CRYPTOCURRENCIES (COMP0143): The Bitcoin Network Layer

Sergi Delgado Segura

THIS TIME (WEEK 3)

Differences between client/server and peer-to-peer paradigms

How a new node joins the network

- How it learns about the network
- How others learn about it

Actors and their role in the network

The gossip protocol

BEFORE WE START

We will use Bitcoin as an 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 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 where actors are split into **clients** and **servers**

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 or 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

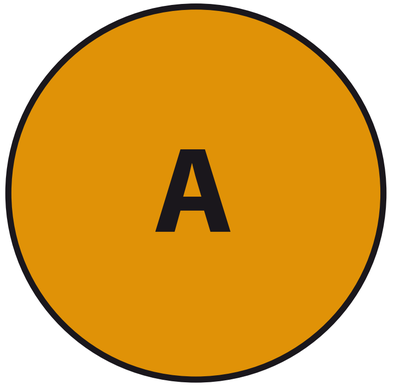
How do you find peers when you run a new node in the network?

P2P BOOTSTRAPPING

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

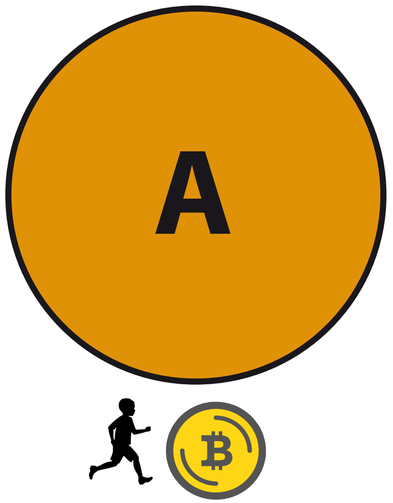
How do peers announce their presence in the network?

PEER DISCOVERY?

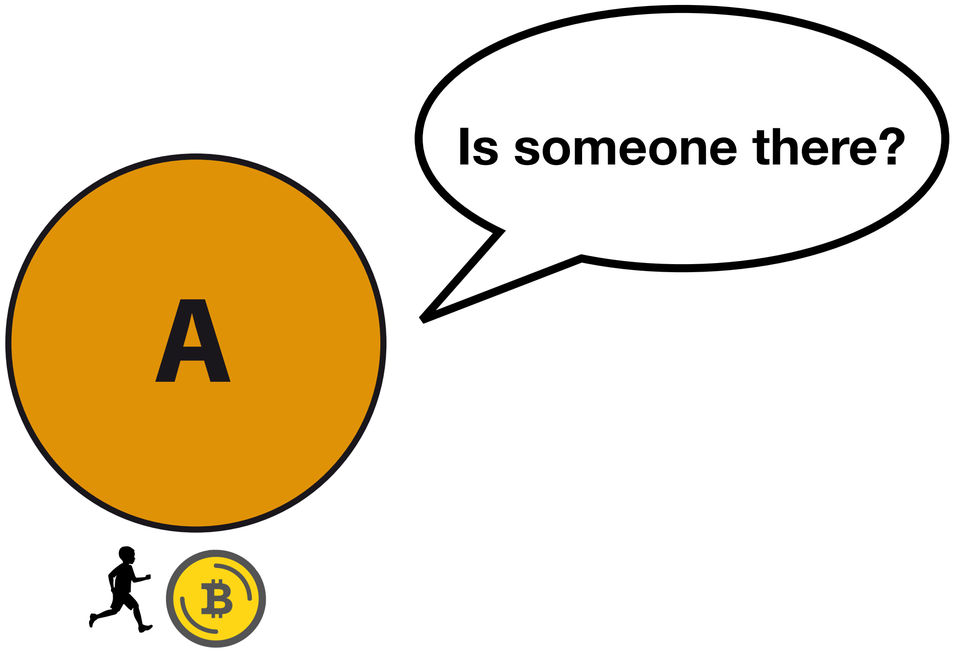


PEER DISCOVERY?

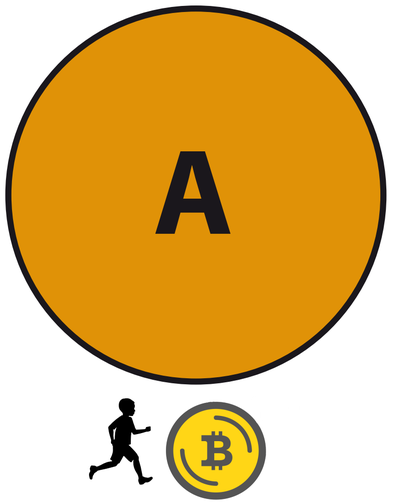
PEER DISCOVERY?



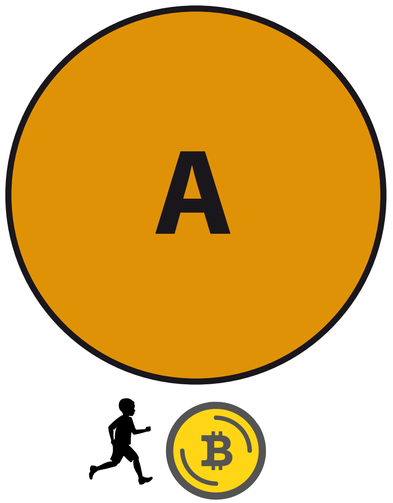
PEER DISCOVERY?



PEER DISCOVERY?

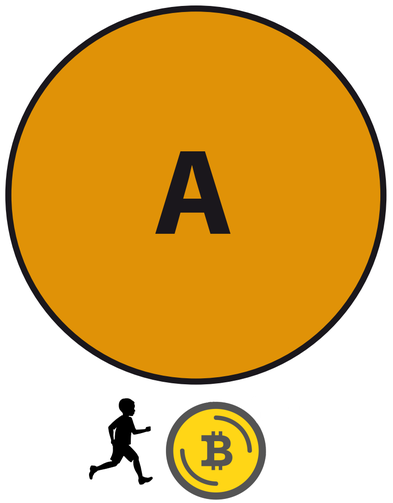


PEER DISCOVERY?



tumbleweed

PEER DISCOVERY?

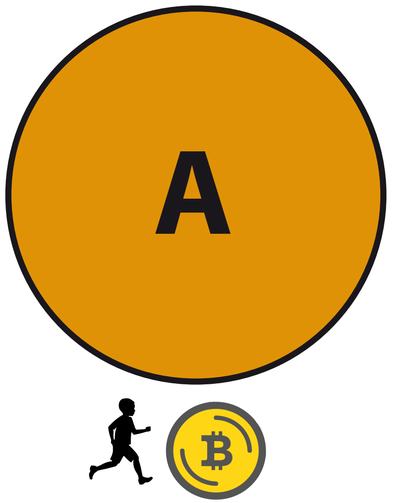


tumbleweed

PEER DISCOVERY?

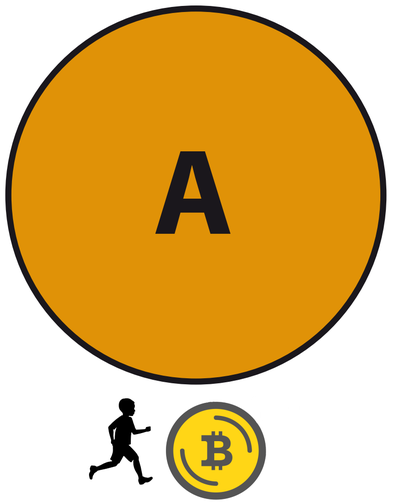
tumbleweed

PEER DISCOVERY?



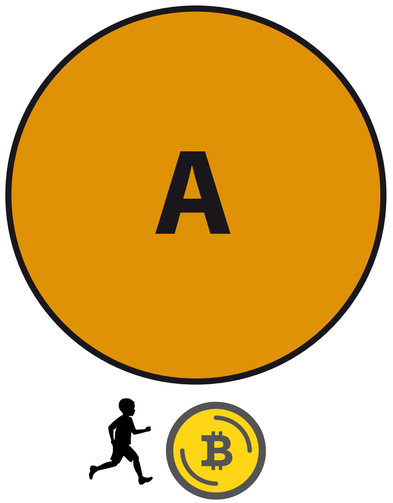
tumbleweed

PEER DISCOVERY?

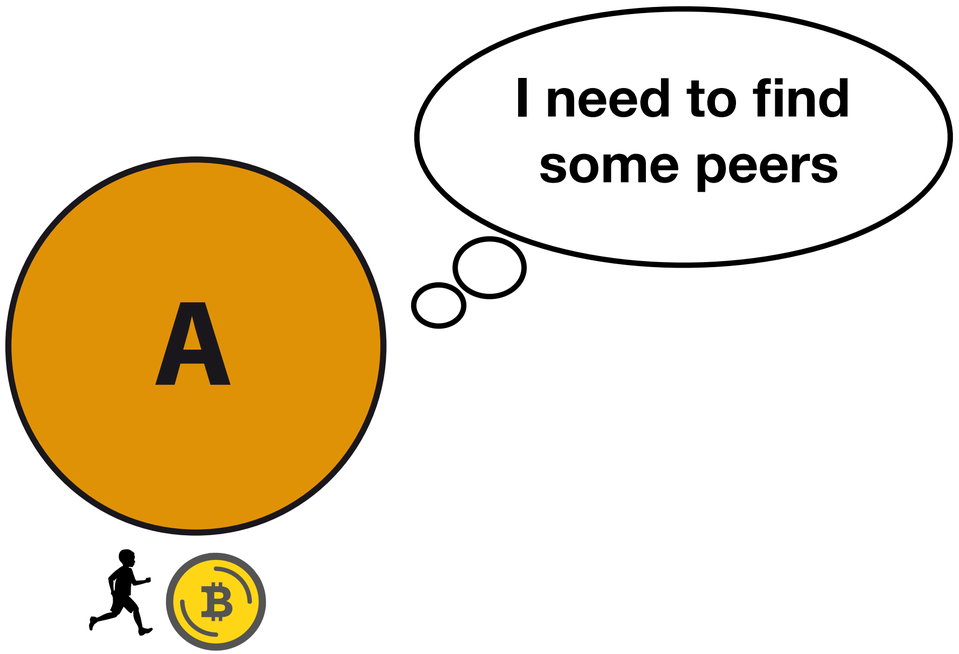


tumbleweed

PEER DISCOVERY?



PEER DISCOVERY?



P2P BOOTSTRAPPING

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

How do peers announce their presence in the network?

P2P BOOTSTRAPPING

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

How do peers announce their 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 are files served?

P2P FILE SHARING (1/2)

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

How are files 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 are 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 are 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 do cryptocurrency networks follow?

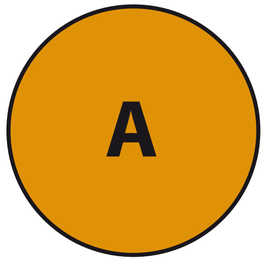
P2P FILE SHARING (2/2)

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

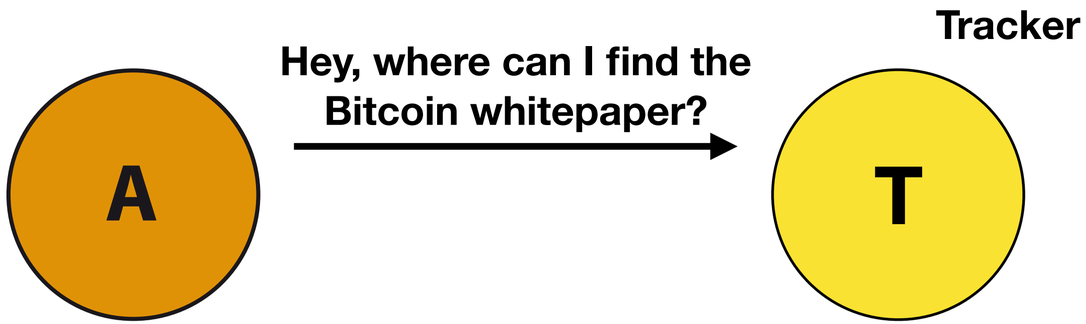
What paradigm do cryptocurrency networks follow? **Announce**

REQUEST PARADIGM (BitTorrent)



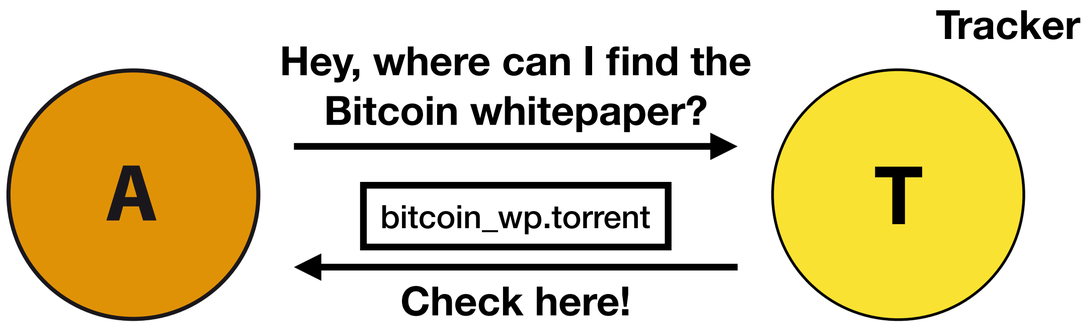
REQUEST PARADIGM (BitTorrent)

REQUEST PARADIGM (BitTorrent)



- Get file information from a tracker

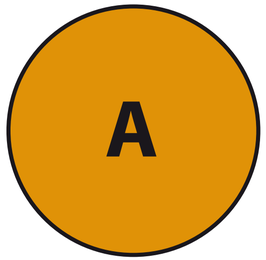
REQUEST PARADIGM (BitTorrent)



- Get file information from a tracker

REQUEST PARADIGM (BitTorrent)

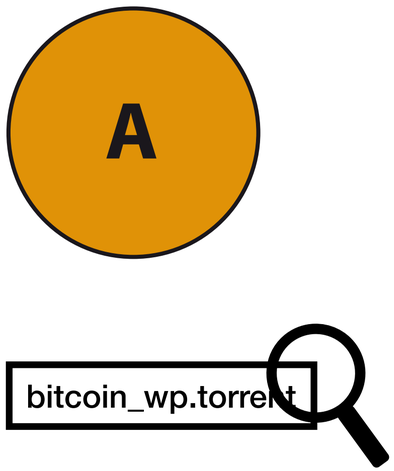
- Get file information from a tracker



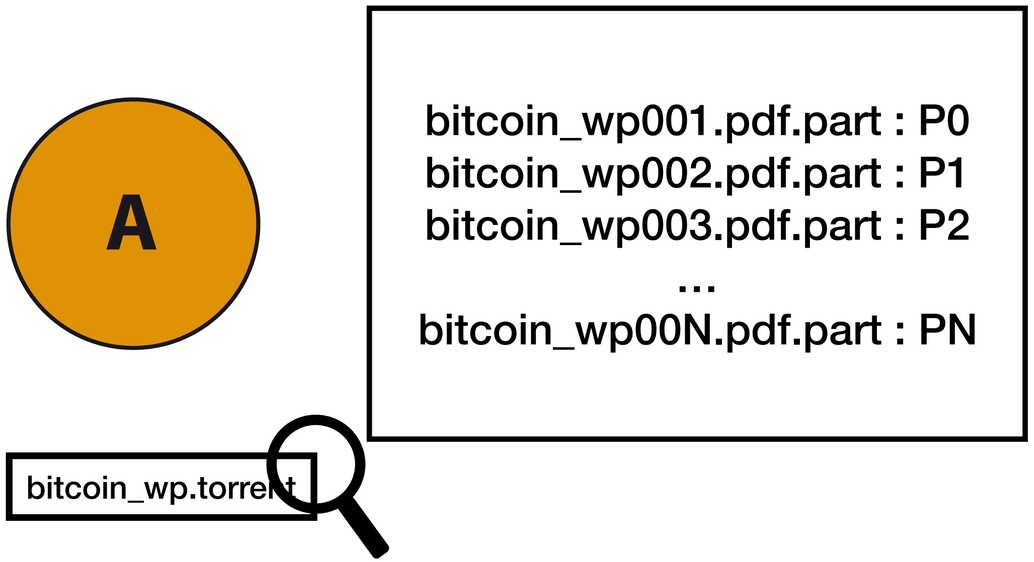
bitcoin_wp.torrent

REQUEST PARADIGM (BitTorrent)

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

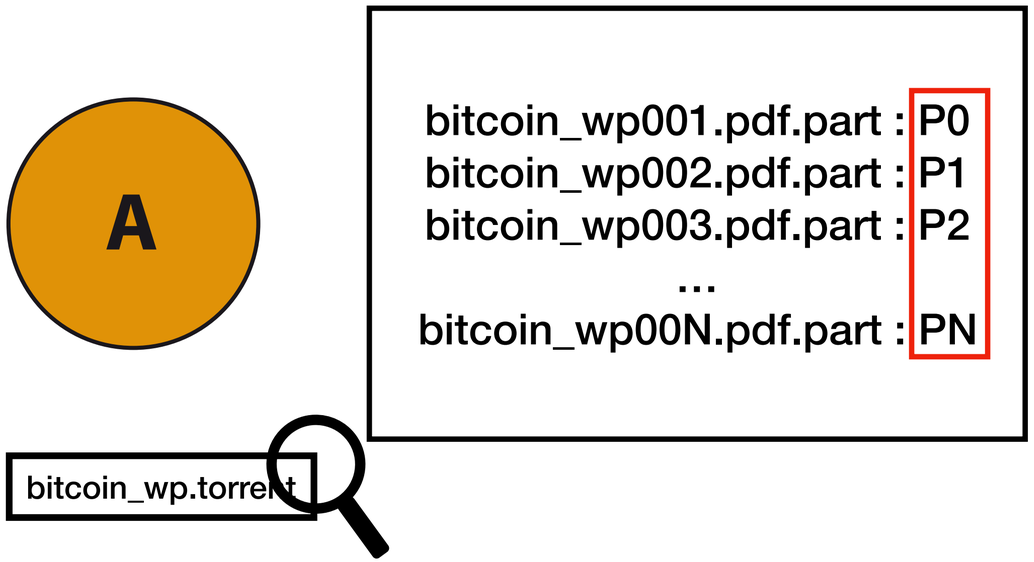


REQUEST PARADIGM (BitTorrent)



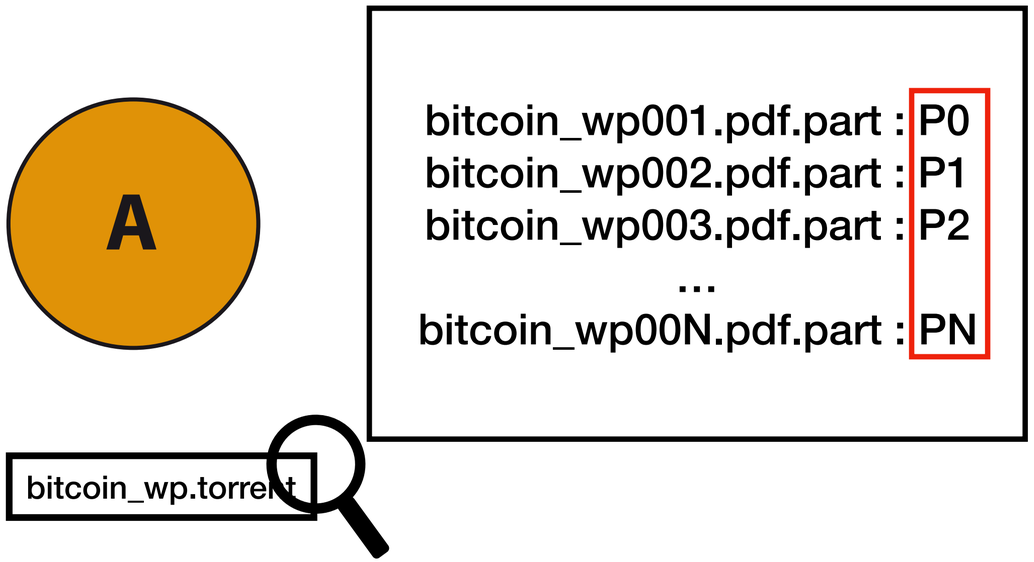
- Get file information from a tracker
- Check the .torrent file

REQUEST PARADIGM (BitTorrent)



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

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

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

ANNOUNCE VS REQUEST

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

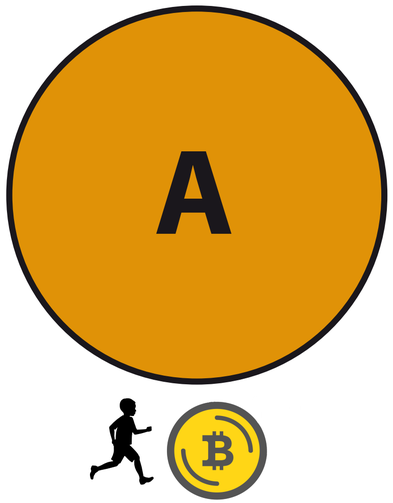
New items (transactions and blocks) can be created by others, so we can’t know about them if they are not offered

What information should a node know about the system?

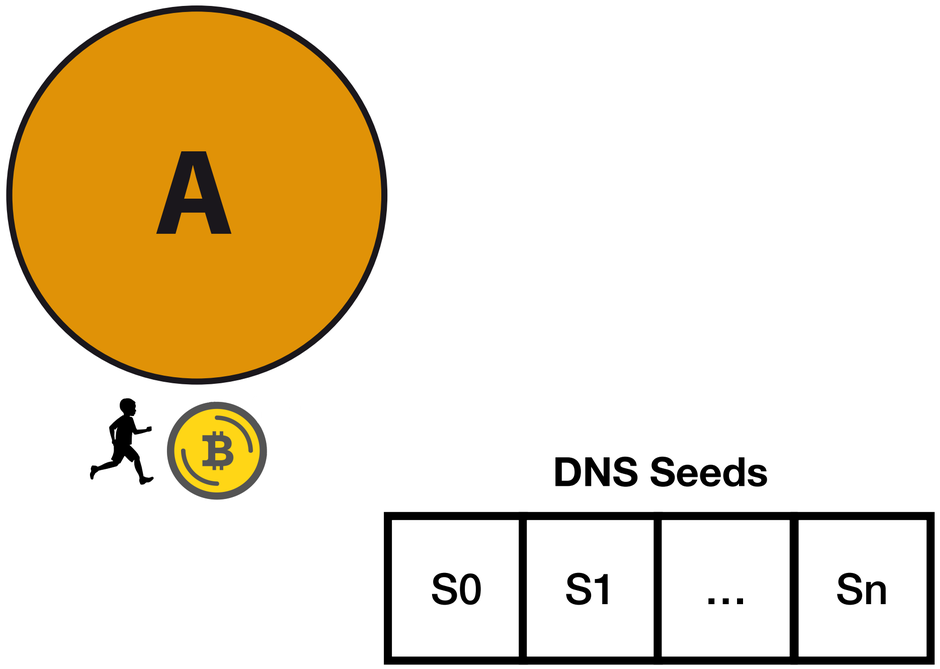
A (full) node needs all the information in order to validate new items

Node bootstrapping and peer discovery

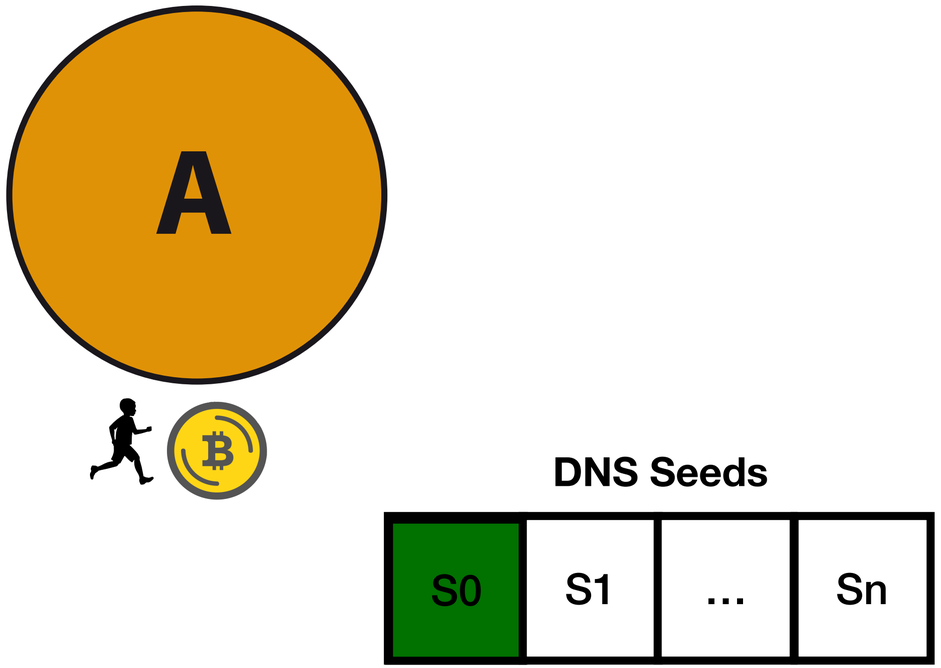
BITCOIN PEER DISCOVERY



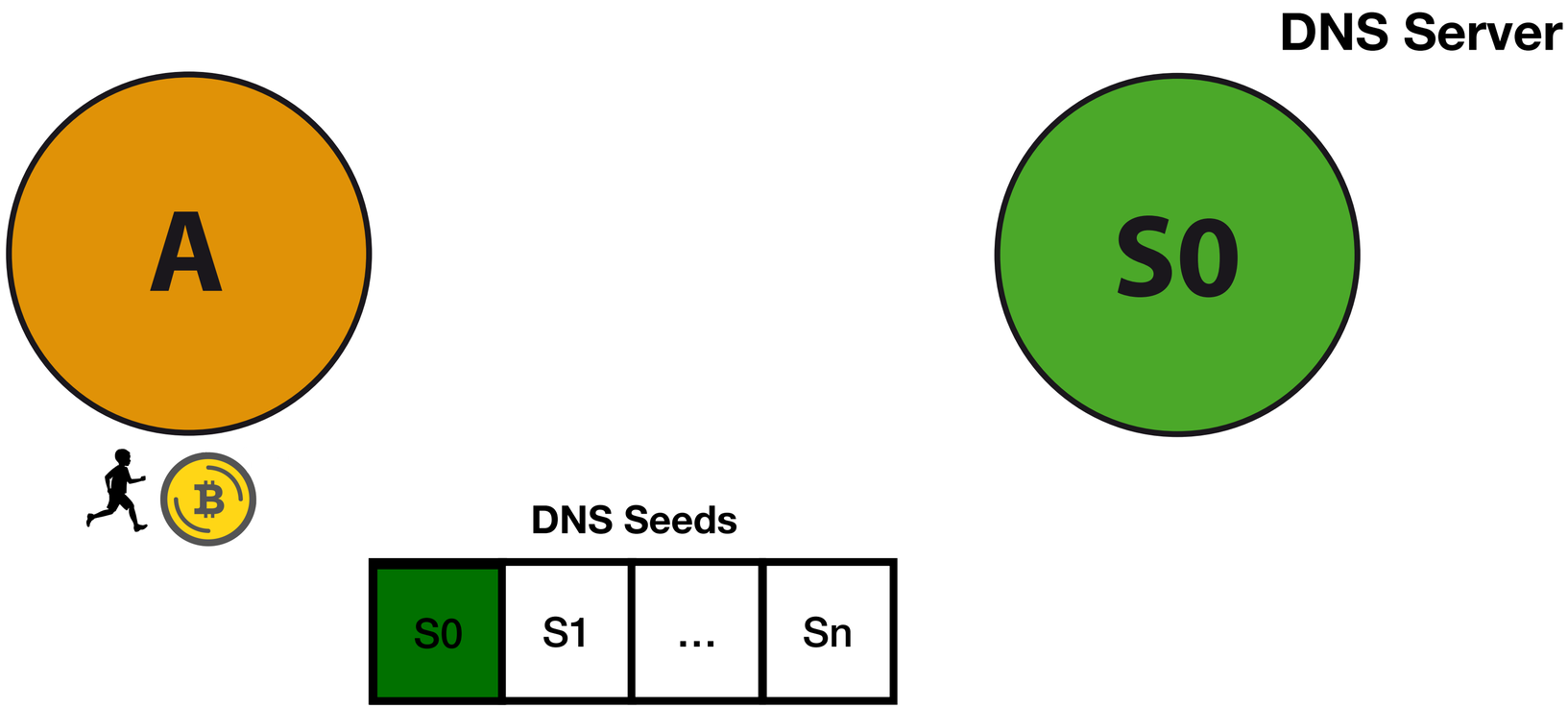
BITCOIN PEER DISCOVERY



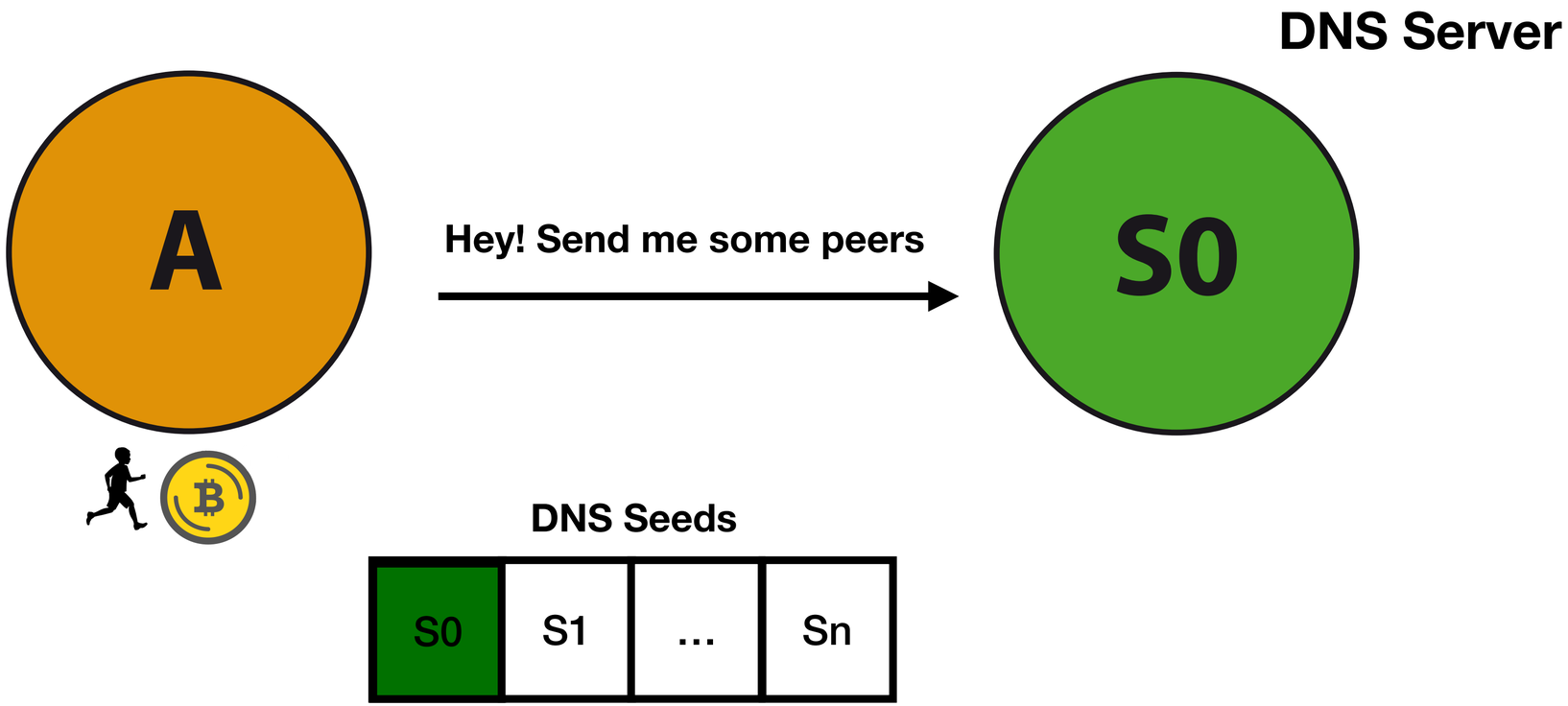
BITCOIN PEER DISCOVERY



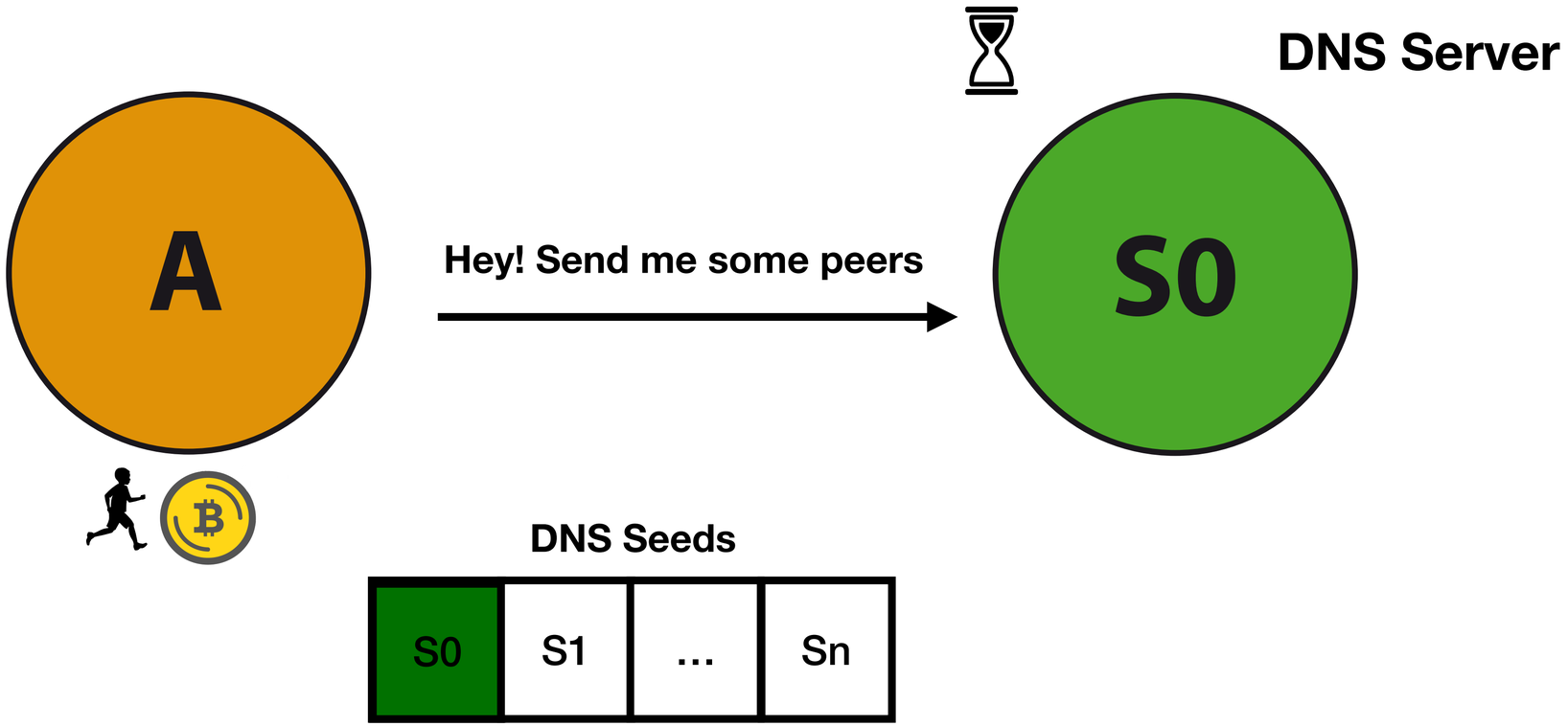
BITCOIN PEER DISCOVERY



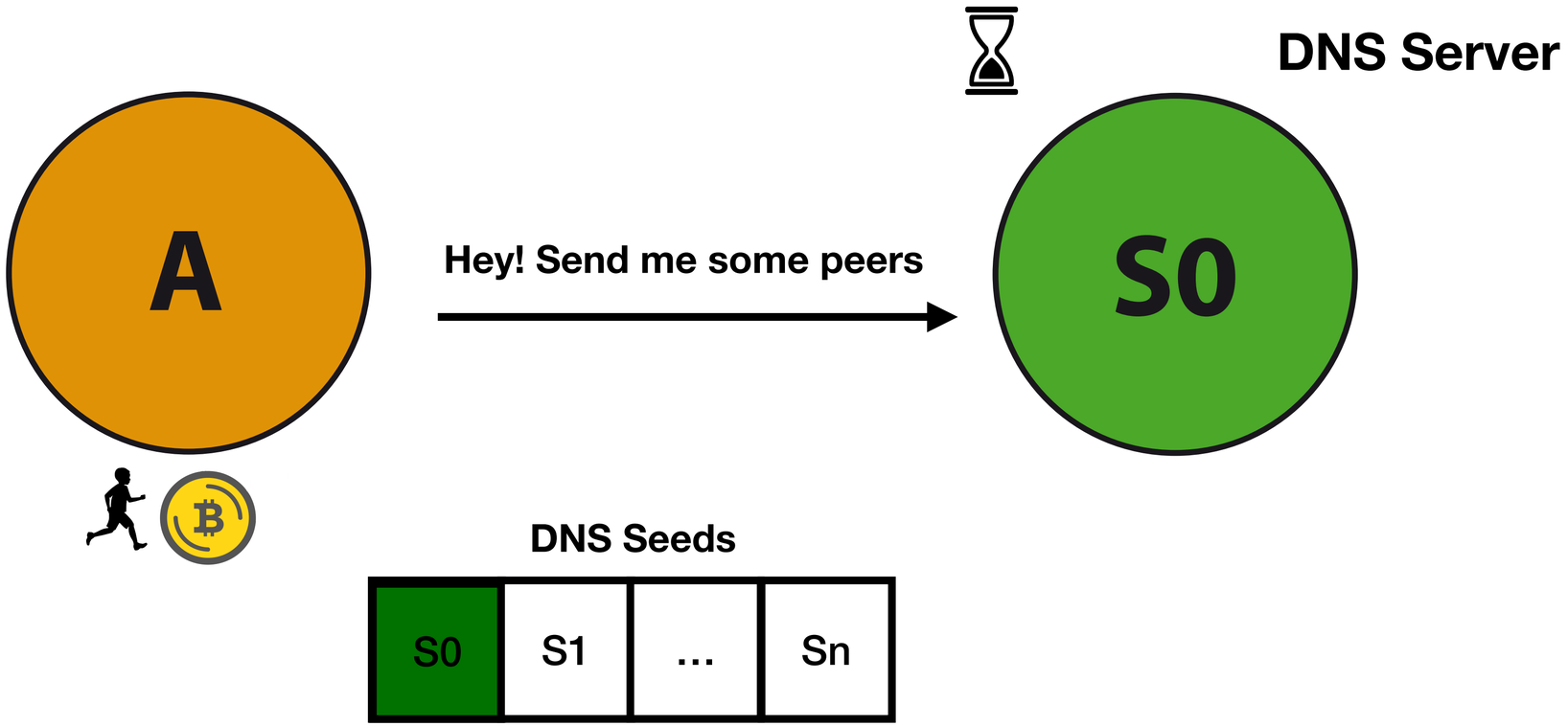
BITCOIN PEER DISCOVERY



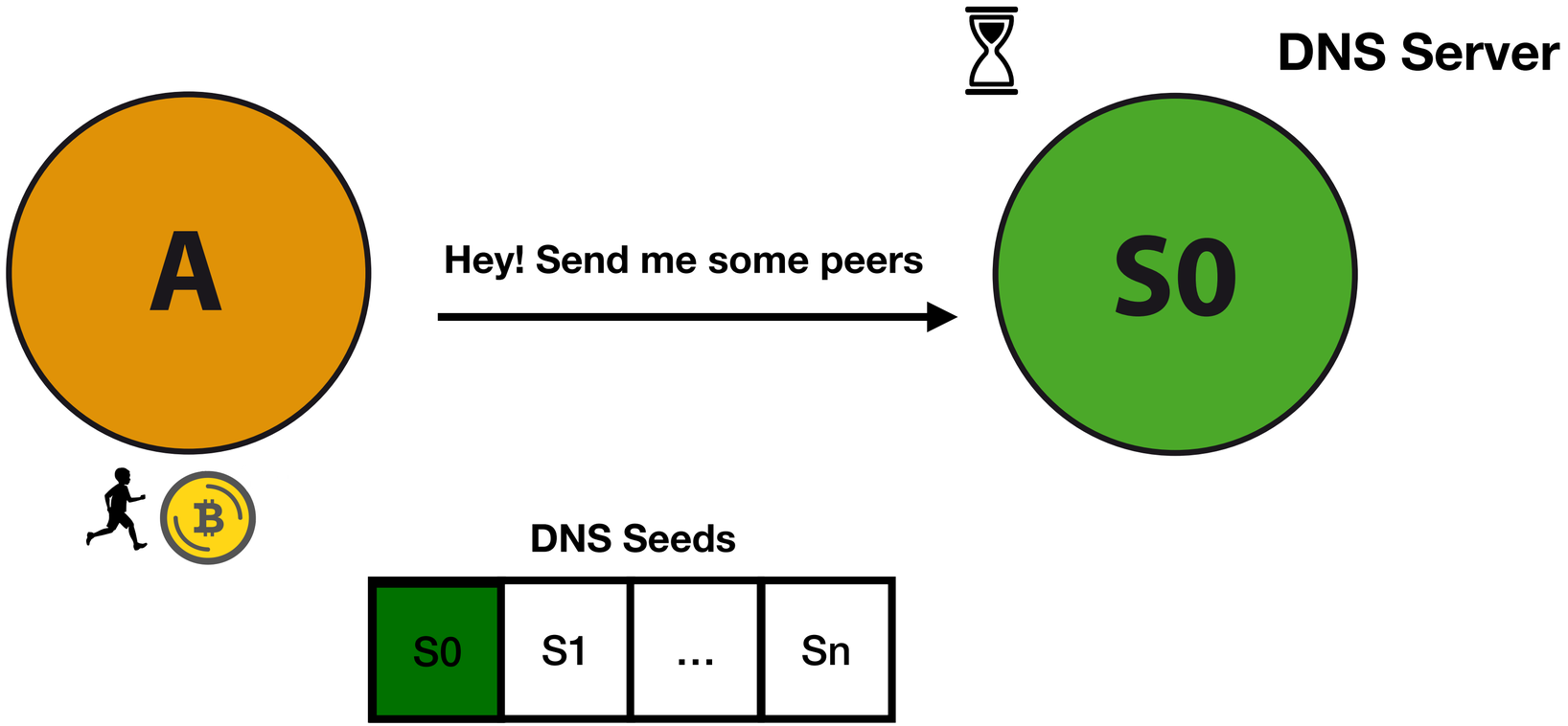
BITCOIN PEER DISCOVERY



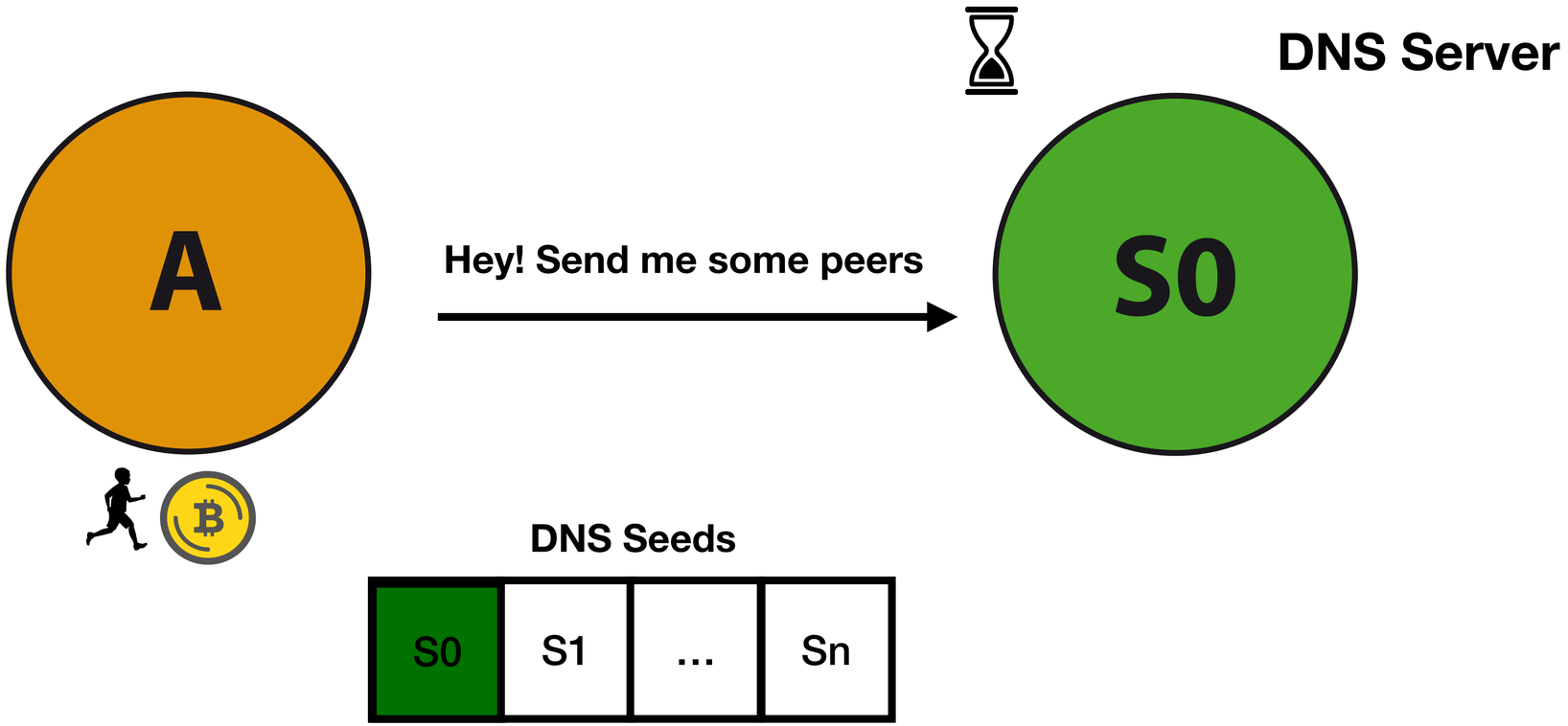
BITCOIN PEER DISCOVERY



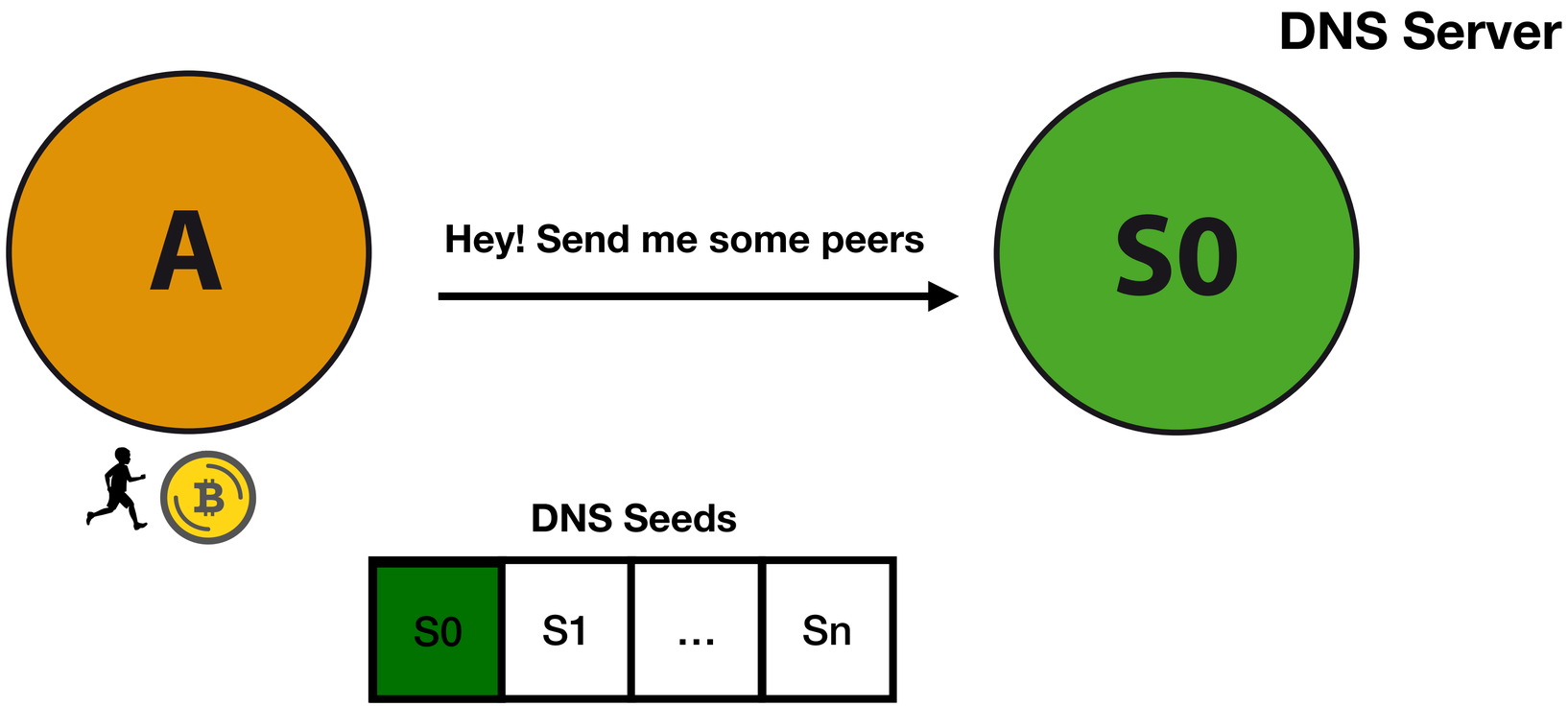
BITCOIN PEER DISCOVERY



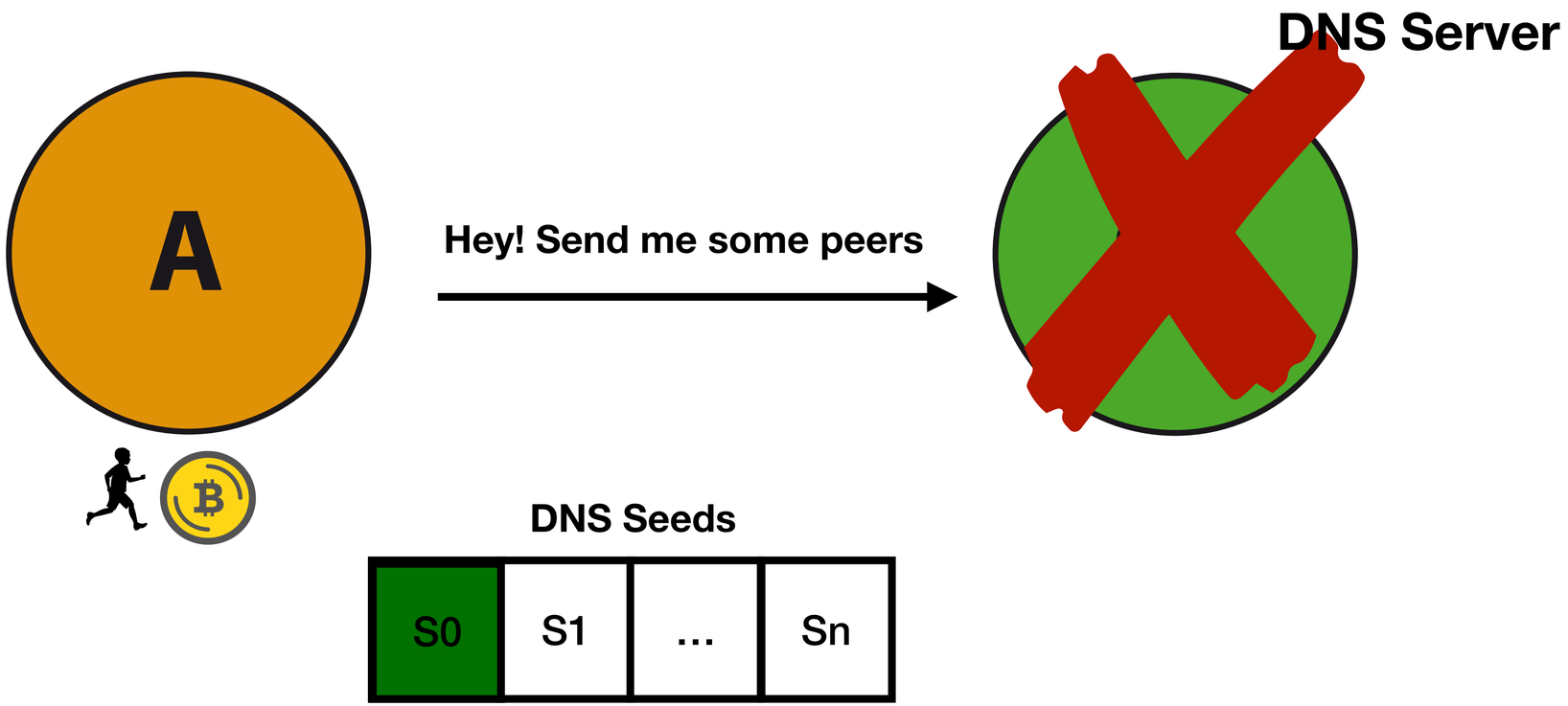
BITCOIN PEER DISCOVERY



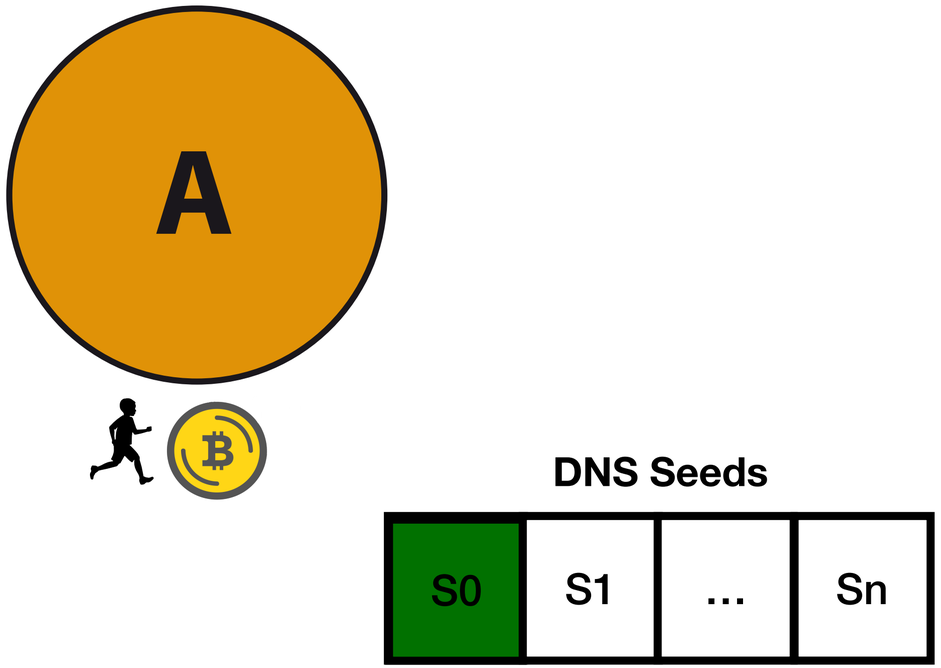
BITCOIN PEER DISCOVERY



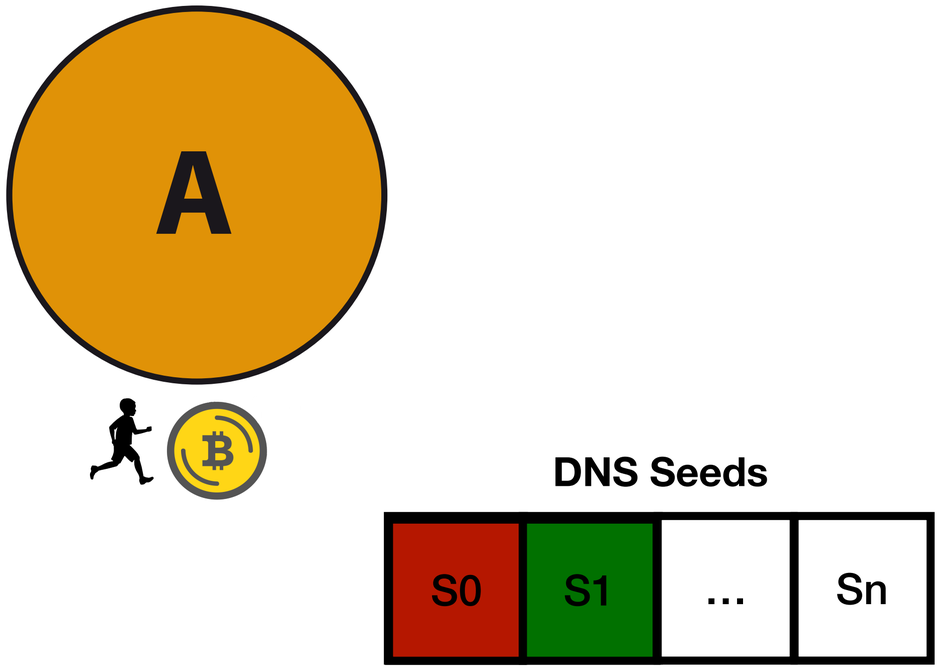
BITCOIN PEER DISCOVERY



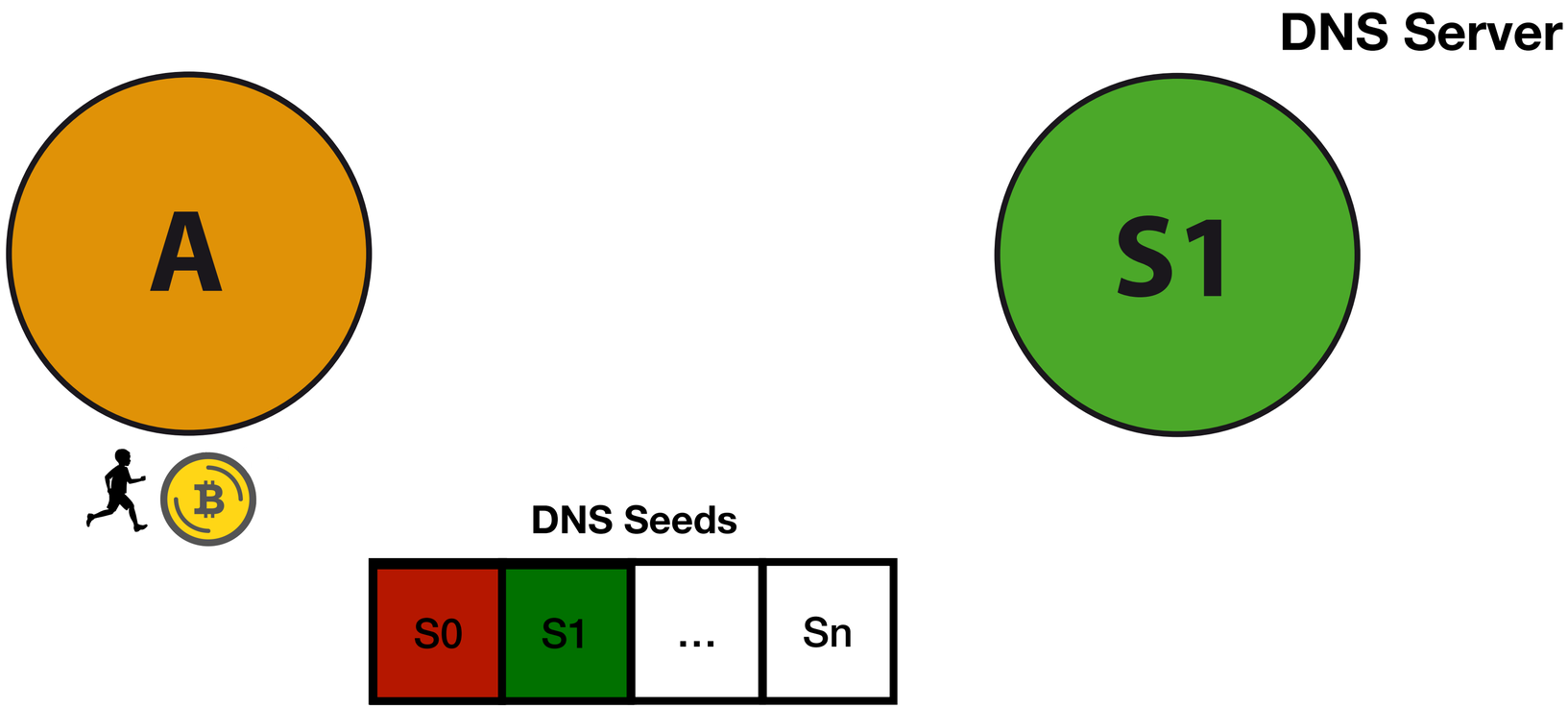
BITCOIN PEER DISCOVERY



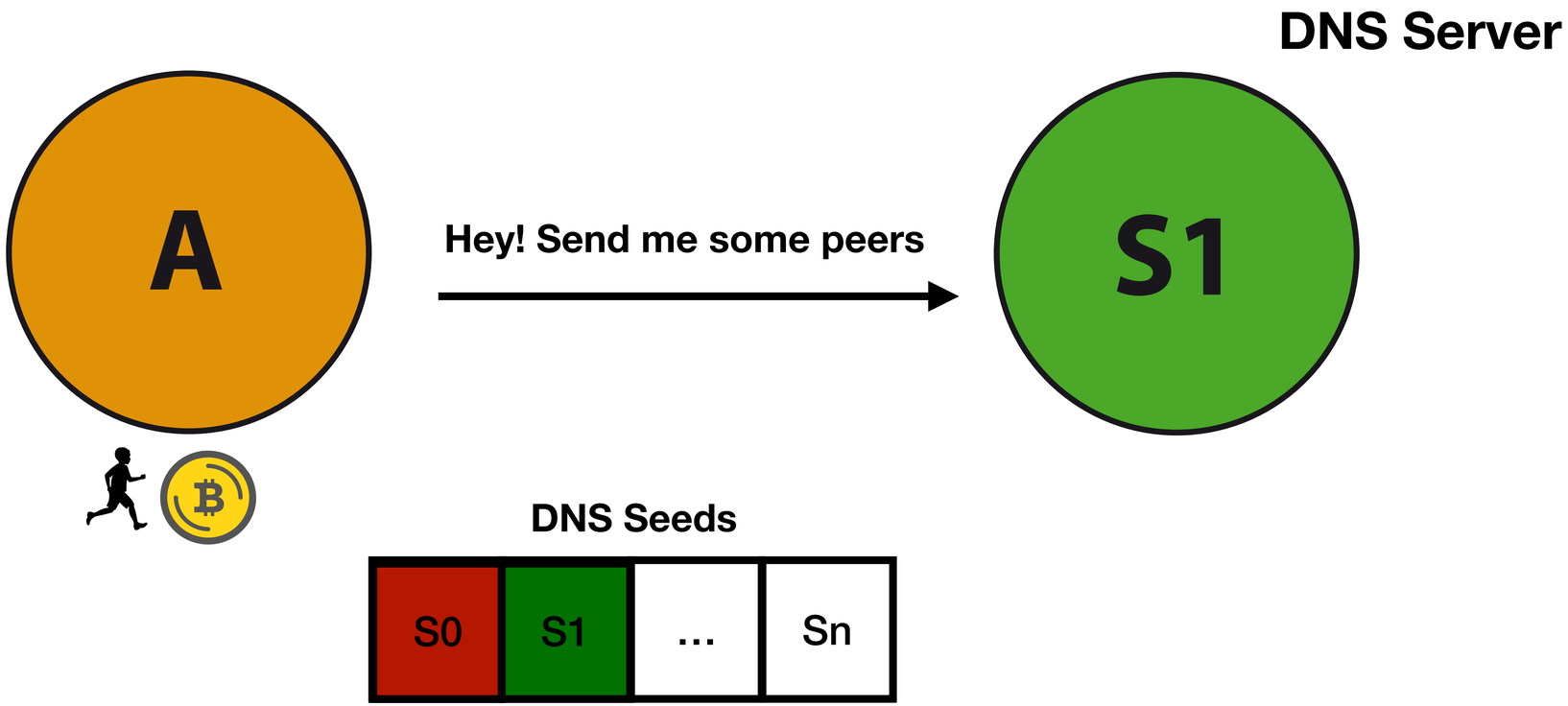
BITCOIN PEER DISCOVERY



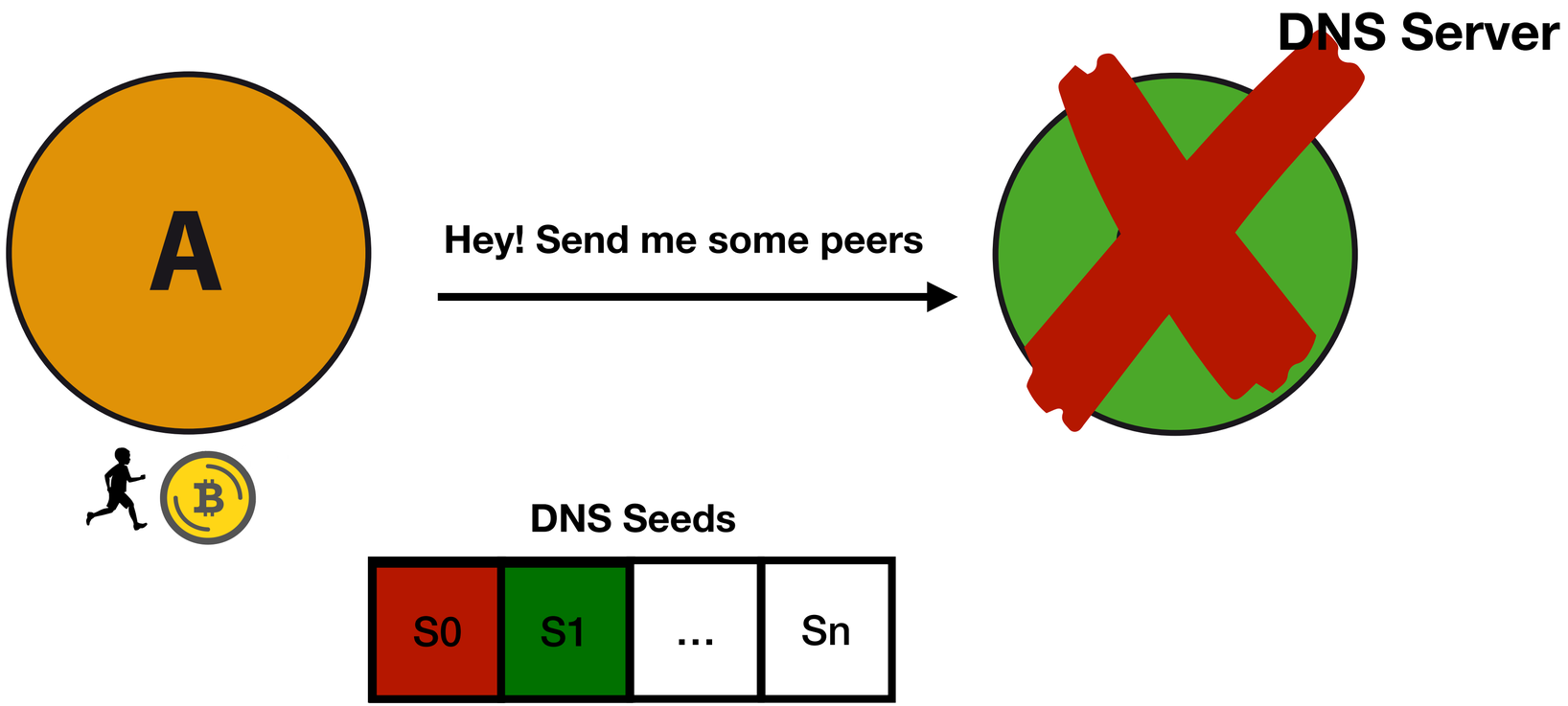
BITCOIN PEER DISCOVERY



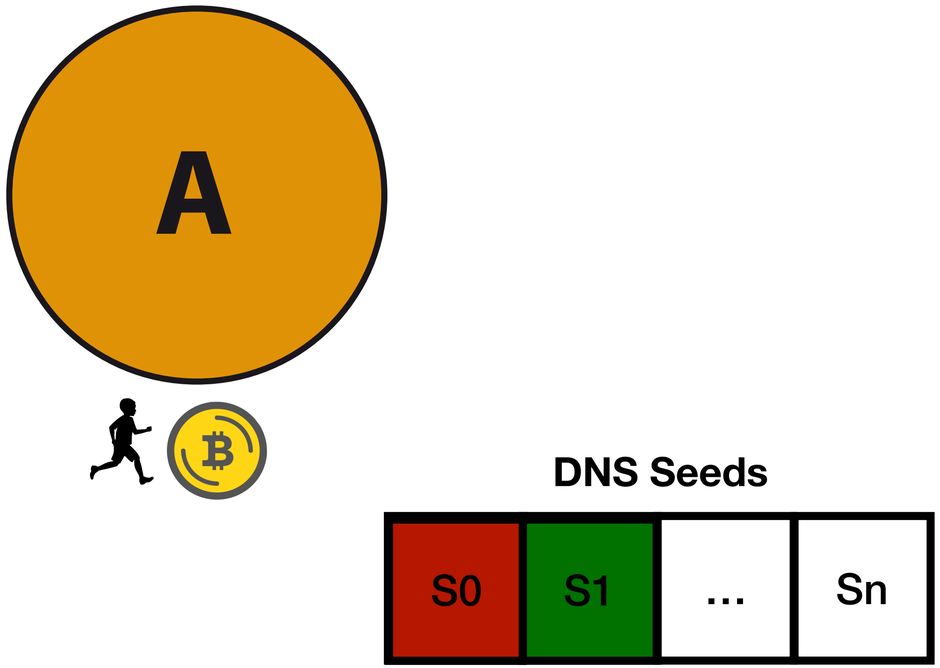
BITCOIN PEER DISCOVERY



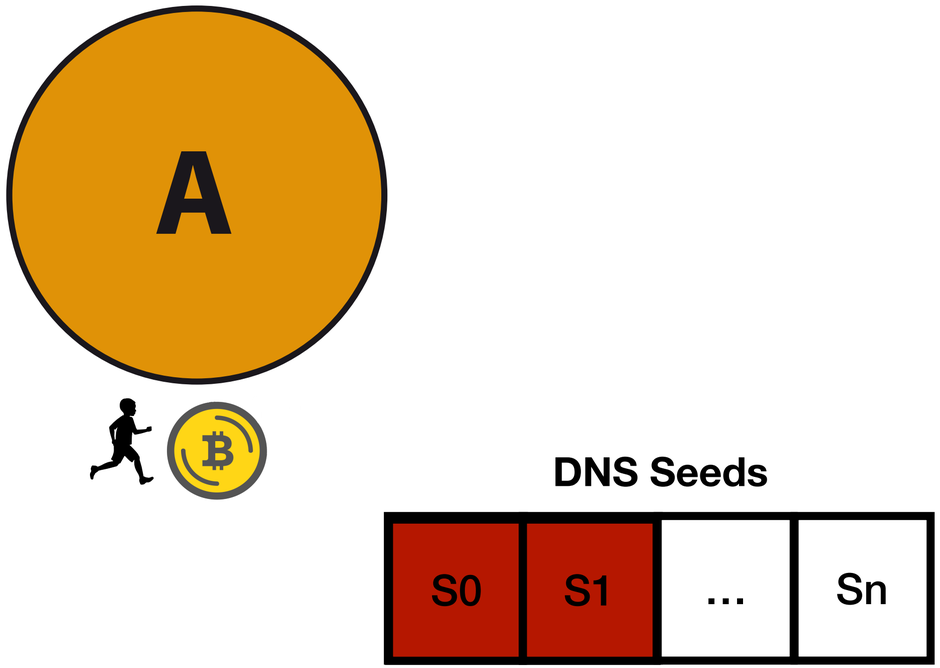
BITCOIN PEER DISCOVERY



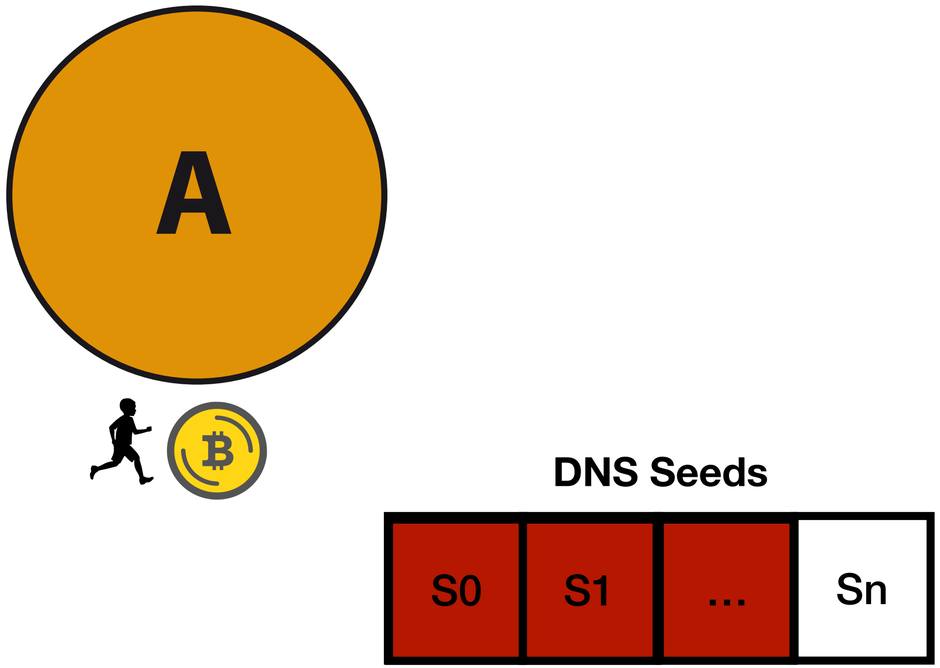
BITCOIN PEER DISCOVERY



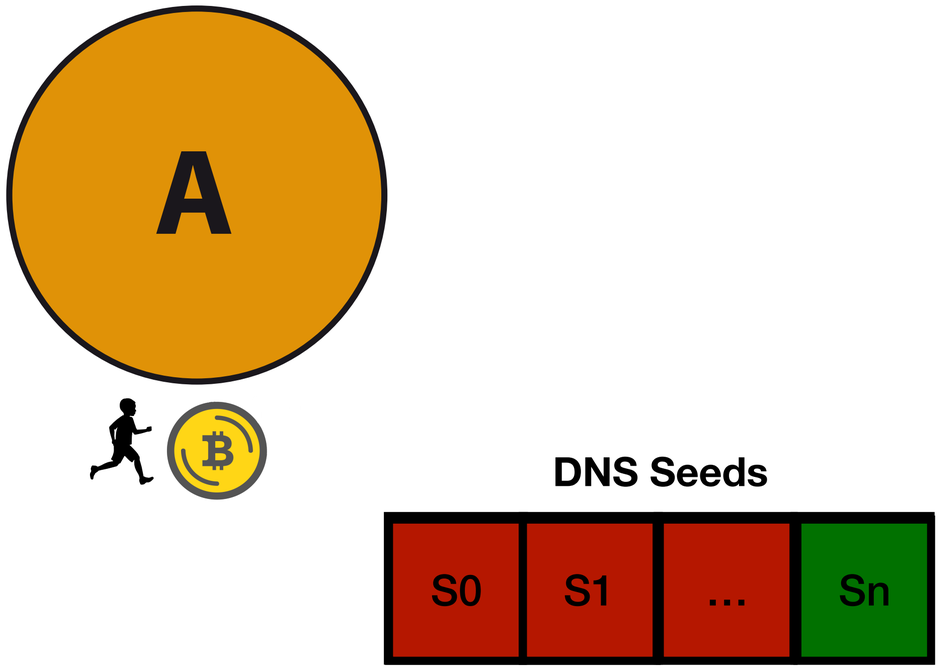
BITCOIN PEER DISCOVERY



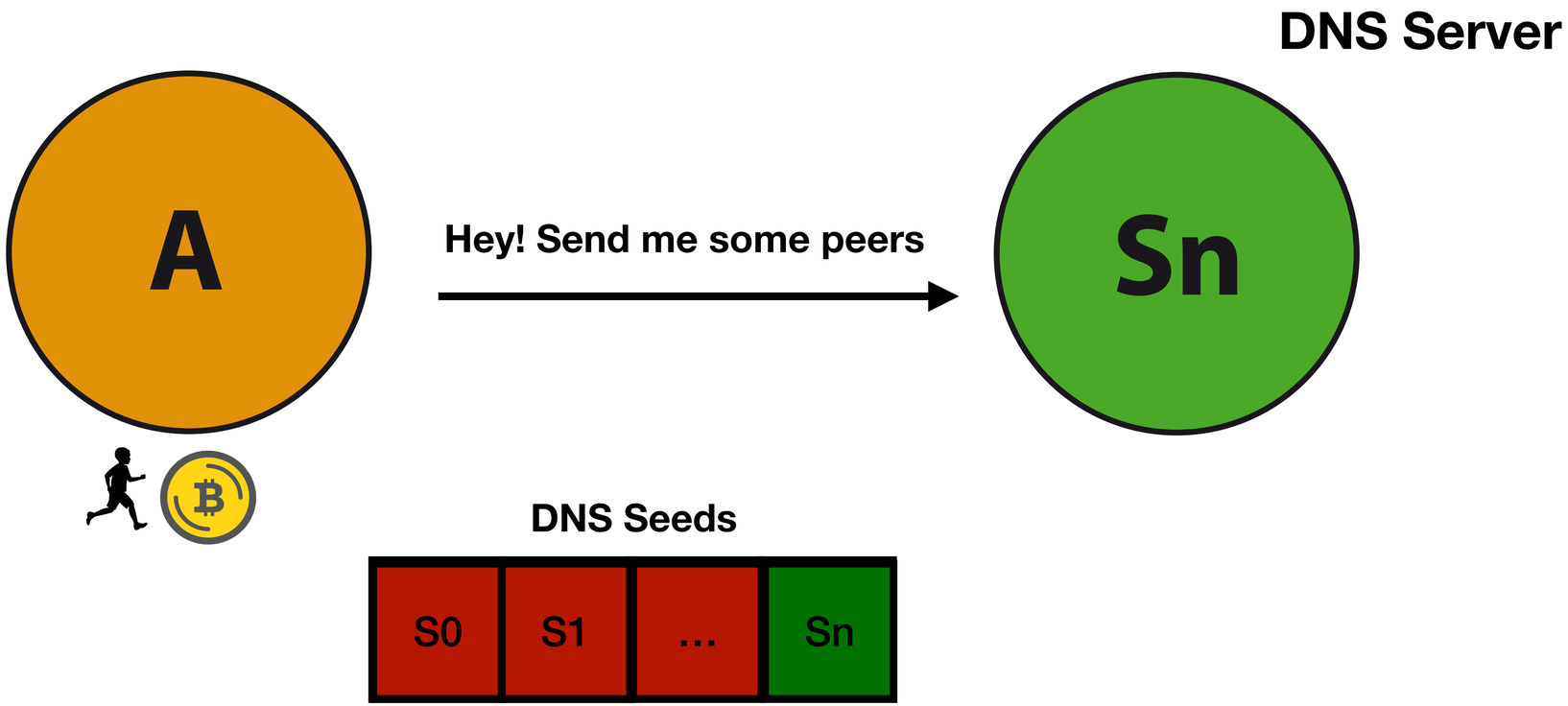
BITCOIN PEER DISCOVERY



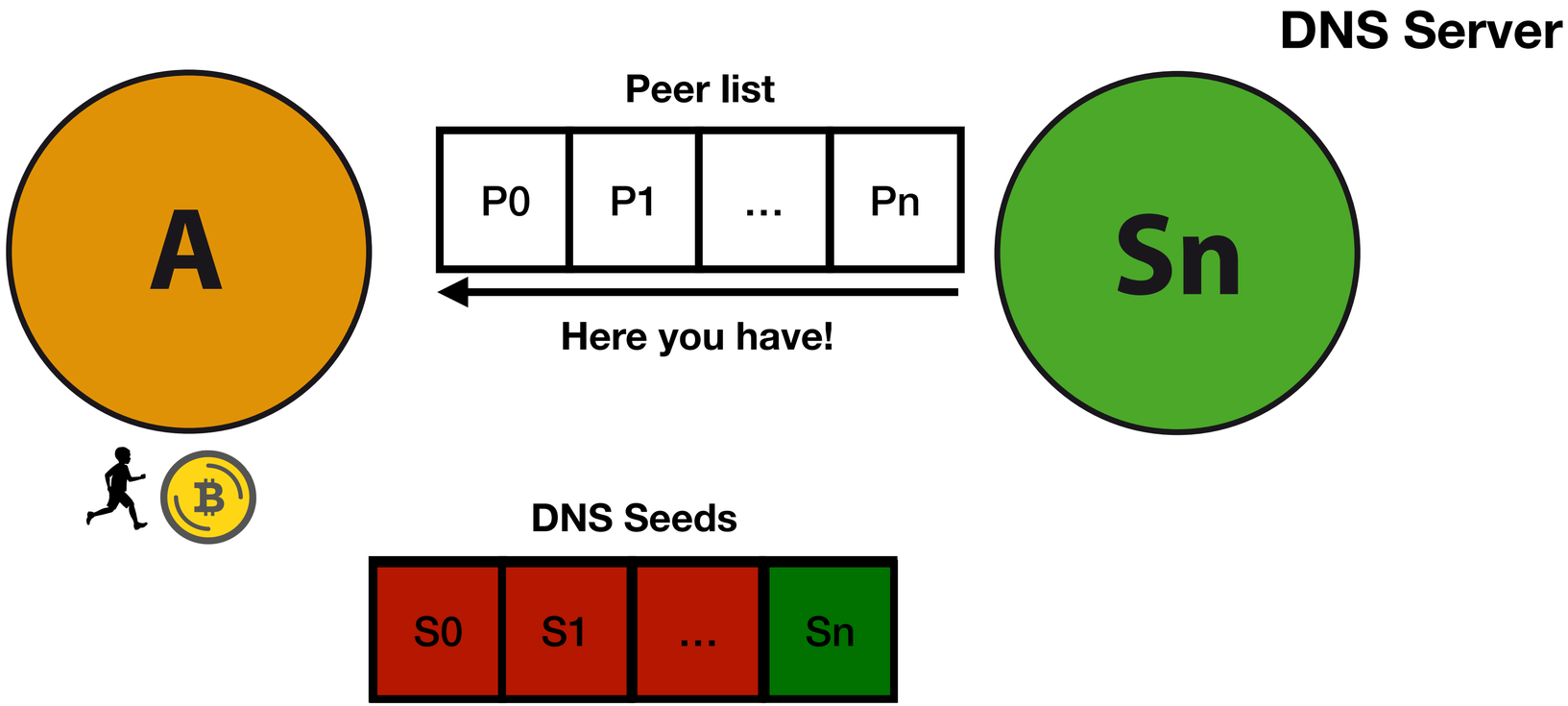
BITCOIN PEER DISCOVERY



BITCOIN PEER DISCOVERY



BITCOIN PEER DISCOVERY

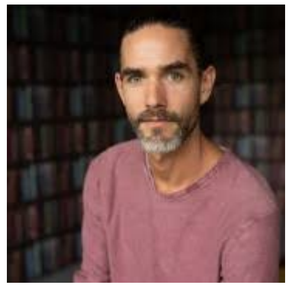


BITCOIN DNS SERVER HOSTS

```
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
```





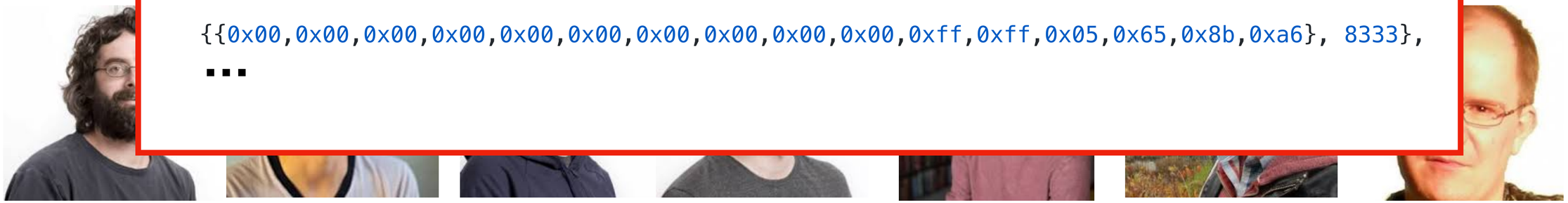


BITCOIN DNS SERVER HOSTS

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
vSeeds.
vSeeds.
vSeeds.
vSeeds.
vSeeds.
```

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



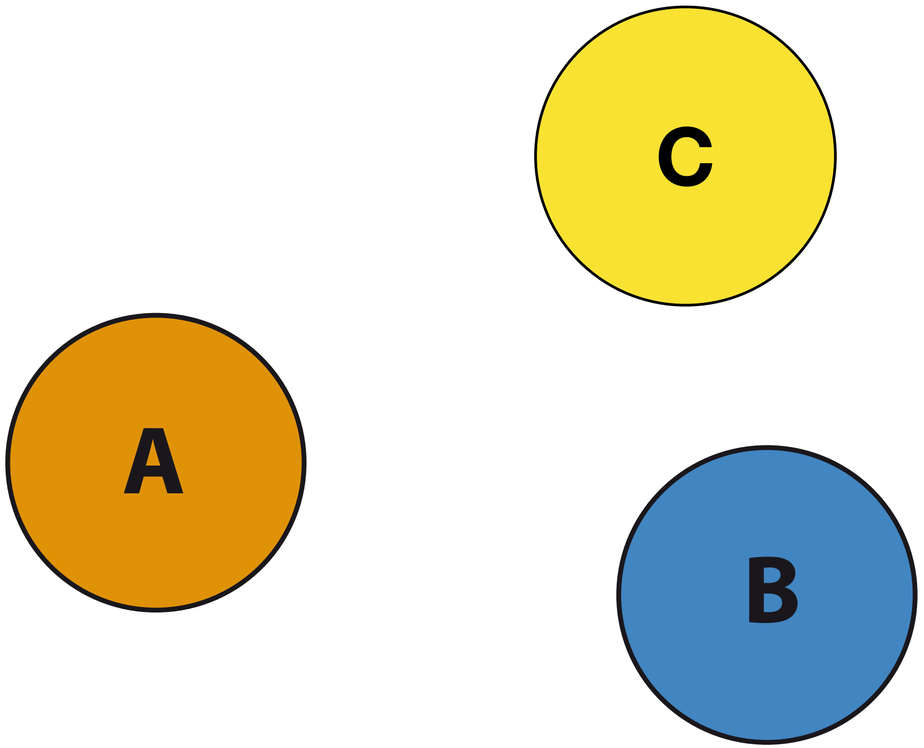
BITCOIN P2P BOOTSTRAPPING (RECAP)

A node bootstraps with no known peers

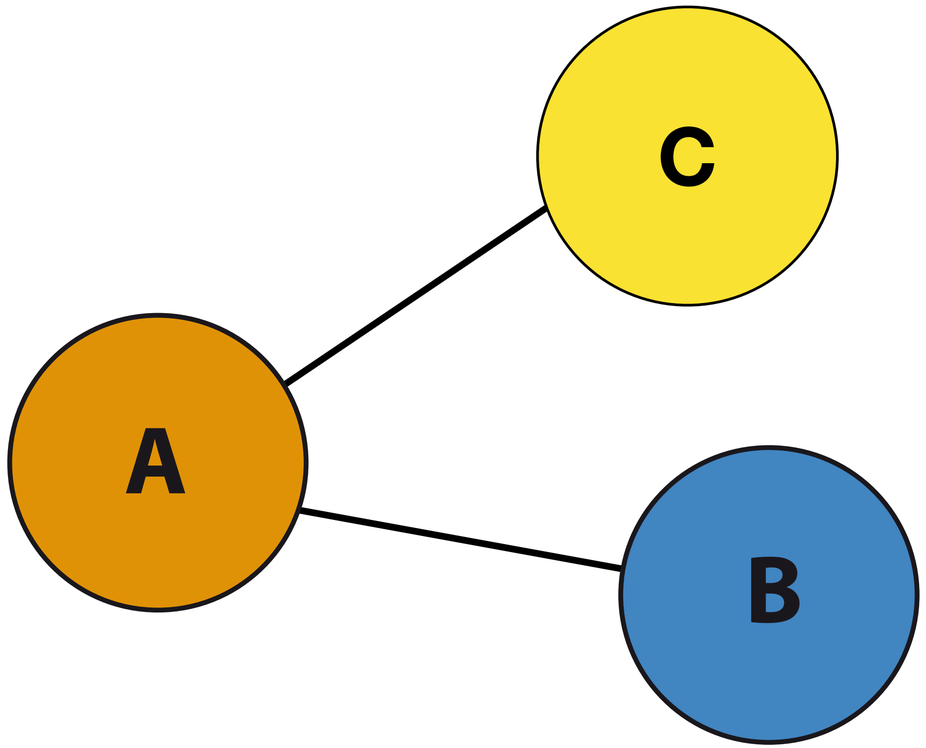
First it tries to query a list of well known DNS seeds

As a **last resource** it uses a hardcoded seed

POPULATING THE PEERS DATABASE

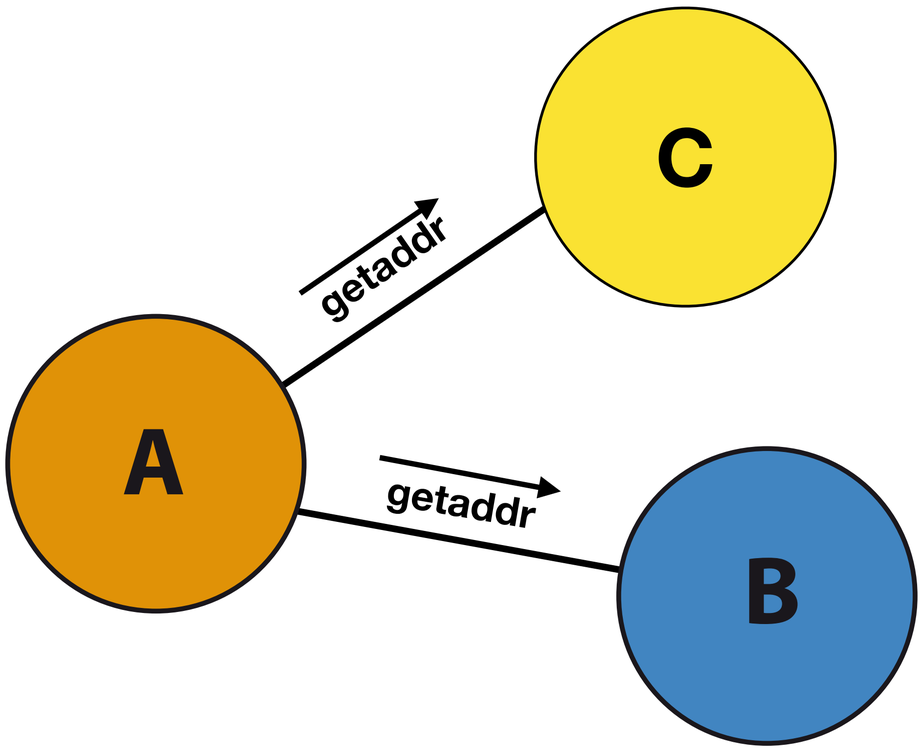


POPULATING THE PEERS DATABASE



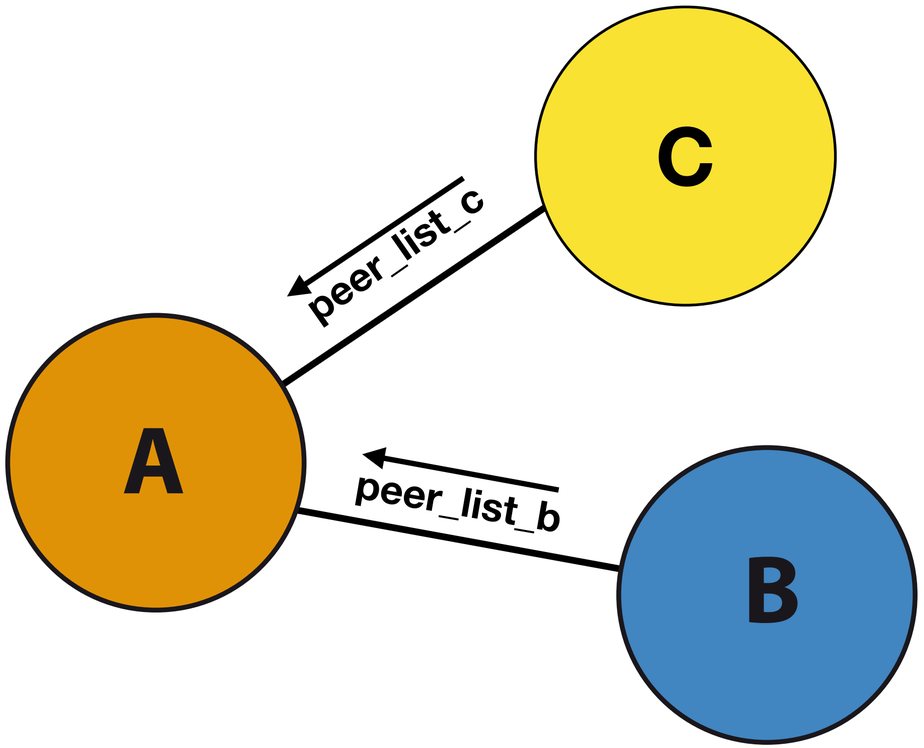
- A connects a subset of peers from the ones learned from the DNS seeds

POPULATING THE PEERS DATABASE



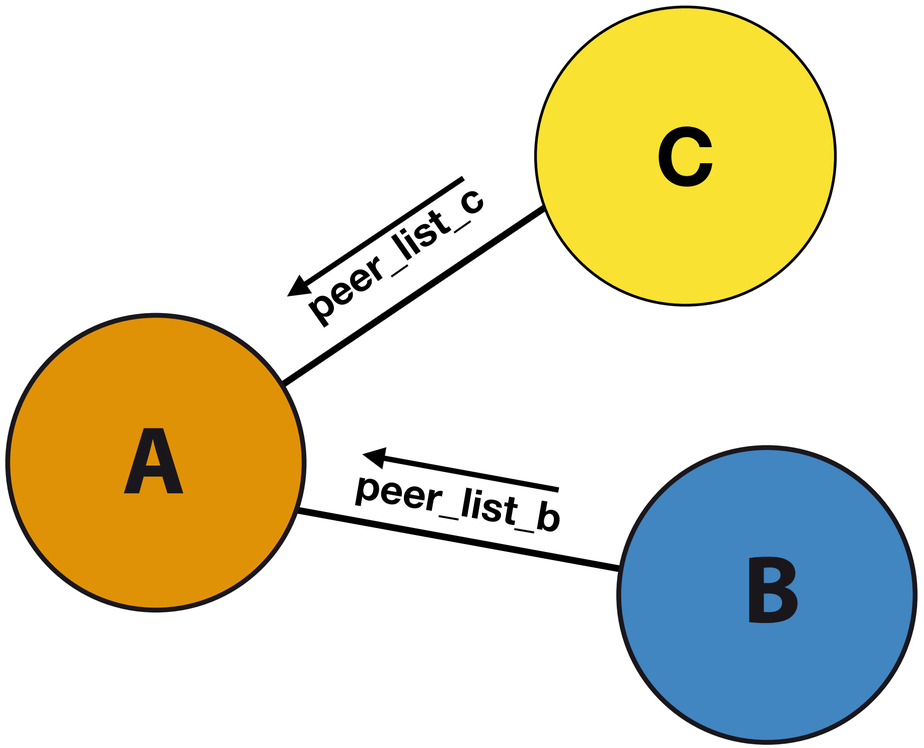
- A connects a subset of peers from the ones learned from the DNS seeds
- A requests more peers to his neighbors (**getaddr**)

POPULATING THE PEERS DATABASE



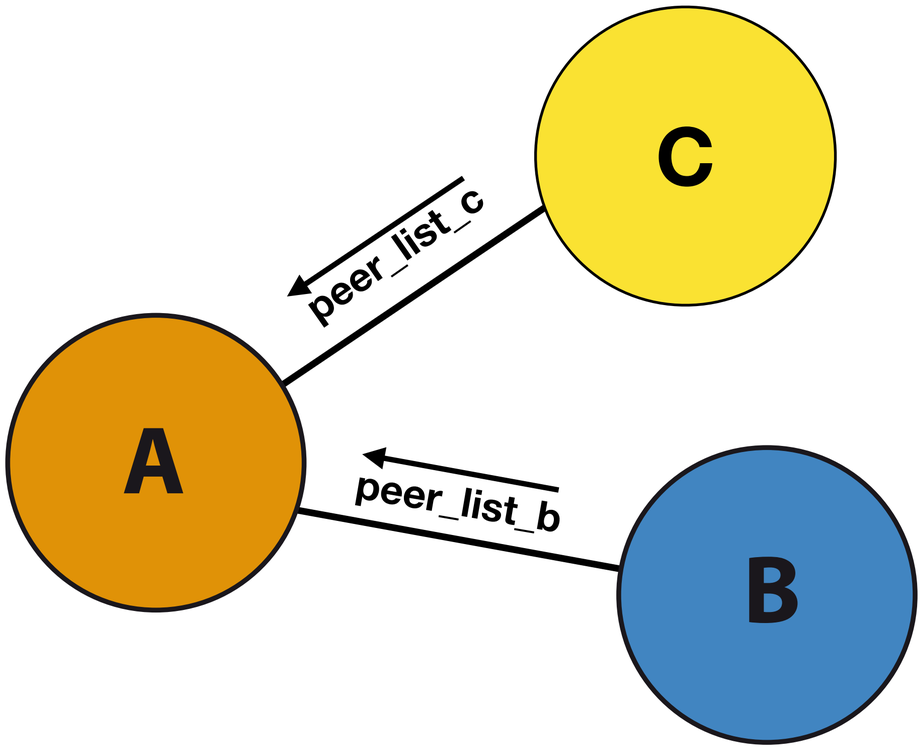
- A connects a subset of peers from the ones learned from the DNS seeds
- A requests more peers to his neighbors (**getaddr**)
- Peers reply with some addresses they know about (**addr, up to 1000 addresses**)

POPULATING THE PEERS DATABASE



- 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 know about (**addr, up to 1000 addresses**)
- A adds the new addresses to its peers database (or updates the existing ones)

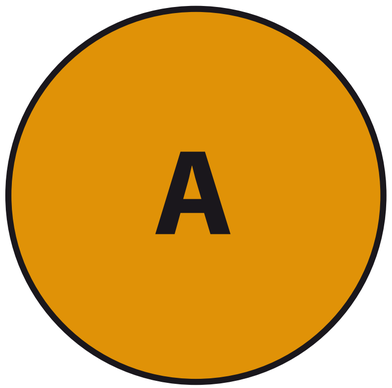
POPULATING THE PEERS DATABASE



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 know about (**addr, up to 1000 addresses**)
- A adds the new addresses to its peers database (or updates the existing ones)
- The database is known as **the addrman**

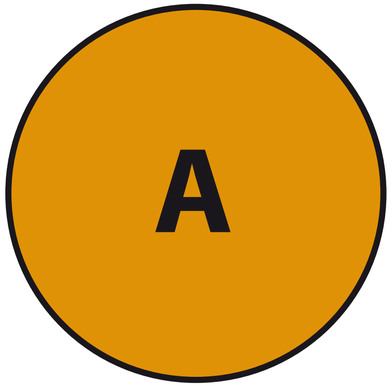
INCOMING/OUTGOING CONNECTIONS



Peer database (addrman)

P0	P1	…	Pn

INCOMING/OUTGOING CONNECTIONS

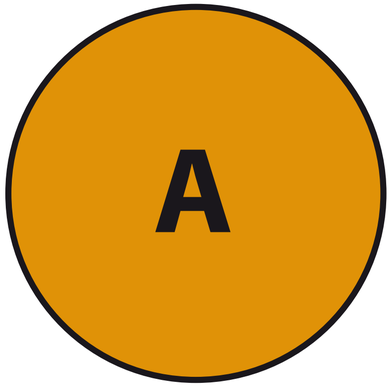


- During bootstrap, a node will start some **outgoing connections** with peers it has learnt about (**8 by default**) and tries to maintain them

Peer database (addrman)

P0	P1	…	Pn

INCOMING/OUTGOING CONNECTIONS



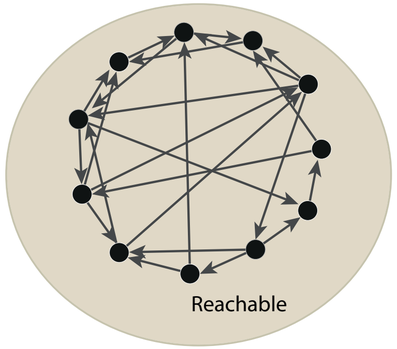
Peer database (addrman)

P0	P1	…	Pn

- During bootstrap, a node will start some **outgoing connections** with peers it has learnt about (**8 by default**) and tries to maintain them
- A node will also accept **some incoming** connections (**117 by default**)

NETWORK TAXONOMY

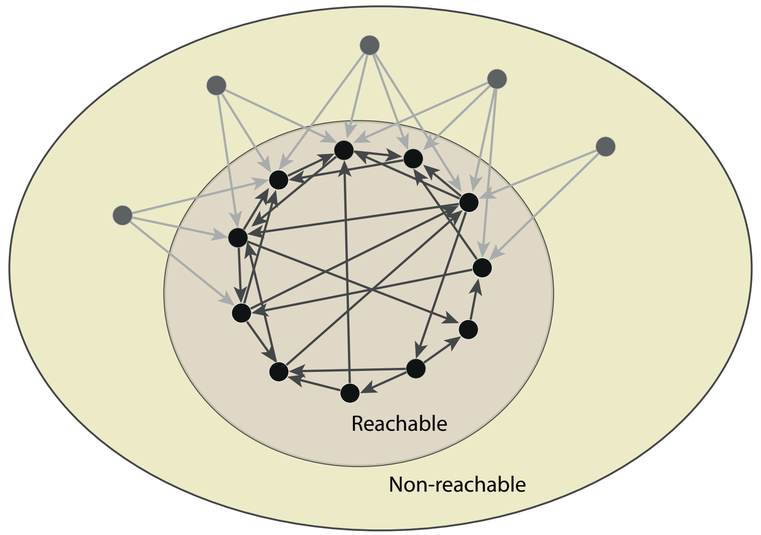
Reachable network: all nodes accept incoming / outgoing connections



NETWORK TAXONOMY

Reachable network: all nodes accept incoming / outgoing connections

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

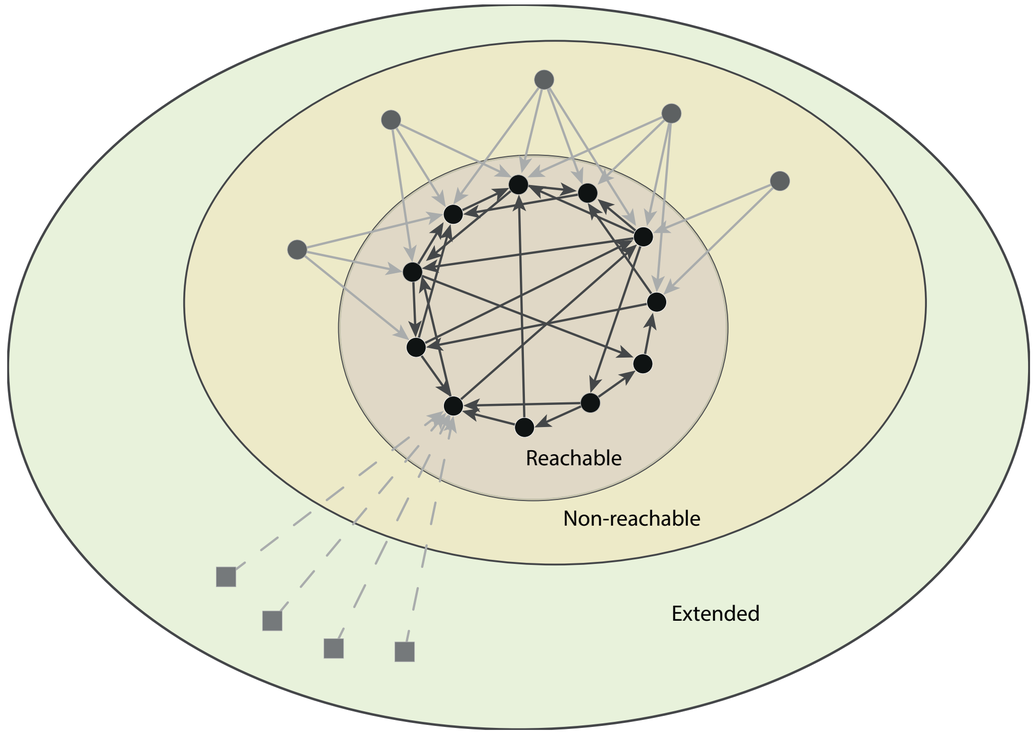


NETWORK TAXONOMY

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)

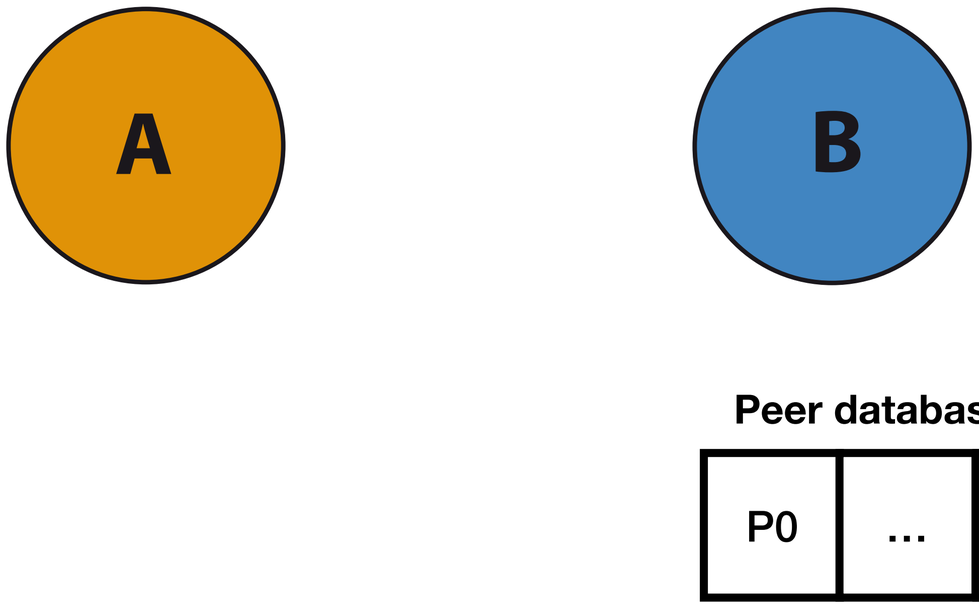


ADDRESS PROPAGATION (1/2)

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

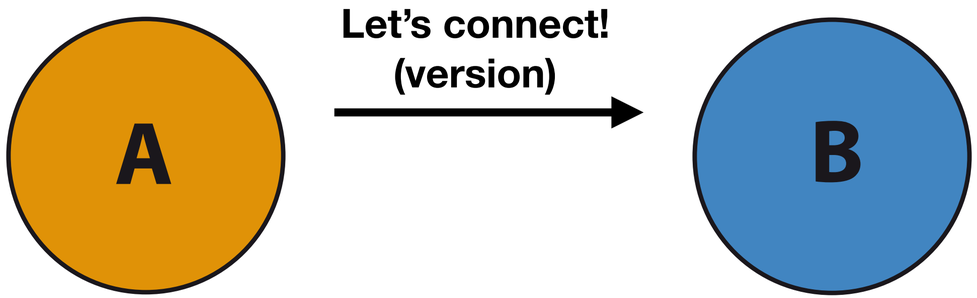
ADDRESS PROPAGATION (1/2)

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



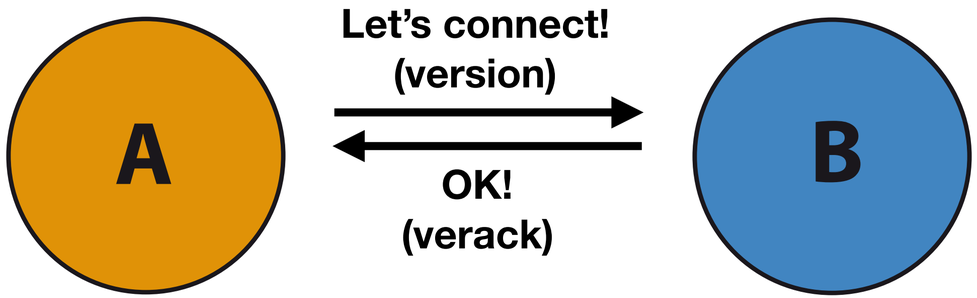
ADDRESS PROPAGATION (1/2)

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



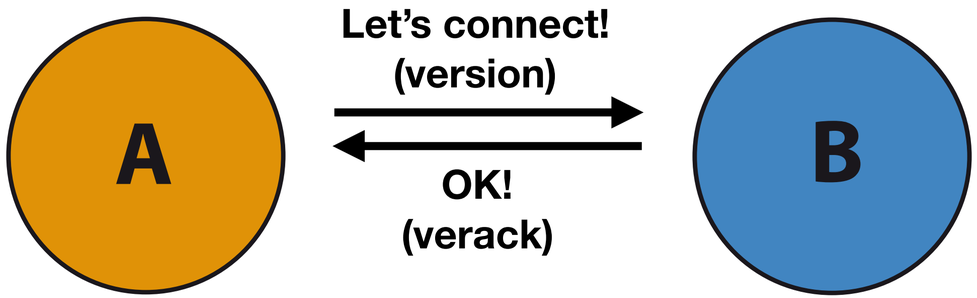
ADDRESS PROPAGATION (1/2)

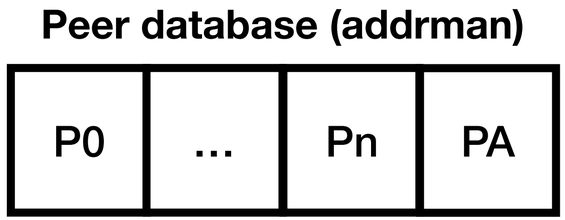
How does a node announce his presence to the rest of the network?



ADDRESS PROPAGATION (1/2)

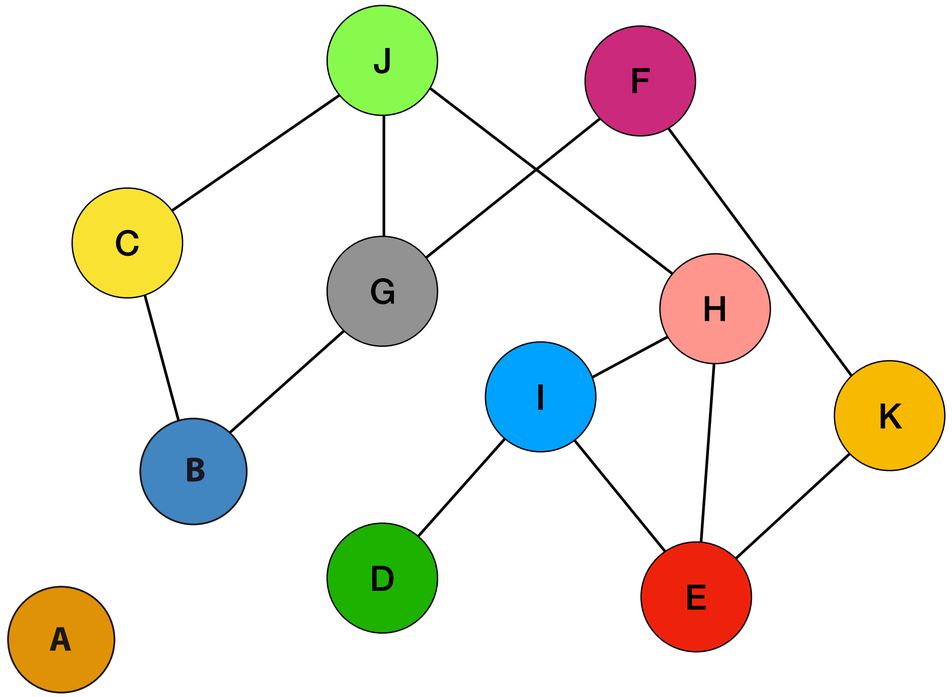
How does a node announce his presence to the rest of the network?





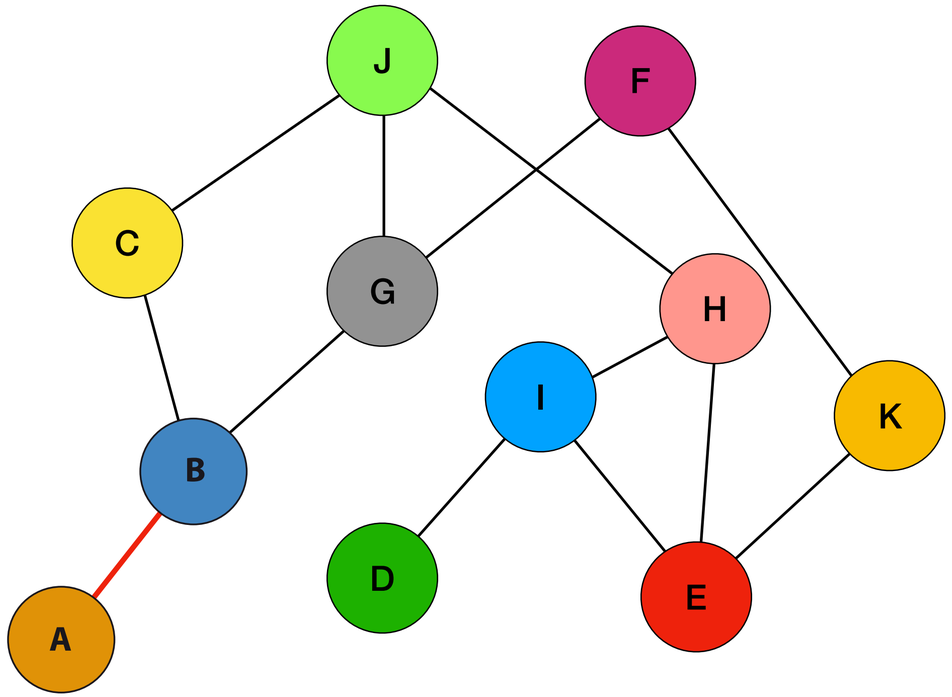
ADDRESS PROPAGATION (2/2)

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



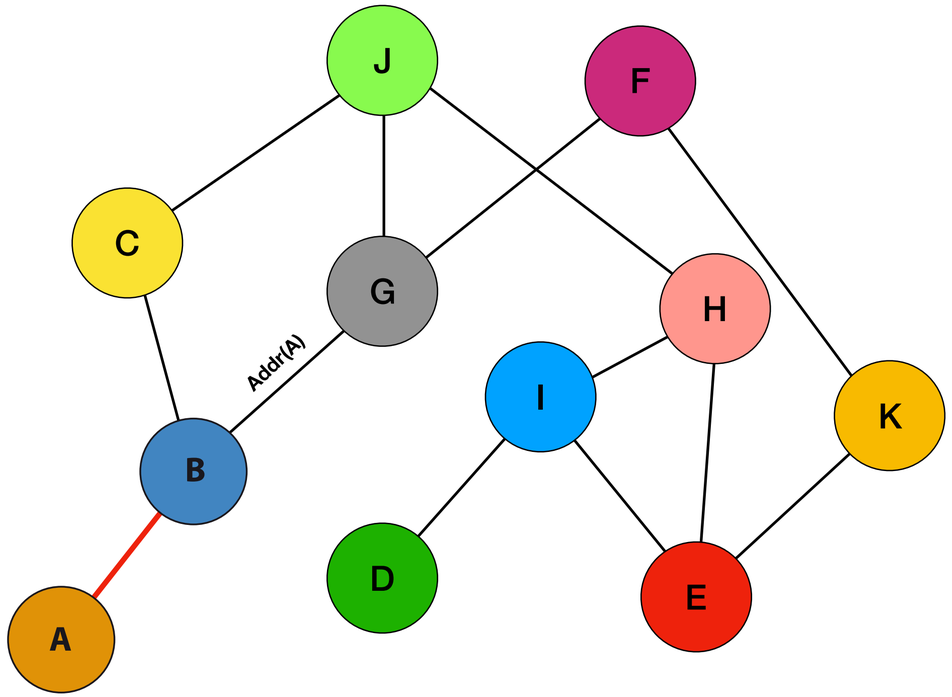
ADDRESS PROPAGATION (2/2)

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



ADDRESS PROPAGATION (2/2)

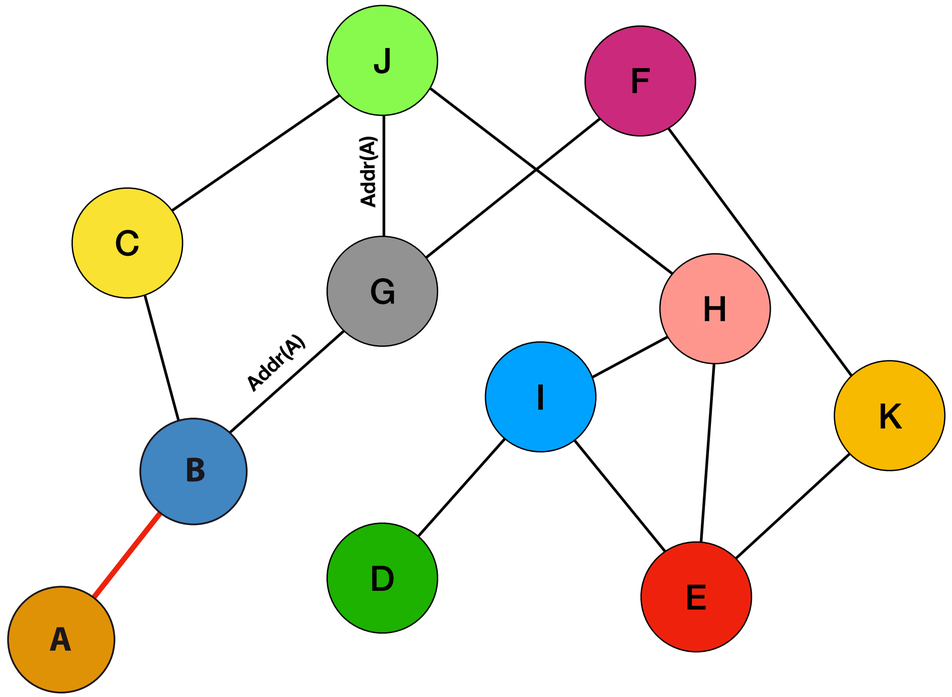
How does a node announce his presence to the rest of the network?



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

ADDRESS PROPAGATION (2/2)

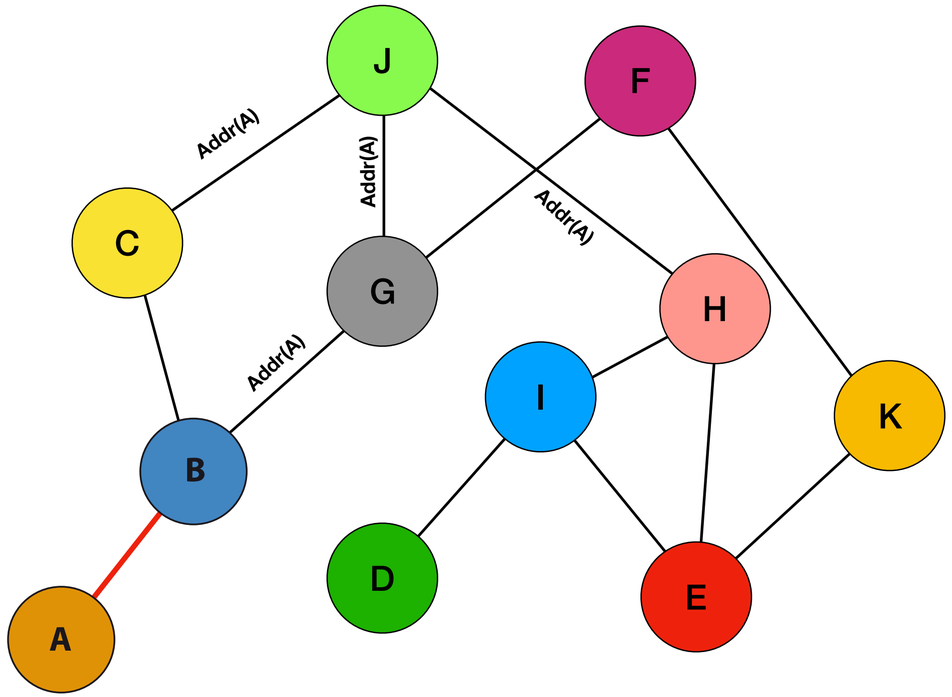
How does a node announce his presence to the rest of the network?



- 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

ADDRESS PROPAGATION (2/2)

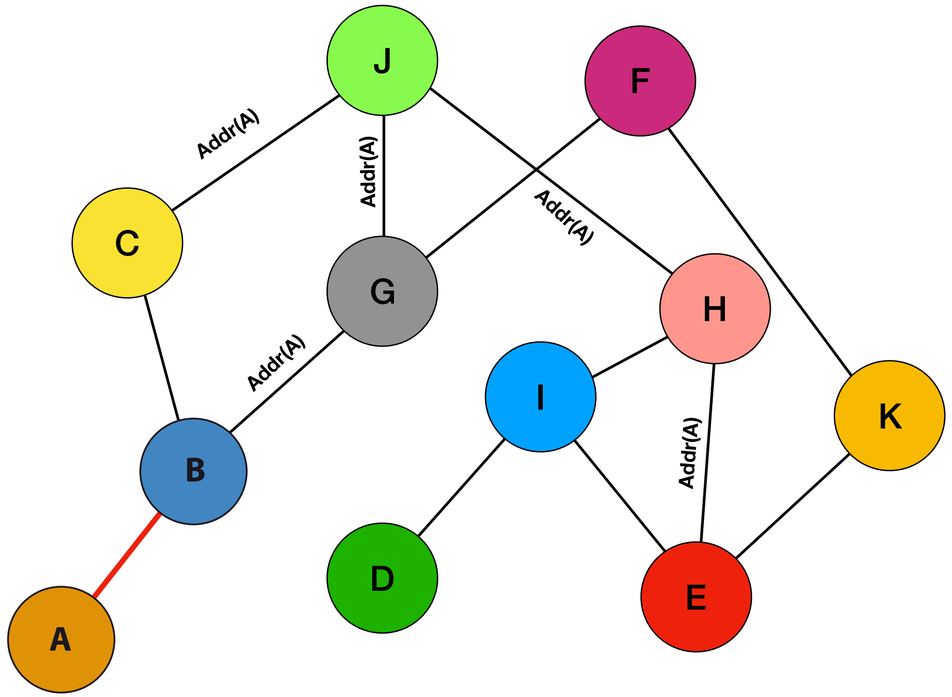
How does a node announce his presence to the rest of the network?



- 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…

ADDRESS PROPAGATION (2/2)

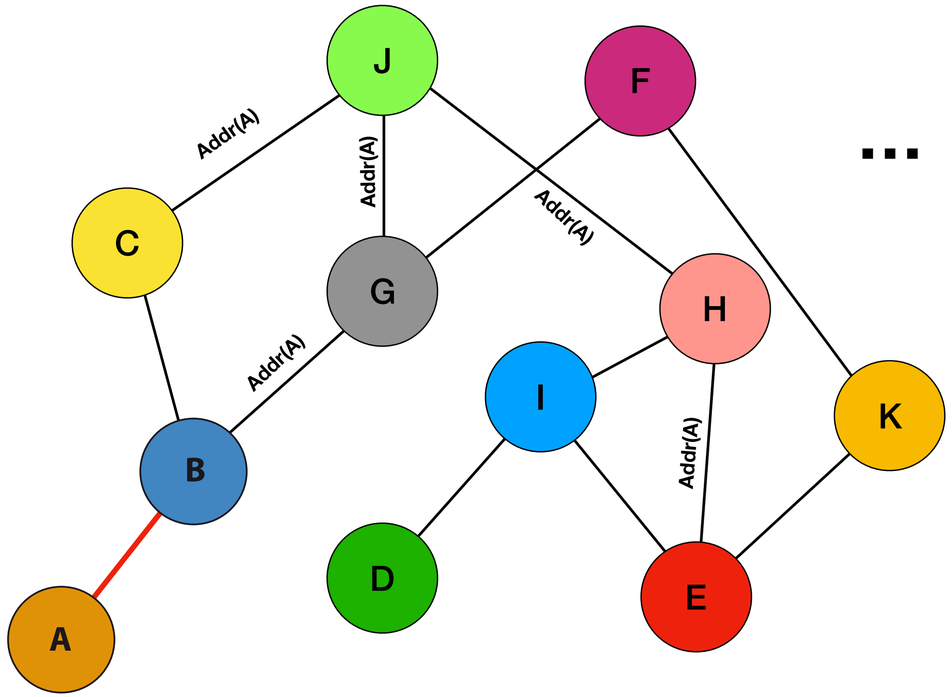
How does a node announce his presence to the rest of the network?



- 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…

ADDRESS PROPAGATION (2/2)

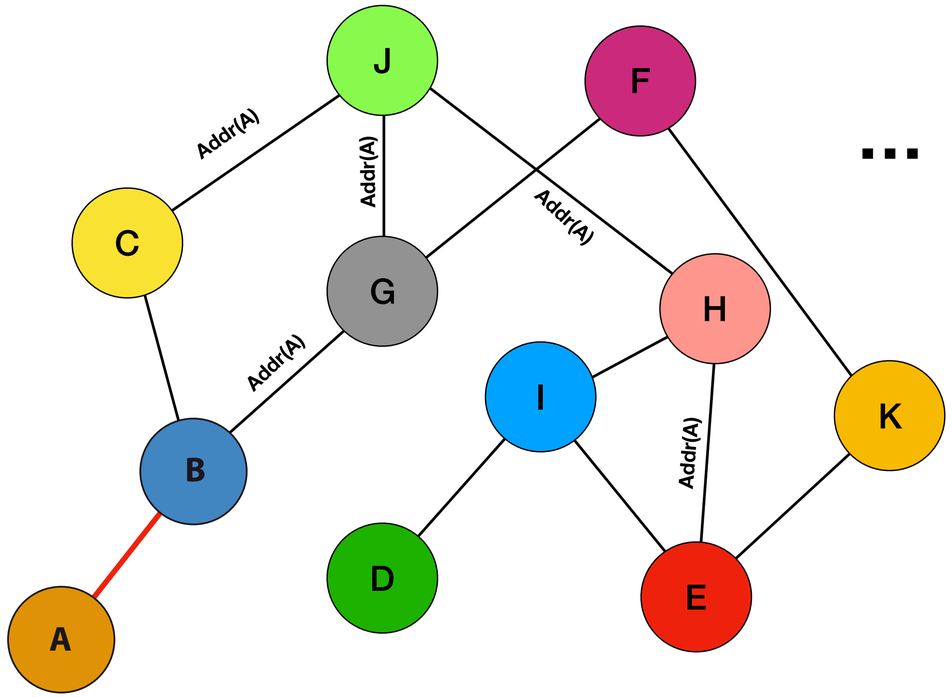
How does a node announce his presence to the rest of the network?



- 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…

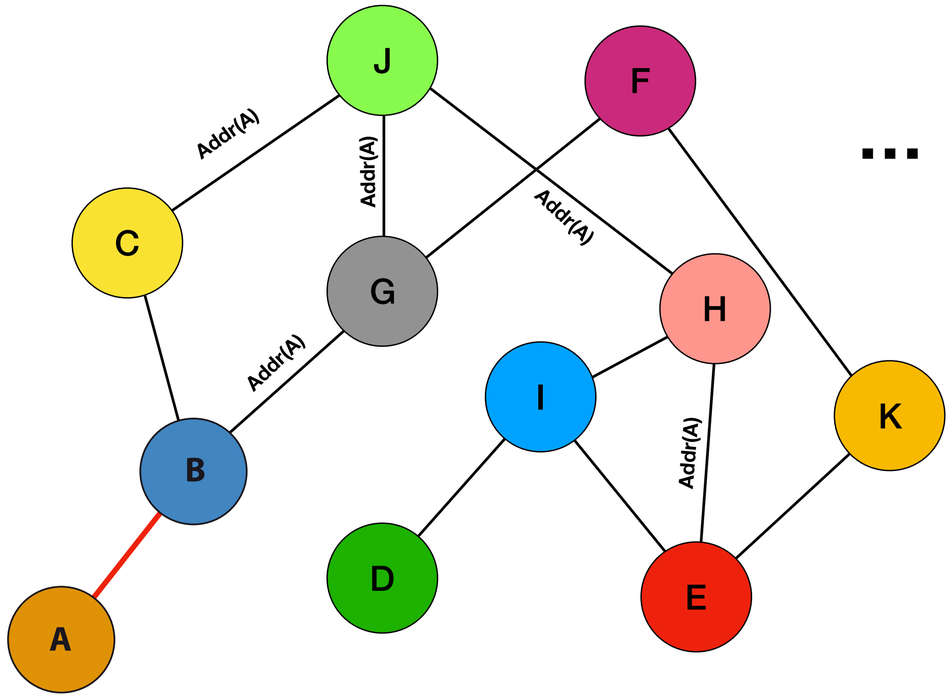
ADDRESS PROPAGATION (2/2)

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



ADDRESS PROPAGATION (2/2)

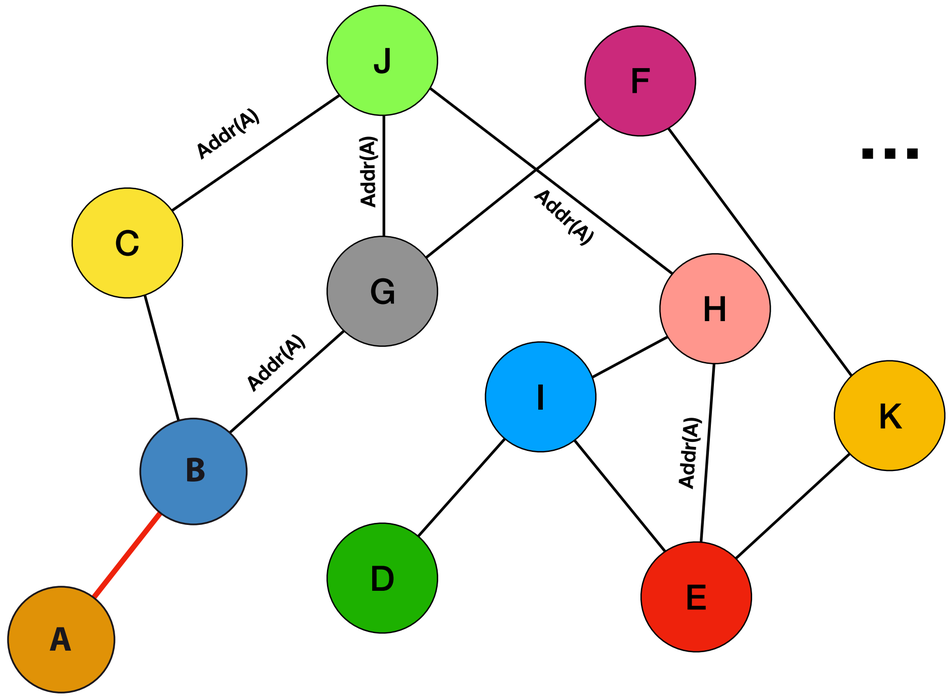
How does a node announce his presence to the rest of the network?



- The address will eventually be spread throughout the network

ADDRESS PROPAGATION (2/2)

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

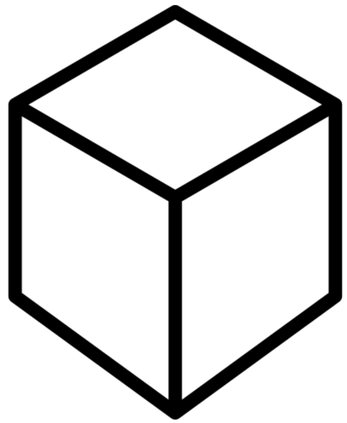
The address of a new node is propagated thought the network so all peers can know about it

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)

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:**

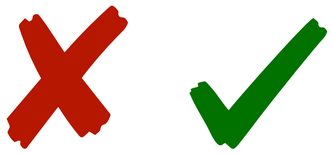
- Can **create transactions** that spend some of their bitcoins

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)

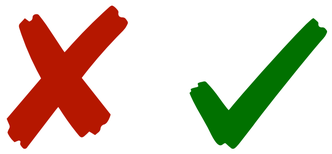


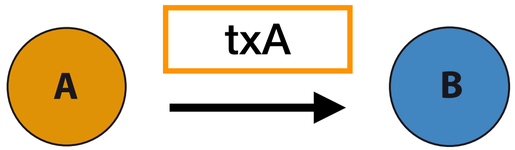
THE ACTORS (WHO?) (1/2)

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)

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

THE ACTORS (WHO?) (2/2)

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

Miners:

THE ACTORS (WHO?) (2/2)

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

Miners:

- Can everything a **peer** could do

THE ACTORS (WHO?) (2/2)

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

Miners:

- Can everything a **peer** could do
- Can **generate blocks** through a process known as mining

THE PURPOSE (WHY?)

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

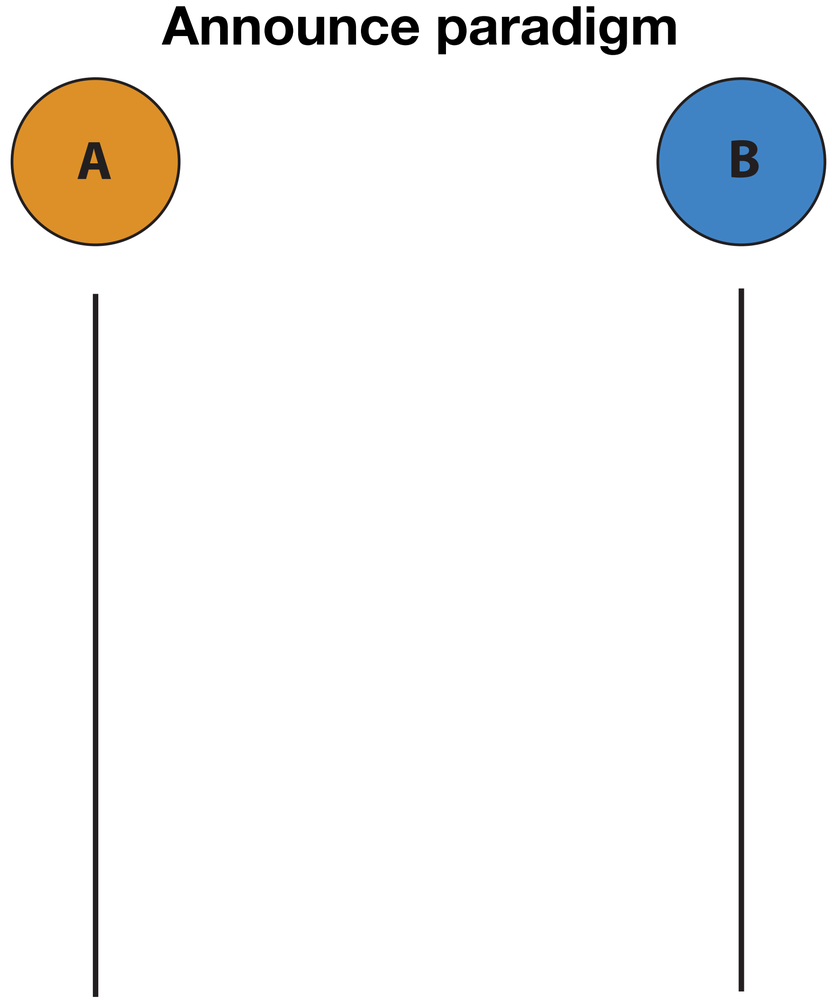
Miners generate blocks to obtain their reward (and also the transactions fees)

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?)

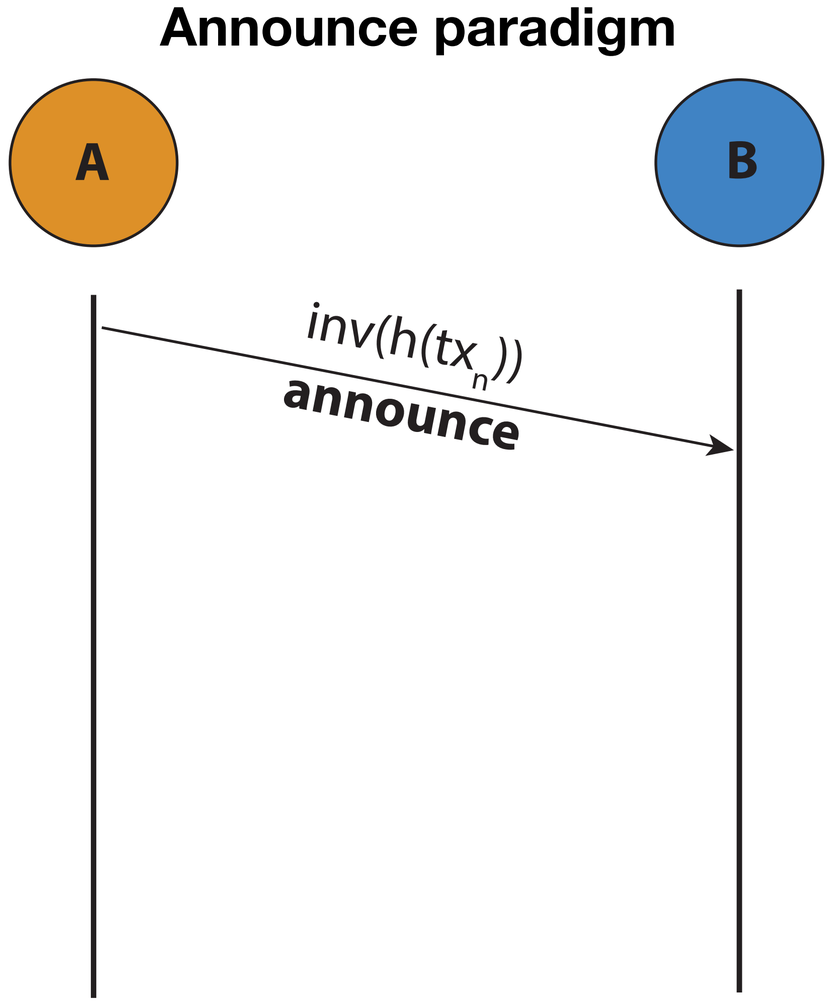
Items (transactions and blocks) are shared between peers in a push manner



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**)

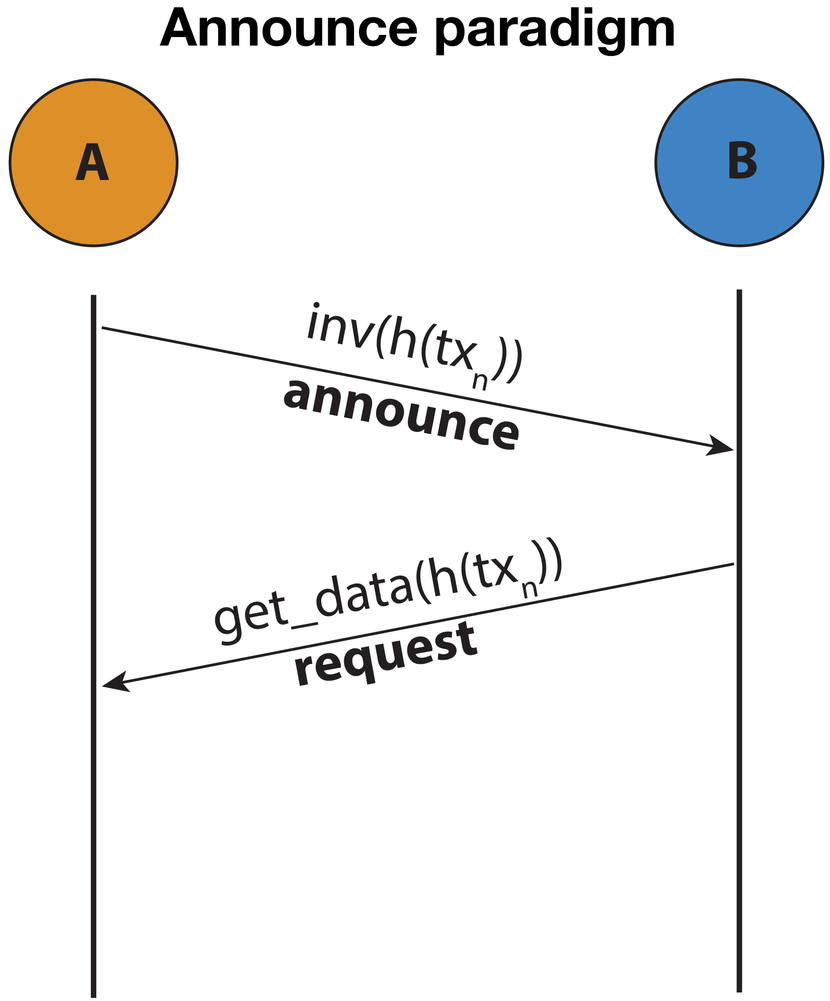


THE GOSSIP PROTOCOL (HOW?)

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 the item back to the announcer (**request**)



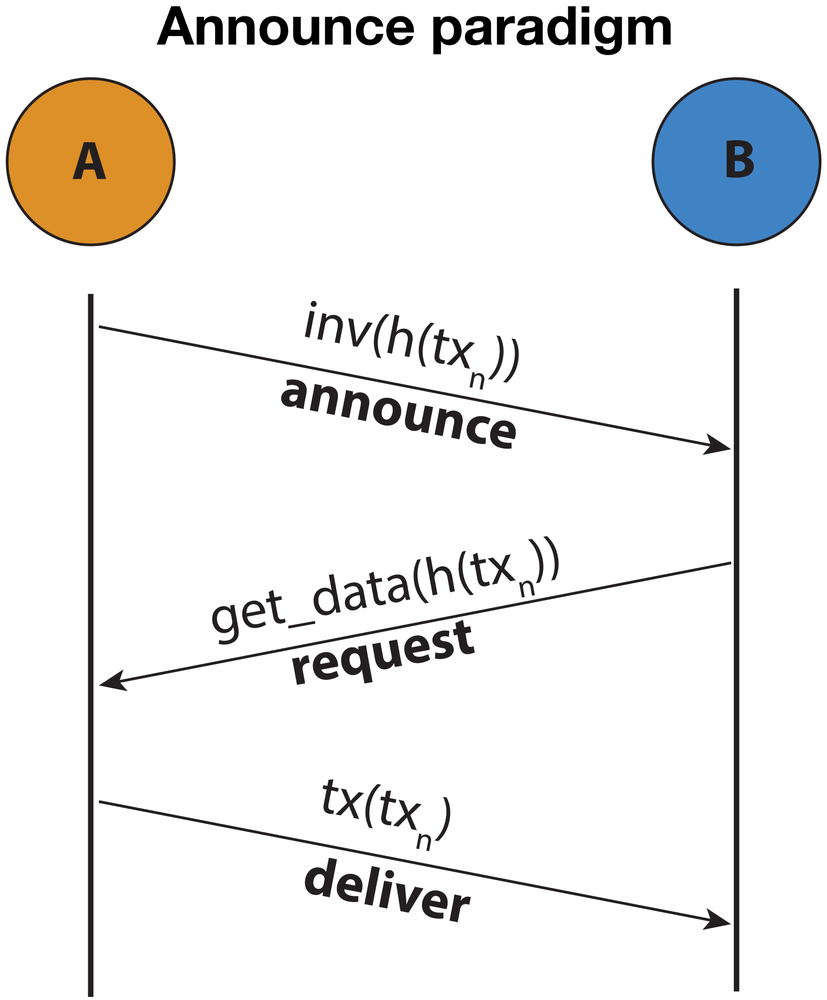
THE GOSSIP PROTOCOL (HOW?)

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 the item back to the announcer (**request**)

Upon receiving a request of a known item, a node will reply back with it (**deliver**)



NEXT TIME (WEEK 3)

Data propagation

Network based attacks

Network topology