# What does censorship-resistance look like in code?


@glozow

6B00 2C6E A3F9 1B1B 0DF0 C9BC 8F61 7F12 00A6 D25C

# censorship-resistance is one of Bitcoin's most valuable properties

anyone can pay anyone, anywhere in the world

no central authority that can confiscate/freeze your money

even in the presence of adversaries

# … but what does that look like in code?


```
send_payment()
```

```
send_payment_uncensorable()
```

**I won’t display code, but am linking code for slides:**

tinyurl.com/seoul-code-links

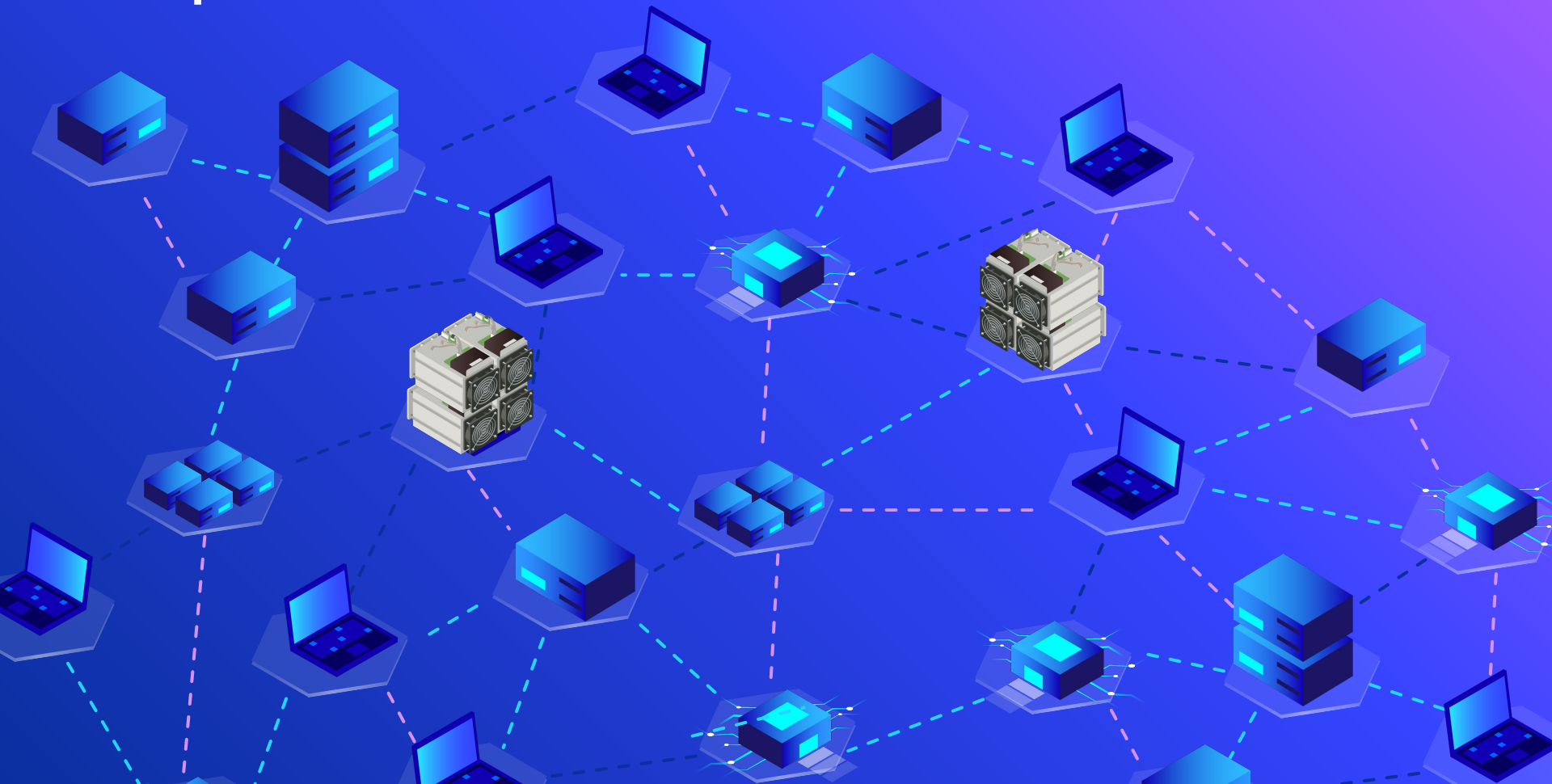
🔎 <- look for this icon on slide

# which payment is easier to stop?

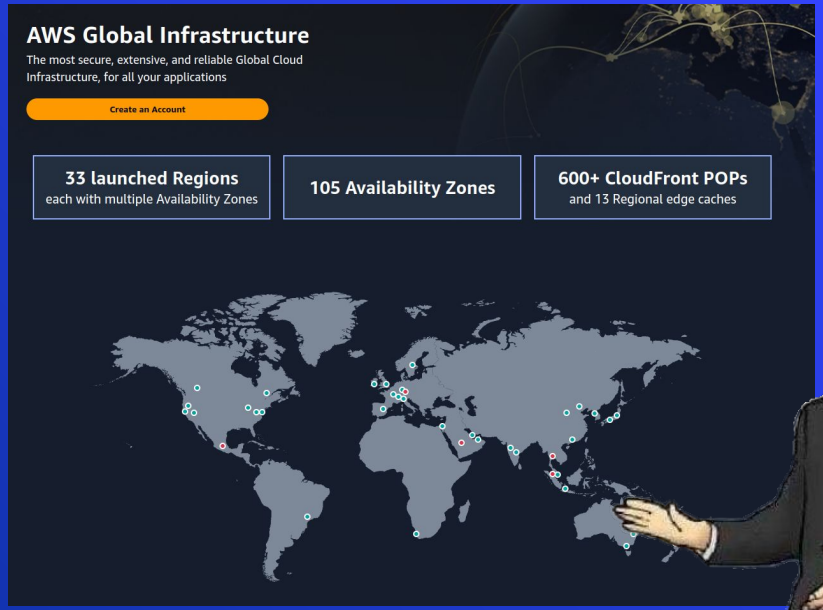
# which payment is easier to stop?

this one

# our equivalent is a distributed network of nodes



# … but what does that look like in code?




## We want many independent users to run nodes.
## From a software engineering perspective, this necessitates:

the software must be supported on many platforms



the code is open source and free to use

node operators have control of what is running



**running a node must be cheap i.e. minimal operating costs, even as usage scales**

mining nodes accept
transactions to collect fees

non-mining nodes
don’t get paid

# The Costs

every node has limited:

- memory
- bandwidth
- computation

# The Costs

every node has limited:

- memory
- bandwidth
- computation

tx relay requires:

- memory
- bandwidth
- computation

# The Costs

every node has limited:

- memory
- bandwidth
- computation

tx relay requires:

- memory
- bandwidth
- computation

block validation requires:

- memory
- bandwidth
- computation

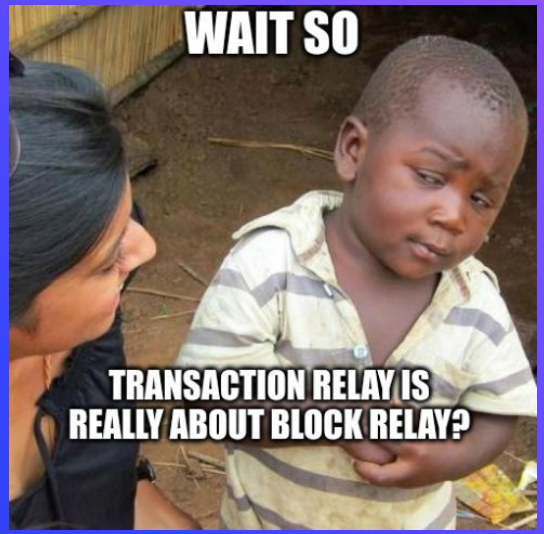
# The Benefits

Transaction relay amortizes block validation:

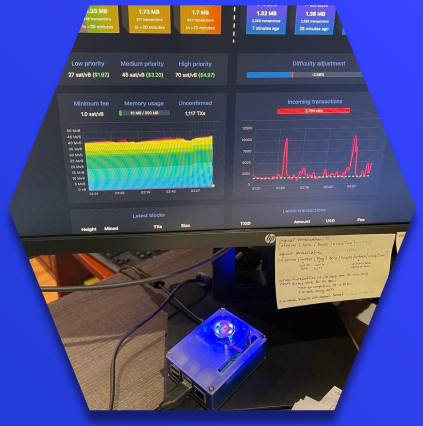
- loading UTXOs from disk
- downloading block data
- verifying signatures and scripts

This means:

- faster block relay (steadily instead of in spikes)
  - when network-wide = fewer reorgs
- more accurate fee estimation



The challenge = write useful code that can run with constrained resources (e.g. a raspberry pi or a laptop app)

transaction relay vs

# **memory**

# Use memory wisely

Several caches exist:

- UTXO (“coins”) cache
- signature cache
- script verification result cache
- memory pool of validated transactions
- orphan pool of transactions for which we are missing UTXOs
- ring buffer of rejected and replaced transactions

# A mempool is just a cache

= a *memory pool* of transactions that may end up in a block

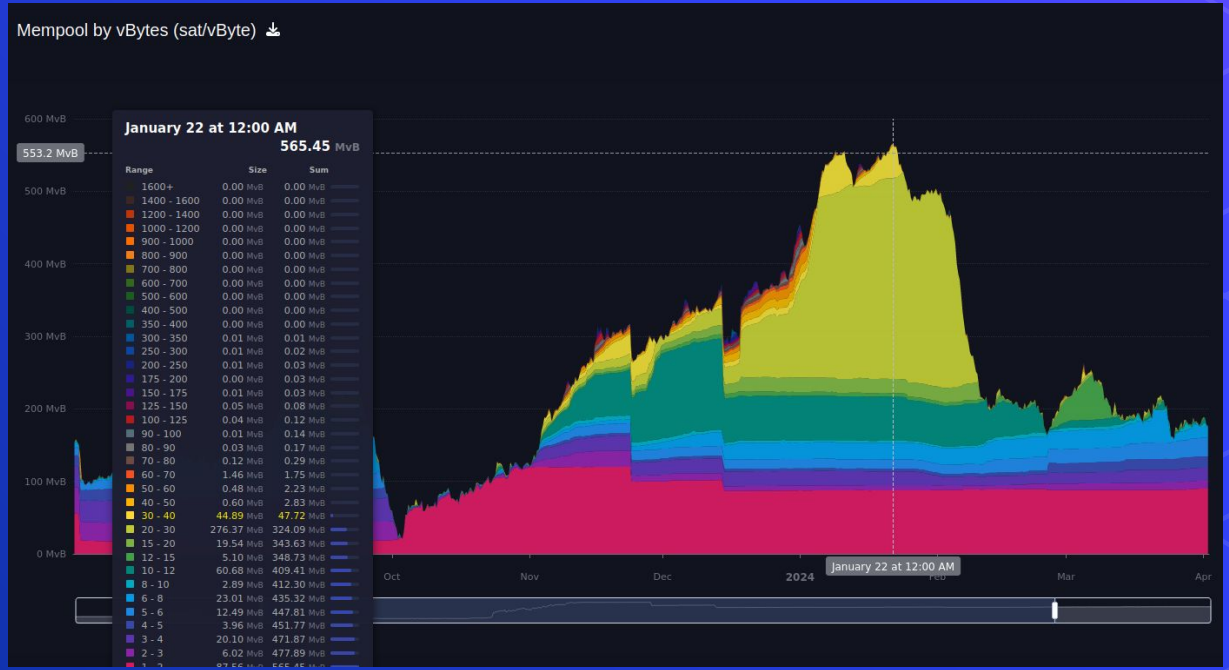
👉 memory bounds: limit maximum usage (300MB)

👉 hit rate: what ultimately matters (compact block reconstruction)

👉 define a utility metric and eviction policy

# Pop Quiz:

What happens when there are so many transactions that your mempool exceeds 300MB?



# utility metric = fees

a tx is useful if it ends up being confirmed

=> a tx is mined if it would be profitable

=> incentive compatibility* is a good utility metric

Eviction policy:

- remove transactions > 2 weeks old
- skim away transactions with the worst fees*
- allow replacements with better fees*

*: it's complicated

# “free relay” and min relay fee

👉 fees are not paid to the node operator

👉 however, fees represent a cost to tx creator

free relay = when total cost < memory (+bandwidth) used at 1sat/vB

- -minrelaytxfee: all txns must pay 1sat/vB, even if mempool is empty
- RBF rule #4: replacement must pay additional fees that cover the new transaction’s size at 1sat/vB
- the tx also needs to be valid (no ultimate cost if it never confirms!)

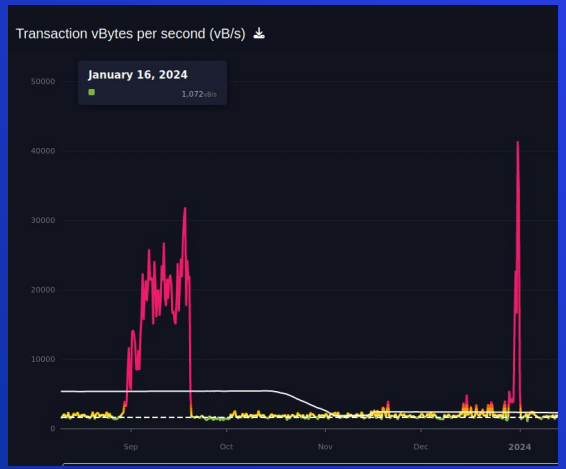
🔎

transaction relay vs

# bandwidth usage

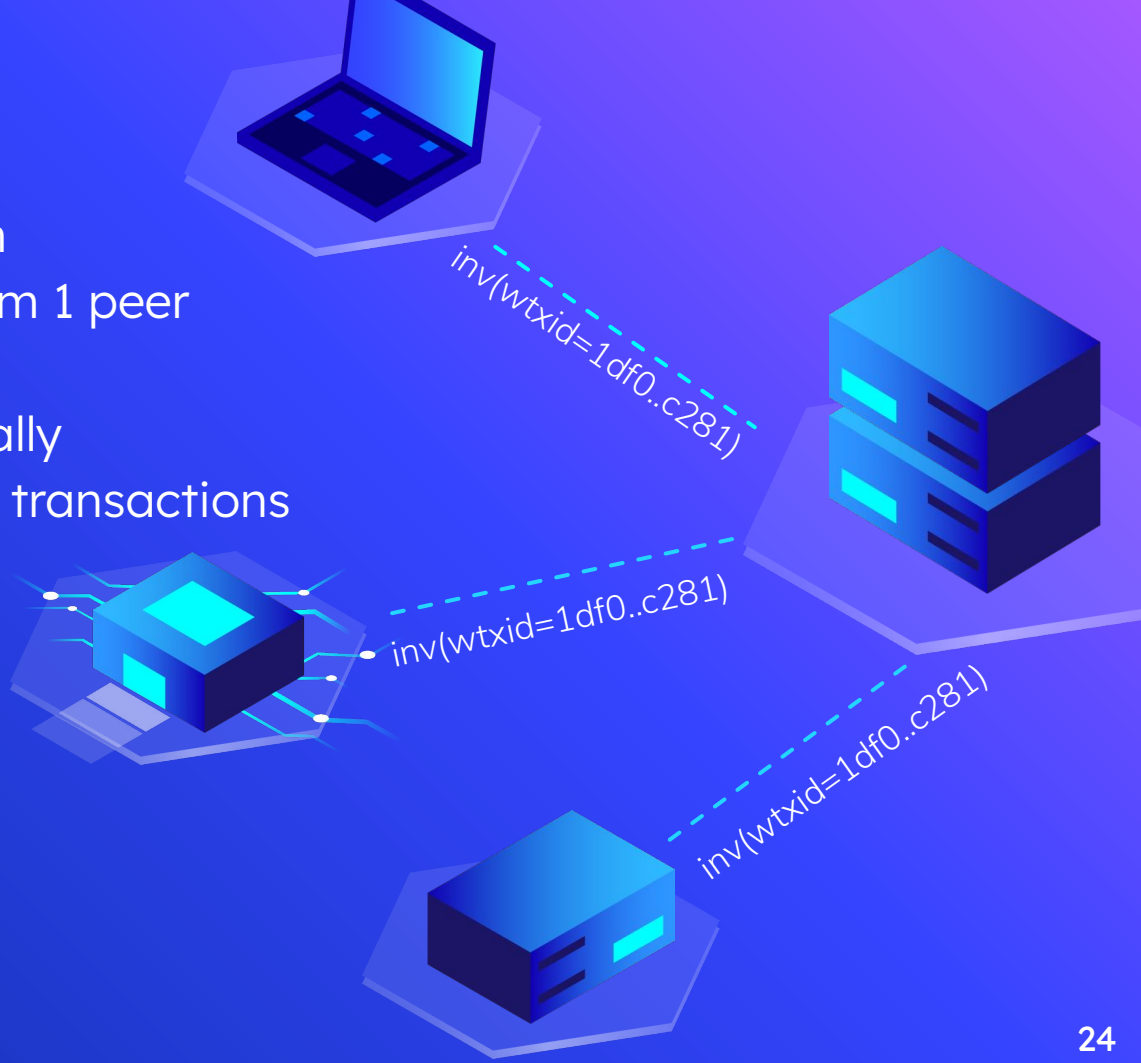
everybody gossip to everybody:

- very fast propagation!
- $O(n^2)$ tx messages
- lots of wasted bandwidth
- they might even be invalid



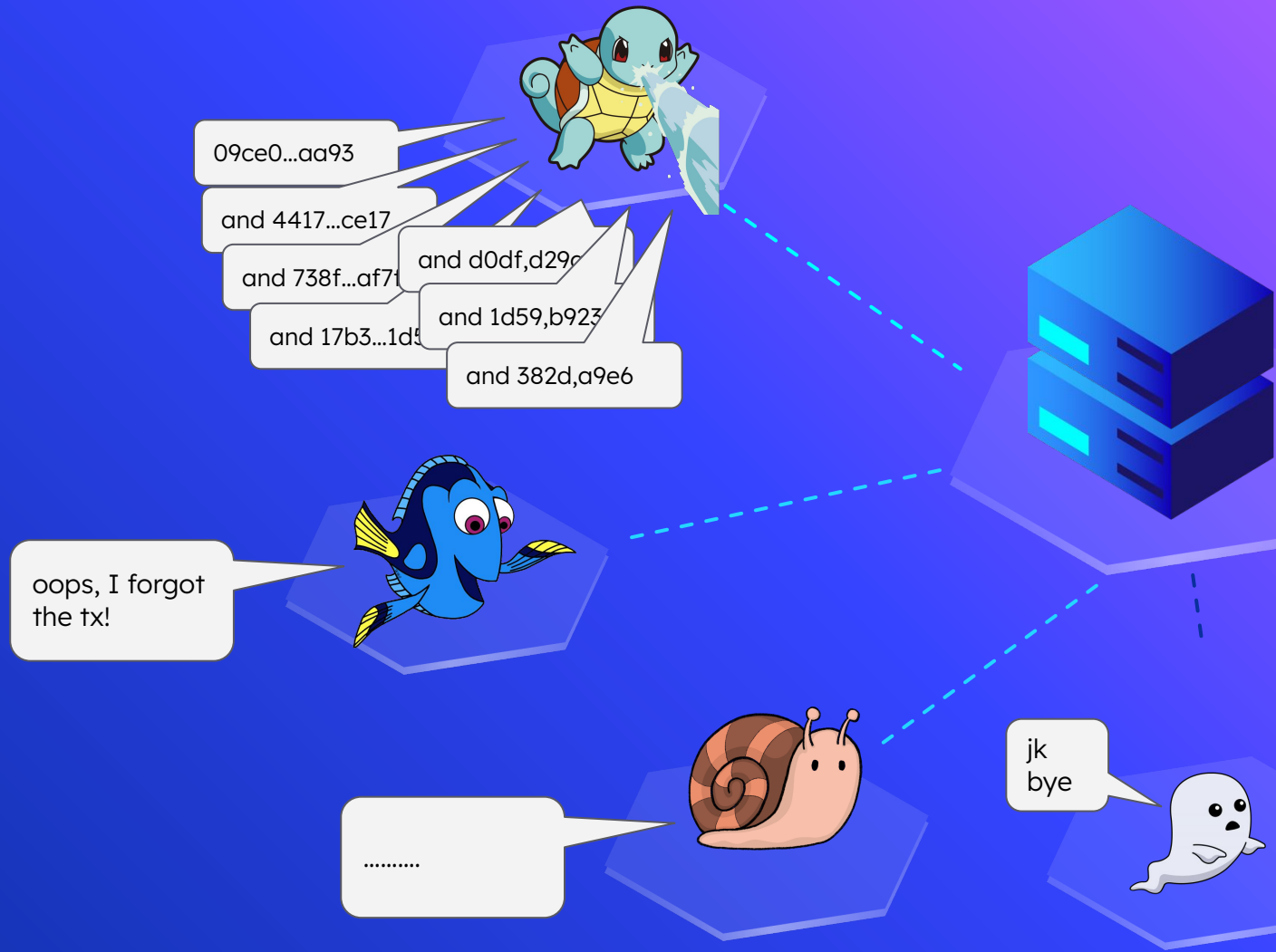

instead:

- gossip inventory by tx hash
- request what is missing from 1 peer
  - $O(n^2)$ 32b invs
  - O(n) tx messages, usually
- don’t re-download rejected transactions



## What if they:

- flood
- forget
- stall
- drop off



## We need to track all announcers

- double up requests after some timeout
- have preferences (e.g. outbounds)
- load-balance between peers

## How else can somebody sabotage a tx?

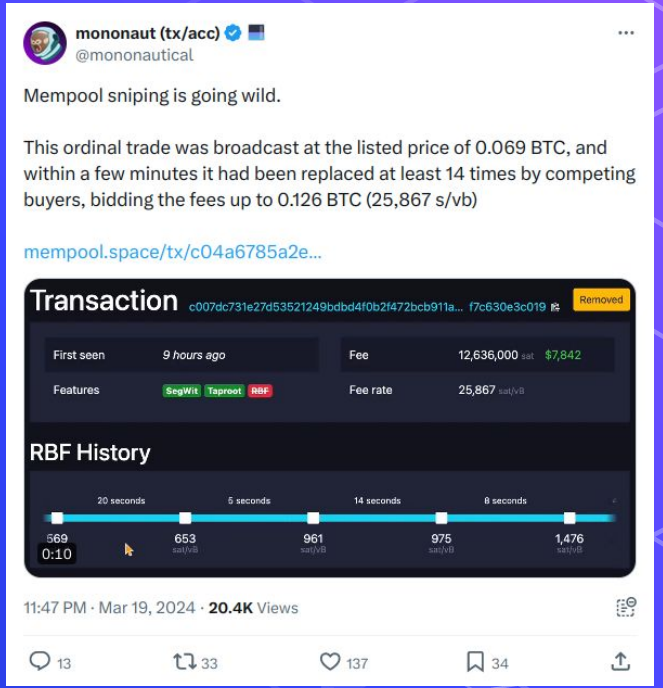
- malleation of tx data
- pinning through attaching children or replacements

# “mempool sniping”

(1) seller creates a malleable transaction, signed SIGHASH_SINGLE | SIGHASH_ANYONECANPAY, exchanging the NFT for the sell price

(2) buyer signs PSBT providing funds and sending NFT to themselves

(3) anyone can malleate the tx after broadcast, providing new funds and changing NFT destination (original buyer does not lose their money)



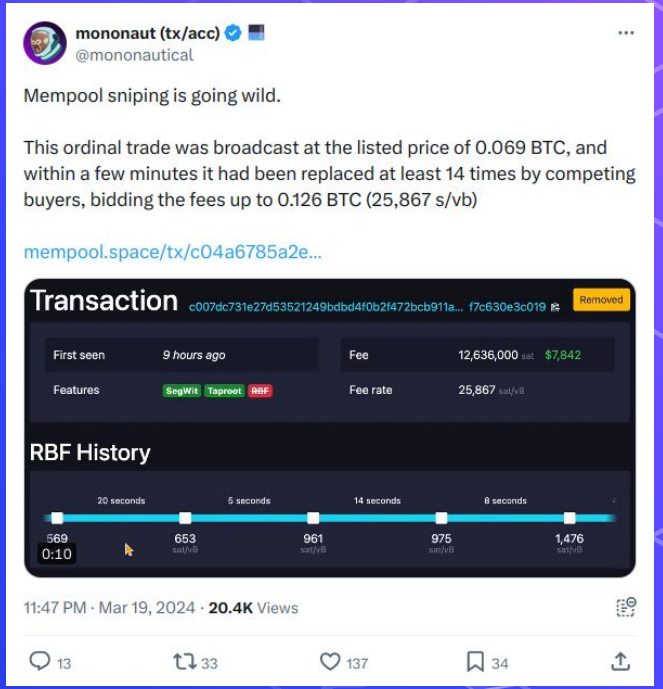
# “mempool sniping”

Lesson: if you sign a transaction with SIGHASH_ANYONECANPAY...

... anyone can change its inputs 

👉 don’t design stuff like this

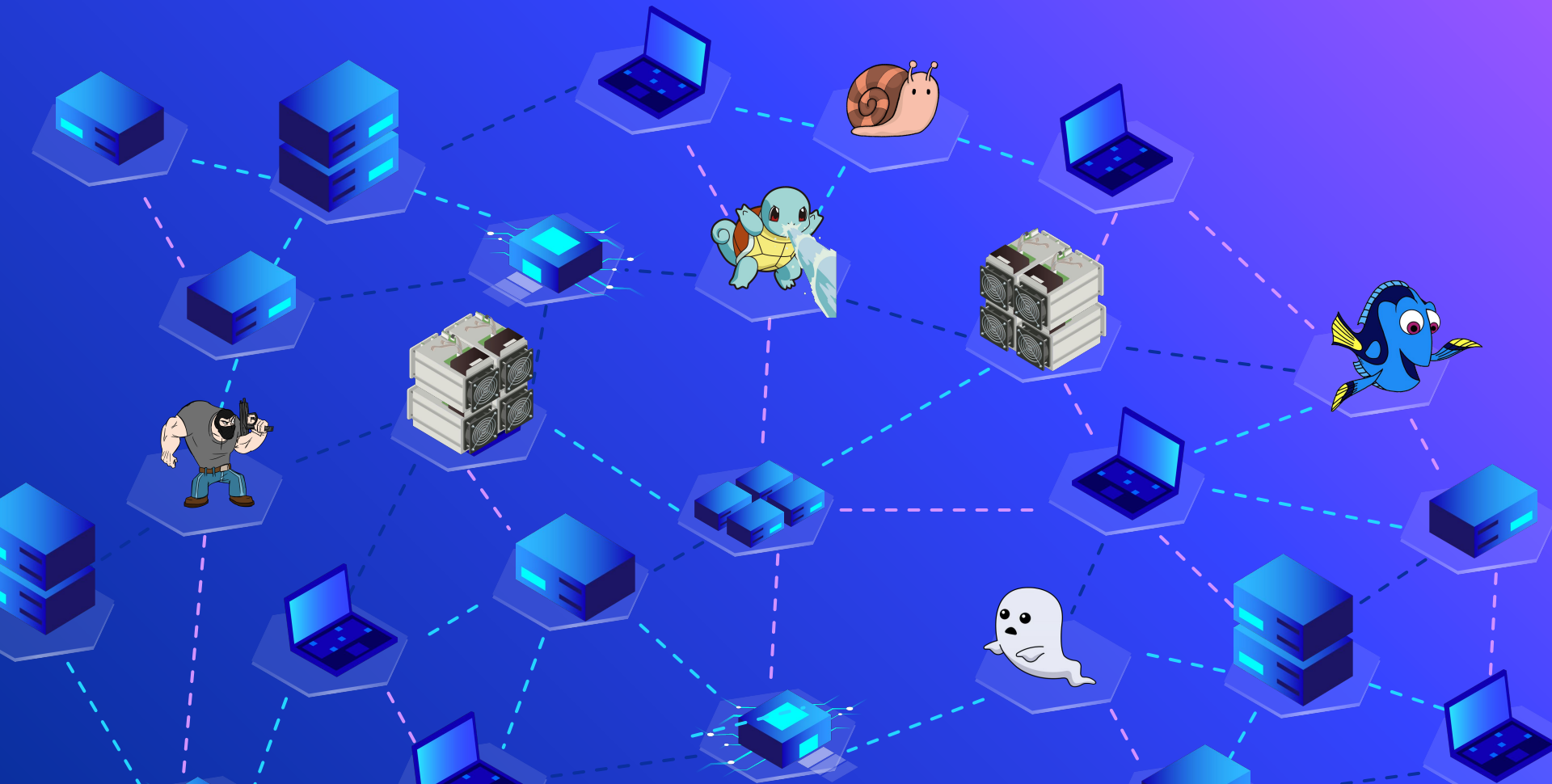
👉 we can’t fix this in transaction relay



# We can test censorship-resistance

- Add 1 honest peer + n “evil” peers programmed with possible actions:
  - forgetting 
  - stalling 
  - flooding 
  - malleating txs 
  - disconnecting 
- Simulate random combinations of mesages and assert that we always receive the tx

so if the network looks like this, the transaction can still go through



transaction relay vs

# computation

# computation has to be limited

our operating environment = anonymous entities on the internet
connect to you and send you data to process

👉 blocks are naturally DoS-resistant (must have PoW!)

👉 transactions can be computationally expensive to handle:

- signature verification
- mempool package updates
- etc

# signature verification

quadratic sighashing and the 1MB “Megatransaction”

Rusty Russell's Quiet Corner of The Internet About

## The Megatransaction: Why Does It Take 25 Seconds?

Jul 8, 2015

Last night f2pool mined a 1MB block containing a single 1MB transaction. This scooped up some of the spam which has been going to various weakly-passworded "brainwallets", gaining them 0.5569 bitcoins (on top of the normal 25 BTC subsidy). You can see the megatransaction on blockchain.info.

It was widely reported to take about 25 seconds for bitcoin core to process this block: this is far worse than my "2 seconds per MB" result in my last post, which was considered a pretty bad case. Let's look at why.

👉 only validate transactions of limited size and sigops, only allow up to 15-of-15 legacy multisig

# mempool package updates

insertion and removal require updating ancestors and descendants

👉 limit the number and size of ancestor and descendant packages


👉 … limit the cluster size too

# Transaction Relay / Mempool Policy

Bitcoin Core rejects some consensus-valid transactions

📌 **decentralization**: minimize memory, bandwidth, computation

📌 **security**: prevent DoS, specific bugs not fixed in consensus

📌 **upgrade hooks**: don’t use future things before they are defined (avoid collisions and confiscation)

📌 **best practices aka paternalism**: discouraging network resource-intensive practices

# default policy is *like* a transaction relay interface, but it's optional and varies

# Wait, but isn’t this CENSORSHIP?

# They’re all tradeoffs

- accuracy of assessing incentive compatibility / computational complexity
- allowing complex topologies / fixing pinning (censorship) problems
- relaying all consensus-valid transactions / safely updating consensus later

👉 to design for censorship-resistance (and other goals), we have to reject some

👉 the point is to agree on an interface that is both useful for users and acceptable for node operators, not stop users from broadcasting transactions

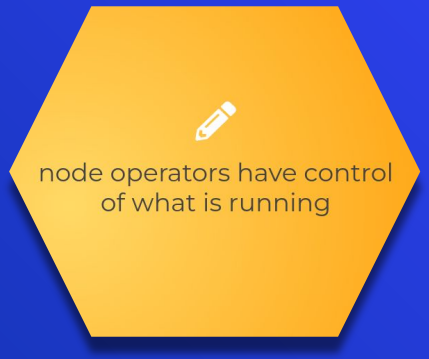
mempool policies can differ, but it's best for caches to be hot

# Practice question from twitter

- In the past, some rules were added to discourage practices proactively
  - uneconomical (“dust”) outputs
  - large (nonstandard) scriptPubKeys

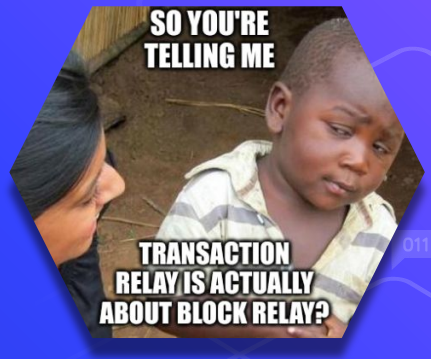
Q: Why can’t Bitcoin Core just filter {inscriptions, ordinals, brc-20s} too? #FixTheFilters

# “Just make Bitcoin Core filter these transactions”




miners are not obligated to run software that throws away fee revenue


this doesn’t work on transactions that already bypass policy rules

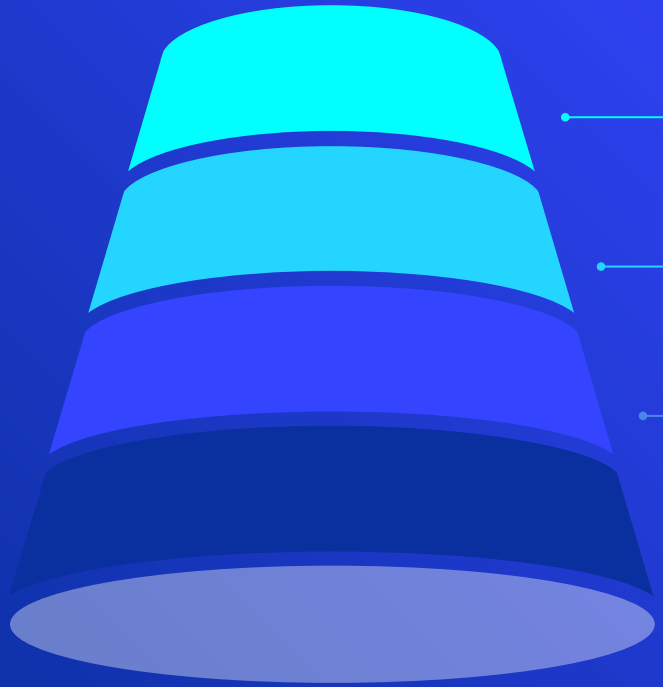



if you discard transactions that are likely to be mined, your cache is useless

# Summary

- 💸 Design goals like “censorship resistance” and “decentralization” aren’t easy to achieve
- 💸 They do create interesting technical puzzles
- 💸 All the code is open source - that’s important
- 💸 YOU can see for yourself how it works

# Go deeper!



**Subscribe to the Optech Newsletter**
bitcoinops.org

**Attend a Chaincode Learning Seminar**
chaincode.gitbook.io/seminars

**Join Bitcoin Core PR Review Club**
bitcoincore.reviews

**Contribute to the code**
github.com/bitcoin/bitcoin

# Thanks!

**Any questions?**

@glozow


slides template: slidescarnival.com