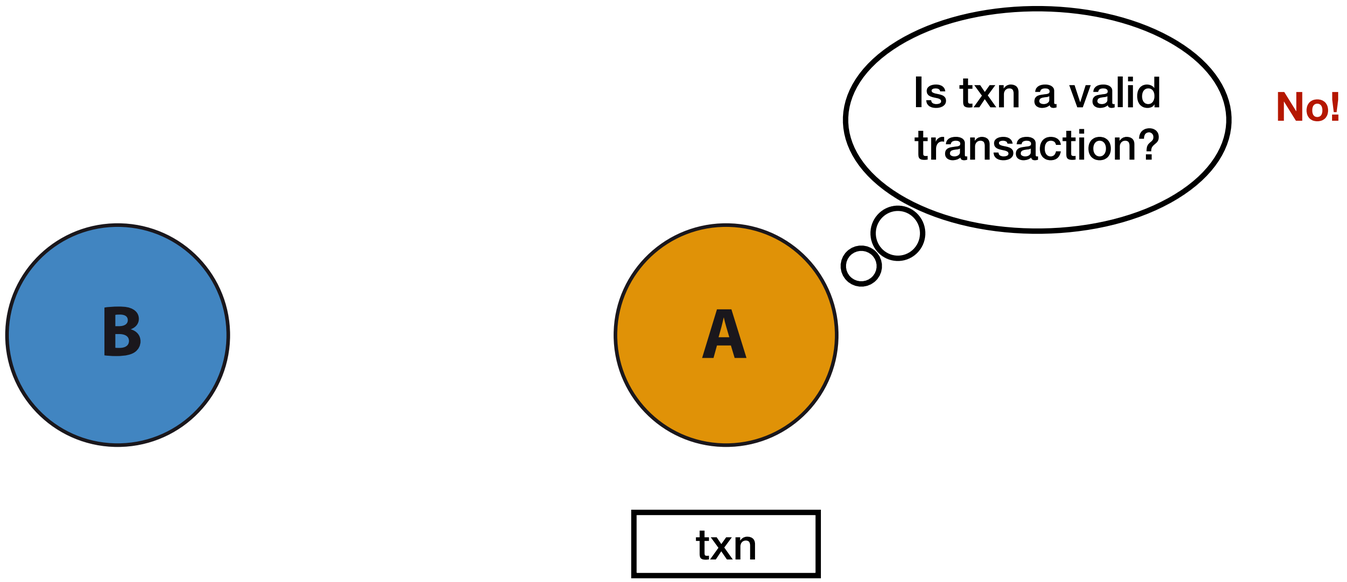
- Known transaction will be rejected

Information propagation (1/2)



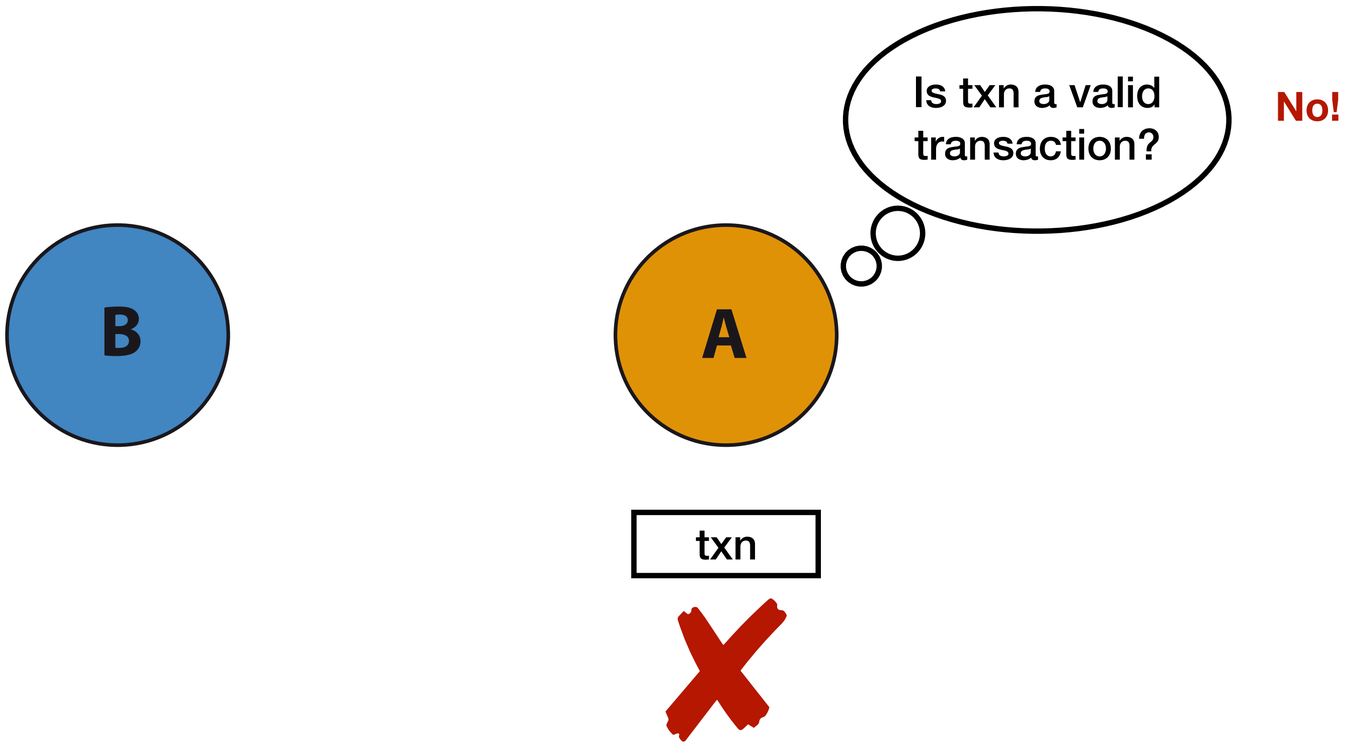
- Known transaction will be rejected

Information propagation (1/2)



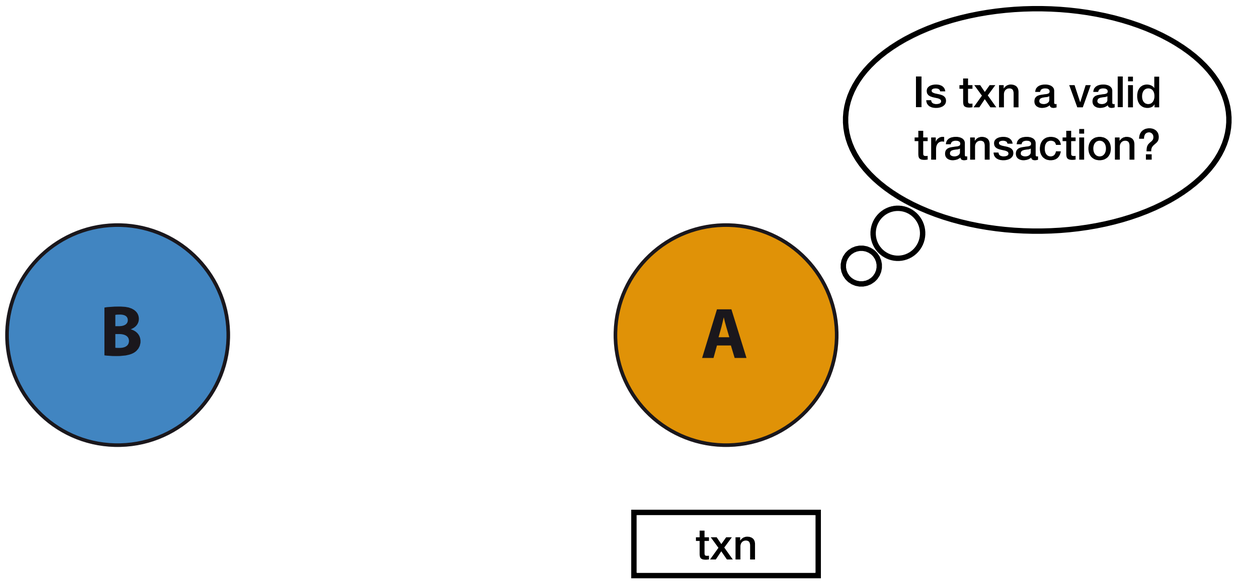
- Known transaction will be rejected

Information propagation (1/2)



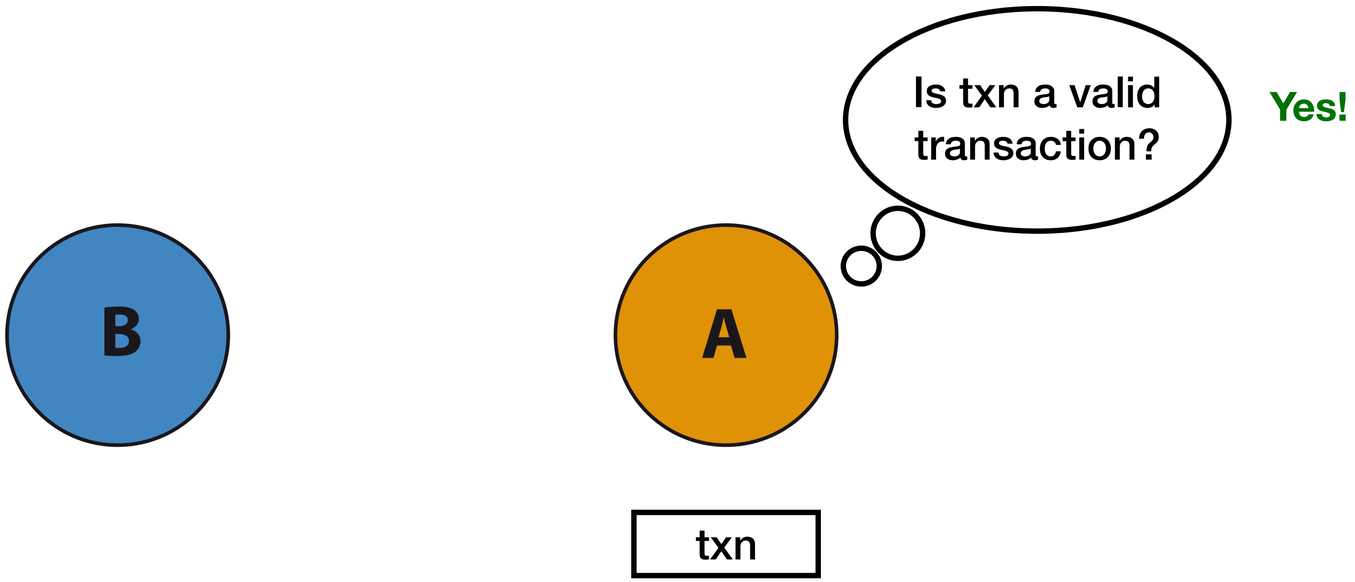
- Known transaction will be rejected
- Invalid transaction will also be rejected

Information propagation (1/2)



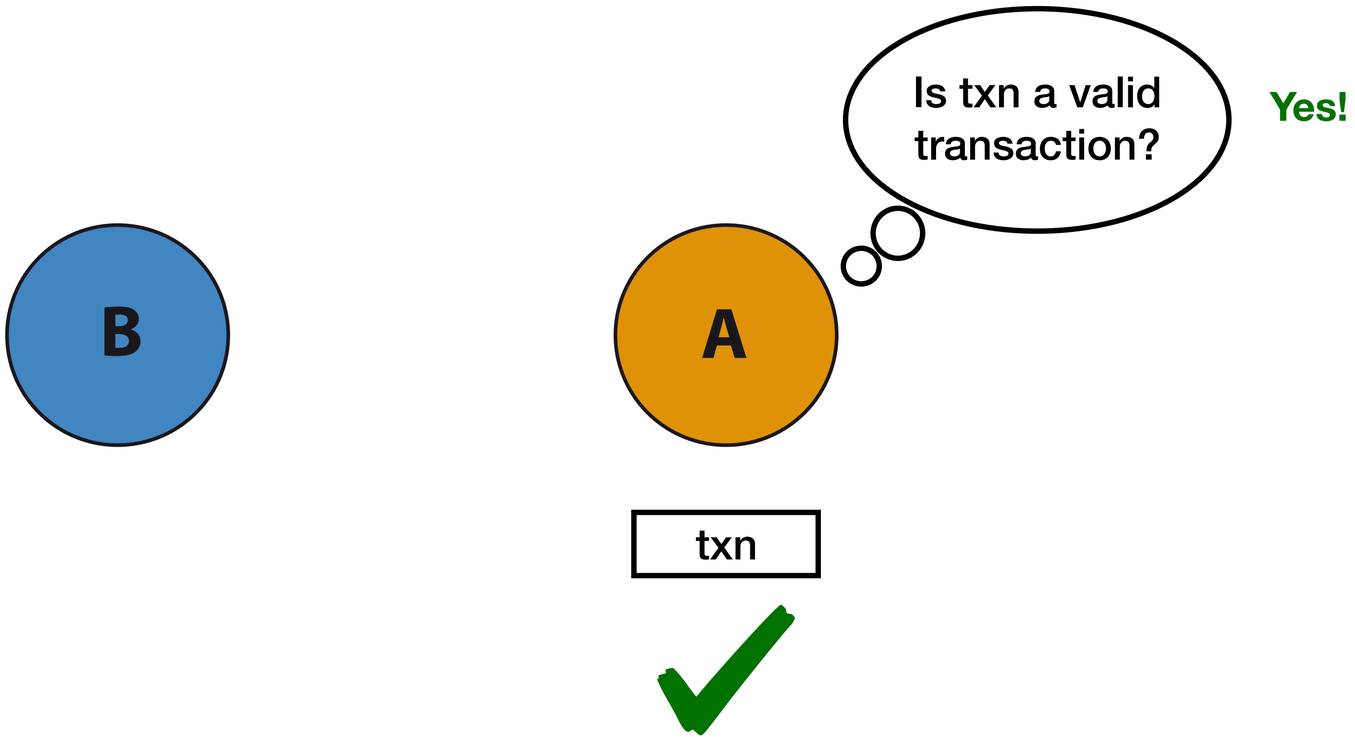
- Known transaction will be rejected
- Invalid transaction will also be rejected

Information propagation (1/2)



- Known transaction will be rejected
- Invalid transaction will also be rejected

Information propagation (1/2)

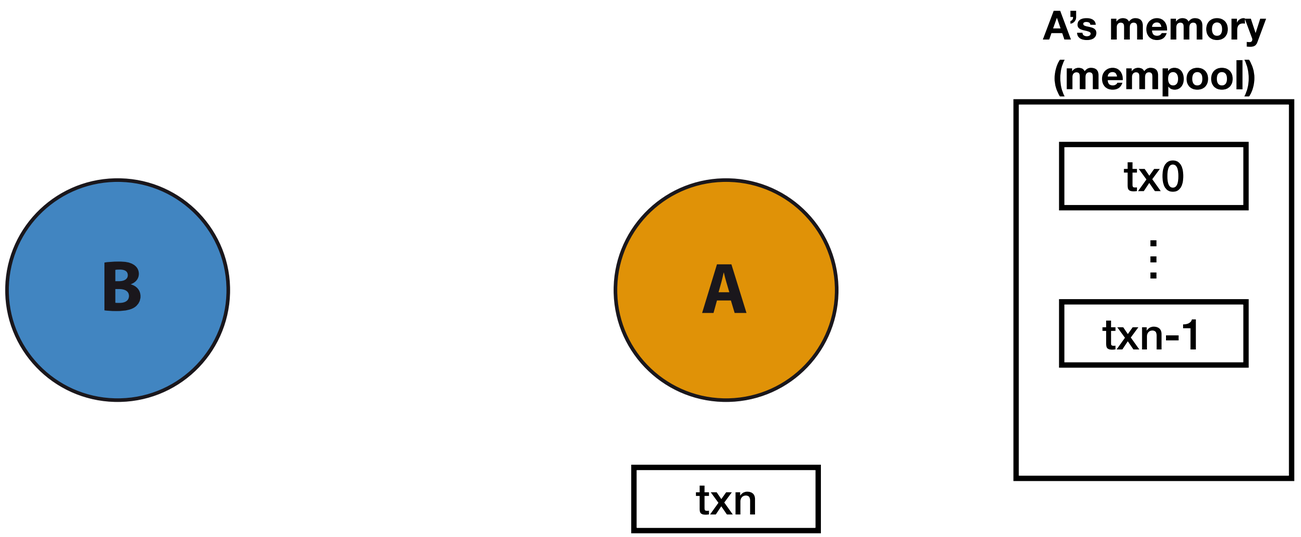


- Known transaction will be rejected
- Invalid transaction will also be rejected

Information propagation (1/2)

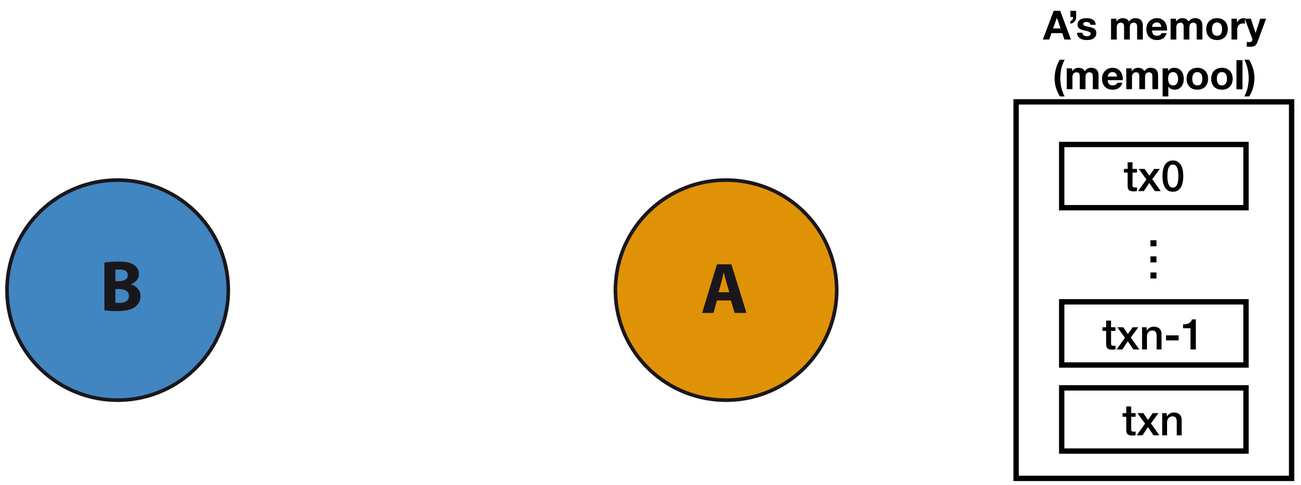
- Known transaction will be rejected
- Invalid transaction will also be rejected

Information propagation (1/2)



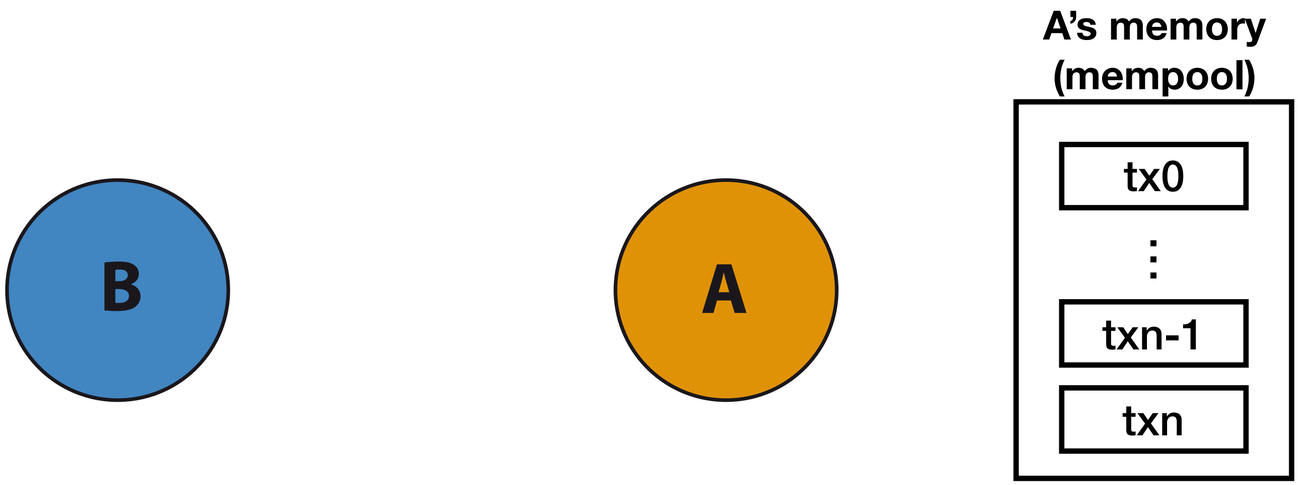
- Known transaction will be rejected
- Invalid transaction will also be rejected

Information propagation (1/2)



- Known transaction will be rejected
- Invalid transaction will also be rejected

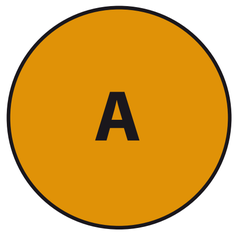
Information propagation (1/2)



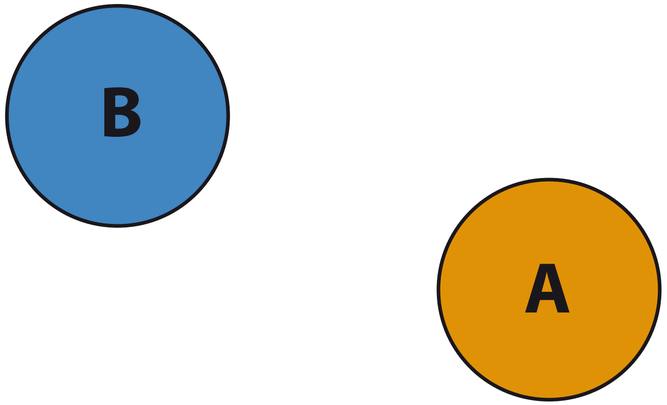
- Known transaction will be rejected
- Invalid transaction will also be rejected
- Valid (new) transactions will be kept in memory (mempool)

Information propagation (2/2)

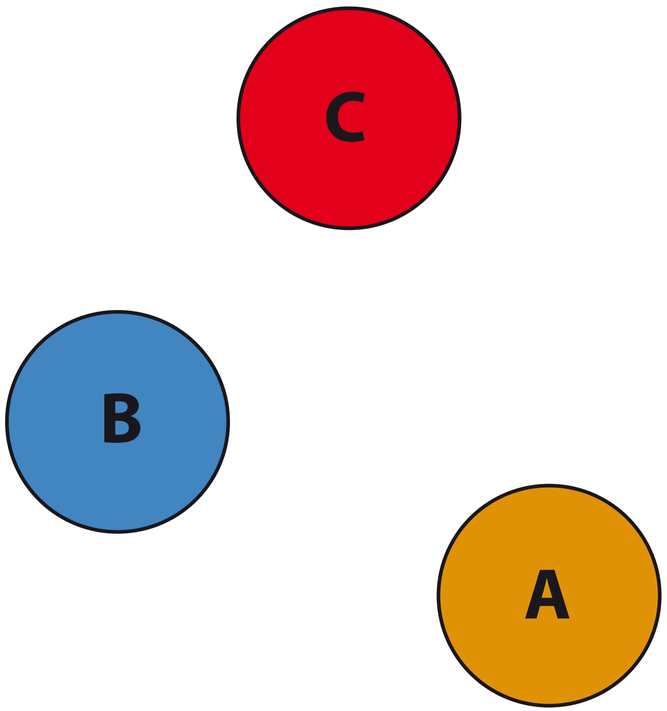
Information propagation (2/2)



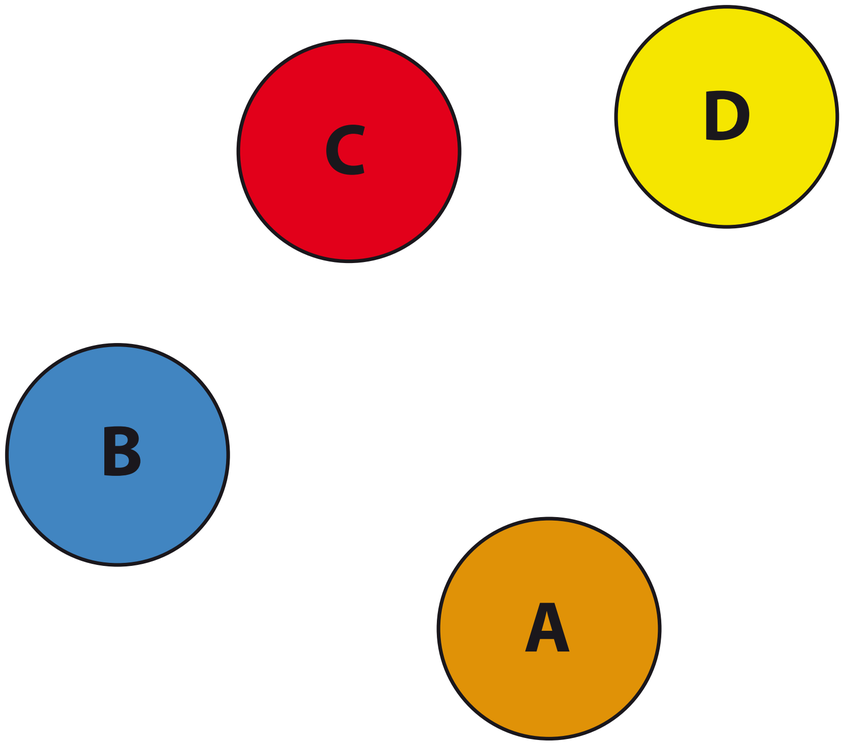
Information propagation (2/2)



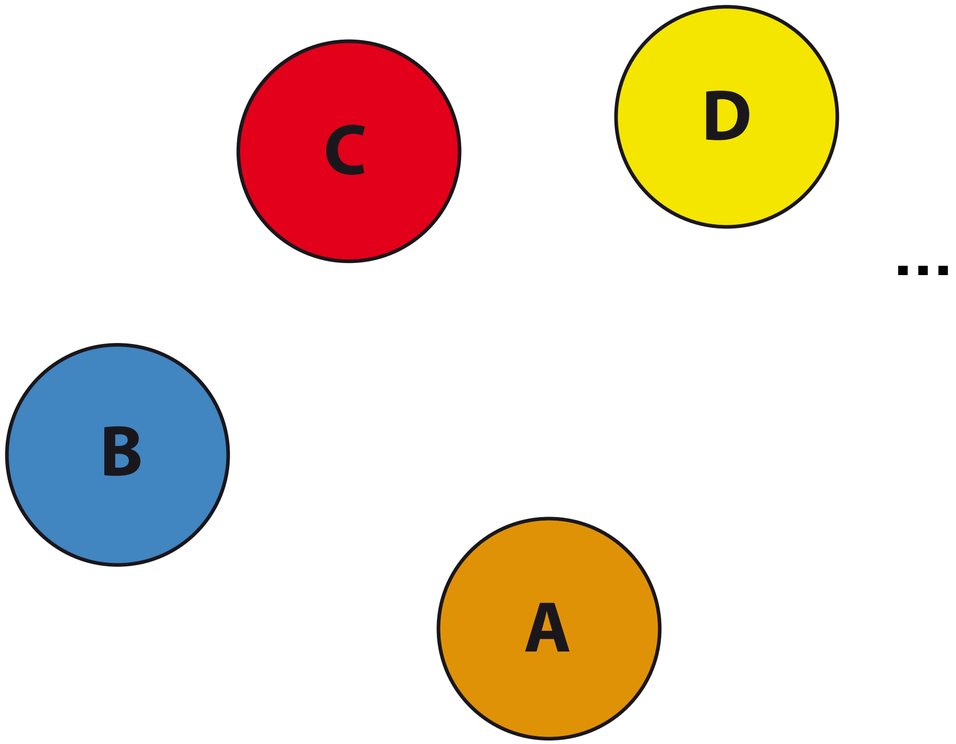
Information propagation (2/2)



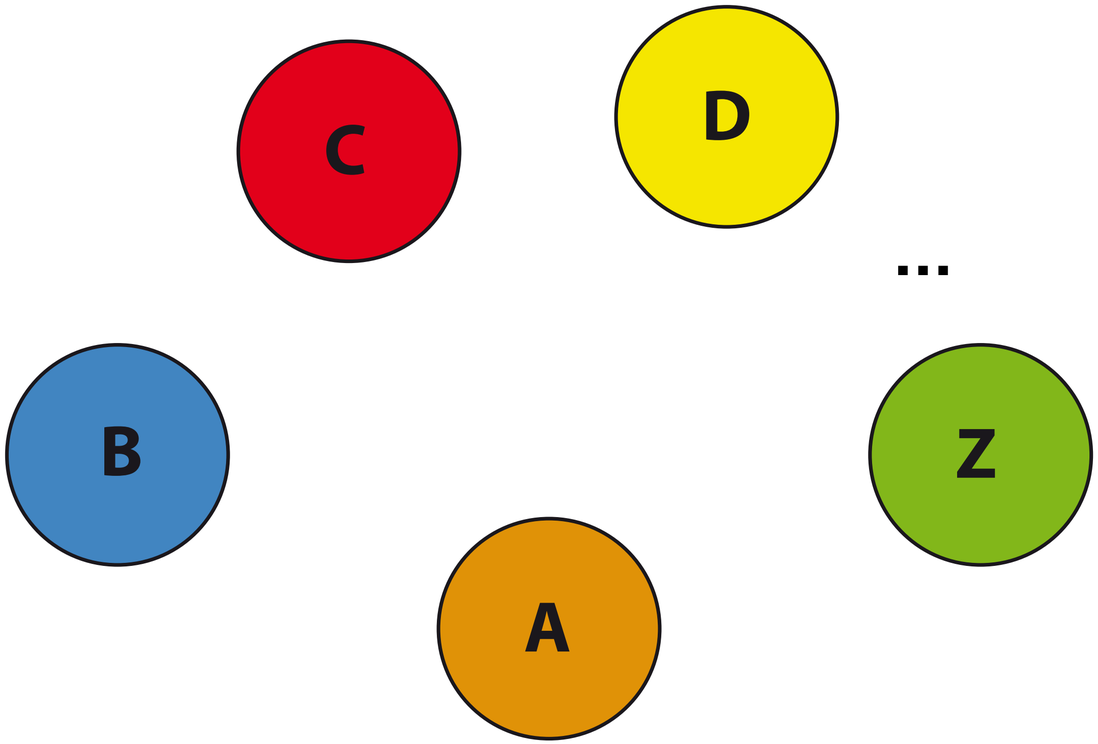
Information propagation (2/2)



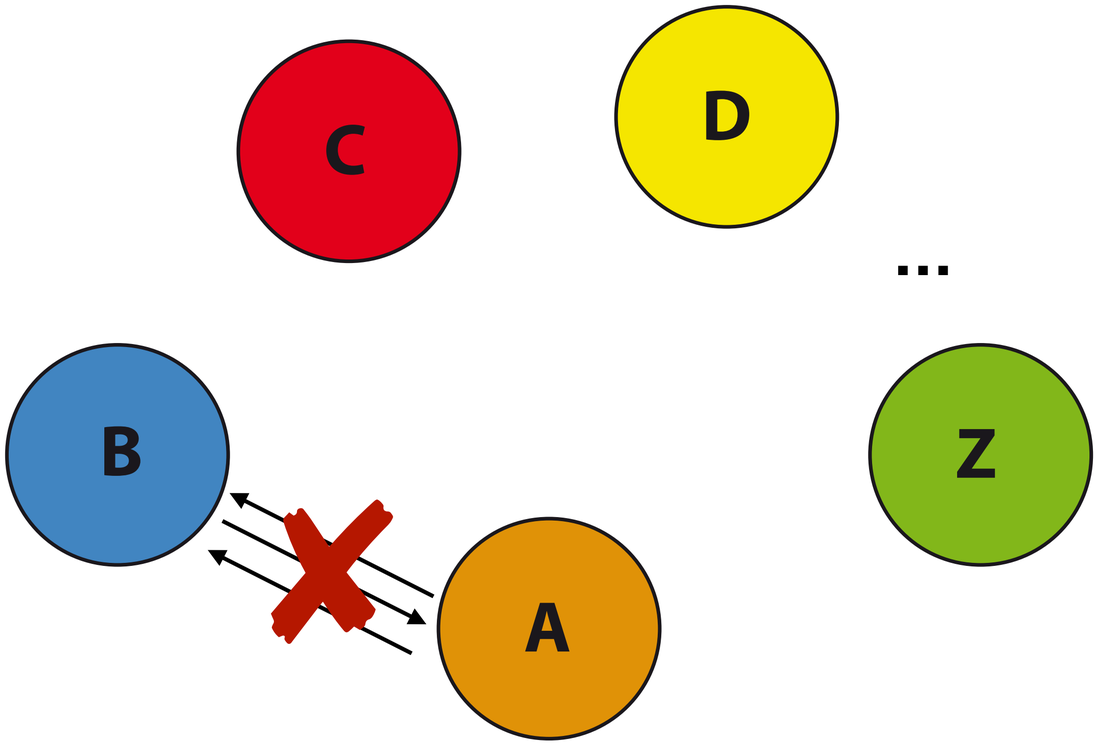
Information propagation (2/2)



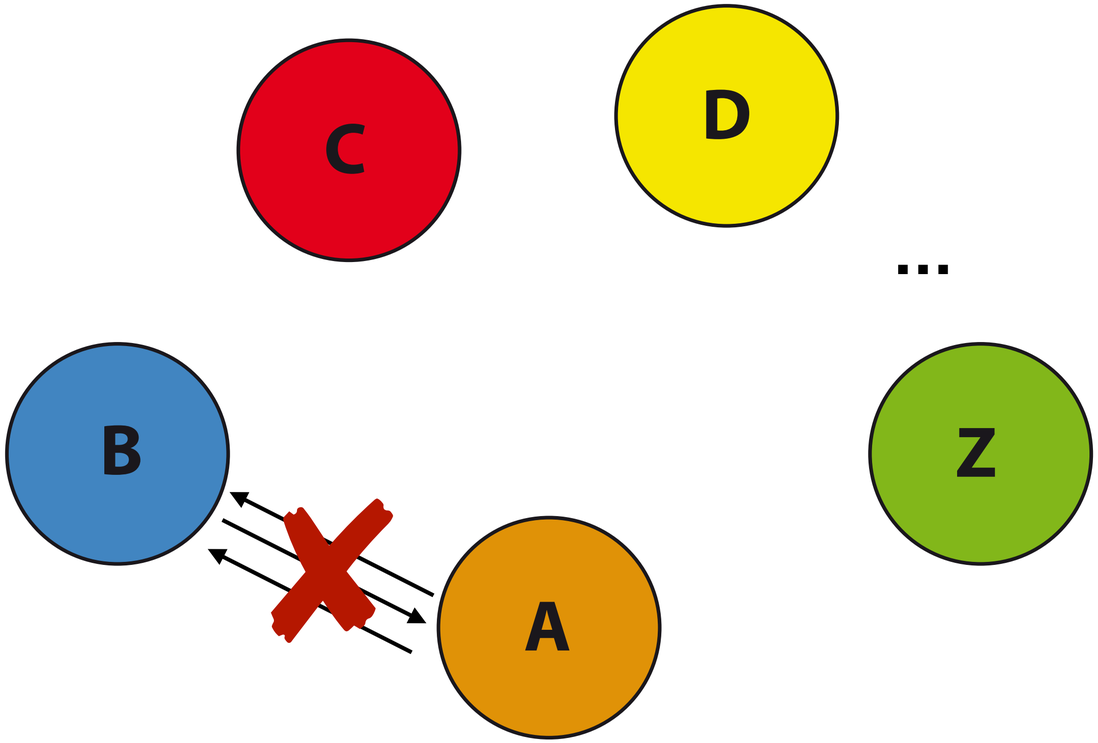
Information propagation (2/2)

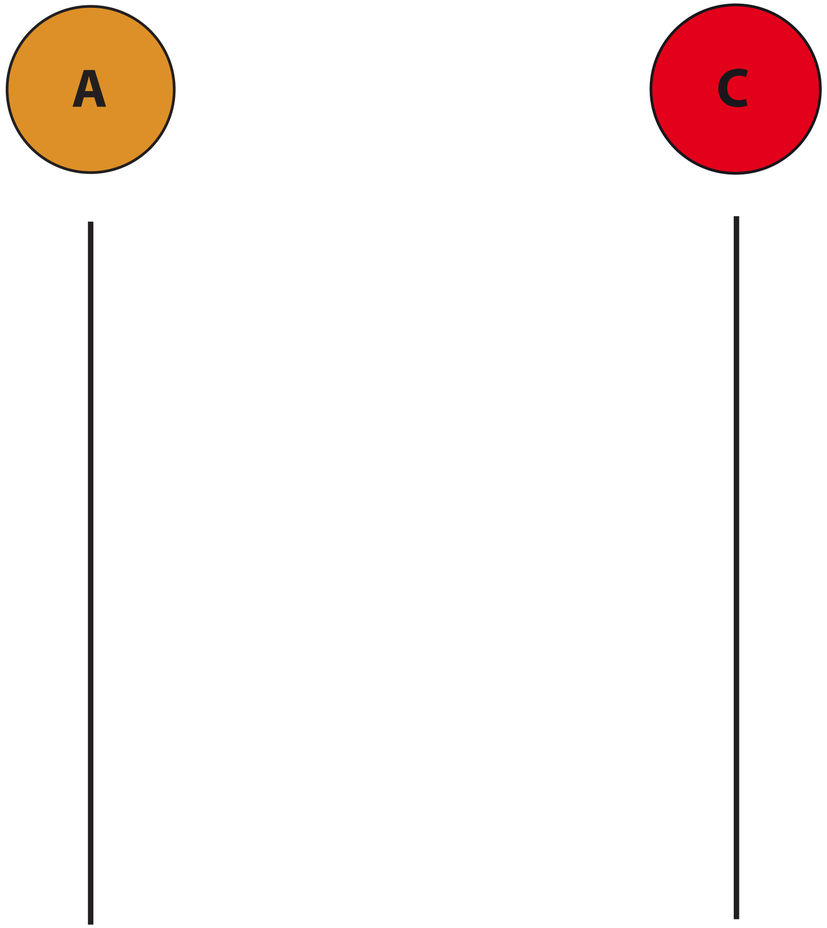


Information propagation (2/2)

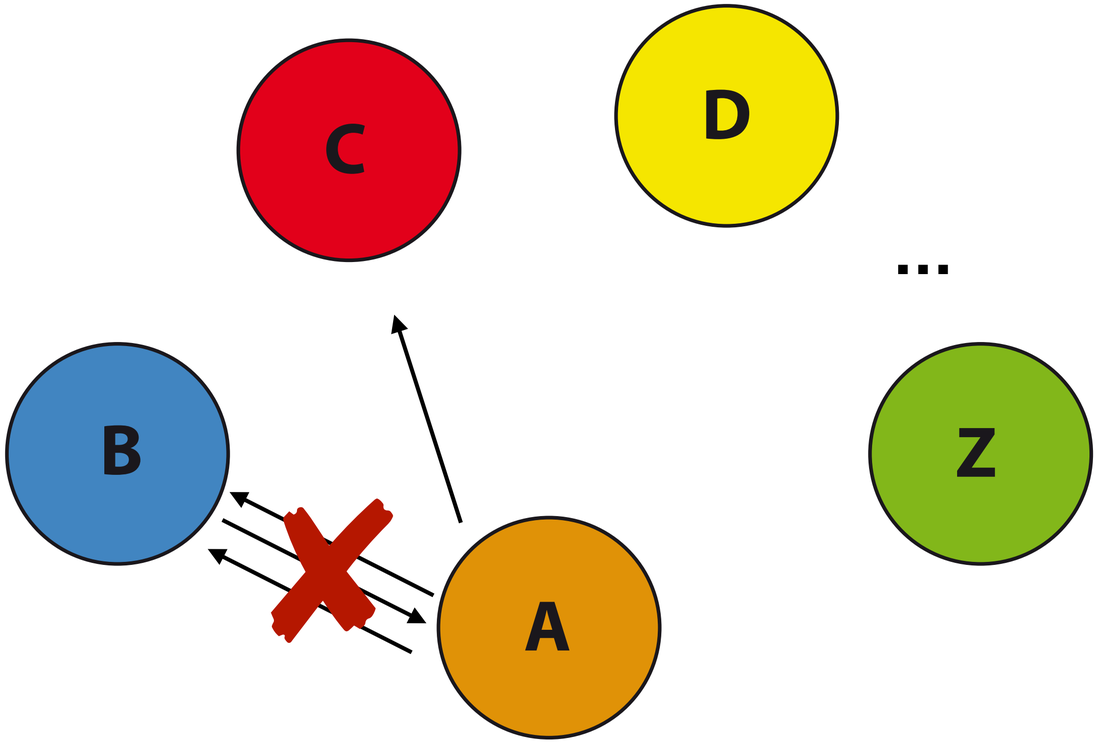


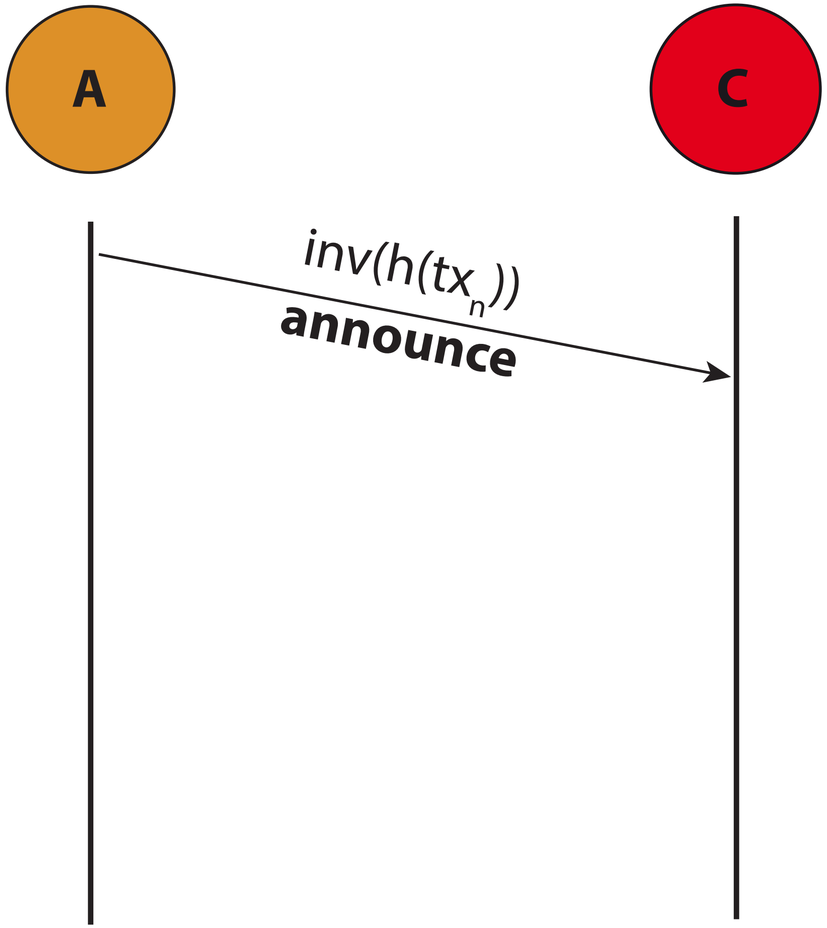
Information propagation (2/2)



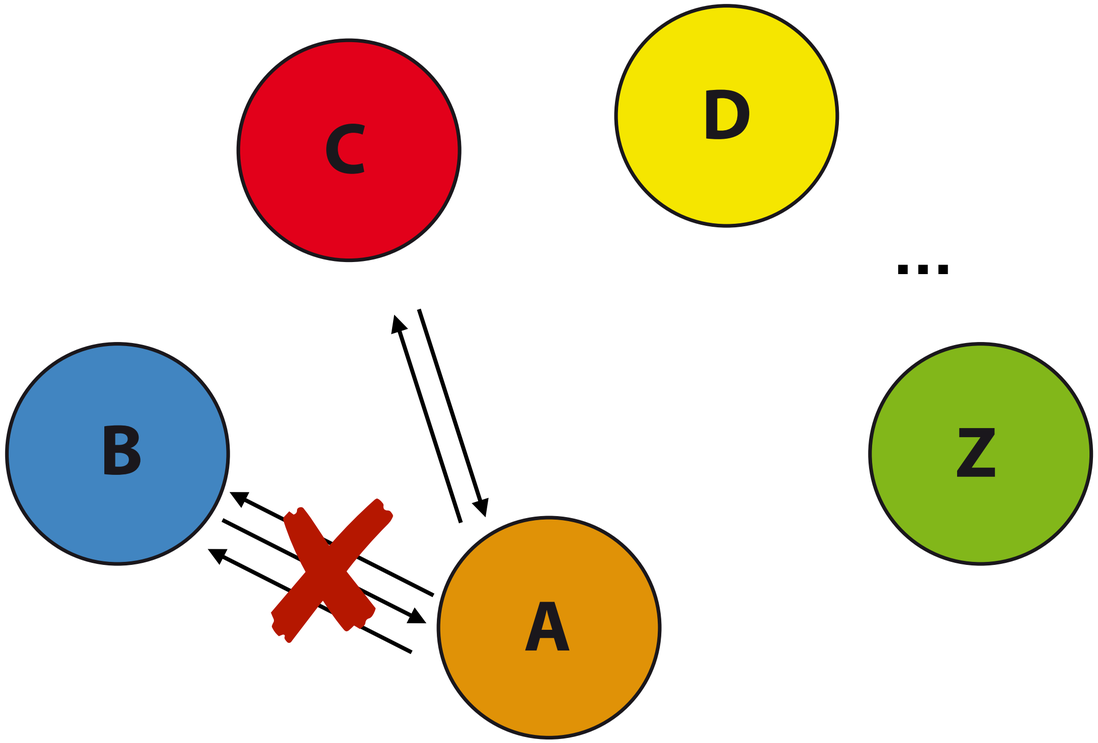


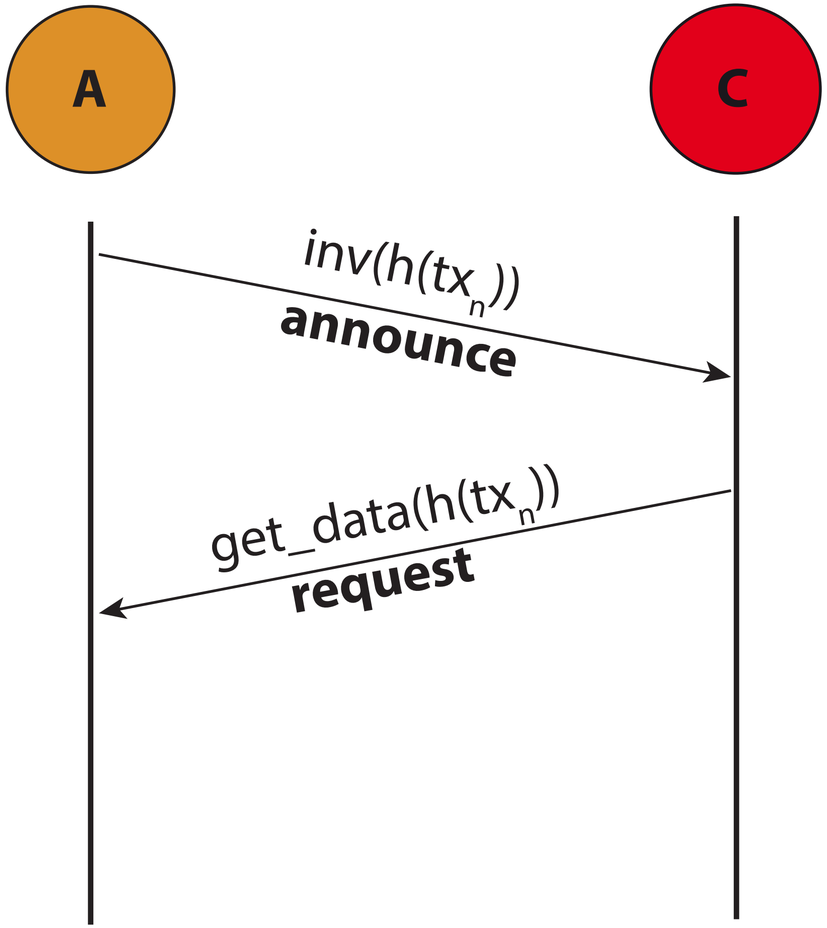
Information propagation (2/2)



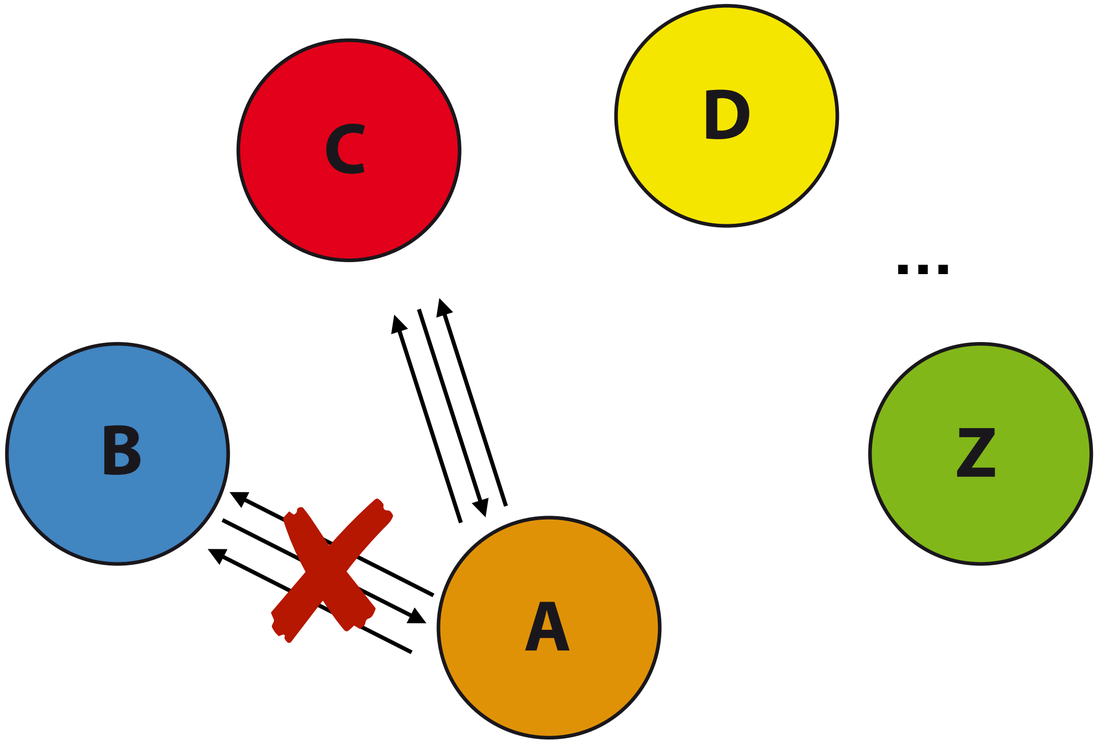


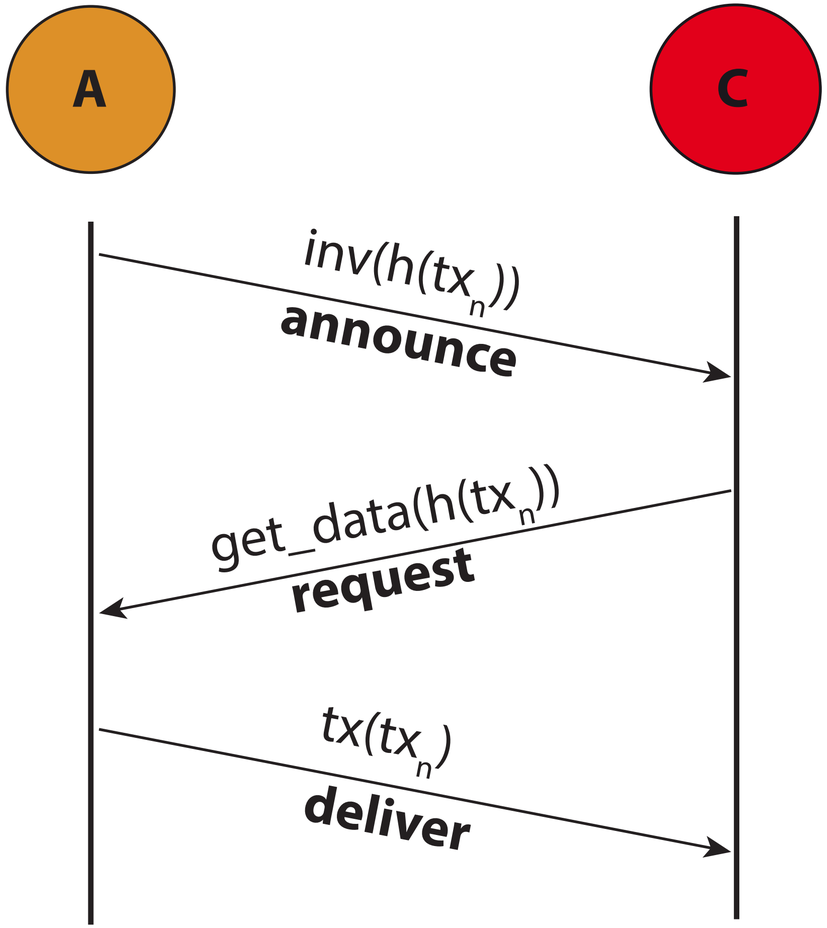
Information propagation (2/2)



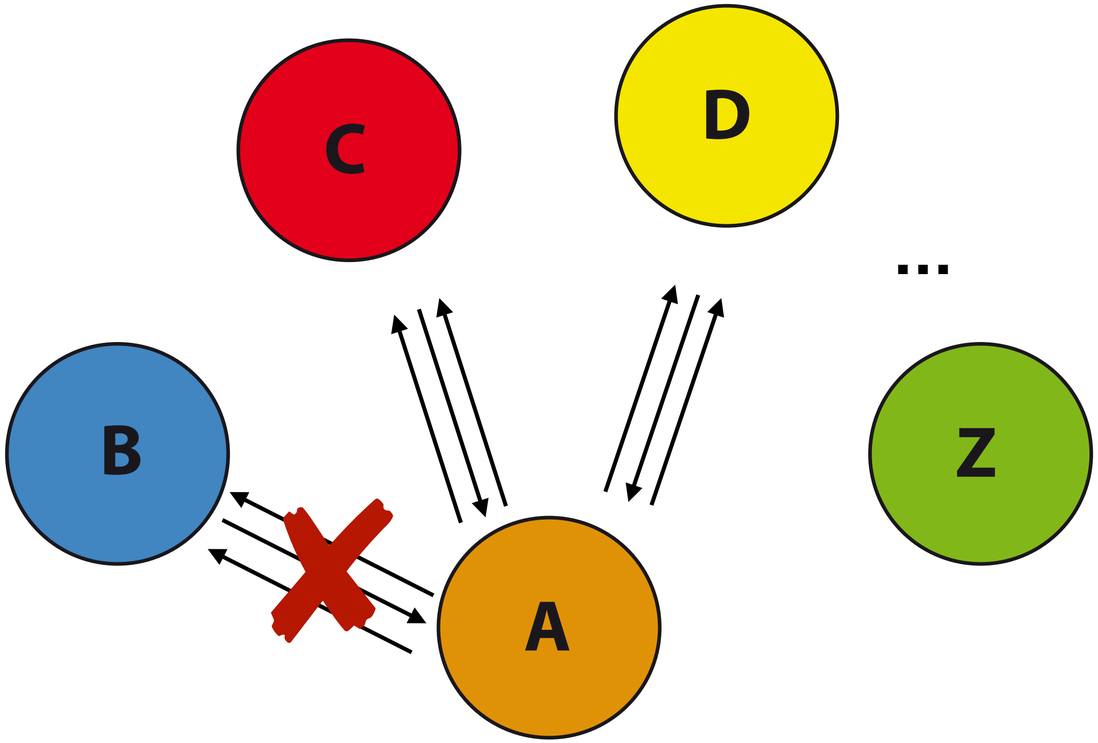


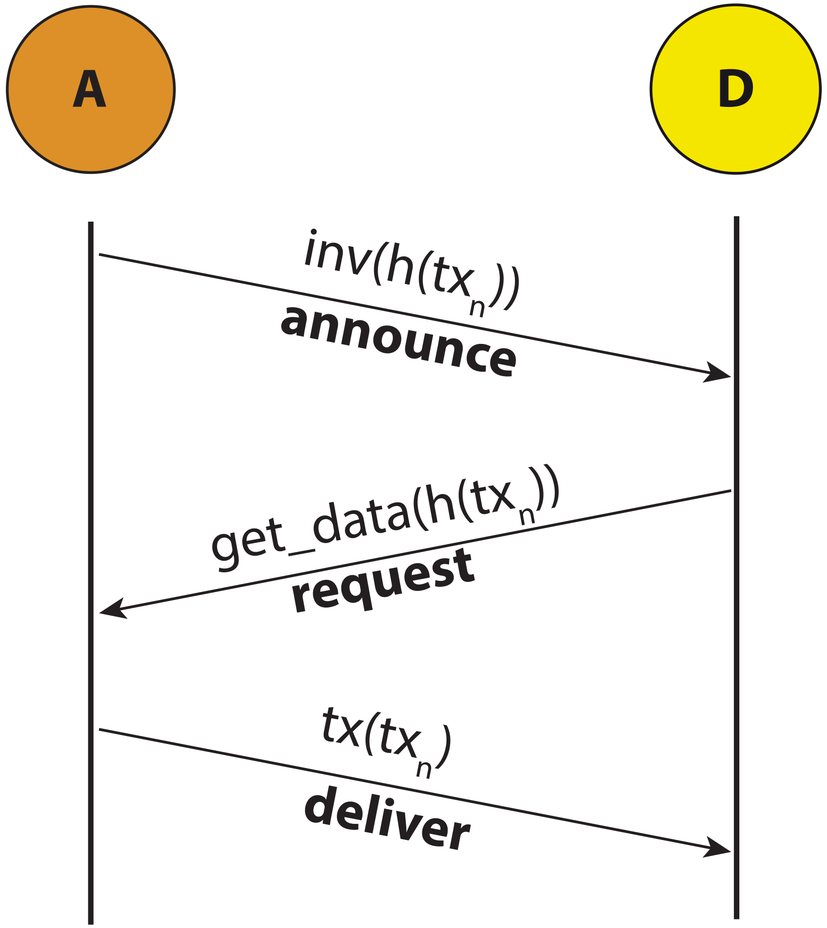
Information propagation (2/2)



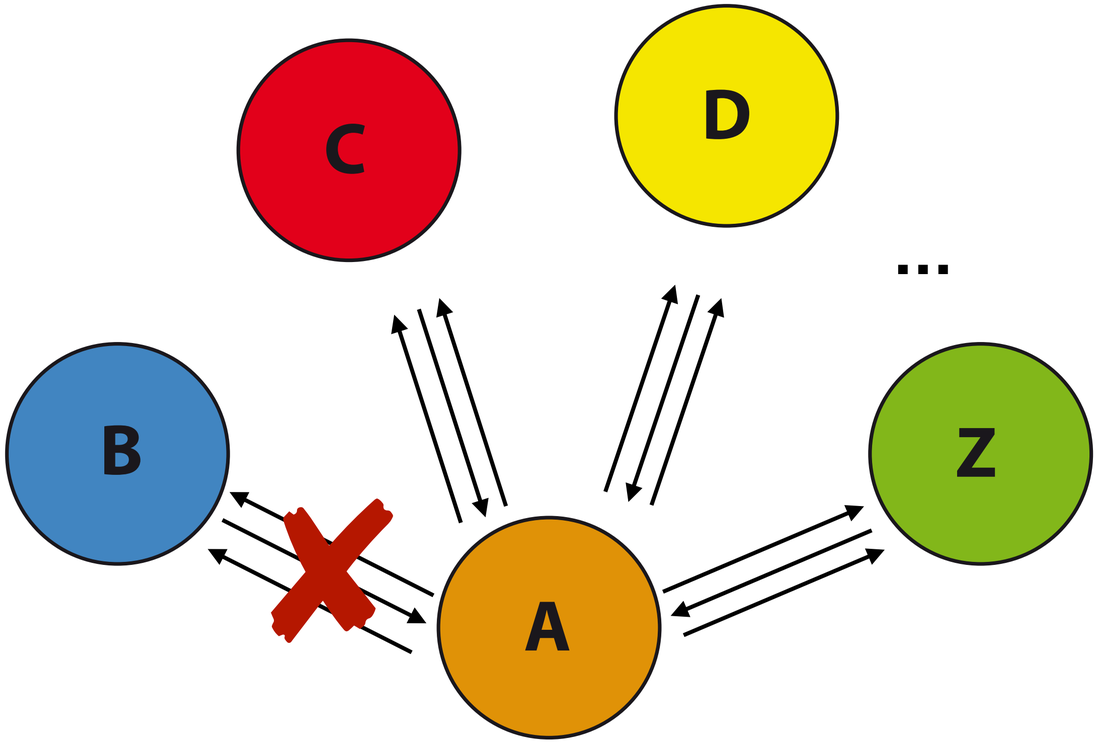


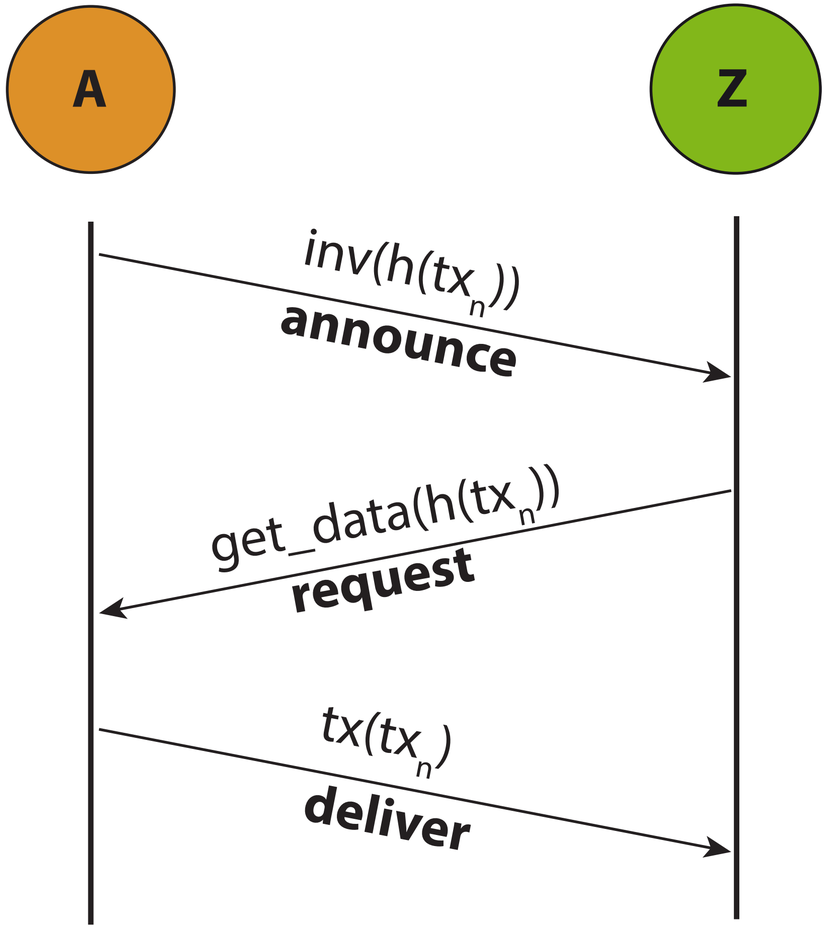
Information propagation (2/2)



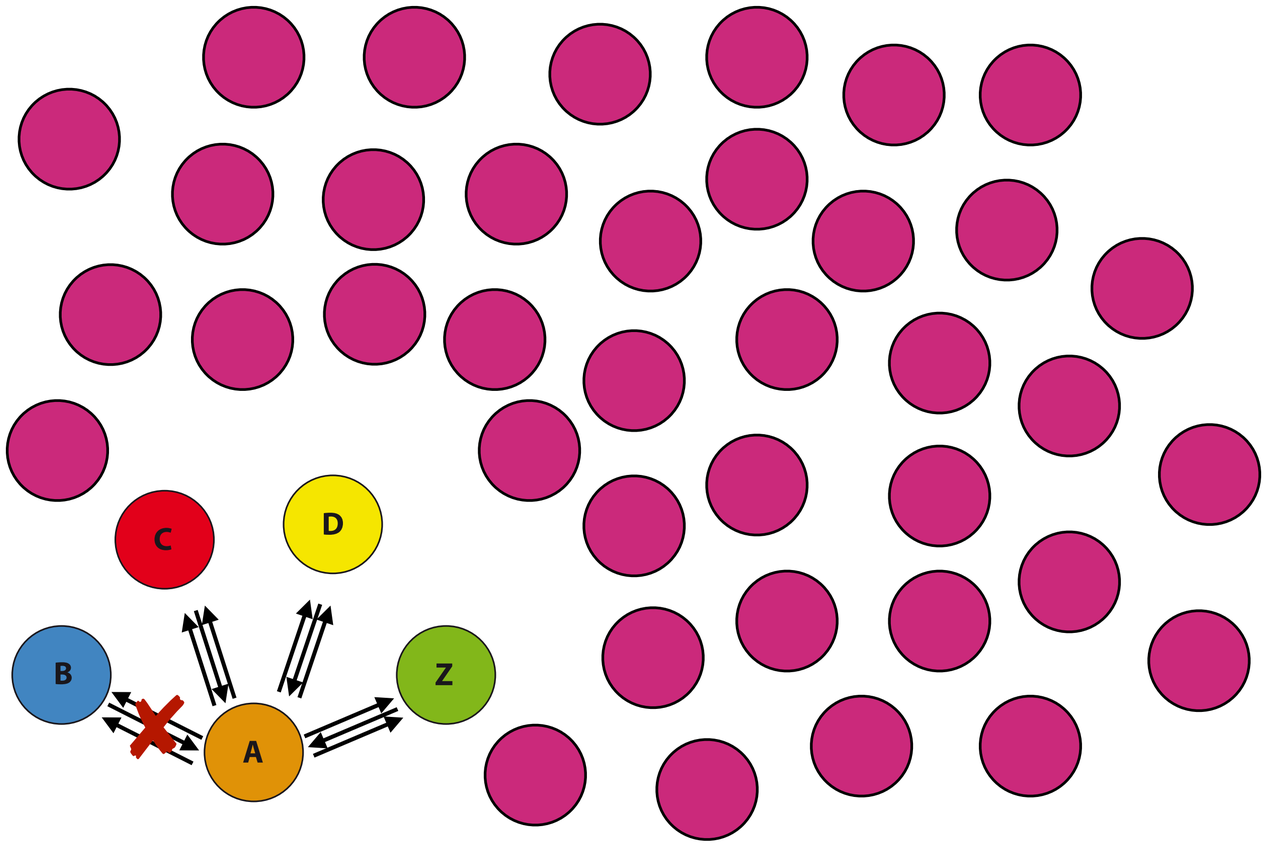


Information propagation (2/2)

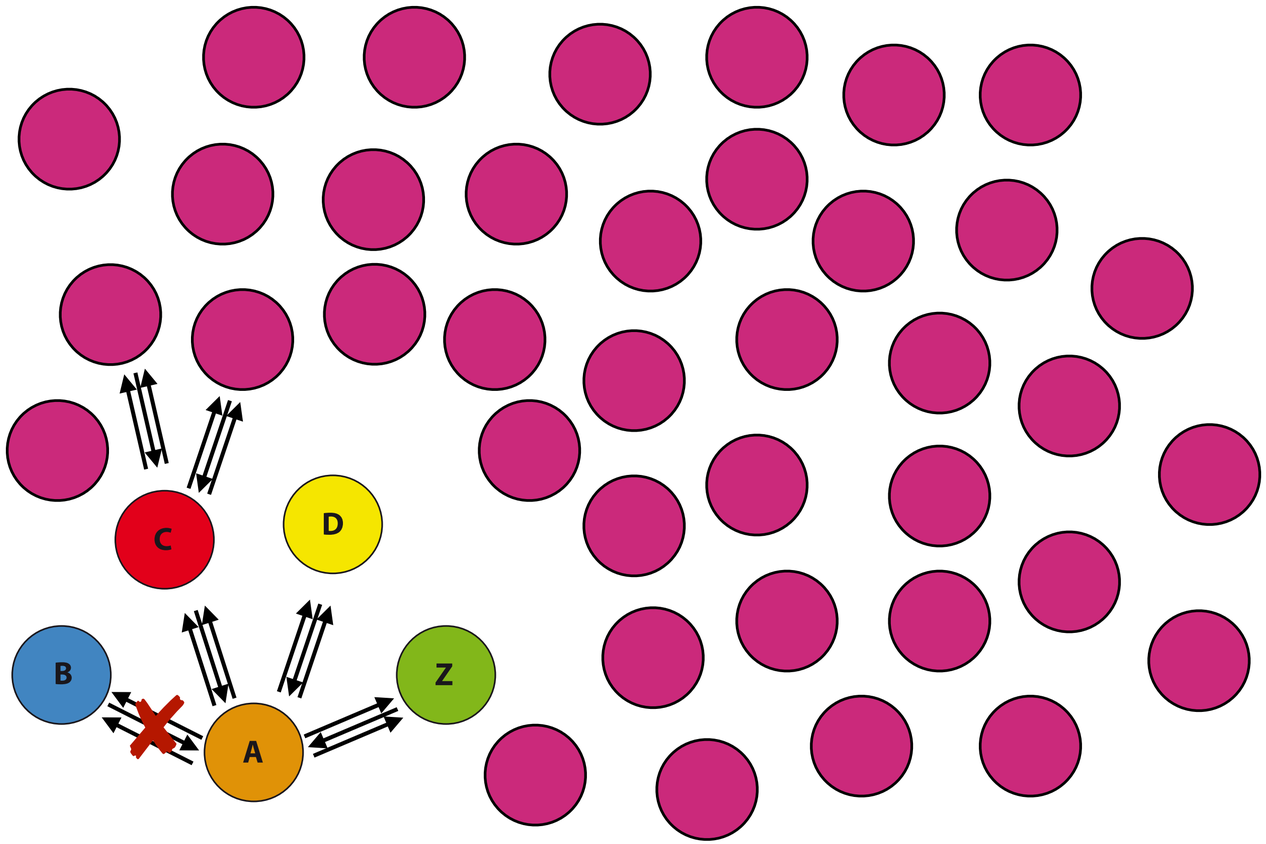




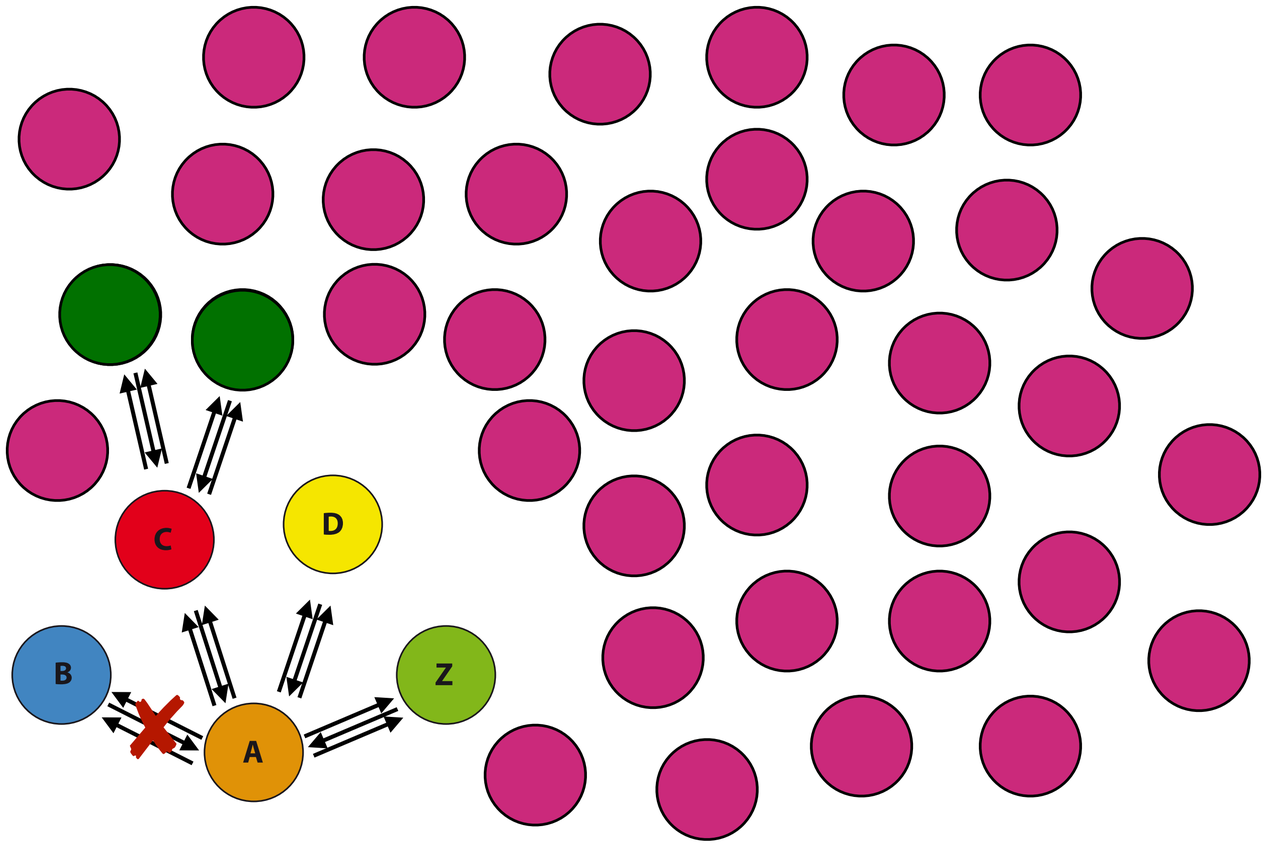
Information propagation (2/2)



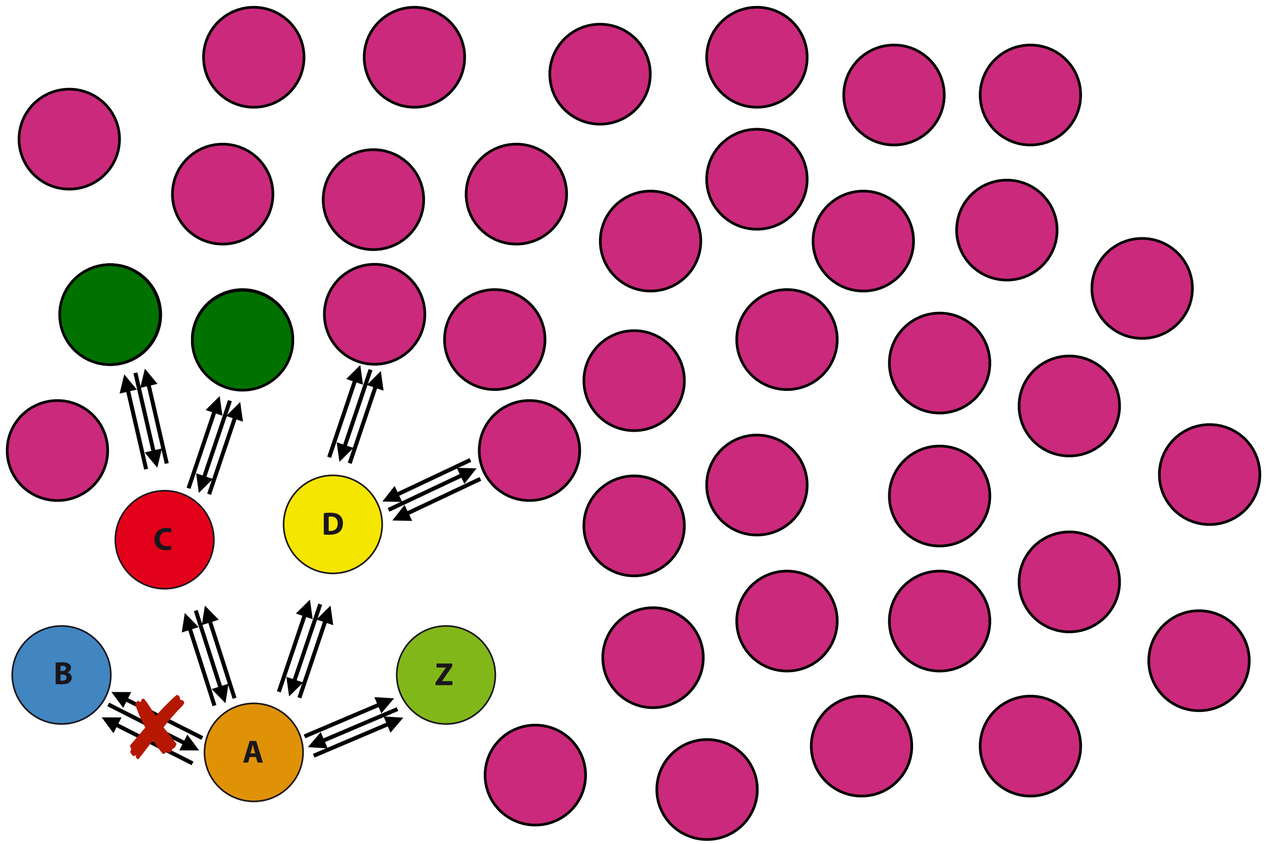
Information propagation (2/2)



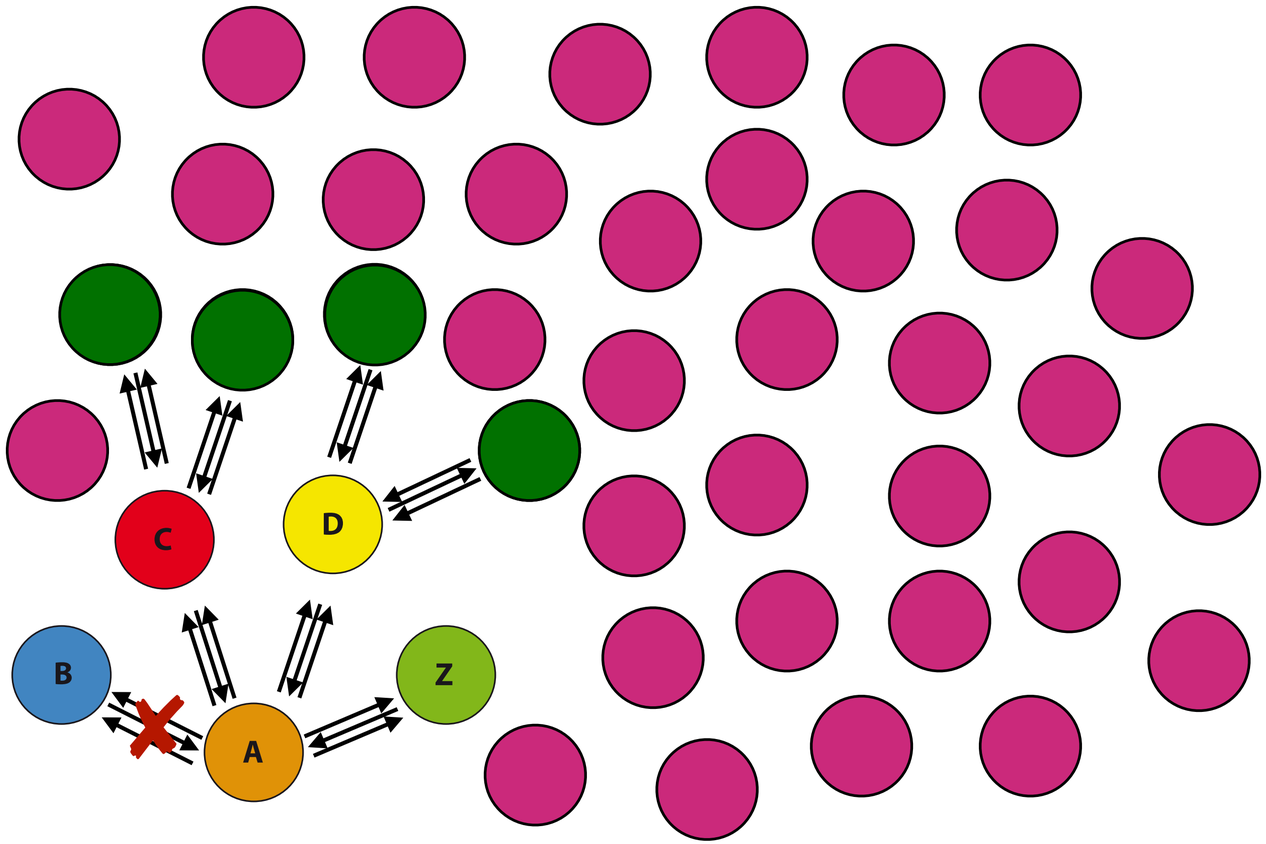
Information propagation (2/2)



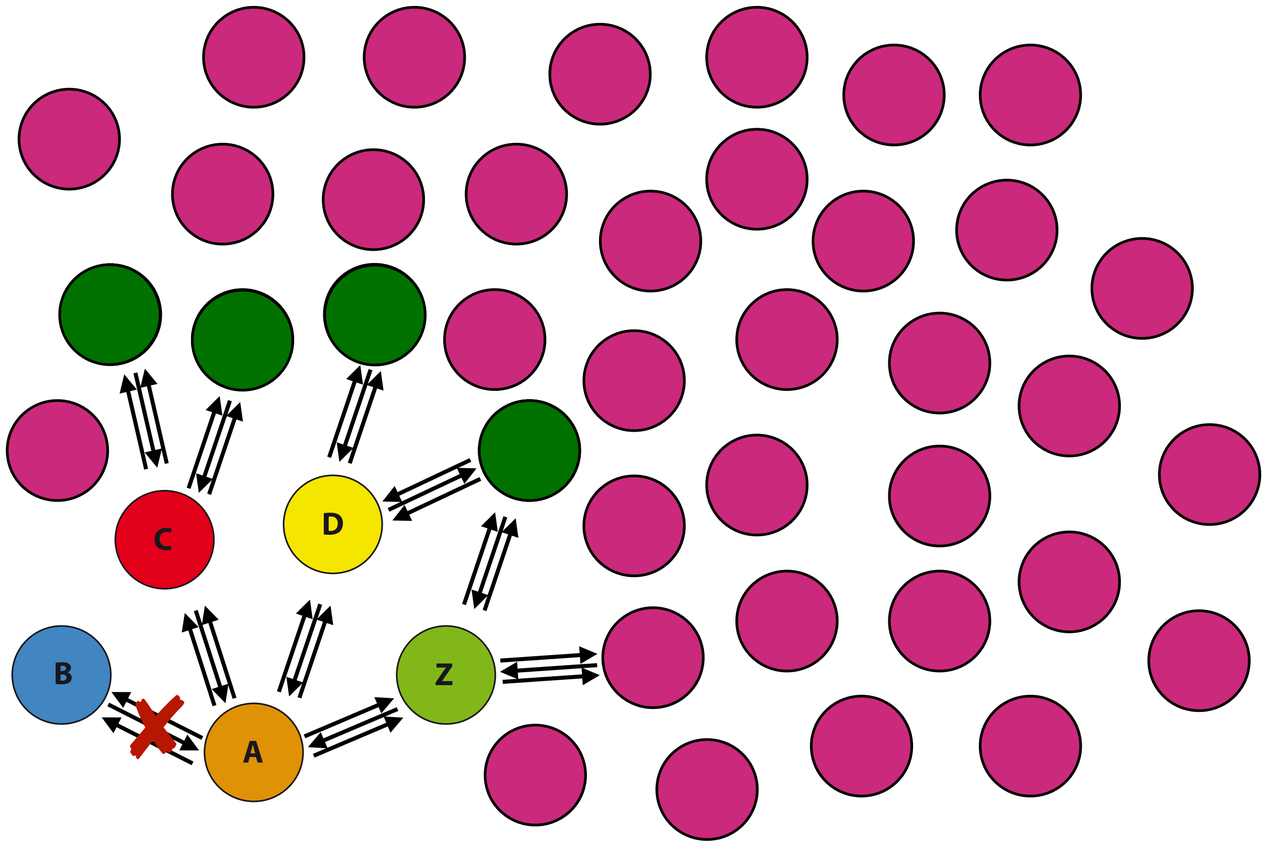
Information propagation (2/2)



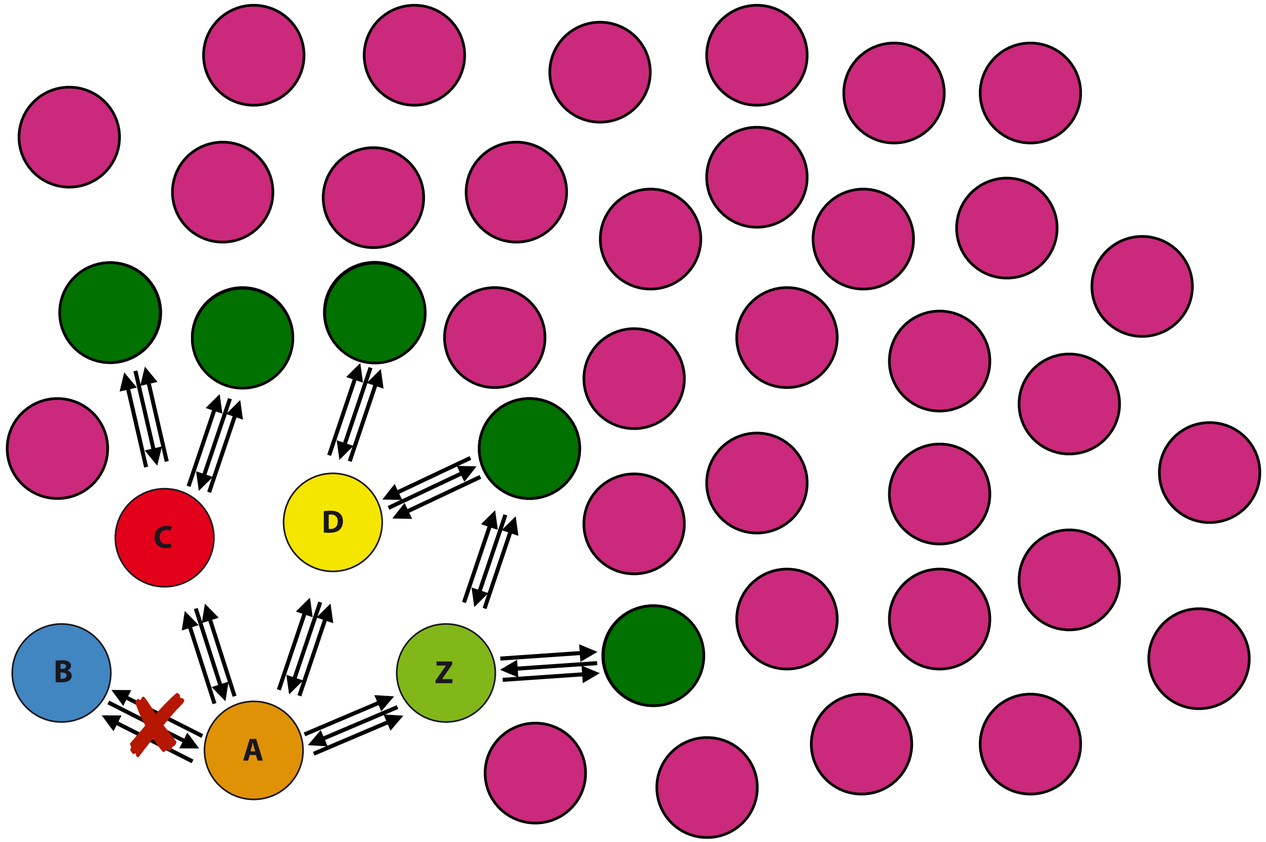
Information propagation (2/2)



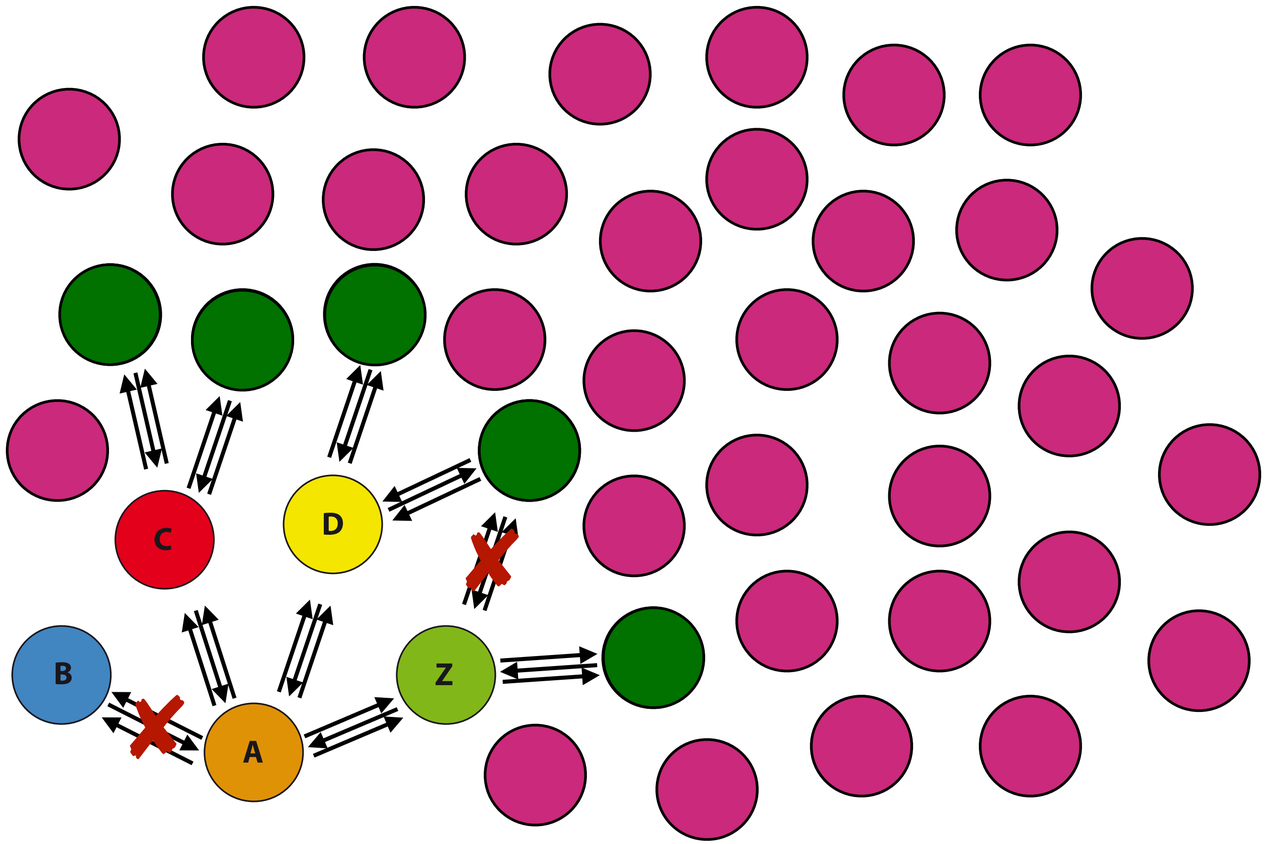
Information propagation (2/2)



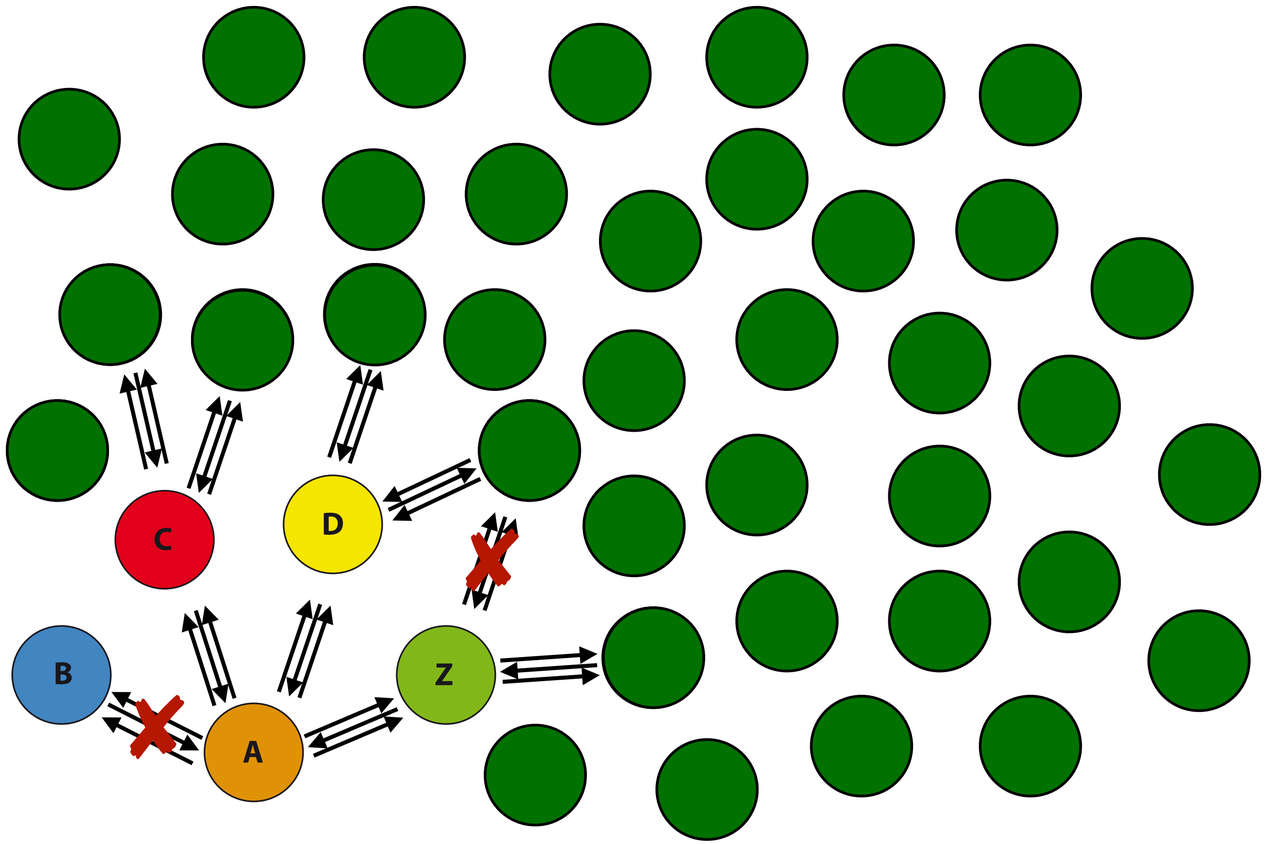
Information propagation (2/2)



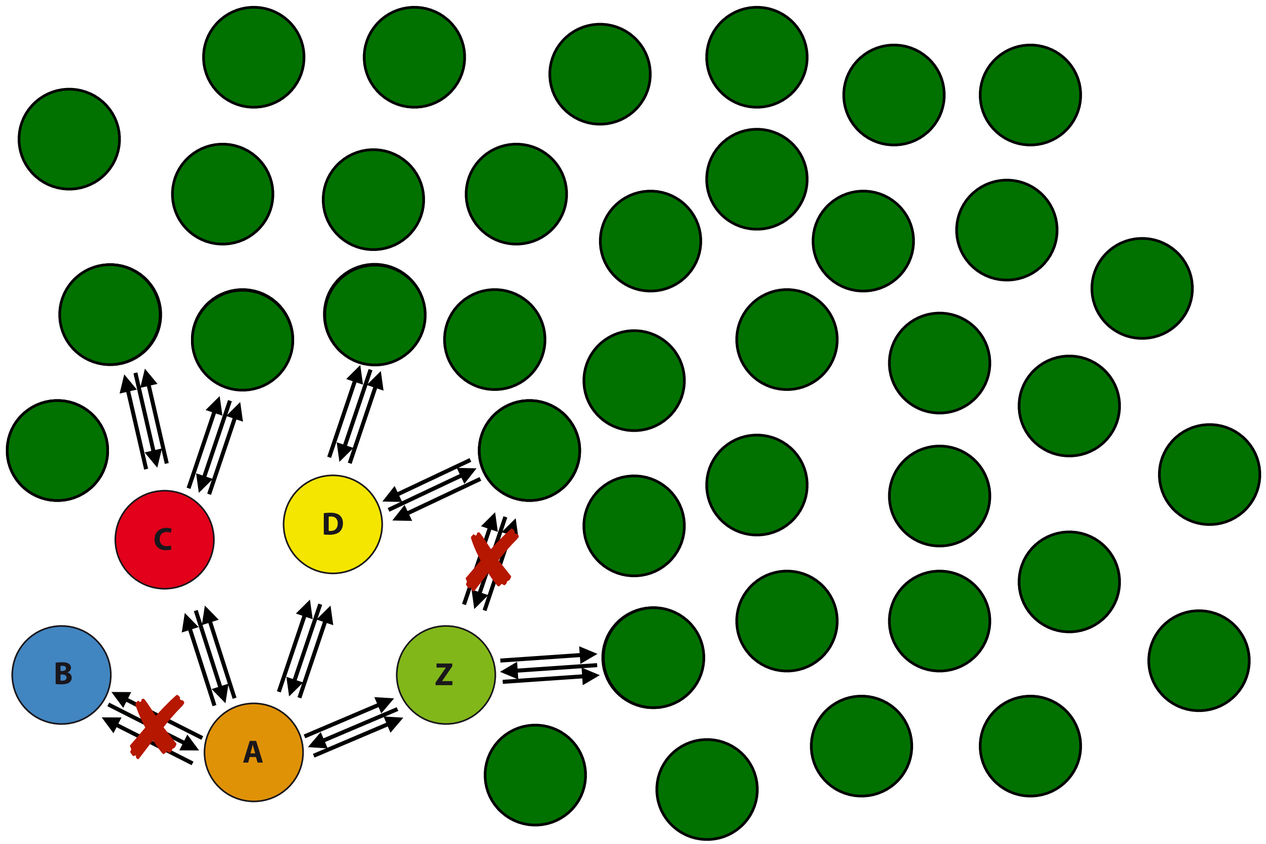
Information propagation (2/2)



Information propagation (2/2)

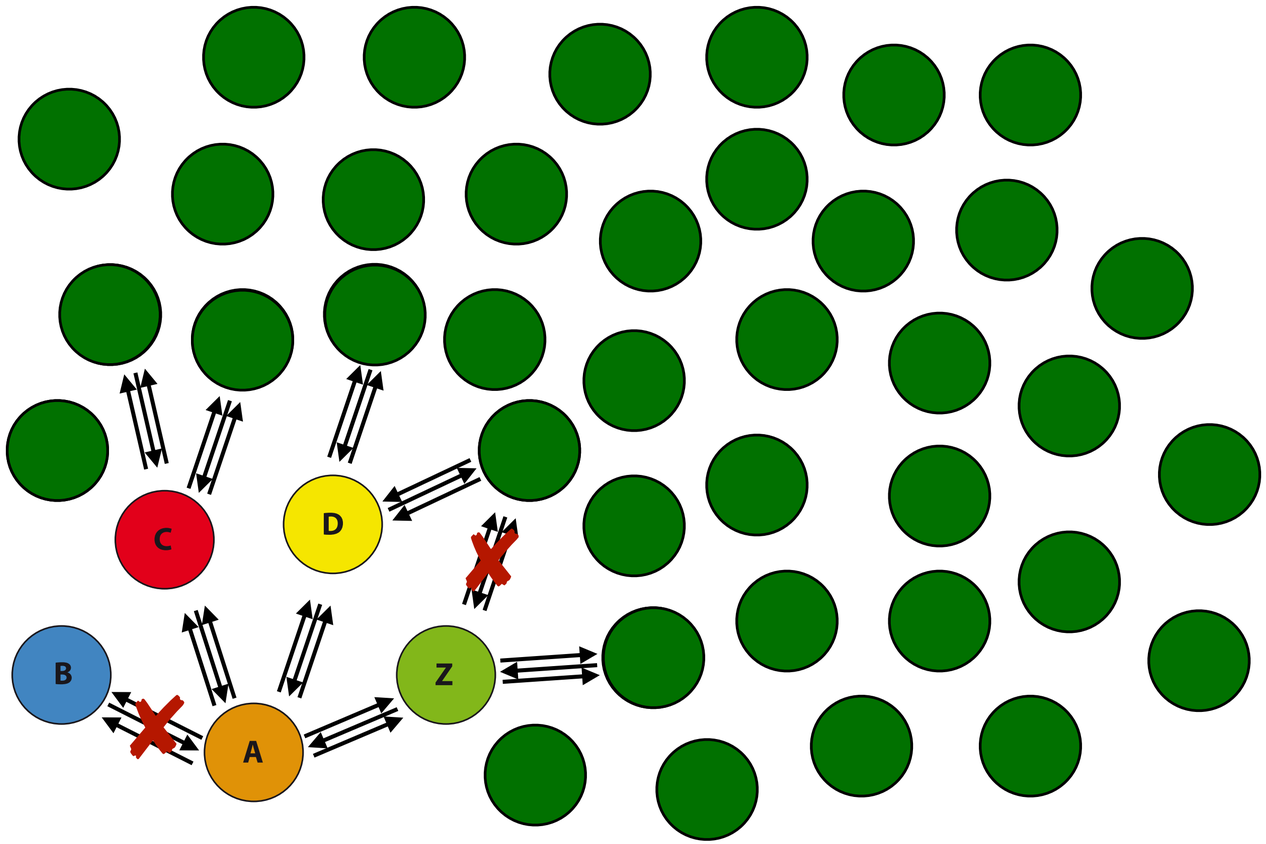


Information propagation (2/2)



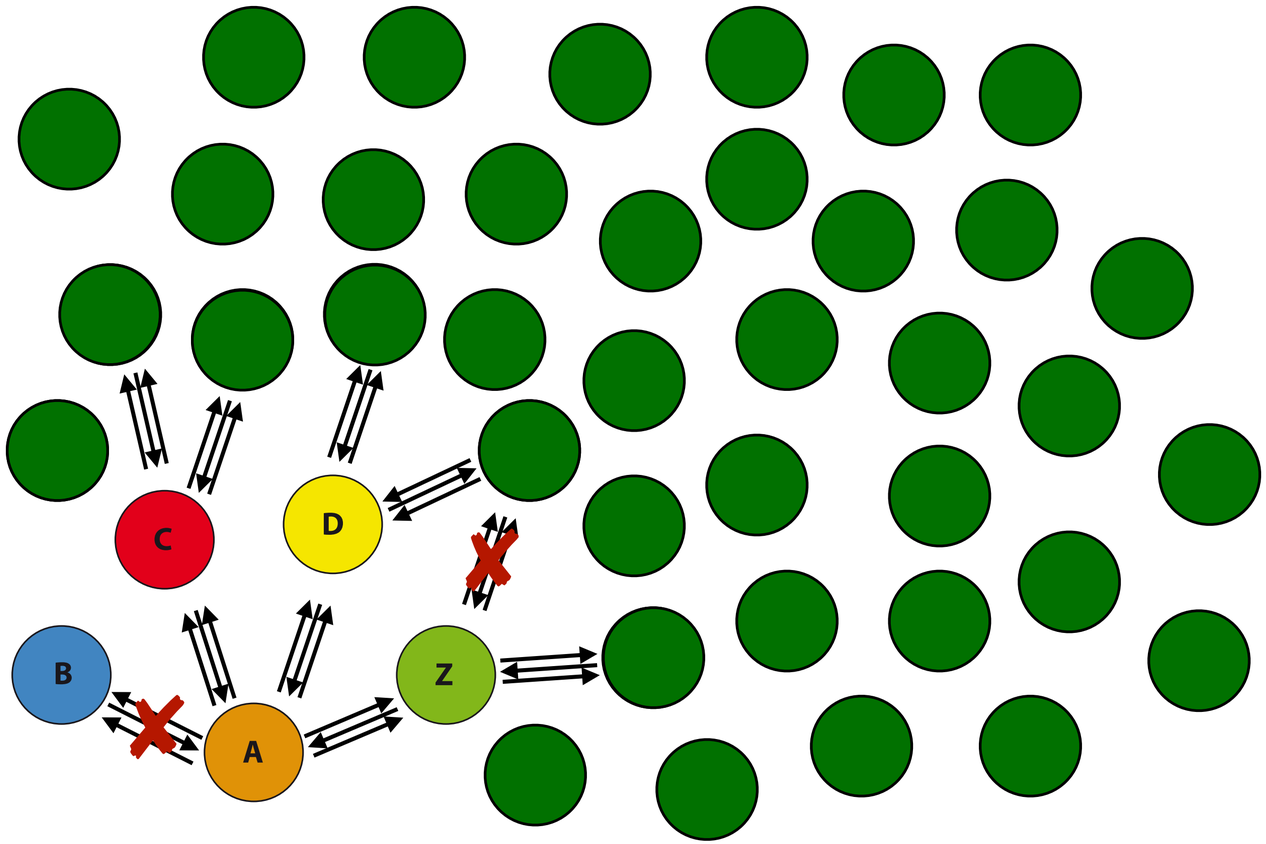
- And so on and so forth until all the nodes are reached

Information propagation (2/2)



- And so on and so forth until all the nodes are reached
- Recall that a node will reject a transaction if it has already learnt about it from any of its neighbors

Information propagation (2/2)



- And so on and so forth until all the nodes are reached
- Recall that a node will reject a transaction if it has already learnt about it from any of its neighbors
- The same procedure applies for blocks

Implications

- The bigger the network the more it takes for an item to propagate (**this can be counterintuitive**)

- Long propagation times (**for blocks**) imply bigger fork rate (bigger likelihood of forking) REPHRASE

-

Implications

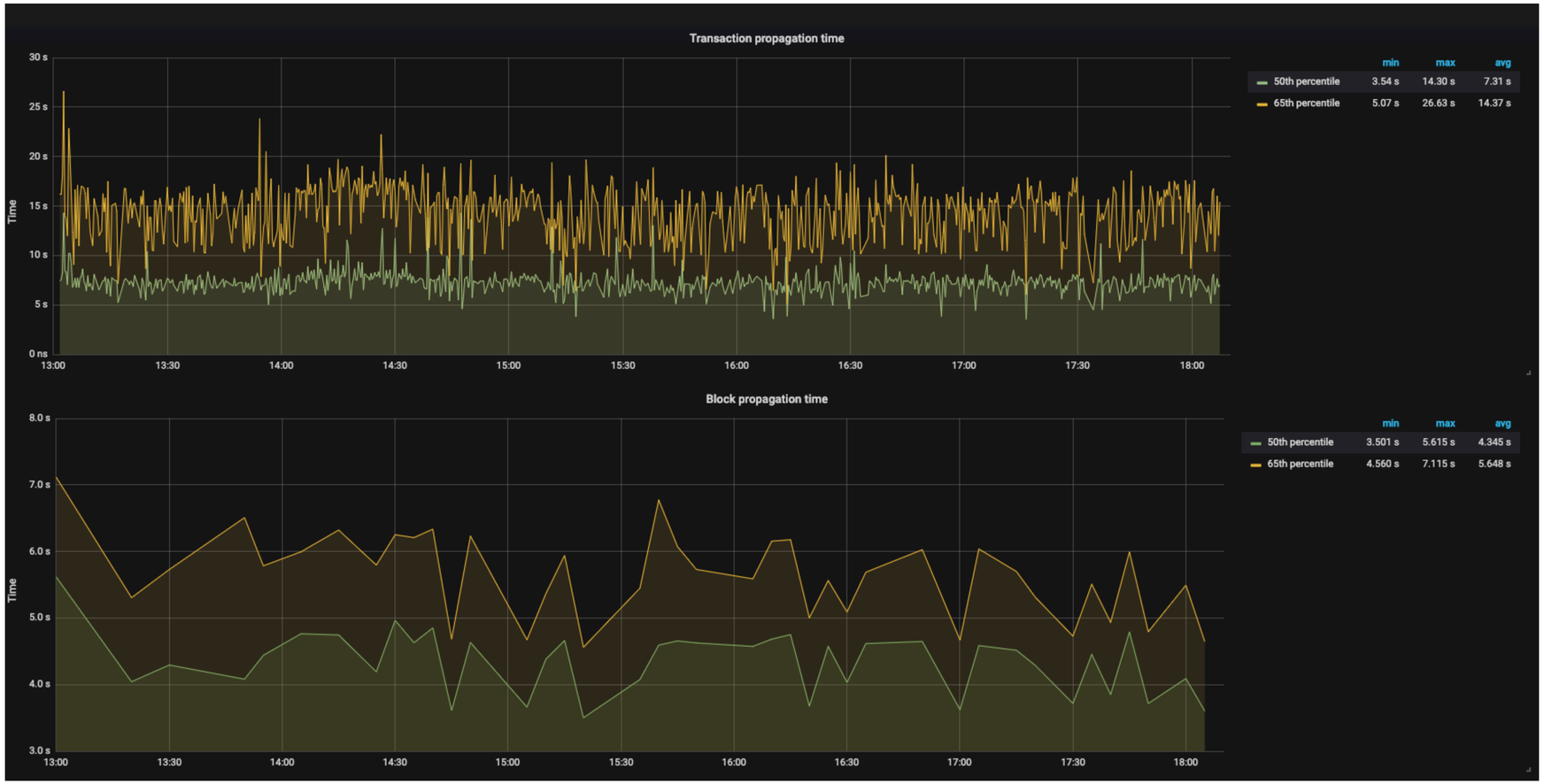
- The bigger the network the more it takes for an item to propagate (**this can be counterintuitive**)

- Long propagation times (**for blocks**) imply bigger fork rate (bigger likelihood of forking) REPHRASE

-

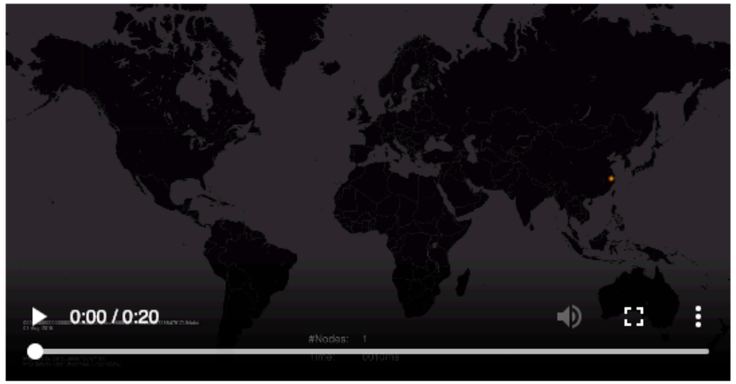
Christian Decker and Roger Wattenhofer
Information propagation in the Bitcoin network
https://ieeexplore.ieee.org/document/6688704

Bitcoin testnet data propagation times

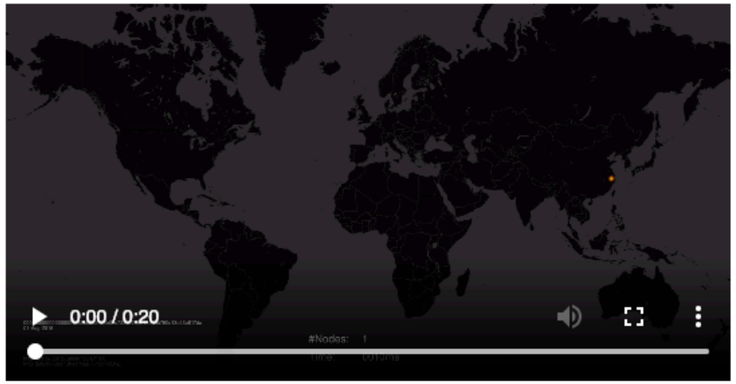


source: charts.satoshi.uab.cat

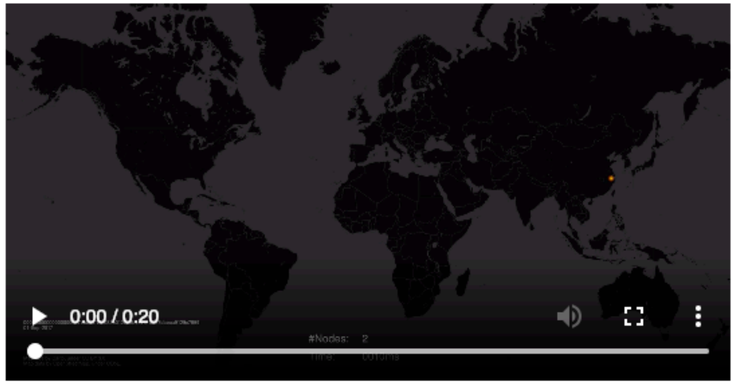
More about propagation times



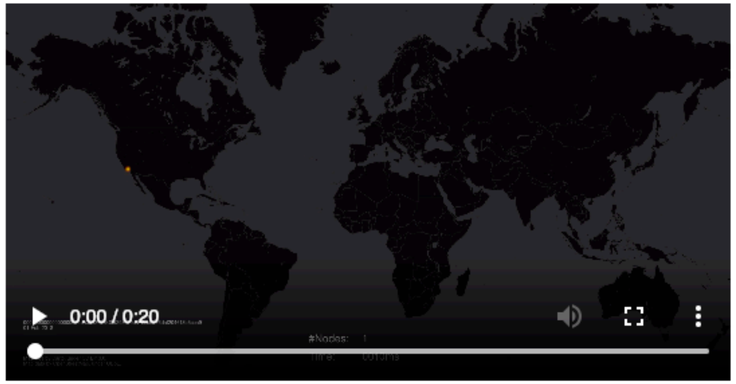
Block propagation | 01.08.2015



Block propagation | 01.08.2016



Block propagation | 01.09.2017



Block propagation | 01.02.2018

source: https://dsn.tm.kit.edu/bitcoin/videos.html

Propagation delays (1/2)

Propagation delays (1/2)

- How can blocks propagate faster than transactions if the former are bigger than the later?

Propagation delays (1/2)

- **How can blocks propagate faster than transactions if the former are bigger than the later?**
    - Transactions are accumulated in buffers and forwarded in batches to break the link between first relayer and origin of a transaction

Propagation delays (1/2)

- **How can blocks propagate faster than transactions if the former are bigger than the later?**
  - Transactions are accumulated in buffers and forwarded in batches to break the link between first relayer and origin of a transaction
  - The propagation of blocks is not delayed, in order to reach full network coverage as soon as possible

Propagation delays (2/2)

Propagation delays (2/2)

- But blocks are way bigger than transactions, how can they be propagated so fast!?

Propagation delays (2/2)

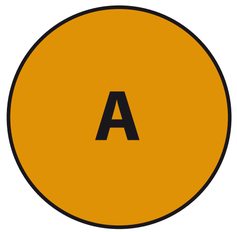
- **But blocks are way bigger than transactions, how can they be propagated so fast!?**
    - Fast relay networks on top of Bitcoin exists (Falcon, FIBRE, etc) to enhance the propagation time of blocks

Propagation delays (2/2)

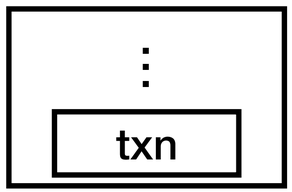
- **But blocks are way bigger than transactions, how can they be propagated so fast!?**
    - Fast relay networks on top of Bitcoin exists (Falcon, FIBRE, etc) to enhance the propagation time of blocks
    - Miners use such networks to ensure minimal propagation times as well as ensure being mining on top of the most recent block

0-conf transactions and double-spending

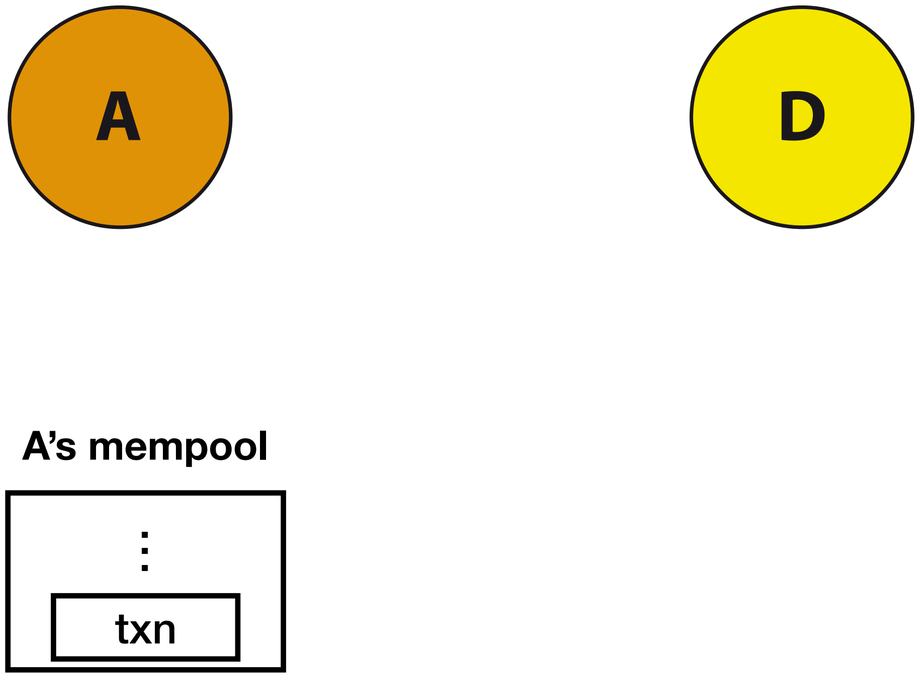
Confirmed transactions



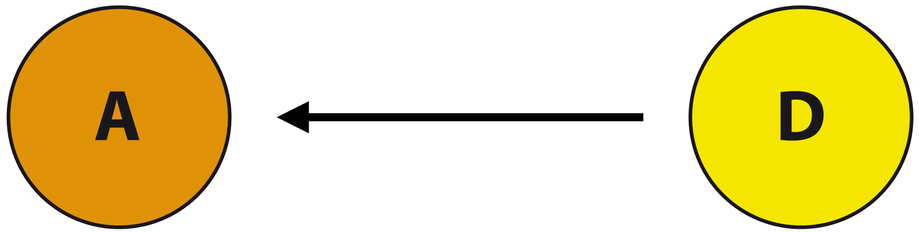
A's mempool



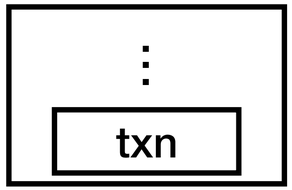
Confirmed transactions



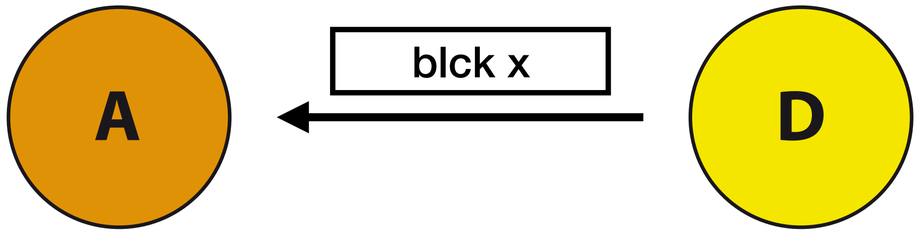
Confirmed transactions



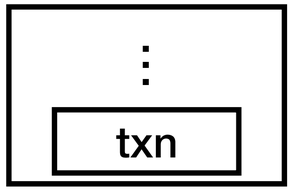
A's mempool



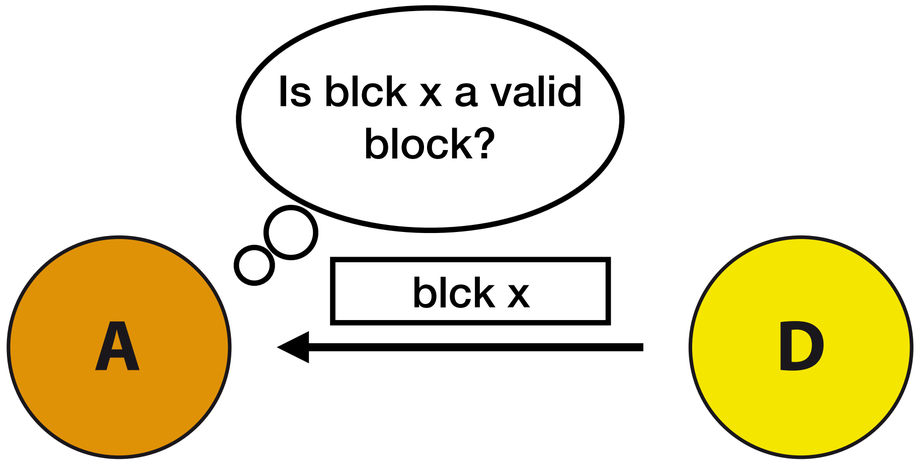
Confirmed transactions



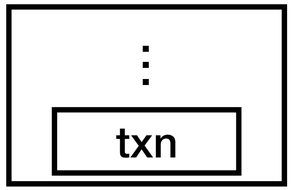
A's mempool



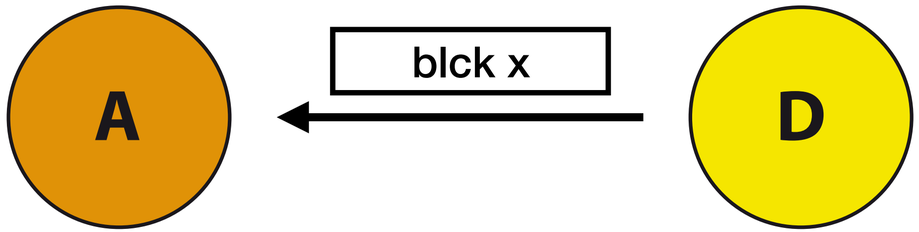
Confirmed transactions



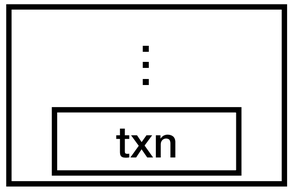
A's mempool



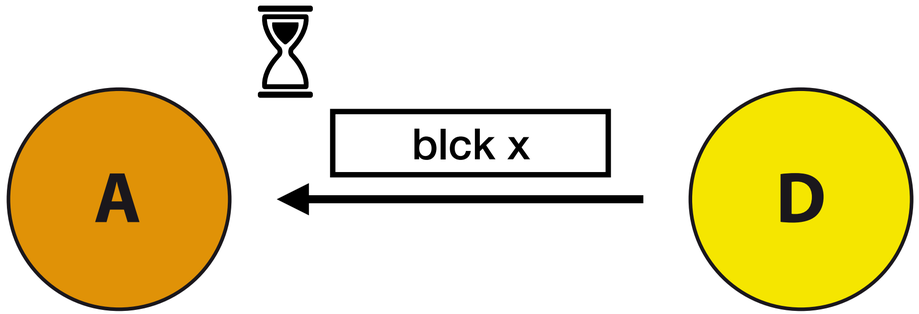
Confirmed transactions



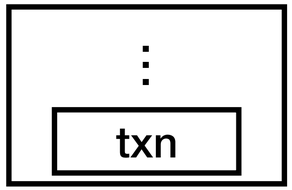
A's mempool



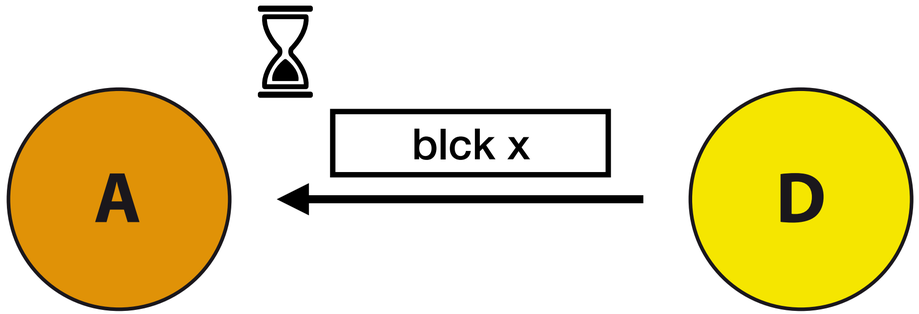
Confirmed transactions



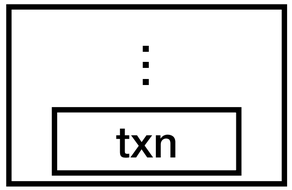
A's mempool



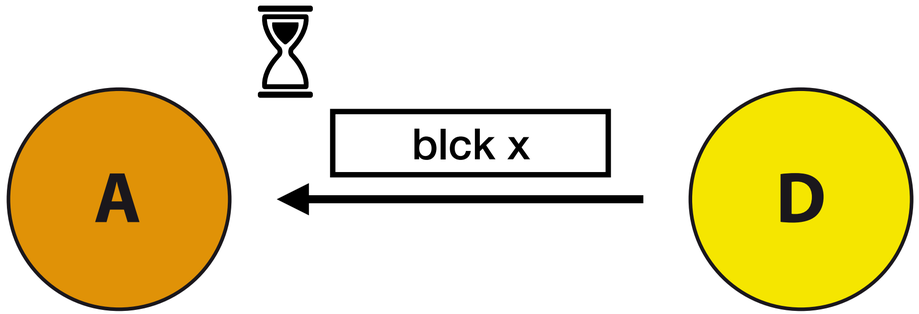
Confirmed transactions



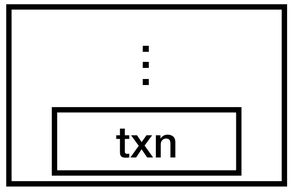
A's mempool



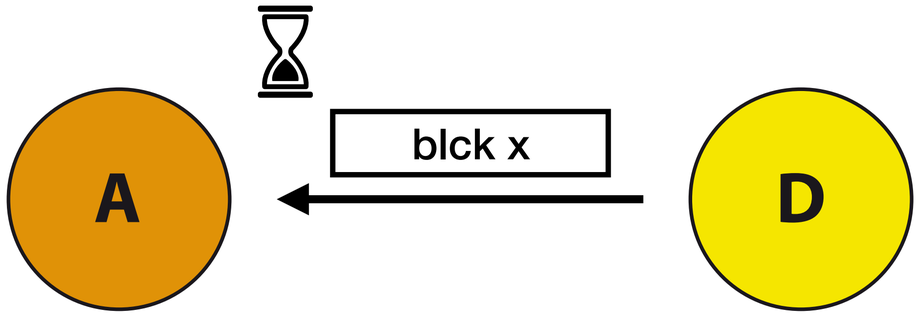
Confirmed transactions



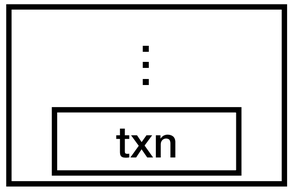
A's mempool



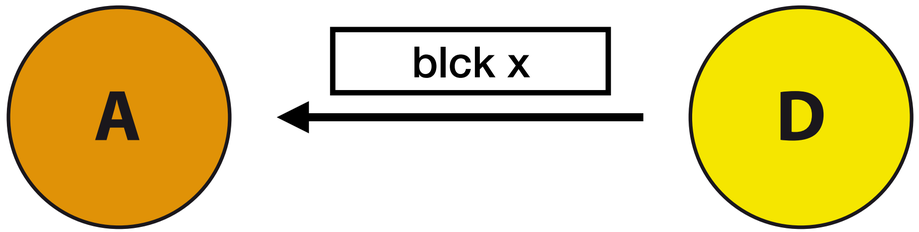
Confirmed transactions



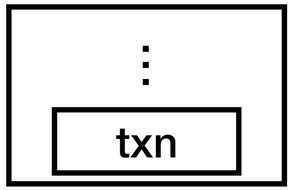
A's mempool



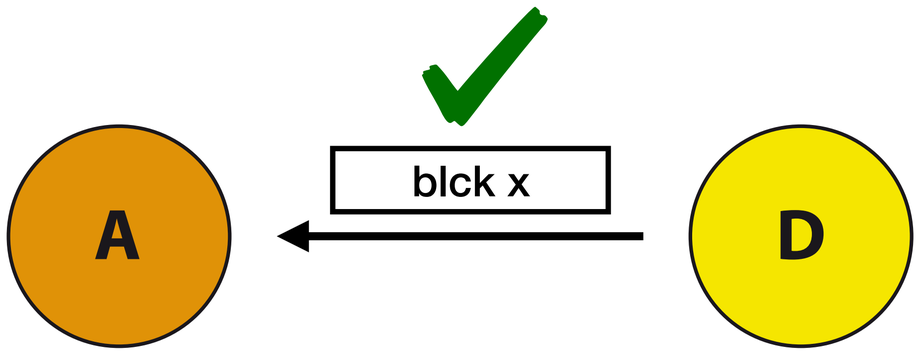
Confirmed transactions



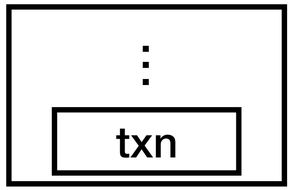
A's mempool



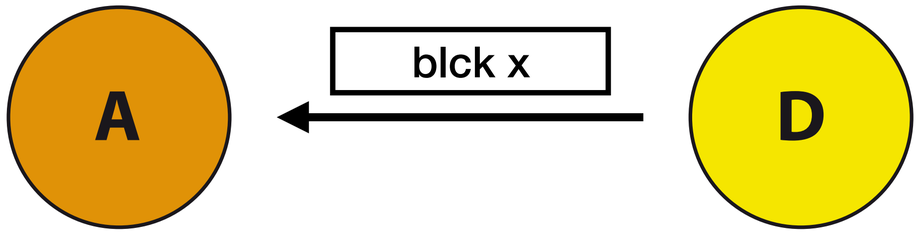
Confirmed transactions



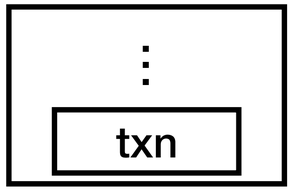
A's mempool



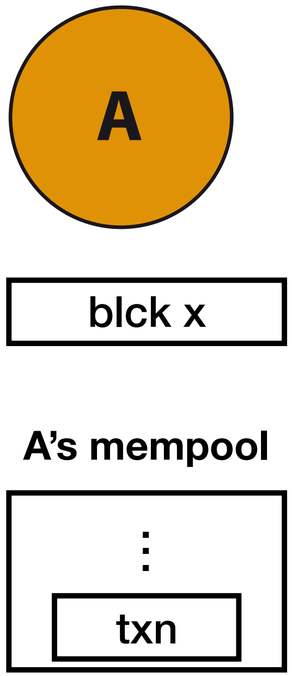
Confirmed transactions



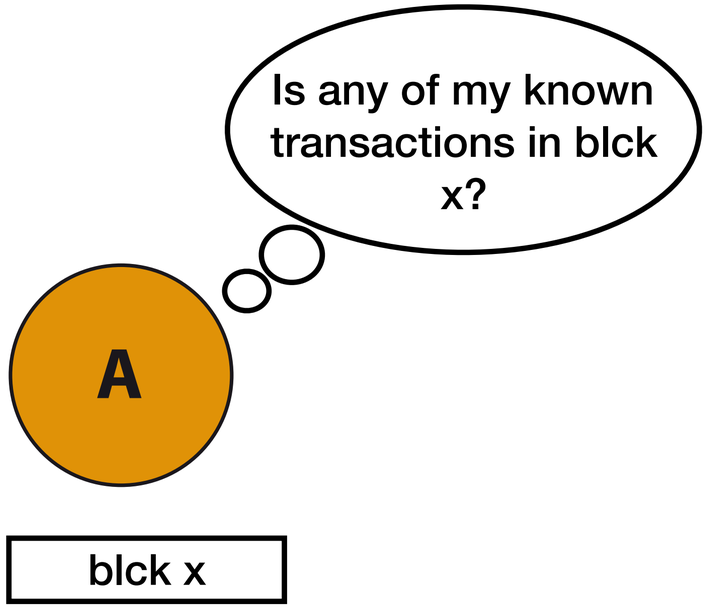
A's mempool



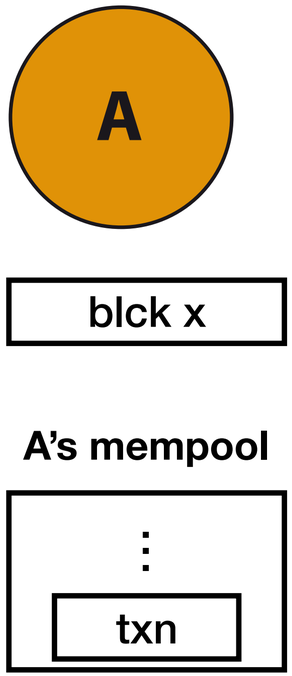
Confirmed transactions



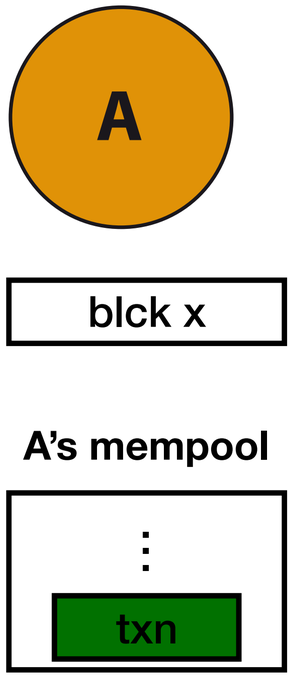
Confirmed transactions



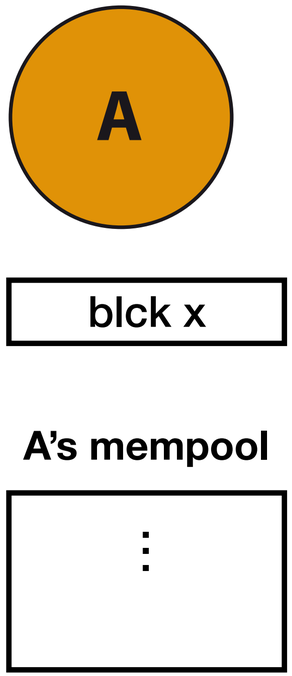
Confirmed transactions



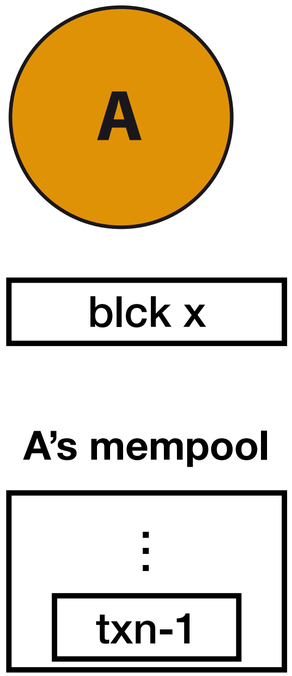
Confirmed transactions



Confirmed transactions



Confirmed transactions



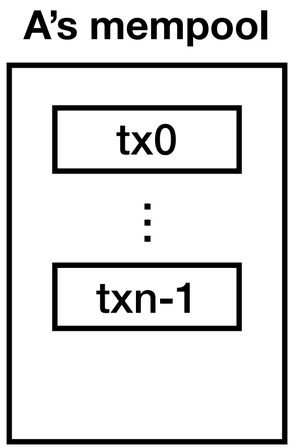
0-confirmation transactions (1/2)

- **0-conf** transactions / **unconfirmed** transactions are those that are not part of the blockchain (**they are stored in the mempool**)

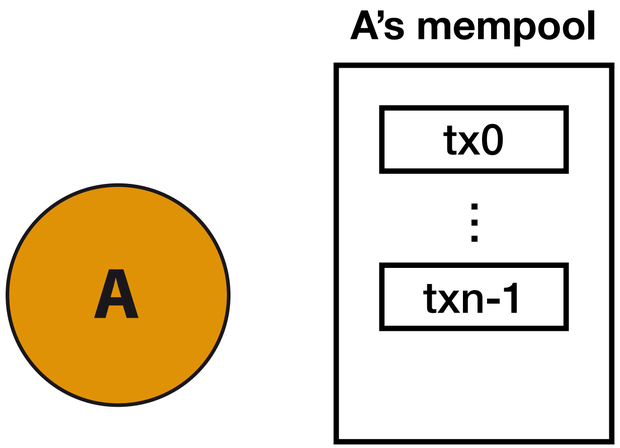
- 0-conf transactions are not covered by the double-spending protection offered by the blockchain (they are not part of it)

- Different nodes can have conflicting version of the **“same transaction”**

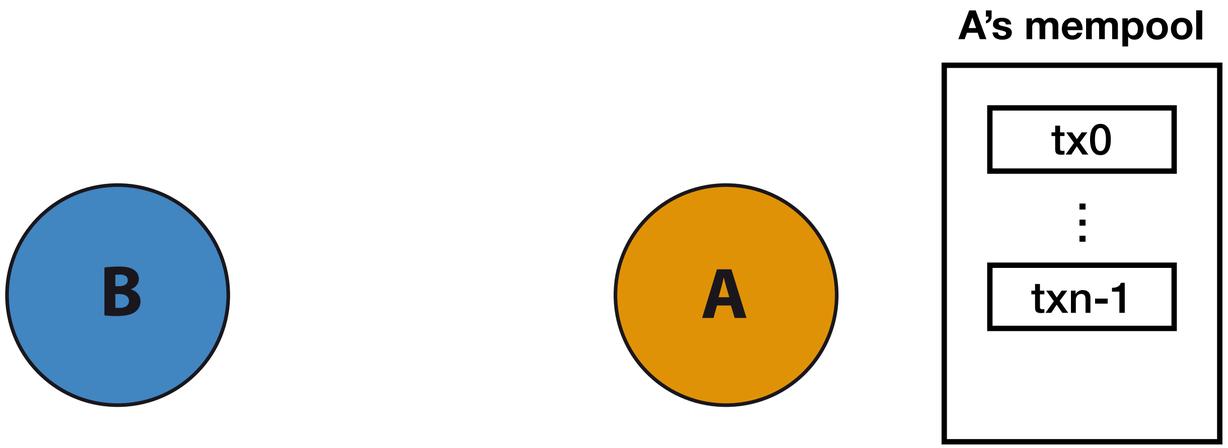
0-confirmation transactions (2/2)



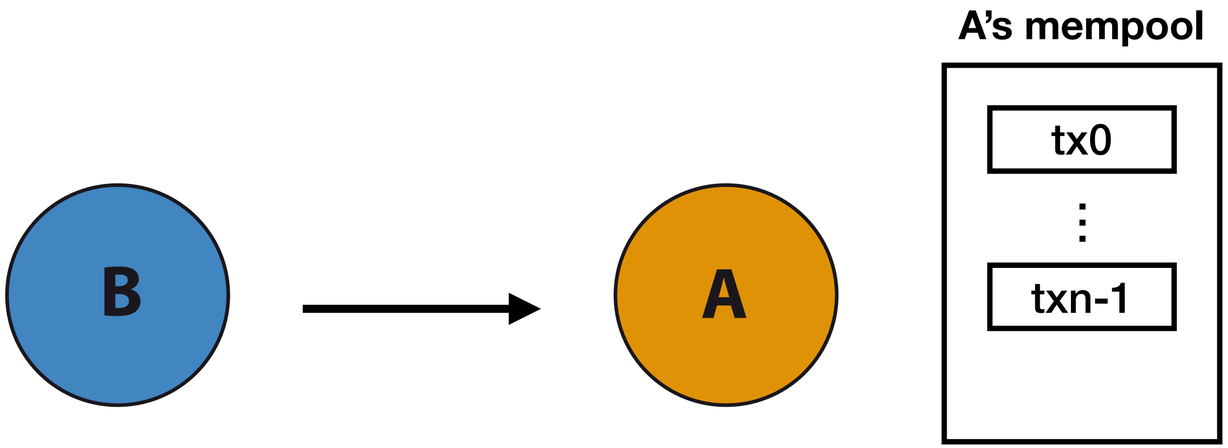
0-confirmation transactions (2/2)



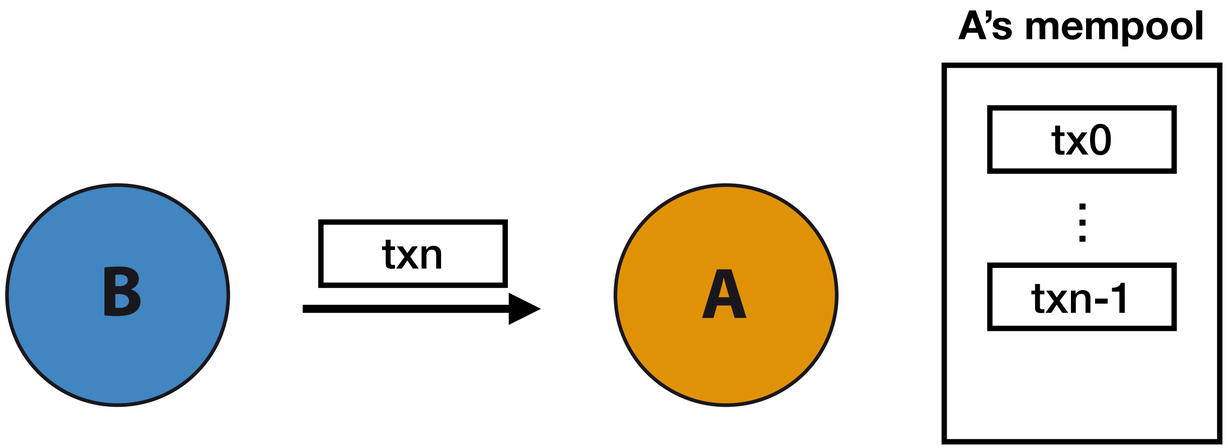
0-confirmation transactions (2/2)



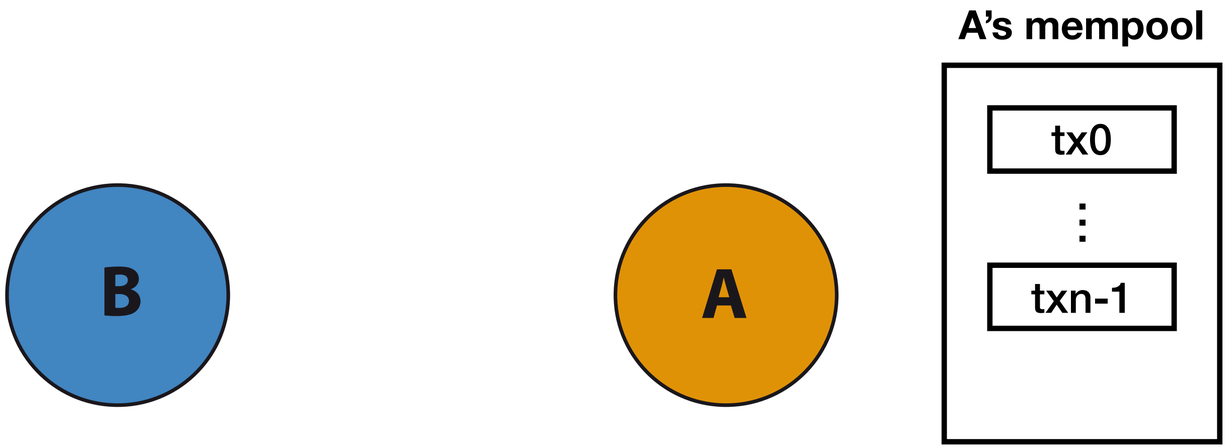
0-confirmation transactions (2/2)



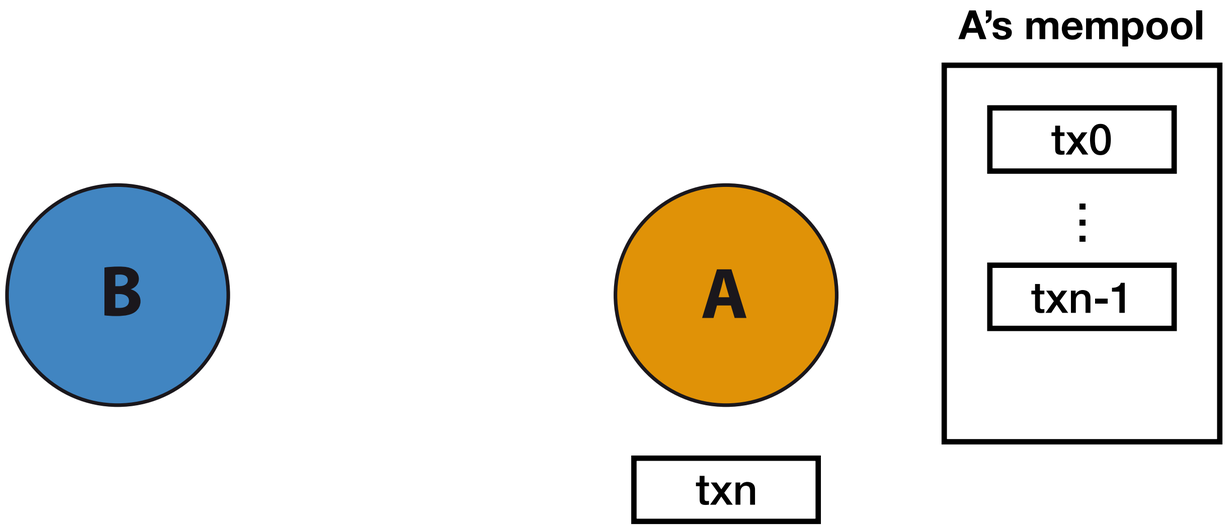
0-confirmation transactions (2/2)



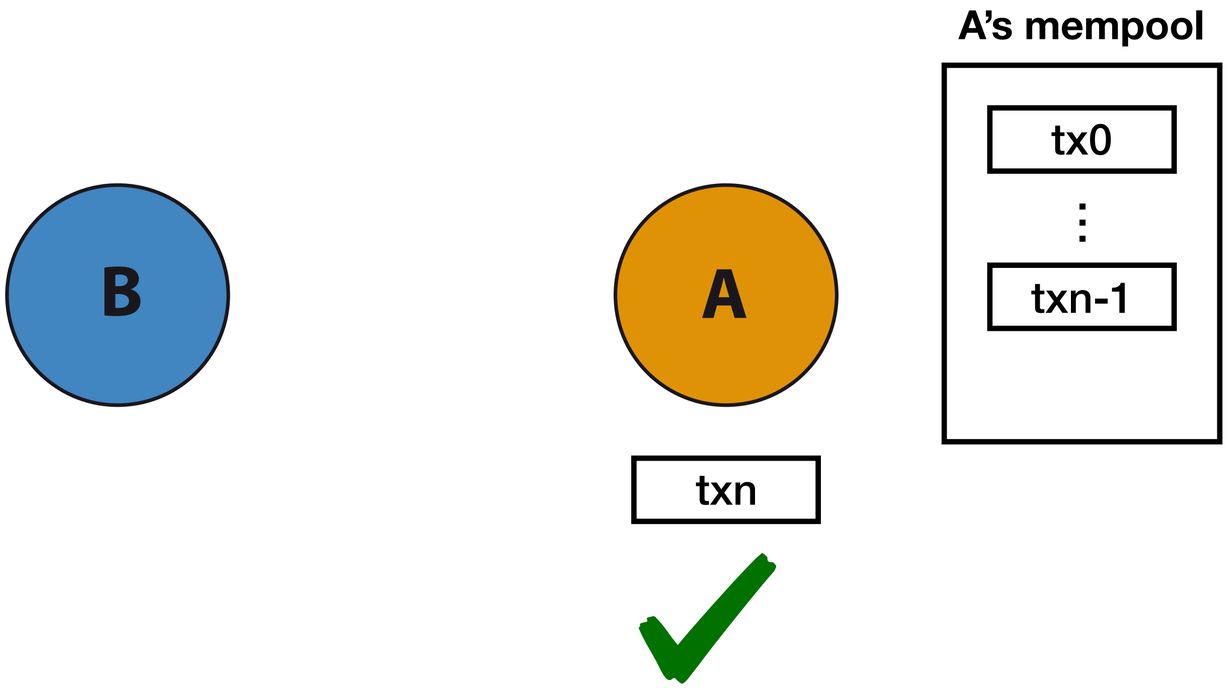
0-confirmation transactions (2/2)



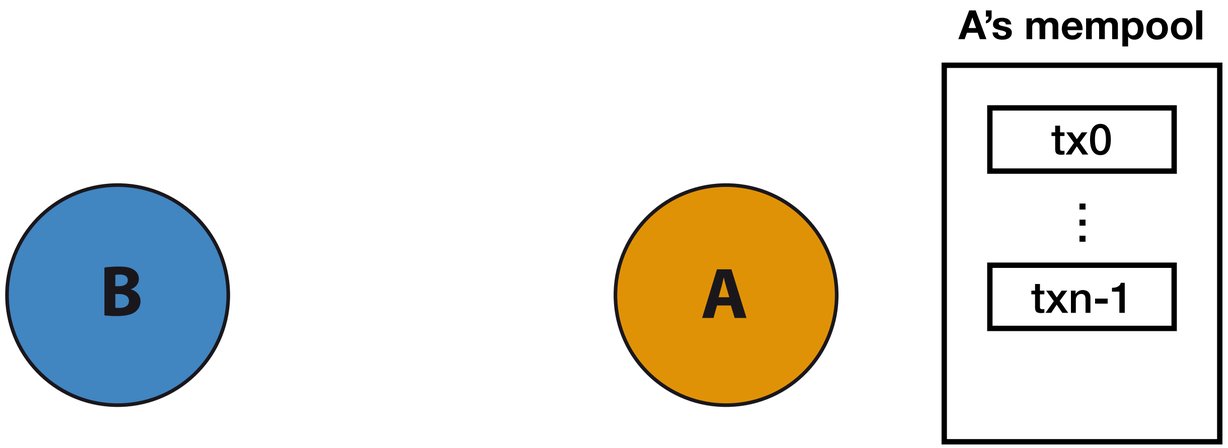
0-confirmation transactions (2/2)



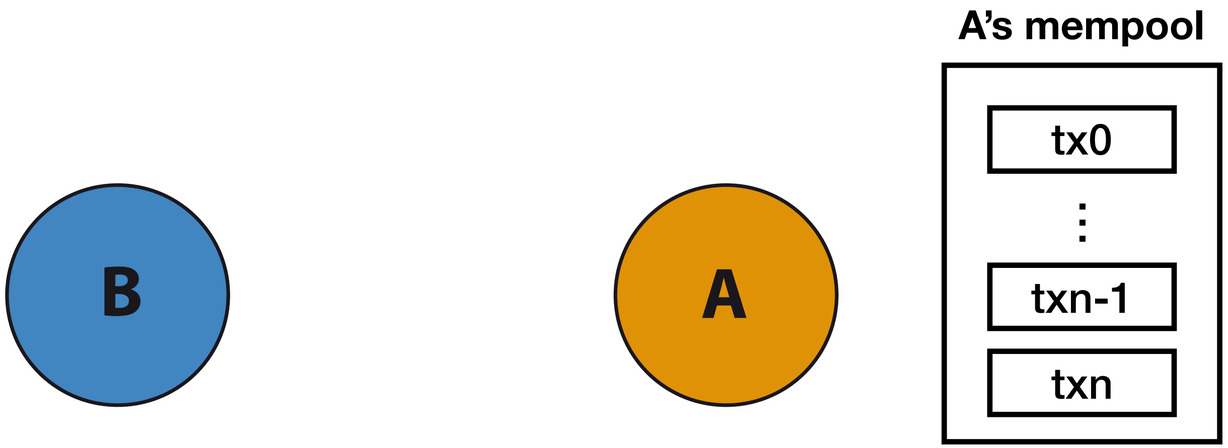
0-confirmation transactions (2/2)



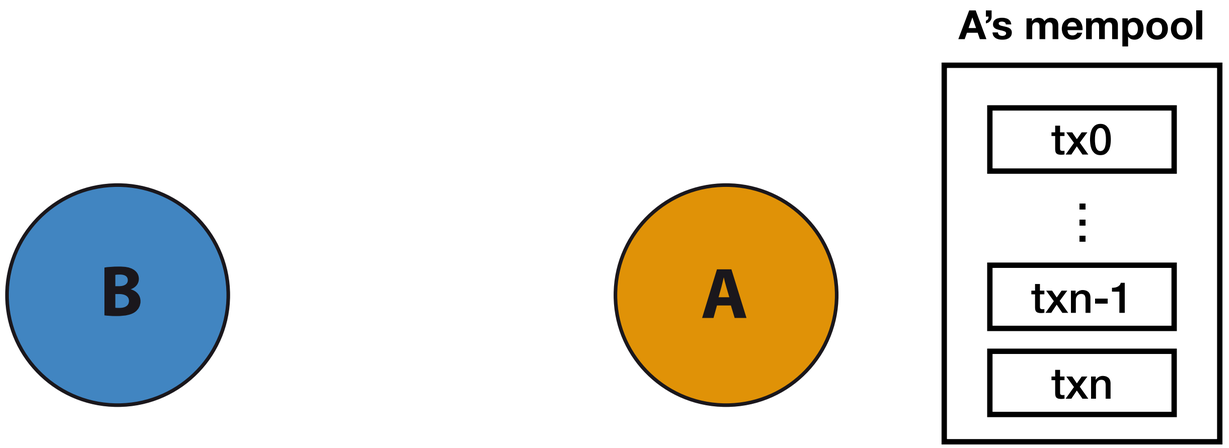
0-confirmation transactions (2/2)



0-confirmation transactions (2/2)



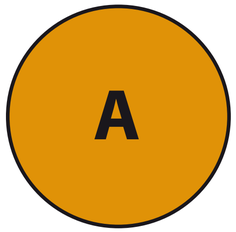
0-confirmation transactions (2/2)



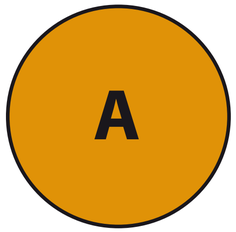
Transactions sitting in memory (mempool) are unconfirmed and should not be trusted

Double-spending transactions (1/2)

Double-spending transactions (1/2)

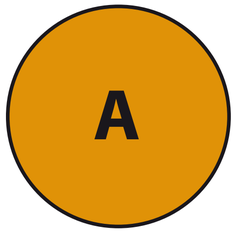


Double-spending transactions (1/2)



id = 4F3…ED

Double-spending transactions (1/2)



id = 4F3…ED

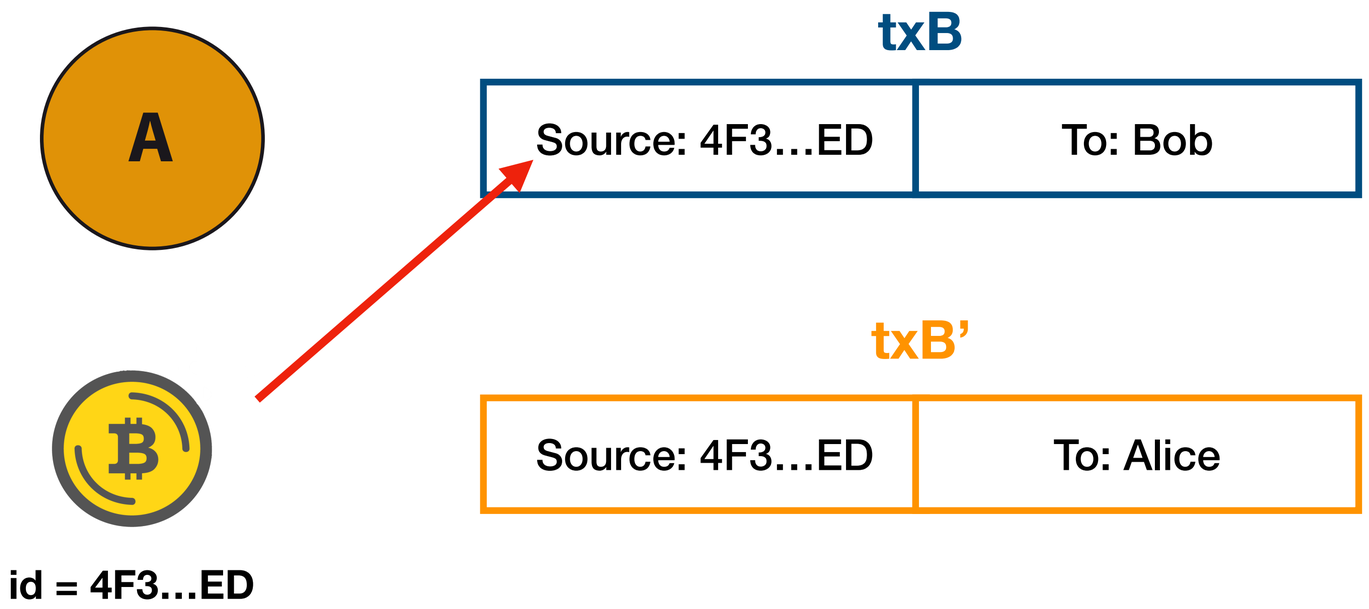
txB

Source: 4F3…ED	To: Bob

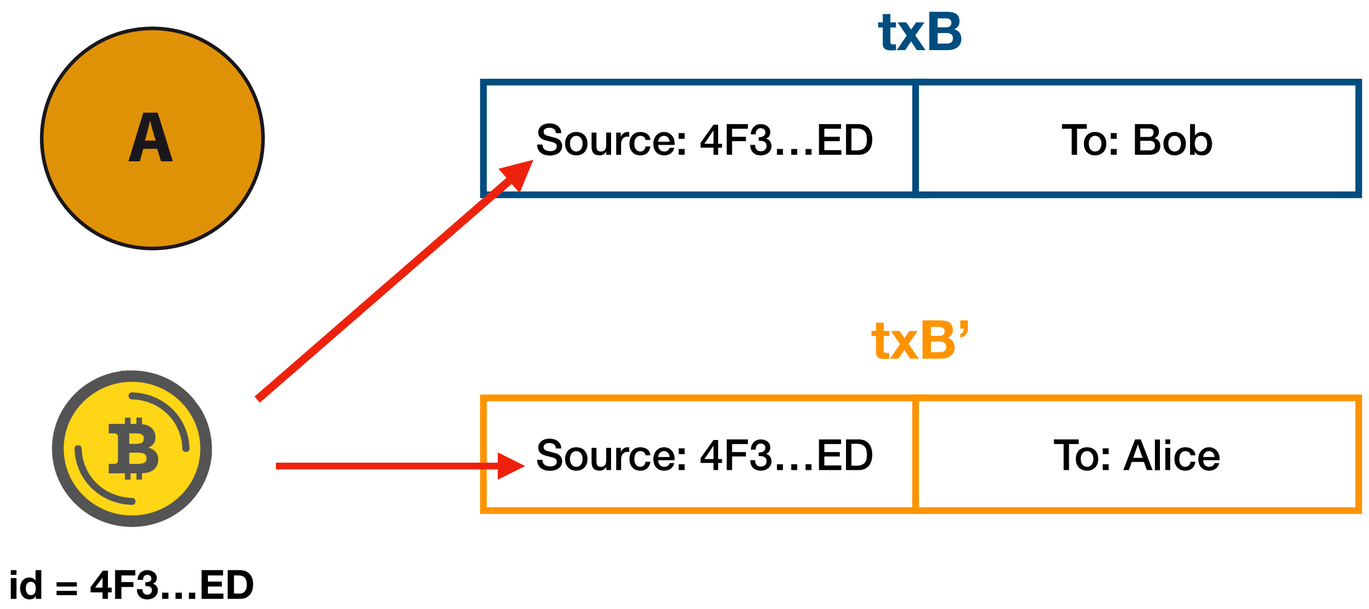
txB’

Source: 4F3…ED	To: Alice

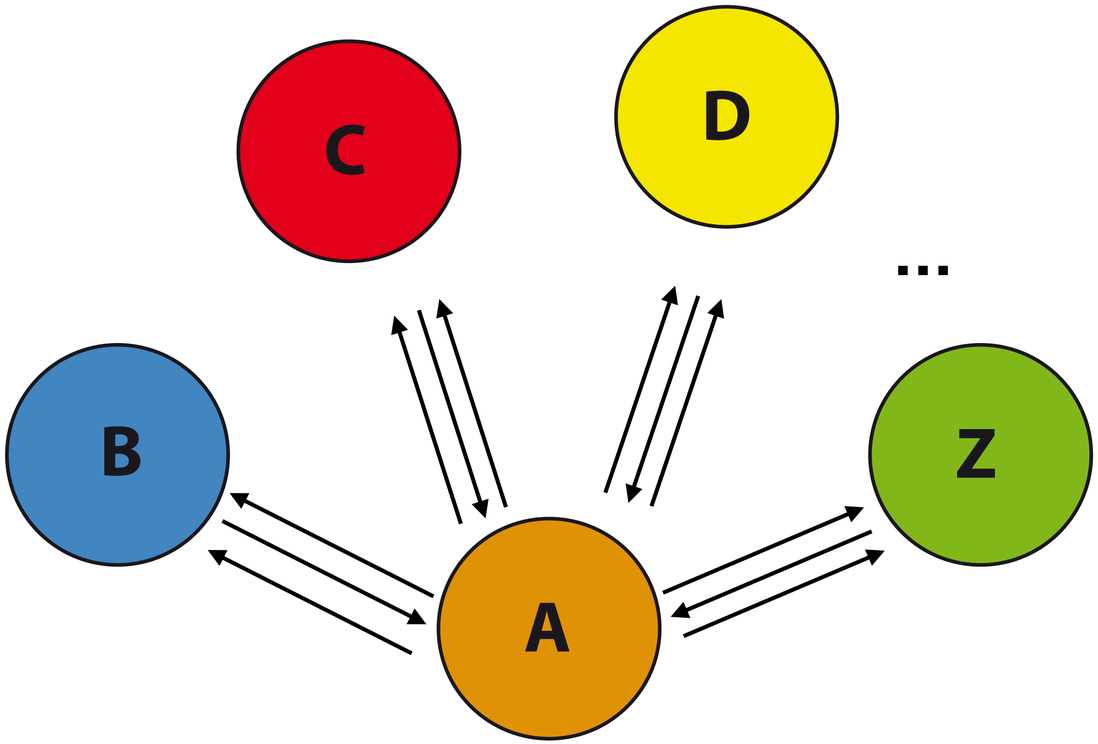
Double-spending transactions (1/2)



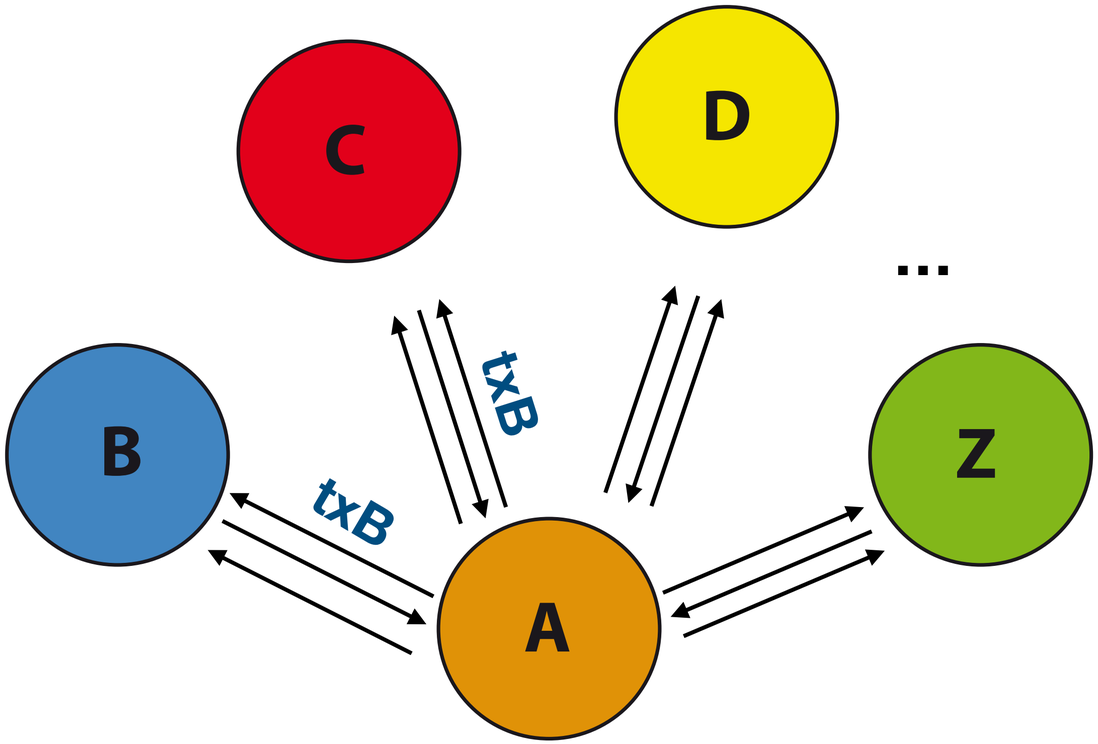
Double-spending transactions (1/2)



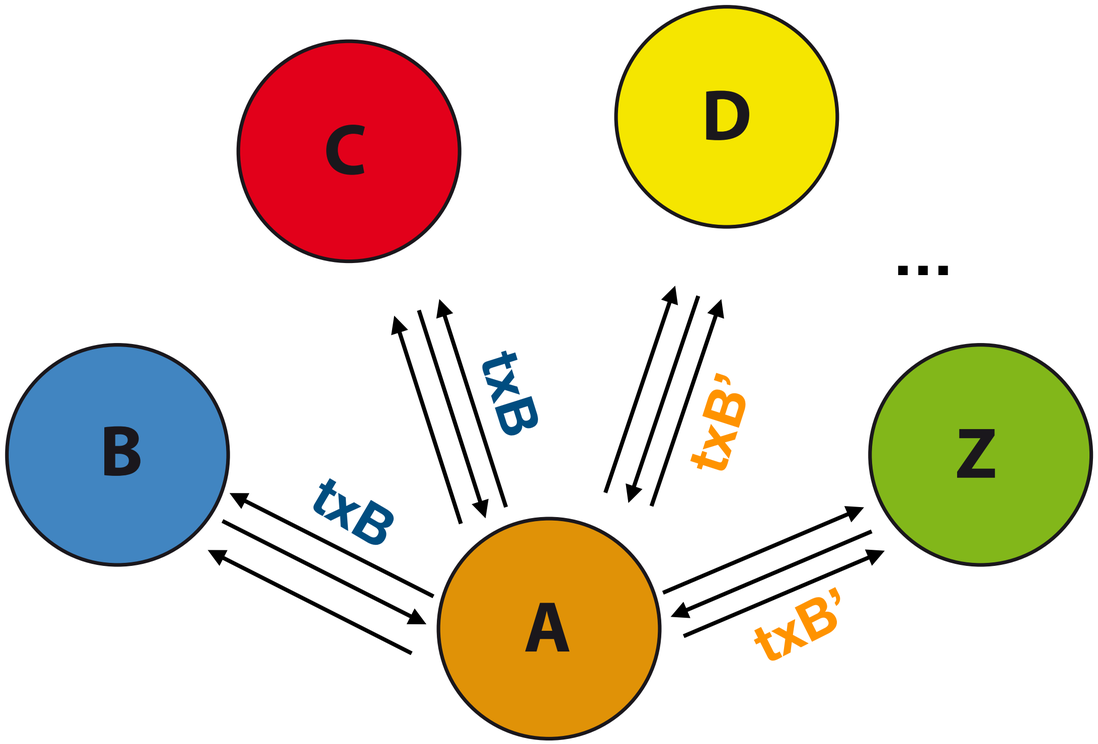
Double-spending transactions (2/2)



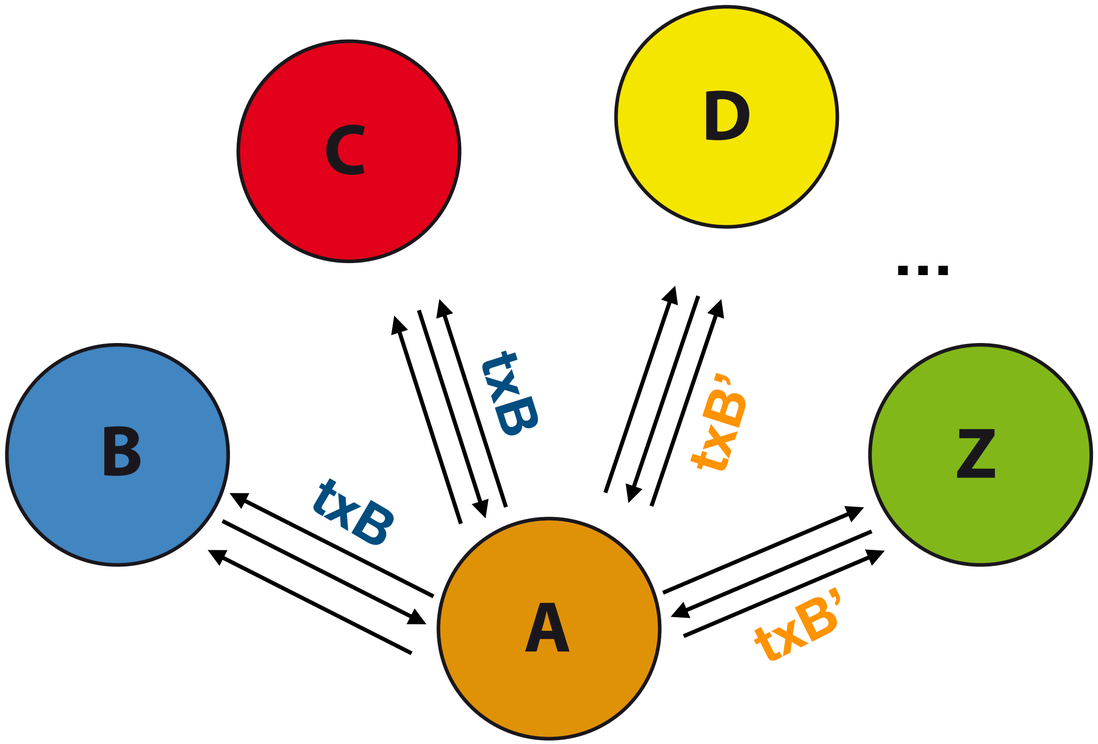
Double-spending transactions (2/2)



Double-spending transactions (2/2)

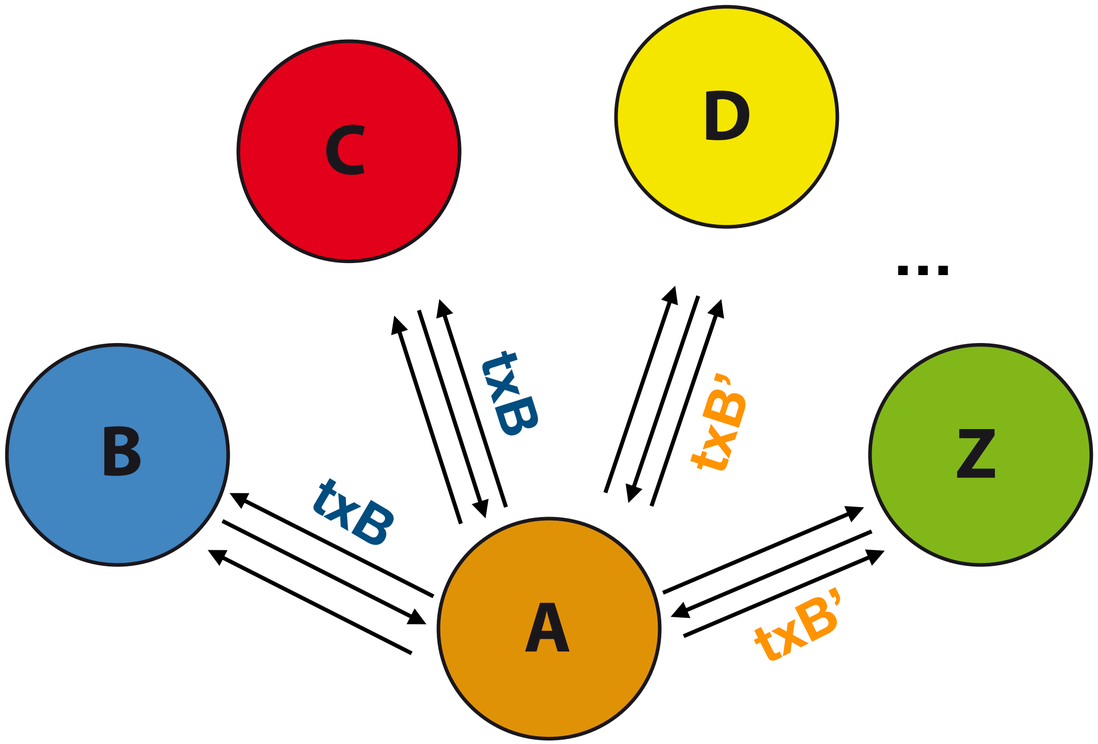


Double-spending transactions (2/2)



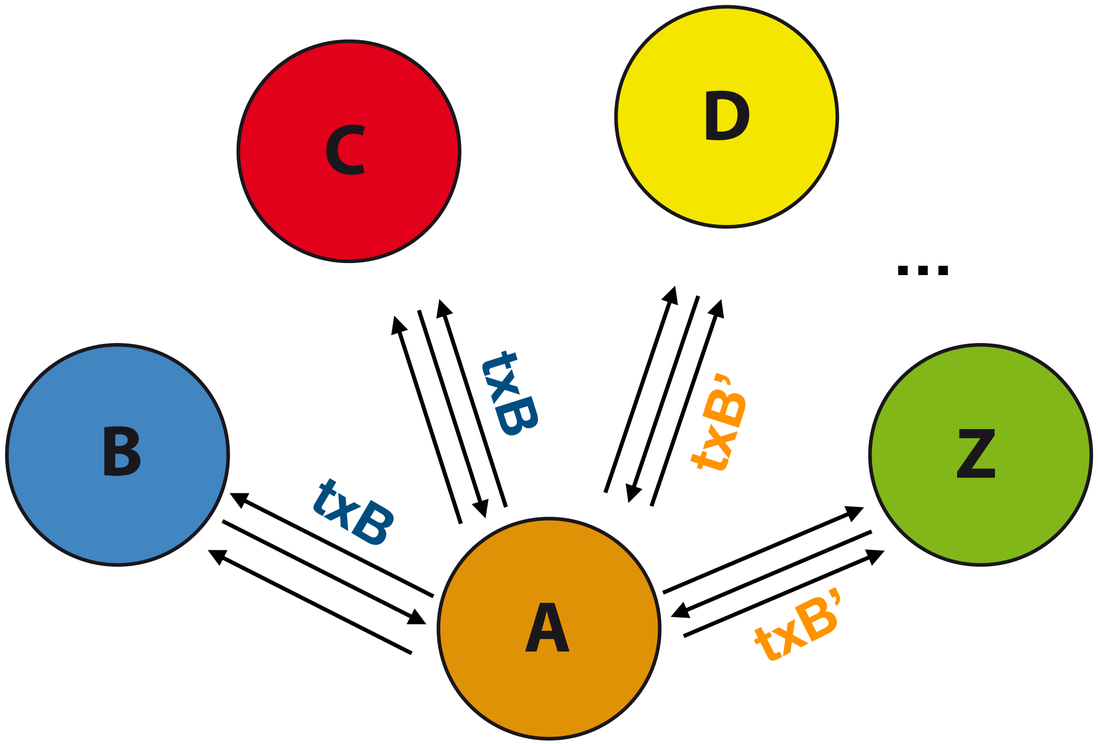
- **0-conf** transactions should not be trusted

Double-spending transactions (2/2)



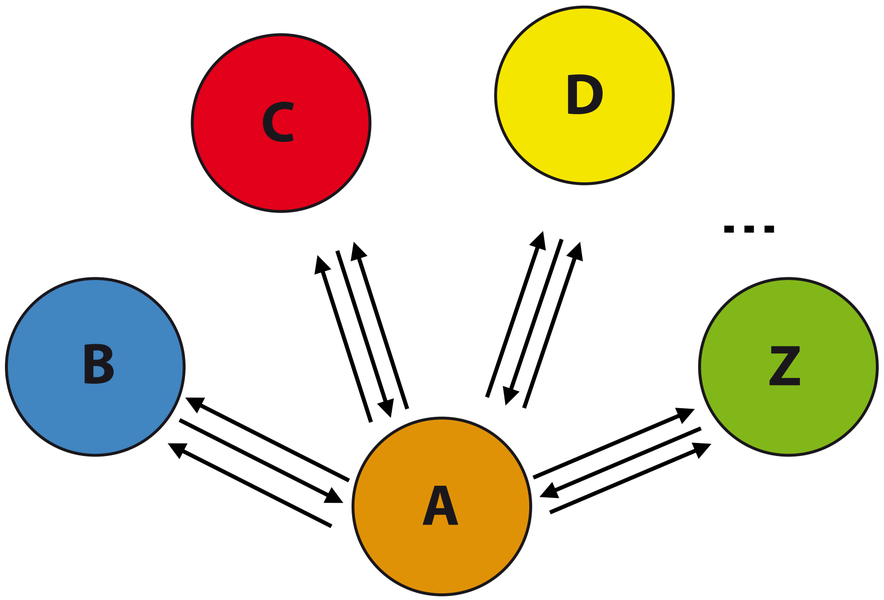
- **0-conf** transactions should not be trusted
- If B accepts txB before it appears in a block he **can be deceived** by A

Double-spending transactions (2/2)

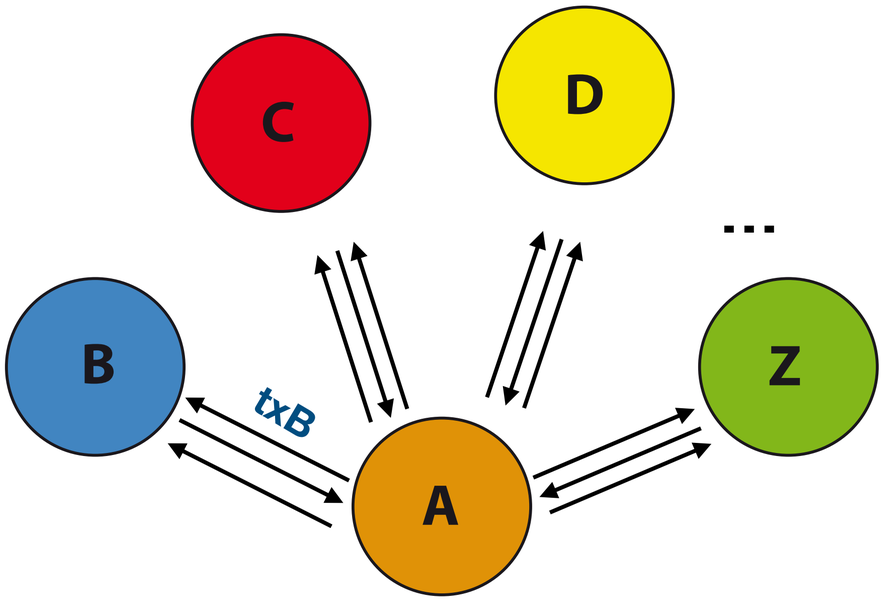


- **0-conf** transactions should not be trusted
- If B accepts txB before it appears in a block he **can be deceived** by A
- The de facto confirmation time is **6 blocks** (5 on top of the one including a certain transaction)

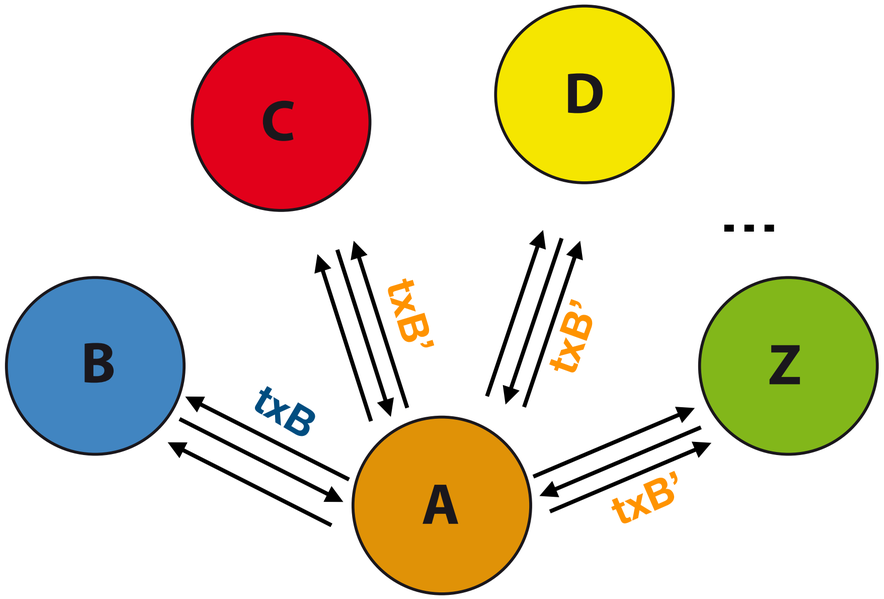
When things go south



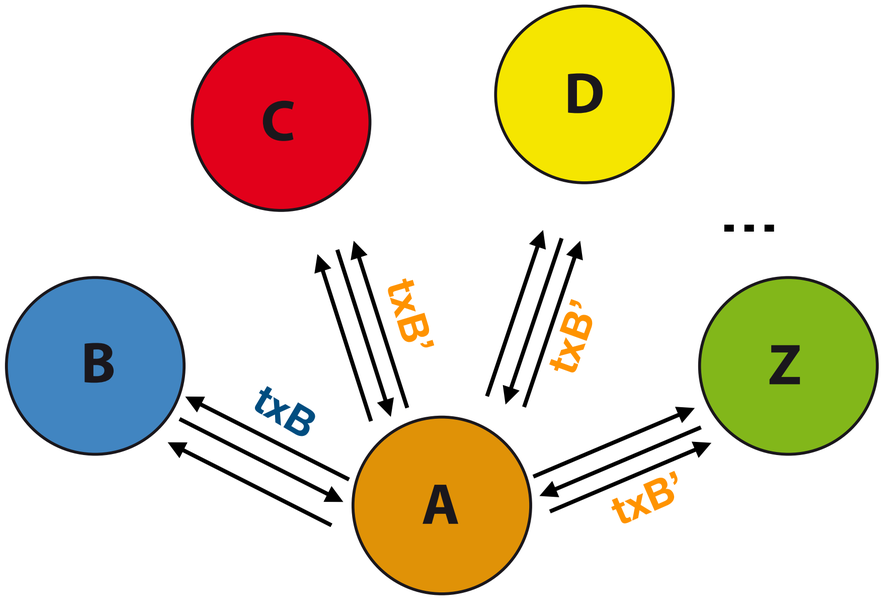
When things go south



When things go south

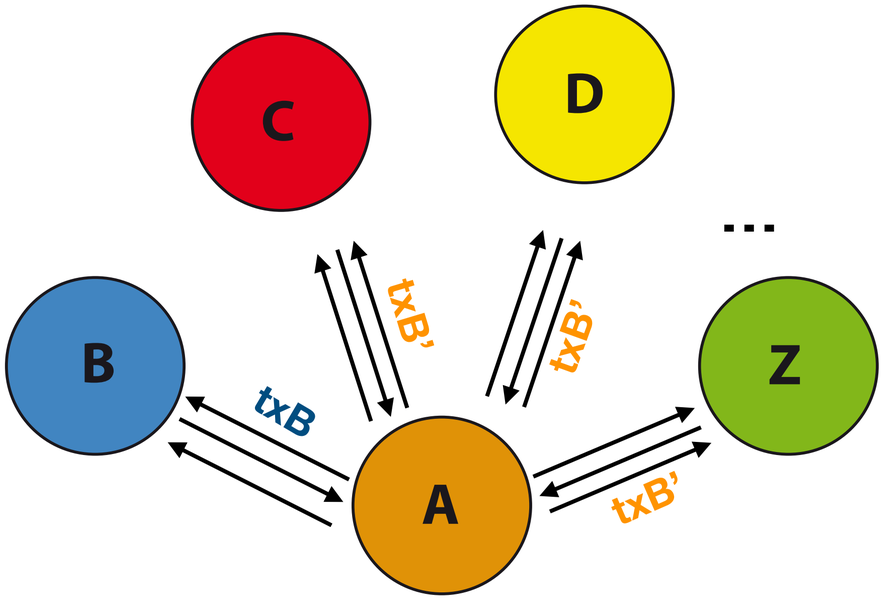


When things go south



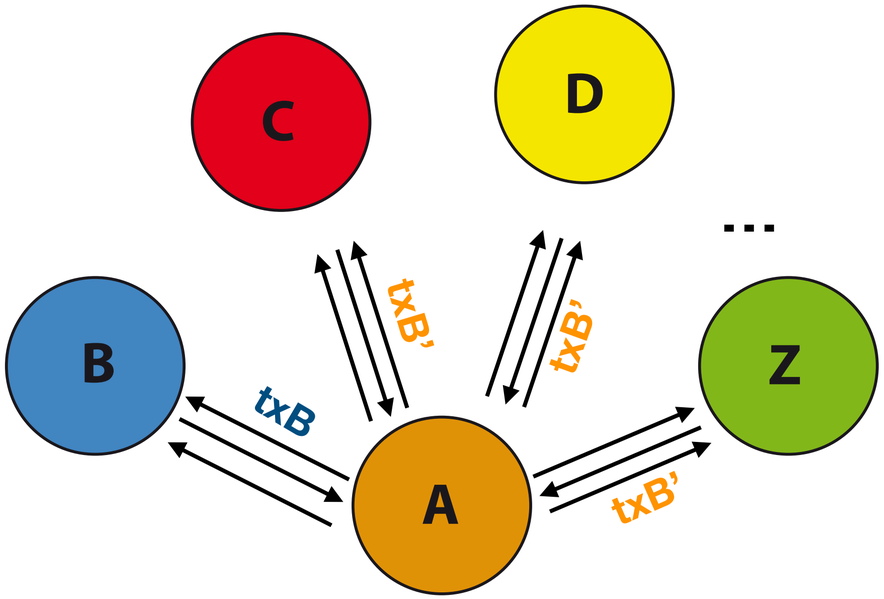
- If A controls the **network view** of B, A can control what B know about the currency

When things go south



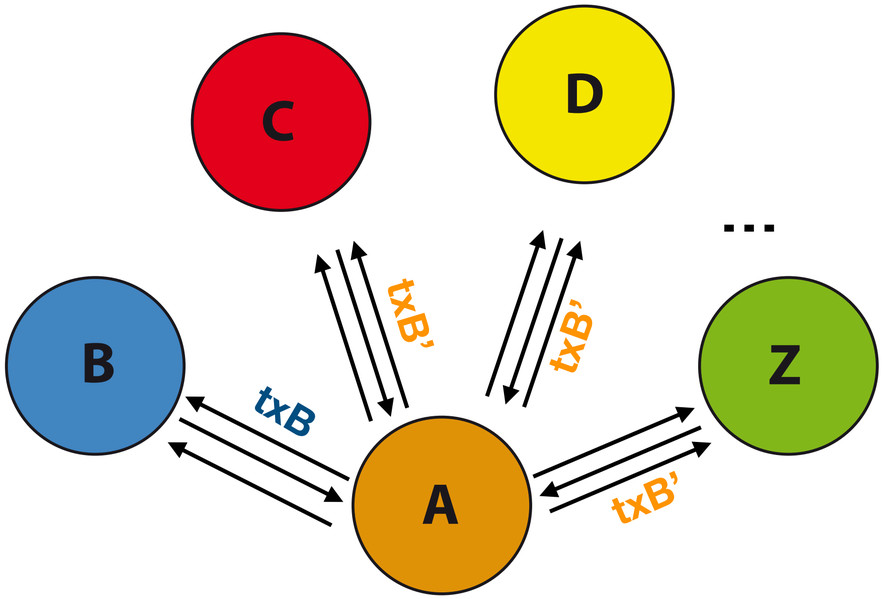
- If A controls the **network view** of B, A can control what B know about the currency
- B is said to be **eclipsed**

When things go south



- If A controls the **network view** of B, A can control what B know about the currency
- B is said to be **eclipsed**
- A will be able to **easily fool** B

When things go south



- If A controls the **network view** of B, A can control what B know about the currency
- B is said to be **eclipsed**
- A will be able to **easily fool** B

Ethan Heilman, Alison Kendler, Aviv Zohar and Sharon Goldberg
Eclipse Attacks on Bitcoin's Peer-to-Peer Network
https://www.usenix.org/node/190891

Network topology

Unknown topology by design

Unknown topology by design

- Peers are chosen pseudorandomly from the peer database of a node in order to become neighbors

Unknown topology by design

- Peers are chosen pseudorandomly from the peer database of a node in order to become neighbors

- Peers can be requested from other peers, but no information about whether the responder is (or has been) a neighbor of any of the provided peers is given

Unknown topology by design

- Peers are chosen pseudorandomly from the peer database of a node in order to become neighbors

- Peers can be requested from other peers, but no information about whether the responder is (or has been) a neighbor of any of the provided peers is given

- The network topology should mimic a random network

Inferring the topology

Inferring the topology

- Does the network really look random?

Inferring the topology

- Does the network really look random?

- How can we known if we don’t know how the topology looks like?

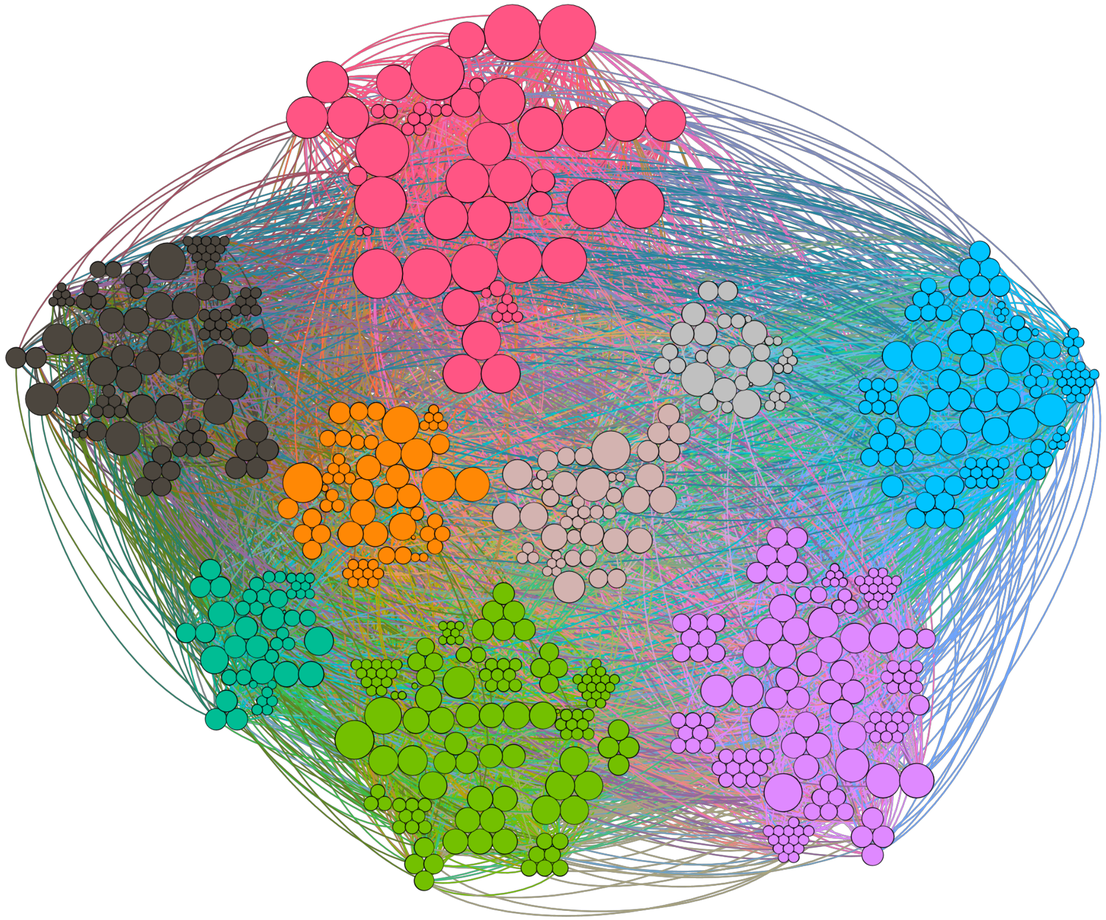
Inferring the topology

- Does the network really look random?
- How can we known if we don’t know how the topology looks like?
- Can we do anything to infer the topology?

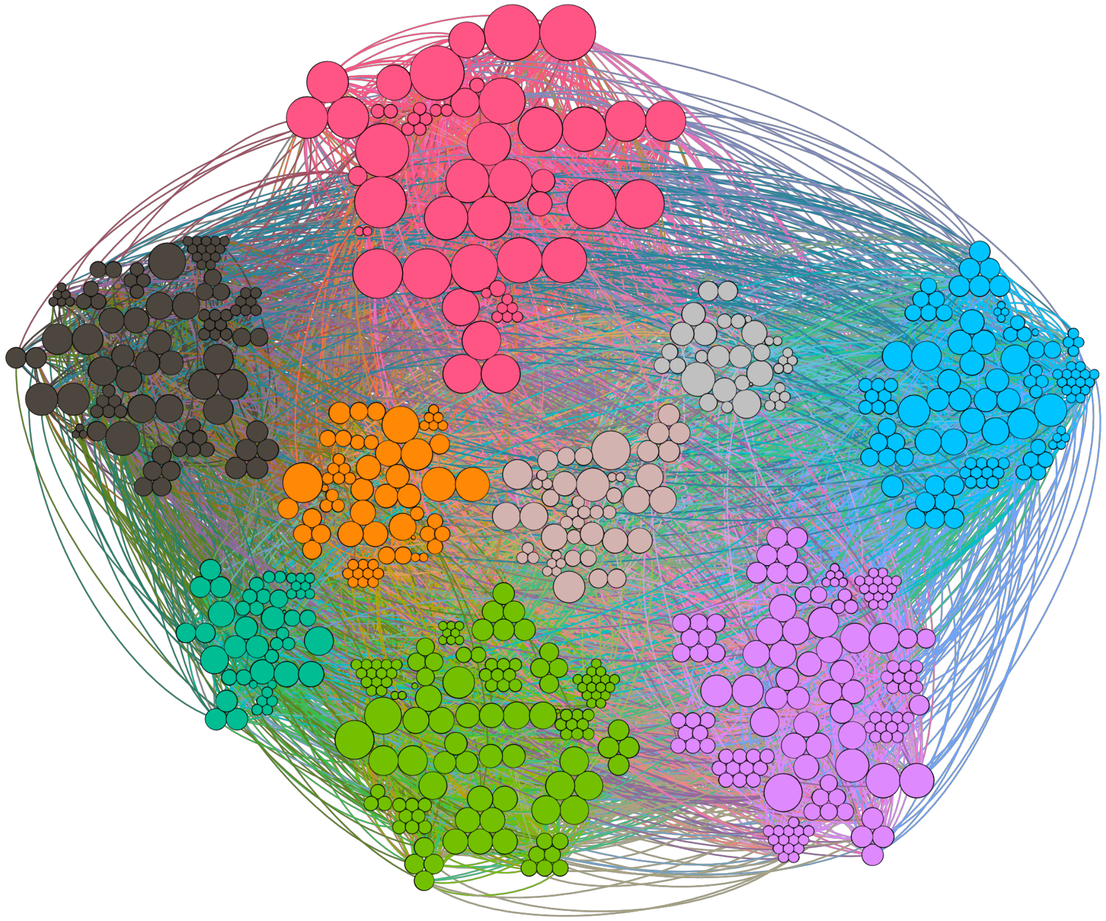
Inferring the topology

- Does the network really look random?
- How can we known if we don’t know how the topology looks like?
- Can we do anything to infer the topology?

Bitcoin testnet topology



Bitcoin testnet topology



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

TxProbe: Discovering Bitcoin's Network Topology Using Orphan Transactions

https://fc19.ifca.ai/preproceedings/58-preproceedings.pdf