Sycomore, a Permissionless Distributed Ledger that Self-adapts to Transaction Demand

Emmanuelle Anceaume, CNRS / IRISA

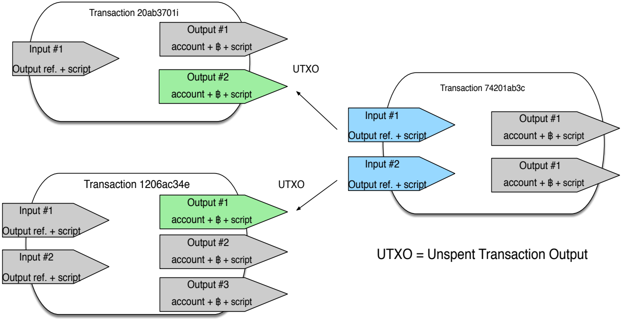
Antoine Guellier, CNRS / IRISA
Romaric Ludinard, IMT / IRISA
and Bruno Sericola, INRIA / IRISA

Cryptocurrency and payment systems (Bitcoin-like)

- Fully decentralized
- No public key infrastructure
- Permissionless
  - ✗ Such constraints make the **general problem of consensus impossible to solve**
- Relies on rational behavior and incentives mechanisms
  - ✓ To **reach a consensus** on the cryptocurrency state

The state of the cryptocurrency system is represented by transactions

- Transfers currency from one user to another
- No inherent notion of identities or individual accounts which « own » bitcoins
- Ownership simply means knowing a private key which is able to make a signature that redeems outputs (scripts)



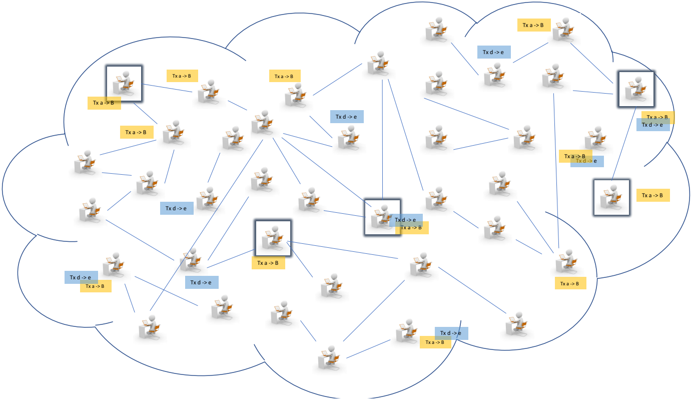
Do not forget Tx fees, i.e., ฿ input > ฿ output

- ✓ Signed transactions guarantee that only the owner of an output can redeem coins
- ✗ However signatures do not prevent double-spending attacks
- ✓ All transactions must be published in a global permanent transaction log, and any output must be redeemed by at most one subsequent transaction

> This log is implemented as a series of **blocks** of transactions

- Each block containing the hash of the previous block, committing this block as the unique antecedent
- This is the blockchain

- Topology formed through a randomized process
- No coordinating entity
- Broadcast is based on flooding/gossiping neighbors' link



Properties of the Network

1. Connectivity
   - Each node should receive any broadcast information

2. Low latency [1]

$$\frac{\text{msg. transmission time}}{\text{block creation time interval}} \ll 1$$

✓ Allows to keep the probability of fork small

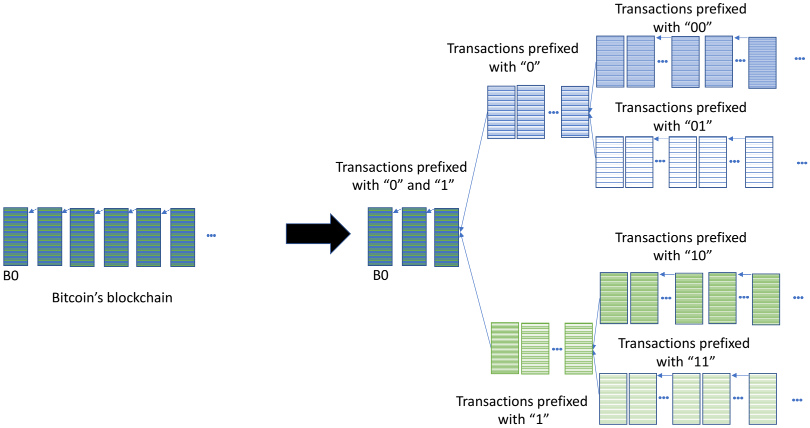
✓ PoW mechanism allows this ratio to continuously hold

✗ No more than 7 Tx/s can be permanently confirmed in average ! !

1. J. Garay and A. Kiayias, The Bitcoin Backbone Protocol : Analysis and Application, Eurocrypt 2015

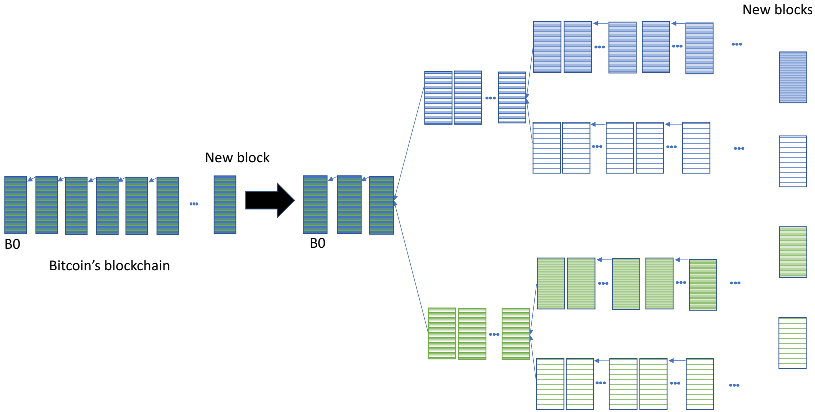
Objective : Building a ledger with the following properties

1. A ledger with several parallel (but not conflicting) chains such that
   - Causality between transactions is respected
   - Transactions should be partitioned among the chains



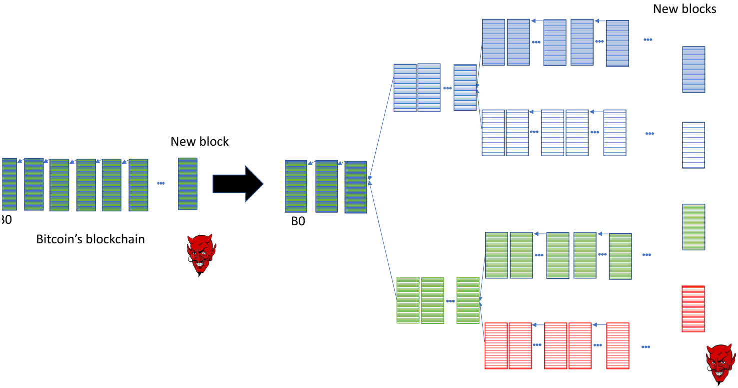
Objective : Building a ledger with the following properties

1. A ledger with several parallel (but not conflicting) chains
2. Valid (and not conflicting) blocks should be mined in parallel



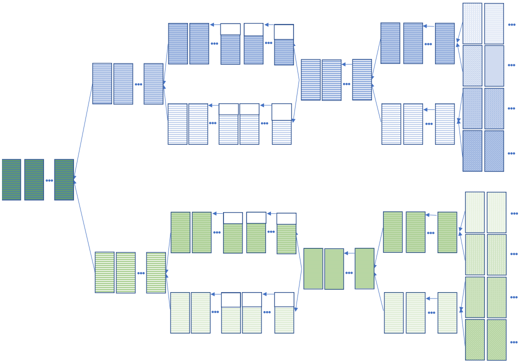
Objective : Building a ledger with the following properties

1. A ledger with several parallel (but not conflicting) chains
2. Valid (and not conflicting) blocks should be mined in parallel
3. Miners should not be able to predict the chain of the ledger to which their blocks will be appended
   - Cannot devote their computational power to a specific chain



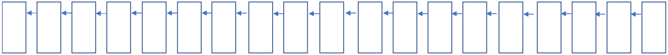
Objective : Building a ledger with the following properties

1. A ledger with several parallel (but not conflicting) chains
2. Valid (and not conflicting) blocks should be mined in parallel
3. Miners should not be able to predict the chain of the ledger to which their blocks will be appended
4. Overloaded chains should dynamically split and underloaded ones should dynamically merge



⚠ All these features should be verifiable by anyone at any time

Switching from a chain of blocks ....



... to Sycomore, a directed acyclic graph of blocks

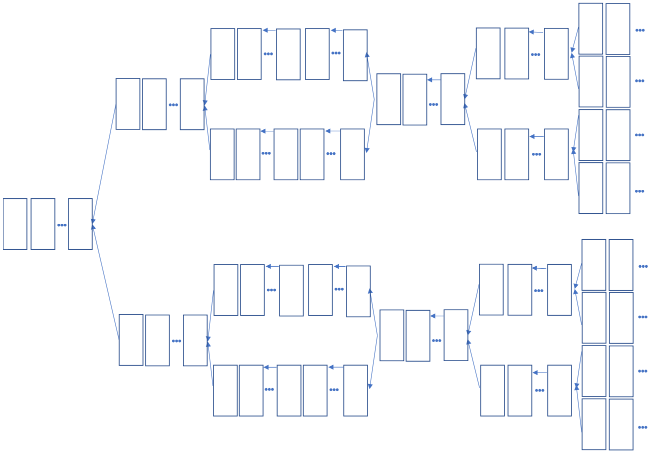


Figure – Sycomore ledger

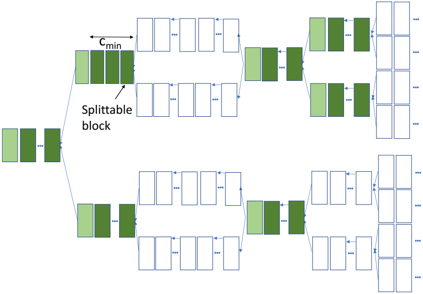
Model

- Permissionless system
- Adversary : no more than 50% of the network hashing power is held by the adversary
- Nodes have access to safe cryptographic functions (hash function and signature scheme)
- Each object of the system (i.e., transaction and block) is assumed to be uniquely identified
- No public key infrastructure to establish node identities

Definition (Splittable block)

- Let $c_{\min} \geq 1$ be a constant
- Let $\mathcal{C} = \langle b_1 b_2 \ldots b_c \rangle$ be a chain with $c \geq c_{\min}$ and $0 < \Gamma < 1$
- Block b_c is called a splittable block if

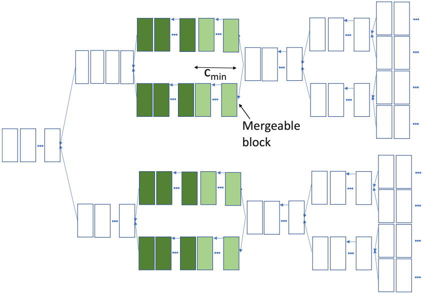
$$\frac{1}{c_{\min}} \sum_{j=1}^{c_{\min}} (\mathrm{Load}(b_{c-c_{\min}+j}) > \Gamma$$



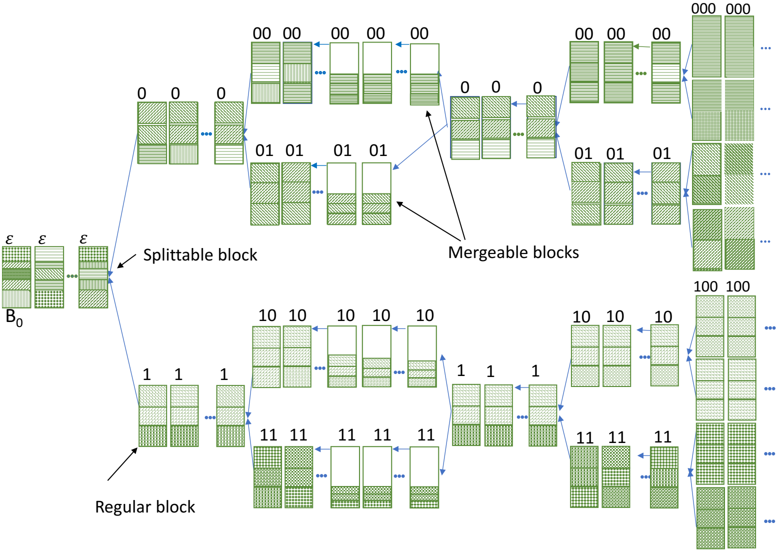
Definition (Mergeable block)

- Let $c_{\min} \geq 1$ be a constant
- Let $\mathcal{C} = \langle b_1 b_2 \dots b_c \rangle$ be a chain with $c \geq c_{\min}$ and $0 < \gamma < \Gamma \leq 1$
- Block b_c is called a mergeable block if

$$\frac{1}{c_{\min}} \sum_{j=1}^{c_{\min}} (\mathrm{Load}(b_{c-c_{\min}+j}) < \gamma$$



Any block in $\mathcal{C}$ which is neither splittable nor mergeable is called a regular block.



Splittable block
Mergeable blocks
Regular block
B_0
ε ε ε
0 0 0
00 00 00 00 00
01 01 01 01
0 0 0
00 00 00
01 01 01
000 000
1 1 1
10 10 10 10 10
11 11 11 11 11
1 1 1
10 10 10
11 11 11
100 100

The predecessor of a block is neither predictable nor choosable

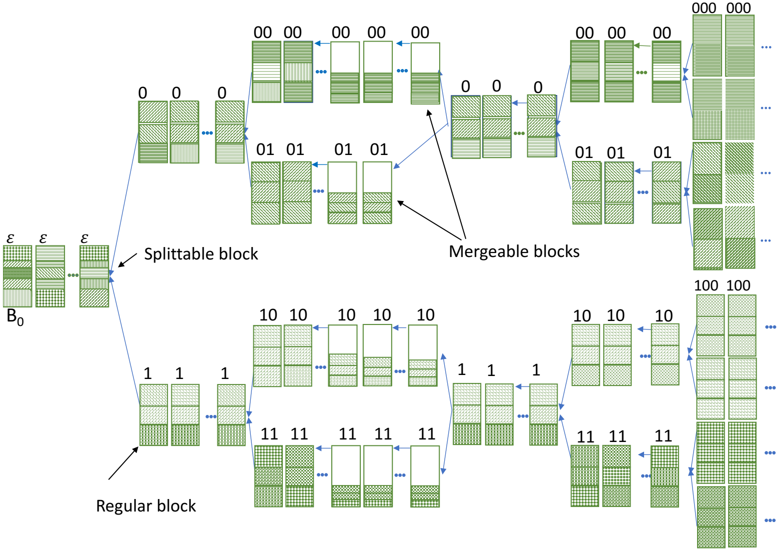


Figure – Local view $\mathcal{L}_u$ of the ledger $\mathcal{L}$ at some node u

The predecessor of a block is neither predictable nor choosable

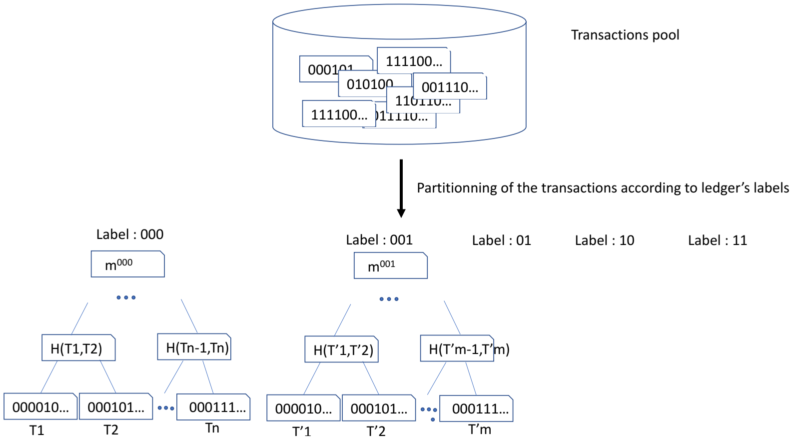
Recall that Bitcoin's block header contains a commitment of the chain state

$$\mathcal{H} = \{(\mathfrak{h}(b_k^{\epsilon}), m^{\epsilon})\}$$

1. In Sycomore, the header of a block b contains a commitment of the DAG state

$$\mathcal{H} = \left\{ \left(\mathfrak{h}(b_1^{\ell_1}), \ell_1', m^{\ell_1'}\right), \ldots, \left(\mathfrak{h}(b_j^{\ell_j}), \ell_j', m^{\ell_j'}\right) \right\}, \text{ where}$$

   - $\mathfrak{h}(b_i^{\ell_i})$, $1 \leq i \leq j$: reference to each leaf block
   - ℓ_i', $1 \leq i \leq j$: label of the future successor of $b_i^{\ell_i}$
   - $m^{\ell_i'}$, $1 \leq i \leq j$: Merkle root of the set of pending transactions whose prefix matches label ℓ_i'


Transactions pool
000101...
111100...
010100...
001110...
110110...
111100...
011110...
Partitionning of the transactions according to ledger's labels
Label : 000
m^{000}
H(T1,T2)
H(Tn-1,Tn)
000010...
000101...
000111...
T1
T2
Tn
Label : 001
m^{001}
H(T'1,T'2)
H(T'm-1,T'm)
000010...
000101...
000111...
T'1
T'2
T'm
Label : 01
Label : 10
Label : 11

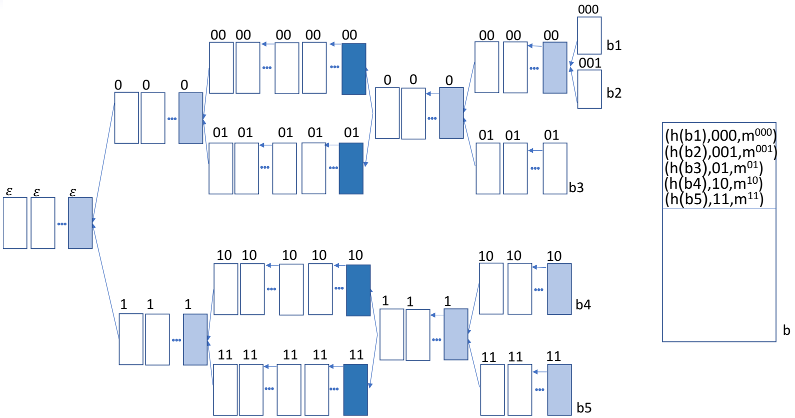


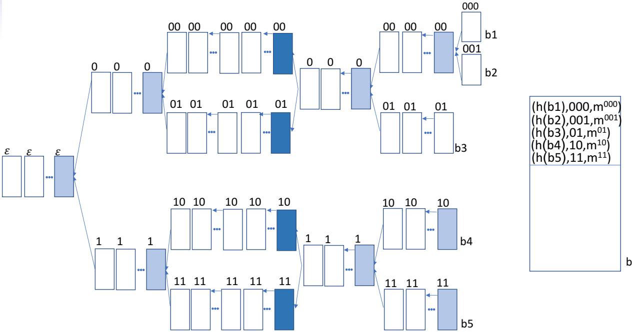
Figure – Local view $\mathcal{L}_u$ of the ledger $\mathcal{L}$ at some node u

The predecessor of a block is neither predictable nor choosable

2. Characterization of the predecessor of block b
   - Computation of the proof of work ν on block b's header
   - Predecessor of block b = leaf block $b_i^{\ell_i}$ s.t. the label ℓ_0 of its successor in $\mathcal{H}$ verifies

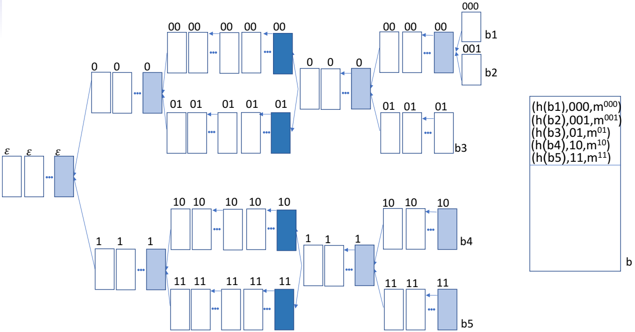
$$\ell_0 = \underset{\left(\mathfrak{h}(b_i^{\ell_i}),\ell_i',m^{\ell_i'}\right)\in\mathcal{H}}{\arg\min} \mathcal{D}(\ell_i', \mathfrak{h}(\mathcal{H}||\nu) \bmod 2^d)$$

   d is the number of bits of the longest label of the successors in $\mathcal{H}$
   $\mathcal{D}$ is the distance function between two bit strings (numerical XOR value)



$$\mathcal{H} = \left\{ \left(\mathfrak{h}(b^{000}), 000, m^{000}\right), \ldots, \left(\mathfrak{h}(b^{11}), 110, m^{110}\right), \left(\mathfrak{h}(b^{11}), 111, m^{111}\right) \right\}$$

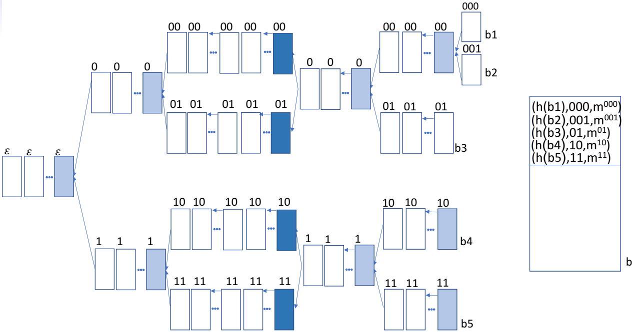
1. Computation of ν s.t. $\mathfrak{h}(\mathcal{H}||\nu) \leq T$



$$\mathcal{H} = \left\{ \left(\mathfrak{h}(b^{000}), 000, m^{000}\right), \ldots, \left(\mathfrak{h}(b^{11}), 110, m^{110}\right), \left(\mathfrak{h}(b^{11}), 111, m^{111}\right) \right\}$$

1. Computation of ν s.t. $\mathfrak{h}(\mathcal{H}||\nu) \leq T$

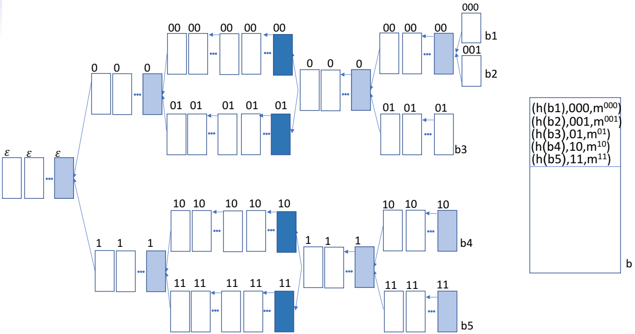
$$\mathfrak{h}(\mathcal{H}||\nu) = 000000\ldots0010010$$



$$\mathcal{H} = \left\{ \left(\mathfrak{h}(b^{000}), 000, m^{000}\right), \ldots, \left(\mathfrak{h}(b^{11}), 110, m^{110}\right), \left(\mathfrak{h}(b^{11}), 111, m^{111}\right) \right\}$$

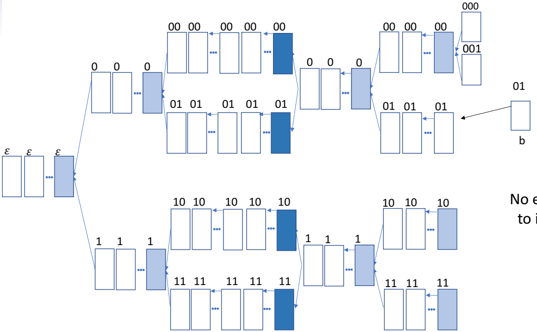
$$\mathfrak{h}(\mathcal{H}||\nu) = 000000\ldots0010010$$

2. For each b^{ℓ}, compute $\mathcal{D}(\ell', 000000\ldots0010010 \bmod 2^d)$



$$\mathcal{H} = \left\{ \left(\mathfrak{h}(b^{000}), 000, m^{000}\right), \ldots, \left(\mathfrak{h}(b^{11}), 110, m^{110}\right), \left(\mathfrak{h}(b^{11}), 111, m^{111}\right) \right\}$$

2. For each b^{ℓ}, compute $\mathcal{D}(\ell', 000000\ldots0010010 \bmod 2^d)$
3. Thus predecessor of b is block whose label is 01



$$\mathcal{H} = \left\{ \left(\mathfrak{h}(b^{000}), 000, m^{000}\right), \ldots, \left(\mathfrak{h}(b^{11}), 110, m^{110}\right), \left(\mathfrak{h}(b^{11}), 111, m^{111}\right) \right\}$$

For mergeable blocks (i.e. two tuples are the argmin), the predecessor is both mergeable blocks

The predecessor of a block is neither predictable nor choosable

To summarize

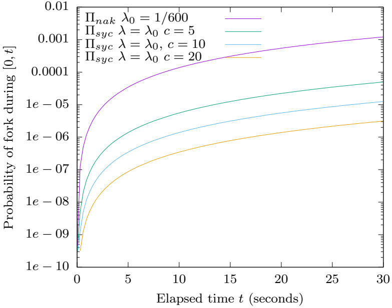
- Deriving the predecessor of a block requires to solve a notoriously difficult problem
- The predecessor is sealed in block header
- No explicit reference to the predecessor in block header
- Verifiable at any time by anyone

How forks are resolved ?

Rule (Fork Rule)

At any time, keep the DAG *for which the confirmation level of the genesis block is the largest.*

- Computed by determining the longest path of blocks that commits B_0
- Fork rule is exactly the same as Bitcoin's one !
- Natural since it amounts to favor the DAG that has been acknowledged by the majority of the miners



(a) Probability of fork as a function of time. The block creation rate is Bitcoin's one, i.e., $\lambda = 1/600$).

Let $\{N(t),\ t \geq 0\}$ be a Poisson process with rate λ representing the number of events in the interval $(0, t)$
For every $i = 1, \ldots, c$,

$$\begin{aligned} p_i(t) &= \mathbb{P}\{N_i(t) \geq 2\} \\ &= 1 - e^{-\lambda p_i t}(1 + \lambda p_i t). \end{aligned}$$

If $p_i = 1/c$, we get

$$\begin{aligned} p_i(t) &= \mathbb{P}\{N_i(t) \geq 2\} \\ &= 1 - e^{-\lambda t/c}(1 + \lambda t/c). \end{aligned}$$

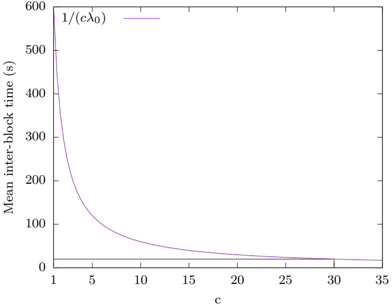


Figure – Mean inter-block time as a function of the number of leaf blocks c to meet Bitcoin's probability of fork

- E.g., $c = 30$ leaf blocks, blocks can be mined every 20 seconds !
- This adaptiveness is a remarkable feature of Sycomore

Conclusion and Open issues

- We have presented a new way to organise both transactions and blocks in a distributed ledger
- Sycomore allows us to keep all the remarkable properties of the Bitcoin blockchain in terms of security, immutability, and transparency, while enjoying higher throughput and self-adaptivity to transactions demand.

What's next ?

- Antoine Durand will present you our solution to switch from a proof-of-work setting to a **proof-of-stake** one !