

Blockchain –консенсусни механизми

проф. д-р инж. Венета Алексиева

ОСНОВНИ МОМЕНТИ

- Дефиниции
- Консенсусни механизми
 - доказателство за работа (Proof Of Work)
 - доказателство за залог (Proof Of Stake)
 - делегирано доказателство за залог (Delegated Proof Of Stake)
 - доказателство за авторитет (Proof Of Authority)
 - доказателство за история (Proof Of History) на Solana
 - Practical Byzantine Fault Tolerance за Hyperledger Fabric
 - доказателство за капацитет (Proof Of Capacity)
 - доказателство за тежест (Proof Of Weight)
- Формиране на мрежа

Консенсусен механизъм

- Консенсусният механизъм е метод, чрез който участниците в мрежата колективно се споразумяват за текущото състояние на блокчейн мрежата.
- Този процес включва четири основни действия
 - Избор на главен/валидатор
 - Генериране на блокове
 - Проверка на данни
 - Качване (upload)

Консенсусните механизми се различават по:

- Консумация на енергия
- Брой consensus nodes
- Бързина на транзакциите
- Време за създаване на блок
- Наличие/липса на математически изчисления
- Наличие/липса на гласуване между възлите
- Степен на централизация
- Степен на цензура (блокиране на транзакции от трети страни)
- Устойчивост на атаки

Консенсусни механизми

- доказателство за работа (Proof Of Work)
- доказателство за залог (Proof Of Stake)
- делегирано доказателство за залог (Delegated Proof Of Stake)
- доказателство за авторитет (Proof Of Authority)
- доказателство за история (Proof Of History) на Solana
- Practical Byzantine Fault Tolerance за Hyperledger Fabric
- доказателство за капацитет (Proof Of Capacity)
- доказателство за тежест (Proof Of Weight)

Proof of Work (PoW)

- Исторически първи се появява в биткойн мрежата
- Хешът на блока се изчислява чрез математическо предизвикателство - намиране на „nonce“ чрез bruteforce метод
 - Миньорите се състезават в решаването на сложен математически пъзел, като намират хеш, който отговаря на специфични условия (намиране на nonce – число с определени водещи нули в двоична форма), зададени от мрежата.
 - Миньорът, който успешно реши пъзела, получава правото да добави новия блок към блокчейна.
- Изисква значителни изчислителни ресурси и енергия
- Доверието зависи от инвестицията в хардуер и от „консумираните“ изчислителни ресурси
- Използва се в: Bitcoin, Bitcoin Cash , Bitcoin SV, Litecoin, Dogecoin, Monero, Kaspа
- Ethereum стартира с PoW, но сега е с PoS

Proof of Stake (PoS)

- Валидаторът се избира на случаен принцип според инвестираната валута
 - Валидаторите на мрежата се избират въз основа на количеството криптовалута, което могат да зложат като обезпечение.
 - Колкото повече залага даден възел, толкова по-голяма е вероятността да бъде избран за валидиране и добавяне на нови блокове към блокчейна.
- Избягва се тежко изчисление
- Включва по-малко спекулации
- Използва се в: Ethereum, Solana , Cardano и негови разновидности в други блокчейни

Delegated Proof of Stake (DPoS)

- Работи чрез система за гласуване, при която участниците избират делегат, отговорен за добавянето на нови блокове към блокчейна от тяхно име.
- Този механизъм повишава ефективността, като намалява броя на участниците, пряко ангажирани във валидирането на блокове.
- Счита се за по-централизиран модел, тъй като шепа делегати (21 или 27) контролират цялата мрежа.
- Използва се в: Tron, EOS, Steem, Hive, BitShares, Lisk, Ark

Practical Byzantine Fault Tolerance (PBFT)

- Той е с висока отказоустойчивост, тъй като остава работещ дори при наличие на до 33,3% злонамерени възли.
- Всеки възел в мрежата представя предложение на други възли, които след това гласуват въз основа на предварително дефинирани правила.
- Окончателно решение се взема след като е подаден определен брой гласове
- Използва се в: Hyperledger Fabric, Hyperledger Besu / Enterprise Ethereum, R3 Corda, Ziliqa

Delegated Byzantine Fault Tolerance

- Комбинираща сигурността на класическия PBFT със скалируемостта на Proof-of-Stake (PoS) чрез механизъм за делегиране (гласуване).
- Всеки притежател на токени може да гласува за Consensus Nodes, но притежателите на токени не участват директно в одобряването на блоковете.
- Consensus Nodes отговарят за проверката на транзакциите и записването им в блокчейна. Броят им е между 7 и 21, затова комуникират изключително бързо по модела на PBFT.
- Измежду делегираните възли на случаен принцип се избира един Лидер (Speaker), а останалите са Делегати (Delegates).
- Лидерът събира транзакциите в нов блок и го изпраща на Делегатите.
- Делегатите проверяват блока и гласуват.
- Ако поне $\frac{2}{3}$ (две трети) от делегатите се съгласят, че блокът е валиден, той се записва в блокчейна завинаги.
- Използва се в: NEO

Proof of Authority

- Той заменя финансовия залог (както е при Proof of Stake) или изчислителната мощност (при Proof of Work) с **репутация и идентичност**.
- Транзакциите се обработват **почти мигновено**, няма сложни математически изчисления, няма гласуване между стотици възли.
- Минимален разход на енергия
- Валидатори -валидират транзакции и да създават нови блокове. Одобрените възли се редуват по предварително зададен алгоритъм, за да подписват блоковете, което прави процеса изключително предвидим.
- Валидаторите са сертифицирани - трябва официално да разкрият кои са (чрез нотариални документи или държавни регистри).
- Марка за репутация - Ако валидаторът се опита да измами мрежата или да действа злонамерено, неговата реална идентичност бива публично компрометирана, което води до правни и професионални последиствия
- Недостатък – пълна централизация и цензура (могат да бъдат принудени от регулатори да филтрират или блокират конкретни транзакции).
- Използва се в: VeChain, BNB Chain (тук е в комбинация с PoS)

Proof Of History

- PoH използва *Verifiable Delay Function (VDF)*, за да създаде времеви отпечатък (timestamp) и да подреди транзакциите по хронологичен ред *преди* те да стигнат до същинския консенсус.
- Той винаги се комбинира с друг механизъм (обикновено Proof of Stake)
- Това технически **не е самостоятелен консенсусен алгоритъм**, а по-скоро „децентрализиран логически часовник“.
- Изисква изключително бърза последователна обработка на данни и огромна интернет честотна лента за синхронизация.
- Валидаторите са по-мощни и скъпи сървъри.
- Използва се в Solana

Proof Of Capacity

- Вместо изчислителна мощност, тук се използва **свободното пространство на твърдия диск (HDD/SSD)**.
- Преди да започне процесът, дискът се "начертава" (plotting) с предварително изчислени решения на криптографски пъзели.
- Когато се генерира блок, миньорът просто претърсва диска си за най-бързото съвпадение.
- Висока децентрализация и сигурност
- Използва се в: Signum (бивш Burstcoin), Chia Network (използва модификация *Proof of Space and Time*)

Proof Of Weight

- Вариация на Proof of Stake, при която правото на глас не зависи само от броя притежавани монети, а от **друг измерим показател ("тегло")**, свързан с приноса към мрежата.
- Това премахва монопола на най-богатите участници, т.к. разпределя тежестта на гласа според реална полезна работа.
- Доказва се чрез **специфичен мрежов ресурс** (пространство за данни, реално съхранени файлове, репутация).
- Например в мрежи за съхранение на данни, "теглото" е количеството реална информация, която се пази във времето.
- Използва се в: **Filecoin** (*Proof of Spacetime* като тегло)

Сравнение

Критерий	PoW	PoS	PBFT/dBFT	PoA	PoC	PoH	PoWeight
децентрализация	висока	средна	Ниска (делигиране)	Много ниска	висока	средна	висока
Скорост на транзакциите	Много ниска	средна	Много висока	Екстремно висока	ниска	Екстремно висока	Средно висока
Консумация на енергия	Много висока	ниска	ниска	Много ниска	Много ниска	Екстремно ниска	Много ниска
Финалност на записване на блок	вероятно стна	бърза	мигновена	мигновена	вероятностна	Почти мигновена	незабавна
Тип мрежа	публична	Публична хибридна	Частна консорциум	Частна корпоративна	публична	публична	Публична специализирана

Изводи

- PoW – е първият консенсусен механизъм и на негова база се появяват следващите
- Всяка блокчейн има своя модификация на консенсусен механизъм
- Консенсусните механизми се различават по много критерии, за да постигнат оптимизация по скорост, сигурност или децентрализация

Избор на блокчейн мрежа

- Публични (Разрешени за всички / Permissionless):
 - Протоколи като PoW, PoS, PoC и PoH са създадени за глобални мрежи, в които всеки може да се включи анонимно (като миньор, валидатор или потребител), без да иска разрешение от централен орган.
 - Доверието се гарантира изцяло от математика, икономически стимули или разход на ресурси.
- Частни и Консорциумни (С право на достъп / Permissioned):
 - Механизми като PBFT и PoA се прилагат там, където участниците са известни предварително (банки, държавни институции, партньори във верига на доставки).
 - Тук децентрализацията се жертва съзнателно в името на пълна конфиденциалност, правна отговорност и максимална скорост.
- Специализирани (Целеви мрежи):
 - PoWeight е перфектен за публични децентрализирани мрежи, но с конкретно функционално предназначение (например мрежи за споделено дигитално архивиране или облачни изчисления), където консенсусът е обвързан с реално свършената специализирана работа.

Ще бъдат разгледани:

- Proof-of-Work
 - Имплементиран в първата публична мрежа на Bitcoin и Ethereum (до 2022г.)
- Practical Byzantine Fault Tolerance
 - Имплементиран в частната мрежа на HyperLedger Fabric

Защото първите, които позволяват създаване на умни договори и излизат извън рамките на криптовалутите са:

- Ethereum от 2015г. (и китайската му модификация NEO – 2016г.)
- HyperLedger Fabric от 2017г

Bitcoin

- От 2009г. Bitcoin има (и все още има) вграден език за скриптове - **Bitcoin Script**.
- Той умишлено е направен „Тюринг-непълнен“ от Сатоши Накамото от съображения за сигурност.
- Той позволява само базови интелигентни функции (като мулти-подписи или времеви заключения на транзакции – HTLCs), но не и сложни, автоматизирани и взаимосвързани приложения.
- Биткойн използва счетоводния модел **UTXO (Unspent Transaction Output)** – система от "неизразходени изходи" от транзакции, подобно на хартиени банкноти. Той е за трансфер на стойност, но е „безпаметен“ (stateless).

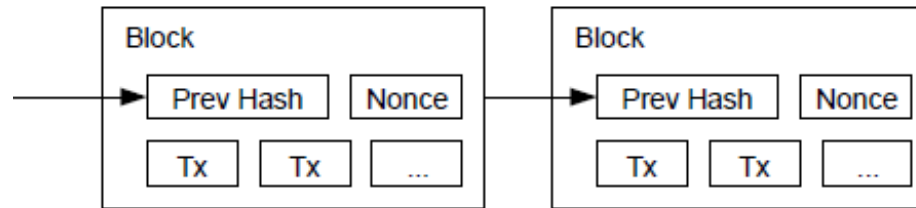
Bitcoin

- Първоначалният дизайн на Биткойн ограничава размера на блока до **1 MB**, а блоковете се генерират на всеки **10 минути**. Сложните умни договори изискват съхранение на много данни и бързо изпълнение.
- Биткойн няма вградена виртуална машина, която да изпълнява сложен код изолирано на всеки компютър в мрежата. Етериум създаде **EVM (Ethereum Virtual Machine)** – софтуерен слой, който действа като глобален компютър за обработка на алгоритми.

Bitcoin

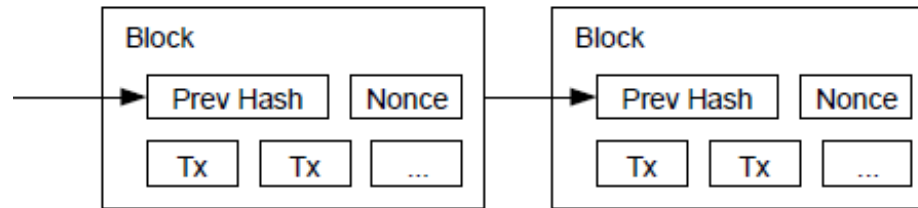
- Биткойн вече поддържа интелигентни функции:
- Taproot (2021 г.) -Подобрява скриптовия език и въвежда по-сложни условия за транзакции с повишена поверителност.
- Втори слоеве (Layer 2) като Stacks и Rootstock (RSK) – предлагат странични вериги (sidechains), които носят пълноценни умни договори, защитени от сигурността на основната Биткойн мрежа.

Proof-of-Work – същност (1/3)



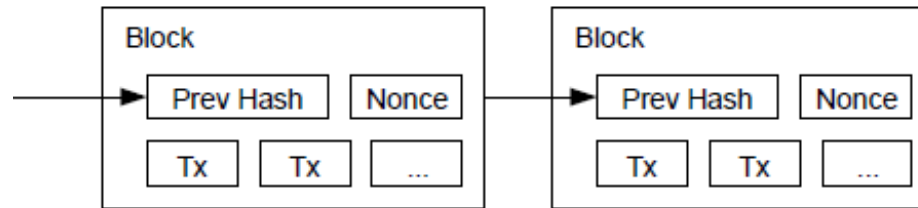
- proof-of-work изисква търсене на число, което при хеширане, с SHA-256 да формира хеш, който започва с определен брой нули.
- Средната необходима работа експоненциално нараства с нарастване на броя на необходимите нули и може да бъде проверена чрез изпълнение на едно хеширане.
- След като CPU усилията са изразходвани, за да се удовлетвори доказателството за работа, блокът не може да бъде променен, без да се извърши повторно изпълнение на работата.
- Тъй като по-късни блокове се свързват след него, работата по промяната на блока би включвала повторно изпълнение на всички блокове след него.

Proof-of-Work – същност (2/3)



- Proof-of-Work също решава проблема с определянето на представителството при вземането на решения от мнозинството. Доказателството за работа е по същество един процесор - един глас.
- Решението на мнозинството е представено от най-дългата верига, в която са вложени най-големи усилия за доказателство за работа.
- Ако по-голямата част от мощността на процесора се контролира от честни възли, честната верига ще расте най-бързо и ще изпревари всички конкурентни вериги.

Proof-of-Work – същност (3/3)



- За да модифицира минал блок, атакуващият ще трябва да повтори доказателството за работа на блока и всички блокове след него, а след това да настигне и надмине работата на честните възли.
- Вероятността по-бавен атакуващ да настигне намалява експоненциално с добавянето на следващите блокове.
- Трудността на доказателството за работа се определя от пълзяща средна, насочена към среден брой блокове на час. Ако се генерират твърде бързо, трудността се увеличава.

Proof-of-Work - Стартиране на мрежата

- Новите транзакции се излъчват до всички възли.
- Всеки възел събира нови транзакции в блок.
- Всеки възел работи по намирането на трудно доказателство за работа (proof-of-work) за своя блок.
- Когато възел намери доказателство за работа, той излъчва блока до всички възли.
- Възлите приемат блока само ако всички транзакции в него са валидни и не са вече похарчени.
- Възлите изразяват приемането си на блока, като работят по създаването на следващия блок във веригата, използвайки хеша на приетия блок като предишния хеш.

Bitcoin P2P мрежа

- Ad-hoc protocol (по TCP port 8333)
- Ad-hoc мрежа със случайна топология
- Всички устройства (nodes) са равноправни
- Ново устройство може да се добави по всяко време
- Забравят се неотговарящите устройства след 3 часа

Bitcoin

Функция	порт	протокол	достъпност
Mainnet (реална мрежа)	Синхронизация 8333 RPC 8332	TCP	P2P е публичен; RPC е изцяло локален
Testnet (тестова мрежа)	Синхронизация 18333 RPC 18332	TCP	за тестова среда
Signet	Синхронизация 38333 RPC 38332	TCP	За контролирана тестова среда
Regtest	Синхронизация 18444 RPC 18443	TCP	Изцяло локална (затворена) среда

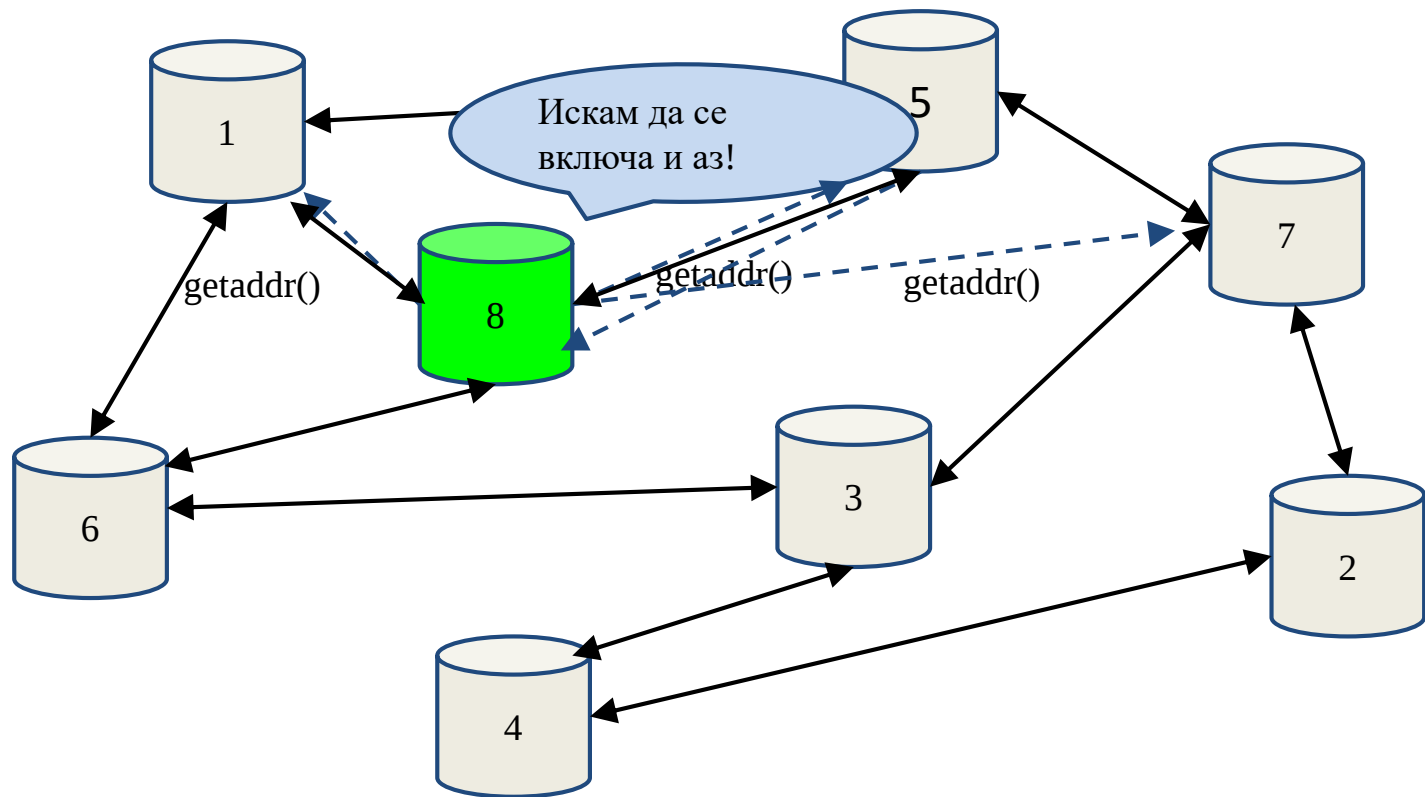
Ethereum

Функция	порт	протокол	достъпност
Синхронизация на мрежата	30303	TCP/ UDP	публичен
Разработчици JSON-RPC	8545	TCP	localhost
webSockets	8546	TCP	localhost
Валидатори	13000 12000 5052	TCP/ UDP TCP	публичен

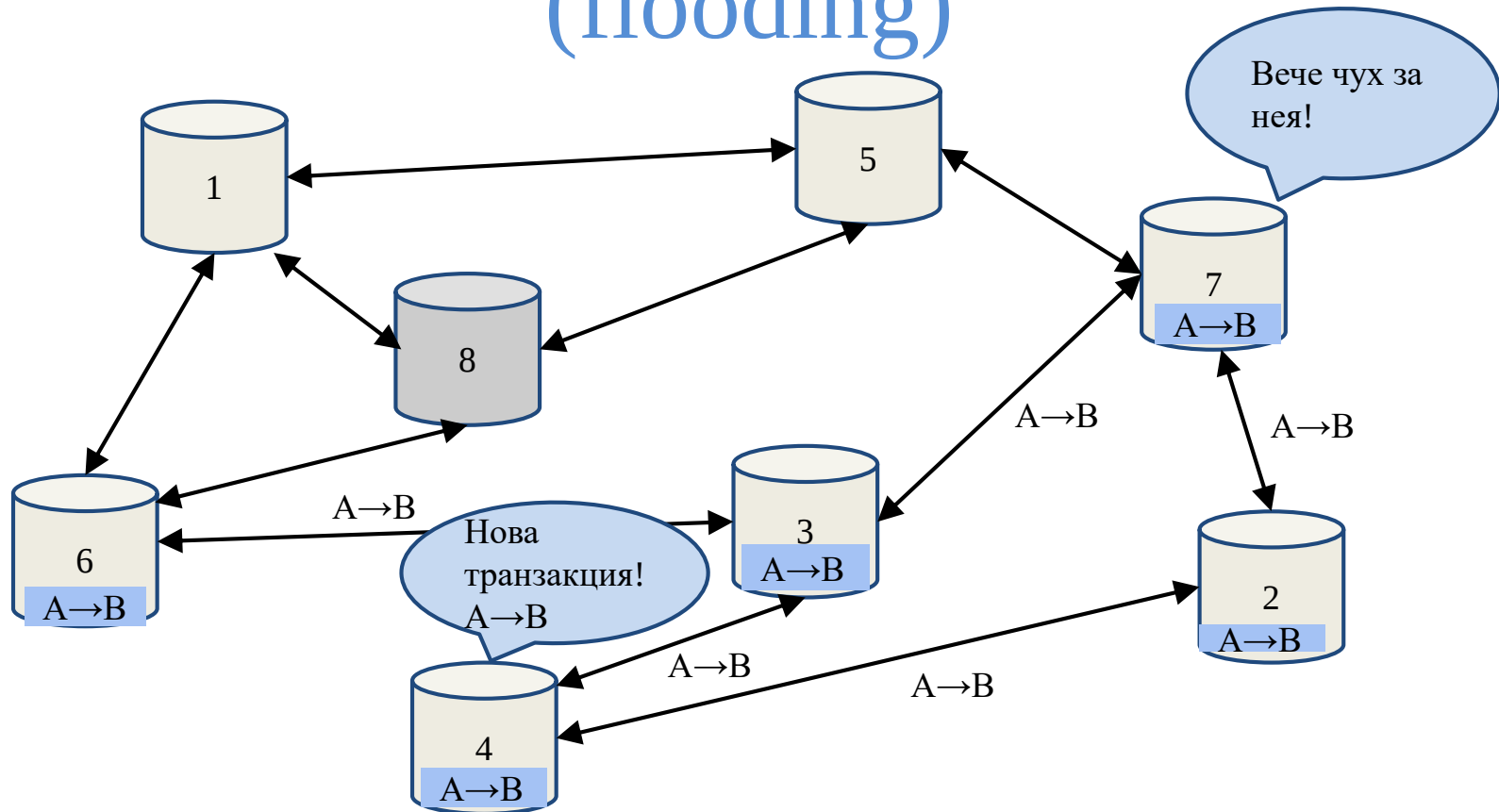
HyperLedger Fabric

Функция	Предназначение	порт	протокол	достъпност
Peer Възел Основен	Приема транзакции от приложения, разпространява блокове и изпълнява <i>Chaincode</i>	7051	TCP (gRPC)	Вътрешна за мрежата / През Gateway
Peer Chaincode	Използва се в по-стари версии или специфични конфигурации за комуникация между самия Peer и Docker контейнера на умния договор.	7052	TCP (gRPC)	Изцяло вътрешна (Local/Docker)
Orderer Възел	Събира транзакциите, подрежда ги в блокове и ги разпределя към Peer възлите за запис.	7050	TCP (gRPC)	Ограничена (Само за Peer и Администратори)
Orderer Raft Consensus	Използва се за вътрешна комуникация между самите Orderer възли за постигане на консенсус чрез Raft/SmartBFT.	7050/ 7053	TCP (gRPC)	Изцяло вътрешна между Orderer възлите
Certificate Authority	Сървърът за дигитални сертификати. Използва се за регистрация и идентификация на потребители и възли (MSP).	7054	TCP (HTTP)	Достъпен за администратори и SDK
Operations Service	Метрики за производителност и мониторинг (използва се от инструменти като <i>Prometheus</i>).	9443	TCP (HTTP)	Локална/ Административна

Присъединяване на нов node



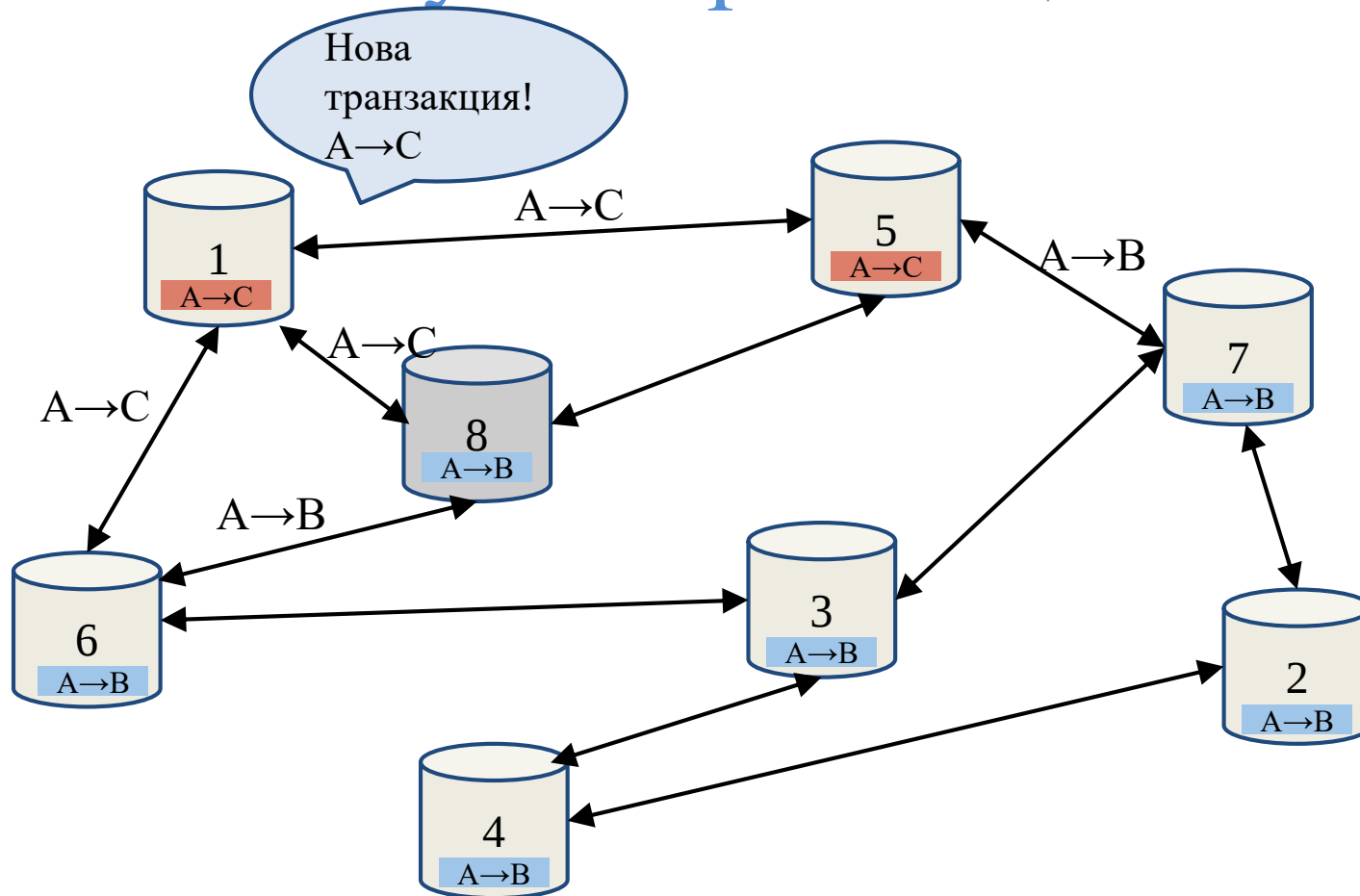
Разпространение на транзакции (flooding)



Кога се препредава предложена транзакция?

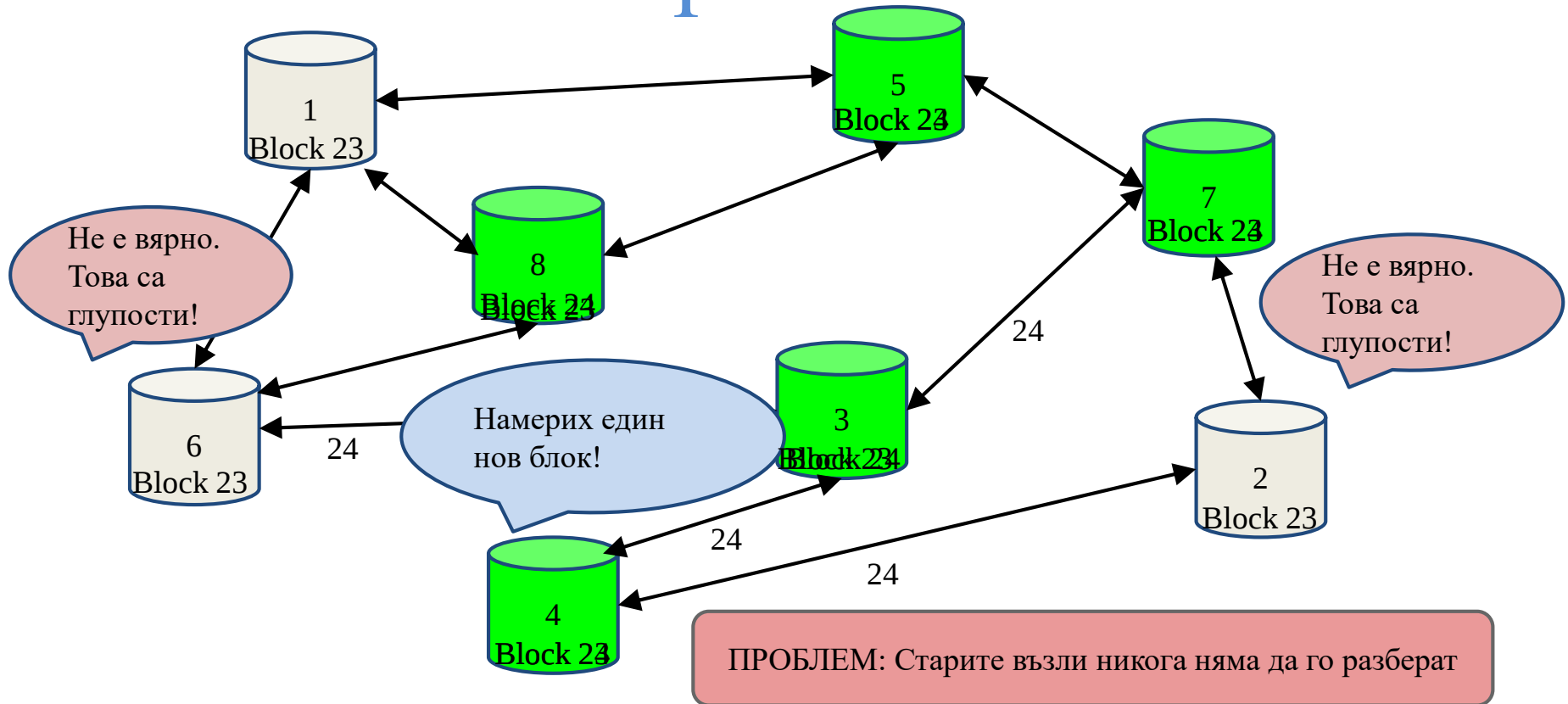
- Транзакцията е валидна с текущата блокчейн верига (по подразбиране)
- Изпълнява се скрипт за всеки предишен изход, който се използва, като скриптът трябва да връща true
- Проверява ли дали съвпада с „белия списък“
- Препредава се ако не е „видяна“ вече
- Препредава се ако не е в конфликт с други, които вече възелът е препредавал
- Да се избягват безкрайни цикли и double-spends

Възлите може да се различават в пула с транзакции



- Транзакциите или блоковете може да са в конфликт
- Поведение по подразбиране: първо се приема това, което се чуе първо
- Позицията в мрежата е важна

Проблем



Решение soft fork:

- да се добавят нови функции, които ограничават само набора от валидни транзакции.
- Необходими са мнозинство от възлите, за да се наложат новите правила.
- Старите възли ще одобрят. Но има риск старите възли да добиват вече невалидни блокове

Други възможни решения

- Нови схеми за подписване
 - Допълнителни метаданни за всеки блок
 - Вмъкване на параметъра coinbase
 - Свързване с UTXO дървото във всеки блок
 - Нови операционни кодове
 - Промени в ограниченията за размер
 - Промени в скоростта на добив
 - Много малки корекции на грешки
-но все още не са приложени ☺

Разпространение на блокове

- То е почти идентично
- Препредава се нов блок, когато устройството(възелът) го чуе, ако:
 - Блокът отговаря на хеш целта
 - Блокът има всички валидни транзакции – трябва да се изпълнят всички скриптове, дори и да не се препредава
 - Блокът се изгражда върху най-дългата текуща верига – да се избягват/игнорират forks

Размер на мрежата

- Не може да се изчисли точно
- Появяват се нови 1 млн. IP адреса на месец
- Но... Напълно активни са около 5-10000 възли:
 - които са постоянно включени
 - и напълни валидирани(възелът съхранява цялата блокчейн верига и чува и препраща всеки възел/транзакция)

Съхранение на транзакциите

Проблем:

- Unspent Transaction Output – се съхраняват в RAM (но това са около 65 млн. от 300 млн. транзакции, което е няколко GB за съхранение)
- Останалите – на диска на устройството

Решение:

- Тънки/SPV клиенти (невалидиращи напълно) – Изисква само няколко десетки мегабайта - съхраняват block headers; проверяват дали пъзелът е решен правилно, но не може да проверят всяка транзакция във всеки блок; валидират само онези транзакции, които ги засягат; заявяват транзакции, когато е необходимо, за да проверят входящо плащане; Доверяват се на напълно валидиращите ВЪЗЛИ

Параметри на bitcoin мрежата

- 10 мин. средно време за създаване на блок
- 1 MB в 1 блок
- 20 000 операции за подписване на блок
- >250 bytes/transaction
- 7 transactions/sec (**VISA: 2,000-10,000 transactions/sec; PayPal: 50-100 transaction/sec**)
- Само 1 алгоритъм за подпис (ECDSA/P256)
- Твърдо кодирани хеш функции
- 100 М сатоши на биткойн
- 23 000 000 общо максимум биткойни (ще се изчерпят към 2040г.)
- 50, 25, 12.5... награда за добив на биткойн

Средно време за потвърждение на транзакция

Average Confirmation Time

The average time for a transaction with miner fees to be included in a mined block and added to the public ledger.

Scales
Linear

1D
Average

Type
Line

Colors
Blue Black



1M

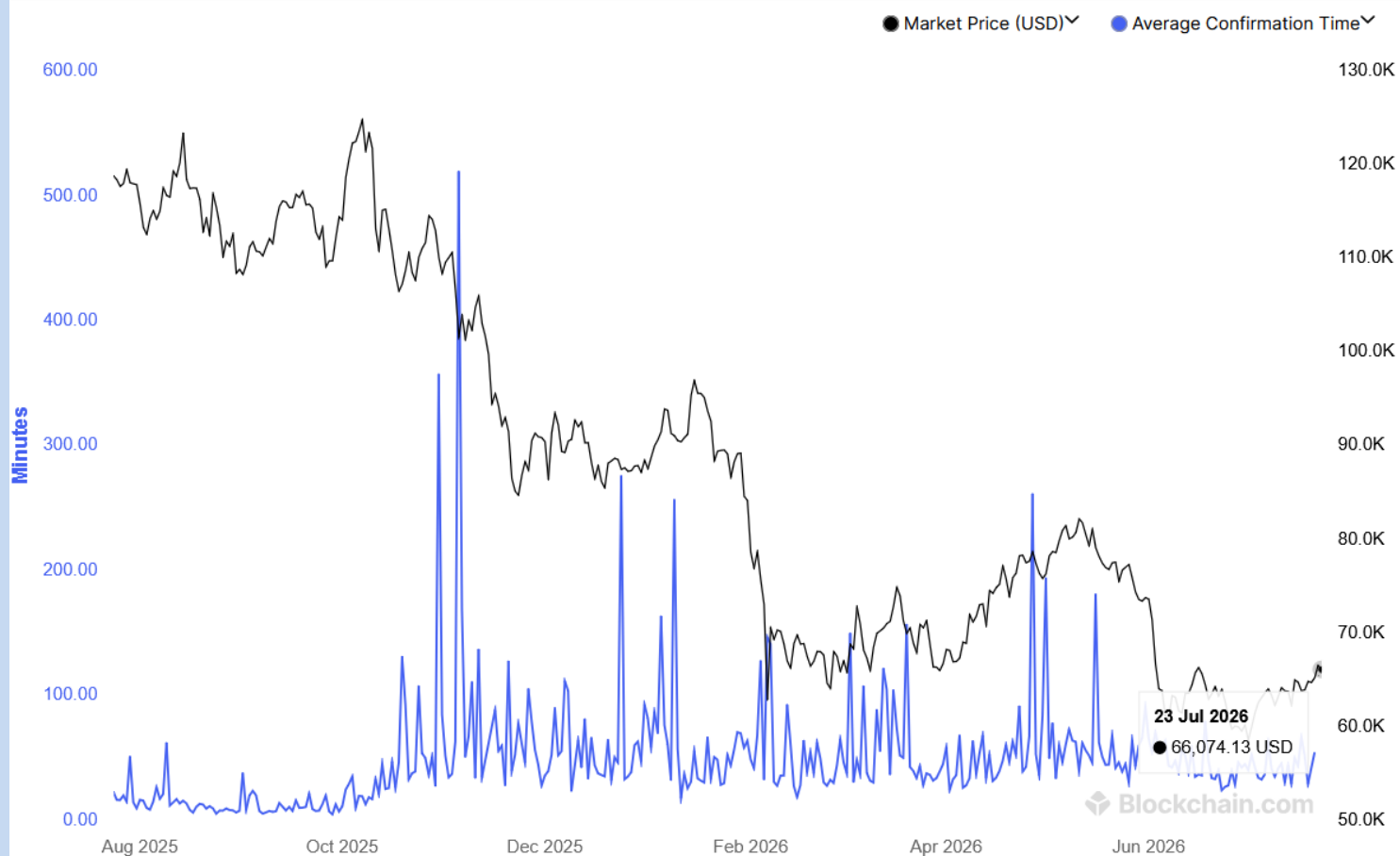
3M

6M

1Y

3Y

All

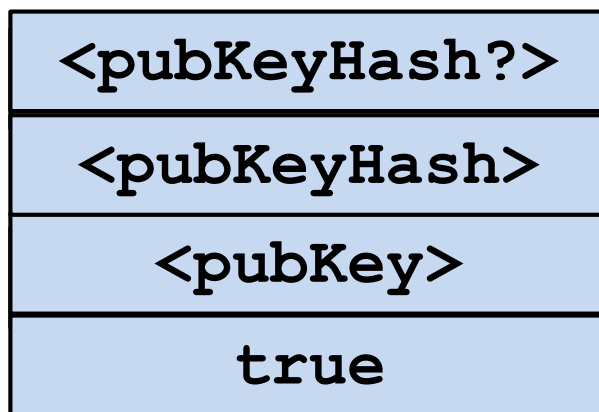


Най-важни скрипт инструкции

Name	Functions
OP_DUP	Дублира най-горния елемент в стека
OP_HASH160	Хешира два пъти: първо с SHA-256, след това с RIPEMD-160
OP_EQUALVERIFY	Връща true, ако входните данни са равни, в противен случай-false (маркира транзакцията като невалидна)
OP_CHECKSIG	Проверява дали входната signatures е валидна, използвайки входния публичен ключ за хеша на текущата транзакция.
OP_CHECKMULTISIG	Проверява дали t signatures върху транзакцията са валидни от t (out of n) от посочените публични ключове

Примерно изпълнение на скрипт

- Повечето взли добавят в белия списък известни скриптове
- 99,9% са прости проверки на подписи
- ~0,01% са MULTISIG
- ~0,01% са Pay-to-Script-Hash
- Останалите са грешки

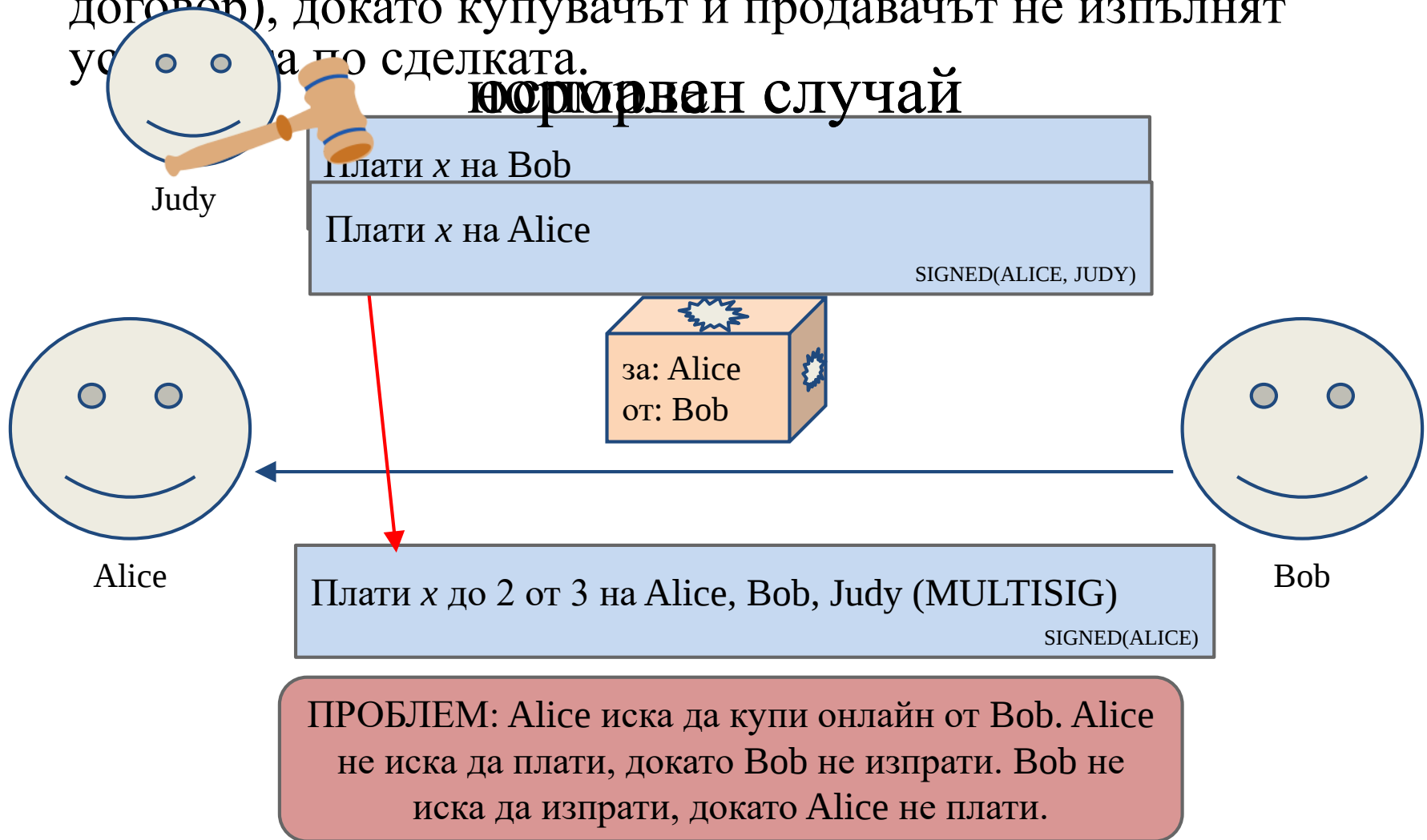


```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Депозитни транзакции

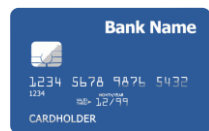
Средствата се задържат от неутрална трета страна (смайт договор), докато купувачът и продавачът не изпълнят условията по сделката.

Юристован случай

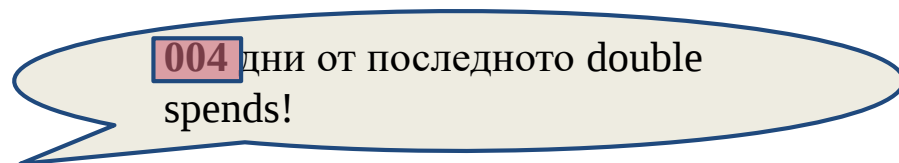


Транзакции с green address

- "Green address" е специален адрес, използван от доверени борси или портфейли (като Blockstream Green).
- Той позволява получателят да кредитира биткойните незабавно, без да чака стандартното потвърждение от блокчейн мрежата, тъй като услугата гарантира транзакцията.



Bank



Аз съм, Алис! Можеш ли да
направиш зелено плащане на
Боб?



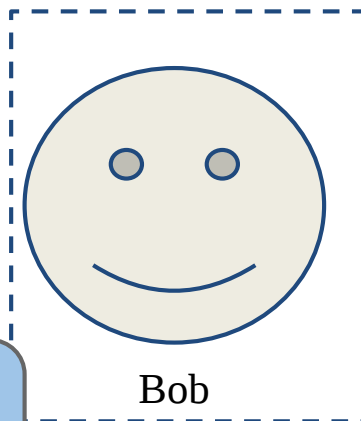
Alice

Плати x на Bob, y на Bank

No double spend

SIGNED(BANK)

Студено съхранение



Bob

ПРОБЛЕМ: Алис иска да плати на Боб.
Боб не може да чака 6 верификации, за да се
предпази от double spends, или е напълно офлайн.

Стартиране на HyperLedger мрежа

На локална Linux машина:

- Инсталиране на системни изисквания - Docker и Docker Compose, Git, Go
- Изтегляне на Fabric примери, бинарни файлове и Docker изображения
- Стартиране на самата мрежа
- Създаване на Канал (Channel)
- Инсталиране и стартиране на умен договор (Chaincode)
- Когато приключите работа, е изключително важно да изтриете контейнерите и генерираните криптографски ключове, за да не се застъпят при следващо стартиране

Присъединяване на нов node в HyperLedger

- Присъединяването на ново физическо или виртуално устройство (наричано още **възел / Node**) към съществуваща мрежа на **Hyperledger Fabric** се състои в добавянето на нов **Peer** или **Orderer** възел.
 - Подготовка на новото устройство - инсталиране на софтуера и конфигуриране
 - Генериране на Криптографски Сертификати (Идентичност) – със Certificate Authority (CA) сървър (на порт 7054) или за тестова среда – статичен ключ
 - Конфигуриране и стартиране на Docker контейнера
 - Присъединяване на новото устройство към Канал (Channel)
- След успешно изпълнение на peer channel join, новото устройство ще започне автоматично да комуникира с останалите възли (чрез Gossip протокола) и ще изтегли цялата история на транзакциите от блокчейна, за да се синхронизира.

Разпространение на транзакции в HyperLedger

- **Предложение (Proposal):** Клиентското приложение изпраща транзакцията през gRPC (порт 7051) само до специфични Endorsing Peers (одобряващи възли), определени от умния договор.
- **Симулация и Подпис (Endorsement):** Одобряващите възли изпълняват умния договор изолирано, генерират т.нар. Read/Write set (какви данни ще се променят) и подписват транзакцията криптографски. Те я връщат обратно на клиента, без да я разпространяват в мрежата.
- **Изпращане към Консенсуса (Ordering):** Клиентът събира нужните подписи и изпраща транзакцията към Ordering Service (на порт 7050).
- **Подреждане в Блокове:** Организаторът (Orderer) събира транзакции от различни клиенти, подрежда ги хронологично и ги пакетира в блок.
- **Финално разпространение (Broadcast):** Orderer възелът изпраща готовия блок обратно към Leader Peers на всяка организация, които го разпространяват локално към останалите възли чрез вътрешен Gossip протокол, за да бъде записан (Committed) в дигиталния регистър.

Кога се препредава предложена транзакция?

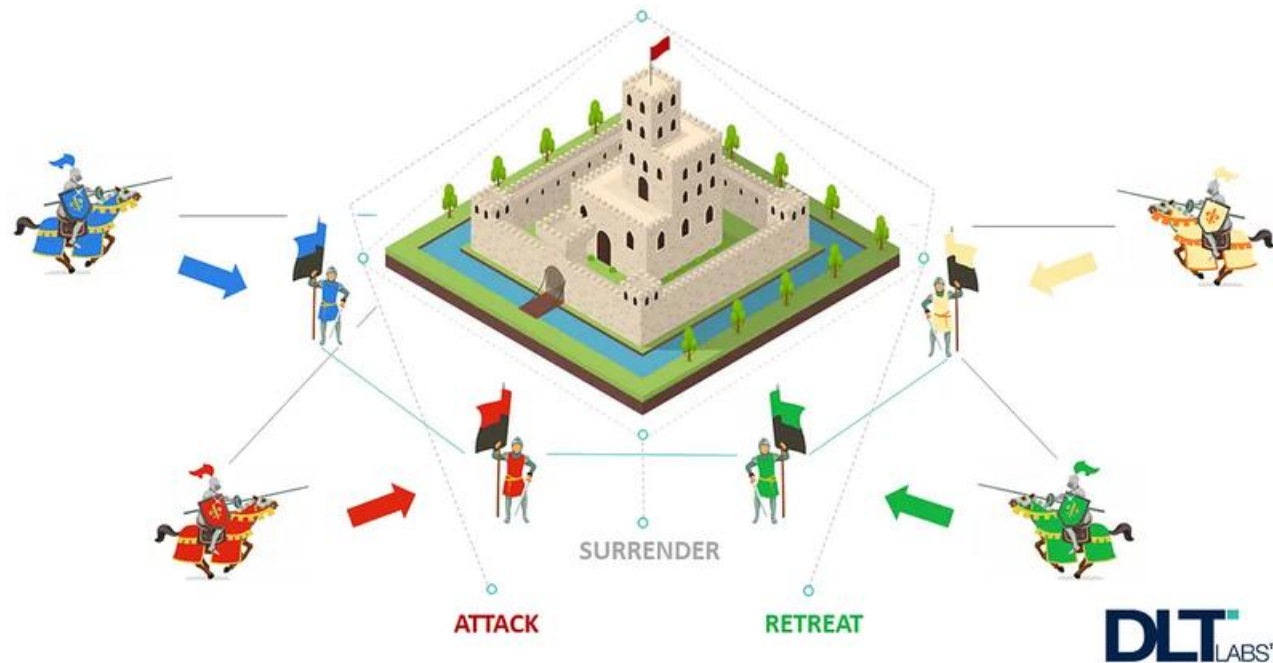
Разпространение на блокове

- Генериране на блока от Orderer възела
 - Услугата по подреждане (Ordering Service), която използва алгоритми като Raft или SmartBFT, събира валидираните транзакции, подрежда ги хронологично и ги пакетира в нов блок.
- Изпращане (Deliver) към водещите възли (Leader Peers)
 - Orderer възелът изпраща готовия блок само към т.нар. **Водещи възли (Leader Peers)** на всяка отделна организация в конкретния Канал (Channel).
 - Блокът се изпраща през криптиран TLS канал
- Вътрешно разпространение чрез Gossip протокол
 - Leader Peer изпраща по порт 7051 до произволно избрани съседни възли от неговата организация.
 - Тези възли го записват и препращат на съседите си
- Валидация и Запис (Commit) на блока
 - Проверка дали транзакциите в блока имат нужния брой подписи, проверка за двойно харчене, маркират се невалидни транзакции (тези, които не минат проверката), валидните се записват

Размер на мрежата

- Размерът на една **Hyperledger Fabric** мрежа не е фиксиран и не може да се измери в гигабайти по същия начин, по който измерваме публичните блокчейни
- **технически (физически) размер на данните** - размерът на дисковото пространство на един възел зависи изцяло от конкретния бизнес проект
- **структурен размер (мащаб на мрежата)** – средно между 2 и 20+ организации, като всяка организация поддържа по 2 до 4 Peer възела, 3, 5 или 7 orderer възела

Practical Byzantine Fault Tolerance



Всяко феодално владение на Византийската Римска империя е имало армия под командването на генерал и генералът е можел да разчита само на пратениците, за да предава информация между тях. По време на война византийската армия е можела да спечели целта само когато е имала числено превъзходство. Въпреки това, в армията може да има предатели и когато вражеските сили се обединят с тях и броят на лоялните генерали е превъзхождан числено, атаката ще се провали.

Practical Byzantine Fault Tolerance

Проблемът с Византийския генерал е въпрос за малцинството, подчиняващо се на мнозинството.

Решението на проблема:

- n генерали трябва да постигнат консенсус
- f генерали може да са предатели
- Тогава алгоритъм за гласуване е:
 - Ако един генерал види $f + 1$ еднакви отговора, този отговор трябва да е правилен

Пример:

- Четири реплики, които се опитват да се споразумеят за стойността на един бит

	Replica 1	Replica 2	Replica 3	Replica 4
Replica 1	1			
Replica 2		1		
Replica 3			1	
Replica 4				0

Пример

- Всички реплики изпращат стойността си към другите реплики

	Replica 1	Replica 2	Replica 3	Replica 4
Replica 1	1	1	1	0
Replica 2	1	1	1	0
Replica 3	1	1	1	0
Replica 4	1	1	1	0

Пример

- Сега всички реплики изпращат техния вектор към всички други реплики
- 2 изпраща стойности за $\langle 2,3,4 \rangle$ до 1 и $\langle 1,2,4 \rangle$ до 3

	Replica 1	Replica 2	Replica 3	Replica 4
Replica 1	1	$\langle 1,1,0 \rangle$	$\langle 1,1,0 \rangle$	$\langle 0,0,0 \rangle$
Replica 2	$\langle 1,1,0 \rangle$	1	$\langle 1,1,0 \rangle$	$\langle 0,0,0 \rangle$
Replica 3	$\langle 1,1,0 \rangle$	$\langle 1,1,0 \rangle$	1	$\langle 0,0,0 \rangle$
Replica 4	$\langle 1,1,1 \rangle$	$\langle 1,1,1 \rangle$	$\langle 1,1,1 \rangle$	0

- Резултатът е най-често срещаната стойност във вектора

	Replica 1	Replica 2	Replica 3	Replica 4
Replica 1	1	1	1	0
Replica 2	1	1	1	0
Replica 3	1	1	1	0
Replica 4	1	1	1	0

Пример

- Ще работи ли това с 3 реплики, едната от които е дефектна?
- Подсказка: това е асинхронна среда
- Извод: Правилото е:
- n реплики трябва да постигнат консенсус
- f реплики са „предатели“

$$n \geq 3f + 1$$

Practical Byzantine Fault Tolerance

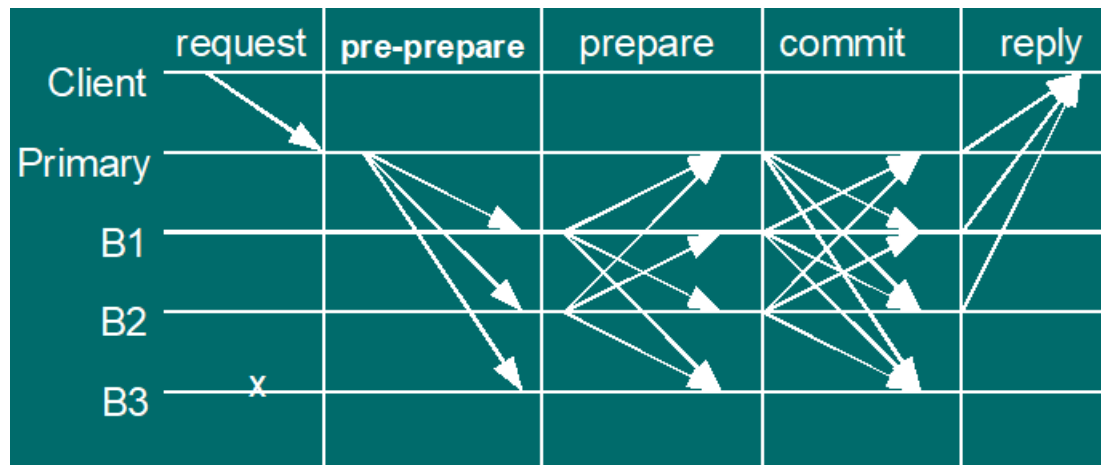
- Този алгоритъм (BFT) осигурява безопасност и работоспособност в асинхронна мрежа.
- Безопасност
 - системата поддържа състоянието и изглежда на клиента като нерепликирана отдалечена услуга. Безопасността включва пълно подреждане на заявките.
- Работоспособност
 - клиентите в крайна сметка ще получат отговор на всяка изпратена заявка, при условие че мрежата функционира.

РВFT- същност

- Базирано на репликация на машина на състоянията
- Съобщения, подписани с криптографски публичен ключ
- Message digests, създадени с помощта на хеш функции, устойчиви на колизии
- Използва консенсус и разпространение на системни изгледи: състоянието се променя само когато функциониращите реплики се споразумеят за промяната

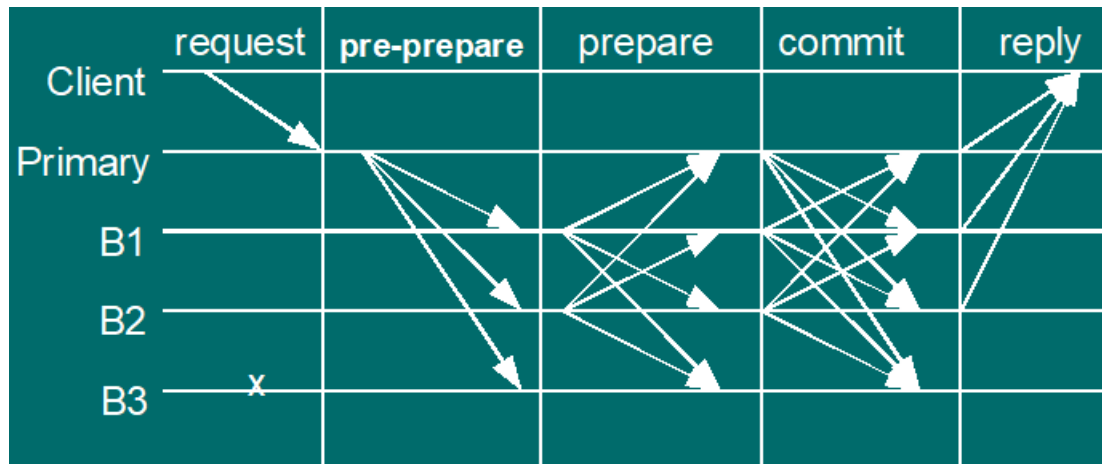
РВFT- алгоритъм (1/2)

- Клиентът изпраща заявка до основния node.
- Основният node присвоява на заявката пореден номер и го излъчва до всички реплики (предварителна подготовка).
- Репликите потвърждават този пореден номер (подготовка).
- След като бъдат получени $2f$ заявки за подготовка, клиентът излъчва приемането на заявката (*commit*).



РВFT- алгоритъм (2/2)

- След като са получени $2f + 1$ *commits*, клиентът поставя заявката в опашката.
 - В клиент без дефекти (*non-fault*), опашката от заявки ще бъде напълно подредена по пореден номер.
- След като всички предишни заявки бъдат изпълнени, заявката ще бъде изпълнена и резултатът ще бъде изпратен директно на клиента.
- Всички тези съобщения се регистрират.



РВFT- Преглед на промените (1/2)

Ако основният node е повреден

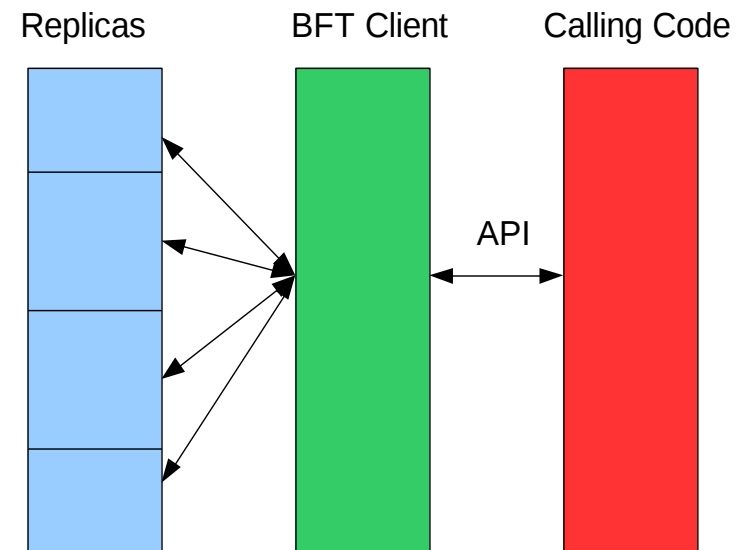
- Клиентът използва време за изчакване. Когато това време изтече, заявката се изпраща до всички реплики.
- Ако реплика вече **знае** за заявката, повторното излъчване се игнорира.
- Ако реплика **не знае** за заявката, тя ще стартира таймер. При изтичане на времето за изчакване на този втори таймер, репликата стартира процеса на промяна на изгледа.

РВFT- Преглед на промените (2/2)

- Ако таймерът на реплика изтече, тя изпраща съобщение за промяна на изгледа.
- Това съобщение съдържа състоянието на системата (под формата на архивирани съобщения), така че другите възли да знаят, че репликацията не е претърпяла неуспех.
- Ако текущият изглед е v , възел $v+1 \pmod n$ чака $2f$ валидни съобщения за промяна на изгледа.
- След като $v+1$ получи $2f$ съобщения за промяна на изгледа, той изпраща мултикаст съобщение за нов изглед.
 - Това съобщение съдържа всички валидни съобщения за промяна на изгледа, получени от $v+1$, както и набор от O от всички заявки, които може да не са завършени все още (поради първична повреда).
- След като реплика получи валидно съобщение за промяна на изгледа, тя влиза в изглед $v+1$ и обработва O .
- Докато се извършва промяна на изгледа, не се приемат нови заявки.

РВFT алгоритъм: гледна точка на клиента

- Клиентът трябва да внедри време за изчакване за промяна на изгледа
- Трябва да се чакат отговори директно от реплики
- Клиентът чака $f+1$ отговора, след което приема резултата.
- Безпроблемната интеграция изисква тристепенен подход (за да се осигури безпроблемно взаимодействие с извикващия код, трябва да има клиентски слой между репликите и програмата)



PBFT алгоритъм в HyperLedger

Предимства	Недостатъци
	Не е мащабируем
	Значителни разходи
Висока отказоустойчивост	

Публично срещу частно бизнес решение

критерий	Ethereum	HyperLedger Fabric
Глобален размер	Еднакъв за всички (Над 500+ GB през 2026 г.)	Различен за всеки възел (Зависи от каналите)
Брой участници	Хиляди анонимни възли по целия свят	Ограничен брой (Известни бизнес партньори)
Ръст на данните	Непрекъснат и неконтролируем от бизнеса	Контролиран чрез бизнес логика и изтриване на данни
Съхранение на данните	Скъпо съхранение директно на веригата	Използване на Private Data (Хеш на веригата, файл извън нея)

Въпроси ?

Благодаря за вниманието !