



# Classical Consensus

# Possible Timing Models

## Asynchronous

- A sent message will *eventually* be delivered

## Synchronous

- Guarantees on the time of delivers
- e.g. message sent at time  $t$ , will be delivered at time  $t+x$

## Eventually Synchronous

- A mix between both, where there is a *known upper bound* on the delivered time which is *variable*.

# Possible Fault Models

Up to **f out of N processes can fail**

- Typically  $f < N/2$ ,  $f < N/3$

Honest nodes

- Remain available and do not behave byzantine

Availability failure

- A node might suddenly crash/Internet connectivity drop

Byzantine failure

- Malicious failure of an adversary

# Broadcast Models

## Consistent Broadcast

- A corrupted sender implies that not every party might terminate/deliver a request.

## Reliable Broadcast

- Sender emits value  $v$ 
  - Termination: if sender honest, correct party outputs  $v$
  - Reliability: every correct party outputs  $v$
  - Consistency: two distinct parties output  $v_1, v_2$  and  $v_1=v_2$

## PBFT, Paxos, RAFT, et al.

### Leader election

- Every node can become eventually a leader (e.g. round-robin)

### Safety

- The output is guaranteed to be consistent, even under an unstable network and malicious leader
- Although asynchronous eventually synchronous

### Liveness

- If no progress, new leader elected
- If network stable and honest leader then liveness

# Known Impossibility Results

Fischer, Lynch, Patterson (1985)

- FLP result
- In a fully asynchronous system, there is no deterministic consensus solution that tolerates one or more failures
- No algorithm can always reach consensus in bounded time.

Implication

- Timing assumptions are required for every protocol
- Randomness is crucial

## **Given any protocol**

Is the impossibility result respected?

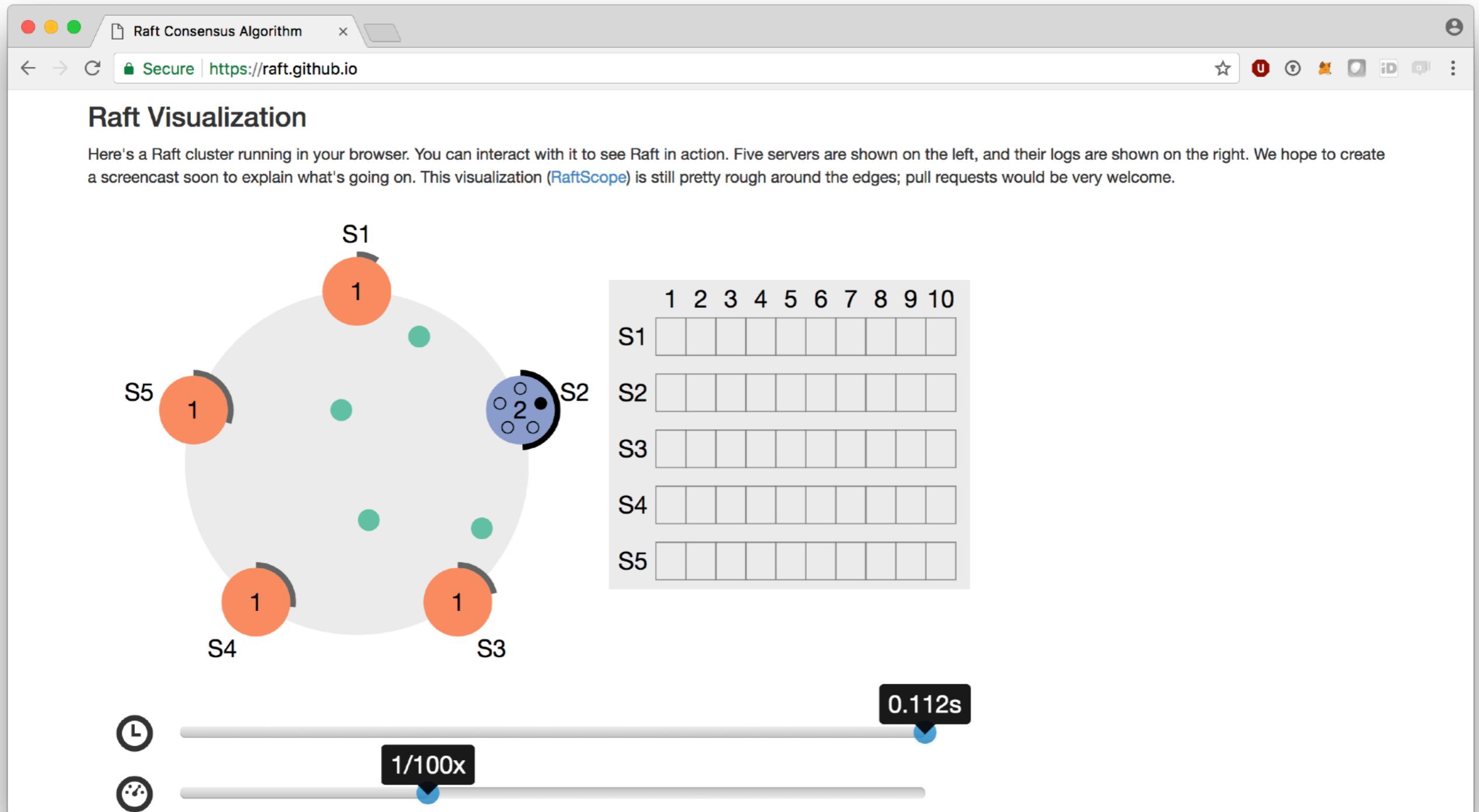
What's the message complexity/number of rounds?

How many nodes are allowed to fail?

Where does the randomness come from?

Can an adversary manipulate the randomness/become leader?

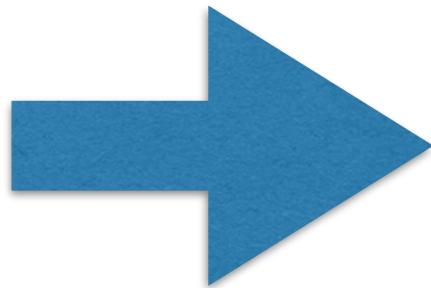
# RAFT



<http://thesecretlivesofdata.com/raft/>    <https://raft.github.io/>

## How does this relate to Bitcoin?

- No need for a final consensus output
- Block/transaction reward as incentive to participate
- **The participating nodes do not need to be known upfront!**



Fundamentally different to the results of years of research

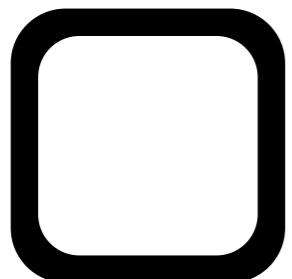


# Blockchain Bootstrapping

# Bootstrapping



where is the chain?



Genesis Block

# Bootstrap Methods



where is the chain?



## DNS (Domain Name System) Bootstrap

- A few dedicated nodes that serve other IP addresses

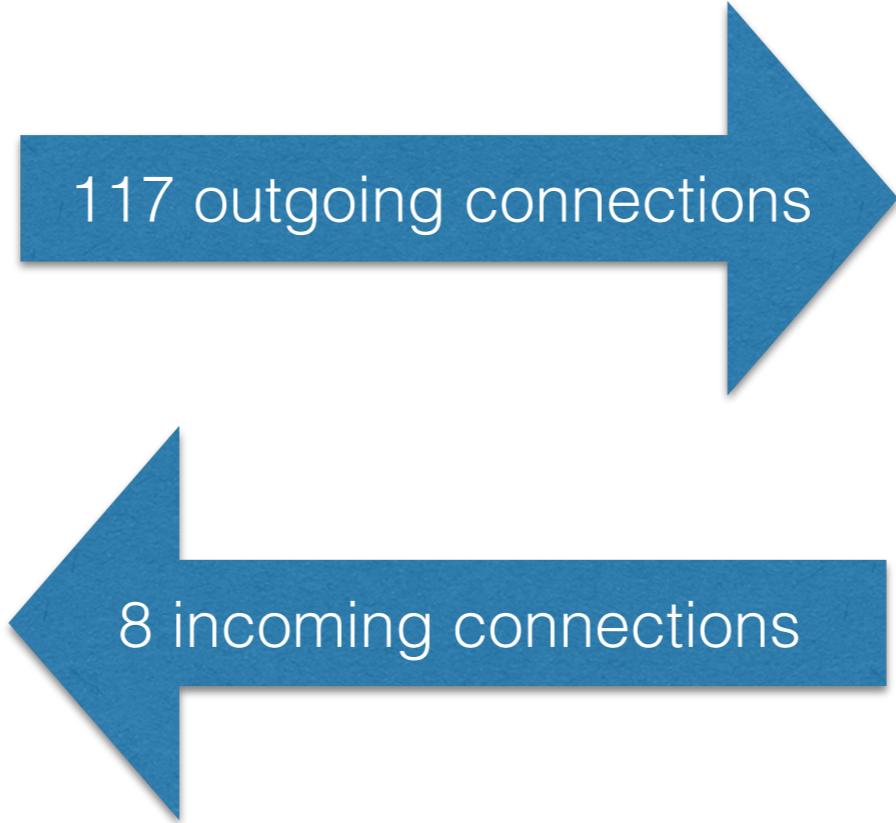
## Hardcoded IP's

- These IP's need to be static

## Chat/Forums

- Ask around for IP's of participants

# Bitcoin Node



## TCP based

- No authentication/encryption

## Incoming Packet Validation

- Signature validation
- Consistency, UTXO validation
- Denial of service protections



# Network Denial of Service Protection

# DoS Protection Mechanism

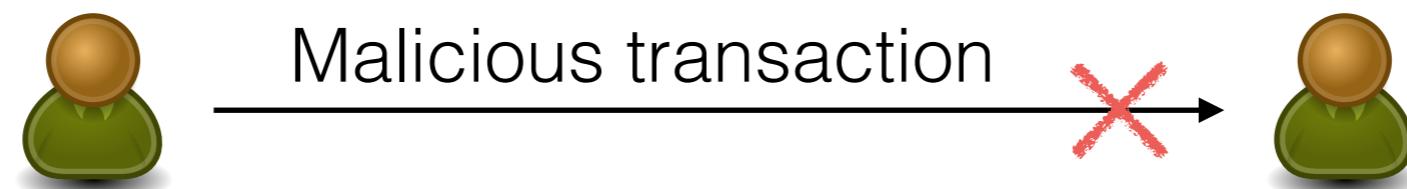


Sanity checks (lower penalty)

- Size
- Encoding

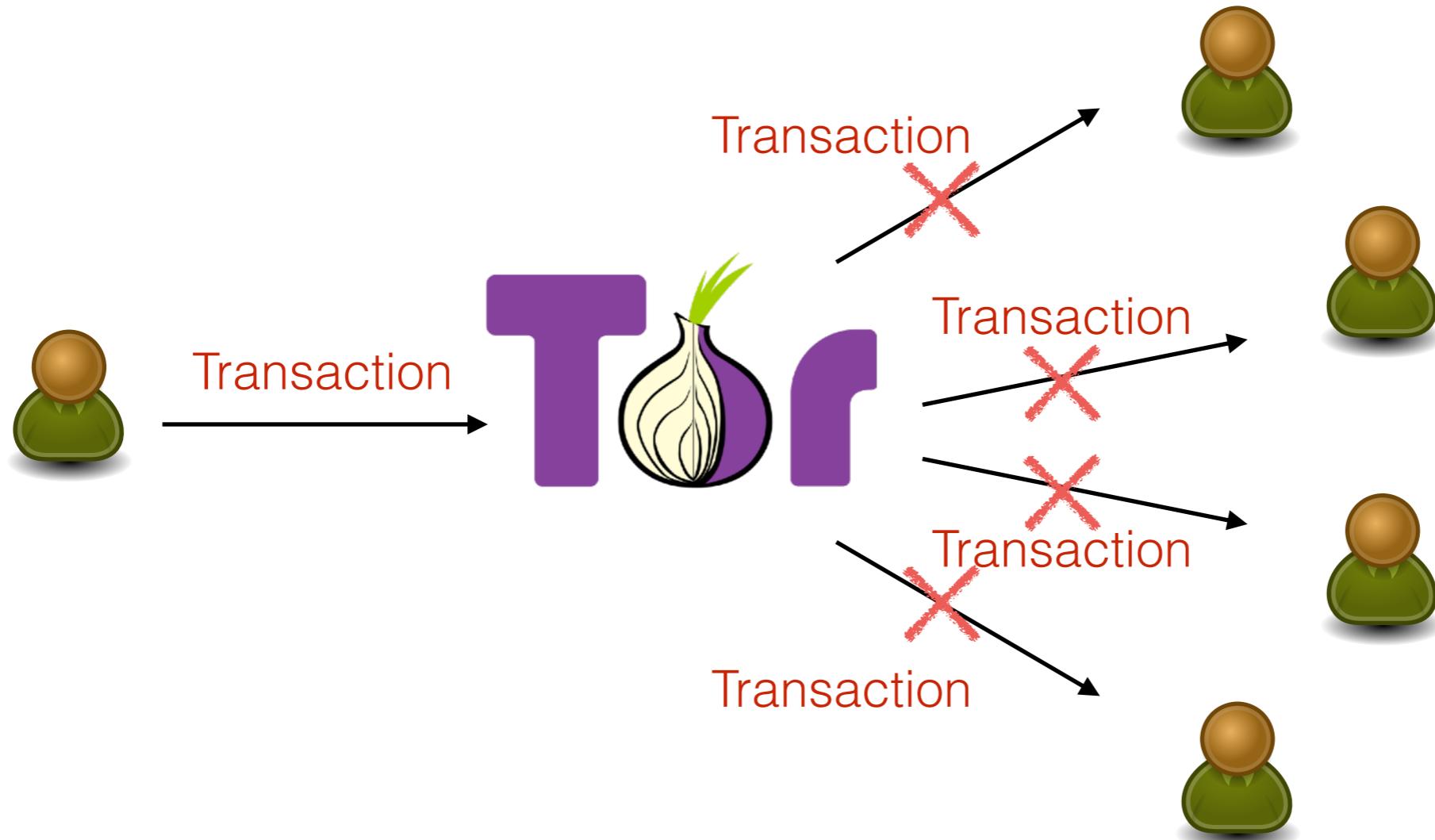
Critical points (penalty 100)

- Signature validation
- UTXO/double spending validation



Banned for 24 hours!

# DoS Protection Mechanism is a weakness

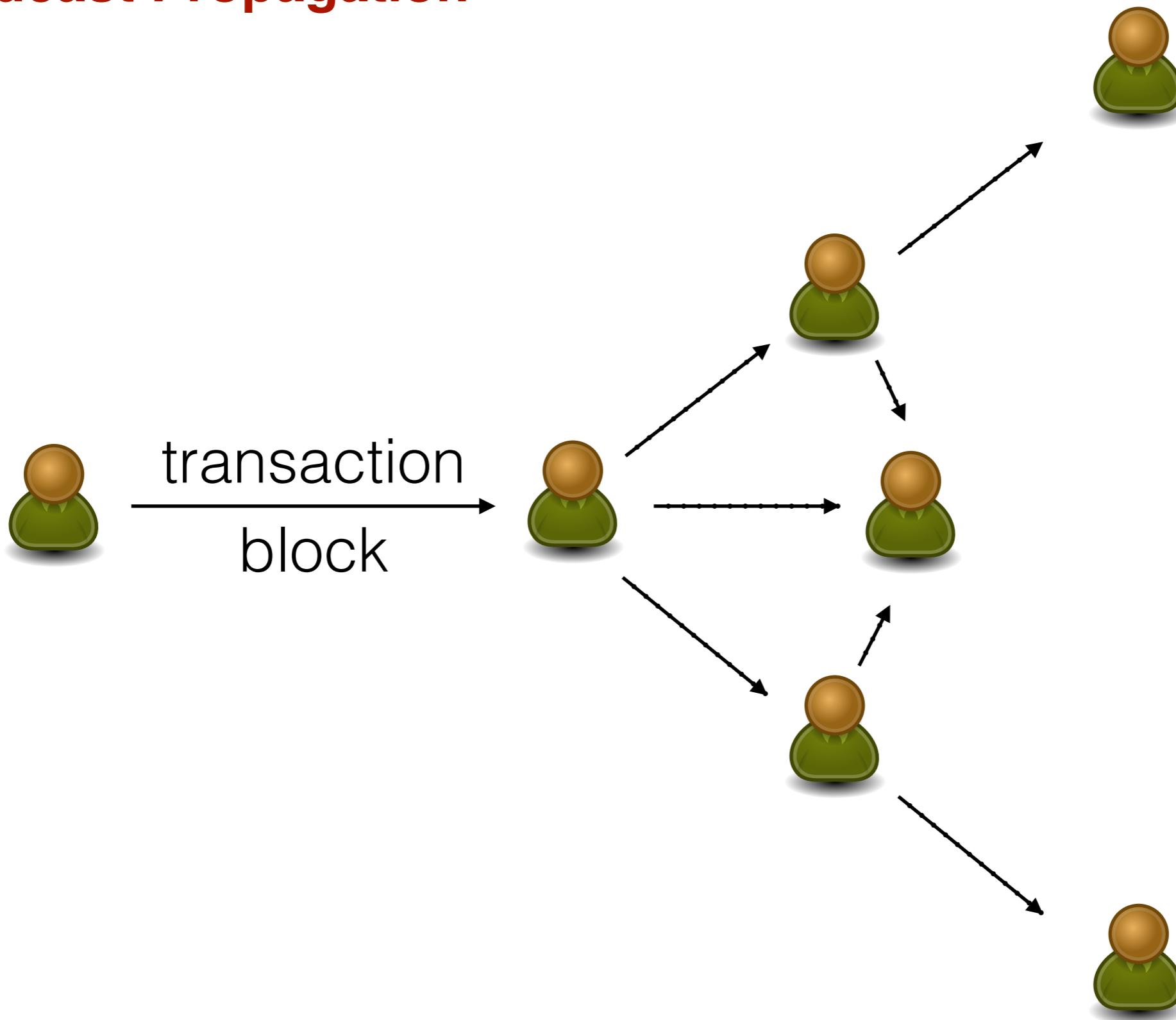


All Tor exit nodes banned through Bitcoin.

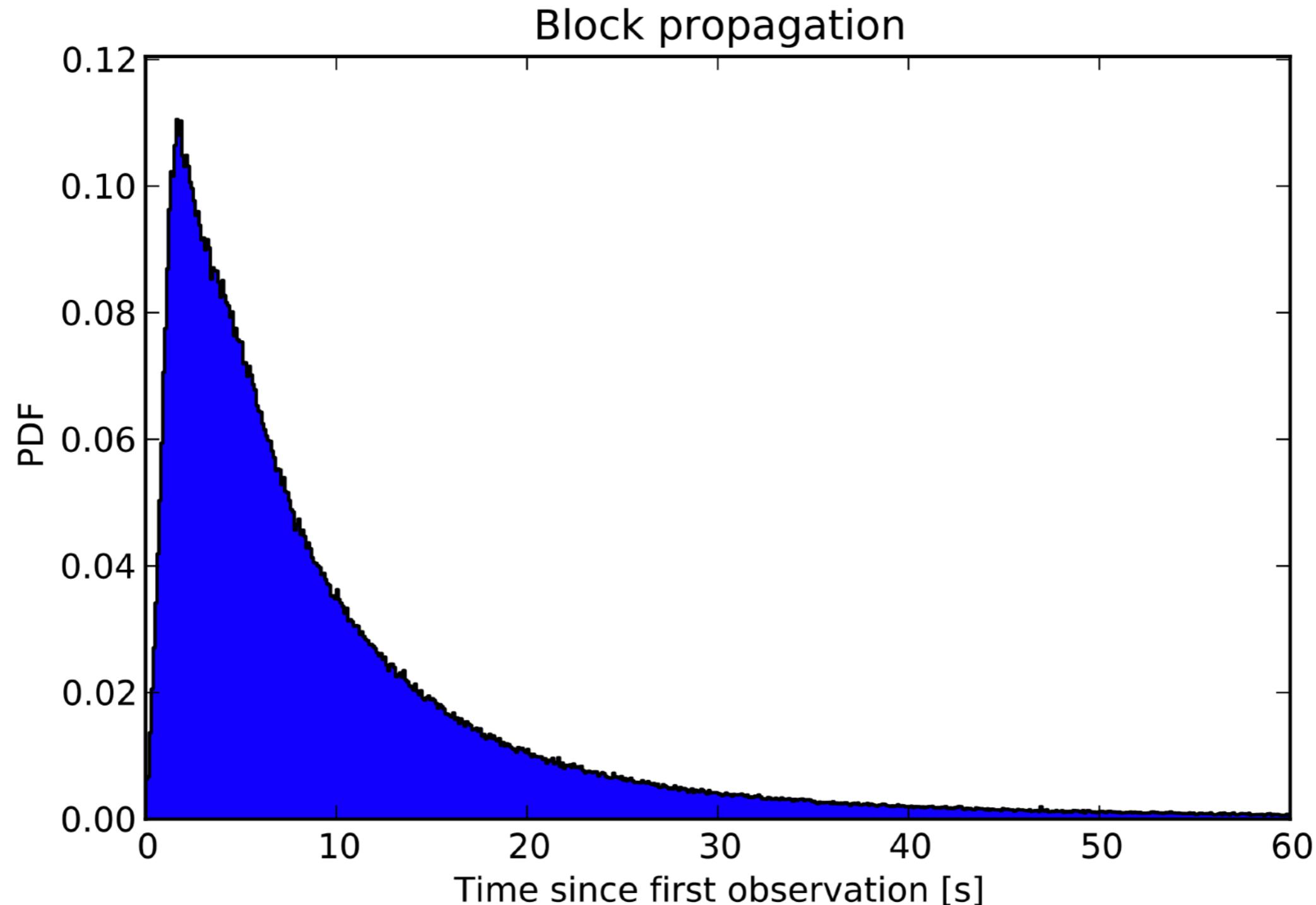


# Network Gossip Protocol

# Broadcast Propagation

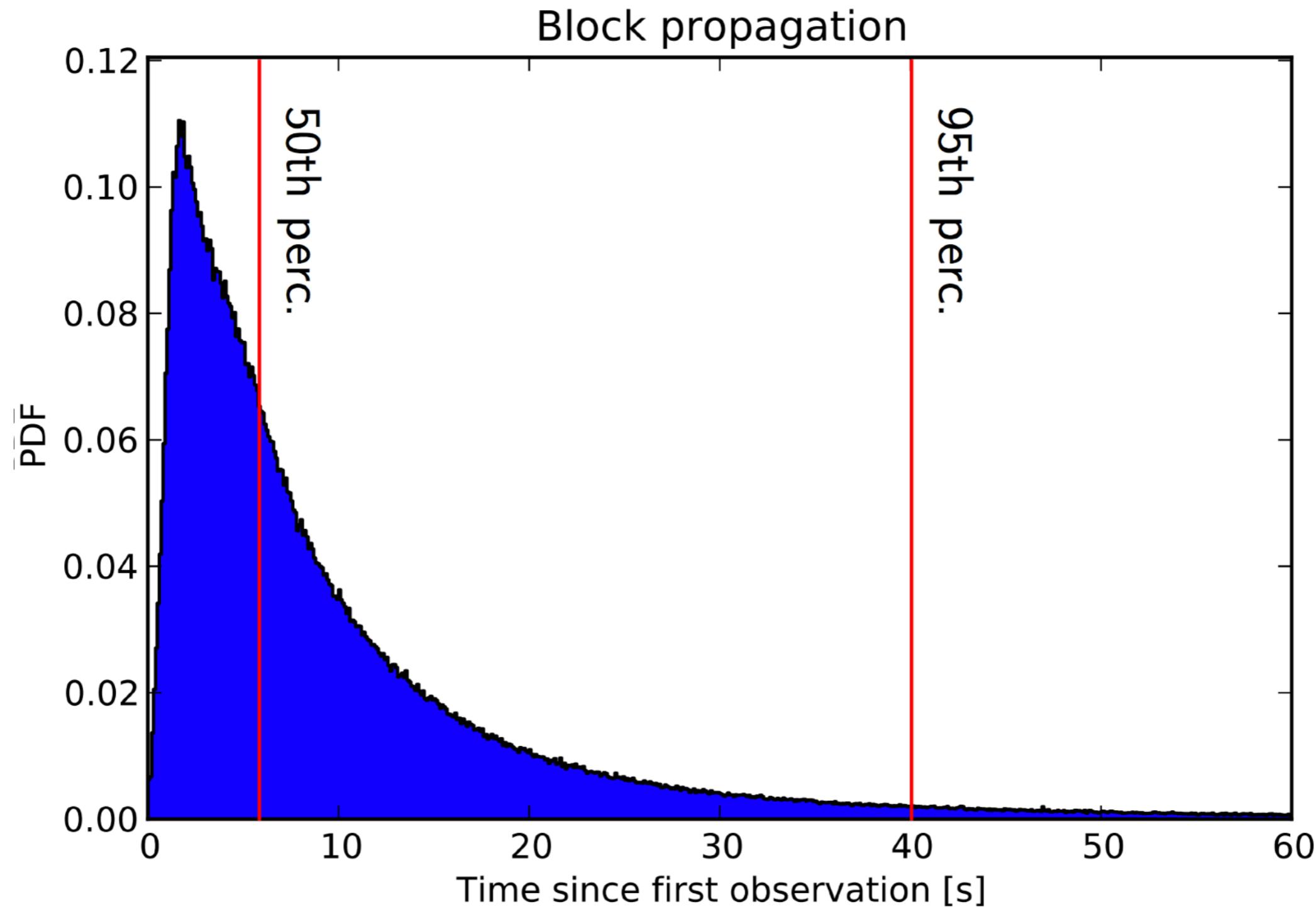


## Broadcast Propagation



\* Christian Decker et al., Information Propagation in the Bitcoin Network

## Broadcast Propagation



# Propagation Methods

## Standard

- Send first the hash of an object, transaction/block
- Recipient requests the object
- Sender transmits the object

## Send Headers

- Send first the block header (no more block hash)
- Then block

## Unsolicited Block Push

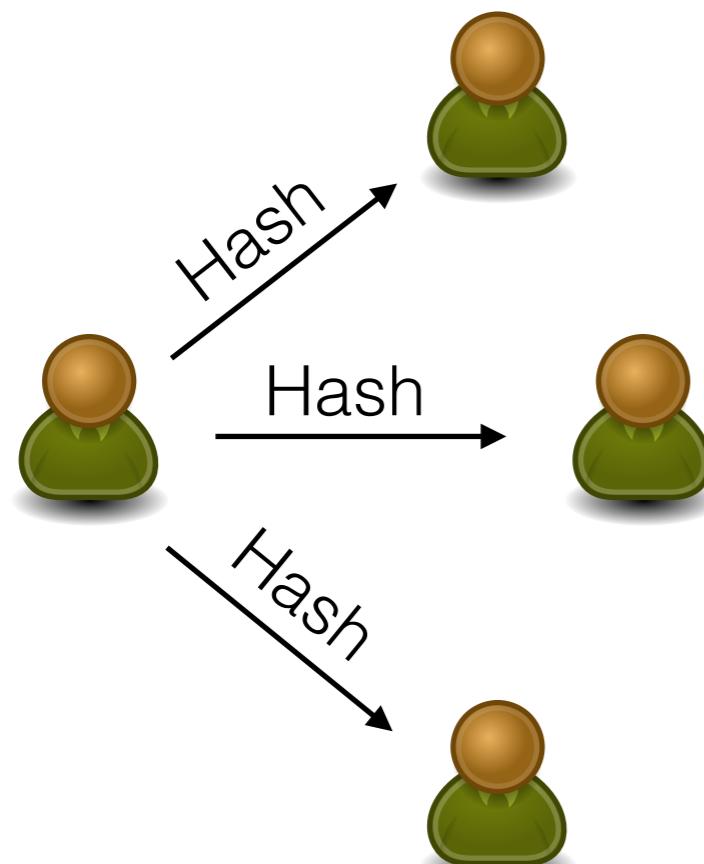
- Miners can push a block directly, without pushing the header

## Fibre (Fast Internet Bitcoin Relay Engine) Network

- Optimized network for miners

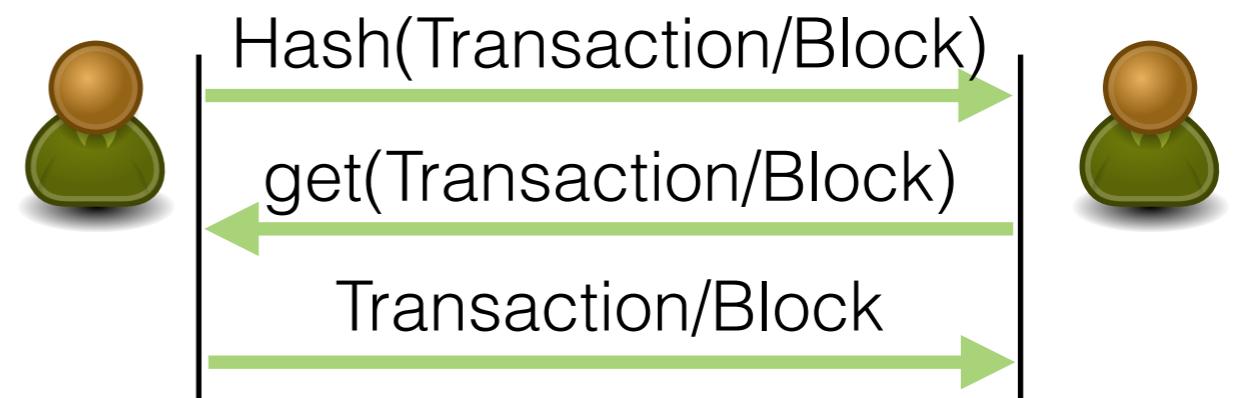
# Standard Transaction/Block advertisement

1. Transaction/Block hash broadcast



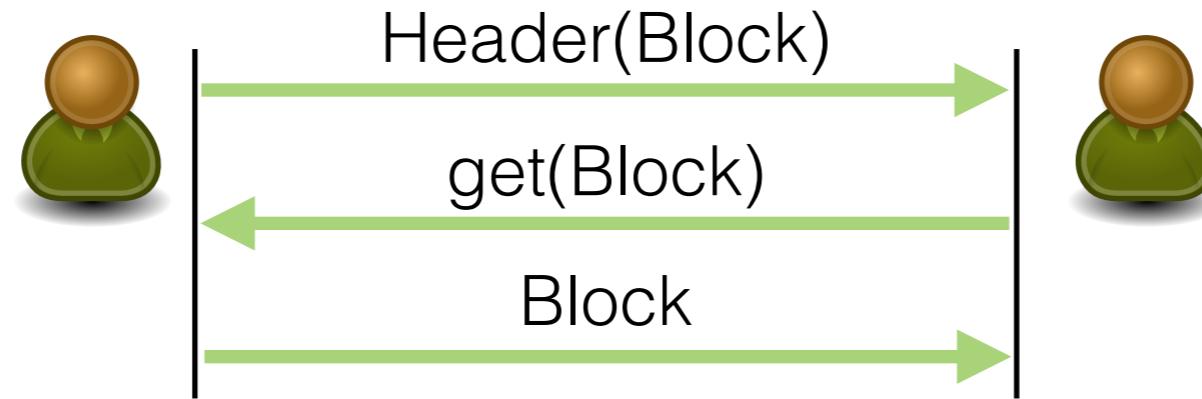
Broadcast

2. Transaction/Block request



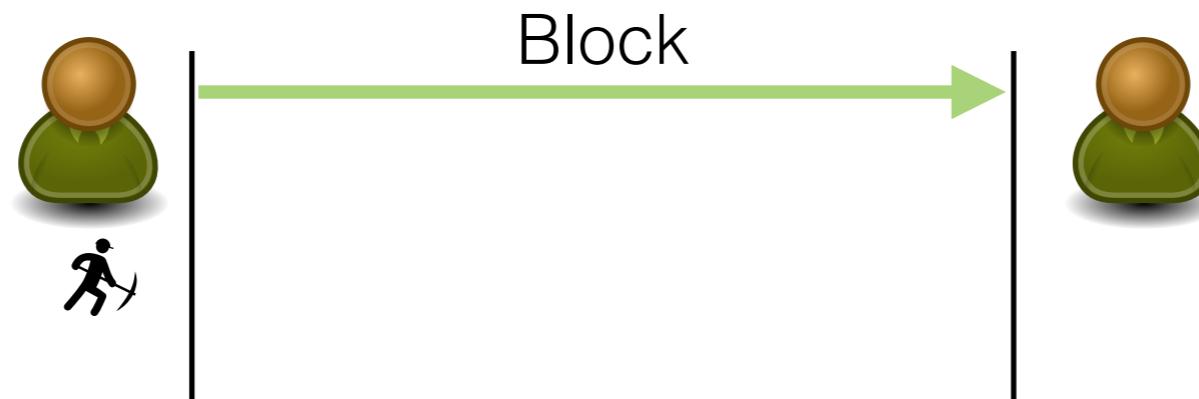
Request from only 1 peer!

## Send Headers Block advertisement



Header in Bitcoin about 80 bytes,  
hash about 36 bytes

# Unsolicited Block Push



Nobody else knows about the block

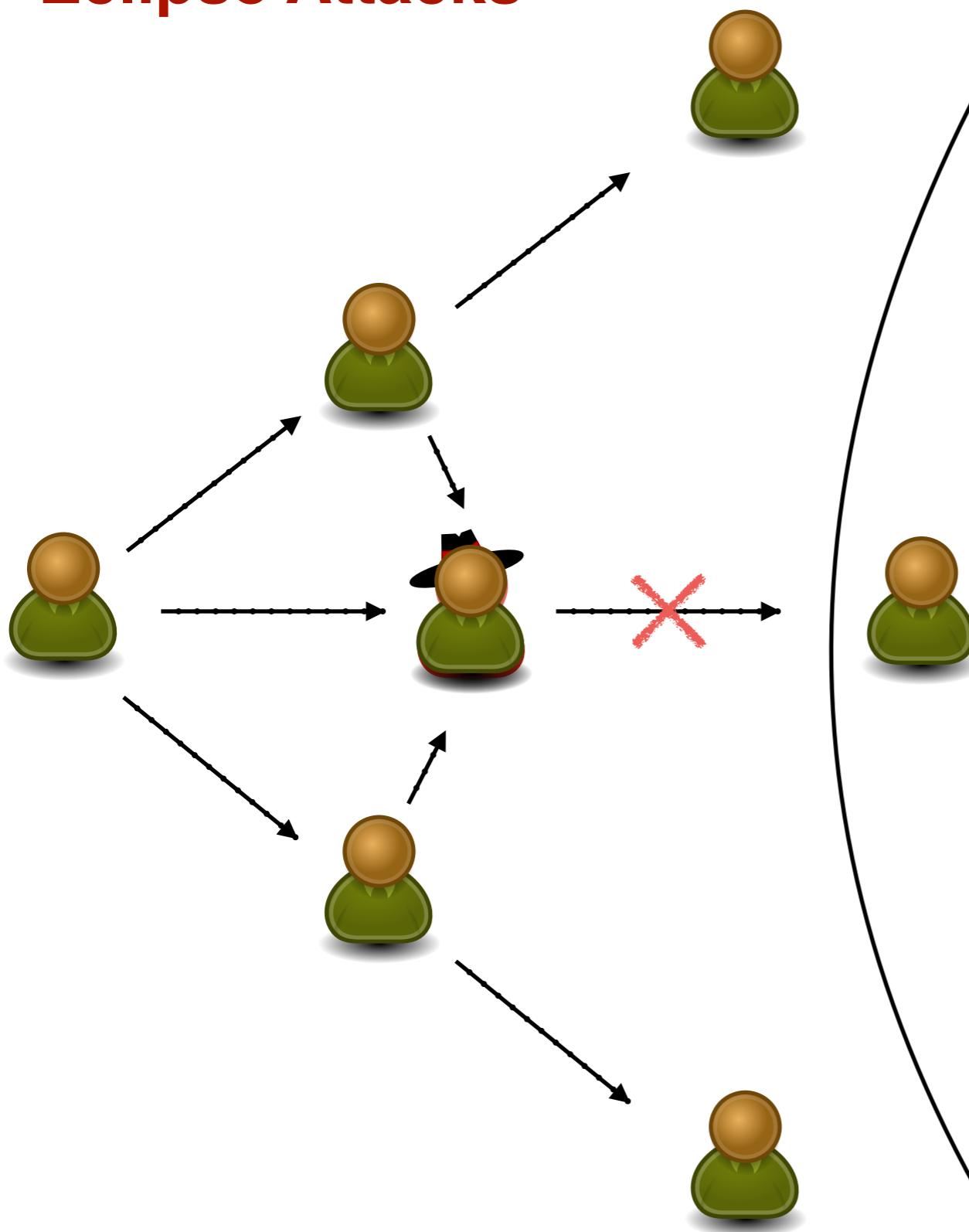
## Bitcoin Fibre

- Fibre node sends a short block sketch
  - List of short hashes, lengths
- Receiver can reconstruct block based on memory pool and construct a block with holes
- Fibre sender breaks block into chunks and sends error correction data
  - Receiver can reconstruct block, without the sender knowing what's missing.
- Once received and reconstructed the block, the fibre node emits novel chunks —> no redundancy.
- UDP based —> no ramp up



# Network Security - Eclipse Attacks

# Eclipse Attacks



1 connection sufficient  
No victim restart necessary

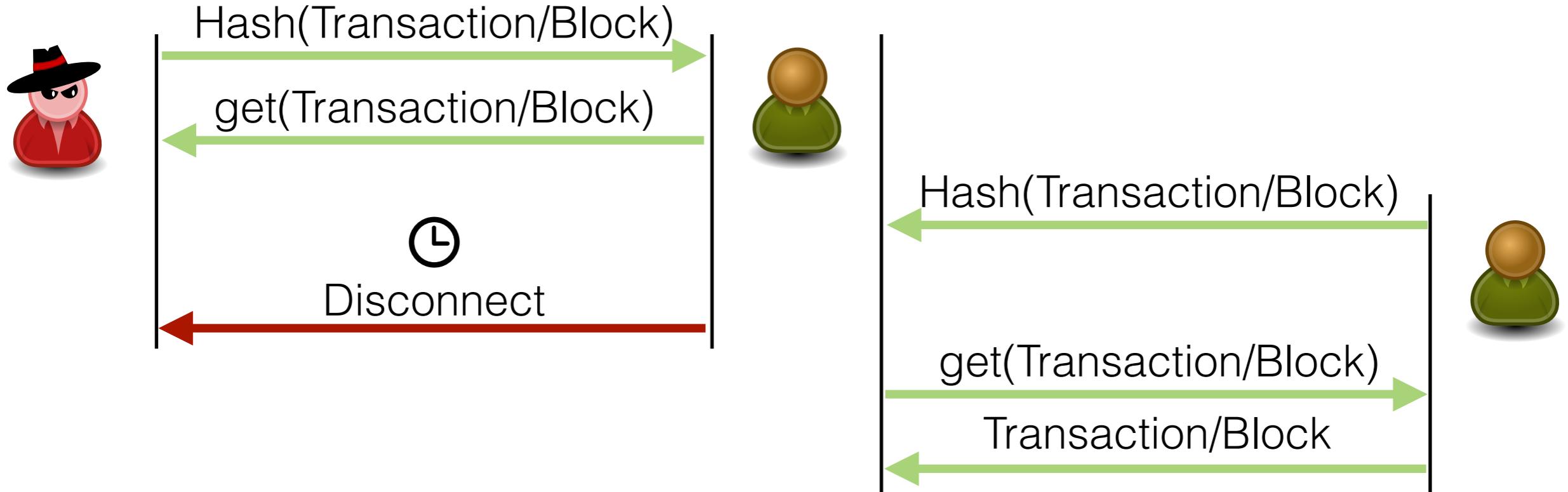
Eclipse attacks  
Heilman et al., Usenix '15  
Monopolize connections  
Spamming addresses  
Forcing node restart  
Requires many bots

Denial of Service

Double Spending



## Request timeouts



Block timeout: 20 minutes

Transaction timeout: 2 minutes

# Implications

## Adversary

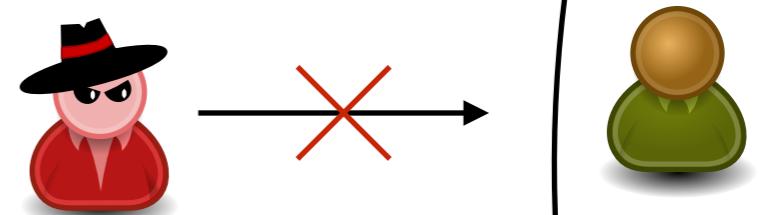
- Blinds victim from blocks and transaction > 20 min
- Experimental validation

## Impact

- **Double spend transactions**
- Aggravated selfish mining
- **Network wide Denial of Service**

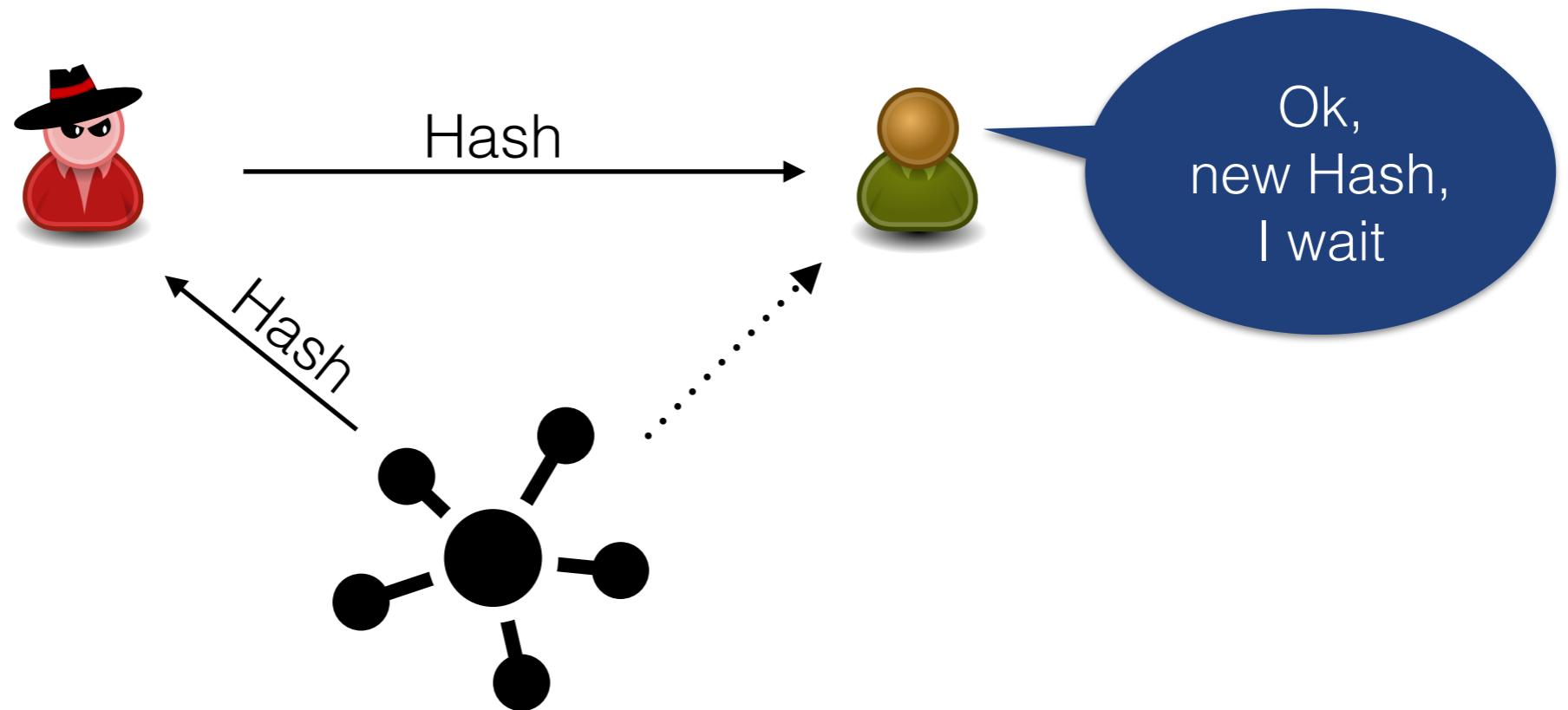
## Mitigations

- **Hardening measures**
- Estimate waiting time for secure transactions



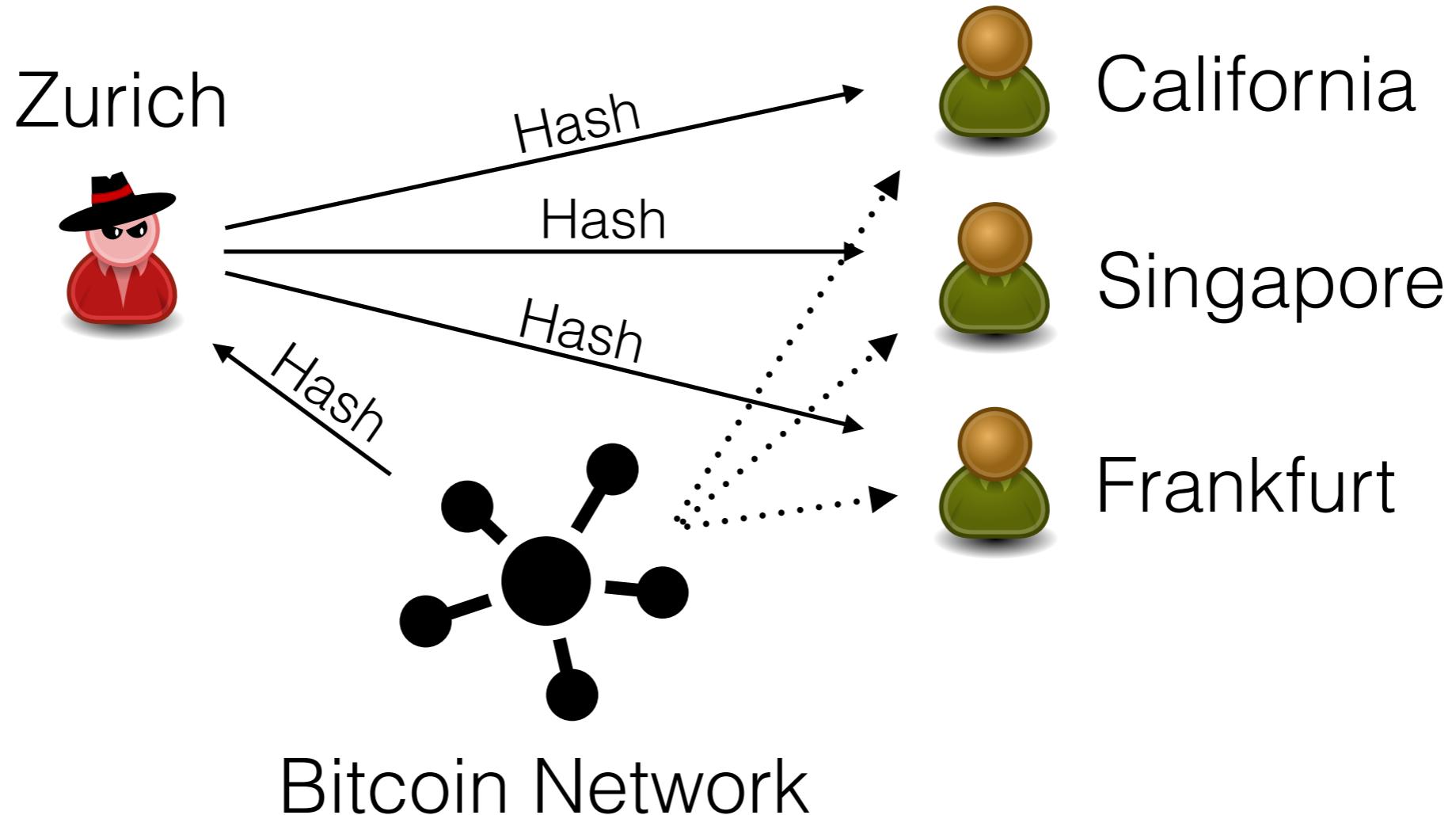
## Necessary requirements

1. Must be **first** peer to advertise Transaction/Block



2. Victim should wait
  - Block timeout: 20 minutes
  - Transaction timeout: 2 minutes

# Being First

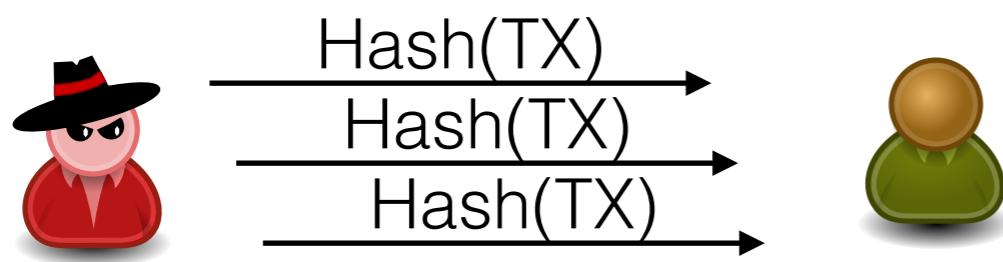


Connections of Adversary	40	80	200	800
Connections of Victim	40	40	40	40
Average success in being first	$0.44 \pm 0.14$	$0.57 \pm 0.20$	$0.80 \pm 0.14$	<b><math>0.89 \pm 0.07</math></b>

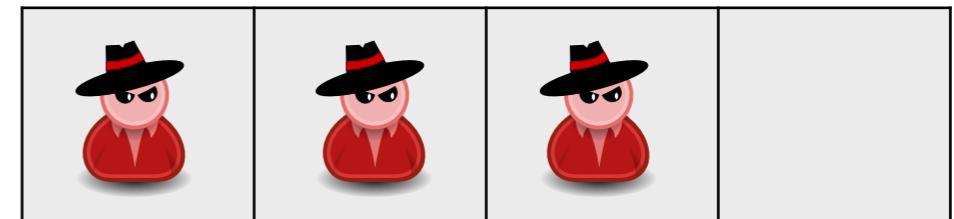
# Waiting

## Transactions

- After 2 minutes request from other peer (FIFO)



FIFO queue



→ 6 minutes timeout

## Blocks

- After 20 minutes disconnect and do nothing
- If received header, disconnect and request block from another peer



# Eclipse Attacks - Delaying Blocks

## **Block delivery time > 20 min**

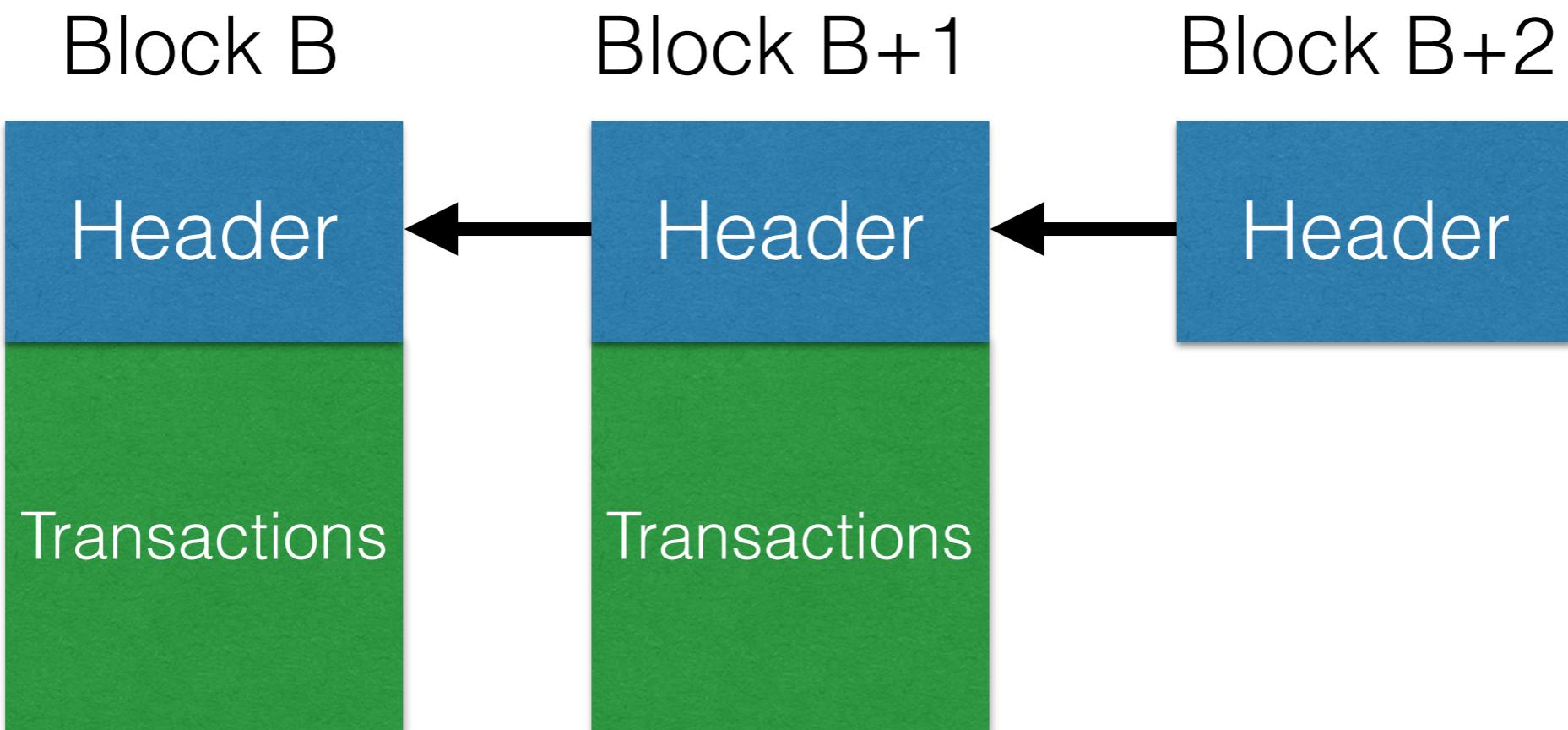
### 1. Requirement for victim



- Must not receive block header
- Must not receive version message

## Block delivery time > 20 min

Header-first synchronization



## **Block delivery time > 20 min**

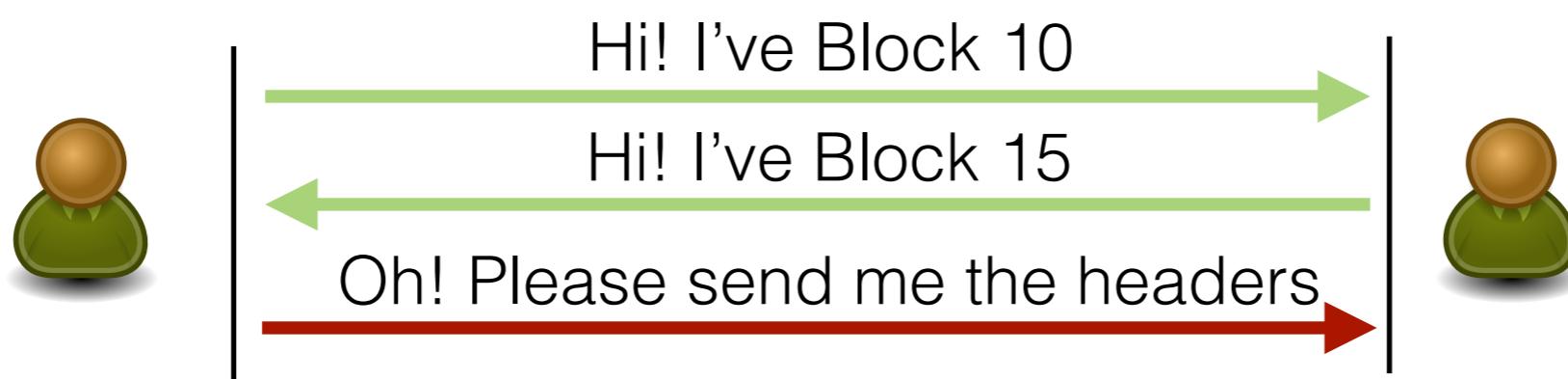
### 1. Requirement for victim



- Must not receive block header
- Must not receive version message

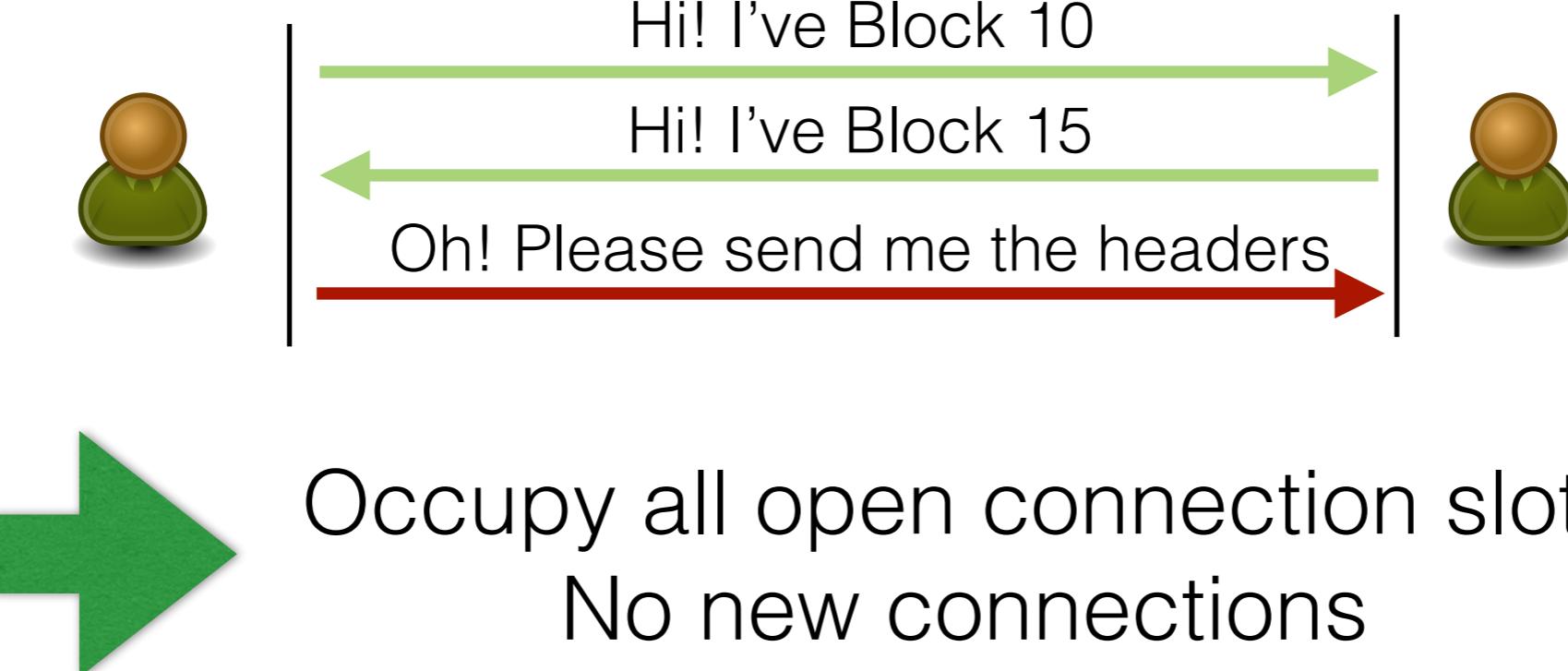
## Block delivery time > 20 min

Version message exchange  
on connection initiation



## Block delivery time > 20 min

Version message exchange  
on connection initiation



## **Block delivery time > 20 min**

### 1. Requirement for victim



- Must not receive block header
- Must not receive version message

## Block delivery time > 20 min

### 1. Requirement for victim



- Must not receive block header
- Must not receive version message

### 2. Requirements for adversary



- Must be first relayer for all blocks
- Should perform connection depletion

# Extending the block delivery time - Example



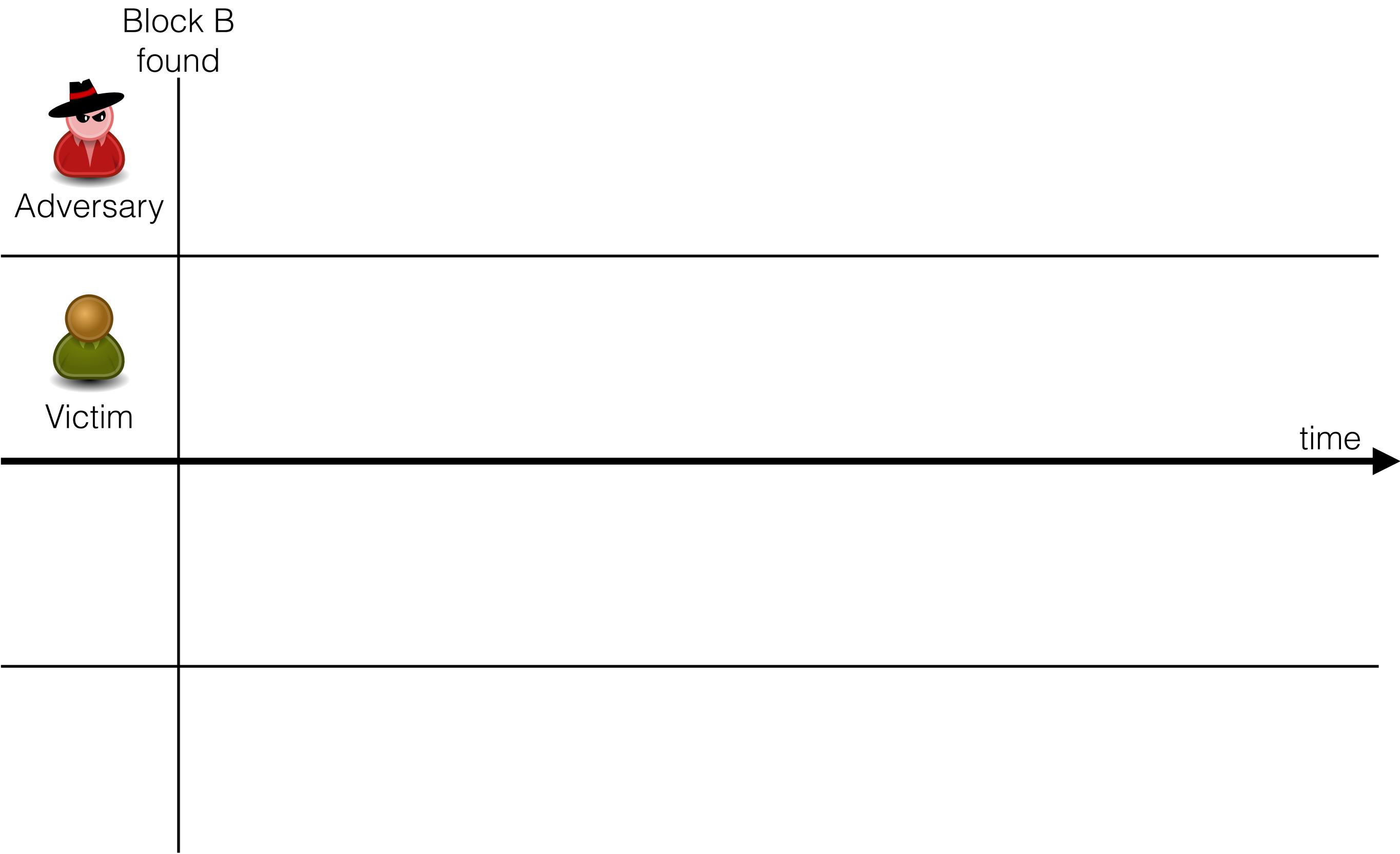
Adversary



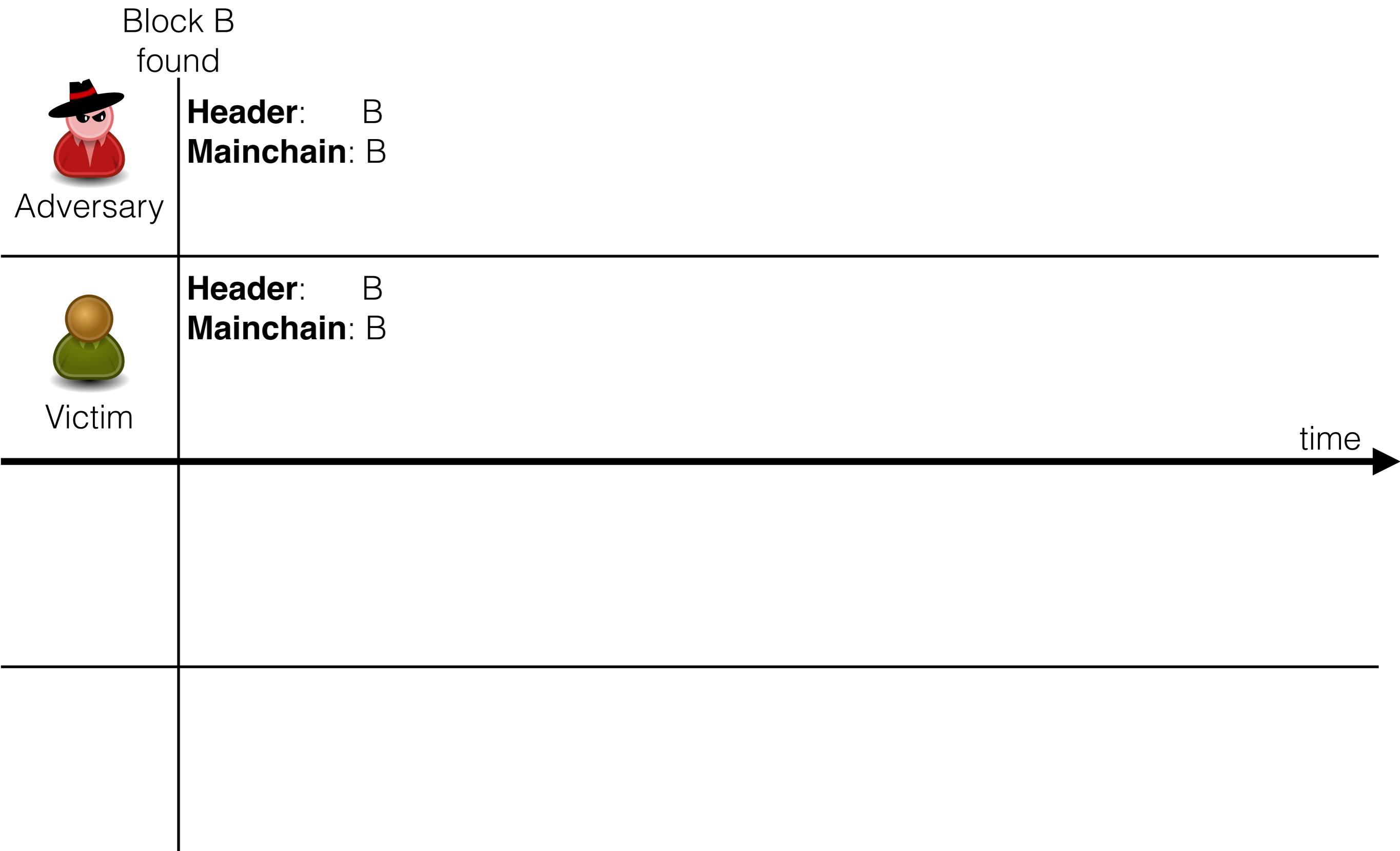
Victim

time

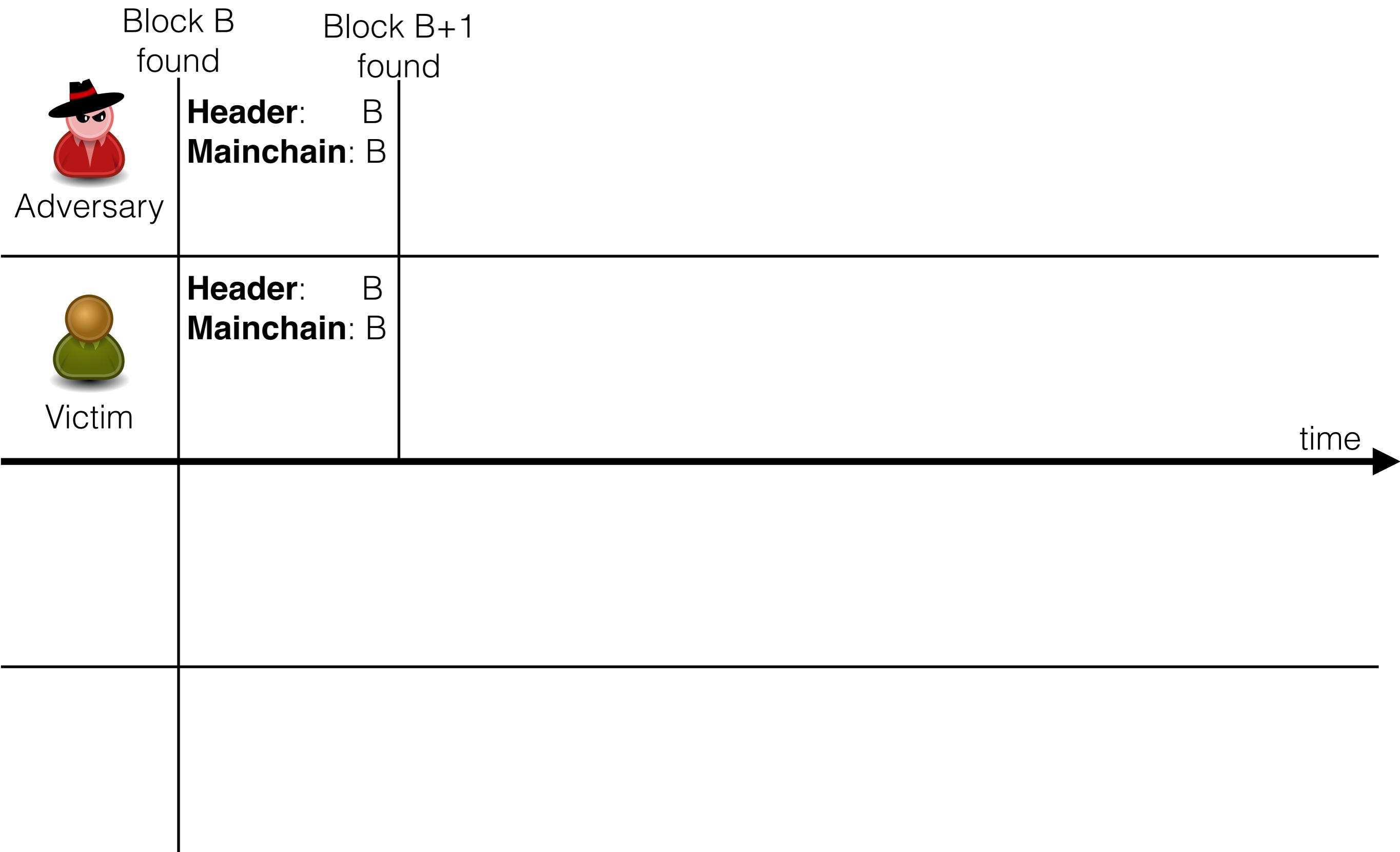
## Extending the block delivery time - Example



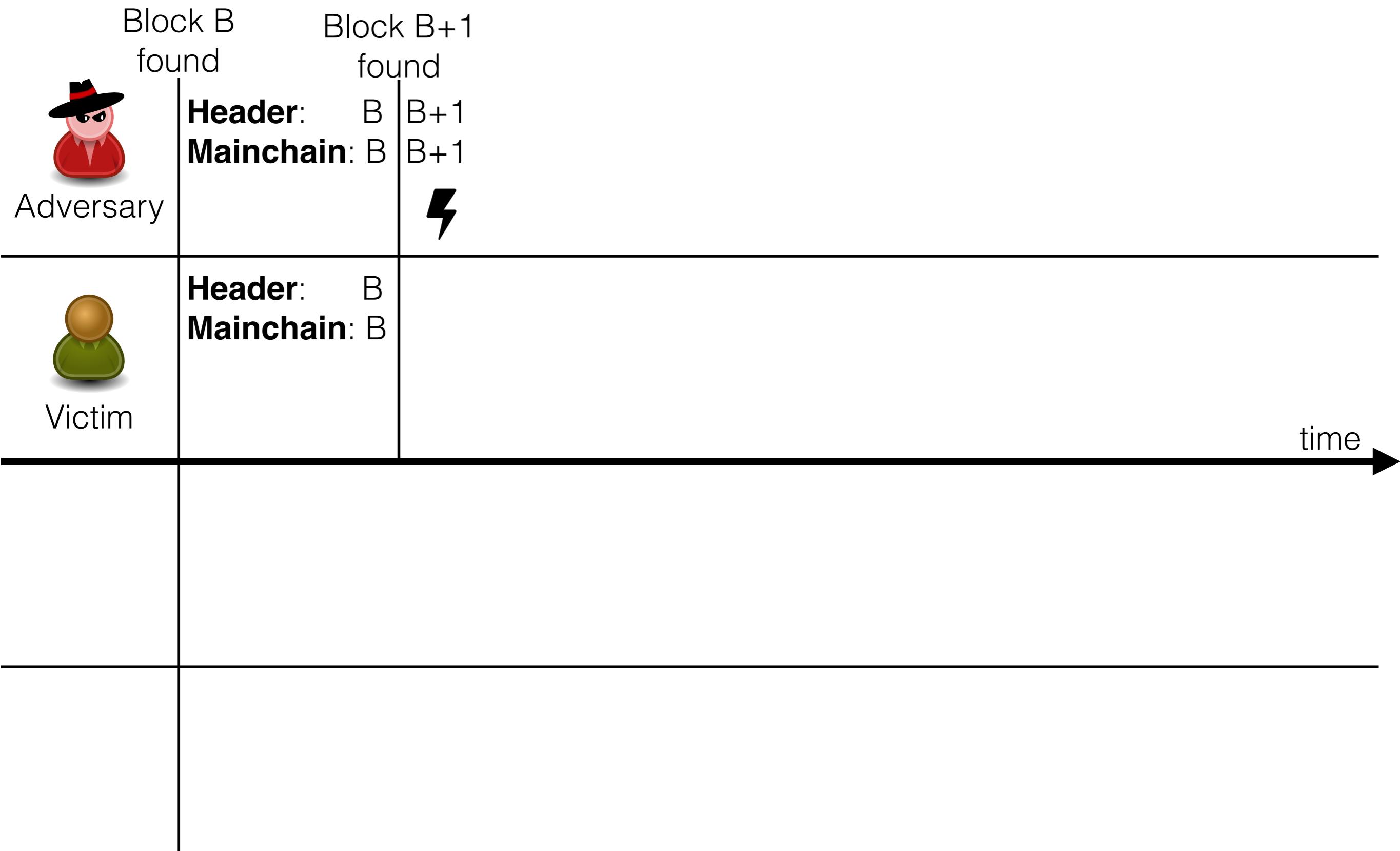
## Extending the block delivery time - Example



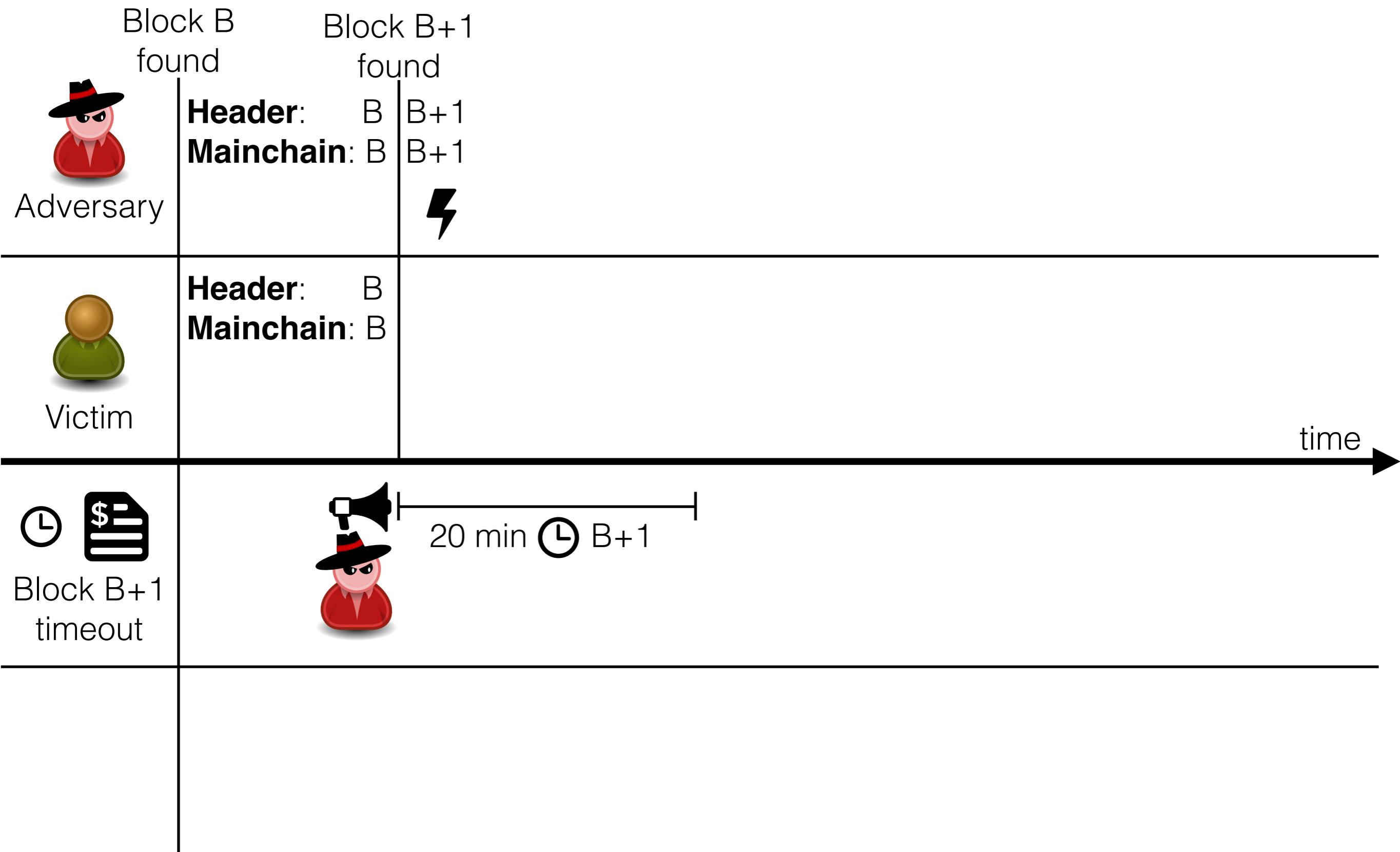
## Extending the block delivery time - Example



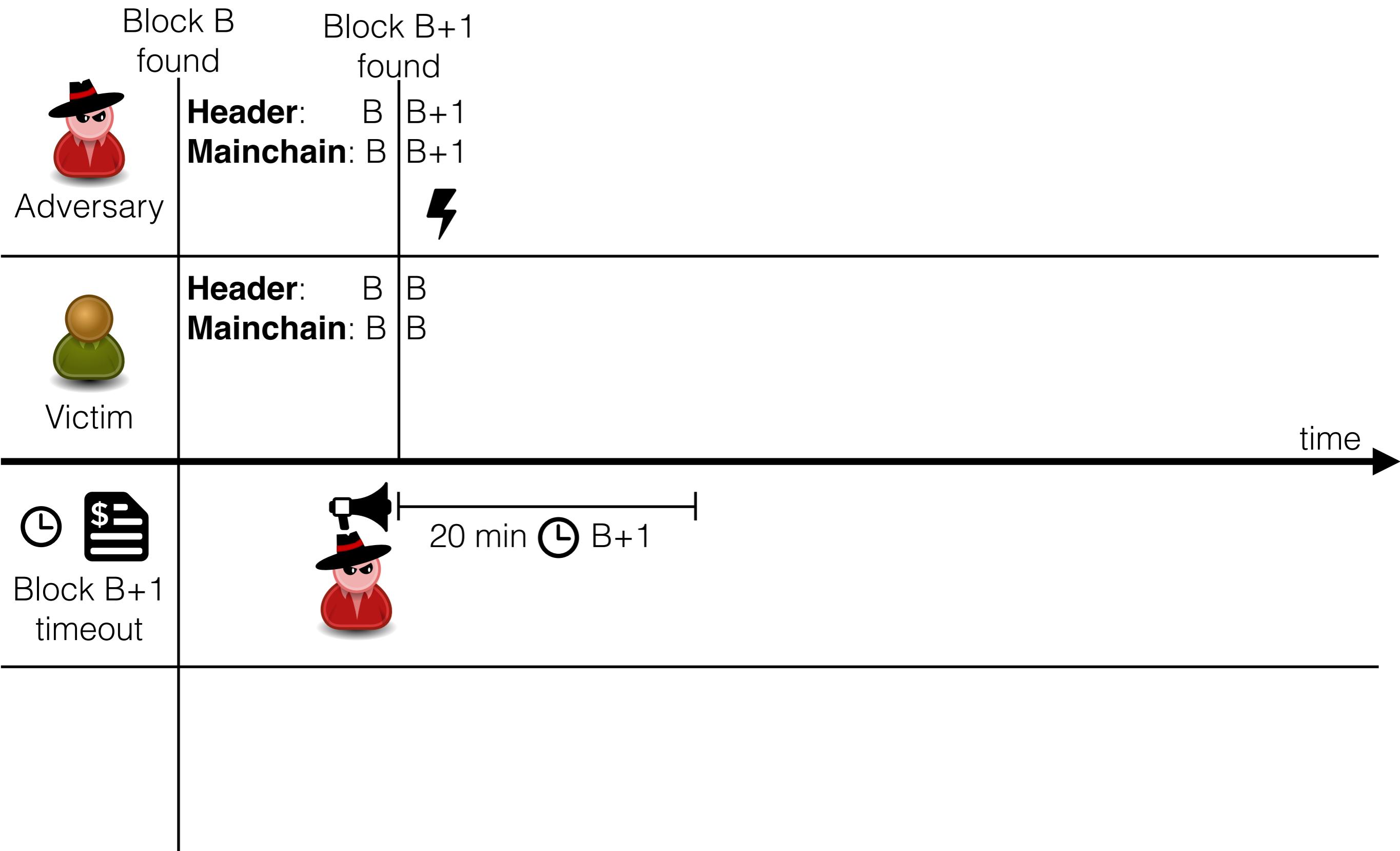
## Extending the block delivery time - Example



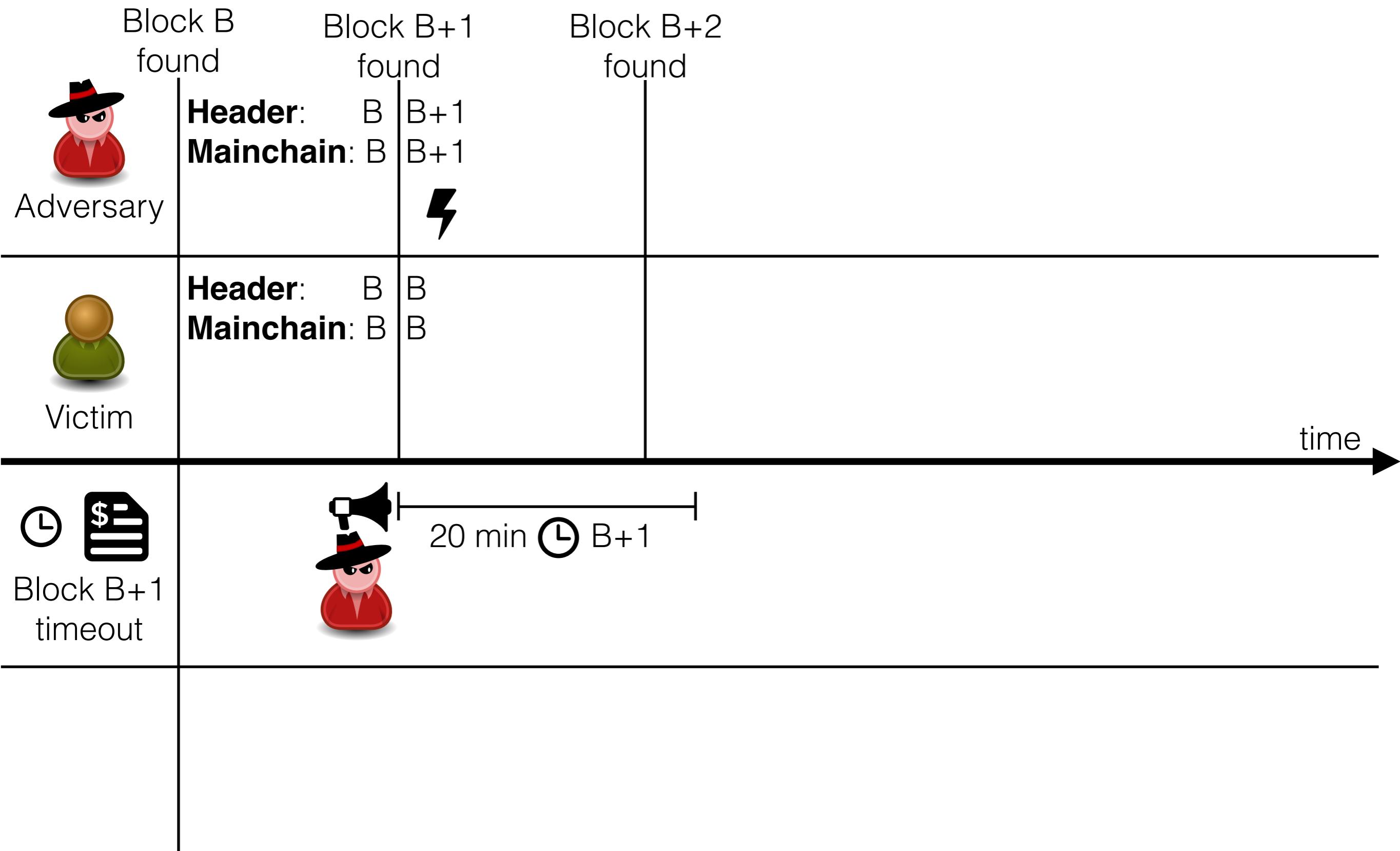
## Extending the block delivery time - Example



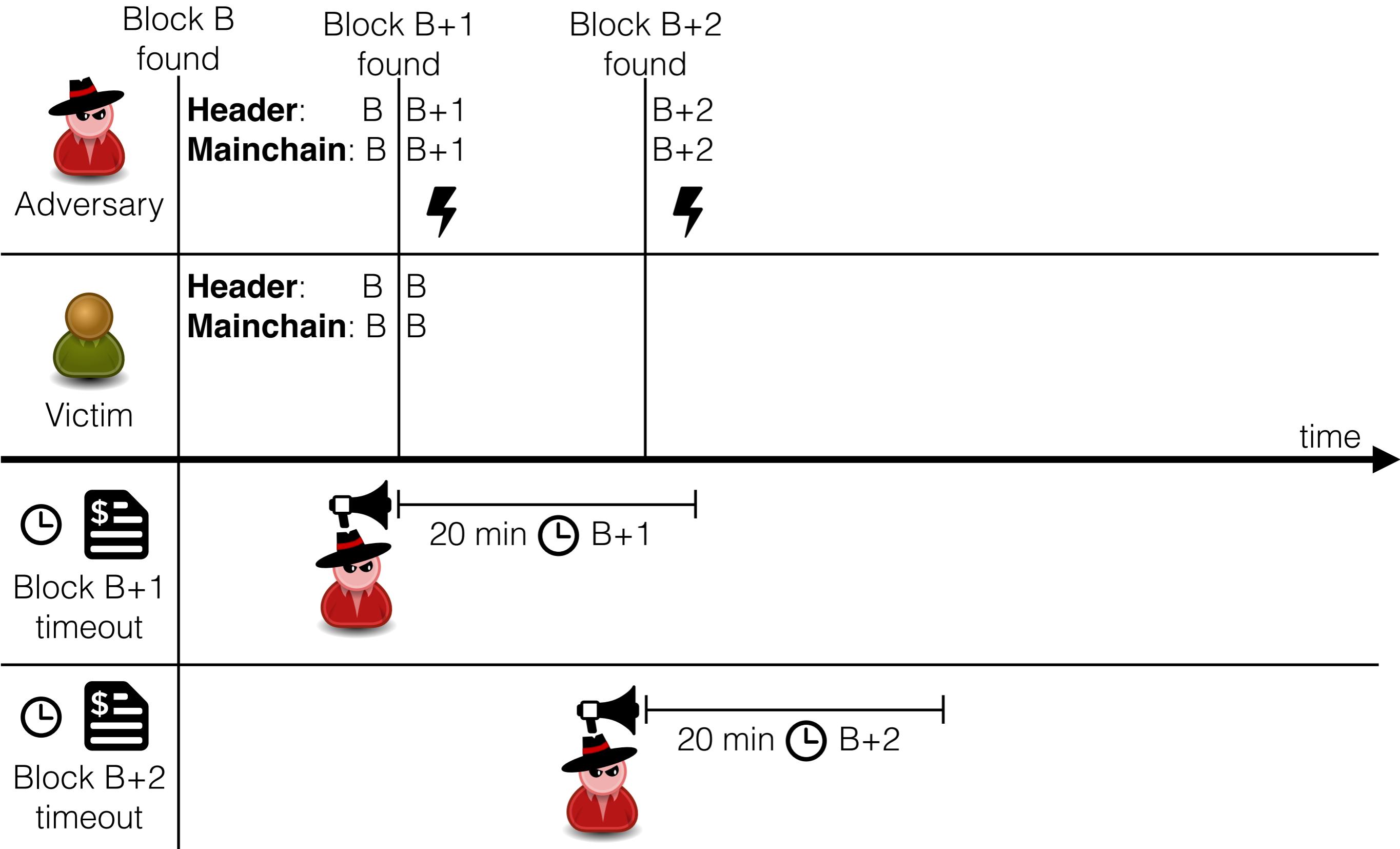
## Extending the block delivery time - Example



## Extending the block delivery time - Example



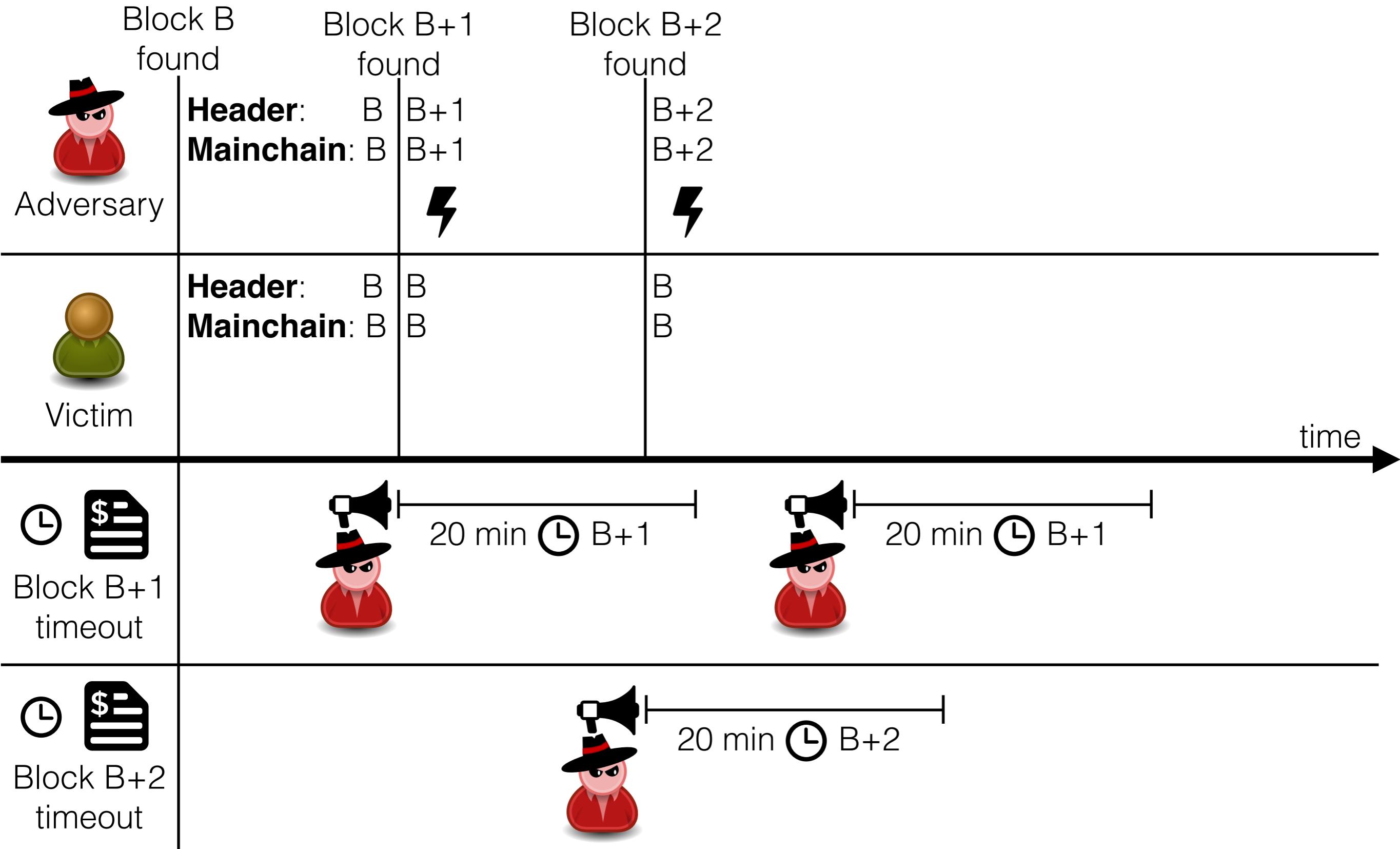
## Extending the block delivery time - Example



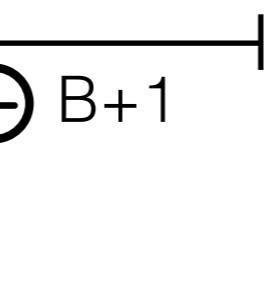
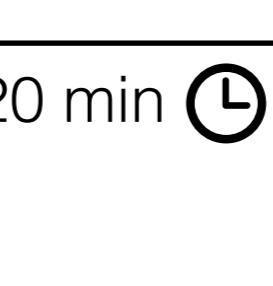
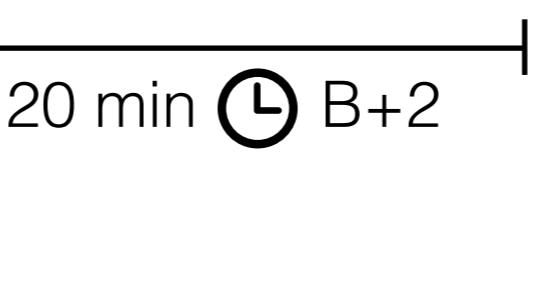
## Extending the block delivery time - Example

	Block B found	Block B+1 found	Block B+2 found
Adversary	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+1 <b>Mainchain:</b> B+1	<b>Header:</b> B+2 <b>Mainchain:</b> B+2
Victim	<b>Header:</b> B <b>Mainchain:</b> B	B	B
Block B+1 timeout	 20 min  B+1		
Block B+2 timeout		 20 min  B+2	

## Extending the block delivery time - Example



## Extending the block delivery time - Example

	Block B found	Block B+1 found	Block B+2 found	Block B+3 found
Adversary	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+1 <b>Mainchain:</b> B+1	<b>Header:</b> B+2 <b>Mainchain:</b> B+2	
Victim	<b>Header:</b> B <b>Mainchain:</b> B	B	B	
				time →
Block B+1 timeout		20 min  B+1		20 min  B+1
Block B+2 timeout			20 min  B+2	

## Extending the block delivery time - Example

	Block B found	Block B+1 found	Block B+2 found	Block B+3 found
Adversary	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+1 <b>Mainchain:</b> B+1	<b>Header:</b> B+2 <b>Mainchain:</b> B+2	<b>Header:</b> B+3 <b>Mainchain:</b> B+3
Victim	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B <b>Mainchain:</b> B	
				time →
Block B+1 timeout		 20 min  B+1	 20 min  B+1	
Block B+2 timeout			 20 min  B+2	

## Extending the block delivery time - Example

	Block B found	Block B+1 found	Block B+2 found	Block B+3 found
Adversary	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+1 <b>Mainchain:</b> B+1	<b>Header:</b> B+2 <b>Mainchain:</b> B+2	<b>Header:</b> B+3 <b>Mainchain:</b> B+3
Victim	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+3 <b>Mainchain:</b> B
				time →
Block B+1 timeout		 20 min  B+1		 20 min  B+1
Block B+2 timeout			 20 min  B+2	

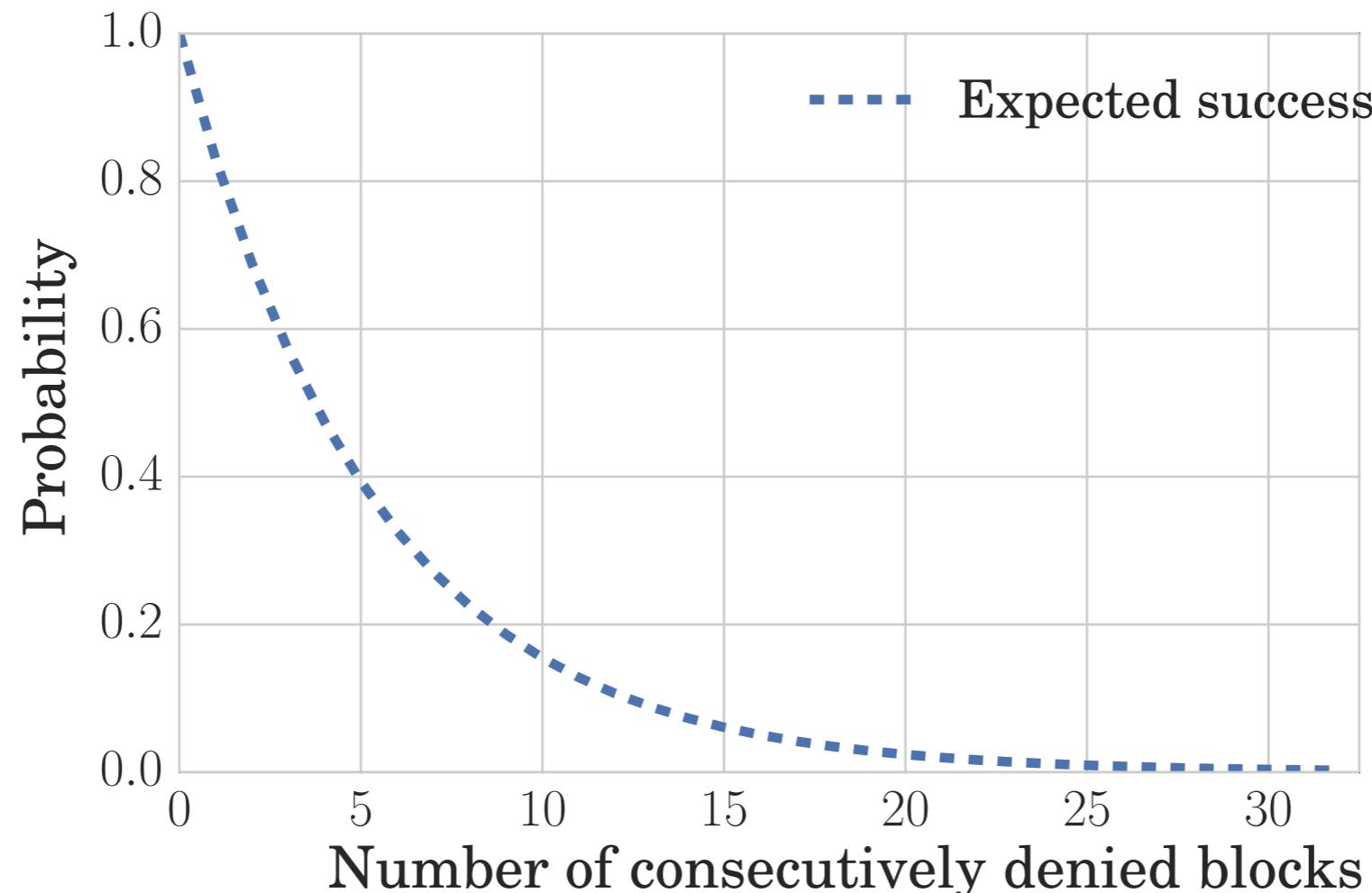
## Extending the block delivery time - Example

	Block B found	Block B+1 found	Block B+2 found	Block B+3 found
Adversary	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+1 <b>Mainchain:</b> B+1	<b>Header:</b> B+2 <b>Mainchain:</b> B+2	<b>Header:</b> B+3 <b>Mainchain:</b> B+3
Victim	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+3 <b>Mainchain:</b> B
				time →
Block B+1 timeout		 20 min  B+1		 20 min  B+1
Block B+2 timeout			 20 min  B+2	<b>Learns B+2</b>

## Extending the block delivery time - Example

	Block B found	Block B+1 found	Block B+2 found	Block B+3 found	
Adversary	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+1 <b>Mainchain:</b> B+1	<b>Header:</b> B+2 <b>Mainchain:</b> B+2	<b>Header:</b> B+3 <b>Mainchain:</b> B+3	
Victim	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B <b>Mainchain:</b> B	<b>Header:</b> B+3 <b>Mainchain:</b> B	<b>Header:</b> B+3 <b>Mainchain:</b> B+3
	time				
Block B+1 timeout		 20 min	 20 min		<b>Learns B+1</b>
Block B+2 timeout			 20 min		<b>Learns B+2</b>

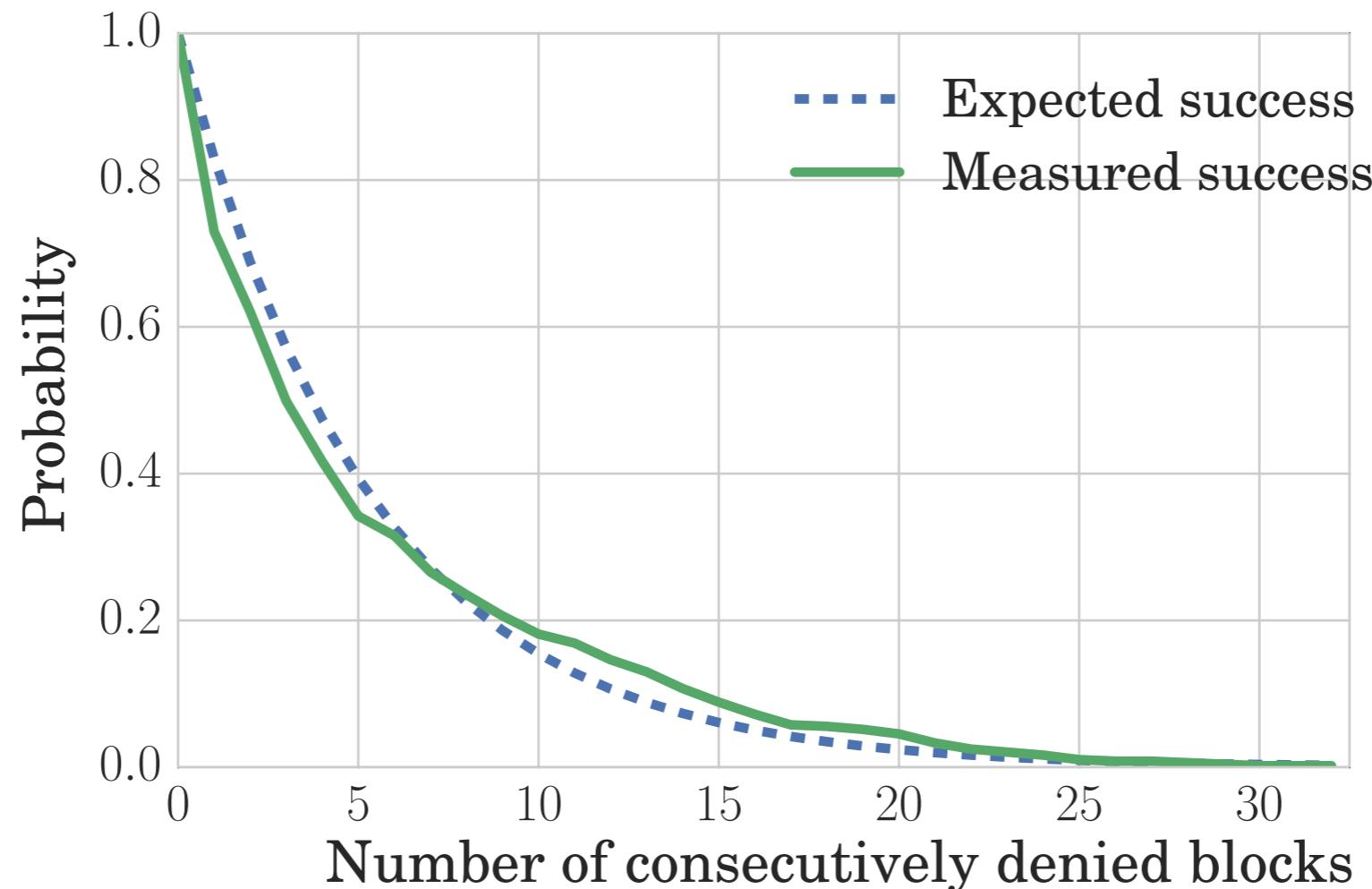
## Experimental validation - Continuous block denial



Probability for N blocks =  $P^N$

based on  $P = 0.83$

## Experimental validation - Continuous block denial



Probability for  $N$  blocks =  $P^N$

based on  $P = 0.83$



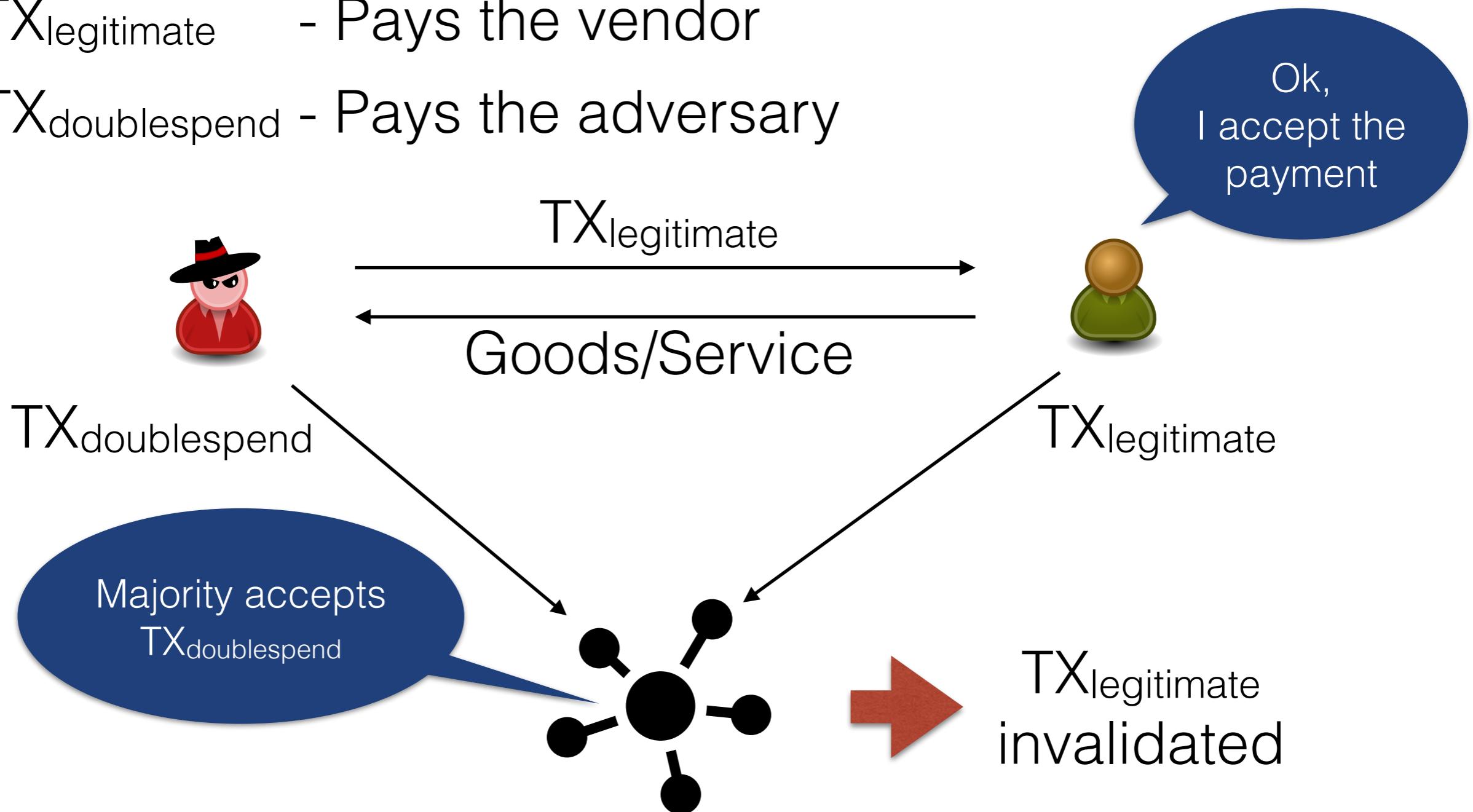
# Eclipse Attacks - Double Spending

# What is Double Spending?

Spending money more than once

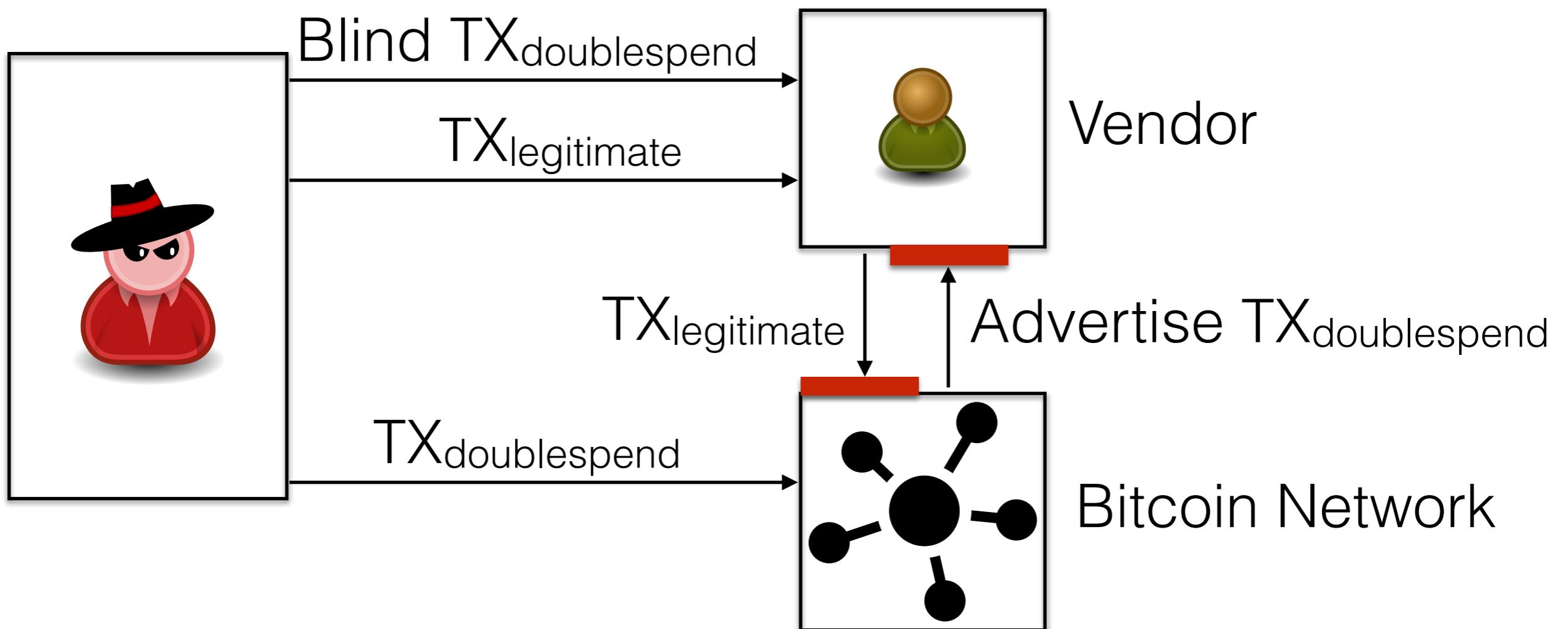
$\text{TX}_{\text{legitimate}}$  - Pays the vendor

$\text{TX}_{\text{doublespend}}$  - Pays the adversary

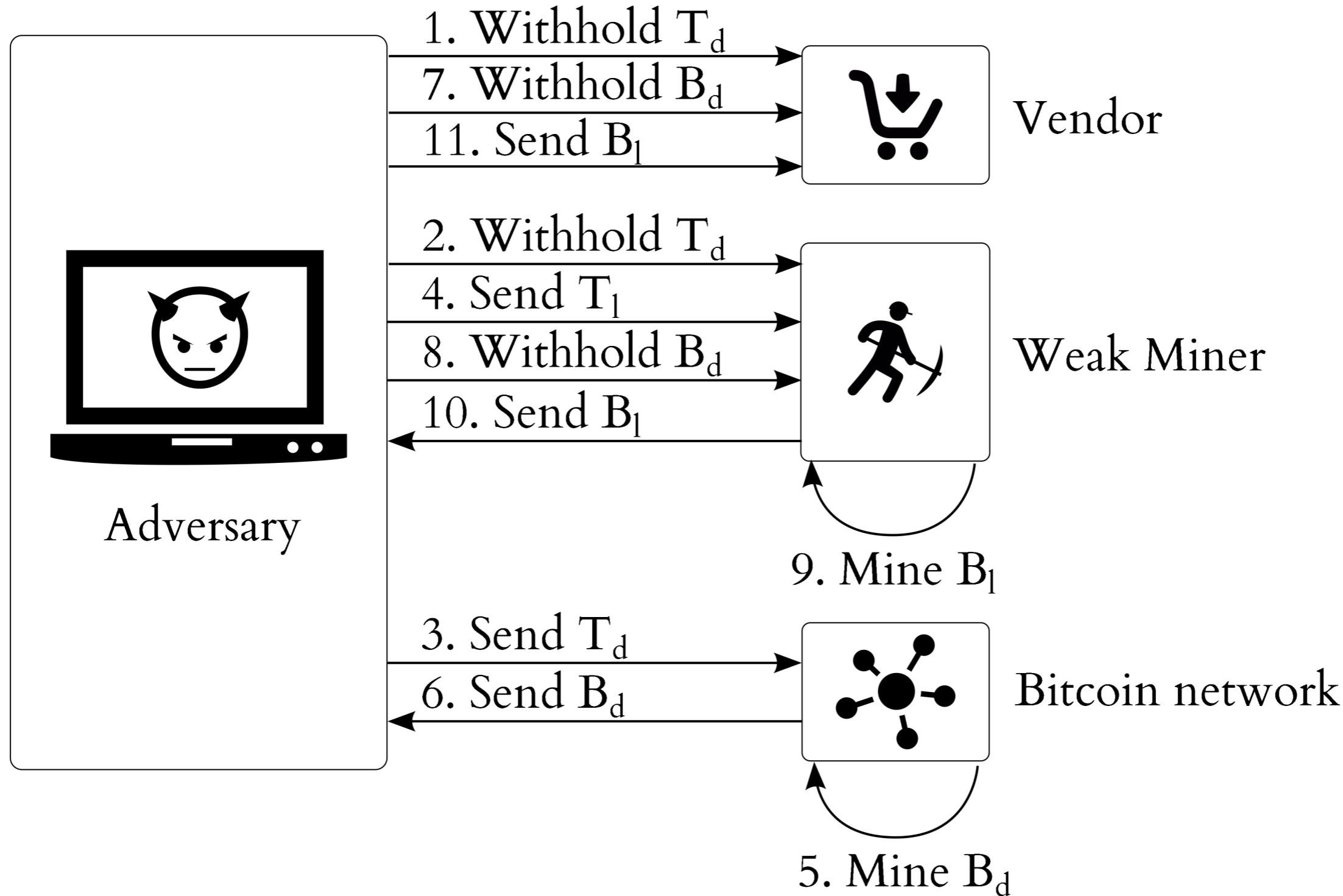


# Double Spending 0 Confirmation Transactions

- Very reliable attack
- Regardless of protection (double spend relay)



# Implications - Double Spending 1 Confirmation Transactions



$Xl$  = legitimate  
 $Xd$  = double spend attempt



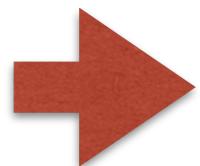
# Eclipse Attacks - Denial of Service

## Denial of Service

6000 reachable Bitcoin nodes

Preventing the delivery of blocks to these

- 450 000 TCP connections required
- 600 KB of advertisement / block / 20 min



Network wide Denial of Service



# Eclipse Attacks - Hardening the P2P Layer

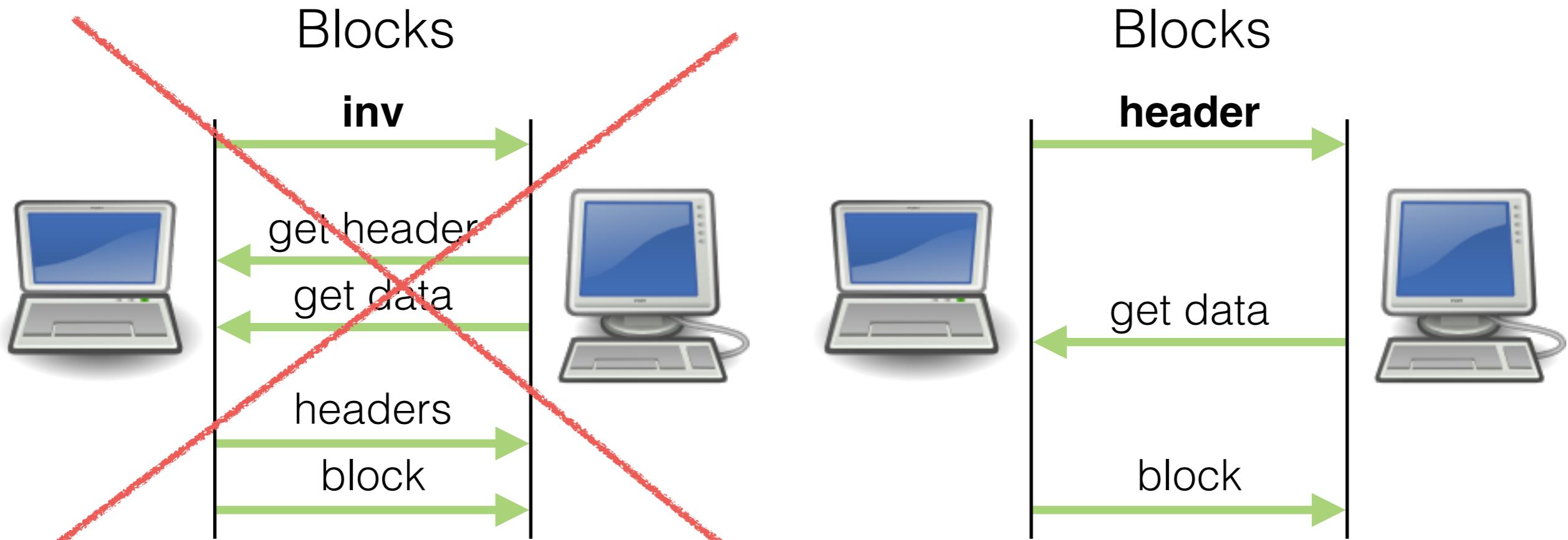
## **Hardening the P2P overlay network**

1. Why not smaller static timeouts?
2. Why not requesting from multiple peers?
3. What about alternative relay networks?

## **Security vs Scalability tradeoffs**

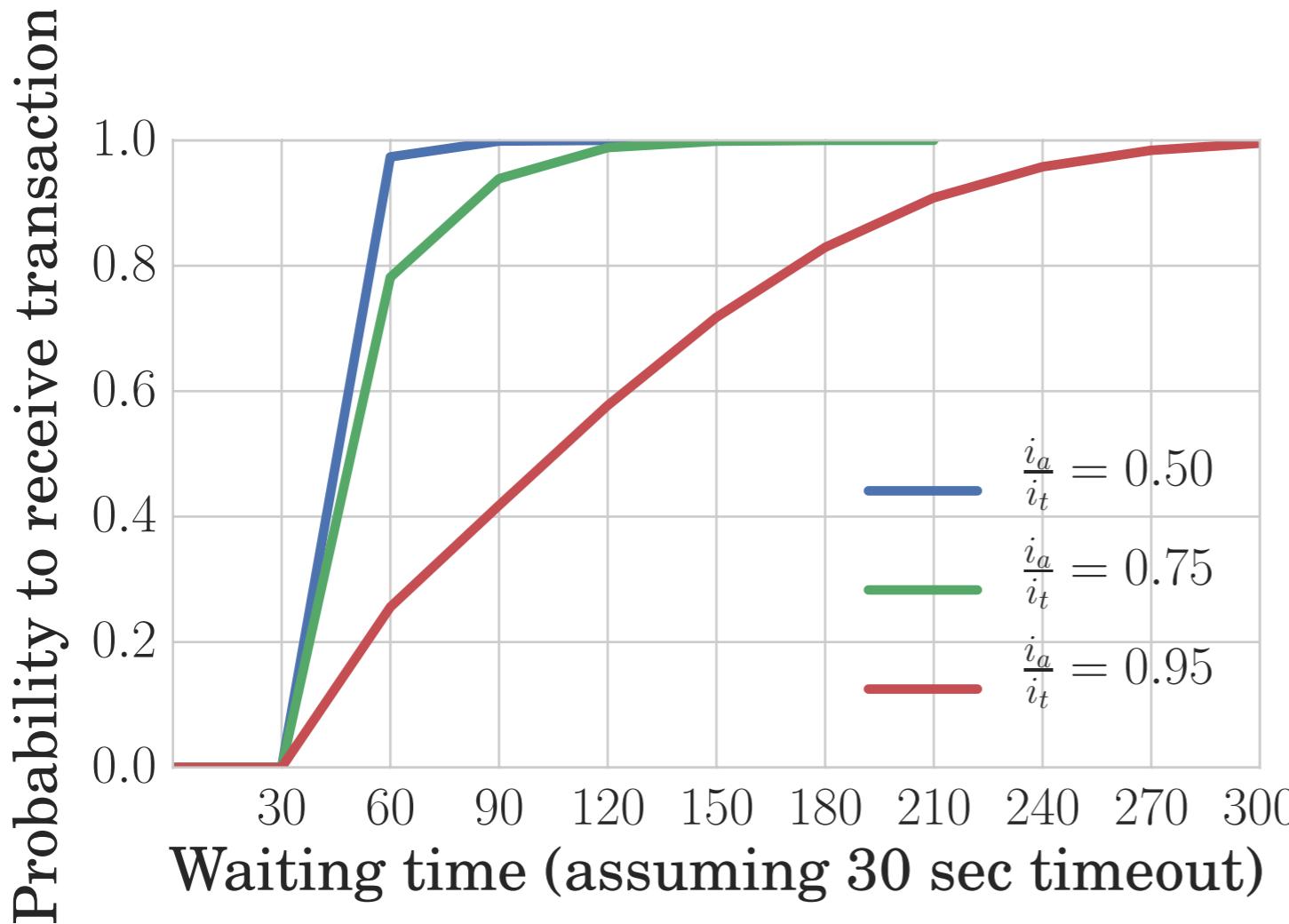
# Hardening the P2P overlay network

Better Block request management



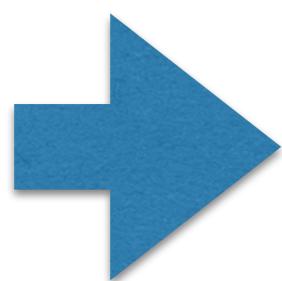
$\text{size(inv)} = 36 \text{ bytes}$   
 $\text{size(header)} = 80 \text{ bytes}$

## Hardening the P2P overlay network



$i_a = \mathbf{inv}$  messages sent by adversary

$i_t = \mathbf{total inv}$  messages



After 5 minutes, transaction is received, even if the adversary controls 95% of the inv



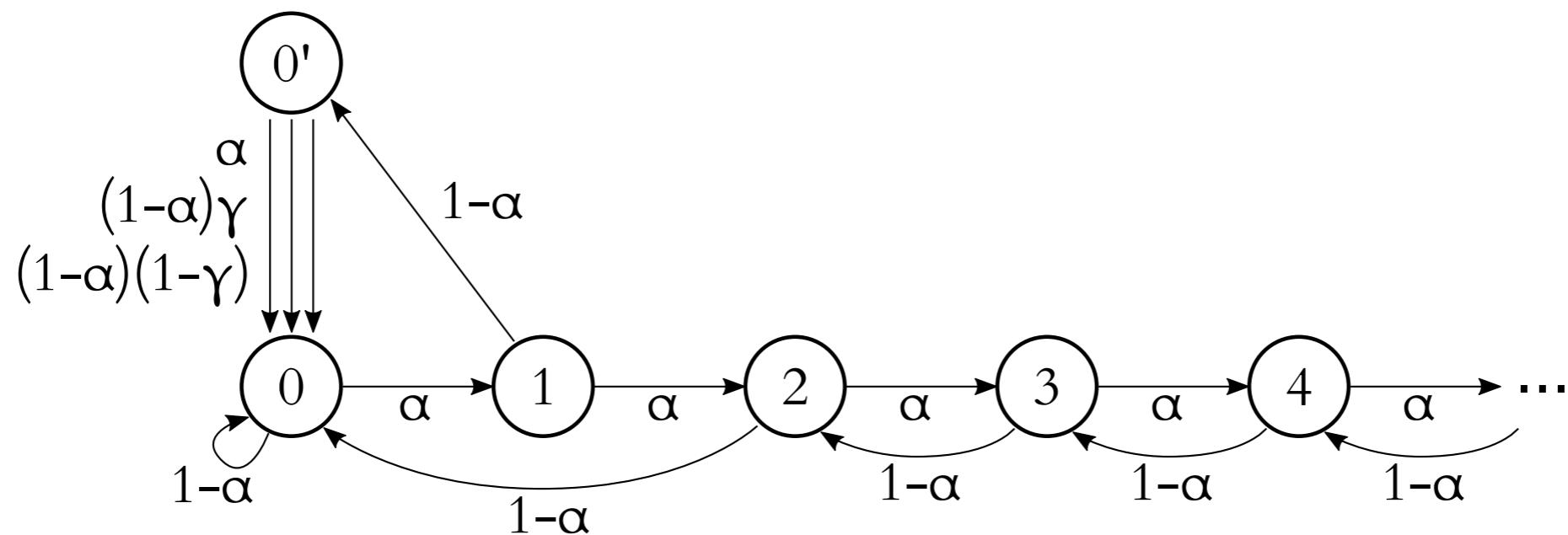
# Selfish Mining and Eclipse Attacks

## Implications - Increasing Mining Advantage

Idea from Eyal et. al:

- Instead of publishing, keep a block private

→ Other miners will perform wasteful computations

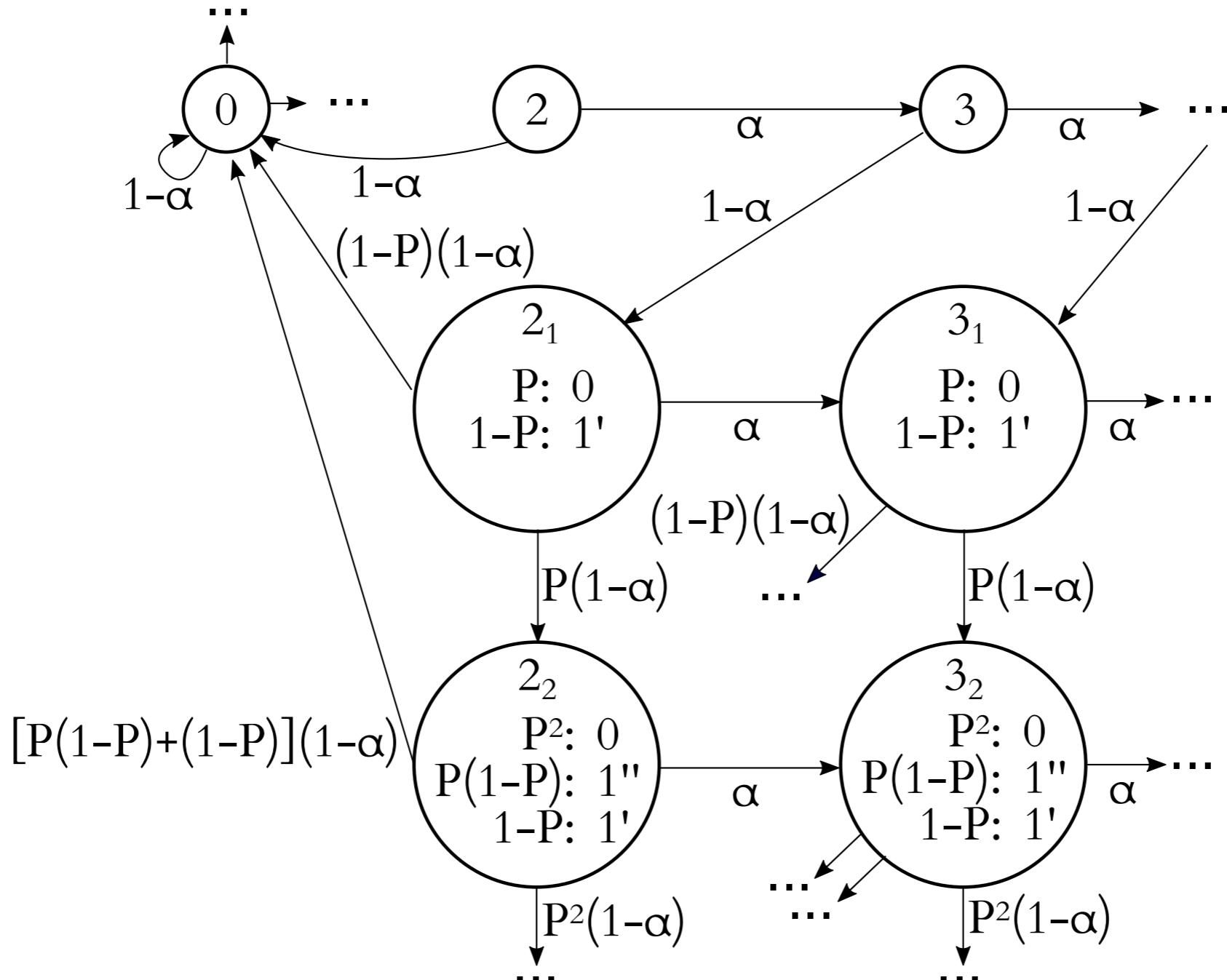


$\alpha$ : hashing power of adversary



$\gamma$ : propagation parameter

## Implications - Increasing Mining Advantage



P: probability to deny a block to a miner



## Implications - Increasing Mining Advantage

