distributed ledger technologies, blockchain and cryptocurrencies

a (largely incomplete) timeline

- 1999: first popular p2p service (Napster)
- 2008: Bitcoin: A Peer-to-Peer Electronic Cash System (PoW)
- 2010: first real transaction
  - 2 pizzas for 10K BTC
- 2011: “Altcoins” begin to appear
  - Namecoin, Litecoin, etc.
- 2014: UK treasury commissioned a study on cryptocurrencies
- 2015: Ethereum: supporting smart contracts (PoW)
- 2017:
  - BTC quotation about 16K$
  - Russia and Estonia announce plans for government backed cryptocurrency
  - blockchain (DLT) and cyrptocurrencies regarded as **game-changers**
  - Cardano: unpermissionless blockchain based on PoS
- 2019:
  - BTC quotation 7K$
  - DLTs mainly regarded as a decentralized applicative platform
  - many pilot projects with permissioned DLTs, a few real applications
  - Algorand: unpermissionless blockchain based on PoS
- 2022: Ethereum transition to PoS

Bitcoin, blockchain and DLT

- Bitcoin is a cryptocurrency...
- ...based on a technology called **blockchain**
- a number of variations of the blockchain technology are possible and many are used
- they collectively are called **Distributed Ledger Technologies (DLT)**

- for our purposes, DLT=blockchain

state and transactions

- **state**: the state of any data structure at a certain instant of time
  - e.g., the content of a database,
    the content a key-value map
- **transaction**: a change between two consecutive states
  - e.g.,
    for a database, an SQL INSERT
    for a key-value map, an update(k,v) operation

- a state can be represented explicitly or by a sequence of transactions from an initial state

a DLT solves one fundamental problem

- **many subjects** need to **agree on transactions**...
- ...**without trusting each other**

- transactions are recorded on a **ledger**
  - the ledger is essentially the log of the **accepted transactions**
- the ledger is **replicated**
  - each participant has a copy of it

- **consensus** on what is a “good copy” of the ledger is reached in a **distributed** manner
  - **no central authority** to be trusted

potential applications of DLT

- real estate registry
- companies registry
- parcels delivery tracking
- civil registry
- financial transactions
- insurance
- medical records
- trial records
- ...

suited for applications with legal implications

concepts

- **nodes**: the machines that run the DLT
  - they have to be connected, e.g. over the Internet
  - these are the subjects that agree on the accepted transactions
- **(consensus) rules**: rules transactions must comply with
- **transaction lifecycle**: candidate (or pending) transactions are *submitted*, then *validated* for consensus rules, then *accepted* or *rejected*, the accepted ones are appended to the ledger
- **users (or parties)**:
  - read the past accepted transactions
  - create and submit transactions
    - i.e., write on the blockchain
- **block**: a piece of the ledger
  - often storing consecutive accepted transactions
- **blockchain**: a sequences of blocks
  - sometimes it is used as synonym for the ledger, sometimes for the network of the nodes

the cryptocurrency application

- transactions are payments
- the ledger records payments
  - the state is the balance of all the accounts of the users
- a “good copy” conforms to plain accounting rules, e.g....
  - only owners of money can spend it
  - **no double spending** of money
  - controlled **money creation**
  - **no charge back**
  - possibly other conditions to unlock funds

...and many other technical rules

  - e.g. the format of the records

DLT security requirements

- **accepted transactions have to comply to consensus rules**
  - correctness
- **past accepted transactions cannot be “undone”**
  - immutability of the ledger
- all involved nodes **see and agree on the same ledger content** at a certain instant
  - ...in which all transactions conforms to all consensus rules
  - consistency among nodes
- DLTs fulfill these requirements…

without centralized trusted authority

DLT classification

permissioned vs. permissionless DLT

- permissionless DLT
  - anybody can contribute (with a new node) to run the DLT
  - large networks
  - slow
  - e.g., Bitcoin
- permissioned DLT
  - only authorized/trusted nodes can join
  - small networks
  - fast
  - typically belonging to industry/banking consortiums, but may be exposed to the public

private/public DLT

- subjects access the ledger by contacting nodes
- private DLT
  - only authorized subjects can access the ledger (either r/w or read-only)
    - “read” means inspect the ledger
    - “write” means send a transaction
  - nodes perform *access control*
- public DLT
  - any subject can read the ledger and send transactions
    - no access control by nodes

DLT

		Who can operate a node?	
		Permissioned	**Permissionless**
Who can access the ledger?	**Private**	set up by consortia for internal use e.g. Ripple, inter-bank money transfers, parcel tracking	**-** this is possible from a technical point of view but unlikely to occur since no public open community would support a private objective
	Public	set up by consortia or industry association for providing public services e.g. Sovrin for self sovereign digital identity, Diplome for study degree certificates	community driven infrastructure to provide a public service e.g. cryptocurrencies like Bitcoin, Ethereum, etc.

architecture

architectural elements

- **identifiers** of transaction parties, i.e. ***users*** (a.k.a., addresses)
- **the ledger** (content, format, consistency)
  – many technical rules
- **nodes:** computers that run the software realizing the DLT
  – nodes ≠ users, but sometimes a node is associated with a user identifier
- **p2p messaging protocol** to broadcast accepted (blocks) and pending transactions among *nodes* over an overlay network
- **distributed consensus algorithm**
  – a way to reach consensus "securely"

- **smart contracts** and related **executor** to evaluate them

additional elements for permissioned DLT

- a certification authority to…
  - … state which nodes can participate to the DLT
  - … state which users can use the DLT, possibly

additional elements for permissionless DLT

- no checks are performed to join the DLT as a node or users
  - Hence, no CA is needed
- however, nodes contributor need to be incentivized
- **reward** (or incentive): the reason for a subject to spend some resources to help running a public blockchain
- usually, the reward is in the form of **cryptocurrency**
  - which may be used to transact within the blockchain itself or exchanged for fiat money outside the blockchain
  - hence, even if the purpose of permissionless DLT is not to provide a cryptocurrency, all of them provide one (or more) cryptocurrency
- usually, reward is given to whom participate in the creation of a new block

identifiers and their use

- identification of users is done by **private/public key pairs**
  - in permissionless DLT, each user autonomously create private/public key pairs, possibly many of them
  - **having many IDs improves confidentiality**
- users own assets in the DLT
  - e.g., cryptocurrency, smart contracts, etc.
- owner of an asset may be specified in the ledger by a public key
  - or equivalently by its cryptographic hash
- users act on those assets by submitting transactions
  - e.g., to spend cryptocurrency from their account or to ask a smart contract to do something
- consensus rules impose that only certain users can manage certain assets
  - e.g. only the owner of an account can spend the cryptocurrency “contained” in that account
- **a user proves his/her identity by signing the transaction**
- signature is verified by nodes against public key(s) associated with the involved asset

p2p messaging protocol

- nodes discovery
  - what is the first node to connect to? then, recursively ask for other nodes to consider as neighbors
- node interconnection
  - routing in the peer-to-peer overlay network
- broadcasting
  - by gossip protocol
    - each node resends to neighbors all received messages, only one time
- **each new/pending transaction is broadcasted**
  - these are not yet accepted into the ledger
- **a block that contains new accepted transactions is broadcasted**

we do not go further into the p2p messaging protocol

ledger

- addition of transactions to the ledger occur on a block basis
  - a block contains many transaction
- transactions should respect certain "consensus rules" that are application-specific
    - e.g. for money: no double spending
- the order of transactions is fundamental!
  - e.g. for money: can't spend before getting money
  - in other words, consensus rules may look at the history of accepted transactions

- most of the machinery of a DLT is about the addition of a block to the ledger

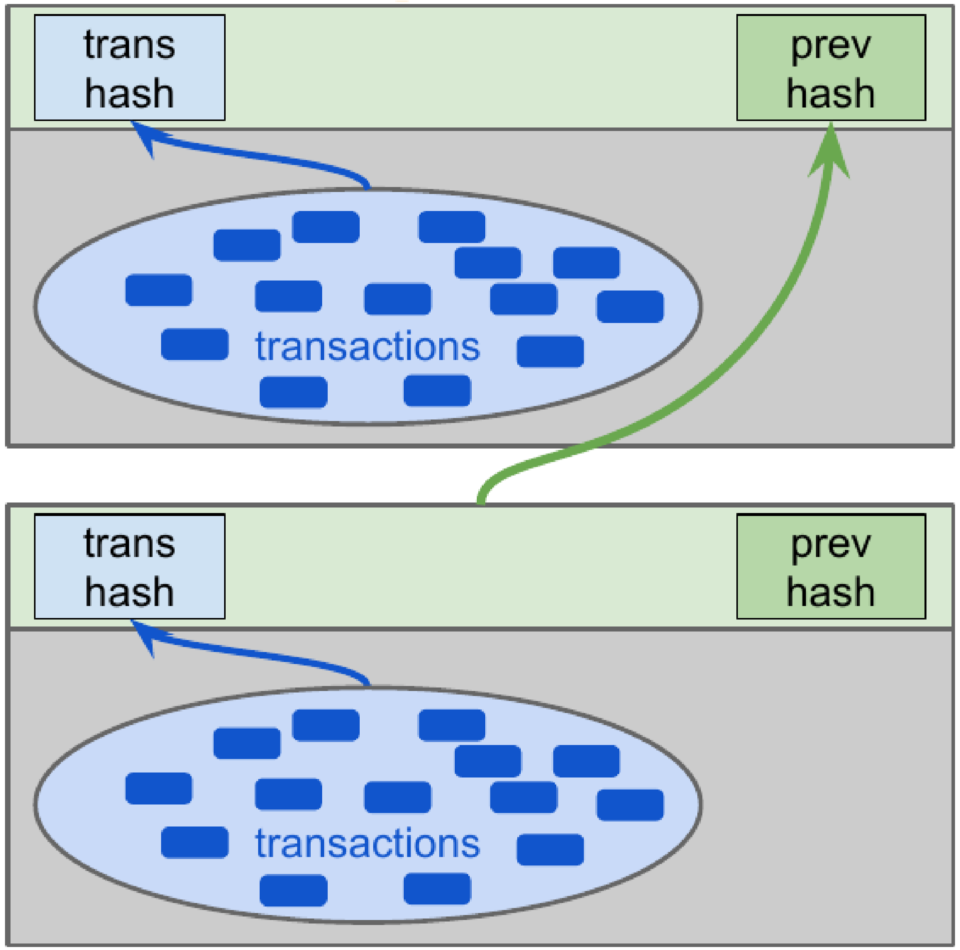
lifecycle of a transaction (1/2)

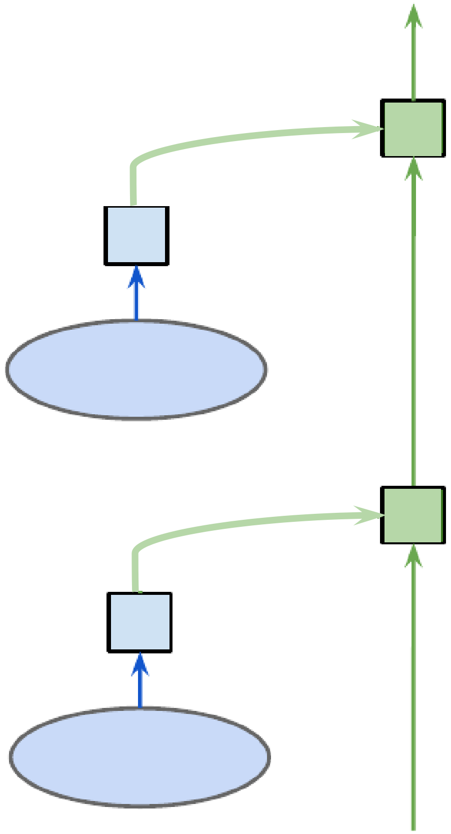
- a user u creates a tx locally
  - in general, the user may need to know all previous transactions to make a correct transaction
    - e.g. to know the amount of cryptocurrency (s)he owns
    - it may ask nodes for the past transactions (e.g., wallet apps do this)
  - u signs tx to prove his/her identity
- u sends tx to any node n
- n broadcasts it to the whole network
- a node m that receives tx checks for its syntactic validity
  - this is not the final check for acceptance, but it is just to discard evidently malformed transactions

lifecycle of a transaction (2/2)

- m puts tx into a **pool** of **pending transactions (or candidate transactions)**
- m tries to put tx in a new block B
  - this depends on the consensus algorithm (see later)
    - e.g. for PoW all nodes concurrently create a candidate block B picking transactions from the pool and trying to solve the *cryptographic puzzle*
    - e.g. for BFT or PoS, leaders create a candidate block B picking transactions from the pool and then propose B to all other *committee members*
- when consensus is reached on B, it is broadcasted to all nodes
- nodes that receive B updates their persistent ledger (and possibly some index) to take into account all transactions in B (including tx)
- now, transaction in B (including tx) are considered accepted
  - …or almost accepted, depending if the consensus “has finality” or not (see later)

block content and chaining





courtesy of G. Di Battista and R. Tamassia (adapted)

current block hash

- each block contains the cryptographic hash of the previous block
- hence, the **hash of the last block** (the current one) **identifies the whole ledger instance**
  - this is a property of so-called *authenticated data structures* (see later)

ledger state for a cryptocurrency

- the ledger state for a cryptocurrency can be represented in two ways
- implicit (Unspent Transaction Output, UTXO)
  - just transactions are recorded, the ledger is a DAG in which nodes are transaction, inputs spend (attach to) transaction output, and output may be spent or unspent.
  - the state is given by all Unspent TX Output).
  - this is the model of Bitcoin.
- explicit (Account balance)
  - an Authenticated Data Structure (ADS) is kept that maps addresses to account balances

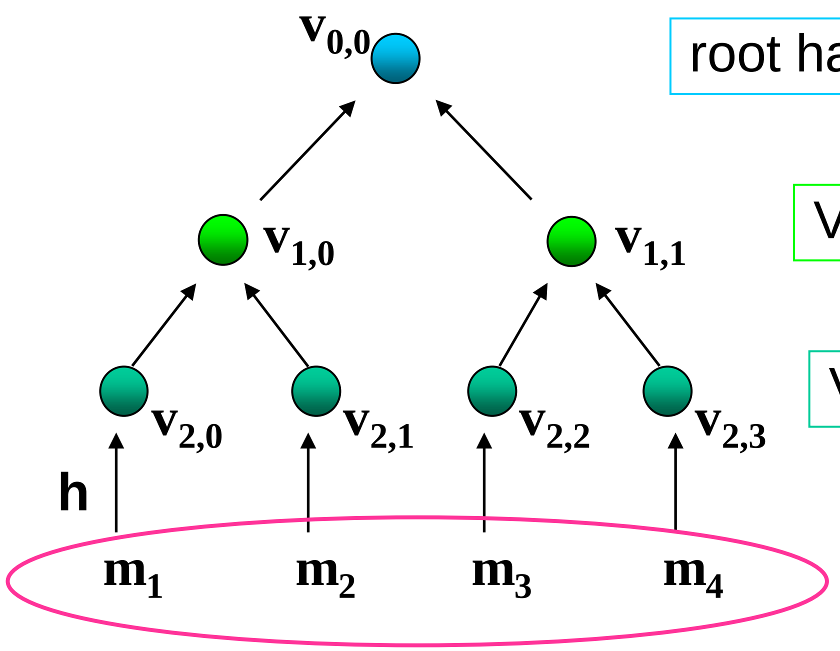
…. Let’s see what an ADS is…

Authenticated Data Structures

- an ADS is a data structure that is “easy” to check for integrity, even for parts of it
- basics
  - it collects elements
  - it associates a cryptographic hash h with its content
  - h is called ***root hash*** or *basis*
    - value of $h \leftrightarrow$ content of the ADS
- integrity verification
  - each query comes with a *proof* that can be checked against h
  - each update can update h without knowing the whole ADS
- the idea is the if you have a trusted copy of the root hash it is easy to verify the integrity of every part of the ADS without recomputing the hash of all data

ADS example: Merkle Hash Tree (MHT)

- a (balanced binary) tree
- each node *v* contains a hash of the data associated with leaves of the subtree rooted at *v*



root hash= $V_{0,0} = h(V_{1,0} \mid V_{1,1})$

$V_{1,1} = h(V_{2,2} \mid V_{2,3})$

$V_{2,2} = h(m_3)$

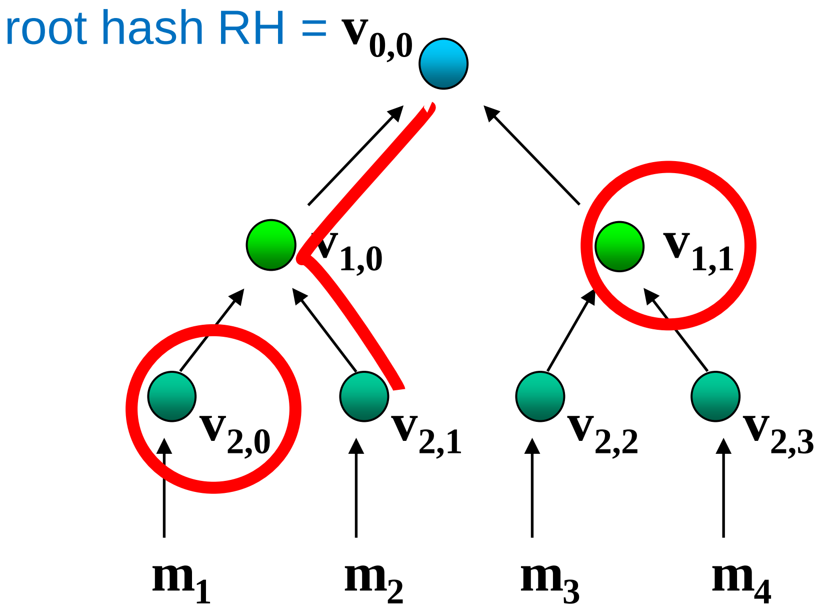
$V_{2,3} = h(m_4)$

data must be ordered

h(.) is a cryptographic hash function

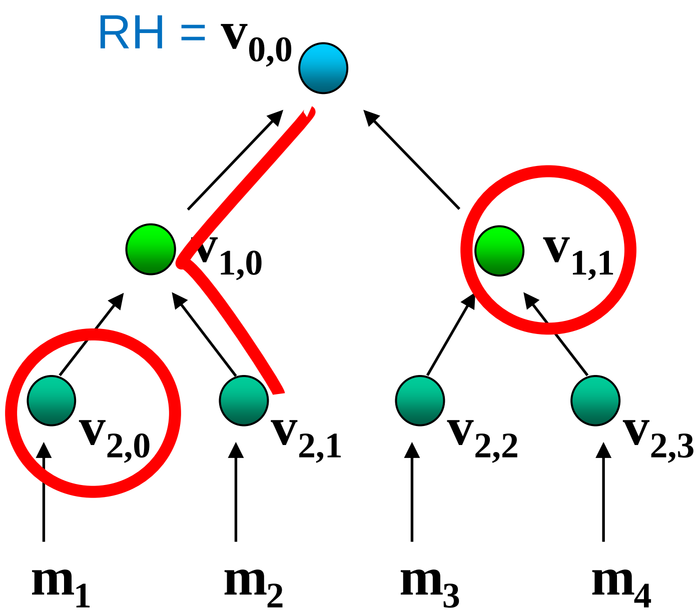
MHT: proof construction

- proof for m_i:
  - consider the path p from m_i to root (excluded)
  - the proof is made of "steps", one for each node v of p
  - each step is a pair
    - label **L**eft or **R**ight depending on how parent of v is entered
    - (hash in the) sibling of v
- example: m_2
  - $p = \mathbf{v_{2,1}}\ \mathbf{v_{1,0}}$
  - proof
    - R $\mathbf{v_{2,0}}$
    - L $\mathbf{v_{1,1}}$



MHT: query verification

- suppose that verifier has a trusted version of the root hash: *tRH*
- procedure for integrity check
  - from proof re-compute RH,
    in the example
    RH = h(h($\mathbf{v}_{2,0}$ |h(m_2)) | $\mathbf{v}_{1,1}$)
  - compare
    RH == tRH



ledger state for a cryptocurrency

- explicit (Account balance)
  - an Authenticated Data Structure is kept that maps addresses to account balances
  - the block header contains also the *root hash* of this ADS
  - execution of a transactions contained in a block B also update the ADS (with respect of the state of the previous block)
    and the new root hash is included in B
  - this is the model of Ethereum
    - actually, Ethereum stores in the ADS also the persistent state of all smart contracts

distributed consensus algorithms (or protocols)

distributed consensus algorithm

- used to accept a new block
  - …and all its transactions and in which order!
- mandate that “all honest nodes” accept the same block
  - hence, they will have the same view of the ledger
- nodes should check for compliance of all transactions of the block to all consensus rules
  - order is important
- **contrast “byzantine” nodes**...
  - ... which might pretend to subvert the rules
  - byzantine: **any possible malicious behavior!**
    - comprising keep silent, lying, colluding with other byzantine nodes, but not impersonating other nodes
  - this is the hard part of the consensus algorithm

consensus attacks: general objectives

- accepting transactions that do no conforms to consensus rules
  - subvert correctness
- changes to old blocks already accepted by at least some nodes
  - subvert integrity
  - might allow chargeback, double spending, and illegitimate change of other parameters of the network
- DoS: denial of acceptance of certain transactions

consensus algorithm: a first attempt

- suppose there is a special node called *the leader*

1. the leader proposes the next block and broadcast it
2. each node broadcasts its vote (yes or no)
3. the block is accepted if the "majority" votes yes

the Sybil attack

from “Sybil: The True Story of a Woman Possessed by 16 Separate Personalities” – F. R. Schreiber - 1973

- suppose an attacker can freely create nodes that participate to the consensus
  – this true for permissionless DLT(!)
  – the attacker can run a “script” that creates many nodes

- the attacker can create more nodes that the honest ones winning each voting
  – …subverting integrity and correctness of the DLT

countermeasures to the Sybil attack

for permission**ed** DLTs:

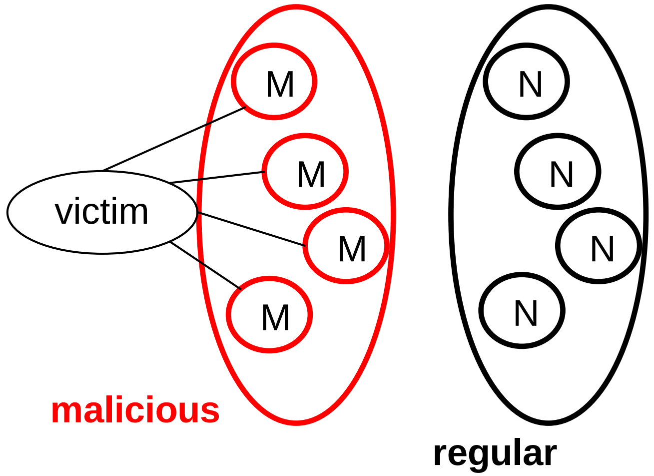
- centrally identify all nodes by a CA

for permission**less** DLTs:

- avoid free creation of new nodes,
  i.e., participating to consensus has some cost:
  - Proof of Work: perform computation
  - Proof of Stake: put some cryptocurrency at stake
- in general
  - Proof of <something> : spend some <something> to participate
    - it might be also some physical work
      e.g., Helium proof-of-coverage for LoRa hotspots

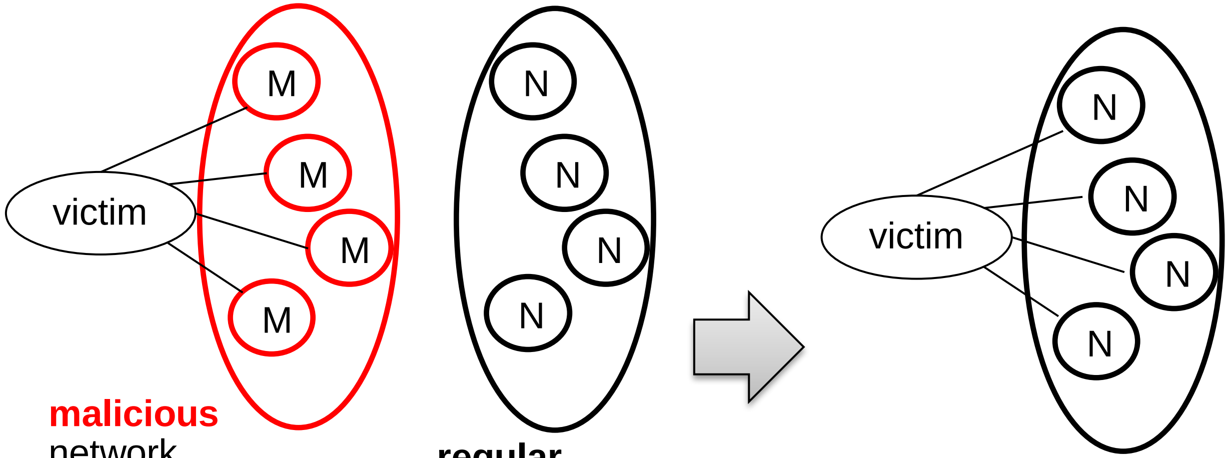
the eclipse attack

- this is actually a (typical) vulnerability of the p2p messaging protocol
- it makes sense in permissionless DLT
- the attacker controls a large number of malicious nodes (not necessarily the majority) and can isolate a “victim” node
  - this is done by proposing malicious nodes as neighbors of the victim much more frequently than honest nodes
  - the attacks make malicious nodes repeatedly asking the victim for being neighbor



the eclipse attack

- during the eclipse…
  the malicious nodes show to the victim a malicious DLT state, and the victim receives a “malicious payment”
- the malicious payment disappears when the attack ends, and legitimate chain is broadcasted
  – the net effect is a chargeback or double spending



malicious network where payment is recorded

the victim assume payment was correctly finalized

regular network, that is screened by the malicious one, where **no** payment is recorded

when malicious nodes go away, the victim connects to the regular network and see no payment

some considerations on the eclipse attack

- this attack is independent from consensus algorithm
  - e.g., it works with PoS
  - it does not work in permissioned DLTs where nodes all know each other

- p2p messaging protocols should be equipped with countermeasures

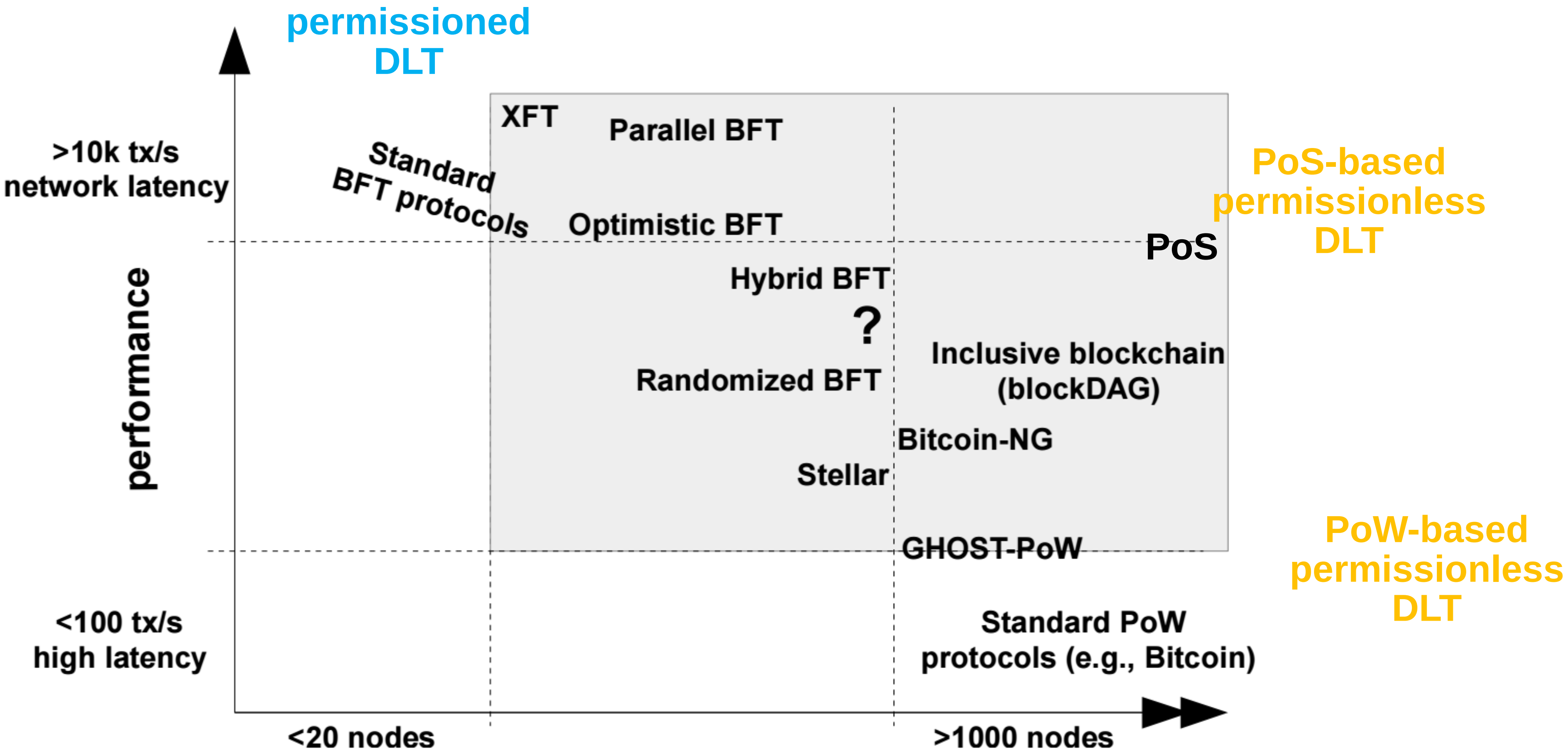
- for PoW, it can be detected by observing an anomalously low “hash power”

distributed consensus algorithms

many solutions, a few are very famous:

- Proof-of-Work – for permissionless DLTs
  - slow but it scales to a high number of nodes
  - Bitcoin and many other permissionless DLTs are based on this
- Byzantine-Fault-Tolerant – for permissioned DLTs
  - fast but feasible only for a small number of nodes
- Proof-of-Stake – for permissionless DLTs
  - fast, scales, but affected by some security concern
  - it relies on BFT-like approach, selecting a subset of nodes

distributed consensus algorithms overview



source: M. Vukolić. The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. iNetSec 2015 (adapted)

Proof of Work

Proof of Work (PoW)

- adding a block requires to solve a cryptographic puzzle
  - that can be only solved by enumeration approach (i.e., brute force)
- **in PoW consensus is implicit**
  - **a node that works for the next block is accepting all previous ones**
  - for this reason, it scales, since no communication among nodes is needed!
- the puzzle is something like the following

  given the previous block P,

  find the next block B(x), where x is a field of B that can be freely changed, such that

  hash(header of B) **< threshold**

  that is

  hash(hash(transactions of B), hash(P), x) **< threshold**

- changing threshold changes the difficulty
  - e.g, in Bitcoin, threshold is periodically adjusted to have an expected block time of 10 minutes, by a feedback control loop that is part of the consensus rules

PoW and the Sybil attack

- PoW is not based on voting
- controlling many nodes does not bring any advantage
- just computing power is important
  - but computing power cannot be increased by a script!

forks may occur

- two nodes may solve the next block at roughly “same time”
  – with two distinct solutions
- the two blocks are broadcasted (fork)
  – actually, some nodes see only one of them (non instantaneous broadcast), others see both and choose one (fork resolution)
- the two chains might grow independently for a while

fork resolution: the longest chain rule

- a node that sees more chains **chooses the longest one**
  - transactions that are in a discarded block are put in the pending transaction pool again
  - they might not be accepted any more
    - … and definitely discarded after a timeout
    - depends on the consensus rules and previous transactions
    - possible double spending!
- which chain grows faster is random
- the longest chain has more work done on it
  - in terms of computation performed
  - hence it is more appealing for a node working on that

transaction confirmation a.k.a. *finality*

- **confirmed: stored in an immutable block, forever**
- PoW does not provide “mathematical guarantee” of confirmation (i.e. it has *no finality*)

- a transaction is considered confirmed if it is enough deep in the blockchain
- “enough” depends on the criticality of the transaction!
- usual confirmation depths are 1 to 6

consensus attacks and confirmation depth

- changing of a deep block *b*...
- ...requires the attacker to solve again all blocks above *b*
- the attacker needs a huge amount of computing power to reach and surpass the legitimate chain

- the more *b* is deep the more is “confirmed”

consensus attack: 51%

- this is an attack to PoW
- who controls more than 50% of the computational power can...
  - …disconfirm recently confirmed blocks
    - by surpassing with its chain all other forks
  - …solve 100% of the blocks, get 100% of the rewords
    - by keeping adding blocks and reverting those that by chances are solved by other nodes
- it can also impact certain consensus rules
  - certain features, i.e. changes to consensus rules, may be triggered when something occurs
  - crafting specific blocks an attacker can convince other nodes to activate them

BFT and PoS

consensus “Byzantine Fault Tolerant” (BFT)

- used by
  - permissioned DLT
  - as part of PoS for permissionless DLT
- $O(N^2)$ messages, where N is the number of nodes
  - largely sent in parallel
- 3 stages to reach consensus
  - quite fast
- $N=3f+1$, where f is the number of byzantine-faulty nodes
  - i.e., it tolerates <N/3 byzantine-faulty nodes
- majority is $>2N/3$

- vote is given by cryptographic signature
- hence, finality is provided (a block either has $>2N/3$ signatures or not)

BFT and Sybil attacks

- no countermeasure for the Sybil attack
- regular BFT adopter just know who are the other nodes
  - admission in the club is strictly regulated
  - i.e., it is a permissioned DLT

- further, BFT does not scale to large number of nodes, it cannot be used in permissionless DLT

Proof of Stake (1/2)

- used by permissionless DLT
- nodes “put at stake” some amount of cryptocurrency
  - i.e. the amount is blocked to have the right to participate in the consensus and to obtain some reward
- these nodes are called *block producers* or *validators*
- the consensus is performed by a limited number of nodes belonging to a committee of size C<N
  - among the validators
- committee performs BFT-like consensus
  - finality provided by cryptographic signature for votes
  - C is constant even if N increases
  - C is low enough so that BFT can be used

Proof of Stake (2/2)

committee choice can be done in several ways

- by election, i.e., by voting or delegating stake to a node
  - the more you stake the more your votes count
  - e.g. EOSIO do this, usually called *delegated* PoS
- by cyclically changing in a round-robin fashion with some sort of randomicity
  - stake amount is fixed for each node involved in PoS
  - if you want to stake more you get more nodes involved in PoS
  - randomly select committee
  - e.g., Ethereum 2 do this
- by randomly extracting nodes of the committee with probability proportional to the staked amount
  - e.g., Algorand and Cardano do this

Proof of Stake and Sybil attacks

- participation to consensus il linked to money
- …and money cannot be created by a script!

- note:
  suppose an adversary controls a fraction f of the whole stake,
  the expected fraction of stake controlled in the committee is f

- who own large amount of cryptocurrency has big power
  - e.g. many blockchains are run by foundations that own large part of the related cryptocurrency (pre-minting)

the “nothing at stake” problem

- a validator may produce a fork sending two different “competing” blocks
- unless there are countermeasures in place, nodes grow both chains, since
  - it costs nothing more
  - they are rewarded independently from which chain finally wins

this makes double spending easy

PoS approaches should ensure either no-fork or fast fork resolution.

BFT vs. Pow vs. PoS

	PoW	BFT	PoS
messages	none	$O(N^2)$	$O(C^2)$ C is the size of the committee
latency	random, depends on the threshold, >20 seconds	depends on the network latency (assuming no network bottleneck)	depends on the network latency (assuming no network bottleneck)
throughput	sequentially processing consecutive blocks → throughput $O(1/latency)$ assuming constant block size		
majority	$>1/2$	$>2N/3$	$>2C/3$
finality	no	yes	yes
who can contribute to consensus	whoever the greater the computing power the higher the probability to make a block (and earn rewords)	certified by a central Certification Authority	who complies to two requirements: (1) have put at stake some amount of cryptocurrency (2) is in the current committee

the blockchain (scalability) trilemma

first stated by V. Buterin (Ethereum founder)

- say n the number of nodes
- desirable properties
  decentralization: nodes have limited $O(1)$ resources (bandwidth, cpu, storage)
  scalability: transaction throughput $O(n)$
  security: attacker can spend $O(n)$ for the attack
- the trilemma:

you cannot fulfill all the three completely

- any DLT is a compromise, e.g.:
  - many current permissionless DLT: limited scalability
  - EOSIO, Solana, some permissioned DLT: limited decentralization
  - sharded blockchains: limited security
- it is not a theorem
  - research is ongoing for the perfect solution!