Scalability II

50.037 Blockchain Technology

Paweł Szałachowski

Directed Acyclic Graph (DAG)

DAG

- Nakamoto-like consensus is very easy
- The chain structure is simple
    - Can we introduce a more efficient data structure?
- Graphs
    - More powerful, as blocks can encode their worldviews
    - Why acyclic?

SPECTRE

- https://eprint.iacr.org/2016/1159.pdf (payment oriented)
- Intuitions and insights:
    - DAG can be used to encode the longest-chain rule
    - Bitcoin is related to voting
        - Every block votes for its chain
        - Cloning in Bitcoin
    - Vote amplification (miners strengthen the majority decision)

SPECTRE

- Miners create PoW-protected blocks with Txs
  - Blocks point to all know *tips* of the DAG
    - Claiming they were created after them
- Blocks can have conflicting Txs
- Nodes maintain local copies of DAG and accept/reject Tx
  - RobustReject, Pending, RobustAccepted

SPECTRE

- Vote over blocks (pairwise)
  - How many think A<B vs how many think B<A
    - This is just interpretation, does not have to be “*true*”
- Use results to accept/reject Txs
  - Tx of block B if:
    - All inputs are accepted
    - For every conflicting Tx’ in B’, B < B’

SPECTRE

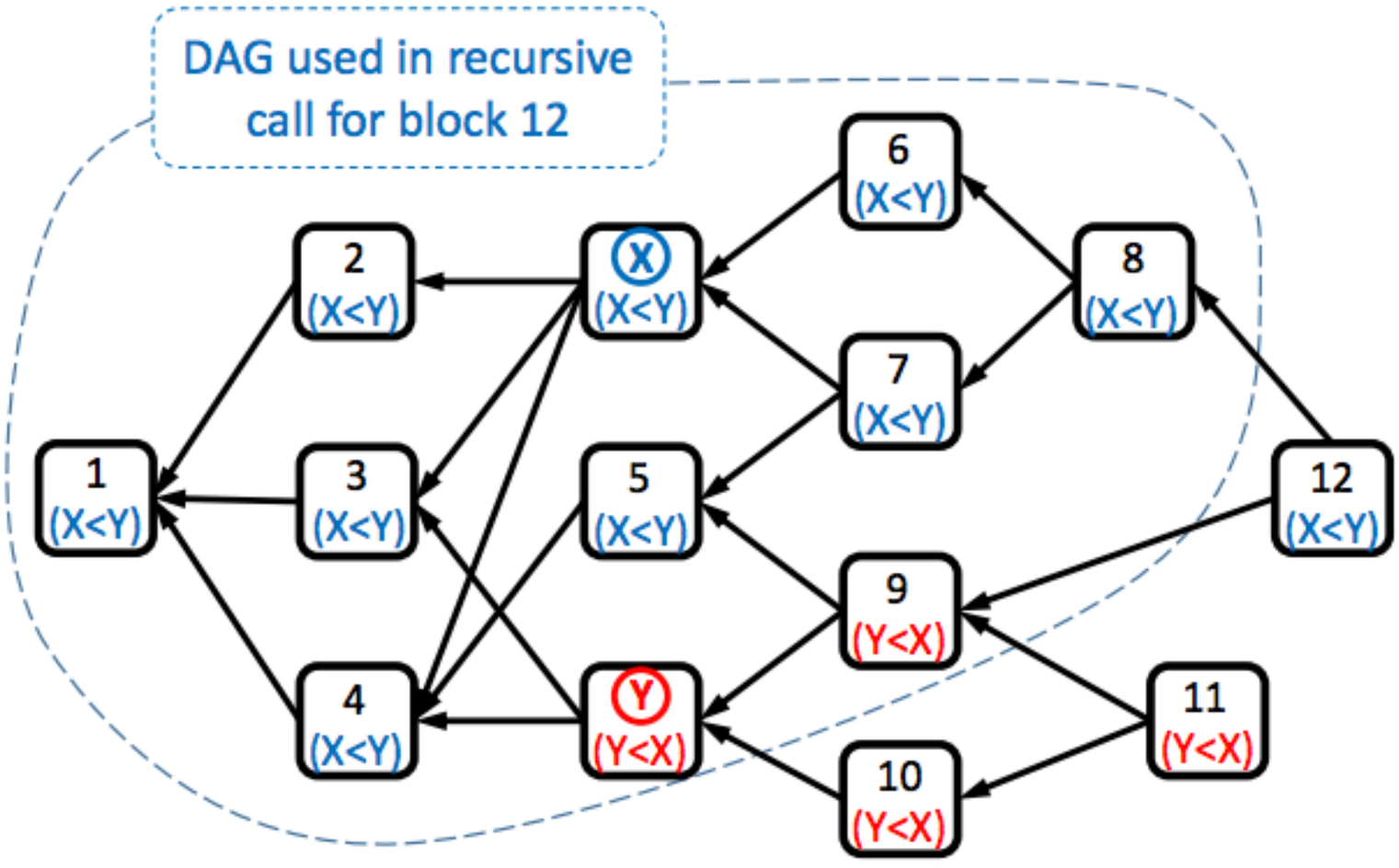


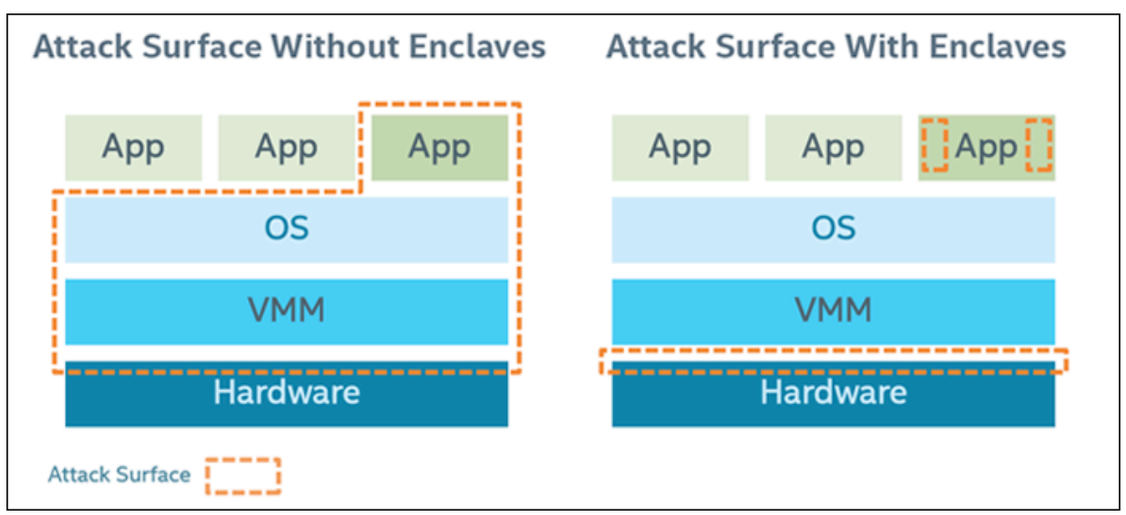
Fig. 1: An example of the voting procedure on a simple DAG. Block x and blocks 6-8 vote $x \prec y$ as they only see x in their past, and not y. Similarly, block y and blocks 9-11 vote $y \prec x$. Block 12 votes according to a recursive call on the DAG that does not contain blocks 10,11,12. Any block from 1-5 votes $x \prec y$, because it sees more $x \prec y$ voters in its future than $y \prec x$ voters.

- Clone-proof and amplified decisions
- Quick confirmations

Intel’s Proof of Elapsed Time (PoET)

Intel Software Guard Extensions (SGX)

- Set of new CPU instructions (Trusted Execution Environment — TEE)
- User code can allocate private regions of memory (**enclaves**), protected from other processes (even those running at higher privilege levels)
    - Running code and its memory is isolated from the rest of system
- Minimize trusted base
    - only CPU is trusted (even DRAM is untrusted, encryption needed)
- Attestation
    - Prove to remote system what code enclave is running



PoET

- Observation: PoW is introduced to mitigate Sybil attacks
  - … but with TEE that could be easier
- Idea: simulate PoW by sleep(…);
  - PoW-like guarantees
  - No energy waste
    - More energy vs more Intel SGX CPUs
  - Intel & SGX are trusted, not fully open, + see the recent attacks

PoET

1. A newcomer node downloads the trusted code and sends a *join* message with the signed attestation

2. Nodes verify and accept/reject

3. In each round, every nodes gets a trusted random R and calls

   sleep(R);

4. The first awake node sends a signed msg that she is a leader

5. The statement is validated and the blocks can be produced

Permissioned Blockchains

Permissioned Blockchains

- We discuss open/permissionless blockchains so far
    - Great for some use cases, not so great for other
        - Good: Append-only, decentralized, available, robust, transparent…
        - Bad: Slow, expensive storage&computation, volatility, immature technology, publicly available data, difficult to manage/update…
            - Mainly caused by the permissionless setting
- Why not reuse some of those ideas and run a classic BFT consensus?
- Efficiency could be one of the major benefits

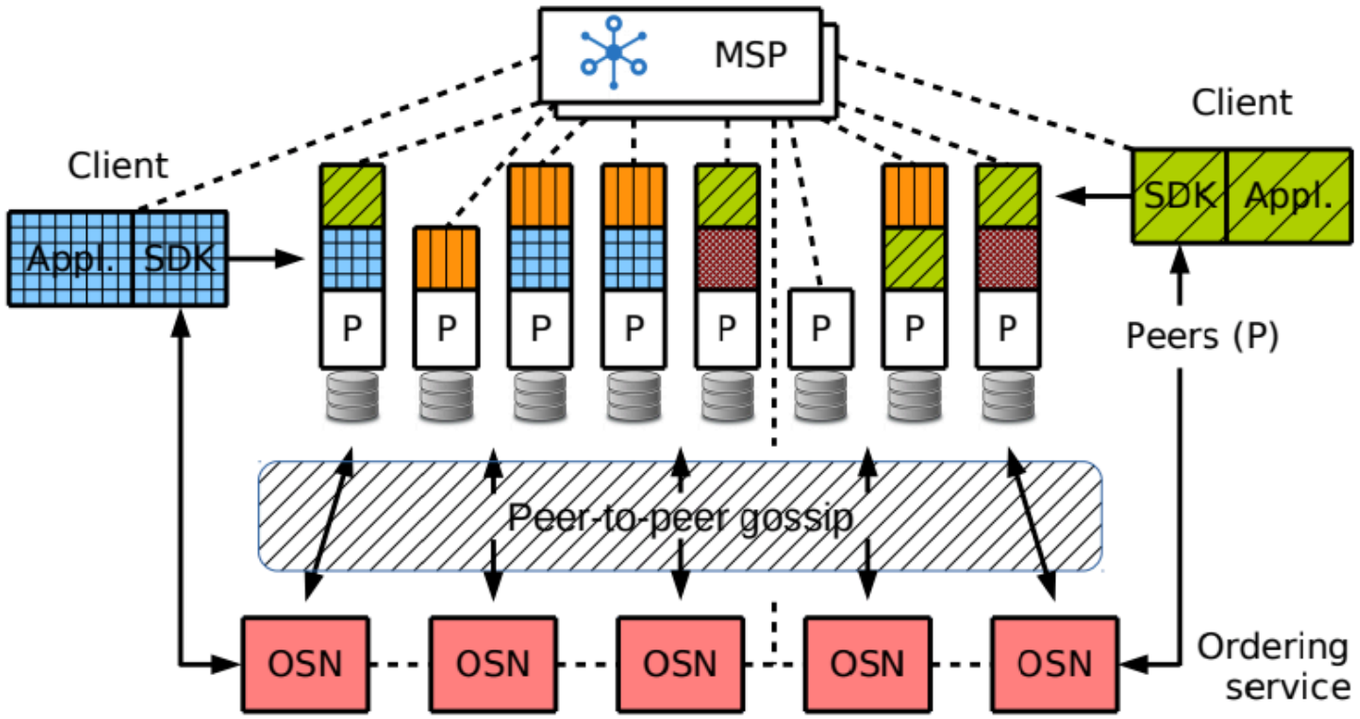
HYPERLEDGER

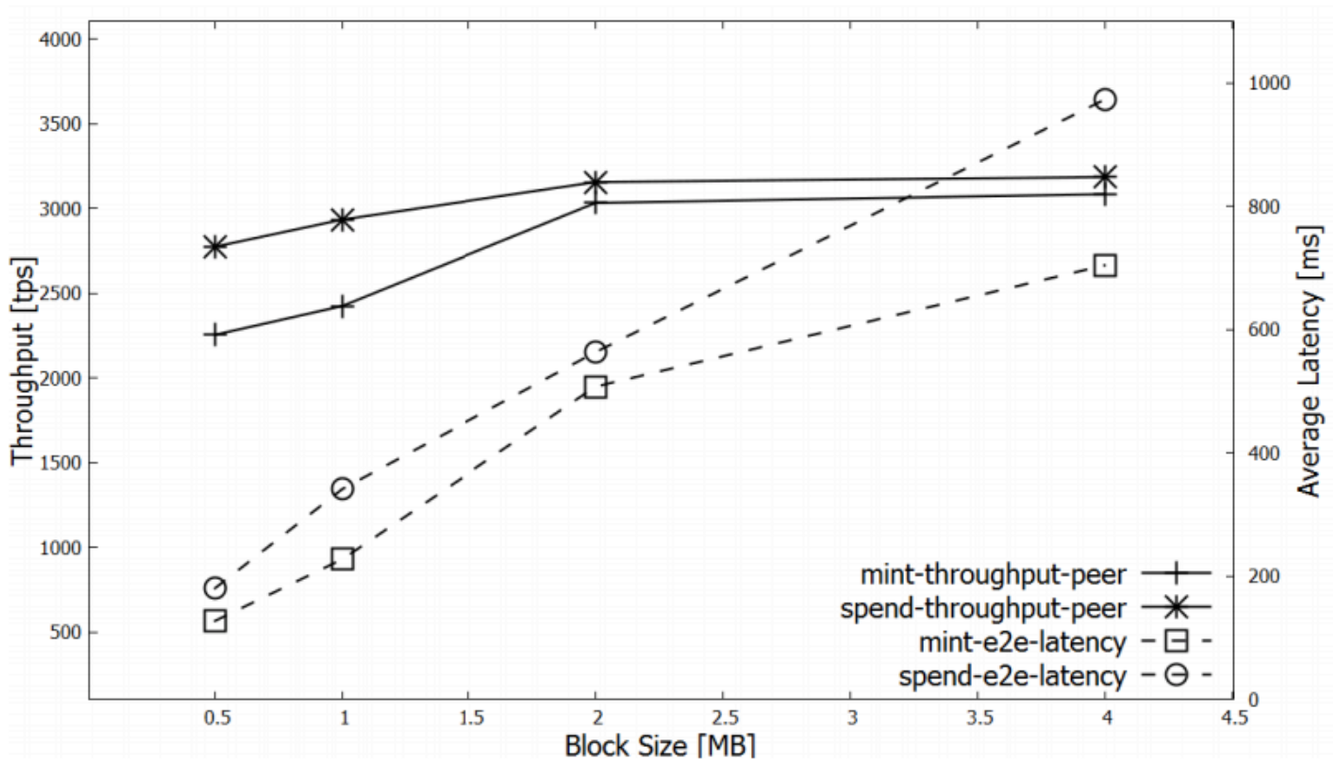
- Consortium
- Hyperledger Fabric (framework)
  - https://arxiv.org/abs/1801.10228
- Business-oriented
- Different consensus protocols supported
- No built-in cryptocurrency
- Powerful smart contracts

Hyperledger Farbic

- Membership service provider (MSP) provides identities to participants
- Clients submit transaction proposals for execution
- Peers execute transaction proposals and validate transactions
  - Only endorsing peers execute transactions (specified by a policy)
  - All peers maintain the blockchain ledger
- Ordering Service Nodes (OSN) establish the total order of all transactions

Hyperledger Fabric





Centralized Ledgers

Certificate Transparency

- Maybe for some applications we could just use a centralized ledger?
- Problem: Certificate Authorities (CAs) can misbehave/get compromised unnoticed
  - How to detect attacks? -> Certificate Transparency
    - Desired properties similar as in blockchain: a public, verifiable, append-only log
- Intended for certificates but can be generalized to logging arbitrary artifacts
- Implemented, standardized https://tools.ietf.org/html/rfc6962, in production
- Pros: simpler, easier to manage, deploy, high-throughput, energy-friendly, …
- Cons: availability, censorship, trusted (in some scope), easier to misbehave, …

Certificate Transparency

- A log maintains an append-only hash tree able to prove a) a presence of a certificate, and b) that the log is append-only
    - Everyone can audit the log
- For every certificate submitted, the log issues an SCT
    - Promise to append the certificate to the log
    - Certificates are sent to clients along with the corresponding SCTs
- Every update, new certificates are appended and the log generates an STH
- Clients need to make sure that: a) the certificate is indeed logged, b) that the view of the log is consistent with other clients (that os challenging!)

Reading

- Textbook 8.5

- inline references