A sledgehammer

to crack

a nut

why blockchain is not (always) a good idea

and what we can do about it

HYDRA 2022

Petr Kuznetsov, Télécom Paris, IP Paris

INSTITUT POLYTECHNIQUE DE PARIS

l'École Polytechnique – IP Paris

TELECOM Paris – IP Paris

TELECOM SudParis – IP Paris

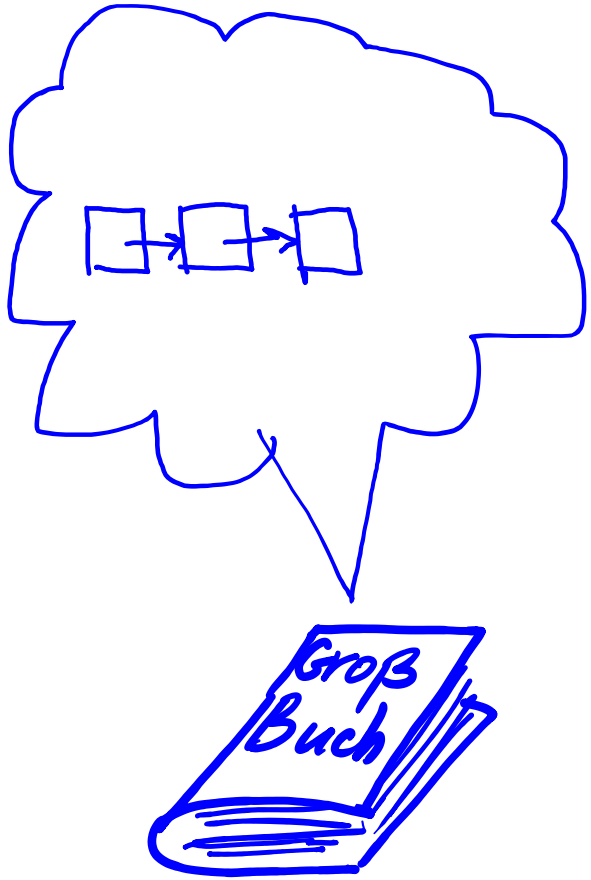
ENSTA – IP Paris

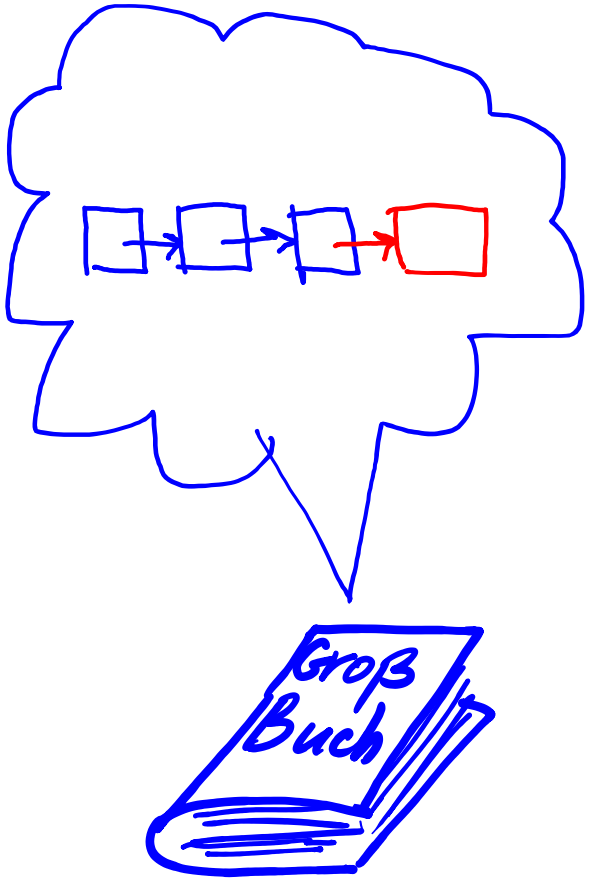
ENSAE – IP Paris

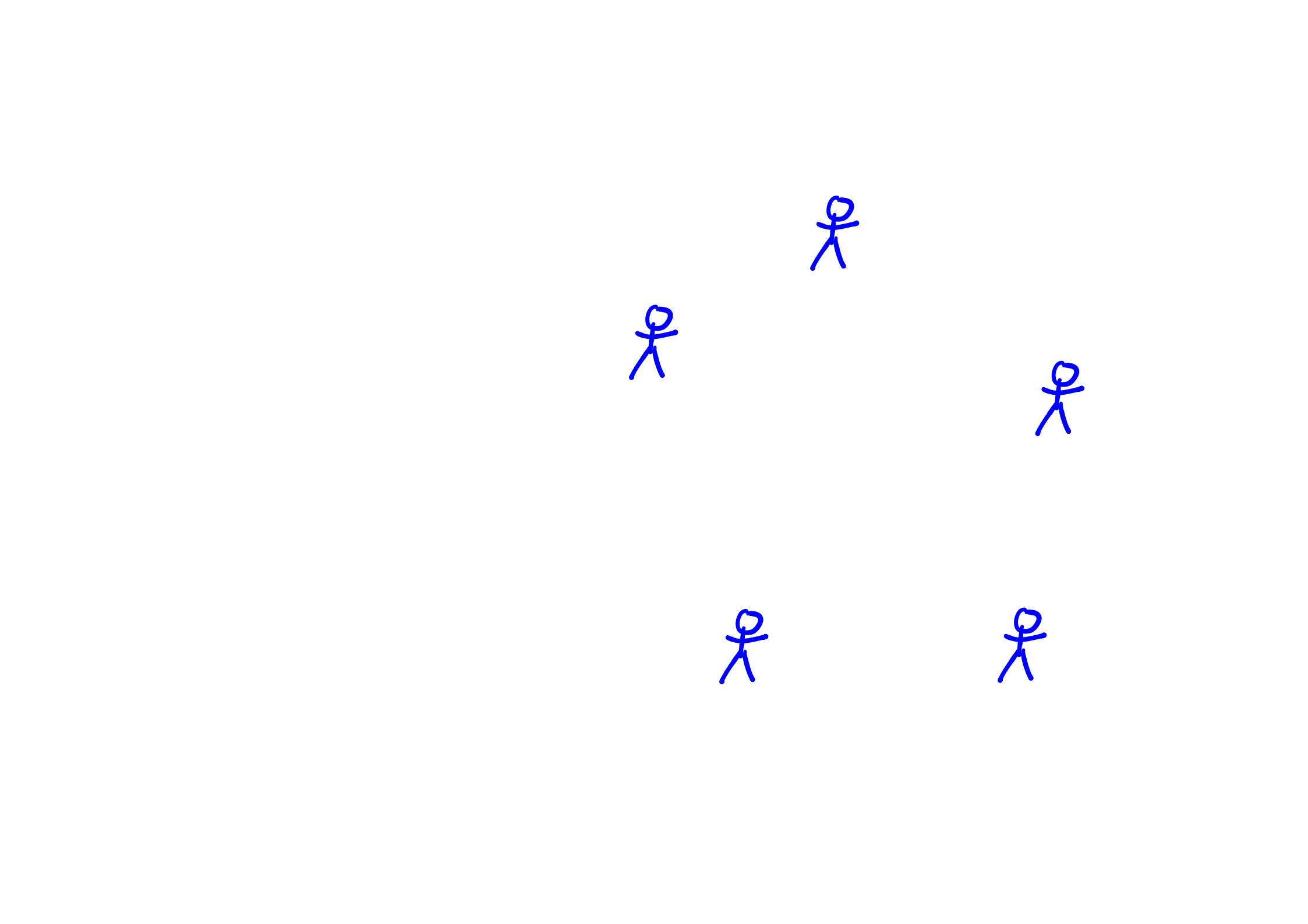
7,325 students
IP Paris plans to host and train nearly 10,000 students by 2022.

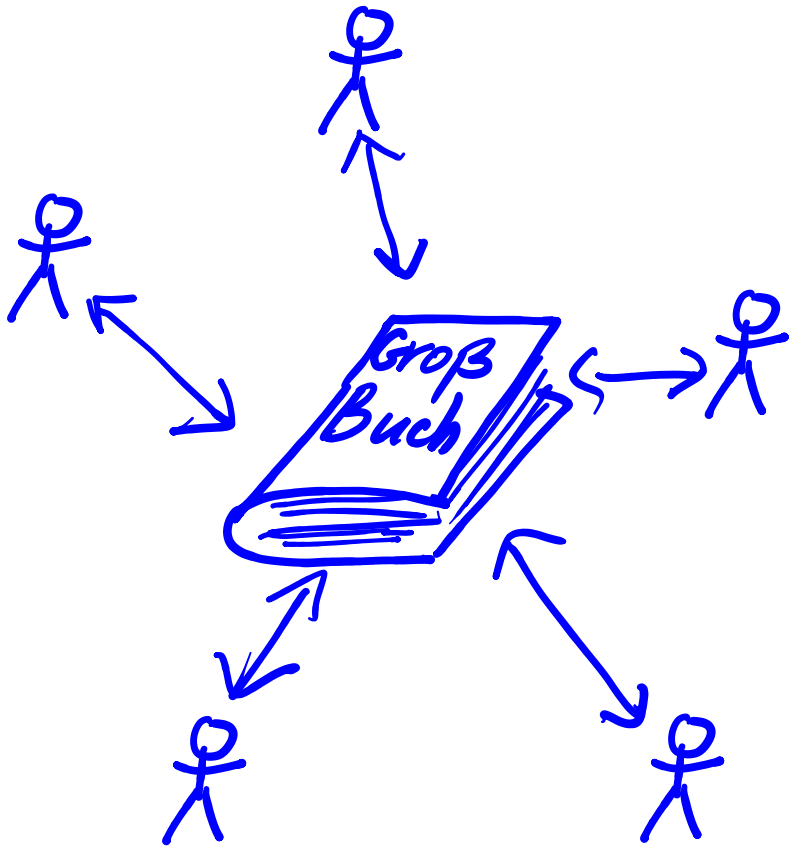
15 master's programs
IP Paris covers all fields of the science of tomorrow, from mechanics to nuclear engineering.

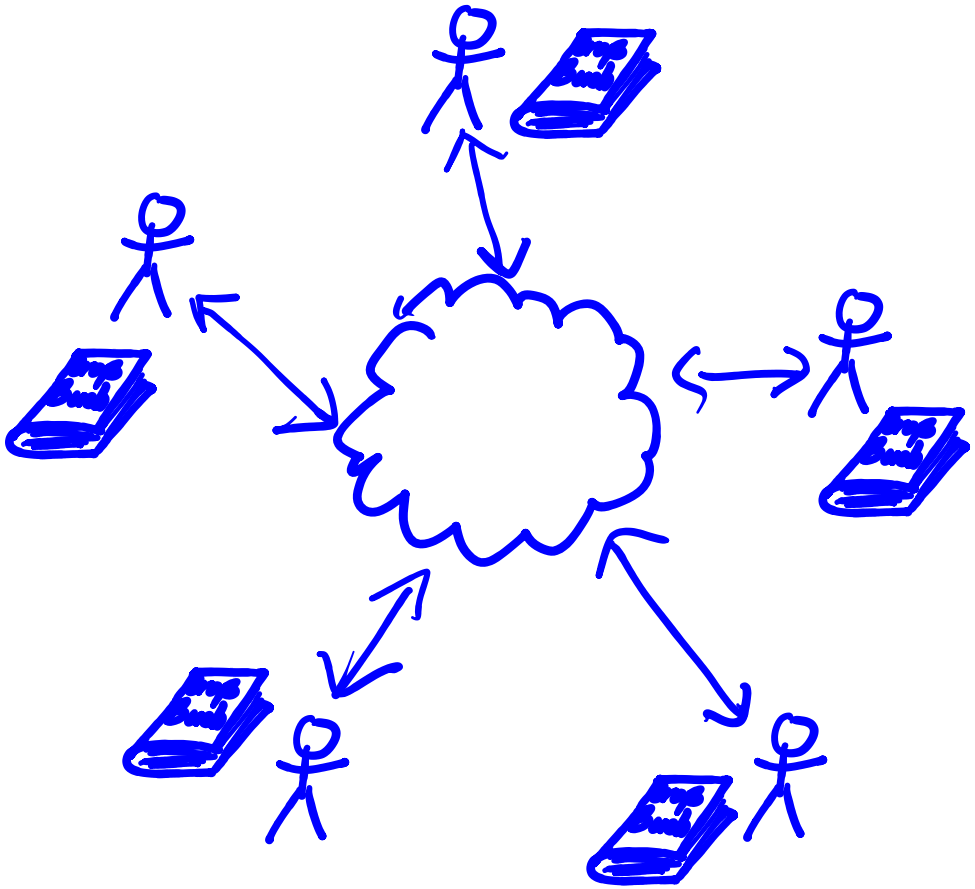
1,000 PhD students

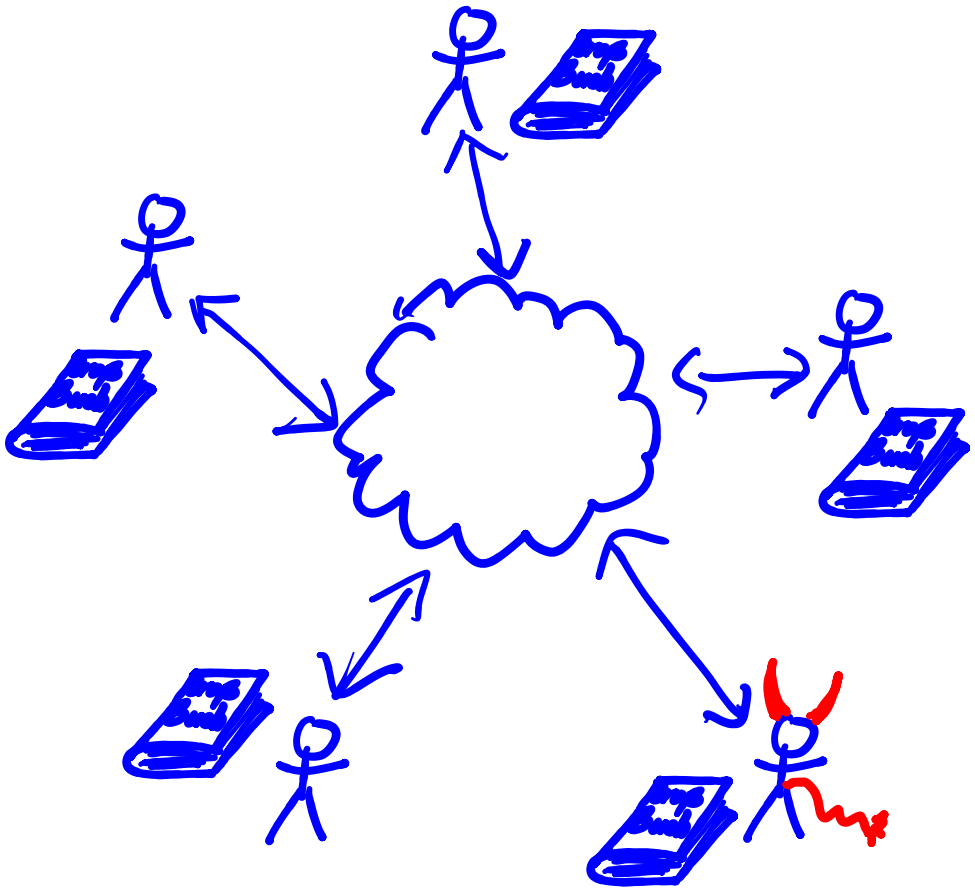

Groß
Buch

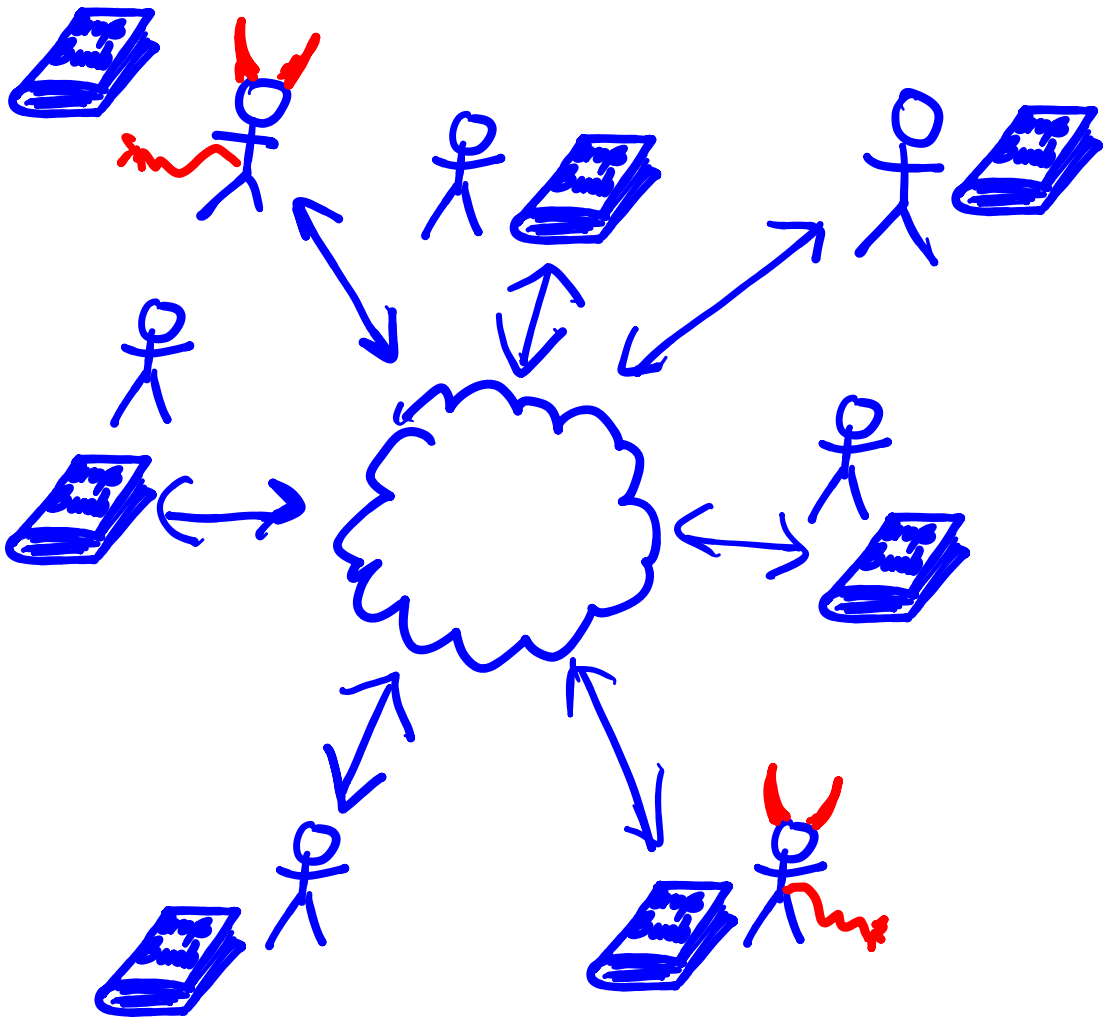

Groß
Buch




Groß
Buch



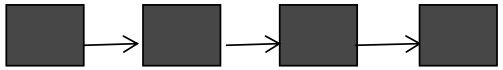




Blockchain = distributed ledger

Shared data structure: linear record of (blocks of) transactions

- Append and read
- Consistent: total order (consensus)



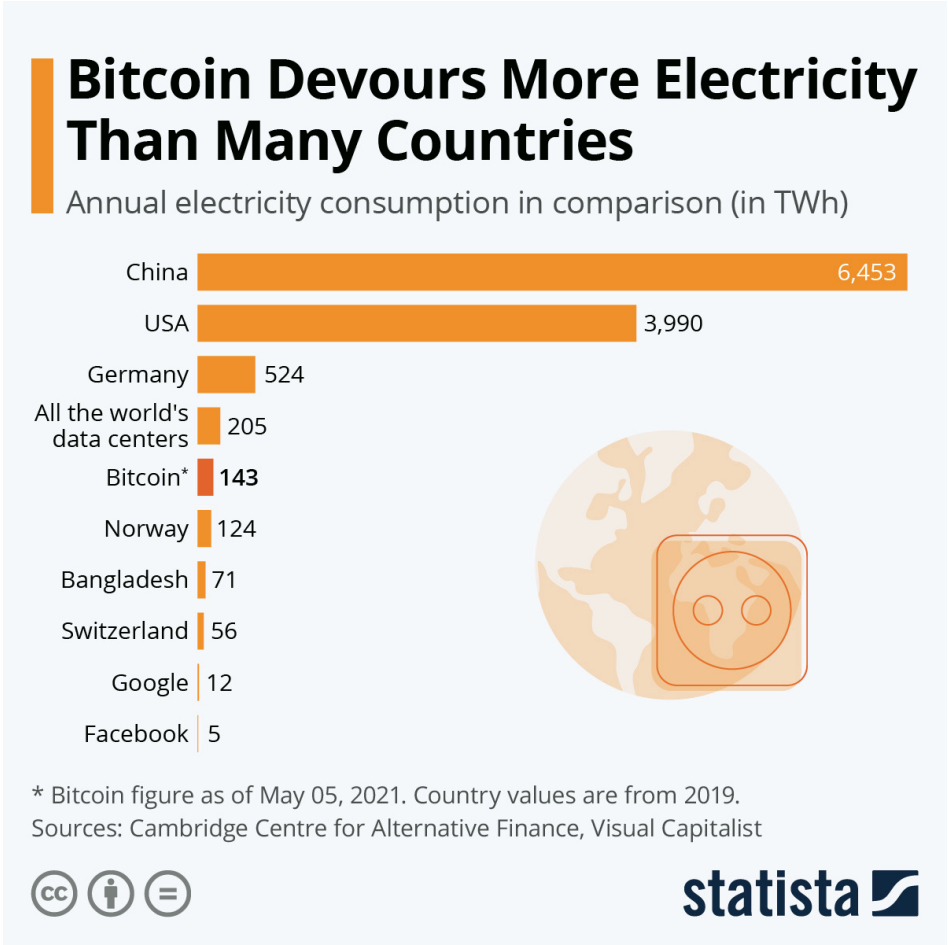
Open (permissionless) environment:

- No static membership
- No identities (public keys)
- Sybil attack: any participant subset can be adversarial

Classical (partially synchronous quorum-based) BFT protocols do not work!

Sybil-resistant consistency: PoW “consensus”

- Synchrony: slow down updates
- Solve a difficult puzzle before updating (PoW)
- Throughput low by design



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.

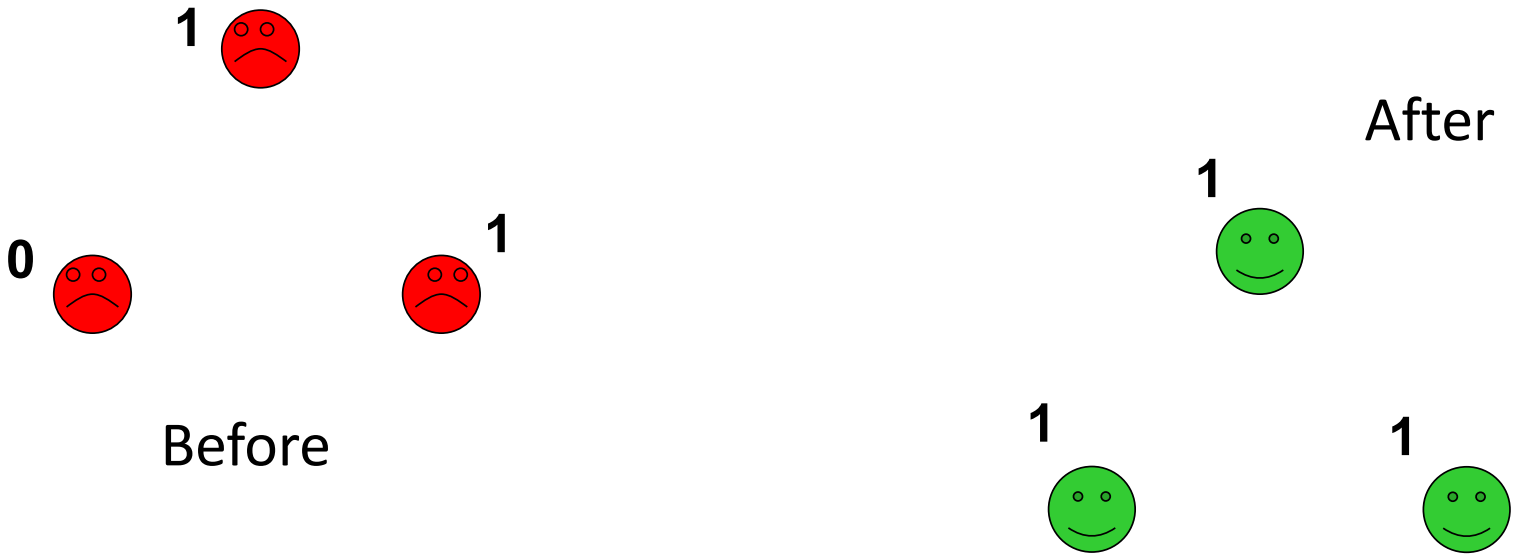
...

The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received.

Is consensus necessary?

Consensus

Processes *propose* values and must *agree* on a common decision value so that the decided value is a proposed value of some process



Why consensus is interesting?

Because it is universal!

- A key to implement a generic fault-tolerant service (replicated state machine or blockchain)

Expensive and cumbersome

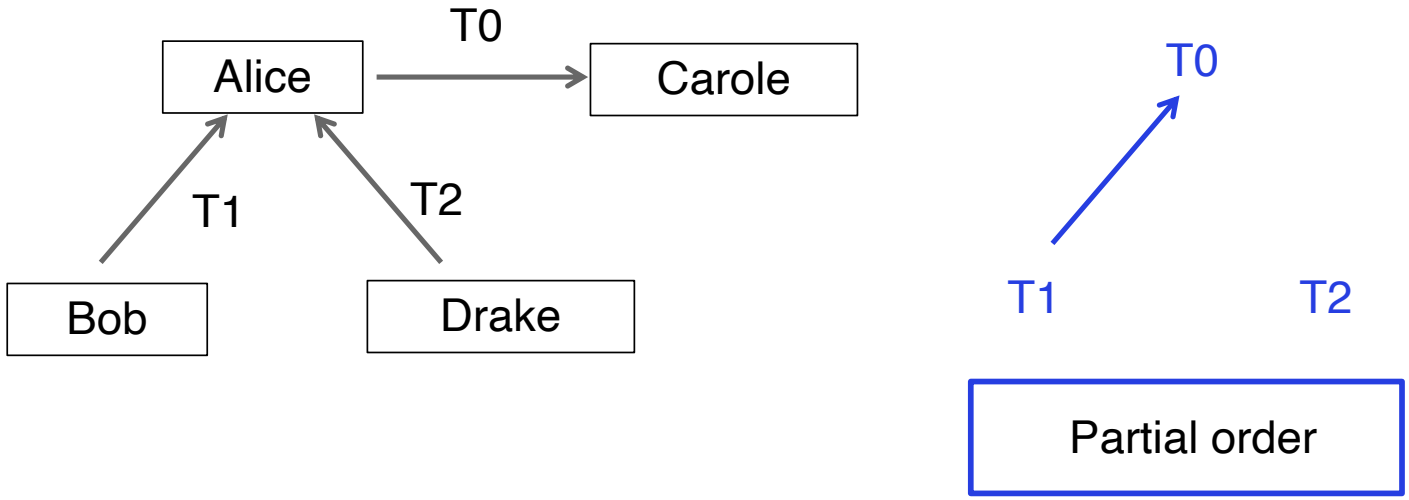
Is consensus necessary for a cryptocurrency?

Guerraoui et al. The consensus number of cryptocurrency. PODC 2019

Commutativity and causality

- T0: $100 from Alice to Carole
- T1: $100 from Bob to Alice
- T2: $100 from Drake to Alice

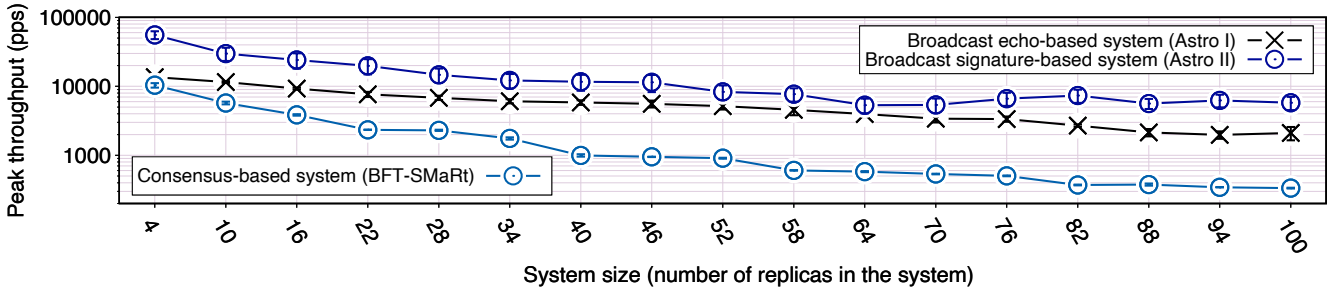
T0 causally depends on T1 (not enough funds otherwise)
T1 and T2 commute (T0 succeeds regardless of the order)

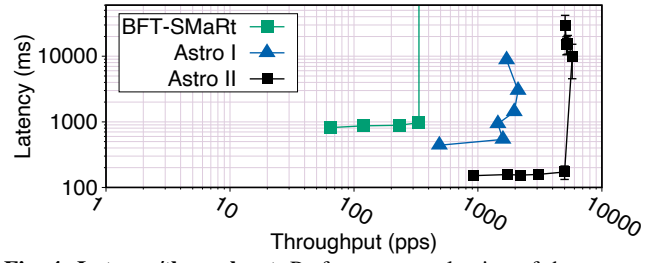


Consensus-less cryptocurrency

- Each transfer relates to its causal past (incoming/outgoing transactions)
- Make sure that a faulty account holder cannot lie about its causal past
- Secure broadcast [Bracha, 1987, Malkhi-Reiter, 1997]
  - ✓Source-order: messages by the same source are delivered in the same order

Collins et al. Online payments by merely broadcasting messages [DSN20]

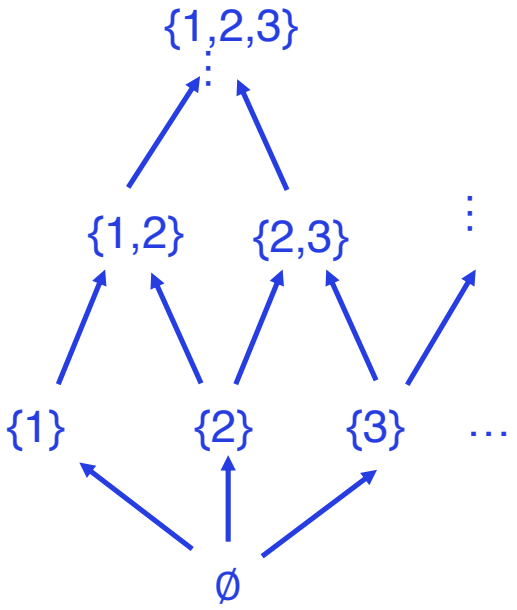




Alternative to consensus: Lattice Agreement on $(L, \sqsubseteq, \sqcup)$

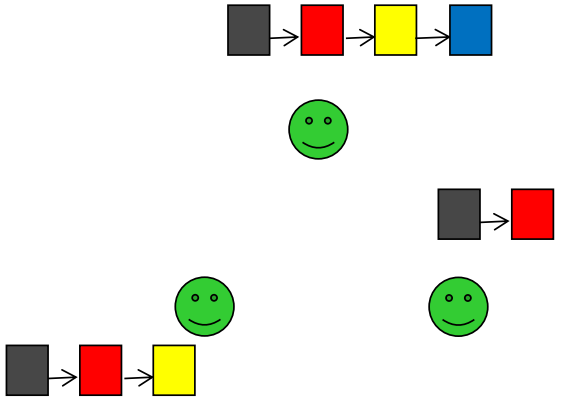
L – set of values, $\sqsubseteq$ - partial order, $\sqcup$ - join operator

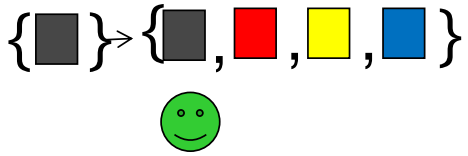
- Comparability: all learned values are comparable
- Validity: every learned value is a join of proposed values
- Liveness: every value proposed by a correct process eventually appears in a learned value



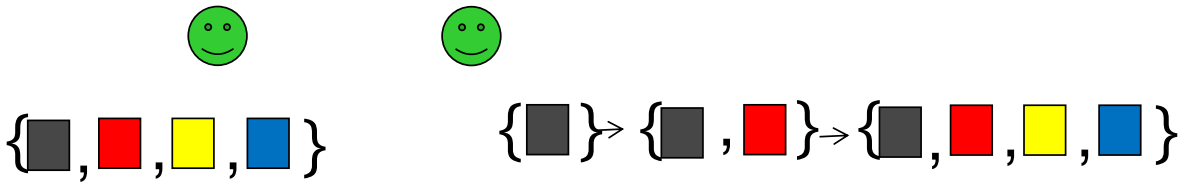
Total order vs. partial order

- Consensus = total order
  - ✓Participants learn an ordered sequence





- Lattice agreement = partial order
  - ✓Participants learn a partially ordered sequence



(Join semi) lattices

$$(L, \sqsubseteq, \sqcup)$$

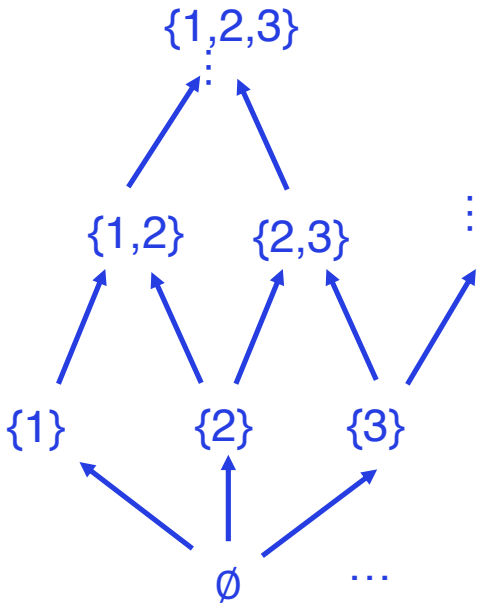
- L is a set of values
- $\sqsubseteq$ partial order on L
- $\sqcup$ join (least upper-bound) operator on L:
  $\forall V \subseteq L, \sqcup V = \min\{u \in L : \forall v \in V, v \sqsubseteq u\}$
- Origin u_0: $\forall u \in L : u_0 \sqsubseteq u$

Set

A set of values with operations *add* and *read*

$$(Lset, \sqsubseteq_{set}, \sqcup_{set})$$

- $L_{set} = 2^{\mathbb{N}}$
- $\sqsubseteq_{set} = \subseteq$
- $\sqcup_{set} = \cup$
- $\emptyset$



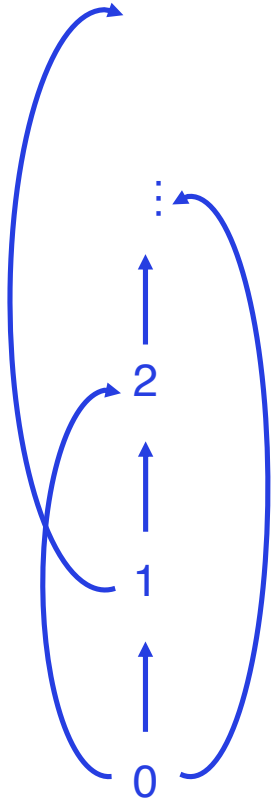
A remove operation can be represented as $add(-v)$, assuming $L_{set} = 2^{Z}$

Max register

Read-write variable: every *read* returns the largest written value

$$(LMR, \sqsubseteq_{MR}, \sqcup_{MR})$$

- $L_{MR} = \mathbb{N}$
- $\sqsubseteq_{MR} = \leq$
- $x \sqcup_{MR} y = \max(x, y)$
- 0



Composed lattice

$$(L, \sqsubseteq, \sqcup) = (L_1, \sqsubseteq_1, \sqcup_1) \times (L_2, \sqsubseteq_2, \sqcup_2)$$

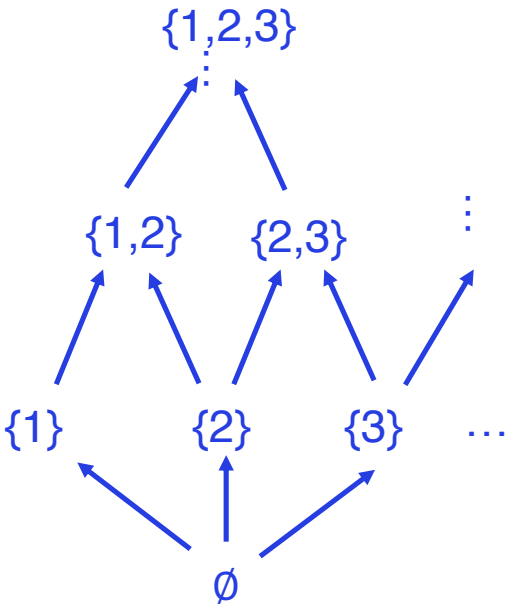
- $L = L_1 \times L_2$
- $(x_1, x_2) \sqsubseteq (y_1, y_2) \Leftrightarrow x_1 \sqsubseteq_1 y_1 \wedge x_2 \sqsubseteq_2 y_2$
- $(x_1, x_2) \sqcup (y_1, y_2) = (x_1 \sqcup_1 y_1, x_2 \sqcup_2 y_2)$

E.g.,N max registers: $\times_{i=1,\ldots,N} (L^i_{MR}, \sqsubseteq^i_{MR}, \sqcup^i_{MR})$

Lattice Agreement on $(L, \sqsubseteq, \sqcup)$

L – set of values, $\sqsubseteq$ - partial order, $\sqcup$ - join operator

- Comparability: all learned values are comparable
- Validity: every learned value is a join of proposed values
- Liveness: every value proposed by a correct process eventually appears in a learned value



Allows for efficient asynchronous implementations [FRR+, 2012]

Perfect fit for asynchronous reconfiguration [OPODIS19,DISC20]

Using LA: atomic snapshot

Solve LA on the composition of m max-registers

$$\times_{i=1,\ldots,m} (L^i_{MR}, \sqsubseteq^i_{MR}, \sqcup^i_{MR})$$

Each storing a pair *(seqnum,value)*

To update position i:

- Read max-register i
- Write back with a higher *seqnum*

To take snapshot:

- Read all max-registers

Snapshots are totally ordered!

Using LA: asset transfer

Solve LA on the composition of m sets

$$\times_{i=1,\ldots,m} (L^i_{set}, \sqsubseteq^i_{set}, \sqcup^i_{set})$$

Each set is the *state* of an account (set of outgoing transfers)

To perform a transfer (a,b,x):

- Read the set, check the balance of x
- Add x to set b and remove from set a
- Propose the new state

Total order of sets implies atomicity of transfers

Reconfiguration?

A set of replicas of a lattice agreement protocol may be reconfigured over time

Reconfiguration?

A set of replicas of a lattice agreement protocol may be reconfigured over time

Reconfiguration in storage systems:

- consensus-based [Rambo'02,03,10]
- asynchronous [Dynastore'11,…]

Configuration lattice

Abstract lattice $(Cf, \sqsubseteq_{cf}, \sqcup_{cf})$

Every element $C \in Cf$ carries:

- $C.members$ – a set of replicas
- $C.quorums \subseteq 2^{C.members}$
- Other attributes (used in $\sqcup_{cf}$)

Configuration lattice: example

$$(Cf, \sqsubseteq_{cf}, \sqcup_{cf})$$

- Elements of Cf are of type {+1,+2,+3,-2}
- $\sqsubseteq_{cf} = \subseteq$ and $\sqcup_{cf} = \cup$
- $C.members$ – all added and not yet removed
- $C.quorums$ – all majorities of $C.members$

Reconfigurable object

- Solve lattice agreement on

$$(L, \sqsubseteq, \sqcup) \times (Cf, \sqsubseteq_{cf}, \sqcup_{cf})$$

- Every configuration update results in a join of proposed configurations
- Every update should be installed in a quorum of each candidate configuration
- A new decided state makes all preceding configurations obsolete

A configuration must be available as long as it is not obsolete!

Plug-and-play reconfiguration

Plug the corresponding lattice and get:

- Max register
- Set
- Atomic snapshot
- Conflict detector
- Commit-adopt
- Safe agreement
- Asset transfer
- …

Kuznetsov, Rieutord, Tucci. Reconfigurable lattice agreement. OPODIS'19

Kuznetsov, Tonkikh. Asynchronous Reconfiguration with Byzantine Failures. DISC'20

Permissionless asset transfer?

- Bitcoin [Nakamoto 2008] and Ethereum [Wood 2015]: **consensus** and **proof-of-work** mechanism.
- **Proof-of-stake** [Bentov et al. 2016, Chen et al. 2019, Kiayias et al. 2017], **proof-of-space** [Dziembowski et al. 2015], **proof-of-space-time** [Moran et al. 2016]: **synchronous** networks, **consensus** and **randomization**.
- **Asynchronous** solutions [Guerraoui et al. 2019, Collins et al. 2020] are built on top of **reliable broadcast** instead of consensus. Quorum-based -> not Sybil-resistant

Kuznetsov, Pignolet, Ponomarev, Tonkikh. Permissionless and asynchronous asset transfer. DISC'21

Permissionless and asynchronous asset transfer [DISC 2021]

Idea:

- Use weighted (stake-based) quorums
- A transaction is accepted if validated by >2/3 of stake

Issues:

- Measuring stake in an asynchronous system?
- Facing dynamic Byzantine adversary?

Solution:

- Treat stake distribution as a configuration
- A transaction is a reconfiguration call
- Reconfigurable Lattice Agreement as a building block

Strong consistency of data in a large scale: a hard problem in a hard model?

- Relax the problem
  - ✓Asset transfer (LADT [OPODIS19]) instead of blockchain [PODC 2019,DSN 2020, DISC 2021]
  - ✓Multiple spending [Bezerra et al., PODC 2022]
  - ✓Accountability vs. fault-tolerance [Freitas et al., OPODIS 2021]
- Strengthen the model
  - ✓(Eventual) synchrony
  - ✓Stake assumptions
  - ✓Some trust (federated quorums)

TrustShare: Innovation Chair

- Asynchronous cryptocurrency [PODC 2019,DISC 2019,DSN 2020]
  - ✓Use stake for permissionless asset exchange [arxiv:2105.04966]
- Accountability [SOSP 2007, OPODIS 2009, PODC 2021]
  - ✓Detect misbehavior rather than anticipate it [OPODIS'21]
- Reconfigurable systems
  - ✓The set of participants can be (actively) reconfigured without consensus [OPODIS 2019, DISC 2020, DISC 2021]
- Decentralized trust assumptions
  - ✓Double spending under control [PODC 2022]
- Security and privacy in sharing data, reconciling blockchains and more…

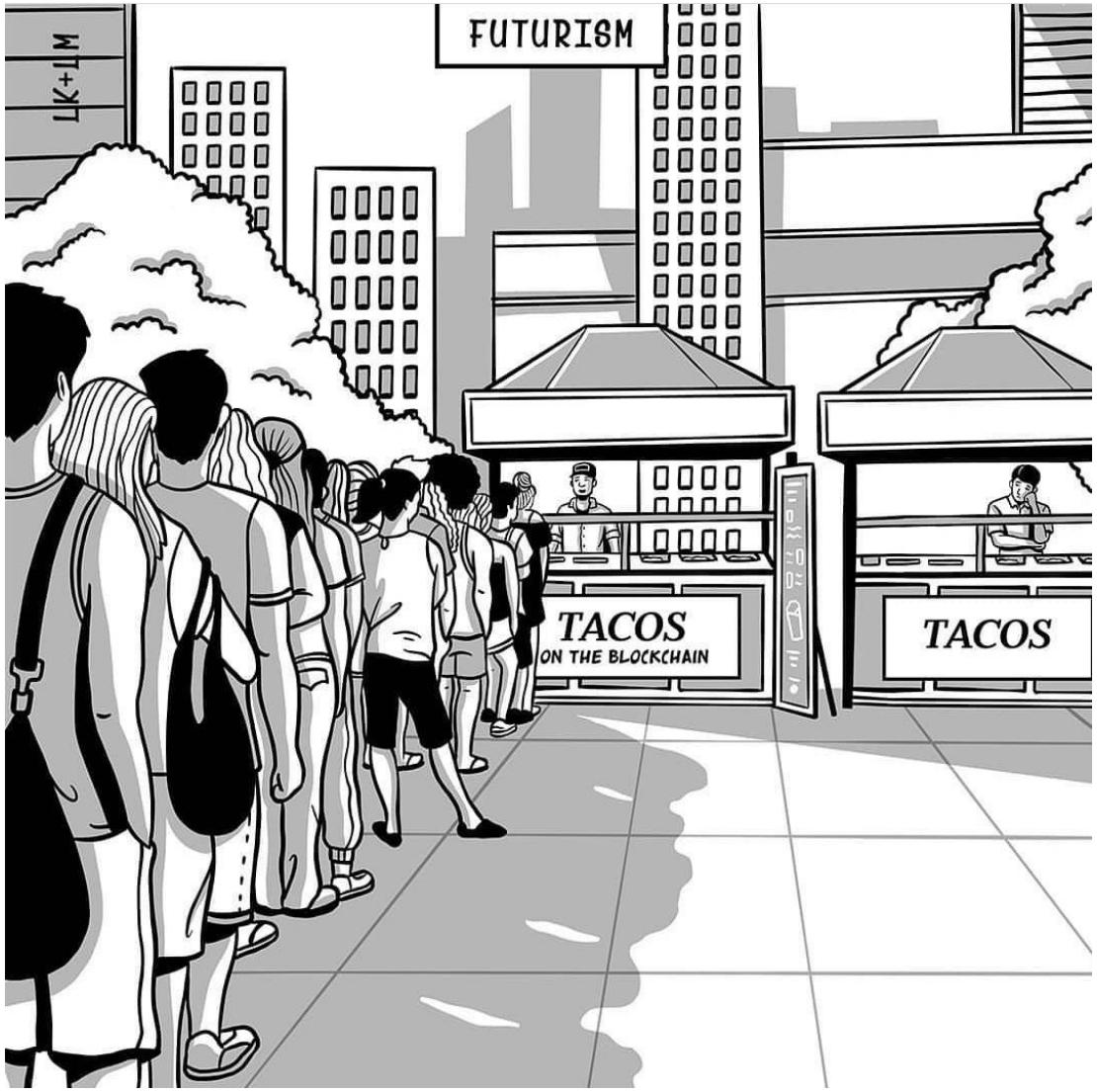
mazars

TELECOM Paris

IP PARIS

Caisse des Dépôts GROUPE

The origin of innovation

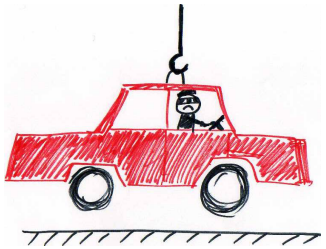

FUTURISM
TACOS
ON THE BLOCKCHAIN
TACOS

Спасибо!

Questions?

How to deal with faults

- Prevention: mask faults
  - ✓Replication and synchronization



- Detection: identify fault originators
  - ✓Accountability

• Complementary: In many systems, we want both!

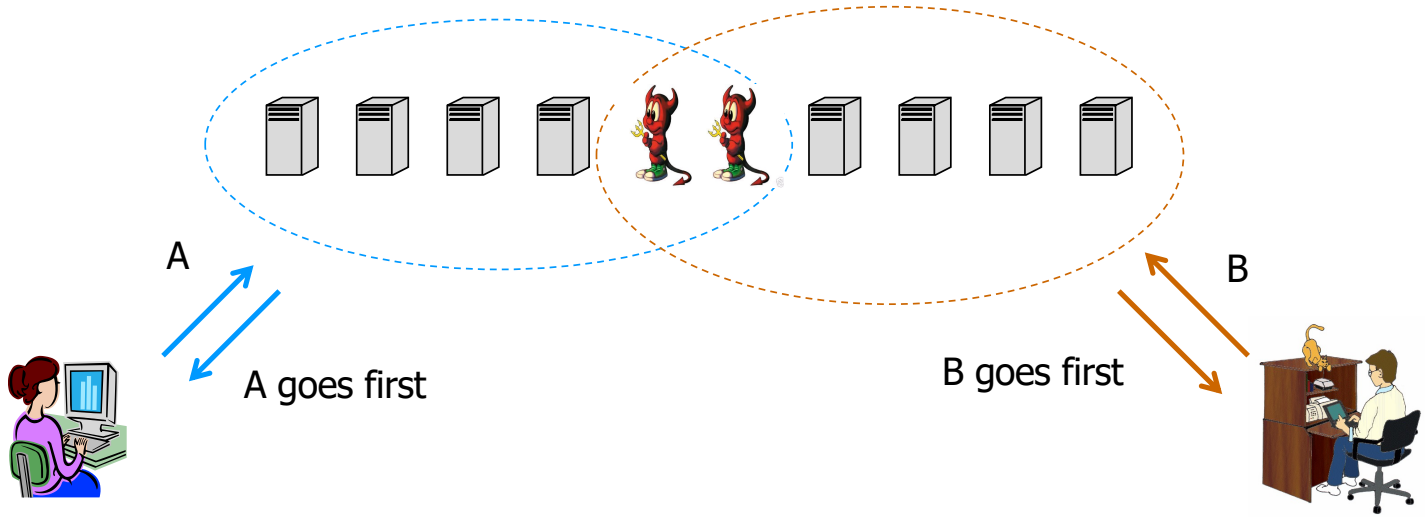
- Application-specific accountability
  - ✓Detect the origin of safety violations
  - ✓Accountability and reconfiguration

Accountability and reconfiguration?

Safety-critical accountability

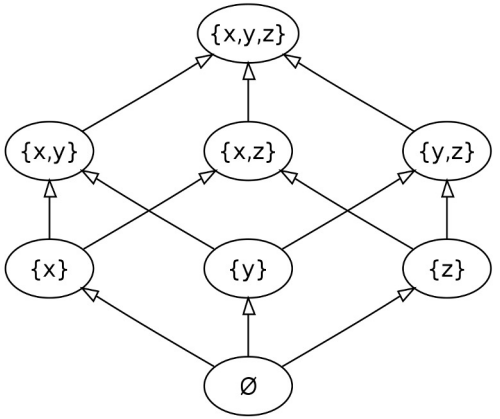
Only detect commission faults that affect safety

- Byzantine fault detection in consensus [Kilhstrom et al., 2003]
- Polygraph: accountable consensus [Civit et al., ICDCS'2021]



Accountability and asynchronous reconfiguration

How to reconfigure?



Consensus-based:

- RAMBO [Gilbert et al., 2010]
- Casper [Buterin-Griffith, 2017]
- Fairledger [Lev-Avirt al., 2019]
- LLB [Ranchal-Pedrosa&Gramoli, 2

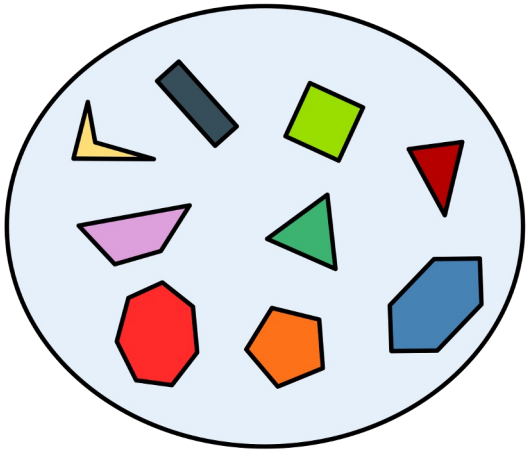
Asynchronous:

- Lattice-agreement instead of consensus [Kuznetsov et al., 2019]

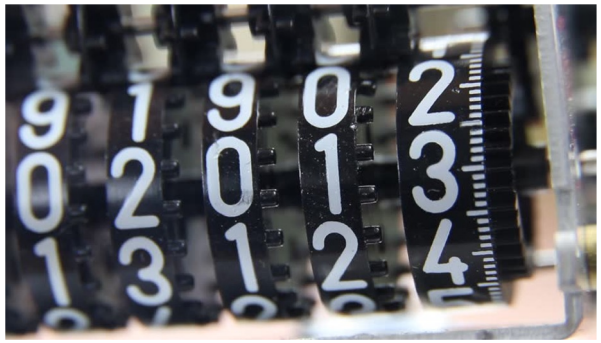
Accountable and reconfigurable lattice agreement [Freitas et al., 2021]

Applications: accountable and reconfigurable

Sets



Counters



Storage Systems

Cryptocurrencies