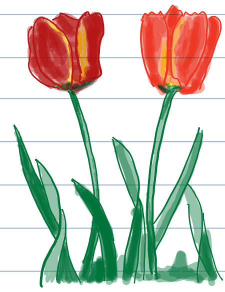
Secure Distributed Systems

CompSci 661 / 461



This video

- The F.L.P. impossibility result
- the CAP theorem (another impossibility result)
- a comparison of BFT systems to Nakamoto blockchains

PART I

Fischer, Lynch, and Patterson

The most important impossibility result concerning consensus.

FLP – in a distributed system in which messages cannot be guaranteed to be delivered within a known, finite time, no agreement is possible if even a single process is faulty

That does not mean the agreement/consensus reported is wrong; you just can't guarantee it won't change

There is another way to State the FLP result

No asynchronous consensus algorithm can guarantee both <u>safety</u> and <u>liveness</u> (if even a single process is faulty)

SAFETY – We say a system is <u>safe</u> if nothing catostrophic (i.e., unrecoverable) happens due to a failure to operate correctly.

For example; in our context of Consensus, we might say a protocol is safe because it has the property that once consensus is reached, it is never violated; i.e., the agreed upon value is not changed.

<u>LIVENESS</u> – We say a system is <u>live</u> is deadlock is not possible even though other failures are possible. Progress will always (eventually) be possible.

For example, in our context of consensus, we might say a protocol is live because a result is always (eventually) returned by the system (perhaps not the final result).

If a system can guarantee liveness but not safety, in terms of the definitions above, then we say that it is eventually consistent.

If instead a system can guarantee safety but not liveness, (in terms of above), then it will return a result regarding current consensus when it is sure that the result is correct; and provides no response until then.

Proof assumptions:

1. Network is reliable.
2. Messages are delivered correctly. (without error; only once; in order; etc)
3. No assumptions about the delay in delivering messages.
4. No synchronized clocks
5. No timeouts allowed (i.e., no deadlines allowed)
6. No ability given to peers to determine if a process is dead or just very slow.
7. No specific consensus algorithm stated

#6 is key difference for Blockchains versus BFTs.

- specifically, in permissionless blockchains where any one can mine, new blocks can come at any time.
- when they come — were you eclipsed?? or is it a late-coming addition to this chain? you don't know...
- you fundamentally can't solve this problem in blockchains.

PART II

Eric Brewer's CAP theorem

It's impossible for a distributed system to simultaneously provide more than two of these three qualities:

① Consistency ② Availability ③ Partition Tolerance

Three possibilities

① Consistency and Partition tolerance: BFT systems. They sacrifice availability, returning an answer only once there are sufficient responses (surviving temporary partitions). Lamport's algorithm ensures consistency when $\geq \frac{2m+1}{3m+1}$ non-faulty nodes exist.

② Availability and Partition tolerance: Blockchains. We know how to survive partitions in blockchains: we take the fork with more work (the longer one, if difficulty is the same). And, blockchains always have a response to queries: the latest block. Blockchains sacrifice guarantees of consistency, as they can never be sure a new fork won't arrive later (FLP).

③ Consistency an Availability: not a practical choice. Network partitions happen, and every system must guarantee they can be tolerated temporarily. But to have a system that can guarantee consistency and availability you could run BFT, and separate permanently at every partition. You'd have to refuse to re-unify when the partition ends, especially if consensus results are different. Changing consensus results violates consistency guarantees.

PART III

Comparison of BFT against Blockchains

See paper by Marko Vukolić

- Nakamoto's algorithm does not implement consensus in the traditional distributed computing sense
  - it executes "eventual consistency" or "probabalistic agreement"

- Vukolić notes that Bitcoin produces a block/10 minutes.
  - 250-byte minimum to specify a transaction
  - in Bitcoin core, blocks are 1MB maximum

So thats

$$\frac{1024 \times 1024 \text{ Bytes/block}}{250 \text{ Bytes/txn}} \times \frac{6 \text{ blocks}}{60 \cdot 60 \text{ secs}}$$

$$= 7 \text{ txns/sec}$$

Bitcoin cash is 8MB maximum, and they may increase.

$$= 7.8 \text{ txns/sec} = 56 \text{ txns/sec}$$

IN comparison, VISA is 2000 txns/sec (i.e. 286MB blocks/10 minutes $= \frac{1}{2}$ MBps)

- Vukolić mentions Ethereum, which we have not discussed yet.
  - while Bitcoin has restricted/limited scripting system, Ethereum has Virtual Machine (storage and computation) and a Turing complete language.

  - Ethereum addressed the problem of never-ending scripts by charging for each command/opcode executed.

Byzantine Fault Tolerant (BFT) state machine replication protocols are the result of 3 decades of research.

- They are practical, with minimal latencies over a network
- Typically support tens of Thousands txns/second.
- The challenge is in supporting a larger # of nodes.
  Everyone must broadcast results to every other node.

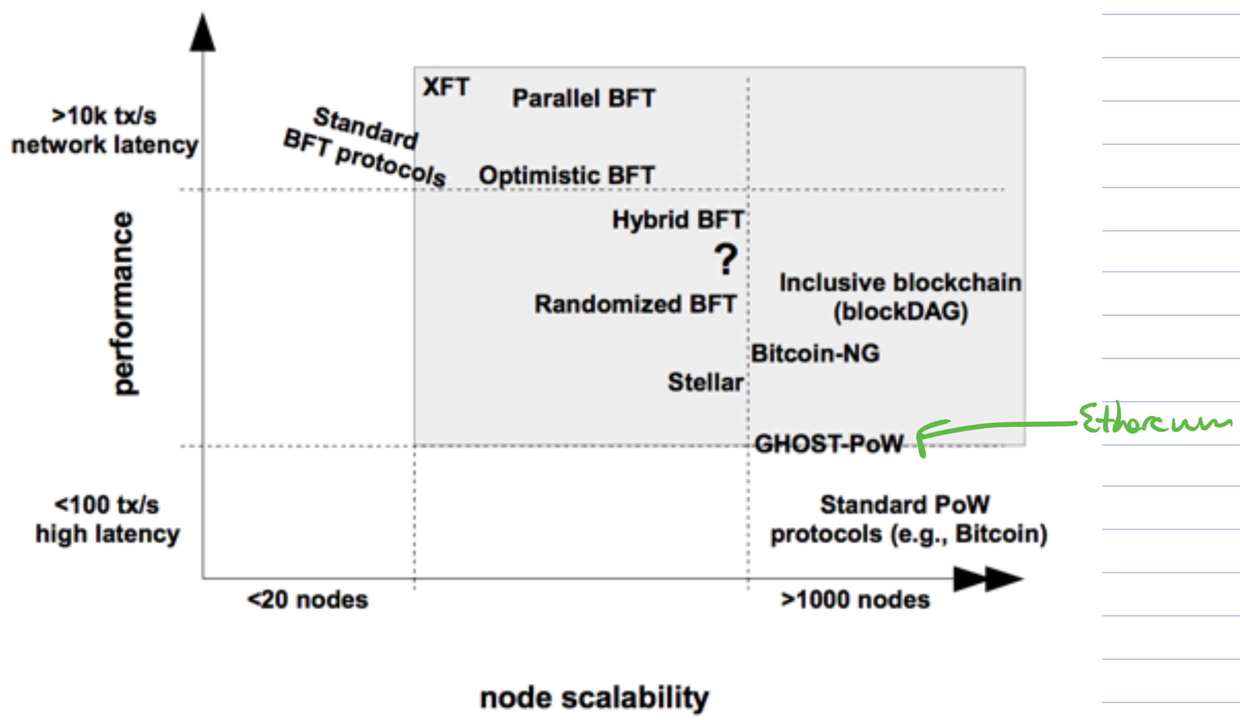
It is illuminating to compare Blockchains to BFT.

Roughly:

- P.O.W. Blockchains scale well with # of peers in terms of (constant) network traffic; but do not have high rate of txns/sec.
- BFT have too much traffic as # of peers increases (linear) but support high # of trans/sec.

Vukolić's table:

grey area is speculation by Vukolić



A high-level comparison of Blockchains (PoW) versus BFT

	PoW consensus	BFT consensus
① Node identity management	**open, entirely decentralized**	permissioned, nodes need to know IDs of all other nodes
② Consensus finality	no	**yes**
③ Scalability (no. of nodes)	**excellent (thousands of nodes)**	limited, not well explored (tested only up to $n \leq 20$ nodes)
④ Scalability (no. of clients)	**excellent (thousands of clients)**	**excellent (thousands of clients)**
⑤ Performance (throughput)	limited (due to possible of chain forks)	**excellent (tens of thousands tx/sec)**
⑥ Performance (latency)	high latency (due to multi-block confirmations)	**excellent (matches network latency)**
⑦ Power consumption	very poor (PoW wastes energy)	**good**
⑧ Tolerated power of an adversary	$\leq 25\%$ computing power	$\leq 33\%$ voting power
⑨ Network synchrony assumptions	physical clock timestamps (e.g., for block validity)	**none for consensus safety** (synchrony needed for liveness)
⑩ Correctness proofs	no	**yes**

① NODE IDENTITY MANAGEMENT

- perhaps their most fundamental difference
- POW - Anyone can join (permissionless)

BFT - requires every node to know the entire set of its peers.
- requires a centralized authority - per Douceur.
- Sometimes regulatory issues require knowing everyone. (markets, HIPPA, etc)
- These are called permissioned chains

② Consensus Finality

This is a property that is not in blockchains but can be in BFT.

It requires that a valid block, once appended by a node can never be removed.

Consensus Finality can follow from two other properties:

ⓐ **Total Order Delivery**: if process (or peer, etc) p and q both T.O.D. messages m and m′, then
p T.O.D.'s m before m′ IF and ONLY IF
q T.O.D.'s m before m′.

ⓑ **Agreement**: if a process T.O.D.'s a message m, then all correct (i.e., non-faulty) processes eventually T.O.D. m.

With both T.O.D. and Agreement in place, we have:

Definition 1: Consensus Finality

IF a correct node p appends block b to its copy of blockchain before appending block b′,
Then no correct node q appends block b′ before b.

This definition is not satisfied by PoW blockchains. because forks are possible at any time, even if always resolved.

- Temporary forks are resolved by highest difficulty chain.
- But the very presence implies no consensus finality.

IS satisfied by all BFT blockchains (assuming sufficient correct peers)

③ Scalability of Nodes and Clients

- Blockchains potentially support 1000s of miners — each block requires one message to each miner to announce it.
- BFT isn't as scalable. Every node must contact every other node to process consensus.

④ Both scale well in terms of clients (1000s of clients)

⑤ Scalability of transactions per second and ⑥ Throughput and latency

P.O.W. has a relatively low transactions per second rate.

To fix? We need either larger blocks or shorter time between blocks.

- If blocks are larger (more transactions per block), it can take longer to propagate across the Internet, which means greater chance of forks.
- If there is a shorter time between blocks, then we also end up with more forks.

(Solutions to both problems in later classes.)

Performance is limited in P.O.W./blockchain approach.

BFT systems don't have these problems.

(7) POWER / ELECTRICITY

No contest!

Proof of Stake is one solution to energy problem. We'll take a look later in this course.

(8) Adversaries (malicious nodes)

P.O.W.: what matters is hashing/mining power.
$\geq 50\%$ must be honest (or consider selfish mining)

BFT: $\geq 2m+1$ out of $3m+1$ must act correctly.

(9) NETWORK SYNCHRONY

Blockchains rely on timestamps (to calculate difficulty)

BFT doesn't rely on physical clocks
uses Lamport Clocks.

BUT synchronous communication is needed to avoid FLP.

(10) CORRECTNESS PROOFS

no mathematical proofs

distributed systems versus distributed computing

Better PoWs

GHOST

BITCOIN-NG

Lower overhead in BFT

STELLAR [44]

optimistic BFT protocols [32,3]

XFT [40]

hardware assumptions [32]

RANDOMIZED BFT

Hybrid

Decker et. al [15]

SCP [42]