Video of my keynote talk "Permissionless Consensus" at ICDCN '22. https://youtube.com/watch?v=EfsSV7ni2ZM… Theme: do the desired guarantees of a blockchain protocol dictate how it must be implemented? Includes highlights from a couple

years of joint research with Andy Lewis-Pye. Details 1/31

First 60% of talk: tutorial/review/bird's-eye view of permissioned vs. permissionless consensus. Consensus, informally: keeping a bunch of computers in sync, despite network outages and malicious attacks. 2/31

State machine replication (SMR): clients submit transactions (txs) to nodes who each locally maintain an ever-growing ordered sequence of txs. Desired properties: consistency (nodes stay in sync), liveness (submitted txs eventually executed). Very relevant for blockchains! 3/31

Wasn't SMR solved in the 1980s (see e.g. Turing Awards given to Liskov and Lamport)? Yes and no. Review of two famous results from the 1980s: 4/31

Famous result 1 (Dolev-Strong '83): in the synchronous model (bounded max msg delay), get everything you want (consistency+liveness, no matter how many nodes are faulty/Byzantine). [Fine print: authenticated setting.] 5/31

Famous result 2 (Dwork-Lynch-Stockmeyer '88): in the partially synchronous model (with periods of unbounded msg delays), can solve SMR if and only if <33% nodes Byzantine. (This is where the "33%" that you see in blockchain whitepapers and L1 debates comes from.) 6/31

Viewed as SMR protocols, Bitcoin/Ethereum look terrible: in synch model, can only tolerate 49% (rather than 99%) Byzantine. In partial synch model, no consistency guarantees even with 0% Byzantine. 7/31

Nakamoto very much knew about classical consensus. https://metzdowd.com/pipermail/cryptography/2008-November/014849.html… Q: Why did he bother inventing a seemingly inferior consensus protocol? 8/31

A: Classical protocols were designed for the *permissioned* setting--set of nodes is static and known up front. (This made sense for 1980s-era SMR applications.) Nakamoto wanted a *permissionless* protocol--anyone can spin up a node at any time, no registration required. 9/31

Classical protocols rely on (non-sybil-resistant) voting, so not at all clear how to adapt them to the permissionless setting. So Nakamoto invented a new protocol, with two radical new ideas. 10/31

Big idea #1: proof-of-work for sybil-resistant leader election (one-CPU-one-vote). PoW was originally proposed for spam-fighting, had been used previously in Hashcash, but its application in Bitcoin to achieving permissionless consensus was totally novel. 11/31

Big idea #2: embrace forks as the price of doing business, resolve them programmatically in-protocol via the longest-chain rule. (Honest nodes coordinate on the longest chain, which are the only blocks that matter.) 12/31

Point: Bitcoin incomparable to classical consensus protocols---permissionless (good), weaker liveness/consistency guarantees (bad). Qs: are these sacrifices necessary for permissionless consensus? Must permissionless consensus use PoW? Must it use longest-chain? 13/31

A: neither PoW nor longest-chain is required for permissionless consensus.
E.g., proof-of-stake sybil-resistance is a popular replacement for PoW.
Majority of PoS protocols are longest-chain (mimicking Bitcoin) or BFT-type (piggybacking on classical permissioned protocols).
14/31

Example (among many): Algorand.
PoS, each block produced by a randomly chosen committee who run a BFT-type protocol.
Inherits consistency guarantees from permissioned subroutine (see talk for caveats), e.g., consistency in partial synch model with <33% Byzantine stake.
15/31

PoW longest-chain protocols like Bitcoin look very different from PoS BFT-type protocols like Algorand, and also have different guarantees.
Coincidence?
Or, does the goal of consistency in the partial synch model fundamentally force PoS and/or BFT-type consensus?
16/31

Last 40% of talk: three recent research results (joint with Andy Lewis-Pye) that provide answers to these and similar questions.
For more details and further results see:
http://timroughgarden.org/papers/RPCAP_public_arxiv.pdf
https://arxiv.org/pdf/2109.04848.pdf
https://arxiv.org/pdf/2101.07095.pdf
17/31

First contribution: general model allowing direct comparisons between very different blockchain protocols (e.g., Bitcoin vs. Algorand).
Key component: *resource pool* (e.g., hashrate or stake), which controls extent to which different nodes can produce blocks, vote, etc.
18/31

Key distinction: -sized setting=blockchain state determines resource balances (e.g., typical PoS) -unsized setting=resource balances independent of blockchain state (e.g., typical PoW)
Distinction turns out to matter a lot! 19/31

Theorem 1 (cf., CAP Theorem):
No blockchain protocol satisfies:
(1) Operates in unsized setting
(2) Adaptively live in synch setting
(3) Consistent in partial synch setting
(see talk for proof sketch)
20/31

"Adaptively live"=remains live even under massive (e.g., 100x) drops in total resource balance.
Bitcoin satisfies properties (1) and (2).
Algorand satisfies (2) and (3).
Our theorem shows you can't have them all!
21/31

Main takeaway from Theorem 1: PoS (or otherwise operating in the sized setting) fundamental to the consistency guarantees offered by e.g. Algorand.
22/31

Next: let's zoom in on protocols that satisfy (2) and (3) (and thus necessarily operate in the sized setting).
To what extent must such protocols "resemble" BFT-type protocols like Algorand?
E.g., could a longest-chain PoS protocol achieve these same guarantees?
23/31

We capture the idea of "resembling Algorand" with our definition of *certificates*.
Intuition (see paper for precise defn): a protocol produces certificates if a node's opinion of a block can only flip from "unconfirmed" to "confirmed," and never vice versa.
24/31

E.g., typical longest-chain protocols can suffer re-orgs and thus do not produce certificates (always possible you'll change your mind if a new longest chain gets announced).
BFT-type protocols like Algorand are based on counting approval votes and produce certificates.
25/31

Theorem 2: a blockchain protocol satisfies consistency in the partial synch setting *if and only if* it produces certificates. ("if" direction is easy, "only if" direction less so)
26/31

Main takeaway from Theorem 2: BFT-type consensus (or otherwise producing certificates) is fundamental to the consistency guarantees offered by e.g. Algorand. 27/31

Conjecture: there's a general theory of permissionless consensus (e.g., delineating when it is/isn't possible) waiting to be developed, plausibly as rich as the existing theory for the permissioned setting.
Next result offers some supporting evidence.
28/31

Theorem 3: for the (single-shot) Byzantine broadcast problem, no deterministic permissionless protocol guarantees validity, agreement, and termination. (cf., the Dolev-Strong protocol for the permissioned case)
(proof somewhat difficult)
29/31

Main takeaway from Theorem 3: permissionless consensus is fundamentally more difficult than permissioned consensus!
30/31

Thanks to Andy Lewis-Pye for being such a fun and brilliant collaborator, and to John Augustine and the rest of the ICDCN organizing committee for the invitation!
31/31