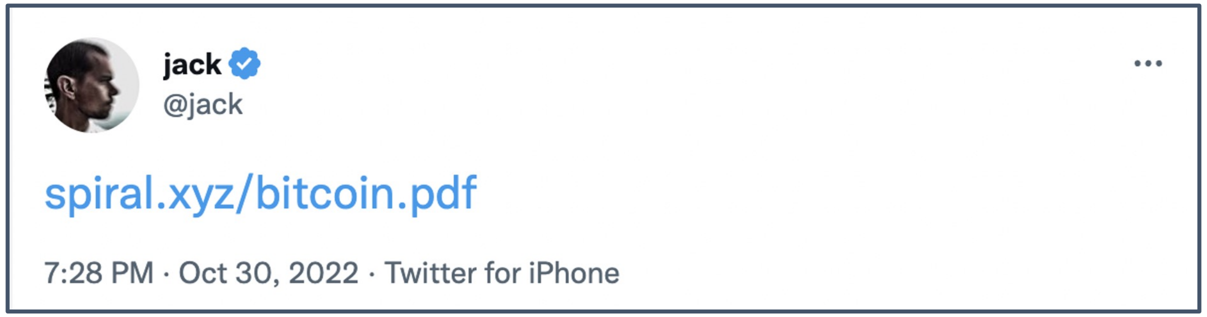
CSE 550: Systems for all

Au 2022

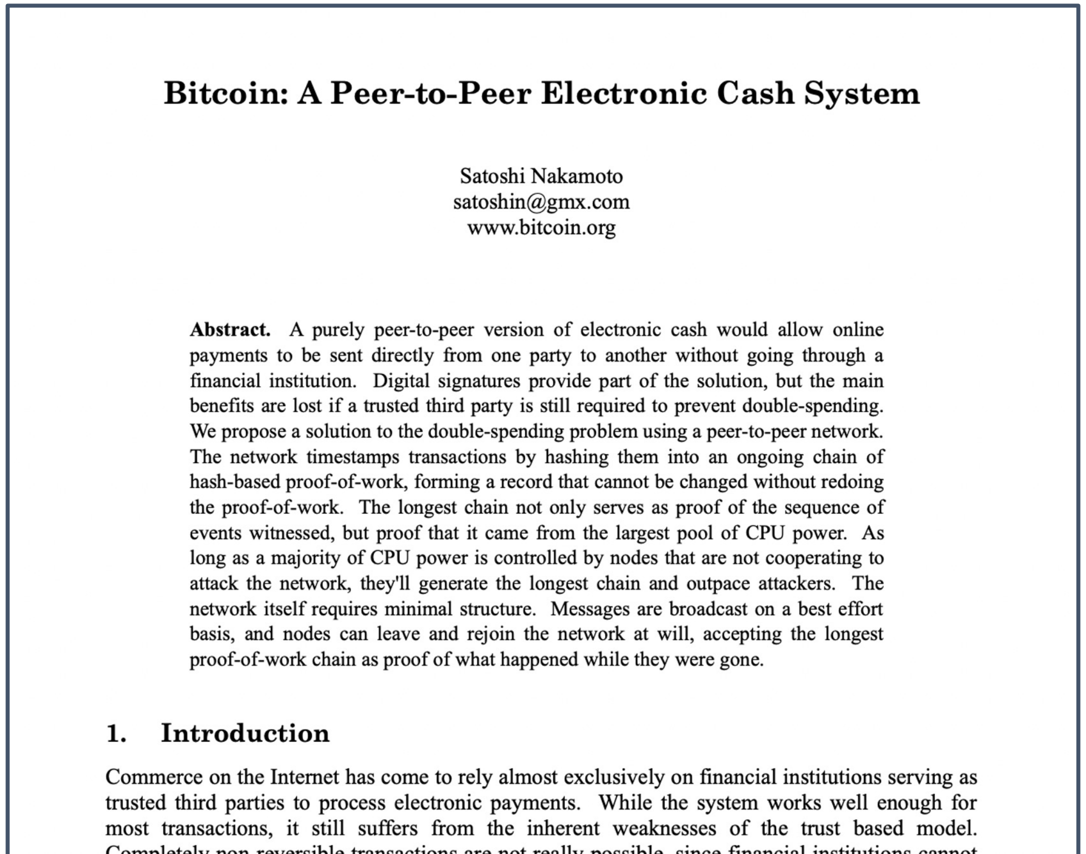
Ratul Mahajan



jack
@jack

spiral.xyz/bitcoin.pdf

7:28 PM · Oct 30, 2022 · Twitter for iPhone



Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot

Faults

Paxos does **fail-stop** faults

- Nodes are dead or follow the protocol
- Hard part: Is the node dead or network too slow?

But faults can be **Byzantine**

- Incorrect protocol implementation
- Willful lying

Handling Byzantine faults

Replicated state machine model

- Replicated servers hold long-term state
- Clients read and modify the state

Fault model

- Up to *f* replicas can fail
- Same attacker can control all *f* replicas
    - Knows what the honest replicas do
    - Can read all network messages
- Clients aren’t faulty
- Crypto cannot be broken

Strawman solution

$2f + 1$ replicas

Clients sends a command to all replicas

Waits for $f + 1$ identical replies

- With a max of f faults (and network eventually working), this will happen

What can go wrong?

Problem with the strawman

f+1 matching replies might be *f* bad nodes & 1 good

- so maybe only one good node got the operation
- next operation also waits for *f*+1
- might not include that one good node that saw op1

example: S1 S2 S3 (S1 is bad)

- everyone hears and replies to write(value="A")
- S1 and S2 reply to write(value="B"), but S3 misses it
- client can't wait for S3 since it may be the one faulty server
- S1 and S3 reply to read(value), but S2 misses it; read(value) yields "A”
- result: client tricked into accepting out-of-date state

What happens here with fail-stop failures?

Basic solution that works

$3f + 1$ replicas

Client waits for $2f+1$ matching replies

- f bad nodes plus a majority of the good nodes
- so all sets of $2f+1$ overlap in at least one good node

Remaining challenges

Message corruption, spoofing, …
→ Use cryptography

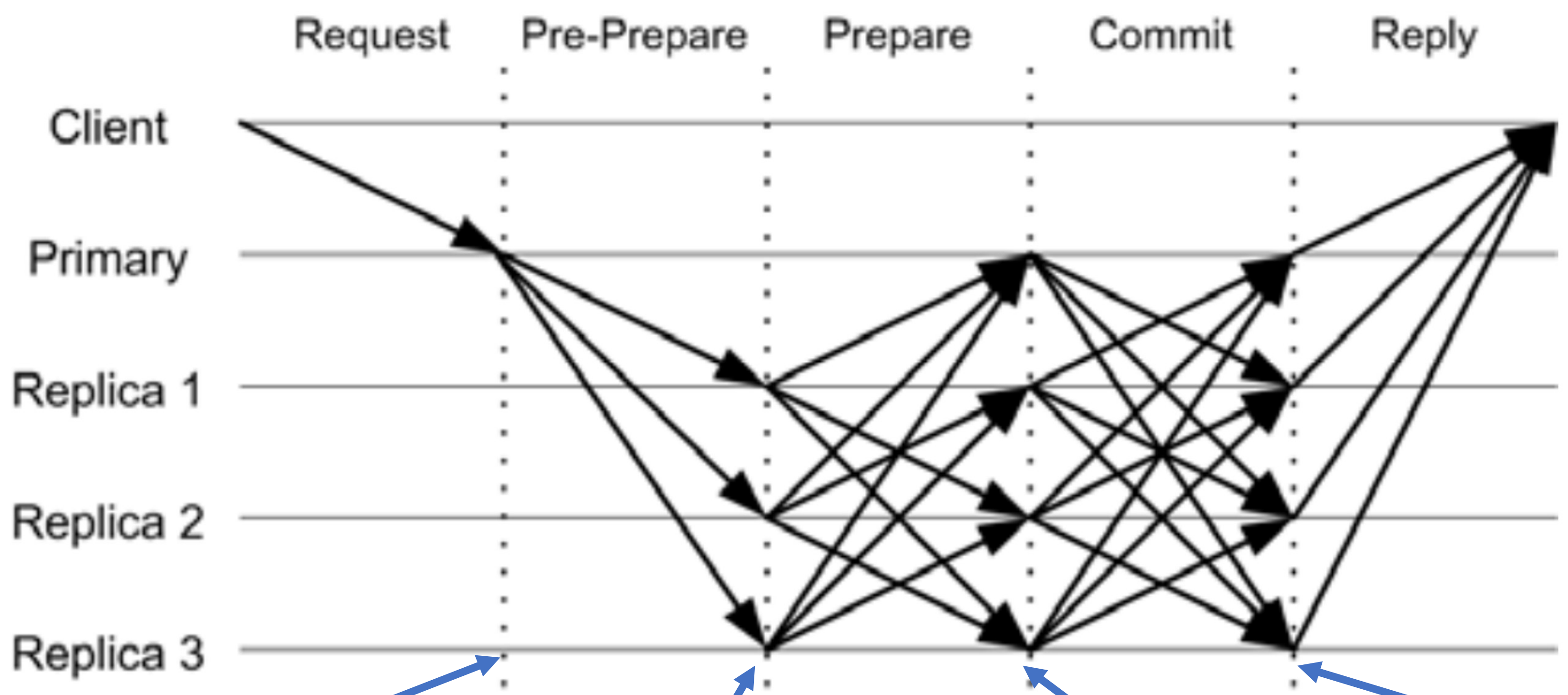
Concurrency

- Maintaining consistent state across all replicas requires that they all execute operations in the same order
- But different replicas will get messages in different orders, especially when you have multiple clients

→ Introduce a Primary

- But what if the primary is (Byzantine) faulty?

Practical Byzantine Fault Tolerance (Normal operation)



Assign sequence number to requests

Sent for the lowest sequence number

After receiving 2*f* Prepare messages

After receiving 2*f* Commit messages

BFT principle

You cannot trust anyone but you can trust the group
(assuming that the group is big enough)

Blockchains inherit ideas from BFT solutions

Power of groups

Broadcast

Cryptography

Over to Anjali