INSA Centre Val de Loire
École doctorale MIPTIS

Etude du passage à l'échelle de la blockchain pour une solution d'e-paiement mobile

présentée par

M. Jean Yves Emmanuel ZIE DIALI

et soutenue

le date

en vue de l'obtention du

Doctorat de l'INSA Centre Val de Loire
Spécialité : Informatique

Arrêté du 7 août 2006

Titre de la thèse

Membre du Jury

Pour ...

Remerciements

...

Summary

Blockchains and cryptocurrencies enable users to exchange value in a trust-less manner. While some blockchains are permissioned, the most popular one are open and permissionless : anyone can join, "can leave and rejoin the network at will"[Nak08]. They offer a decentralized answer to distributed consensus. But they are not as performant as traditional systems. They trade scalability for security and decentralization. Croman *et al.* [CDE+16] raise the following questions : *Can decentralized blockchains be scaled up to match the performance of a mainstream payment processor ? What does it take to get there ?* We focus on the second question.

It is hard to assess the performance of blockchain protocols. This is a young but fast paced sub-field of distributed consensus, with new propositions seemingly everyday. New ideas are turned into new blockchains or cryptocurrencies while the formal assessment is lagging behind. These ideas often completely disregard research and good practices from classical consensus protocols, as discussed in [CV17]. In such an environment, finding where the real innovation or breakthrough lies is hard.

There is one metric that crystallizes the whole blockchain scalability race : the throughput. It is a very marketing-friendly number that summarizes protocols in sentences like "we are 10× more scalable than Bitcoin". The problem is twofold : first the theoretic throughput does not guarantee the implementation results and secondly, benchmarks on the implementation are not thoroughly documented and cannot be reproduced independently. Blockchains are immutable distributed ledgers and, as such, we have a trade-off, a trilemma between scalability, decentralization and security [CDE+16, KKJG+18]. Increasing the throughput, for example, should not come at the expense of security or decentralization.

How to assess a new proposition ? Formal analysis should be the first step with clear goals, assumptions, security models and proofs, like cryptographic protocols propositions [CV17]. But the implementations should also be tested. Indeed, they are supposed to support live networks, whether in permissioned or permissionless

setting, and back the claims of the authors, often expressed in a "whitepaper". We should get the data from a live (or real) network (*mainnet*), such as Bitcoin or Ethereum, if there is any, or through a simulated environment (*testnet*) that mimics the live network parameters.

We explore the comparison of blockchain protocols, with a focus on the scalability axis. We aim to provide a common ground to compare the different protocols and pinpoint where the bottlenecks are. This is a first step toward a comprehensive comparison of the various blockchain protocols. Such comparison requires to have reproducible benchmarks results with thoroughly documented testing environments and scenarios.

Our contributions are three fold :

- Defining metrics to evaluate the performance of a blockchain (Chapter 3)
- Designing the architecture and building a large scale prototype to run and measure blockchain performance (Chapter 4). Experimental results corresponding to this contribution are presented in Chapter 5.
- Propose a new protocol to exchange currencies between two different blockchains (Chapter 6) (presented in [ZDBN19])

Table des matières

Introduction

0.1 Context

Blockchains and cryptocurrencies are this strange mix of cryptography, distributed systems, game theory and finance that captured minds and imagination for the last decade. They solve one central problem in distributed computer science : they enable peers to achieve consensus in a decentralized setting, i.e. without granting explicit permission to be part of the network. Before the advent of cryptocurrencies, transferring money was not possible, on the Internet, in a peer-to-peer manner. It required using the services of intermediaries, companies like banks or fintech.

Bitcoin [Nak08] introduces a way, using its blockchain, to do *peer-to-peer*(P2P) exchanges of value over the Internet. Alice only needs Bob's address to send him money. Blockchains, the underlying structure of Bitcoin and other cryptocurrencies, have three important properties : they are secure, decentralized and transparent in the sense that anyone can join, participate, verify or transact. The main, and arguably most successful, use case is as building blocks (!) of cryptocurrencies with an overall market capitalization over $2 Trillion. [1]

Their main use case revolves around financial services and thus it is a natural question to ask if blockchains would be suited for mobile payment. One method is to directly use cryptocurrencies. Bitcoin, and its code forks[2], is the main implementation used and represents half of the entire market capitalization of the crypto-space. Yet it currently processes, on average, 3 transactions per second (tx/s) with a hard limit around 11 tx/s(see section 4.2.1). This is rather low if we want to use it as the base for payments. One central question of blockchains is the possibility to scale them enough to sustain the load required for near-instant mobile payment compared to more established and mainstream services. One such service is Orange

[1] https://www.coingecko.com/fr/global_charts as of 26/04/2021.
[2] Not to be confused with the blockchain forks.

Money, a mobile payment system targeted at emerging countries and created by the telecommunications company Orange, who sponsored this PhD.

Twelve years ago, Orange rolled out its innovative product in Africa : mobile money. With a digital wallet to access financial services from a feature phone or a smart phone, Orange Money represents financial inclusion for those largely unbanked populations. This service brings instantaneity, security and traceability of transactions to retail users, enterprises and, last but not least, governments. Orange Money is available in 17 countries of Africa and Middle East with more than 18 million Active Users [3]. As of September 2019, the value of the transactions was 43 Billion €, for a volume of 2.65 Billion transactions and 425 million in sales revenues. Orange Money became one of the main Mobile Money providers in the world and now has the goal to become a full-fledged bank, with services such as savings account and loans. Another line of extension is the insurance business which requires different licences and compliance rules.

The subject of this PhD is to assess if blockchains could play a role in this *bank the unbanked* vision. Specifically, we question if their performances can scale for mobile payments. The scalability of those systems is often expressed in a single metric : the *throughput*, the number of transactions processed per second. This metric serves as comparison to existing mainstream financial solutions. One example is Visa with thousands of transactions per second on average and 50K tx/s maximum. Another example is Apple, which processes about 5% of global card transactions [4]. It had "a run rate exceeding 15 billion transactions a year"[5] that translates to just under 500 tx/s.

It is a mistake to summarize blockchain-based systems to this unique assessment. As noted earlier, they enable P2P exchange of value in a distributed and censorship resistant manner across thousands of participants, which was not possible before the introduction of Bitcoin. They are public and transparent, i.e. anyone can join. They are *permissionless*, without gate-keeping, in opposition to traditional consensus which are consortium based(*permissionned*). This gave an apparent trade-off between scalability, decentralization and security, which has been dubbed the *blockchain trilemma*.

We propose in this study to assess the performance of blockchains with mobile payment in mind. Our contributions are three fold :

- Defining metrics to evaluate the performance of a blockchain (Chapter 3)

[3] User that transacts at least once a month.

[4] https://qz.com/1799912/apple-pay-on-pace-to-account-for-10-percent-of-global-card-transactior

[5] Apple earning transcript, https://www.imore.com/apple-earnings-q1-20

- Designing the architecture and building a large scale prototype to run and measure blockchain performance (Chapter 4). Experimental results corresponding to this contribution are presented in Chapter 5.

- Propose a new protocol to exchange currencies between two different blockchains (Chapter 6) (presented in [ZDBN19])

The outline of the document is as follows. Chapter 1 introduces blockchains and consensus. It succinctly presents traditional protocols before diving into blockchain building blocks, the cryptography tools, the economic incentives and the network setting. Chapter 2 presents the state of the art on blockchain comparison with a focus on performance evaluation. Chapter 3 defines the metrics we will use to assess that said performance. Chapter 4 shows our evaluation platforms and the rationales behind our choices. In chapter 5, we exploit the results from the various scenarios run. Chapter 6 studies the problem of cross chain communication and devises a new protocol to enable *trustless* communications between a blockchain with smart contracts capabilities and one without.

The rest of this chapter is organized as follows : section 0.2 tells a story to explain the core of the blockchain in an "Explain like I'm 5 (ELI5)" style ; then section 0.3 presents the emergence of mobile money ; finally section 0.4 gives a bird's-eye view of the current trends of use cases in the Decentralized Finance (DeFi) subspace.

0.2 A bed time story of the blockchain

Mummy, Daddy, what is a blockchain ?

Listen, Kid ! Once upon a time, in a far away land call the Internet, villagers were oppressed. They could not transact without obliging to the will of the Intermediaries, an organization that, in exchange for their services, requires fees and "losing some control".

Then comes a mysterious warrior, or group of warriors, no one really knows for sure, called Satoshi Nakamoto. He brought a magic book with properties unheard of :

- anyone could get a copy of this book, they only had to ask ;

- when a new line was added on one copy, it would appear on all other copies ;

- No sentence, no word, no letter, not a single drop of ink could be removed.

But, who could add a line ? Anyone ?

No. Only the initiated Scribes could. They would take the words of the people and transcribe them. This was a long and arduous task for the Scribes, involving a lot of guesswork to devise the perfect incantation to lay the word on the paper. People only had to whisper their words to one Scribe or as many as they wished to have their line eventually added.

I want to be a Scribe when I grow up.

Well, you will have to choose carefully which Scribe you want to become. Because some of them use too much Fire and it is not clear how they will fare in the future.

What did the villagers use the book for ?

They used it first to transact and record those transactions in the magical book : "Alice the Artist pays 2 coins of gold to Bob the Baker". "Bob pays 1 coin from the previous transaction to Fatou the farmer for the wheat" What would you have done with that book ?

I would have written that I love my parents a lot.

Aw, thanks sweetie.

I would also write about myself, my friends. And the promise grandpa made.

Remember you have to consider what you want to write. Nothing added can be modified. Speaking about promises, this quickly became the other important usage of the book. The villagers could write magical contracts in the book. "Fatou the Farmer will provide wheat all year long to Bob the Baker. In exchange, she will get half of all the coins he earns." "Whoever tells me Charlie the Chancellor's secret receive 10 coins"

This is story of the blockchain and is still being written. Who knows which use cases will be created and which one will not stand the test of time. Maybe you will pen the next chapter ?

0.3 Mobile money

The populations of emerging countries are leapfrogging the traditional banking systems, with deposit account and checks, to mobile money. It is a financial service usually offered by a Mobile Network Operator in developing countries [FUN, SCD+]. Some examples are M-Pesa with Safaricom and Vodafone, Wechat with Tencent , Alipay from Alibaba, Orange Money from Orange. It serendipitously come from the fact that users typically had to pay for mobile airtime upfront, which they could exchange between them. It was therefore possible to build a payment system around this.

This movement resulted from a high penetration of mobile phones. It was very easy to onboard new users, capitalising on network effect with the adoption of mobile phones. It created a cheaper and accessible alternative to the banking system, as we can see in this comparison of bank and mobile money accounts in A.1. It is secure and accessible with well distributed local agents enabling cash-in, toping up your account, and cash-out, withdrawing your cash.

The growth of Mobile Money in the last decade is astonishing and the trend does not show signs of slowing down (see Table A.1). Mobile money plays, more and more, an important part of those economies (See Figure A.2). The natural evolution of those providers is to continue disrupting banking and fintech incumbents. They are starting to propose services like remittance, savings accounts, lending and also insurance. They have a unique place to offer an all-encompassing financial solution to their clients.

0.4 Blockchain, DeFi, Fintech

We here give an overview of financial innovation/protocols that originate from the crypto space, from P2P exchanges of value, smart contracts and blockchain adaptation of services in traditional finance.

ICO : Initial coin offering Those were a blockchain adaptation of IPO (Initial Public Offer), hence the name. This is for e.g. how Ethereum were funded, using bitcoins in 2015(check). The development of Ethereum blockchain and its smart contracts in 2017, especially with the ERC-20 standard, a model for fungible tokens, initiated a "cambrian explosion" of ICOs. It had never been this easy to raise funds : a whitepaper or an idea was often all that was needed. This created a set of bad incentives for projects to launch tokens without clear use cases, and for scammers to partake.

DEX : decentralized exchanges They let users trade using a Dapp (Decentralized application) as sole intermediary. In traditionnal finance, the usual model is to use an order book, where traders and market makers post buy and sell orders. This is hard to replicate on blockchain because of the storage of data and the limited speed. The other model that emerges was pionnered by uniswap, the Automated Market maker model (AMM). Each pool, representing a pair of token x and token y, has a parameter k that remains constant and is governed by a equation such as (in the simplest and earliest version) $k = xy$. Each trade moves the quantities of the tokens, thus also the price. The advantage is that it offers instant and available liquidity for traders to increase one asset size in the pool in exchange of the other. But they can

incur slippage in the execution price depending on the size of the trade. There is also a risk of Impermanent loss, a difference between liquidity provision with trading fees in the pool and just "HODLing" the coins (holding a crypto currency for profit). Despite this AMM became very popular which was extended to CFMM with other protocols like Uniswap v3, balancer, curve for stablecoins etc.

Lending/saving

Nowadays, banks use a ever growing collection of data to understand the risk profile of a client to provide him or her greater lending capacity. This is not possible in public blockchains because of the pseudonymity and the lack of (real life) identification. There is no recourse to retrieve your funds if you lend to some random person online. The solution : *over-collateralization*. A borrower needs to put more crypto-asset than the value she wants to borrow. When that collateral goes below a threshold, it is sold and the proceed are used to repay the loan, hence the need of over-collateralization to mitigate the volatility of crypto-assets. The biggest decentralized lending protocols are Aave and Compound.

A special case of loan are flashloans. Those are loans that harness the atomicity of blockchains. Since most blockchains are executed in sequence (except Solana or those with some sort of sharding), when you transact, your are the Queen/King of that chain during that transaction. Except if your block gets discarded, there is no way to stop you. Enter flashloan which lets you borrow an unlimited amount of funds under the only conditions that they are repayed with interest at the end of that transaction. If not, the transaction fails and it is as if nothing happened. Many usages become popular : liquidation of loans positions and repay with the profit ; arbitrages ; migration of positions from one provider to the other. But the most mediatic one was their use to launch an economic attack. Using those funds, one could manipulate the price on an AMM whose price is used as oracle by the targeted service, then buy cheaply at the victim or trigger some events to be able to drain the funds [QZLG21].

Stablecoins Cryptocurrencies are renowned for their volatility. There was a need for coins of stable value called stablecoins. The first iteration are centralized ones. Here we rely on a Trusted Third Party to back the tokens by actual assets in a bank. Examples are USDT by Tether, USDC, Paxos, Libra. The main issue is the trustoworthiness of the backer. Tether is the main one by size but also the cryptocurrencies with the most volume, yet is famous for its lack of transaparency which raises a lot of questions, with regulators in particular. The other type is decentralized stablecoin. It is a special case of lending : users lock cryptoassets and can then mint some stablecoins, usually pegged to the dollar. The main example is DAI, made by Maker Dao. But it had trouble maintaining its pegged during black

thursday and added more centralized stablecoins to back it[6]. A new iteration to create in some fashion under-collateralized stablecoins, dubbed algo(rithms) stablecoins : FEI, FRAX, UST. They try, using game theory and mechanism design, to create a sustainable protocol not susceptible to death spirale.

CBDC The developpement of cryptocurrency with stablecoins and private moneys like Facebook's Libra, pushed Central Banks to push to (re)gain the next evolution of cash : digital cash. The goal is to prepare for a future where cash has a minor role or disappears altogether. The reflexions are early. Among those, a central one for economists is whether or not commercial banks will retain their ability to create money if users have bank accounts at the central bank. Another one is if digital cash will provide some sort of privacy, on par with physical cash.

Facebook's Libra In 2019, Facebook presented a grand endaviour(entrée) un the mobile payment ecosystemn. the project was interesting in 3 regards :

1. create a new currency that would be of stable value, supported by a basket of the main currencies (like the IMF SDR)

2. use Distributed Ledger Technology thus creating a stablecoin, led by a consortium of 100 companies called such that Facebook would be just one member.

3. target population with low banking coverage, *banking the unbanked*.

The last goal was similqar to other mobile payment products in the African continent such as Orange Money. This project spearhead by Facebook ran into a certain number of problems. It was not clear how Libra would tackle the problem of low (national) Identification. It would need to have KYC/AML/CFT procedures in place but that will be hard considering that most of the 1 billion without official proof of identity lives in Subsaharan Africa & South Asia.

The biggest hurdle is that Facebook vaslty underestimated the reception of Libra by the regulators, with geopolitic ramifications. In the United States, the main responses were centered about China perceived oversight and/or control in the system. A recurring question during the multiple hearings in the Congress and Senate were on the presence of the Chinese Yuan in the basket backing Libra. More Broadly, the G7 had issue with Facebook's control of money :"Regarding systemic concerns, Ministers and Governors agreed that projects such as Libra may affect monetary sovereignty and the functioning of the international monetary system."

Fast forward to the end of 2021, Libra rebranded to Diem, David Marcus left Facebook and Diem has not launched yet.

[6]Current value as of feb 2022 is 1 dollar which is its target stable value

Chapitre 1

Consensus and Blockchains

This chapter presents the building blocks of blockchains consensus protocols.

Contents

1.1 Introduction

"Never go to sea with two chronometers ; take one or three."[1] The problem of consensus is fundamental in the distributed computing field and the previous quote summarizes its difficulty. Take only one chronometer to the sea and you have a single point of failure. If it becomes faulty, imprecise, the marines would have no way of knowing it and deciding their location. Taking two instead of one brings another difficulty : how do we decide which one is right when there is divergence in the time we see on both chronometers ? The odds are rather low that

[1] "A marine chronometer is a precision timepiece that is carried on a ship and employed in the determination of the ship's position by celestial navigation." Wikipedia

they will both be affected in the same way. One easy solution is thus to take three and hope that only one breaks.

This is an illustration of *the byzantine generals problem* [LSP82]. We see in the previous example that many parts come into play in a consensus. We can have different number and type of participants (1,2, 3,...). We can assume different type of *divergences*. In the following sections, we introduce consensus from distributed systems point of view before focusing on blockchains.

1.2 Consensus (BFT) before blockchain

In a distributed networks composed of different and interconnected nodes, consensus is the ability of those actors to coordinate and arrive to a shared decision. A set of distributed process need to come to *agreement* on a valid result (*validity*) from the same set of parameters in finite time(*termination*) [Wat16]. First those network can have different types of actors, such as clients and servers, or all nodes can be equal, like processes. Second those nodes can be connected in a decentralized manner to all or some nodes, or in a centralized one to very few other nodes. To arrive to that shared decision, those nodes need to exchanges messages using those network connexions. They can rely on various timing models[2] :

1. asynchrony : there is no upper bound on the delay between emission and the reception of messages. Users do not know *a priori* when the messages will be delivered.

2. synchrony : the exchange of data between the processes or participants are bounded by time limit known to everyone. Here delayed messages are the same as crashes of nodes.

3. partial synchrony [DLS88] : there is an upper bound on the network latency but it is unknown to the participants. Thus an attacker can not decide how long to delay a message. This also creates practical issues for implementing protocols in this model.

Two types of faults are commonly considered : crashes, and byzantine.

- crashes, bit flips, value changes, time-outs represent restricted attacks.

- adversarial nodes behave arbitraly. Attacks may contain a mix of all the above or Deny of Service attacks. These types of attacks are considered unrestricted.

[2]See "Synchrony, Asynchrony and Partial synchrony" https://decentralizedthoughts.github.io/2019-06-01-2019-5-31-models/

1.2.1 Impossibility results

Achieving a consensus in a distributed setting is a problem that has been considered in the literature, with the well known CAP theorem [Bre00, FB99, GL02]. Indeed, a distributedconsensus needs to achieve three goals :

1. Consistency(*safety*) : nodes agree on the current state
2. Availability(*liveness*) : new requests are processed
3. Partition Tolerance : the system continue to work even with network partition

Theorem 1.2.1 (2000 ; Brewer)**.** *It is impossible for a system to simultaneously provide Consistency, Availability and Partition tolerance.*

Previously, [FLP85] had achieved a similar impossibility[3] result, called FLP theorem [Wat16, Her16] :

Theorem 1.2.2 (1985 ; Fischer, Lynch, Paterson)**.** *There is no deterministic algorithm which always achieves consensus in the asynchronous model , with even one faulty node.*

Theorem 1.2.3 (1980 ; Pease, Shostak, Lamport)**.** *no byzantine aggreement with f>n/3 see page page 44, theorem 4.3*

Thus providing a distributed consensus, the whole Blockchain promise, is not something that is theoretically possible in a general context. Let us see what was proposed.

To work around those imposibility results, could rely on a variety of techniques such as different timing assumptions like [DLS88], randomized (probabilistic) consensus like instead of determinist ones like PAxos such as

1.3 Blockchains : building blocks

In a decentralized consensus, we cannot rely on a third party, trusted or not, to have more capabilities than the rest of the network, even for a limited time. This rules out leader based Byzantine Fault Tolerant (BFT) protocols for example. Thus data sharing in the network need to be peer to peer. Users need an objective way to verify all the history of the transactions : they can assert on their own which *tx* is on the common set. Once there is the agreement, it should not be possible, i.e should be very hard under our assumptions, to modify it.

[3] similar but n,ot equivalent https://www.the-paper-trail.org/post/2012-03-25-flp-and-cap-arent-the-same-thing/

1.3.1 Gossip protocols

Gossip protocols [DGH+87] are important to study the propagation or contamination from from one user to X others. Those users will then contaminate X users and so on.Eventually, all users get contaminated.

1.3.2 Rewards

Financial incentivization is not a usual design choice to achieve security in computer science. It fits more a game theory framework where actors either maximize their gains or minimize their losses, provided they are rational. The difficulty is to reward each honest participant s with something, or more exactly, something ”they” value. This could be for instance actions for the common good, points in some scoring system or in newly created bitcoins in the case of Bitcoin. the threat model is thus that a rational player will act to maximize her bitcoins, or fiat equivalent, rewards. This does not capture other reasons for engaging such as altruistic motivations.

1.3.3 Cryptographic functions

They serve as real life implementation of Random Oracles in cryptographic protocols. They have three properties :

1. Pre-image resistance :
2. Second pre-image resistance :
3. Collision resistance :

In a blockchain, these functions are in particular going to be used to protect the “history” already written by the blockchain.

1.3.4 Merkle trees

Blockchains are not only chain of blocks but also trees of hashes. They give guarantees on the integrity of the date stored in the ledger and to do so use a data structure called Merkle trees. Each piece of data, in our case a transaction, is first hashed using a secure cryptographic hash function and then put on a leave of the tree.Each parent is then the hash of the children hashes concatenated :

$$hash_{xy} = hash_x || hash_y$$

and so forth in the tree until we get a hash root.

Cost of operation : $O(\log(n))$ where n is the number of chunk we divided the data in.

1.3.5 Digital signature

Asymmetrical cryptography introduced a paradigm shift on the requirement to use the same key for encryption and decryption. We have a public key and a private key. The public key is akin to a postal address that anyone can use to send the owner a postcard. Only this owner can open the box and retrieve the card using the private key. Another use case is digital signature. A user can sign a message using the private key like a king seal, and anyone with the public key can verify the signature like that king seal recognizable through the kingdom.

1.3.6 POW

Proof-Of-Work[DN92, JJ99, B+02, Nak08] is what enables a decentralized distributed consensus. Users have a *trustless* way to assess which chains they should use. They are based on cryptographic hash functions and using their properties guarantee that they are hard to generate but very easy to verify. To cite [DN92] : "*The main idea is to require a user to compute a moderately hard, but not intractable, function in order to gain access to the resource, thus preventing frivolous use.*" Here is a description of how they are work. Let us take the sentence "Merci Satoshi Nakamoto". It has a sha256 of 0xdd498dfbb4a5717fa197a077cc4cca8812ecd280fabe2f6e7e83924d0ef43545. Say we want to put a constraint on the hash, a requirement that our hash needs to have. For example we can require the hash to start with a certain number of zeroes in the hexadecimal form. We need to change our message. To do so, we append a number i, on which we can iterate until we get *lucky*. By construction, each hex character should have $\frac{1}{16}$ to appear on each character of the hash string. This gives us an expected number of trials before first occurrence of 16. What about two zeroes ?

$16^2 = 256$. To find 4 leading zeroes, we run our script to $16^4 = 65536$.

$$\text{"Merci Satoshi Nakamoto 1"} \longrightarrow^h \text{0x7a95964d92...}$$
$$\text{"Merci Satoshi Nakamoto 2"} \longrightarrow^h \text{0xe96414d740...}$$
$$\text{"Merci Satoshi Nakamoto 3"} \longrightarrow^h \text{0xc9c37d9617...}$$
$$\text{"Merci Satoshi Nakamoto 4"} \longrightarrow^h \text{0xff3cd1b4b0...}$$
$$\text{"Merci Satoshi Nakamoto 5"} \longrightarrow^h \text{0xe9ae13b53b...}$$
$$\text{"Merci Satoshi Nakamoto 6"} \longrightarrow^h \text{0x2cb27b8325...}$$
$$\text{"Merci Satoshi Nakamoto 7"} \longrightarrow^h \text{0x929b188291...}$$
$$\text{"Merci Satoshi Nakamoto 8"} \longrightarrow^h \text{0x77b6c271d3...}$$
$$\text{"Merci Satoshi Nakamoto 9"} \longrightarrow^h \text{0x06d11f0d88...}$$
$$\text{"Merci Satoshi Nakamoto 61"} \longrightarrow^h \text{0x0089d47f94...}$$
$$\text{"Merci Satoshi Nakamoto 4913"} \longrightarrow^h \text{0x000b2ea160...}$$
$$\text{"Merci Satoshi Nakamoto 25090"} \longrightarrow^h \text{0x00009b09ff...}$$

The more constraints we put on the hash we expect, the more difficult it is to find one that satisfies them. We call those constraints the difficulty (number od leading zeroes) which are inversely proportional to the number of hash trials we expect to do. This is the only method to generate such hash, assuming the cryptographic hash function chosen is secure. PoW was proposed as a mean to solve the problem of junk mail. Each user is required to produce a small proof-of-work that accompany the mail. The recipient verify (compute) the hash of the proof satisfy difficulty asked before opening the mail. A spammer would need to compute the POW for all the junk mails which can turn out to be expensive since he or she has to redo the work for each mail.

Proof-of-work have the following properties :

1. the underlying function f is easy to compute and secure such that it is hard to generate a POW but easy to verify

2. there is no POW without work : there is no means to produce a POW other than *mining*.

1.4 Bitcoin

In 2008, an individual or group, under the pseudonym *Satoshi Nakamoto*, published the Bitcoin whitepaper [Nak08] on the cypherpunks mailing list[Nak]. It was a

proposal to build a P2P "currency" that did not rely on a Trusted third party thanks to the use of cryptography, hence cryptocurrency. On January 3rd 2009, the public network was launched the Bitcoin *mainnet*, with an open source implementation. Users download and execute the software to start a node and interact with the blockchain. "Nodes can leave and rejoin the network at will".

Suppose Alice, an avid "bitcoiner", wants to send 0.001 BTC(= 1mBTC= 100,000 satoshis) to Bob for his birthday. First she needs his address, which is the hash of his public key. Assuming a secure digital signature scheme and a secure cryptographic hash function, Bob, with this private key, would be the only one that could spend, i.e. prove through signing, the ownership of the coins. Thus Alice's transaction (tx) contains a signature under her private key.

She broadcasts the tx to her peers. They verify that the signature is valid but also that there were enough satoshis on her address. Once verified, they send the tx to their peers and so on. Eventually, it reaches a node that is mining bitcoins (a miner). In theory, anyone can run be a miner but in practice the mining power required to have a return on investment in a reasonable time frame is huge. Miners put transactions in blocks, structuring them in a Merkle tree, and attach to it a POW, a proof that miners spent energy solving this puzzle. For their work, they get rewarded (with) freshly minted coins, called lock rewards. They also get the transactions fees. There is a constant competition, arms race between miners. Once one of them finde a block, he or she is incentivized to broadcast it to the whole network such that all miners work on the top of it. But forks happen. A fork is when the chain suddenly bifurcates in two (or more) different chains : two (or more) miners find a block for the same height. A node who receives the conflicted block can decide which chain will become the truth. It suffices to compare their POW and take the chain with more work on it, i.e. the heaviest chain. If it is a draw, the nodes wait for a new block that breaks the equilibrium. The network is then secure as long as the majority of the network is honest, which means it is not under the "51%" of having the whole history rewritten.

According to [Nak08]"Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone."

The block with Alice's tx eventually reaches Bob's node(or explorer, exchange...). There is a chance that a longer and heavier chain comes and drops the block with the tx. But each block on top of his block substantially reduces the probability of that happening. This is confirmation, the number of blocks one has to wait to deem a transaction anchored in the ledger. This means that only a miner with a non

negligible amount of mining power can take over the ledger with an alternative chain.

Bitcoin is parametrized to produce a block every 10 minutes on average. The confirmation time was established by[GKW+16] to be 6 blocks. So Bob can spend the received coins after an hour. Taking the whole network hash rate (number of hash per second), the difficulty to find a block is adjusted so that the network produce the expected number of blocks on a 2016 blocks period. This means it should 2 weeks to produce this number of blocks. To accommodate for the miner's arms race, the difficulty increases for the next period if the hashrate was too high or decreases if there was a drop in hashrate.

1.5 Blockchains consensus

1.5.1 PoW

PoW is a consensus that relies on attaching an *easy to verify but hard to generate* work to every addition to the ledger of transactions. the block producers, called miners, mine those blocks using the transactions sent by the users and broadcast them to the rest of the network. In case the users' node are presented with conflicting version of the ledger (forks) , they can objectively decide which chain is the correct one. PoW is secure as long as the majority of the network (PoW) power is honest, which gives the so-called *51% attack*. The idea here is that an attacker enters a race versus the honest share of the network, each trying to extend their respective fork to overtake the other one. This implies that we need to wait many blocks so that the blockchain become prohibitively expensive to overtake. This is colloquially called block confirmation.

Gervais et al. [GKW+16] showed that Bitcoin needs 6 confirmation blocks, about an hour, and Ethereum 40 blocks (10 min) devising optimal strategies for attacks like double spending, selfish mining and eclipse attacks.

Garay et al [GKL15] first proved, on a synchronous setting (instant message propagation), later Pass et. al[PSS17] in an asynchronous one, Nakamoto's PoW[Nak08] achieves consistency and liveness as long as the mining difficulty, hardness to mine a new block, is appropriately set as a function of the maximum delay in the network.

1.5.2 POS

The goal of Proof-of-Stake is to replace the ever increasing energy consumption of PoW while achieving consensus in a decentralized manner. The validators(block

producers) vote using their coin balance on which blocks are added on the ledger. The working assumption is that the majority of the staking power is honest.

There are issues in PoS that are not present in PoW chains. While in PoW, miners have to choose, wisely, on which fork to continue working when there is disagreement, it does not cost more for a validator to support both forks. One fix is to punish (*slashing*)when there is a proof of cheating, for example when there is a validator's signature on two conflicting blocks. Another issue is that it is cheap on a node resources to create a chain of a certain, compared to PoW cumulative difficulty and that in addition an attacker can get hold of an old private with significant stake at some past date[4].

Classical consensus can not be readily applied because of the lack of identity and trust (no Sybil resistance).

Delegated POS is a mirror of mining pools on PoW. The consensus does not have to scale for thousands of potential block producers and can restrict, more or less heavily, their number.

We can look at the protocols in two way : first as round base protocols borrowing ideas from classical consensus ; second as streamline protocols, an idea introduce by bitcoin chain of block.

1.6 Permissioned blockchains

Permissioned blockchains are a marriage of classical consensus, where among other things the number of participants are limited, and cryptocurrencies and blockchain ideas. In permissioned blockchains, the identity of the block producers are know and controlled by a consortium or federation. The original use case was to adapt and apply blockchains to the enterprise setting. The rationale behind is that enterprises want to retain control, ownership and privacy of their operations and information and will be reluctant to transact on a public and permissionless blockchain. But permissioned blockchains have so far little adoption compared to permissionless blockchains. They suffer from the concurrence with databases which is perfectly summarized by Wüst and Gervais in their paper titled "Do you need a Blockchain ?" [WG18]. There is renewed interested with the discussions on Central Bank Digital Currencies(see [Til20]). The most popular solutions are Hyperledger's Fabric and R3's Corda. Hyperledger has a host of solutions that propose different consensus. For instance, Fabric is crash fault tolerant via the use of Raft ordering[5]. Previous version

[4]Some users reported getting inquiries on reddit to buy genesis private keys.

[5]As of , they are working on a BFT ordering service github

uses Solo, Kafka (that uses Zookeeper) and earlier versions had BFT consensus with PBFT.

1.7 Conclusion

In this chapter, we presented the building blocks of blockchains consensus. We show how those parts come together to enable decentralized consensus. In the next chapter will focus on the scalability of blockchains and performance assessment. From there we detail in chapter 3 how those components can impact the overall performance of blockchain and how they shed light on our benchmarking decisions.

Chapitre 2

State of the art of blockchains scalability

In this chapter, we present the ways blockchain can achieve scalability, and discuss how to evaluate and compare their performance.

Contents

2.1 Introduction

"Can decentralized blockchains be scaled up to match the performance of a mainstream payment processor ?

This question was asked by Croman et al. in their position paper on scalability of decentralized blockchains [CDE+16]. Scalability and performance of cryptocurrencies have been a recurring questions, dated from the first responses to Satoshi's announcement[Nak] of Bitcoin. Since then, several contributions have been made to assess and improve the performance of blockchains. We present here the state of the art on those subjects.

We first give an overview of the literature with three "Systematization of Knowledge"(SoK) papers. Those SoK present the state of the research, at the time of publication, on blockchains.

Second, we look into blockchain "re-design". Those are tweaks on cryptocurrencies like Bitcoin to improve the scalability, for example changing the underlying chain of blocks to a tree (DAG).

Then we dive into layered approaches. Taking the blockchain as a base layer, those proposals enable users to have many transactions *off-chain* while sporadically anchoring them *on-chain*.

Finally, we summarize works around benchmarking blockchains, which are the main inspiration for our work on scalability.

2.2 Scalability improvements

2.2.1 SoK

Bonneau et al. [BMC+15] stands as a very good introduction to Bitcoin and cryptocurrencies tools, theory and ecosystem. This SoK was the first systematic study of Bitcoin, bridging the gap between the nascent research (2015) on blockchains and the live implementation which surprisingly for researchers "worked so far". They focus on many parts of the protocol and ecosystem, with a special interest on privacy and disintermediation(decentralization).

Bano et al. present in [BSA+19] a systematic and comprehensive study of consensus protocols. They compare many consensus designs, not just on performance(real or simulated throughput), but also on safety assumptions and permissions through a classical consensus lens. The authors describe protocols based on proof-of-work(PoW), proof-of-X (PoX) protocols that replace PoW with more energy-efficient alternatives and hybrid protocols that are compositions or variations of classical consensus protocols. This SoK captures the boom in proposals for Proof-of-Stake protocols in replacement of PoW. This is interesting in hindsight to compare with the live implementations and the current state of the art on cryptocurrencies and consensus.

In [GMSR+20], Lewis Gudgeon et al. provide an overview of layer-two systems since the inception of cryptocurrencies and identifies the complete set of proposed layer-two protocol types, which are channels, commit-chains and protocols for refereed delegation. It also studies the associated synchronization and routing protocols along with their privacy and security aspects.

2.2.2 Blockchain re-design

The first direction to scale blockchains was forking Bitcoin to create an alternative coin, colloquially called *altcoin*, with different parameters, mainly block time and block size. This did not fundamentally change the blockchain design and we will show later the limitations of this approach. In this section we present approaches that went beyond this re-parametrization. One attempt was to replace the blockchain with a Directed Acyclic Graph. Each block or transaction can reference more than a parent, thus creating a tree instead of a chain.

Sompolinsky and Zohar proposed in [SZ15] an alternative to the longest-chain rule called GHOST, that changes the conflict-resolution procedure for the block chain. GHOST selects at each fork in the chain the heaviest subtree rooted at the fork. A variant of GHOST is the base of uncles blocks in Ethereum that permits much lower block time compared to Bitcoin. They first recall the impact of improving throughput by increasing the block size or reducing the block time : we end up with more forks. GHOST uses those forks that appear in the network to choose the main chain. Even though the content of the forks will not be used, they are Proof of Work that can be counted toward the main chain security.

In [SLZ16], Sompolinsky, Lewenberg and Zohar introduce a DAG to Bitcoin that includes all blocks on the ledger. Using those blocks as votes on previous and recent blocks, hence the DAG, they can order each pair of blocks . The one with the majority's aggregate vote becomes irreversible very fast. This gives a partial order such that conflicting transactions can be rejected from the notarized history.

Li et al. propose in [LLZ+18, LLZ+20] to go a step further than the GHOST protocol and Ethereum's uncles. It uses non conflicting transactions from the forks that naturally occur to increase the throughput. This scheme gives a total order of the transactions from the directed acyclic graph(DAG), instead of a chain, of blocks.

Scaling can also be made using sharding *à la* database scaling. The premise here is that all node do not need to store all the informations, all the state from all accounts or smart contracts. They can keep only the history that is if interest for them. Thus the whole blockchain will be divided into shards that can be maintain in parallel while also allowing cross-shards communication. This is the long term vision of Ethereum 2.0[1].

Omniledger [KJG+18] by Kokoris-Kogias et al. introduces horizontal scaling *á la* database scaling, through sharding while retaining decentralization and security. It introduces several contributions to avoid security pitfalls such as DoS attacks or

[1]In january 2022, Ethereum 2.0 is simply known as ethereum, see https://blog.ethereum.org/2022/01/24/the-great-eth2-renaming/

scalability bottlenecks like cross chard commitments & communication.

Another approach to scalability was to decouple Sybil resistance and transaction serialization. The idea behind this is too use mining to choose eligible nodes to become the elected leader. Then those elected parties can order transactions. In Bitcoin-NG [EGSR16], there is only a "chosen one" that accepts transactions and order them for her tenure. She is replace by the next person to mine a block to become the acting dictator. In Decker et al. [DSW16], the elected parties use a byzantine agreement, namely an adaptation of SGMP [Rei96] and PBFT [CL99], to commit transactions to the shared history. This appraoch is interesting because it shows that Bitcoin bundles many operations into one and that by carefully dividing those steps, one can improve the performance with appropriate new security assumptions.

2.2.3 Layered approaches

The bitcoin lightning network [PD16] describes the main approach to scalability in the Bitcoin protocol using a separate network, a Layer Two, of payment channels. It enables two users to open, update, and close in a trustless manner bi-directionnal channels to do micropayments off-chain with extensive use of Multisig, Timelocks and HTLCs (see 6.2). Only the opening and the closing transactions end up on-chain. Leveraging those channels, one user can find a path to transfer satoshis to another user, thus creating a network of payment channels.

Christian Decker, Roger Wattenhofer present in [DW15a] a duplex micropayment channel protocol, another L2 scaling solution for Bitcoin. It also uses HTLC, Timelocks, and Multisig to create a network of channels like the lightning protocol. The tx are organised in a tree like structure, called the invalidation tree, where the most recent tx have the lowest timelock and hence can be redeem before older tx/state.

Teechain [LNE+19] is a L2 payment network that executes off-chain transactions asynchronously harnessing the power of trusted execution environments (TEEs), to establish off-chain payment channels between parties. It uses committees of TEEs to prevent theft or loss of funds, and remove the requirement to access the underlying blockchain under a bounded time. Teechain can function correctly even in the presence of a compromised or failing subset of TEEs.

Plasma [PB17] is a L2 proposal for smart contract chain that aims to increase blockchain scalability by only publishing Root hashes of the off-chain Merkle tree to the L1 blockchain. The plasma chains can be organized in a Merkle tree hierarchy of parent-children chains. Disputes can be resolved by users on any of the parent chains or directly to the L1 chain.

The Arbitrum [KGC+18] paper describes a L2 scaling solution that supports smart contracts. This is what we now call an optimistic rollup. It uses mechanism design to reach an agreement off-chain on the VM execution. Malicious behavior is penalized by a loss of deposit after a challenge resolved in a multi-round game. And honest parties can advance the VM state on-chain.

Vitalik Buterin discusses in a blog post[2] that presents an overview of scaling blockchains, specifically Ethereum, through Rollups. It gives a quick definition of state channel and plasma scaling, before expanding on the general design of Rollups and the main two flavours : Optimistic and ZK based. It concludes with the current unsolved issues, such as cross-rollup transactions.

2.3 Performance assessment

Croman et al. [CDE+16] analyze bottlenecks in Bitcoin ability to scale to higher throughputs and lower latencies. They look into many parts of the system, among which the network, consensus and storage, to conclude that change of parameters is only the first step and not enough for scalability.

This is an interesting paper that explores how the current network could achieve higher scalability. For example, they measure the impact big blocks could have and propose sustainable new parameters. They also emphasize how new designs are needed and where the research could focus to do so.

2.4 Consensus comparison & evaluation

Gencer *et al.* [GBE+18] focus on decentralization of Bitcoin and Ethereum. For instance, they measure the provisioned bandwidth of the nodes of these *mainnets*. To do so, they try to place enough vantage points to monitor and measure the network. This comes, in a decentralized setting, with a certain level of uncertainty since the experiment nodes will not see all the information that transits through the network. Therefore, we focus on the bandwidth usage of our nodes under the testing conditions.

Other works also focus on measuring the Bitcoin network, mainly [DW13] and [CDE+16]. They try to overcome the uncertainty we just mentioned and measure how information propagates through the network. Decker and Wattenhofer[DW13] introduce a function expressing the ratio of nodes that receive some information. We use this function in section 3.2.3. They link the information propagation with

[2] https://vitalik.ca/general/2021/01/05/rollup.html

the probability of the chain forking and experimentally validate that an increase in information propagation reduces the security of the network. Croman *et al.* [CDE+16] focus on the "effective throughput", defined as the ratio of the block size by the time it takes a block to reach a certain percentage of the network.

Cachin and Vukolić [CV17] study mostly permissioned blockchains. They formally assess the safety and liveness assumptions with regard to crash faults and Byzantine faults. However, they do not examine their implementations and performance.

There are two Systematization of Knowledge (SoK) that are relevant for blockchains comparisons. SoK are works that evaluates existing knowledge on a specific field or subject. Bonneau *et al.* offer an overview of Bitcoin in [BMC+15], as it was the main proposal at that time. Bano *et al.* extend this work to a comprehensive comparison of consensus protocols, blockchain-based consensus and classical consensus, and some implementations in [BSAB+17]. They assess safety and performance but unfortunately do not provide enough details to reproduce their experimental setup and results.

BLOCKBENCH [DWC+17] is a benchmark platform that evaluates private or permissioned blockchains : Ethereum [eth] applied in a consortium context and Hyperledger Fabric [ABB+18]. It measures *throughput* (transaction per second) and *latency* (response time per transaction). In this work, Dinh *et al.* measure changes in throughput and latency when the number of nodes and transactions increases (*scalability*) and during node failures (*fault tolerance*). They conduct these experiments under various scenarios. They do not measure the propagation time of information. As such this is included in the measure of the *latency*. They conclude that blockchains are still not suited for large scale applications.

Gervais *et al.* [GKW+16] analysed *Proof-of-Work* (POW) blockchains implementation to determine the times to finality. They built a platform to run simulated blockchains with various parameters and implemented double-spending [Nak08], selfish mining [ESSN14] and eclipse [GRKC15, HKZG15] attacks. They show how to determine the minimum number of blocks to wait for a transaction to be immutably added on a POW blockchain.

2.5 Conclusion

Here we reviewed the different approaches to improve blockchain performance. We also described how those performances have been assessed so far. This motivates us to explore in chapter 3 and evaluate in chapter 5 how each design decision of blockchain contributes to its performance to devise the meaning and the recipe of

the perfect solution.

Chapitre 3

Blockchains Performance evaluation

This chapter we present our first contribution, which consists in defining interesting metrics in order to evalute blockchain performance.

Contents

3.1 Introduction

Bitcoin was announced on the cryptography mailing list[Nak] in late 2008. The comments rightly highlighted several shortcomings, broadly defined as *scalability*, that the design of Bitcoin has to this day. First was the throughput. Assuming very small transactions, Bitcoin had a maximum throughput of 7 tx/s (11 with Segwit), which is very far from Visa. Second, decentralisation, beyond its usage for resilience, is a real concern for the communities. Any upgrade proposal should minimize the cost for decentralization such that *anyone can join the network*. But the current state of Bitcoin mining favours Big players with ASICs[1], unlike Bitcoin whitepaper "1 CPU 1 vote". Last, proof-of-work energy consumption is ever increasing, which calls for more sustainable yet secure and decentralized consensus.

[1] Application-specific integrated circuit

This has stemmed many proposals from the cryptocurrency and academic communities such has altcoins with Bitcoin parameters tweaks, POW modification, POS, BFT inspired protocols, information propagation (bloxroute, kadcast, Nym ...).

Assessing the performance of those various blockchains and proposals is hard because they have different parameters and trade-offs. Also, such assessment should go beyond throughput comparison. We detail here our model of evaluation, explaining the rational behind each metric we evaluate. From there we construct our hypothesis on the impact of propagation time in the performance of those systems and build the scenarios to validate the model.

3.2 Blockchain and Metrics

Definition 3.2.1. Let tx be a transaction between two or more addresses(users).

Definition 3.2.2. Let B_i be the the block of height i containing $|B_i|$ transactions. B_i also contains a header in the form , where is the hash of the previous block, the POX.

Note that blocks may (cite fruitchain conflux nano) link more than one previous block. We in this case have a Directed Acrylic Graph (DAG) instead of a blockchain[2]. This can also happen at the transactions level (cite iota nano). We view this as a DAG with only one transaction per block.

Definition 3.2.3. Let $\mathcal{B} = \{B_0, B_1, \ldots, B_{n-1}\}$ denote the set of published blocks in the blockchain $\mathbb{B}$.

We use $\mathcal{B}_r^s = \{B_r, B_{r+1}, \ldots, B_s\}$ to represent the set of blocks that happened during that period. Naturally, $|\mathcal{B}_r^s|$ is the number of transactions that are in the blockchain between blocks B_r and B_s.

We denote by t_i the timestamps of the block $B_i \in \mathcal{B}$. We use $\delta(t) = t_{\text{end}} - t_{\text{start}}$ for the duration we run a test or collect data. We stress that, for some metrics, t_{start} and t_{end} do not necessarily correspond to a block timestamp t_i.

Definition 3.2.4. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_m\}$ denote the set of users in the network. We have $|\mathcal{U}| = m$, which is the number of nodes that participate in the network. These nodes can have very sporadic participation during the measurement. They are divided in nodes that produce blocks and nodes that do not.

[2]Though they are still called blockchains/cryptocurrencies

Definition 3.2.5. Let Δ be the function that retrieves the (inter-)Block time of a protocol. We note $\Delta(\mathbb{B})$ the Block time a blockchain $\mathbb{B}$ is set to achieve. For instance, in Bitcoin, this is set to 10 minutes. We use $\Delta(i) = \Delta(B_i) = t_i - t_{i-1}$ for the Block time of the block B_i, except of course for the genesis block(usually $i = 0$).

In the following section, we present the metrics that we found most relevant to assess the performance of blockchain protocols with an emphasis on their implementations. Our goal with this framework is to correctly characterize some properties of the protocols under study. Each metric is divided in the steps that can influence the measures in order to identify the bottlenecks.We also show the parameters each metric is a function of. Then, in a comparison platform, one can measure these metrics by running benchmark scenarios with various set of parameters.

The first set of metrics can be understood as systems features : Block time, time to finality, transactions and block propagation time and throughput. The other set of metrics can be understood as users (nodes) requirements to participate in the protocol : bootstrap time, blockchain size, bandwidth usage and RAM/CPU usage.

3.2.1 Block time

The Block time ($\Delta(\mathbb{B})$) is a constant that the system is designed to attain, but block production rarely happens at a fixed rate. If block production is non deterministic, *i.e.* block production is a random race, there is no guarantee that a block will be produced after a fixed period of time. This is the case in POW and POS but not in DPOS.

In practice, the Block time of a block B_i is a difference of timestamps :

$$\Delta(i) = t_i - t_{i-1} \tag{3.1}$$

This time can be divided into three phases. First, we have a *verification* time during which the producer verify the validy of the transactions and apply the state changes. This assumes that the transactions *already* reached the block producer's node. Then, we have the *POX* time during which the producer's node build a proof, of work, stake or other consensus, that is needed to add the block into the ledger. Finally, we enter *Propagation* time. It represents a delay from the moment the block is broadcast to the time it reaches the other nodes in the network.

Propagation time

This consists of the propagation of the result of the block production. Using the definitions in section 3.2.3, this time is t_{U_j}, with U_j the relevant block producer for $j \in [\![1, n]\!]$.

Verification & State time

This is the phase in which the producer does verifications and state changes. This phase can be divided in the following steps :

$scryT$ slow cryptographic verification (signature, zero knowledge proof)

$fcryT$ fast cryptographic verifications (hashing function)

scT state change. This is the execution of the operations that change the state like smart contracts or coins transfers.

POX time

This is the phase in which the producer aggregates the transactions to make a block and the proof needed. This consists mainly of pfT, the generation time of the Proof-of-X, being hash bruteforce in POW, a signature for POS and some BFT or just a message for other BFT protocols.

Finally we can extend the relation 3.1 to understand what the measure of Block time encompasses, for a certain block B_i (we omit to reference i for conciseness) :

$$\begin{aligned} \Delta(i) &= t_i - t_{i-1} \\ &= scryT + fcryT + scT + pfT + t_{U_j} \end{aligned} \tag{3.2}$$

3.2.2 Time to finality

The time to finality represents the time needed for a transaction to be considered immutably added to the ledger $\mathbb{B}$. It can also be expressed in terms of *confirmations*. We recall that a transaction is confirmed as soon as it is added to a block. The number of confirmations is then the length of the blockchain, counting from the addition of the transaction.

Blockchain protocols, and consensus protocols in general, are constructed in an adversarial setting where attackers have a *nuisance capability/power* denoted α. These protocols guarantee, for the worst attack on their systems with α power, that a transaction needs k blocks to reach finality *i.e.* $k = k(\alpha)$.

$$ttf(\alpha) = \Delta(\mathbb{B}) \cdot k(\alpha) \text{ with } P(\alpha, k(\alpha)) \leq \epsilon \tag{3.3}$$

where P is the probability that the worst attack with α power happens after k blocks and ϵ is some security parameter. Among the blockchains consensus protocols, POW, thanks to Bitcoin[Nak08], has been the most studied consensus. For example,

Gervais *et al.* [GKW+16] simulated various attacks on Bitcoin to recommend $k = 6$ on that blockchain. It is not obvious how to extend this work to the comparison of different consensus, specially with lack of POS protocols formally proven with deployed implementation. Proven and pure POS protocols (Snow White [DPS17], Algorand [GHM+17], Ouroboros [KRDO17], Ouroboros Praos [DGKR18]) rely on various assumptions to mitigate threats like the nothing-at-stake [Poe14] and costless simulation [DPS17]. It is not clear how they hold in live networks. We need to rely on formal analysis of these protocols. As such the previous equation, assuming there is a proof that $P(\alpha, k) \leq \epsilon$, becomes :

$$ttf(\mathbb{B}) = \Delta(\mathbb{B}) \cdot k \tag{3.4}$$

3.2.3 Propagation time

We are interested here in understanding how a transaction or a block propagates through the network, with a special attention to their inclusion in the blockchain. We can for instance track how long a transaction or block takes, respectively, to go from its creator to the block producer that will include that transaction in a block or will produce the next block. We can also track when a transaction or a block reaches a certain portion (*percentile*) of the network, considering all the nodes or just the block producers.

Following the work by Decker and Wattenhofer[DW13], let $I_{U_j}(t)$ be the indicator function whether node U_j knows about a transaction tx or a block B_i at time t. This means that the tx or B_i have been received and verified by the node U_j. Let t_{U_j} be the time elapsed between the creation of tx or B_i and when U_j learns about the transaction or block. Let $I(t)$ be the indicator functions that counts the number of nodes that saw tx or B_i. We recall that $|\mathcal{U}| = m$.

$$I_{U_j}(t) = \begin{cases} 0 & \text{if } t_{U_j} > t \\ 1 & \text{if } t_{U_j} \leq t \end{cases} \tag{3.5}$$

$$I(t) = \sum_{1 \leq j \leq m} I_{U_j}(t) \tag{3.6}$$

Then the ratio of the informed node after time t, that we want to assess through multiple experiments, is :

$$f(t) = \mathbb{E}[I(t)].m^{-1} \tag{3.7}$$

where $\mathbb{E}[I(t)]$ is the expectation of $I(t)$ over time.

3.2.4 Fork rate

We recall that for POW blockchains like Bitcoin, if $\Delta(B_i)$ is the time difference between a block B_i and the previous one, the probability for the network to find a block at time t is [DW13] :

$$P_B = Pr[\Delta(B_i) < t + 1 | \Delta(B_i) \geq t] \tag{3.8}$$

$$\approx \frac{1}{\Delta(\mathbb{B})} \tag{3.9}$$

This expresses with the fact that in POW, miners have more chance to find a block as time passes (they try more block candidates, thus more block hashes) if the difficulty is appropriately adjusted.

Knowing the probability P_B and the distribution of how nodes learn about new blocks($f(t)$), Decker and Wattenhoffer [DW13] expressed the probability of blockchain fork :

$$Pr[Forks \geq 1] = 1 - (1 - P_B)^{\int_0^\infty (1-f(t))dt} \tag{3.10}$$

This comes from the fact that the miners that will produce a conflicting block haven't received yet pending block. They form $(1 - f(t))$ of the network and they were mining for (in *seconds*) :

$$\int_0^\infty (1 - f(t))dt \tag{3.11}$$

Decker and Wattenhoffer [DW13] made the simplifying assumption that the probability of node finding a block is distributed uniformly at random among all nodes, which is adequate for a simulated environment we built 4.

3.2.5 Block Size

Definition 3.2.6. We define $\sigma(B_i)$ as the function that returns the number of transactions in the block B_i :

$$\sigma(B_i) = |B_i| \tag{3.12}$$

$$\sigma(\mathcal{B}_r^s) = \sum_{i=r}^{s} \sigma(B_i) = \sum_{i=r}^{s} |B_i| \tag{3.13}$$

$$\tag{3.14}$$

When comparing different blockchains, σ shows how the usage of those chains evolves in *time*.

To reconcile our definition with the usual focus on the blockchain *storage size*, we add the following definition :

Definition 3.2.7. Let $\sigma(\mathbb{B})$ be the block size limit, if any, of blockchain $\mathbb{B}$. This is a hard cap on the storage size of the block broadcast to the network. We also define $\sigma(tx)$ as a small(in size) yet interesting transaction. For example, in Bitcoin this is a transaction with one input and two outputs and in Ethereum, this would be a transfer of Eth. We have the following relation :

$$\forall i \, , \sigma(B_i) \leq \lfloor \frac{\sigma(\mathbb{B})}{\sigma(tx)} \rfloor \tag{3.15}$$

3.2.6 Throughput

We define the throughput τ as the number of transactions per second (tx/s) the system can sustain. Blockchains support various types of transactions with different sizes (how much data will be stored on the blockchain) or in complexity (how much change is applied to the blockchain state). We need to benchmark different scenarios to have a good picture on what the system can sustain and under which conditions and assumptions. This gives us :

$$\tau = \frac{\sigma(\mathcal{B}_r^s)}{\delta T}. \tag{3.16}$$

where δT is the duration of the experiment or the period of time considered, r is the first block produced once the experiment starts and s is the last block produced before the experiment ends.

Some blockchains, like Bitcoin, have hard-coded limits on the size of a block. This restricts the maximum number of transactions a block producer can fit in a block. To extract a meaningful measure, the maximum throughput, τ_{max}, is computed using the small size of a transaction, $\sigma(tx)$:

$$\tau_{max} = \frac{\sigma(\mathbb{B})}{\sigma(tx) \cdot \Delta(\mathbb{B})} \tag{3.17}$$

Following the work by Croman *et al.* [CDE+16], we can compute the "effective throughput" τ_e defined as the ratio of the block size by the time it takes a block to reach a certain percentage $X\%$ of the network. They use this metric to express a throughput that the system can effectively sustain and they interpret it as the network provisioned bandwidth. Using the ratio function $f(t)$ defined by equation 3.7, we have :

$$\tau_e = \frac{\sigma(\mathbb{B})}{f^{-1}(X\%)} \tag{3.18}$$

3.2.7 Nodes requirements measures

We present here the requirements to participate in a blockchains protocol. One can not design the system requirements for virtual private servers in data centers and expect users with home computers to join the network. The bandwidth, RAM and CPU usages give us a precise idea of the requirements to join the network.

The bootstrap time represents the time it takes for a node to synchronize a blockchain $\mathcal{B}_0^n$, with B_n the last known block. It is closely related to the blockchain size.

3.3 Hypothesis

Here we detail how we use our comparison framework to focus on the impact of propagation in blockchains protocols. Intuitively it is based on what happens between two blocks anchored in the blockchain as shown in section 3.2.1. The first two phases are dependant of the actual consensus while the propagation time depends on how fast the data can hop around the network. We argue that, ideally, the propagation time should be negligible compare to the block time such that improving consensus would directly improve the performance of the protocol. For instance Decker and Wattenhofer [DW13] show that a new Bitcoin block reached most of network in 2012 in 11.37 seconds for average block size of 90KB. They interpret it as a time the network wasted resources on an alternative chain. This time should be as small as possible to decrease the chances of forks.

Blockchains have many parameters that can be tweaked for for improving the overall performance of the chain. Cryptocurrencies, specifically have mainnets with set parameters the chain aims to keep/follow. For example, the difficulty is a function of the mining power to keep the block time constant, or in practical terms, close to target. So blockchains have configurations that is expected to hold during usage.

Our first parameter is thus the block time, $\Delta(\mathbb{B})$, of the blockchain. It represents how frequent we can expect blocks to appear. From a usability point of view, users would prefer lower block time (" the trains are more frequent").

The second parameter is the block size $\sigma(\mathbb{B})$. Most blockchains have a "hard" block size, meaning there is not easy way to change this parameter as it requires a lot of coordination by the block producers (Bitcoin vs Bitcoin cash, Ethereum gas limit increases). From a usability point of view, users would prefer bigger blocks to accomomodate more transactions ("the trains are longer").

The metric that have been used to characterize the blockchain performance (*scalability*) is a ratio of the previous parameters, called the throughput τ. The

expected throughput would be, in an optimal usage, the "effective throughput, which motivates the use of τ_e. Since τ_{max} is the oft cited when speaking of throughput, we also use τ_{max} or τ interchangeably :

$$\tau_e = \frac{\sigma(\mathbb{B})}{\Delta(\mathbb{B})} \text{where } \sigma(\mathbb{B}) \text{ is in MegaBytes(MB) and } \Delta(\mathbb{B}) \text{ in seconds}$$

$$\tau = \tau_{max} = \frac{\sigma(\mathbb{B})}{\Delta(\mathbb{B}) * \sigma(tx)} \text{where } \sigma(tx) \text{ is the size a of small yet interesting } tx \text{ and } \tau_{max} \text{ is in } txs/s$$

Improving scalability was synonymous of increasing τ, as seen with the altcoins trend (Litecoin[Lee11]), Dogecoin[3]) :

$$\tau_e = \frac{bs}{bt}\,;\ \frac{bs\ \nearrow}{bt} = \tau_e\ \nearrow;\ \frac{\sigma(\mathbb{B})}{\Delta(\mathbb{B})\ \searrow} = \tau_e\ \nearrow;\ \frac{\sigma(\mathbb{B})\ \nearrow}{\Delta(\mathbb{B})\ \searrow} = \tau_e\ \nearrow\nearrow$$

The problem is that forks happen when there is disagreement in the common history. This happens more often when the parameters are tweaked beyond what is sustainable by the network as shown by [GKW+16, CDE+16]. For users, forks brings uncertainty and increase the time to wait (confirmation) to consider a transaction anchored in the ledger. Thus to improve the performance of blockchains we want :

$$\text{maximize } \tau \text{ while we minimize the number of forks}$$

From this statement, we derive two questions :

ρ_1 How close to the bandwidth should the expected throughput be ?

ρ_2 How close to the expected propagation should the block time be ?

To understand the idea behind our first statement, we need to look at the expected throughput $\tau_e = \frac{\sigma(\mathbb{B})}{\Delta(\mathbb{B})}$. This is the value we get if the system processes transactions in a constant manner. So the ratio to the bandwidth(noted BW) expresses the load of a blockchain node on the provisioned physical/virtual machine. Ideally, this ratio should be low compared to the average users machine requirements.

$$\rho_1 = \frac{\tau_e}{BW} = \frac{\frac{\sigma(\mathbb{B})}{\Delta(\mathbb{B})}}{BW}$$

The second statement follows the same logic as we look at the $expected\,propagation = \frac{\sigma(\mathbb{B})}{BW}$. An ideal system would, arguably, have a very low expected propagation time compared to the block time. This means most of the time is spent on the consensus (change the state, build the blocks) and the propagation is a negligible part of the protocol.

[3] https://dogecoin.com/

$$\rho_2 = \frac{expected\,propagation}{\Delta(\mathbb{B})} = \frac{\frac{\sigma(\mathbb{B})}{BW}}{\Delta(\mathbb{B})}$$

We are now able to measure how the system behaves compare to what we expect, with ρ_1, ρ_2. Luckily, we notice that :

$$\left.\begin{array}{ll} \rho_1 & = \frac{\tau_e}{BW} = \frac{\frac{\sigma(\mathbb{B})}{\Delta(\mathbb{B})}}{BW} \\ \rho_2 & = \frac{expected\,propagation}{\Delta(\mathbb{B})} = \frac{\frac{\sigma(\mathbb{B})}{BW}}{\Delta(\mathbb{B})} \end{array}\right\} \Longrightarrow \rho = \frac{\sigma(\mathbb{B})}{\Delta(\mathbb{B}) * BW}$$

Thus ρ summarize the two statements. What is missing is the representation of forks of the system as a function of the other parameters. We show here that ρ can be seen an approximation of the function linking propagation to forks. The intuition comes from comparing those estimations with different values of block sizes, times and bandwidth in table 3.1.

TABLE 3.1 – Fork, Throughput and Ratio estimation

bt(s)	16	16	16	16	64	64	64	64
bs(MB)	8	8	32	32	8	8	32	32
bw(Mb/s)	1000	256	1000	256	1000	256	1000	256
bw(MB/s)	125	32	125	32	125	32	125	32
tx/s	2097	2097	8388	8388	524	524	2097	2097
propag.	0,064	0,25	0,256	1	0,064	0,25	0,256	1
$\rho(\%)$	0,4	1,56	1,6	6,25	0,1	0,39	0,4	1,56
fork	0,41	1,60	1,63	6,25	0,10	0,39	0,40	1,56

Using the set parameters of the studied blockchain and the provisioned bandwidth bw of the network, we can rewrite equation 3.10 with the expected propagation :

$$Pr[F \geq 1] = 1 - (1 - \frac{1}{\Delta(\mathbb{B})})^{\frac{\sigma(\mathbb{B})}{BW}} \tag{3.19}$$

$$\approx 1 - (1 - \frac{1}{\Delta(\mathbb{B})}\frac{\sigma(\mathbb{B})}{BW}) \text{ using Taylor serie for } (1+x)^{\alpha} \tag{3.20}$$

$$\approx \frac{\sigma(\mathbb{B})}{\Delta(\mathbb{B}) * BW} = \rho \tag{3.21}$$

We thus use ρ to track forks in our testing scenarios.

3.4 Conclusion

In this chapter, we defined the metrics of interest for blockchains parameters which are summarized in table 3.2. We also explored the impact of propagation in those

systems which gives an indication on the benchmarks to run. In chapter 5, we run the various scenarios to test our hypothesis, using our evaluation platform described in chapter 4.

Table 3.2 – Comparison metrics & parameters

notation	summary
tx	transaction
$\Delta(\mathbb{B})$	block time
$\sigma(\mathbb{B})$	block size
τ	throughput
$f(t)$	ratio of the nodes which saw a block or tx
k	6 number of blocks to reach finality
ttf	time to finality
ρ	ratio that approximates the number of forks

Chapitre 4

Experimentations platform

In this chapter, we present the experimental framework developped during this thesis in order to evaluate blockchain performance. The experimental framework was developped as a partnership between Orange and INSA Centre Val de Loire.

Contents

4.1 Introduction

Before Nakamoto consensus, scalability *concerns* for distributed consensus were different. They focused on reducing the messaging complexity of the protocols, the type of faults or the synchrony assumptions while increasing the number of nodes. Blockchains have weak synchrony(partial synchrony, asynchrony) assumptions, are designed for Byzantine faults, have mainnets with thousands of nodes and use few messages thanks to gossip protocol. Where the problem lies is the performance of such protocols under heavy usage, dubbed the blockchain scalability problem, which has been summarized as the throughput they could sustain in comparison to Visa peak of 50000 tx/s.

We aim to provide a common ground to compare comprehensively the different protocols and have a better understanding of scalability limitations. Such comparison

requires to have benchmarks results with thoroughly documented testing environments and scenarios across as various blockchains and consensus protocols as possible.

For this purpose, we set to build a simulation environment (figure 4.1), with controls over the configuration in expectation of reproducible results, flexible enough to add new blockchains as needed. It enables us to configure parameters that might have an impact on the blockchain scalability. We build two complementary parts that we call system 1 (developed at Orange Labs) and system 2 (developed at INSA) [1].

We first explore different public blockchains to uncover their inner working and devise a fair comparison framework. We settle for a working architecture that harnesses the capability and flexibility of Docker containers for System 1. We use it to build testnets and mainnets configuration of public blockchains : Bitcoin, Ethereum, Qtum and EOS.

The second one is the benchmark architecture, where ultimately all the other blockchains would be tested since it is a controlled and reproducible environment. In system 2, we use the Openstack suite to create, install and manage Virtual Machines (VM) on bare metal servers. We use Multichain[Mul], a blockchain with permisionned and permissionless settings in this architecture to validate our comparison framework.

We have the following steps that we detail in the subsequent sections :

- PREPARE : install and configure the blockchain(s) with the chosen parameters
- EXECUTE : create transactions to be added in the ledger
- EXTRACT : explore the result from the execution step.

4.2 System 1 : how to build a blockchain agnostic platform

The advantages of studying cryptocurrencies is that they have *mainnets*, public network of their blockchains, which give us a set of defaults parameters to compare with. The disadvantages is that, putting Bitcoin forks aside, there are very different, from consensus (POW, GHOST POW, POS, DPOS),chosen programming language, to accounting model (UTXOs vs Account) or scripting/smart contract capabilities. In the following sections we first detail the rationale behind our choice of blockchain protocols to study. We then explain the steps PREPARE, EXECUTE and EXTRACT when studying and benchmarking them.

[1] System 1 & 2 loosely named after Daniel Kahneman, *Thinking, Fast and Slow*

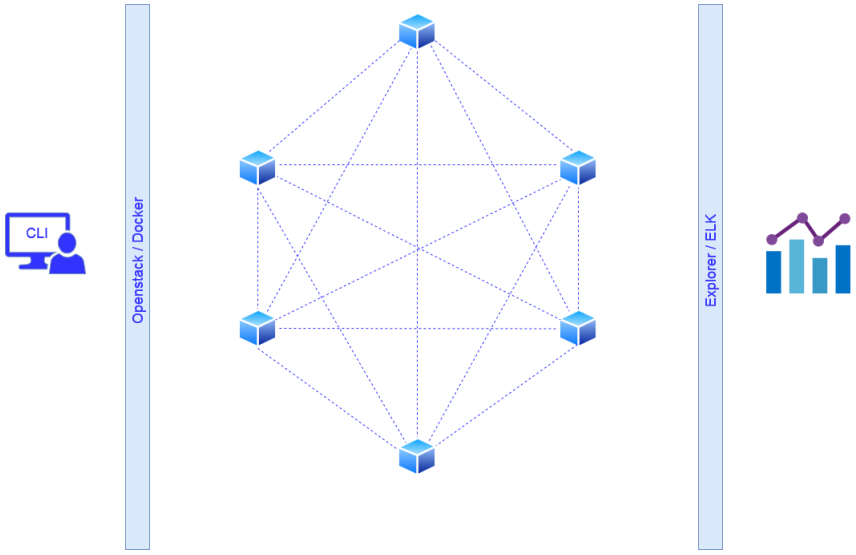

FIGURE 4.1 – High level architecture overview

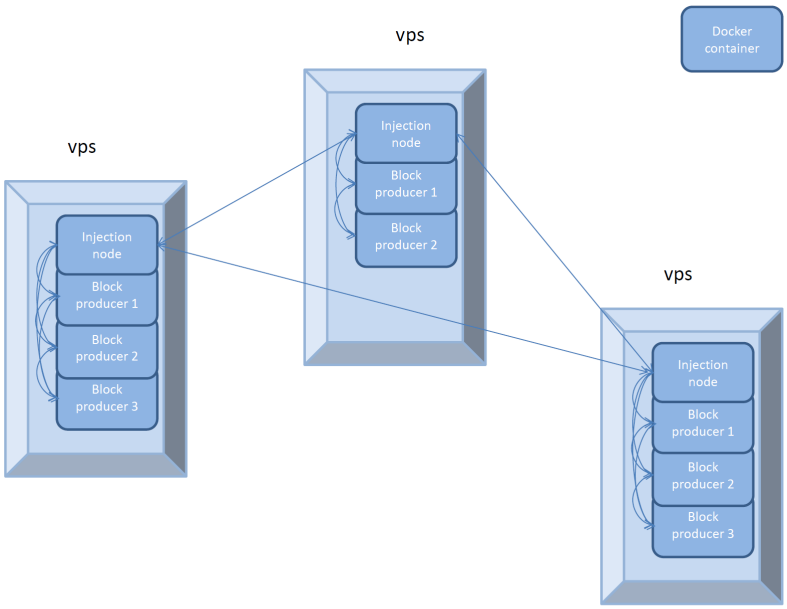

FIGURE 4.2 – System 1

4.2.1 Blockchains choice

We propose to look at a small set of protocols : Bitcoin [Nak08], Ethereum [eth], Qtum [DMEN17] and EOS [eos]. We only choose permissionless protocols, commonly called public blockchains, because they have *mainnets* that are used and battle-tested every day. We can compare how these protocols behave in real setting (*mainnet*) and in simulation setting (*testnet*). We wanted to cover a large spectrum of the field using representative examples. The type of consensus is a natural choice to distinguish these protocols.

Bitcoin is the first consensus protocol that uses a blockchain and has an open source implementation. The majority of the blockchains forked from Bitcoin source code, sharing its advantages and limitations. Ethereum is the first blockchain to enable Turing-complete program executed on the top of its blockchain, called smart contracts. It is not a fork of Bitcoin source code but still uses the POW consensus. To complete our spectrum, we added POS blockchains. To our knowledge, there was no formally proven and *pure* POS proposition with a complete implementation that was live At the time of development. We thus choose Qtum because it supports smart contracts thanks to the Ethereum virtual machine (EVM). We also added EOS which has the particularity to enable smart contracts on the top of a DPOS consensus and is not based on Bitcoin or Ethereum.

The difference in consensus protocols offers different trade-offs but a comparison should also highlight the similarities. Consensus propositions alone will not fix all the blockchain issues and a formal comparison framework can hint where improvements should be made.

We do not cover protocols from classical distributed consensus like PBFT [CL99] and BFT-SMART [BSA14]. We are also missing blockchains that use inspiration from them like Tendermint [BKM18a] and Algorand [GHM+17] or hybrid consensus, for example POS+POW with [DFZ16, CDFZ17, BG17] or POW+BFT with [PS17, DSW16]. We leave this for future work.

4.2.2 Prepare

We first need to get familiar with he solution we are testing. This involves interacting with the mainnets, using online explorer, installing the client, and understanding the logs among other tasks. Then we can build our own testnet, with a special focus on parameters of interest that we need modify for our comparison such as the block time ($\Delta(\mathbb{B})$), generally in the source code but sometimes in a configuration file.

We use Docker to accommodate for different configurations, mainnet or testnet.

TABLE 4.1 – Comparison of blockchains

Points	Bitcoin	Ethereum	Qtum	EOS
consensus	POW	POW	POS	DPOS
Genesis date	Jan. 2009	Jul. 2015	Sept. 2017	Jun. 2018
$\Delta(\mathbb{B})$	600s	15s	120s	0.5s
block size ($weight$) limit	1MB ($4MB$)	dynamic	2MB ($8MB$)	dynamic
τ_{max}	11	15	110	100
k $/ttf$	6 blocks 1h	40 blocks 10min	tbd	252 blocks 126s

4.2.3 Execute

In this step, we want to be able choose which test scenario to run. This can be very simple transfer, Alice sends X coins to Bob, or more complex scenario that use the expressiveness of smart contracts, similarly to scenarios from Tuan et al. in BLOCKBENCH [DWC+17]. Ideally, accounts will be funded in a agnostic manner such that they can send transactions.

So the main part is thus to identify how transferring of coins from A to B in the current blockchain works.It is unclear which complex smart contract can be implemented in Bitcoin in an easy manner while being comparable to the smart contracts on Ethereum, Qtum or EOS. Staking is taken care of in the INSTALL step.

We can then generate many accounts, pre-fund their addresses and *execute* the scenario. We use a PlantUML inspired language to define the test. The translation to the underlying blockchain node RPC is made through an interpreter developed in Scala.

4.2.4 Extract

For System 1, the extraction revolved around *on-chain* data. Each node is configured to open the JSON-RPC(remote procedure calls encoded in JSON) server. Using the APIs provided, we fetch, as much as possible, the same set of data from the chains.

We developed in Go(Golang) mainly in prevision of parallelization with Go routines to explore the blocks. Also there are client implementations for all those blockchains in Go, specially Ethereum with its main client Geth(Go-Ethereum[2]) and

[2] https://github.com/ethereum/go-ethereum

Bitcoin btcd[3] which serves as a base for one of the lightning network implementation (lnd[4]).

4.3 System 2

In the following sections we first detail the rationale behind our choice of blockchain protocol to study. We then explain the steps PREPARE, EXECUTE and EXTRACT when studying and benchmarking them.

Here we focus on one solution, Multichain 5.2, to test various scenarios on blockchains parameter. The choice of Multichain was motivated by the shared source code with Bitcoin (it is a *fork*) and its support for a round robin, hybrid or POW consensus.

We give now a high level overview of our platform, see figure (4.4). It is based on a collection of physical servers that are co-located and connected through a high bandwidth network. We can then run multiple virtual machines(VM) on the top that have the same configuration. One blockchain node is installed per VM and connected to all the other nodes. Once the chain is set up and running, we generate transactions, from all nodes, that are broadcast and eventually added in the ledger. Finally we can extract the results from the execution and compute the previously define metrics using the ELK suite in addition to our explorer.

4.3.1 Multichain

We used Multichain in our testbed for the simplicity to set and modify various parameters of its blockchain, as described in 5.2. In the configuration file, we can choose, for example, the block size limit, the targeted block time. We can also choose the type of consensus by setting the value of $mining - turnover(mt))$ for round-robin(0), hybrid(0.5), Bitcoin PoW (1).

We use one *admin* node whose main role is to administrate the permissions to mine bocks of Multichain to the other nodes.

4.3.2 Prepare

We uses Openstack suite to have, on demand, virtual machines(VM) to host our nodes. It enables us to create our VMs, allocate resources such as virtual CPUs, RAM and storage, to them and configure the network settings. From there, we can

[3] https://github.com/btcsuite/btcd
[4] https://github.com/lightningnetwork/lnd

```
tc qdisc add dev ens3 root netem rate \
 BANDWIDTH_LIMITmbit delay DELAYms
#tc qdisc add dev ens3 root handle 1:0 netem delay DELAYms
#tc qdisc add dev ens3 parent 1:1 handle 10: tbf rate \
 BANDWIDTH_LIMITmbit latency LATENCYms burst 32kbit

#tc qdisc add dev ens3 root handle 1: htb default 12
#tc class add dev ens3 parent 1:1 classid 1:12 htb rate \
BANDWIDTH_LIMITmbit ceil BANDWIDTH_LIMITmbit
#if [ "xLATENCY" != "x" ]
#then
#tc qdisc add dev ens3 parent 1:12 netem delay LATENCYms
#fi
```

[

install and interconnect the Multichain nodes. We can also pause, restart or save the VM at will to explore, test and understand the behaviour of our blockchain. The Openstack tools we use are :

- Horizon
- Nova
- Neutron

Bandwidth and latency control

Cryptocurrencies rely on a patchwork of nodes distributed across the globe, with various hardware configuration. As such, the connexion between peers can vary greatly, from the speed, the reliability or the number of hops in the network. As such we need to test different scenario of connectivity to paint a better picture of the performance of blockchains. For example, Croman et al. [CDE+16] studied the bandwidth usage of the Bitcoin network. They also express how latency impacted data propagation, following works from Gervais [GKW+16] and Decker [DW13]. To reproduce and extend on these works, we needed to run simulations under various parametrization of the network connectivity between nodes. We use Traffic control (**tc**)[5] that gives the ability to simulate packet delay, limit bandwidth or even loss of packets.

[5] https://man7.org/linux/man-pages/man8/tc.8.html

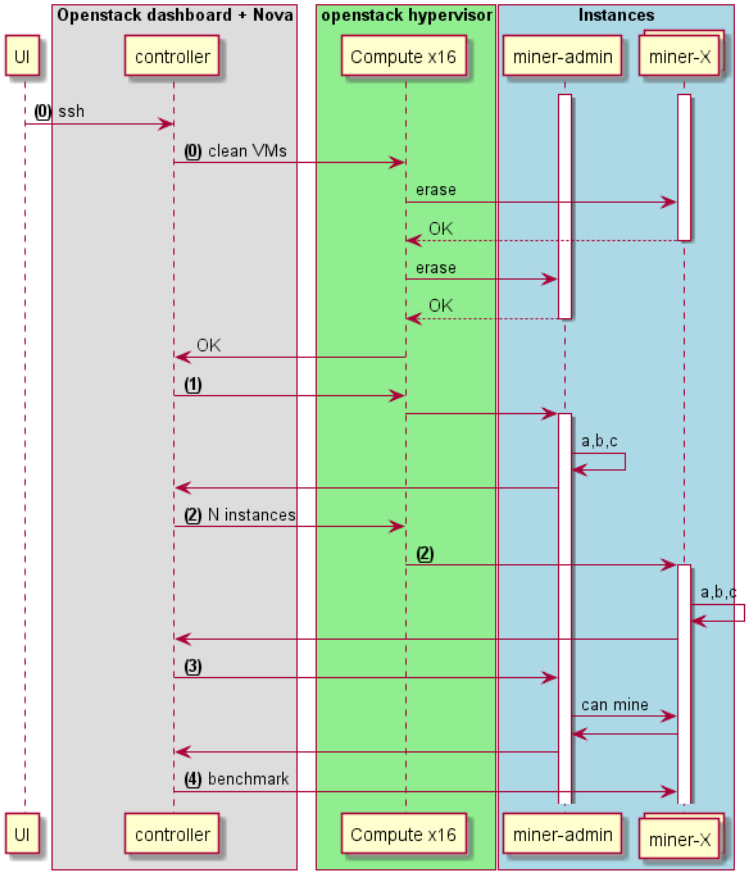

FIGURE 4.3 – Sequence diagram

Sequence diagram

We detail the process to launch a new test configuration, which means launching and configuring the blockchain and the miners.

We first (0) log in the VMs controller to choose the parameters of benchmark. The Virtual Machines, on our park of physical servers, are then erased to make space for the new one.

Next (1), we create the `miner-admin` . This starts by setting up a VM. Following, we install Multichain, set the admin node and start the chain. Finally we create a way to store embed more data in the blockchain (using Multichain flux) in order to have transaction of the targeted size.

Then (2) we create the N instances of miners(`miner-X`). We repeat the same steps as with the `miner-admin` . In addition each miner connects to the admin to join the chain and receive the mining permissions (3).

Finally (4), once all the nodes joined the blockchain and partake in the consensus, we are set to launch our benchmark.

4.3.3 Execute

We want to assess how the protocol performs under pressure, i.e how the transactions are processed in a network under heavy load. We test our configuration under an increasing transactions volume, which also let the network set up calmly before the (transaction) storm. This guarantees that block space are fully used by transactions when the load increases. We use the following script(A) to start sending transactions from each node (except the `miner-admin` node). Each node uses Multichain flux to make an update embeded in a transaction of about 250 bytes. This is the size of a small Bitcoin transaction with one input and two outputs and is the reference when talking about its throughput.

4.3.4 Extract

Since Multichain is a fork of Bitcoin, our *explorer* works out of the box with Multichain for on chain data. This means that after a scenario execution, we can retrieve data from the *then* canonical blockchain.

What is also interesting is to be able to see forks happening, since we use it as a expression of how scalable a protocol is. For this purpose we use the ELK suite. Elastic search is the database that store the data. Logstash performs operations on the logs to retrieve what is of interest. Kibana let us queries the data and make graphical representation.

4.3.5 Architecture

The platform final architecture is described in figure 4.4. Our simulation environment relies on a set of 14 physical servers :

- one X server, with 32GB RAM, 10 CPU cores and HDD
- 3 servers with 32 GB RAM, 10 CPU cores, HDD and GTX1080 (not used for mining Multichain) in addition
- 10 Xeon servers, with 8 to 32 GB RAM and 8 to 12 CPU cores.

All those machines are connected to a local network, provided with a 1 Gbps bandwidth. One machine acts as a controller to provide configurations files and scripts to setup the Virtual Machines. The access is done using the Openstack controller UI(Horizon) through a VPN or using ssh to directly connect to the server. Each VMs, a *compute* in Openstack Nova, is provisioned with a Ubuntu server OS running on 2 vCPUs, 4 Go RAM and 20Go of storage. They are interconnected

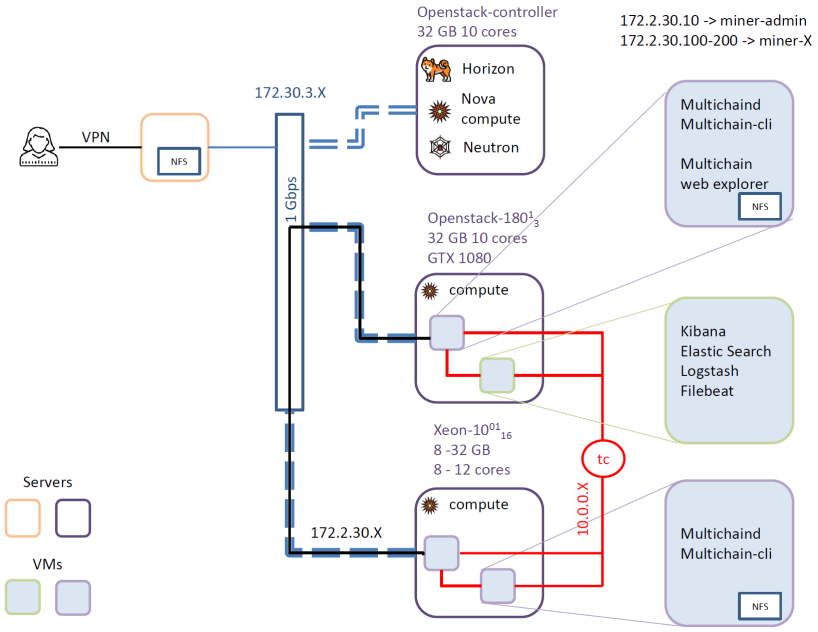

FIGURE 4.4 – System 2 : platform architecture

on the 172.2.30.X network interface, which is also used by the NFS. We put traffic control on another interface, in red, to not obstruct data retrieval through NFS.

4.4 Conclusion

We detailed in this chapter our architecture to test and compare different protocols. We explain the rationales behind the various choices of tools or setting. In the next chapter, we use this platform to benchmark scenarios and extract the metrics define in chapter 3.

Chapitre 5

Experimentations results

This chapter presents the results of extensive experiments run on the infrastructure presented in Chapter 4.

Contents

5.1 Introduction

In this chapter, we present the evaluation of various parameters for consensus and nodes on both platforms to explore some questions around scalability. Due to time constraints, we extensively study with only scenarios for Multichain on System 2.

We first present the Open Source Enterprise blockchain Multichain [Mul] that we use for our tests. We then start with a test to validate our testing infrastructure with default parameters. Finally we focus on different consensus and network setting for a number of participants similar to Orange Money deployment.

5.2 Multichain

We present in this section the various parameters of interest in the Multichain [Mul] protocol. It is a fork of Bitcoin, with modifications to be suitable for enterprise

blockchains. It can run in a permissionless or permissioned setting. We can decide which nodes are able to join the network and which among them have the right to mine blocks. In production, this is the purview of administrator (*admin*) nodes. We capitalize on previous work we have done to develop and improve our System 2 for comparing consensus. Being a fork of Bitcoin, Multichain supports Nakamoto consensus, but also adds round-robin and hybrid style consensus. We can conveniently choose those parameters in the configuration[1] (instead of studying the source code, modifying, compiling and running a new client like we did while working on System 1). Those are :

- targeted block time
- maximum block size
- mining diversity
- mining turnover
- minimum PoW difficulty
- difficulty adjustment frequency

5.2.1 mining-diversity

This parameter represents the minimum proportion of permitted miners required to participate in the round-robin scheme to render a blockchain valid. If we make the simplifying assumption that each miner has the same mining power (same used in section 3.2.3), this can be seen as, loosely, the requirement that a majority of the hashrate mine on the chain. It comes with an additional constraint though, since this is a round-robin scheme, that each miner makes one block and should not try again until the next window. This is implemented[2] in a "stop-go"[3] manner (see code excerpt in Annexe B.2). A miner who just produced a block has to stop mining for some time before rejoining the race. Notice that the resting time is function of the average block time of the last 10 blocks. Thus, if the blockchain has difficulty making progress, because for example there are too many transactions and forks, it will slow down even more.

This parameter must be set between 0.0, which means no constraints, and 1.0, in which case every miner must participate. Default value is 0.8.

[1] https://www.multichain.com/developers/blockchain-parameters/

[2] Last accessed 10/03/2022 at https://github.com/MultiChain/multichain/blob/ca02efcdaa5087f42d85977d322fad1cea58ba48/src/miner/miner.cpp#L1497-L1512

[3] https://www.formula1.com/en/championship/inside-f1/glossary.html

5.2.2 mining-turnover

Mining turnover is, unlike mining-diversity, a recommendation rather than a consensus rule. It adds constraints on the miners with others stepping in only if one fails. This parameter can set a pure round robin scheme (0.0) between an automatically-discovered subset of the addresses with mine permissions. A value of 1.0 prefers pure random block creation and intermediate values give an hybrid consensus between the two behaviours.

Note that for a Nakamoto consensus in which anyone can try to mine a block at any time, there should not be any restriction on the turnover (1.0) and the diversity(0.0). We choose 0.8 over the default 0.5 to allow a margin in our experiments.

5.2.3 difficulty management

The permissioned consensus we studied restrict the nodes that can mine at a given time. This is exacerbated by the emergence of forks, that further split and disturb the mining process. As such, we keep the mining difficulty constant throughout our benchmarks. Our assumption is that, with miners idling period, the block time should stay close to the set parameter in aggregate. When relevant, i.e. for Nakamoto permissionless consensus, we choose a readjustment period of $20\Delta(\mathbb{B})$ while keeping the same starting difficulty. This is to strike a balance between forks resolution if the production of blocks is too fast and blockchain progress if too slow and to accommodate the hybrid and round-robin consensus where not all nodes can find a new block at a time, compared to Nakamoto's consensus.

5.3 Blockchain for e-payment using Multichain

5.3.1 Benchmark

For each scenarios we choose the following parameters :

- number of nodes N
- consensus $(mining\,turnover/diversity(mt, md))$
- block time bt
- block size bs
- network bandwidth bw

First the network is initialized. Per configuration, this means Multichain will take the first 60 blocks (0-59) to initialize. It starts with the `miner-admin` node that gives mining permissions to N nodes to be part of the producers. One consequence is that the `miner-admin` will be the only node producing blocks until the others join in the mining race. We test our configuration under an increasing transactions volume. As such most of the load does not appear while the blockchain is setting up. This guarantees that block space are fully used by transactions when the load increases. We use the following script(See Appendix B) to start sending transactions from each node (except the `miner-admin` node). Once the load subsides, we explore what is then the canonical blockchain on `miner-admin` , assuming most forks are resolved by block 300 and this node is on the right chain.

5.3.2 Number of nodes impact

Our first scenario purpose was to validate the benchmark platform and the extraction of results. We here explore the impact of increasing the number of nodes in a hybrid ($mt = 0.5$) consensus, with the default Multichain parameters $bt = 15s$, $bs = 8MB$ and $bw = 1Gbps$. We tested networks with 5, 10, 15, 20, 25, 30, 40 and 50 nodes. We briefly (see Section 5.2) recall that Multichain round robin consensus imposes the mining production to rotate between the producers such that each of them has a chance to mine a block. For each block, the sliding window of previous N producers constrains who can participate in mining, since the others would get their blocks discarded. Thus in a hybrid setting, half of the block are under the round robin constraints while the other half is under Nakamoto consensus. We expect that it will become more difficult for nodes to achieve consensus as the number of nodes increases. This means more forks should happen and block propagation would take longer.

We present the results for 5, 25 and 50 nodes.

Let us look at one execution of a benchmark. We explore the 5 nodes setting. Per consensus, we have a sliding window of $\lfloor N * mt \rfloor = \lfloor 5 * 2.5 \rfloor = 2$. This means that the block producers of the 2 previous blocks can not mine the current block. Looking at Figure 5.1, we notice that for instance no miner has two blocks in a row. Furthermore, on block 126, `miner-2` made a competing block. This fork would not have become the canonical chain and got discarded.

Let us compare the 3 settings from a high level. Each node added to the network brings more load on the system and for the 5 , 25, 50 nodes setting, we respectively need to process 590500, 2952500 and 5905000 transactions sent. If all of them were created and broadcast at the same time, then added in consecutive full blocks every

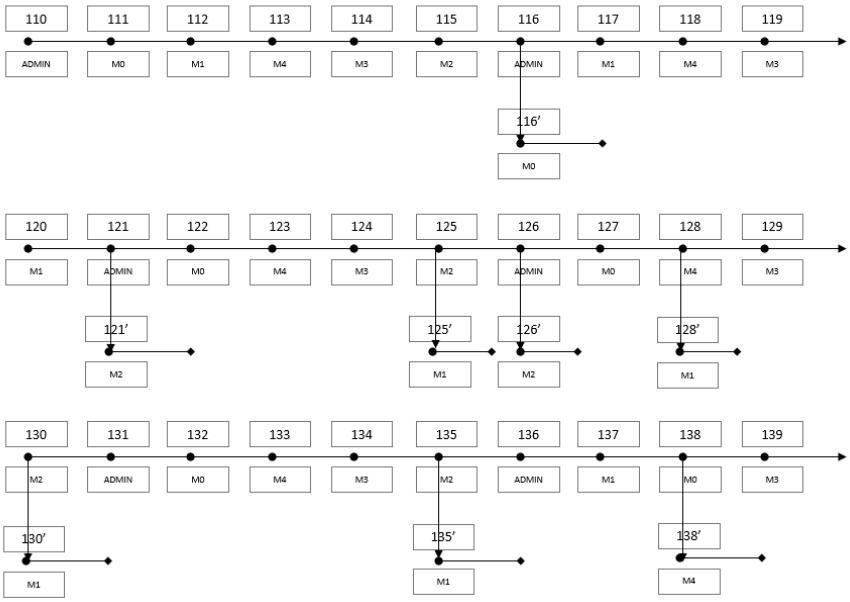

FIGURE 5.1 – Blockchain (5 nodes - default) from block 110 to 140

$bt = 15s$, it would respectively take 4.5, 22 and 44 minutes to process them. Note that we put various breaks in the script(Table A) which amount to about 17 minutes of *sleep*. Taking the *sleep* into account, it would respectively take 21.5, 39 and 61 minutes to clear those tests. Running the experiment with only 5 nodes takes around an hour. We see in Figure 5.2 that the nodes slowly ramp up the transactions generation, as programmed, until they only exchange "alive" status. In comparison 25 nodes, in Figure 5.6, take around 2 days to finish while 50 nodes take more time(Figure 5.10) to clear the load, in about 4 days. We stress that using the logs to study forks, we are exposed to the "quirks" of nodes implementation. Going forward, we focus on the plots with the height to abstract those problems.

Then we plot the total number of transactions in the blockchain which are present in each block, for *Bitcoin Core* based implementation. This useful to see how the increasing number of transactions are included in the blockchain. If blocks would propagate instantaneously and fork be rare, the Max and Min curve would be indistinguishable from each other and rejoin quickly whenever they diverge. The chance that a fork will appear and still have the same number transactions while the blockchain is flooded with transactions is pretty low since all nodes do not have the same *mempool*. Thus, as expected, the network with 5 nodes Figure 5.2 is able to keep up while the bigger networks have a harder time (Figure 5.6, Figure 5.10).

Reading the same data but through block number(**height**), the 5 nodes case

processes its heavy load in less than 20 blocks, between block 120 and block 140. For 25 nodes, 180 to 250. The last scenario shows 50 nodes include most transactions later than in other scenarios, between block 200 and block 290. We study those ranges to compute throughputs, number of forks and propagation of blocks.

TABLE 5.1 – Benchmarks of an increasing number of nodes with hybrid consensus

	5 nodes	25 nodes	50 nodes
date	20-sept	08-avr	30-sept
heure	11 :30	22 :25	17 :55
block time(s)	15	15	15
block size(MB)	8	8	8
bandwidth(Mb/s)	1000	1000	1000
bandwidth(MB/s)	125	125	125
nodes	5	25	50
mining-turnover	0,5	0,5	0,5
proj. propagation	0,064	0,064	0,064
ratio prop/btime(%)	0,426666667	0,426666667	0,426666667
fork estimation	0,44	0,44	0,44
proj. throughput	2236,962133	2236,962133	2236,962133
projected tx	590500	2952500	5905000
proj. N of blocks	18	88	176
proj duration(min)	4,5	22	44
range start	119	179	200
start timestamp	1600597034	1586393309	1601506915
range end	140	250	290
end timestamp	1600597898	1586554144	1601669140
block in range	20	70	89
delta timestamp	864	160835	162225
average blocktime	43,2	2297,642857	1822,752809
tx at start	75966	314292	1947719
tx at end	375834	2780329	4973067
done tx %	63,64674005	94,16863675	84,21790008
number of tx	299868	2466037	3025348
constant btime tx/s	999,56	2348,606667	2266,178277
real tx/s	347	15	19
#forks	7	91	407
#forks in range(%)	35	130	457
#forks 100-199	14	155	294
max propag. (25%)	54	5942	3360
max propag. (50%)	78,499	10746,817	5450
max propag. (75%)	151,996	15991	77239

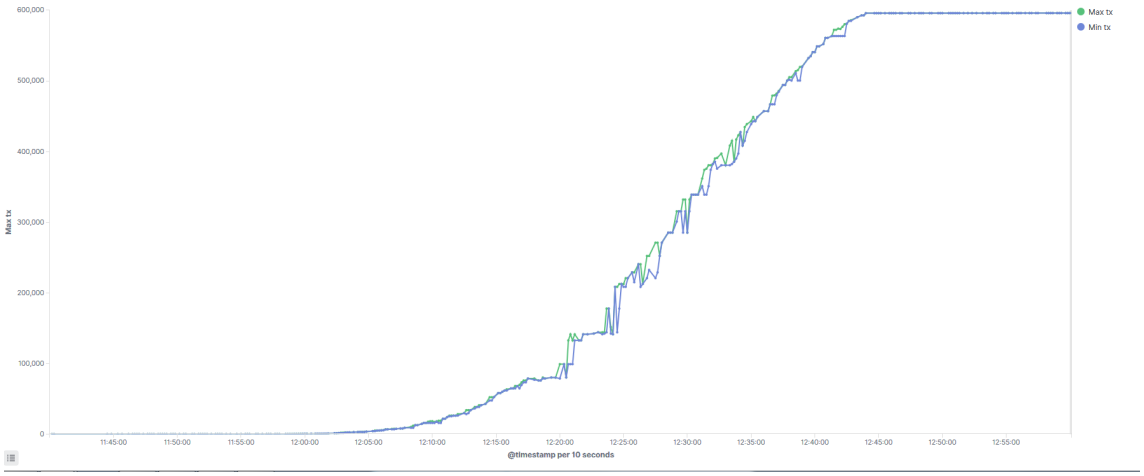

FIGURE 5.2 – Min and Max total tx for 5 nodes to timestamp

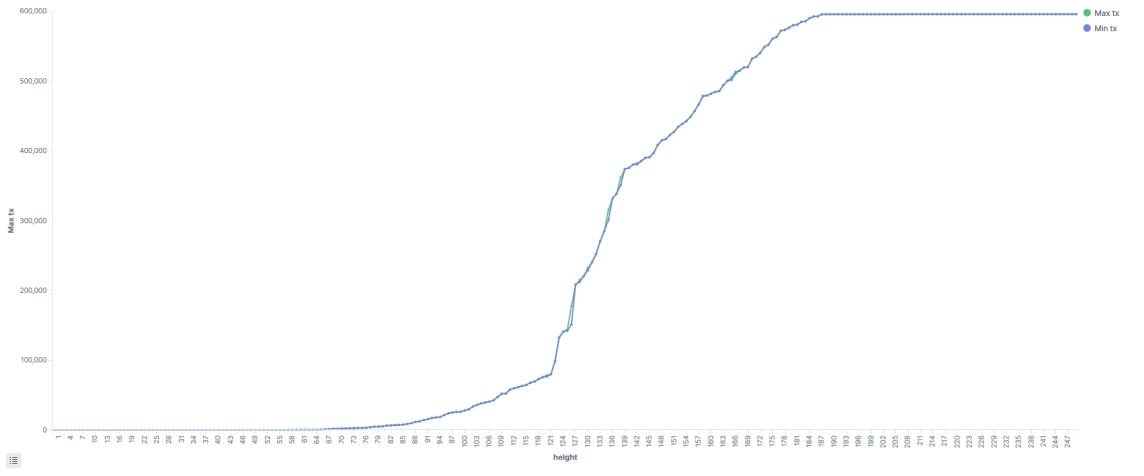

FIGURE 5.3 – Min and Max total tx for 5 nodes to block height

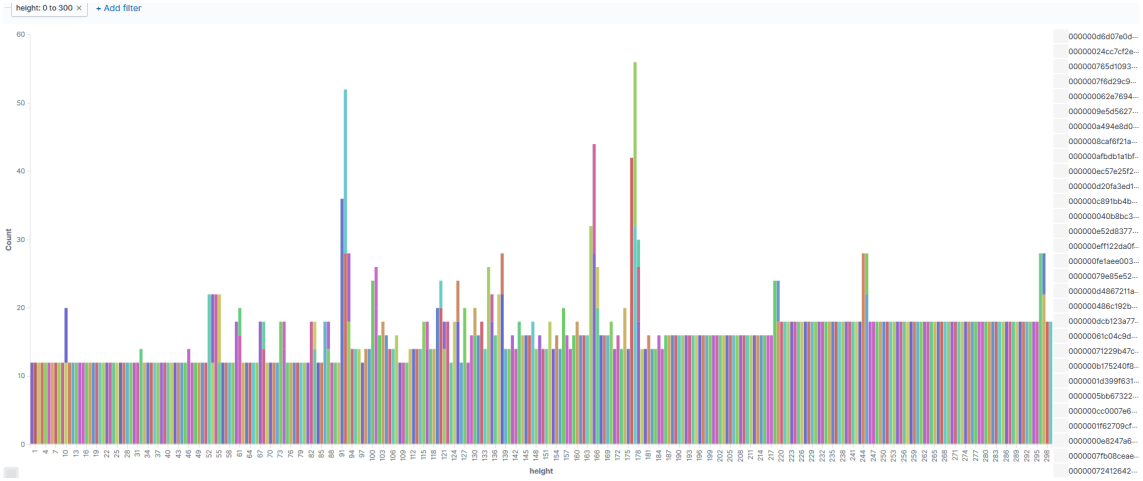

FIGURE 5.4 – Fork occurrence 5 nodes

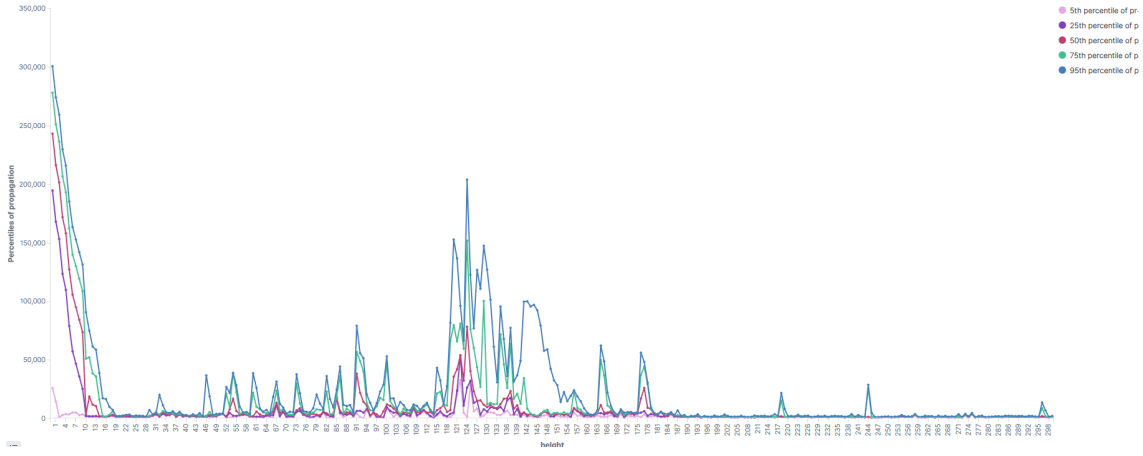

FIGURE 5.5 – Block propagation with 5 nodes

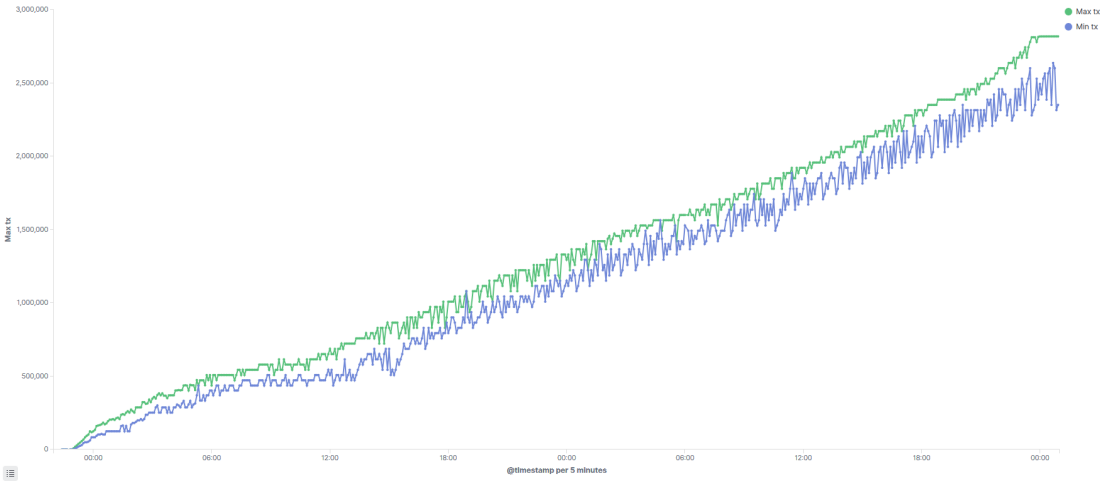

FIGURE 5.6 – Min and Max total tx for 25 nodes to timestamp

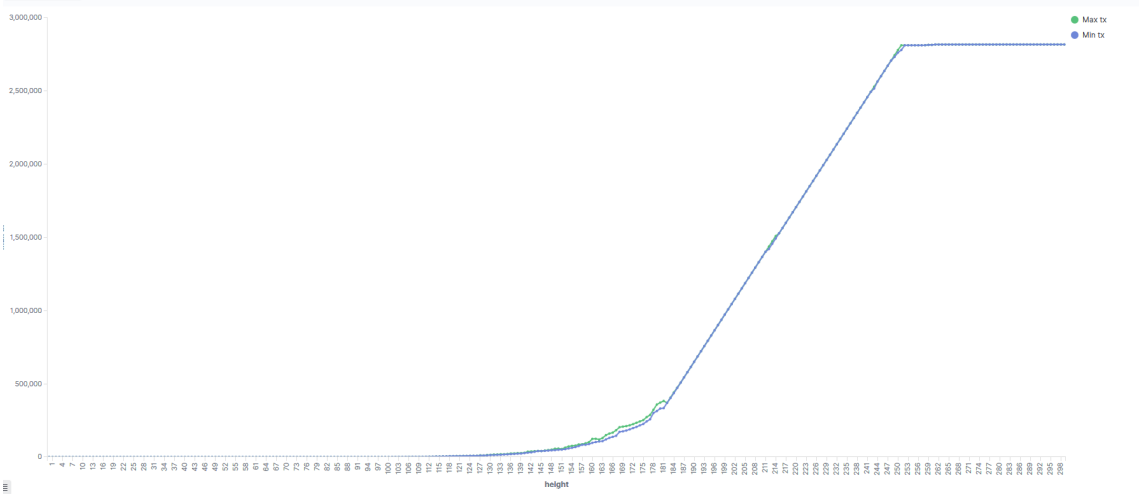

FIGURE 5.7 – Min and Max total tx for 25 nodes to block height

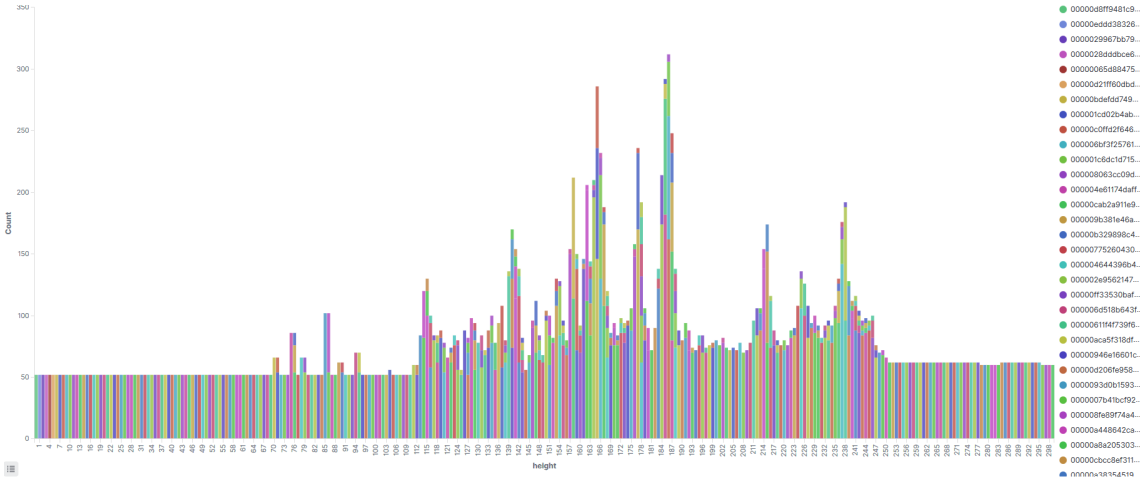

FIGURE 5.8 – Fork occurrence 25 nodes

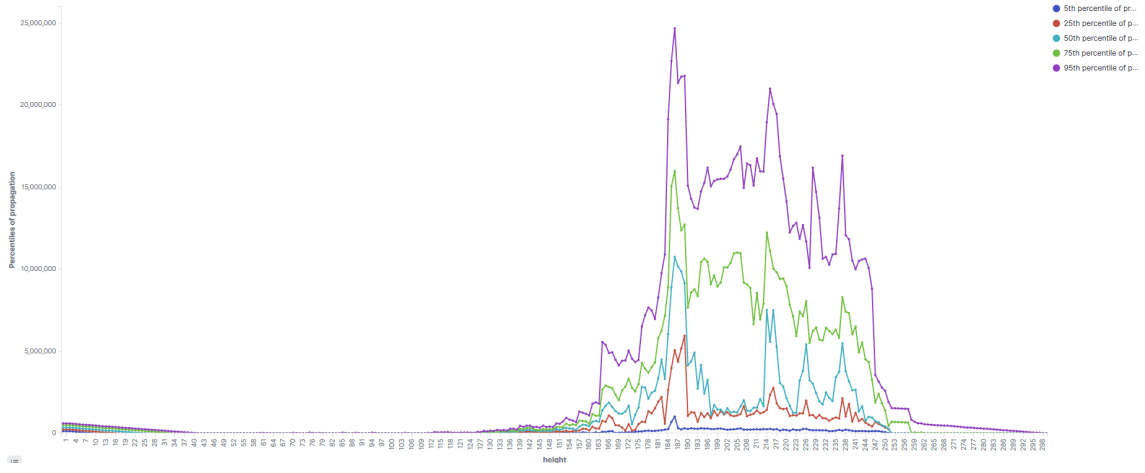

FIGURE 5.9 – Block propagation with 25 nodes

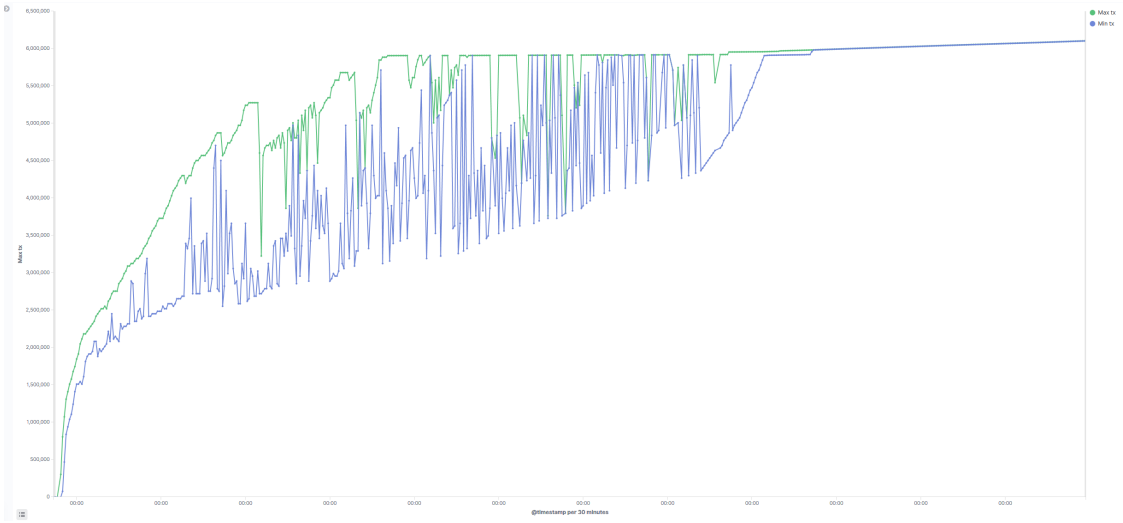

FIGURE 5.10 – Min and Max total tx for 50 nodes to timestamp

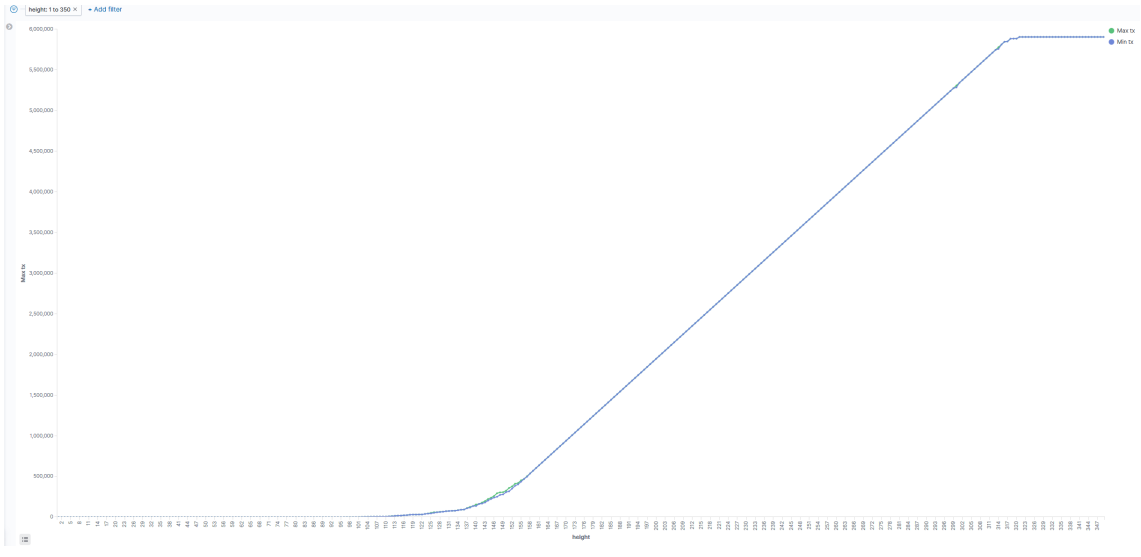

FIGURE 5.11 – Min and Max total tx for 50 nodes to block height

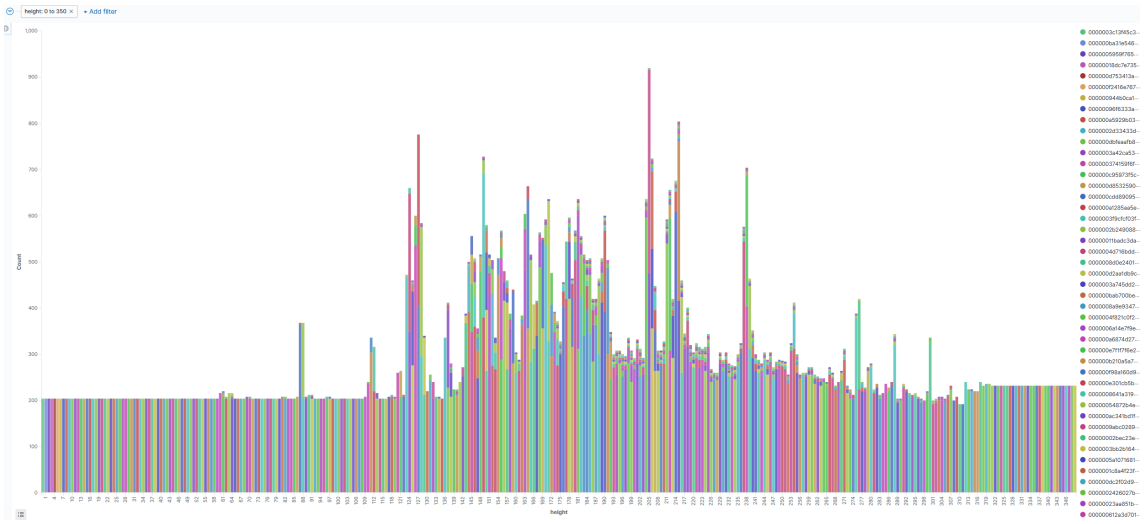

FIGURE 5.12 – Fork occurrence 50 nodes

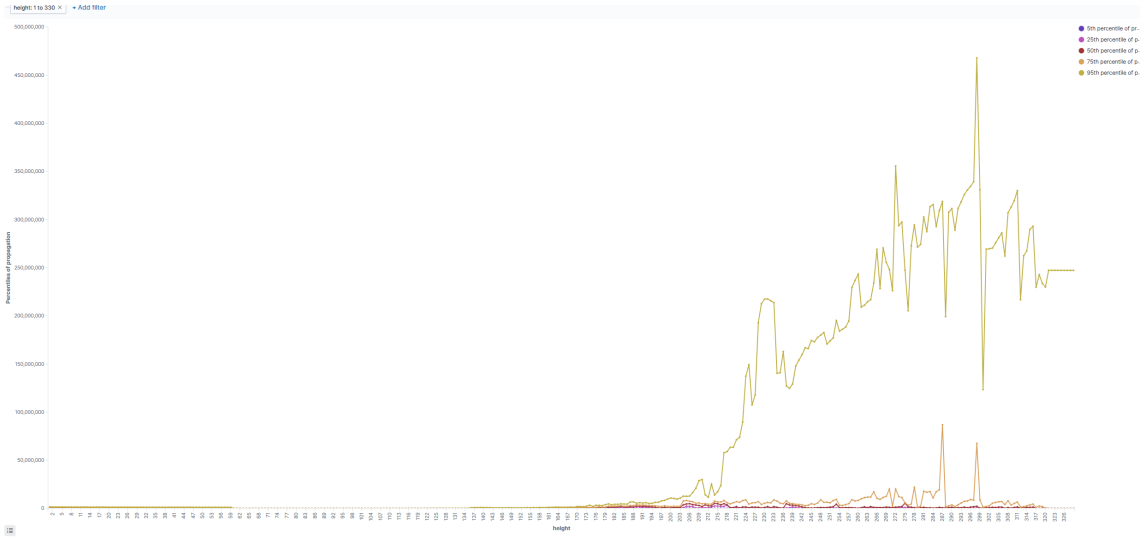

FIGURE 5.13 – Block propagation with 50 nodes

5.3.3 Chasing the perfect ratio

We study here how the blockchain perform under various ρ and scenarios. Our benchmarks ran with a network of 20 nodes, which is roughly the number of countries where Orange Money is present. We use 4 different consensus (mt, md) : *round robin* $(0, 0.8)$, *hybrid1* $(0.5, 0.8)$, *hybrid2* $(0, 0.8)$ and Nakamoto $(1, 0)$. We test different values for the parameters bt, bs and bw (Table 5.2) gives us multiple sets of tests for the same ratio $\rho = \frac{bs}{bt*bw}$. Our goal is to maximize the throughput while minimizing the forks. For example, looking at Table 3.1, if we first minimize ρ, then maximizes τ_e, we get $(\rho, \tau_e) = (0.1, 524 \text{ tx/s})$. If instead we fist maximize τ_e and then minimize ρ, we have $(\rho, \tau_e) = (1.6, 8388 \text{ tx/s})$.

TABLE 5.2 – Benchmark scenarios

bt(s)	16	16	16	16	64	64	64	64
bs(MB)	8	8	32	32	8	8	32	32
bw(Mb/s)	1000	256	1000	256	1000	256	1000	256
propag.(s)	0,064	0,25	0,256	1	0,064	0,25	0,256	1
$rho(\%)$	0,4	1,56	1,6	6,25	0,1	0,39	0,4	1,56
τ_e	2097	2097	8388	8388	524	524	2097	2097

Validation of the ratio

We here explore if each consensus behave the same across different parameters for a given ratio ρ. We study the consensus for $\rho = 0.4$, which is the following triples (bt, bs, bw) : $(64, 32, 1000)$, $(64, 8, 256)$ and $(16, 8, 1000)$. We recall that because our script starts on each VM as soon as the node is ready and we test different block times, the load from transactions will appear at different block height.

The case $(16, 8, 1000)$ is the one with the fastest block time in this set and $(64, 32, 1000)$ has big blocks, a lot of bandwidth and more time to process the transactions. The last case, $(64, 8, 256)$, has less bandwidth and smaller block size that can be limiting. Transactions have to be propagated through the network before they get included. At worst, information and data about those transactions have to go twice between nodes, first as single transactions and second in blocks. At best, peers receive the transaction data once if they signal they have not received it yet [4]. In any case more information than just the blocks will make use of the provided bandwidth. They are the (de facto) proxy used to study the bandwidth usage. Also Croman *et al.* in [CDE+16], extending on work by Gervais *et al.* [GKW+16], showed that blocks propagation is impacted by bandwidth unlike transactions which are,

[4]Example : https://developer.bitcoin.org/reference/p2p_networking.html#inv

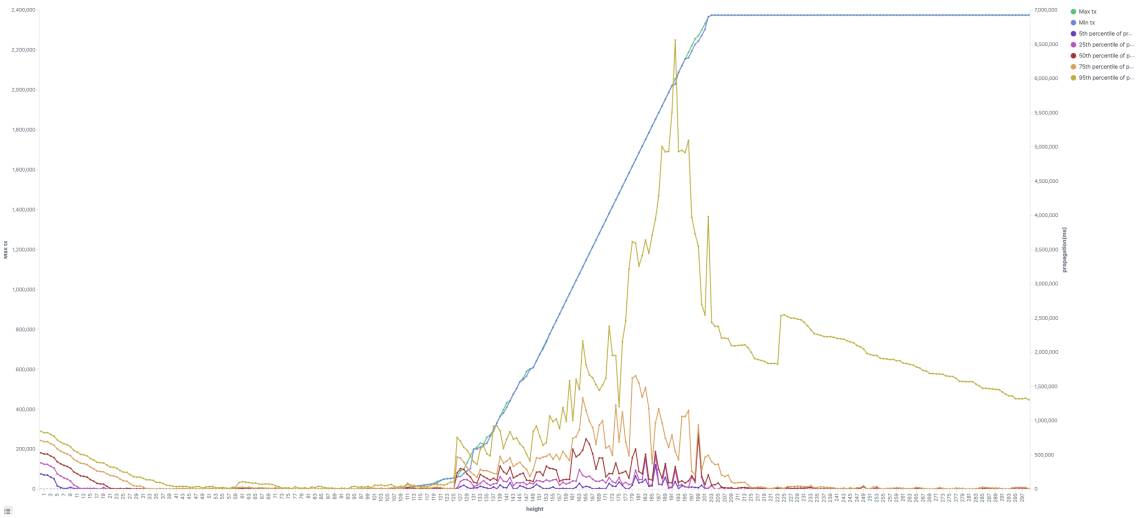

FIGURE 5.14 – Min/Max total tx and propagation for robin-16-8-1000-height

due to their small size, impacted by the network latency. As such, we expect the the cases with the less bandwidth to perform worse.

In the following sections, we present our results for the selected $\rho = 0.4$. The complete data set (csv files) is made available on this Github repository https://github.com/cryptohazard/scalablock/tree/main/data/final.

Round Robin

The network behaves differently under those 3 scenarios. In the $(16, 8, 1000)$ scenario, the network sustains the load the best. We can see that most nodes are in synchrony, as shown by the $50^{th} percentile$ propagation curve in Figure 5.14. The rate of transactions inclusion is also regular despite the forks (Figure 5.15) between block 125 and 170 that appears.

The other two do not perform as well. Out of the two cases, $(64, 32, 1000)$ and $(64, 8, 256)$, the latter performs worse, as expected. Both have most of their forks between blocks 45 and 90 (see Figure 5.17, 5.19) but the latter still have some forks appearing until block 120. Looking at the propagation charts, respectivelyFigure 5.16 and 5.18, we see that some nodes are still catching up with the rest of the network.

Hybrid 1

This hybrid consensus is the same tested in Section 5.3.2 and the case $(16, 8, 1000)$ behave roughly the same as the test with 25 nodes. Forks appear throughout the

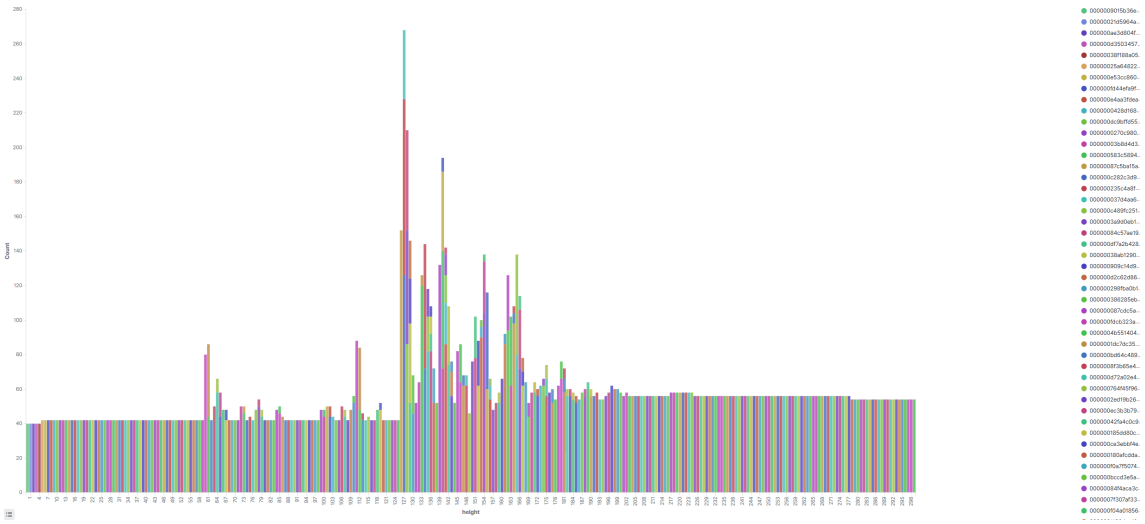

FIGURE 5.15 – Fork occurrence for robin-16-8-1000-height

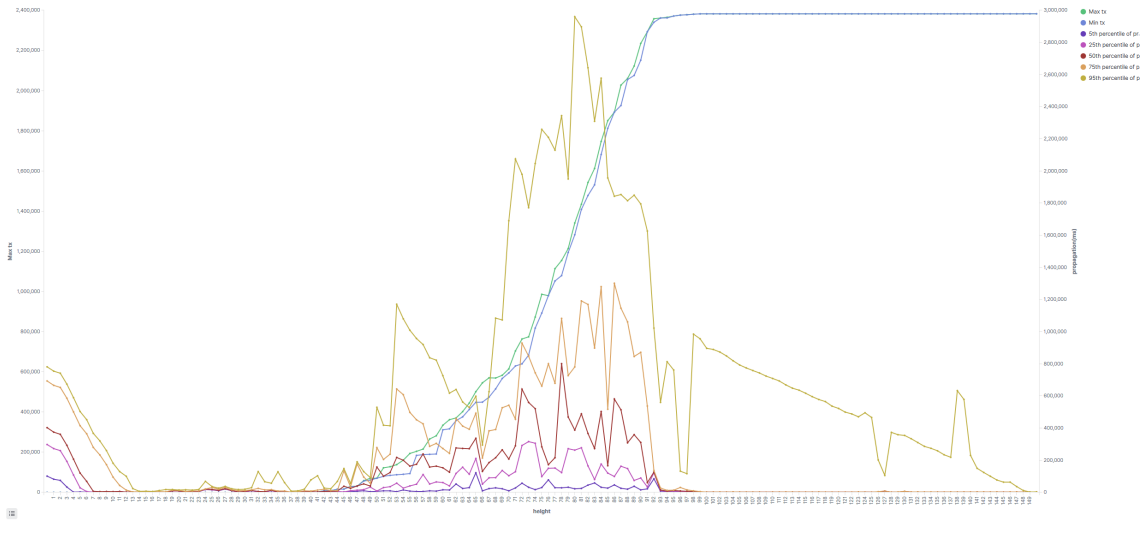

FIGURE 5.16 – Min/Max total tx and propagation for robin-64-32-1000-height

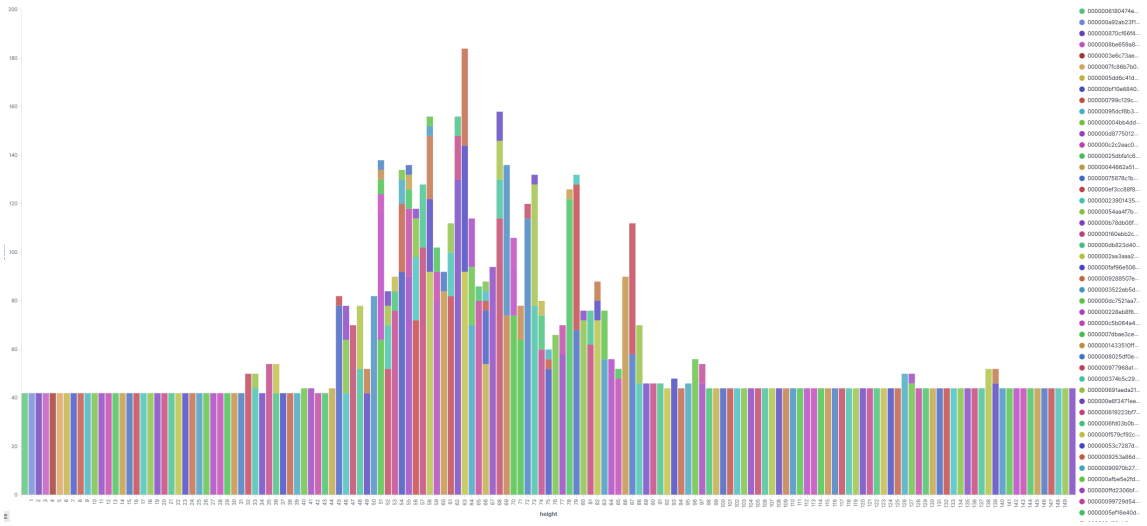

FIGURE 5.17 – Fork occurrence for robin-64-32-1000-height

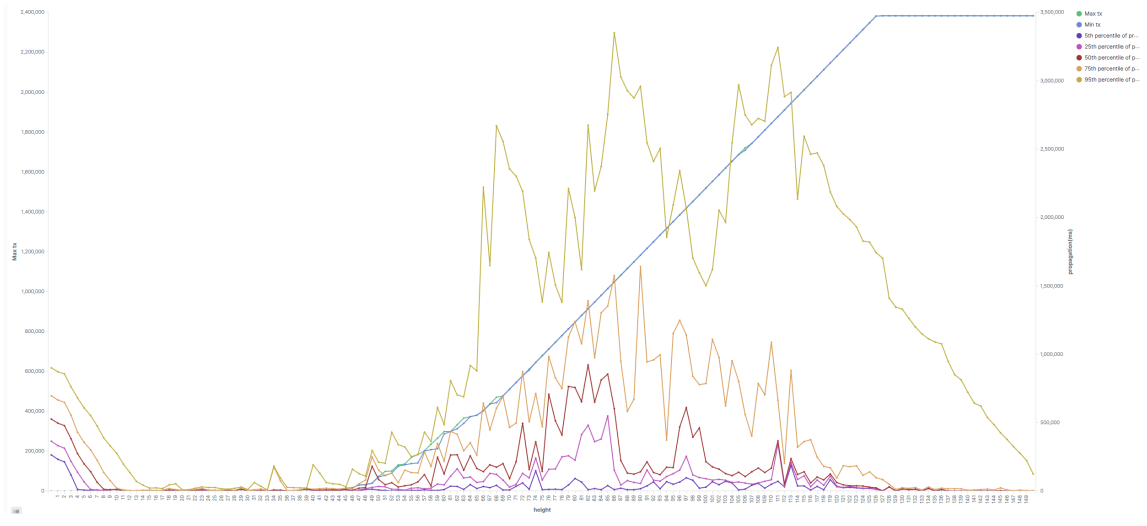

FIGURE 5.18 – Min/Max total tx and propagation for robin-64-8-256-height

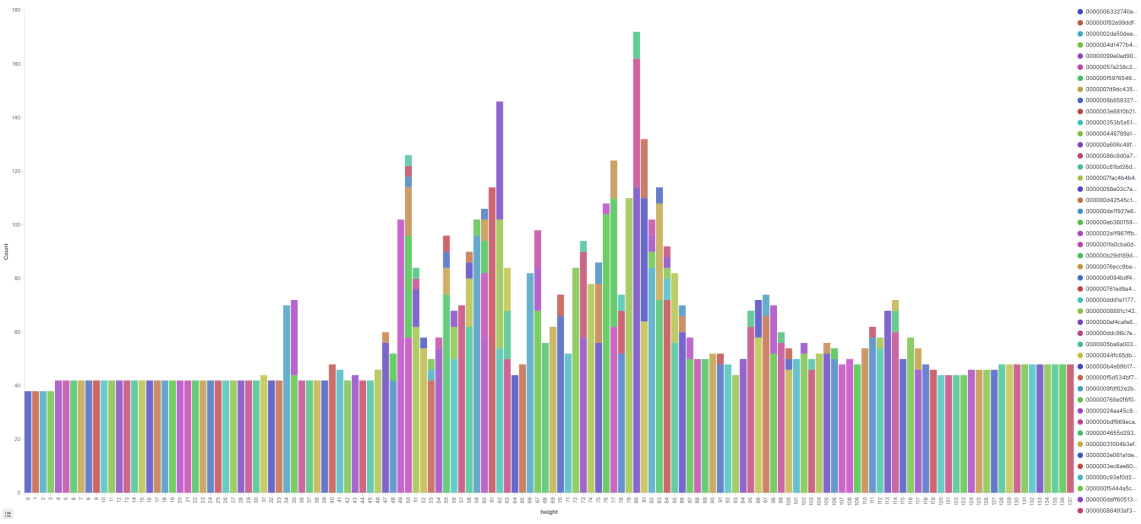

FIGURE 5.19 – Fork occurrence for robin-64-8-256-height

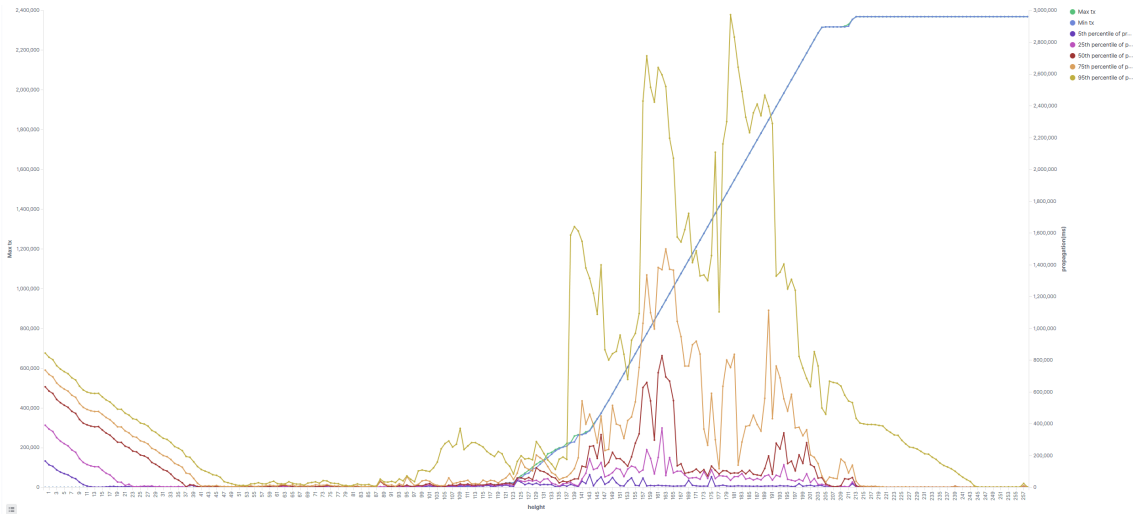

FIGURE 5.20 – Min/Max total tx and propagation for hybrid1-16-8-1000-height

whole experiment which is not desirable despite τ_e being the highest.

The case $(64, 32, 1000)$ has most of the fork between block 60 and 85. Looking at the max/min transactions Figure 5.22, we see the inclusion of new transactions stale. Most of the nodes keep are able to keep up with the network.

The $(64, 8, 256)$on the other hand stale badly between block 89 and 95. Many competing chains appears and nodes go back to block 89 a lot, as we see logged in Figure 5.25.

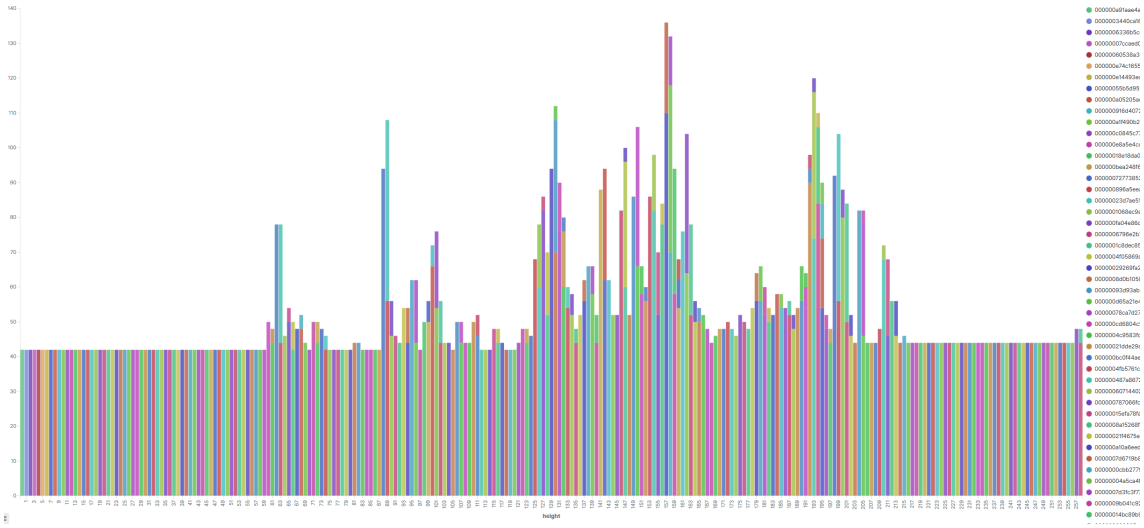

FIGURE 5.21 – Fork occurrence for hybrid1-16-8-1000-height

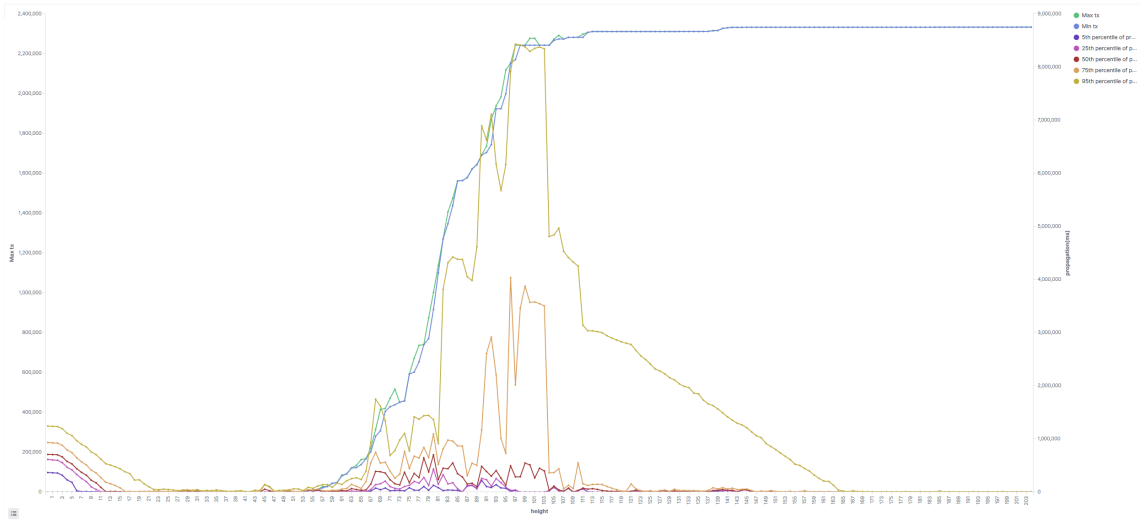

FIGURE 5.22 – Min/Max total tx and propagation for hybrid1-64-32-1000-height

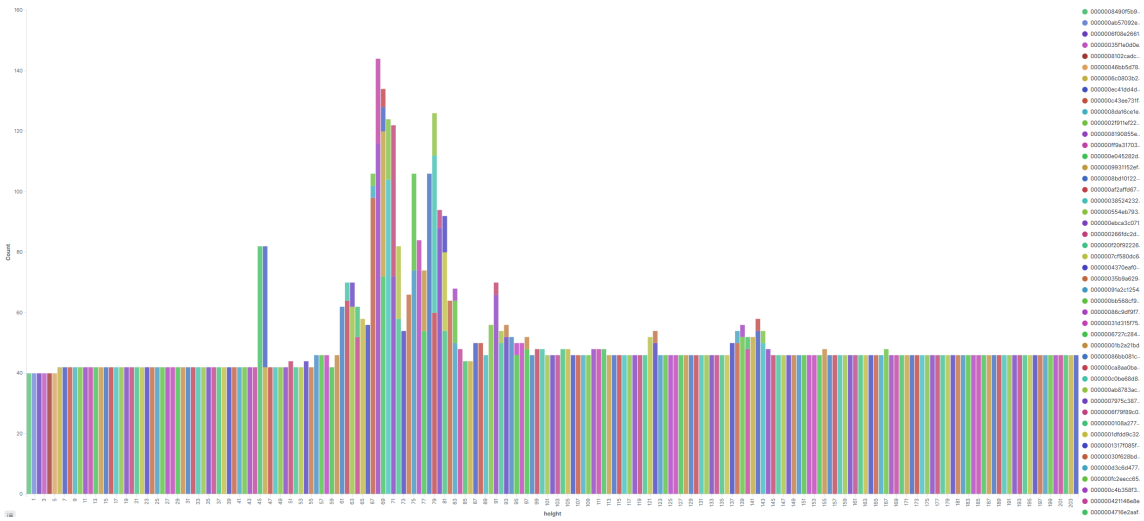

FIGURE 5.23 – Fork occurrence for hybrid1-64-32-1000-height

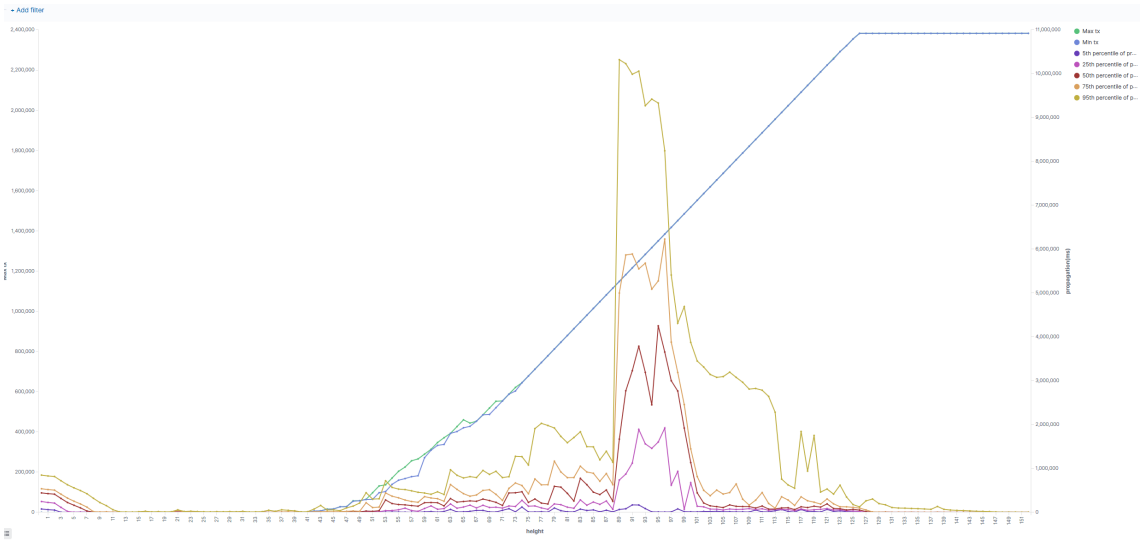

FIGURE 5.24 – Min/Max total tx and propagation for hybrid1-64-8-256-height

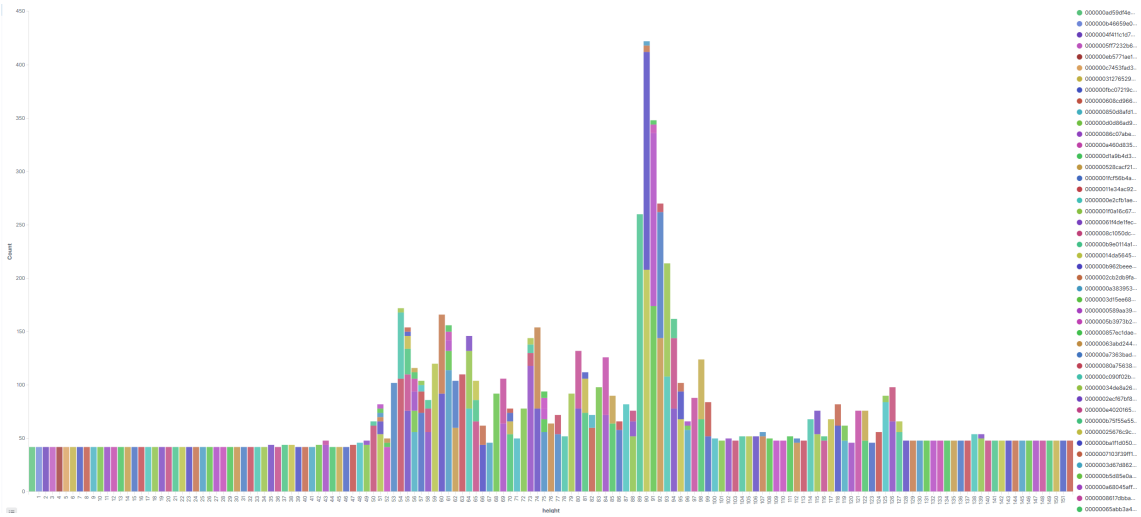

FIGURE 5.25 – Fork occurrence for hybrid1-64-8-256-height

Hybrid 2

In this second hybrid consensus we see for the case $(16, 8, 1000)$ that most of the network cannot followed the fast rhythm of block production. Using the 50^{th} and $25^{th}percentile$ propagation curves in Figure 5.26, we can see the network loose synchrony after block 154.

The case $(64, 32, 1000)$ does comparatively better taking into account the block propagation throughout the whole benchmark Figure 5.28 Most of the forks happen before block 100, which, again, is due to a slower block time.

most of the fork between block 60 and 85. Looking at the max/min transactions , we see the inclusion of new transactions stale. Most of the nodes keep are able to keep up with the network.

The $(64, 8, 256)$on the other hand stale badly between block 89 and 95. Many competing chains appears and nodes go back to block 89 a lot, as we see logged in Figure 5.31.

Bitcoin

Unlike in other consensus cases, the $(16, 8, 1000)$ does not have a steady rate of inclusion of transactions. We can see the pace slowing down around block 225 and being significantly reduced at block 250. The network then slowly get back in sync after 200 more blocks, as shown by the propagation of the $50^{th}, 75^{th}$ and $95^{th}percentile$ in Figure 5.32. The block time is most of the time below the set block

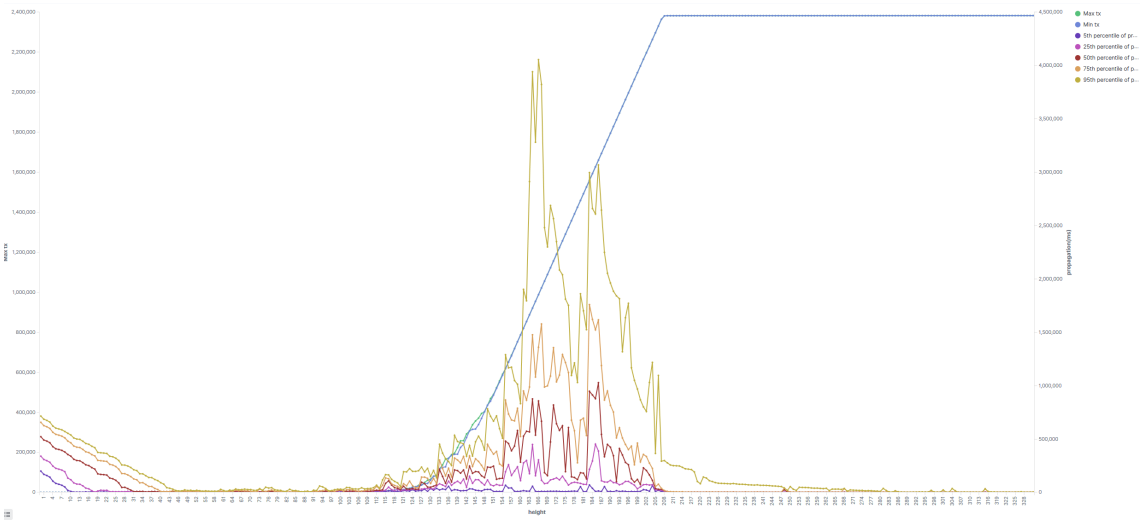

FIGURE 5.26 – Min/Max total tx and propagation for hybrid2-16-8-1000-height

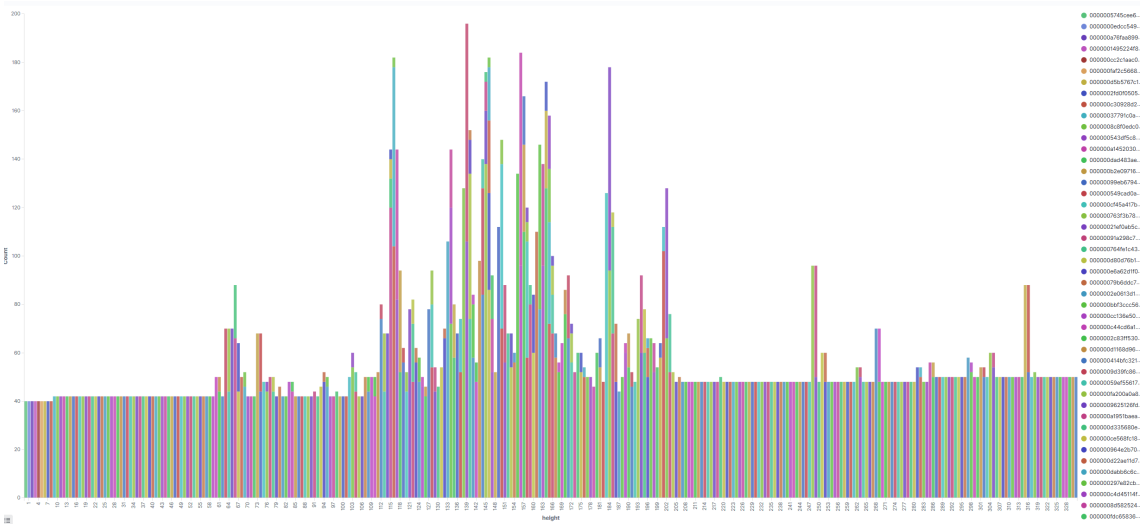

FIGURE 5.27 – Fork occurrence for hybrid2-16-8-1000-height

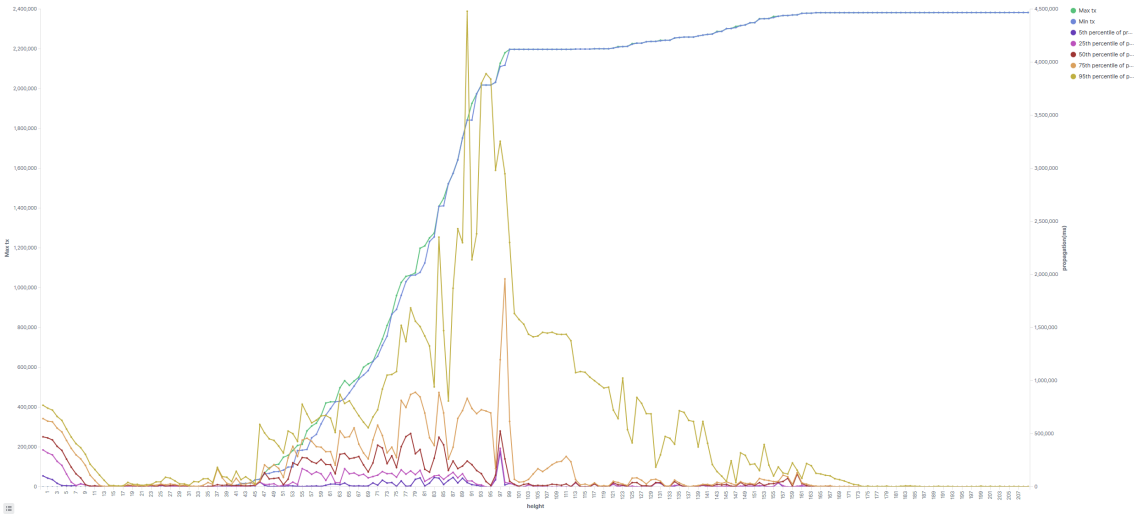

FIGURE 5.28 – Min/Max total tx and propagation for hybrid2-64-32-1000-height

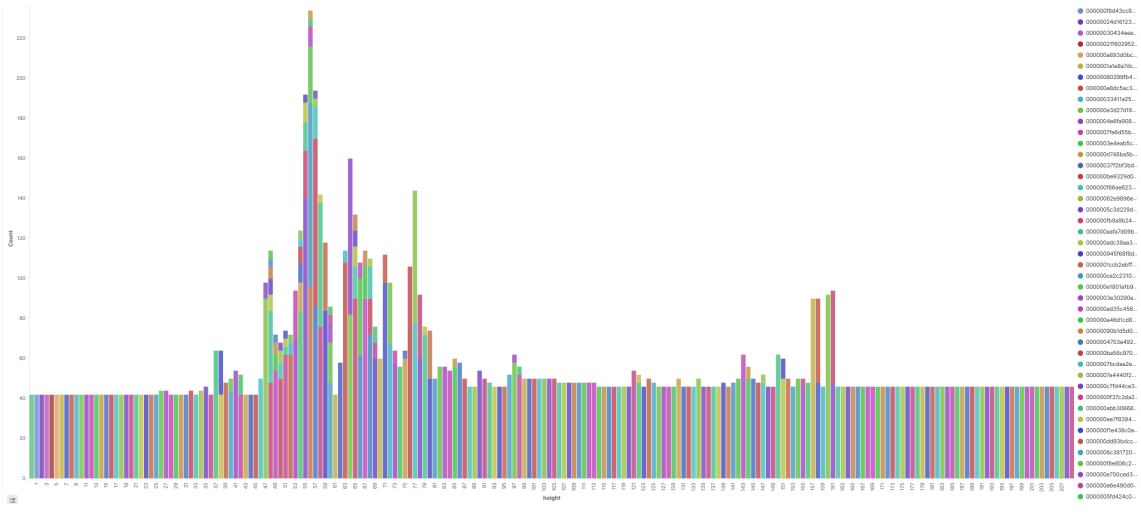

FIGURE 5.29 – Fork occurrence for hybrid2-64-32-1000-height

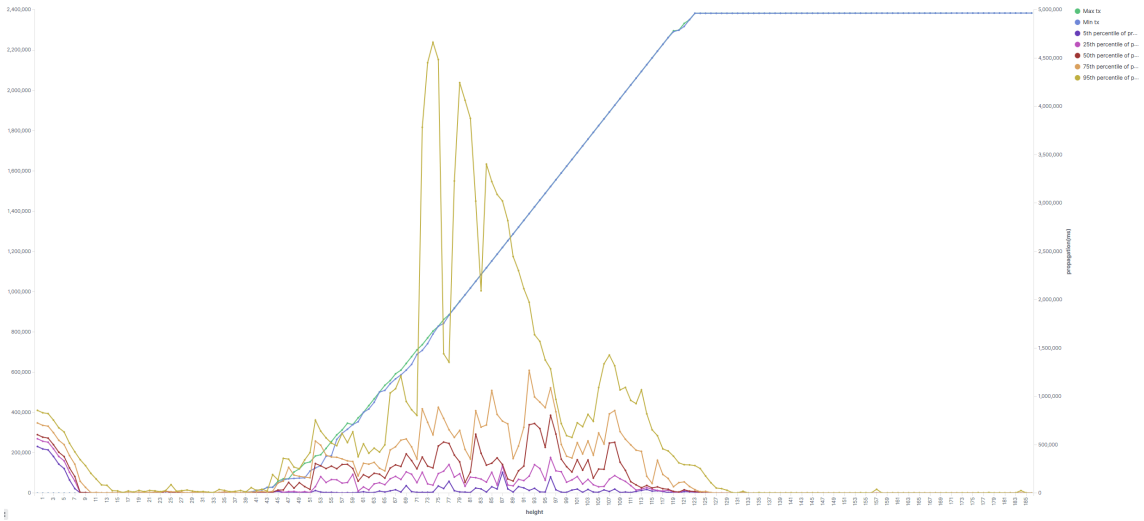

FIGURE 5.30 – Min/Max total tx and propagation for hybrid2-64-8-256-height

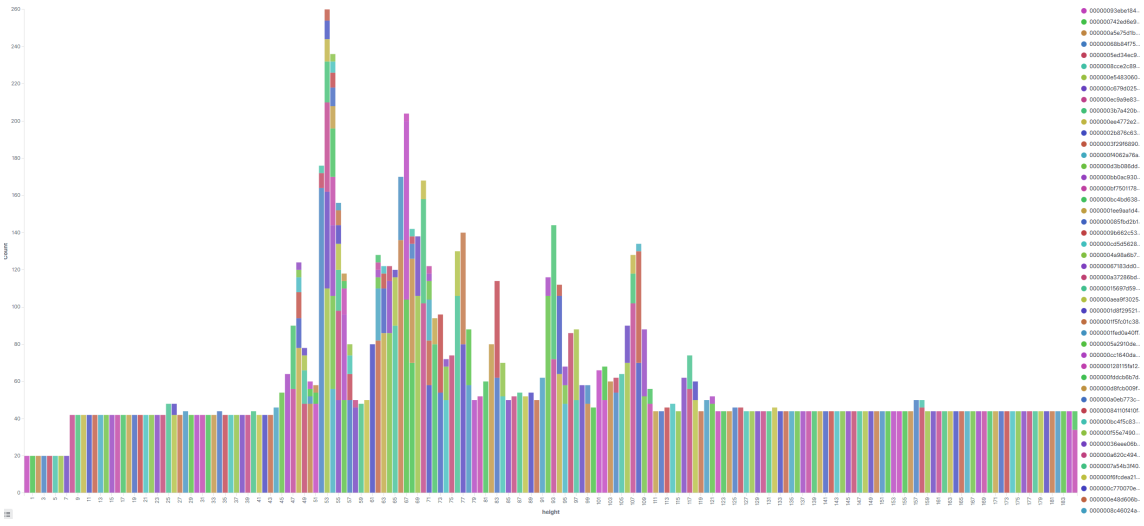

FIGURE 5.31 – Fork occurrence for hybrid2-64-8-256-height

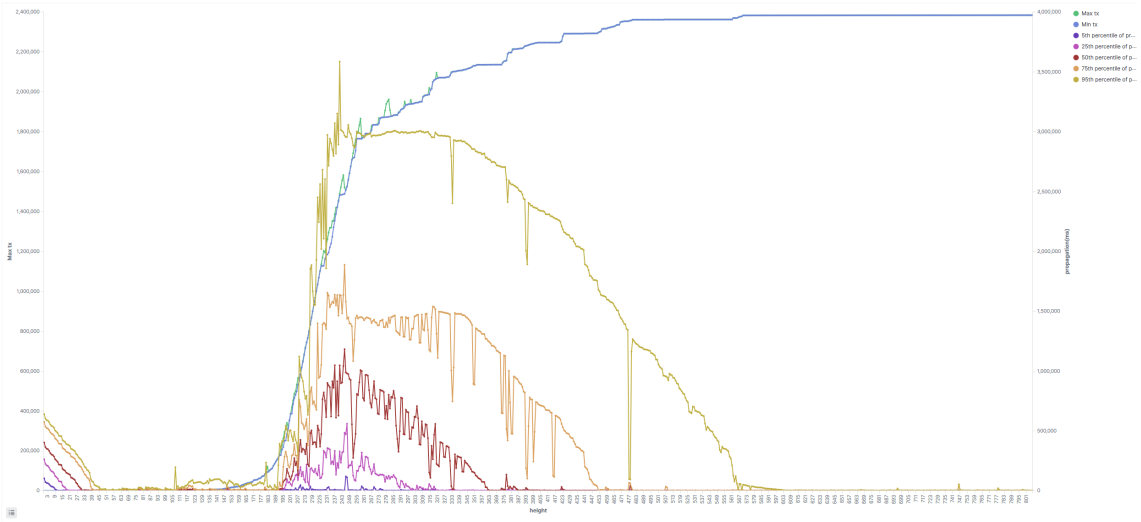

FIGURE 5.32 – Min/Max total tx and propagation for bitcoin-16-8-1000-height

time of 16 seconds between block 100 to 150 but starts to grow closer to the block 200. This is when the transaction rate really starts to pick up and forks appear. The blocks are fuller, with many of them in a row close to the maximum number of transactions, sprinkled with blocks with much lower number of transaction and even some with only 1.

With the case $(64, 32, 1000)$, the $50^{th} percentile$ is indicative of a network that fails to keep most its nodes in synchrony. We observe that half of the network receives some blocks with a delay equivalent to 16 *blocktime* (block 123 & 124). Comparatively, the last case has shorter delay but has more forks during the busy period.

5.4 Conclusion

As we can see, our framework is able to let us run many experiments with various (realistic) parameters. In particular, our system is able to monitor forks, which represent an important problem with regards to scalability. Our system can also be used to check default or suggested parameters of existing blockchains (such as the Bitcoin example). Testing the blockchains on our infrastructure lets us anticipate the real behaviour of the system once deployed on the field.

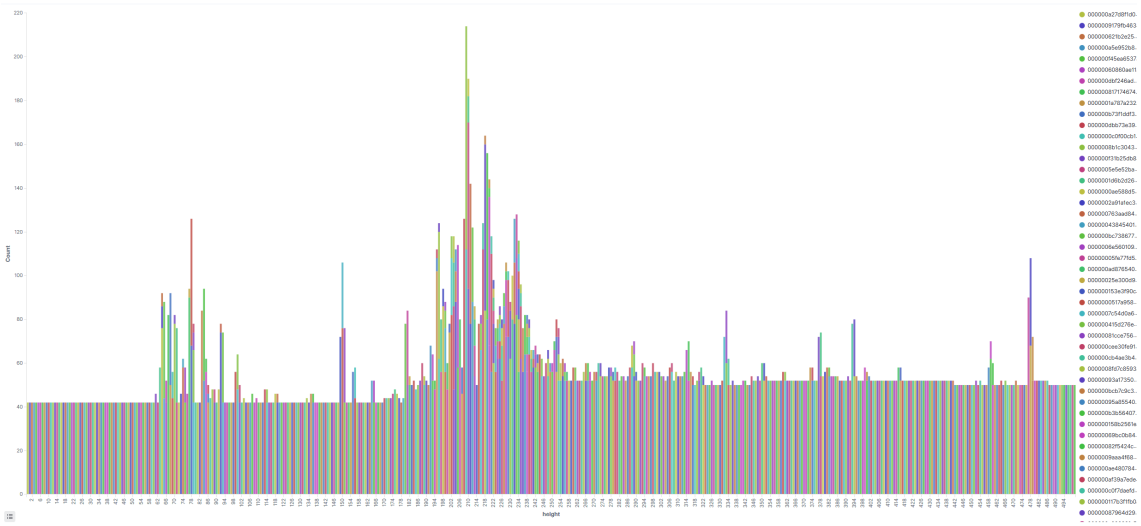

FIGURE 5.33 – Fork occurrence for bitcoin-16-8-1000-height

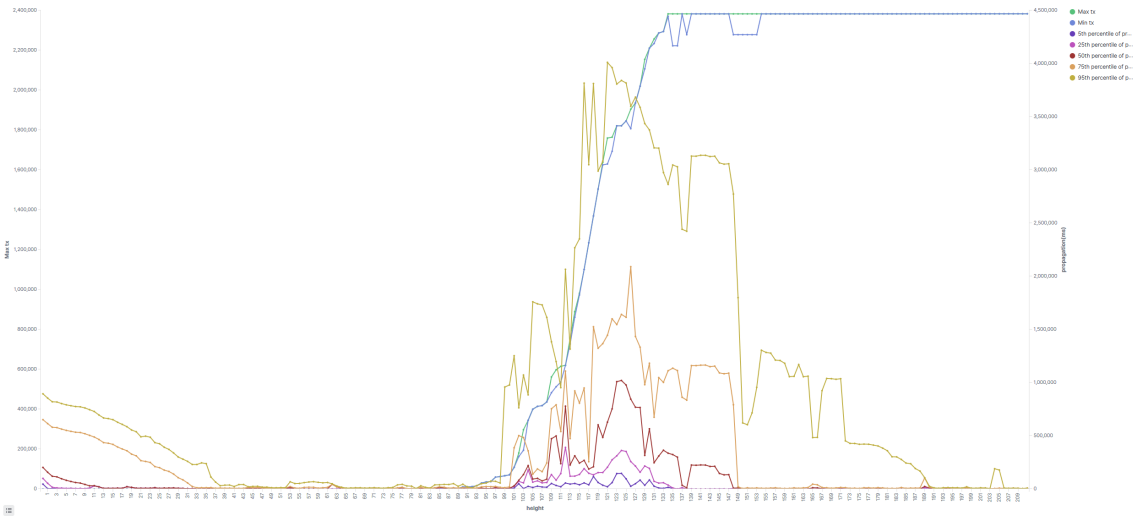

FIGURE 5.34 – Min/Max total tx and propagation for bitcoin-64-32-1000-height

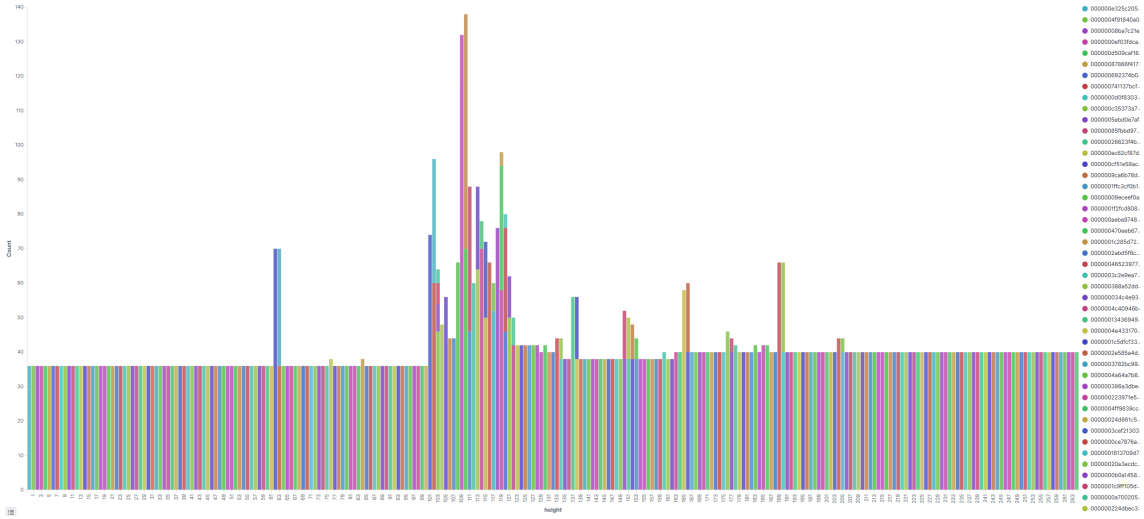

FIGURE 5.35 – Fork occurrence for bitcoin-64-32-1000-height

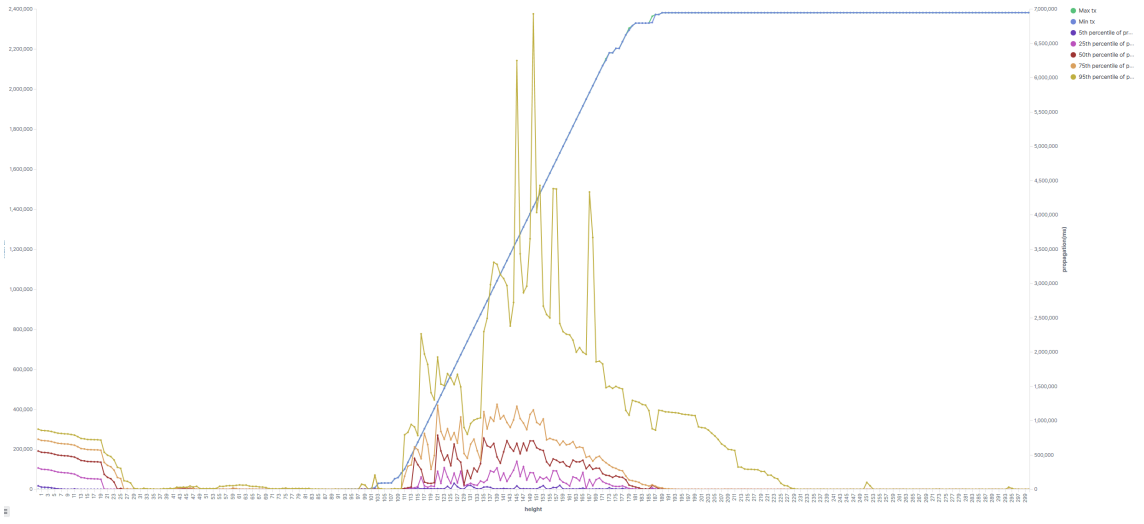

FIGURE 5.36 – Min/Max total tx and propagation for bitcoin-64-8-256-height

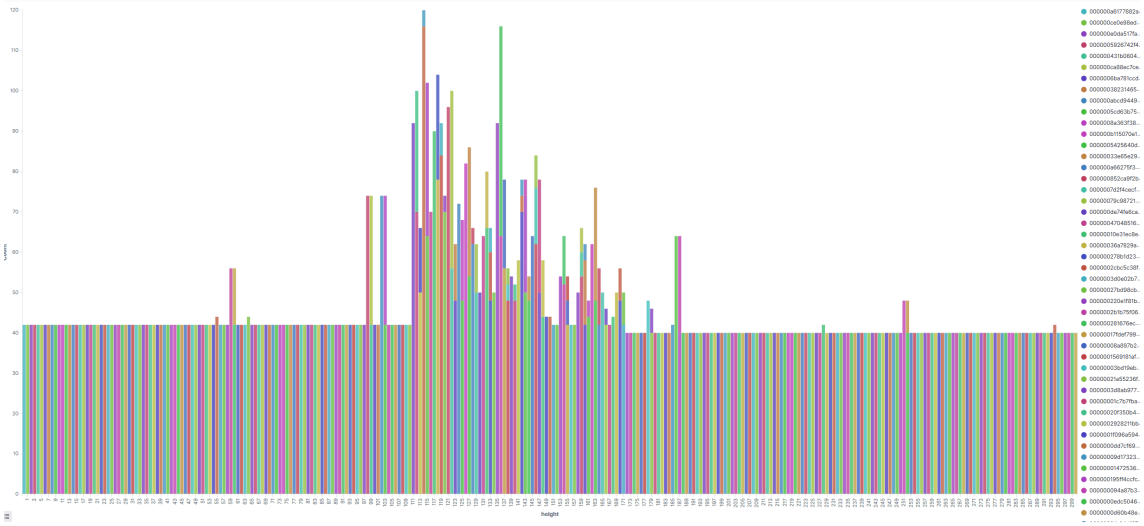

FIGURE 5.37 – Fork occurrence for bitcoin-64-8-256-height

Chapitre 6

Internet of blockchains

This chapter discusses the issues when using several different blockchains in order to exchange currency (called atomic chain swaps). This chapter was presented in [ZDBN19]

Contents

6.1 Introduction

Blockchains enable users to send and receive money in a trust-less manner. They offer a mean for peer-to-peer (P2P) exchanges of crypto-assets such as cryptocurrencies over the Internet, which was not possible without intermediaries before.

Blockchains are a very young ecosystem, where good practices, controls, certifications and regulations still need to be defined. For instance trading the associated assets (located on different blockchains) usually takes place on centralized market places, called exchanges, which become trusted third parties. This expose users to loss of their funds, through hacks of those platforms or straight-up scams when owners disappear with the funds. It emphasizes a common wisdom in this ecosystem :

Not your keys, Not your funds. If we do not control the private key of the account with the assets, we do not actually own the funds.

Atomic cross-chain swaps solve this problem for exchanges of assets without intermediaries. Those swaps are atomic in the sense that either both parties receive the other crypto-assets, or they both keep their own. They are used in two ways. First, as explained earlier, they can serve as a basis for non custodial exchanges. Using these decentralized marketplaces, users keep control of their funds while being able to trade them.

Second, they enable Layer 2 scaling solutions, like [DW15b, PD16], to become a network of payment channels and not just P2P channels, thus increasing the scalability of blockchains. The principle of layered scalability for blockchains is to conduct some transactions off the blockchain (*off-chain* or Layer 2) and notarize the state on the Layer 1 (*on-chain*), the blockchain, when necessary. The payment channel works like a tab between two users, with many transactions happening *off-chain* and only two transactions happening *on-chain*, one to open the channel and another one to close it. The limitation is obviously that Alice has to open a channel first with Bob before they can exchange. Atomic swaps come into play for cross-channels payments. If Alice has a payment channel with Bob and Bob has one with Charlie, Alice can atomically send a payment to Charlie with a swap between the two channels, provided there are enough funds in both channels.

The first proposal for atomic cross-chain swaps came from an online forum, *bitcointalk* [Tie]. It was between Bitcoin [Nak08] and forks of Bitcoin, called altcoins for *alternative coins*. It made use of the scripting capabilities of those protocols, particularly conditional release of the coins, hashed locked and time locked transactions. This restricts the use cases to blockchains with scripting or smart contract capabilities. There are various types of cryptocurrencies with different goals and features and not all of them support Hash Time Lock Contracts (HTLC for short). In this work, we propose an extension of atomic cross-chain swaps to blockchains that do not have such capabilities with the same assumptions as HTLC atomic swaps. We construct our protocol for a blockchain with smart contracts and one that only supports multi-signatures (*multisig*) transactions. Those transactions are control by multiple private keys and require, to be valid, a certain number of signatures. Each signature is generated by a different private key and all those signatures need to be explicitly attached to the transaction. Using multisig transactions, we provide greater capabilities for cross-chain communications without adding any extra trust hypothesis.

The rest of this paper is organized as follows : the properties of atomic cross-chain

swaps and their construction using HTLCs are recalled in section 6.2. We show how to extend them to blockchains with multisig in section 6.3, then discuss the relative advantages and drawbacks of our proposal in section 6.4. A comparison to related works is proposed in section 6.5 before concluding.

6.2 Hash Time Locked Contracts

6.2.1 Properties

Atomic swaps have two properties :

1. If all parties behave, i.e follow the protocol, all swaps happen.

2. If any party misbehaves, everyone is refunded.

This gives the users the guarantee they will not lose their assets by participating in an exchange. As stated earlier, the first proposal for blockchains came from *bitcointalk* [Tie]. We note that most blockchains rely on digital signatures to track the ownership of the coins. Thus, the proposed protocols, so far, make use of some scripting or smart contracts capabilities.

- Hash lock : to release the funds in the contract, one needs to provide the secret preimage s that gives $H(s)$, the lock in the contract,

- Time lock : if nothing happens before some time t on a chain, refund the user on that chain.

Those swaps are currently possible between Bitcoin and altcoins such as Litecoin [Lee11] and on smart contract platforms like Ethereum [eth] or EOS [eos].

6.2.2 HTLC based atomic swaps

We now present the atomic swap based on HTLCs. Let Alice be a user of the blockchain $\mathcal{A}$ that wants to exchange X coins with Bob a user of $\mathcal{B}$ for Y coins. We suppose that there is a common cryptographic hash function $Hash$ available on both blockchains. We also suppose that Alice and Bob agree on the timelocks T for Bob and $2T$ for Alice that are function of the blockchains involved. Specifically those timelocks are function of the Block times and confirmation times. We omit the digital signatures in the description for clarity and brevity.

Setup

1. Alice generates a secret preimage s and computes $h = Hash(s)$.
2. She also generates a HTLC on $\mathcal{A}$ with the hash lock h and the time lock $2T$
3. Alice sends h and the HTLC address to Bob
4. Bob can verify that the HTLC on $\mathcal{A}$ is properly constructed.
5. If it is valid, then Bob can generate a HTLC on $\mathcal{B}$ with the same hash lock h but the time lock T
6. Alice can verify that the HTLC on $\mathcal{B}$ is properly constructed.
7. If it is valid, then Alice funds the HTLC on $\mathcal{A}$ and Bob does the same on $\mathcal{B}$.

We stress that the time locks are different on $\mathcal{A}$ and B and that Bob does not know s at this point.

Swap

The swap works as follows :

1. Alice presents the secret preimage s to the HTLC in $\mathcal{B}$ and receives the Y coins. She does so sufficiently early, with respect to T and the block time and confirmation time of $\mathcal{B}$.
2. Bob, who was monitoring the HTLC on $\mathcal{B}$ learns s.
3. Bob can then present s to the HTLC on $\mathcal{A}$ and receive the X coins. He does so sufficiently early, with respect to $2T$ and the block time and confirmation time of $\mathcal{A}$.

Refunds

Each user can refund herself by waiting for her time lock to expire and call the relevant HTLC. Obviously this is only possible if the funds have not been spent, in which case the other user is able to spend hers also.

Discussions

Alice has the advantage of being able to initiate the swap. This means that she can wait as long as possible to see if the trade gets more or less interesting with prices fluctuations. She can also propose a swap just to lock Bob's funds. We do not address these issues in this paper.

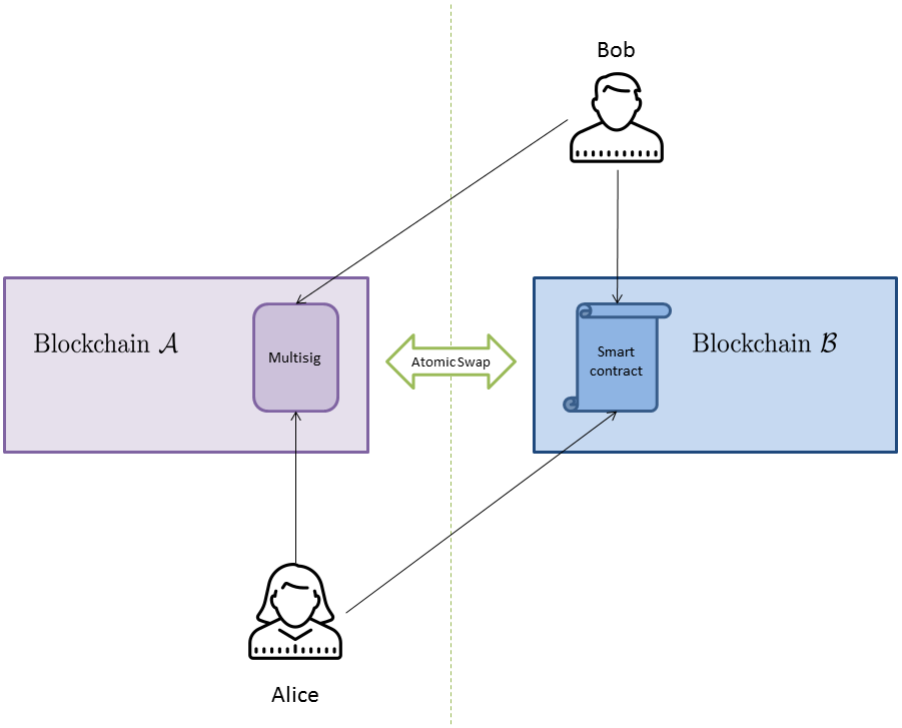

FIGURE 6.1 – Atomic Cross-Chain Swaps using multisig and smart contract

6.3 Extending Atomic Cross-chain Swaps

The previous protocol is designed for blockchains that support Hashed Timelock Contract. But scripting or smart contract capabilities are not supported by all blockchains, including commonly used ones such as Steem [LSZ+16]. We propose a protocol to conduct an atomic swap between a blockchain that supports scripting or smart contracts and one that supports multisig instead of HTLCs. Figure 6.1 presents a high level overview of the actors of the protocol. The basic idea is to transfer all the time locking part on the first blockchain, since the second does not support it. For example Steem supports multisig accounts and uses ECDSA signatures [JMV01]. On Ethereum, the smart contract [Ope] can be used as an implementation basis. We leave the implementation of the full protocol for future work.

6.3.1 Assumptions

Our protocol enables two users, Alice and Bob, of different blockchains to securely swap their coins. As such we rely on the security model of those blockchains. We assume that each of them has a majority ($> 2/3$) of consensus participants, miners for Proof-of-Work and validators for Proof-of-Stake, that are honest [BKM18b, DPS17, ES18, GKW+16, KRDO17, Nak08].

We also assume that Alice and Bob have a secure communication channel to do price discovery and exchange the swap details. We suppose that they agree on the timelocks T for Bob and $2T$ for Alice that are function of the blockchains involved (see Discussion 6.4).

We further assume that one blockchain supports multi-signature(*multisig*) transactions and that the second blockchain supports smart contracts and can verify the signature of a transaction issued in the first blockchain.

6.3.2 Notations

Let $\mathcal{A}$ be the blockchain with only multisig and $\mathcal{B}$ be the smart chain. Let skA be the private key for the signature scheme on a blockchain $\mathcal{A}$ and pkA the corresponding public key. We note $\mathsf{sk}\mathcal{A}_A$ and $\mathsf{pk}\mathcal{A}_A$ if these are Alice's private and public keys and $\mathsf{sk}\mathcal{A}_B$ and $\mathsf{pk}\mathcal{A}_B$ for Bob's. We call SC the smart contract for the atomic swap on $\mathcal{B}$. We call M the multisig account on $\mathcal{A}$ that helps make the atomic swap. This can be understood as a shared account between Alice and Bob and we note $\mathrm{M} = (\mathsf{pk}\mathcal{A}_A, \mathsf{pk}\mathcal{A}_B)$ to express that this account requires both signatures, meaning Alice and Bob need to cooperate to transact on the *behalf* of M.

Let the refund operations be $\mathrm{R}(\mathcal{A}, \mathsf{pk}\mathcal{A}_A)$ and $\mathrm{R}(\mathcal{B}, \mathsf{pk}\mathcal{B}_B)$, which means that Alice (resp. Bob) makes a transaction so that the public key $\mathsf{pk}\mathcal{A}_A$ (resp. $\mathsf{pk}\mathcal{B}_B$) has sole control of the funds on the chain $\mathcal{A}$ (resp. $\mathcal{B}$) and Alice (resp. Bob) is thus refunded. Let the swap operations be $\mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A)$ and $\mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B)$, which means that Alice (resp. Bob) makes a transaction so that the public key $\mathsf{pk}\mathcal{B}_A$ (resp. $\mathsf{pk}\mathcal{A}_B$) has sole control of the funds on the chain $\mathcal{B}$ (resp. $\mathcal{A}$) and Alice (resp. Bob) has completed the swap.

6.3.3 Setup, transactions and operations

Setup

1. Alice generates $\mathsf{sk}\mathcal{A}_A$, $\mathsf{pk}\mathcal{A}_A$, $\mathsf{sk}\mathcal{B}_A$, $\mathsf{pk}\mathcal{B}_A$ and sends $\mathsf{pk}\mathcal{A}_A$ and $\mathsf{pk}\mathcal{B}_A$ to Bob.
2. Bob generates $\mathsf{sk}\mathcal{A}_B$, $\mathsf{pk}\mathcal{A}_B$, $\mathsf{sk}\mathcal{B}_B$, $\mathsf{pk}\mathcal{B}_B$ and sends $\mathsf{pk}\mathcal{A}_B$ and $\mathsf{pk}\mathcal{B}_B$ to Alice.
3. Alice creates $\mathrm{M} = (\mathsf{pk}\mathcal{A}_A, \mathsf{pk}\mathcal{A}_B)$ on $\mathcal{A}$ and sends its address to Bob for verification. She also creates and sends $tx_1 = \mathrm{R}(\mathcal{A}, \mathsf{pk}\mathcal{A}_A)$ and $tx_2 = \mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B)$.
4. After verification, Bob computes (σ_1, tx_1), where $\sigma_1 = Sign_{\mathsf{sk}\mathcal{A}_B}(tx_1)$. He can then create SC, fund this contract and send its address and (σ_1, tx_1).
5. Alice verifies that everything is in order and fund the account M.

OP1	Alice creates $\sigma'_1 = Sign_{\mathsf{sk}\mathcal{A}_A}(tx_1)$ and broadcasts $(\sigma'_1, \sigma_1, tx_1)$
OP2	Bob waits $t \geq T$ and calls SC1
OP3	Bob creates $\sigma_2 = Sign_{\mathsf{sk}\mathcal{A}_B}(tx_2)$ and sends (σ_2, tx_2) to Alice
OP4	Alice creates $\sigma'_2 = Sign_{\mathsf{sk}\mathcal{A}_A}(tx_2)$ and calls SC3
OP5	Bob learns σ'_2 monitoring SC and broadcasts $(\sigma'_2, \sigma_2, tx_2)$
OP6	Alice waits for $t \geq 2T$ and calls SC4
OP7	Bob learns σ'_1 from $\mathcal{A}$ and calls SC2

TABLE 6.1 – List of operations.

The smart contract SC.

It consists of four procedures SC1, SC2, SC3 and SC4 respectively presented in Algorithm 1.

loa 1 – The smart contract SC.

1: $unlock \leftarrow \mathsf{false}$
2: **procedure** SC1
3: **if** $t \geq T$ **and** $unlock = \mathsf{false}$ **then**
4: R($\mathcal{B}, \mathsf{pk}\mathcal{B}_B$)
5: **procedure** SC2(σ'_1, tx_1)
6: **if** $\mathsf{Verify}_{\mathsf{pk}\mathcal{A}_A}(\sigma'_1, tx_1)$ **then**
7: R($\mathcal{B}, \mathsf{pk}\mathcal{B}_B$)
8: **procedure** SC3(σ'_2, tx_2)
9: **if** $\mathsf{Verify}_{\mathsf{pk}\mathcal{A}_A}(\sigma'_2, tx_2)$ **then**
10: $unlock \leftarrow \mathsf{true}$
11: **procedure** SC4
12: **if** $t \geq 2T$ **and** $unlock = \mathsf{true}$ **then**
13: S($\mathcal{B}, \mathsf{pk}\mathcal{B}_A$)

Operations.

The list of possible operations is summarized in Table 6.1. We use OPX if the operation X succeeds and will mention explicit failure if it does not.

6.3.4 Protocol flow and guarantees

The protocol flow is described in Figure 6.2. It starts with $OP3$, an off-chain communication (black arrow). Alice then calls SC3 (blue arrow) while Bob learns σ'_2 (dashed blue arrow). The following action (*) can happen before or after $t = T$ and Bob then controls the X coins (dashed purple arrow). The final steps is $OP6$, where

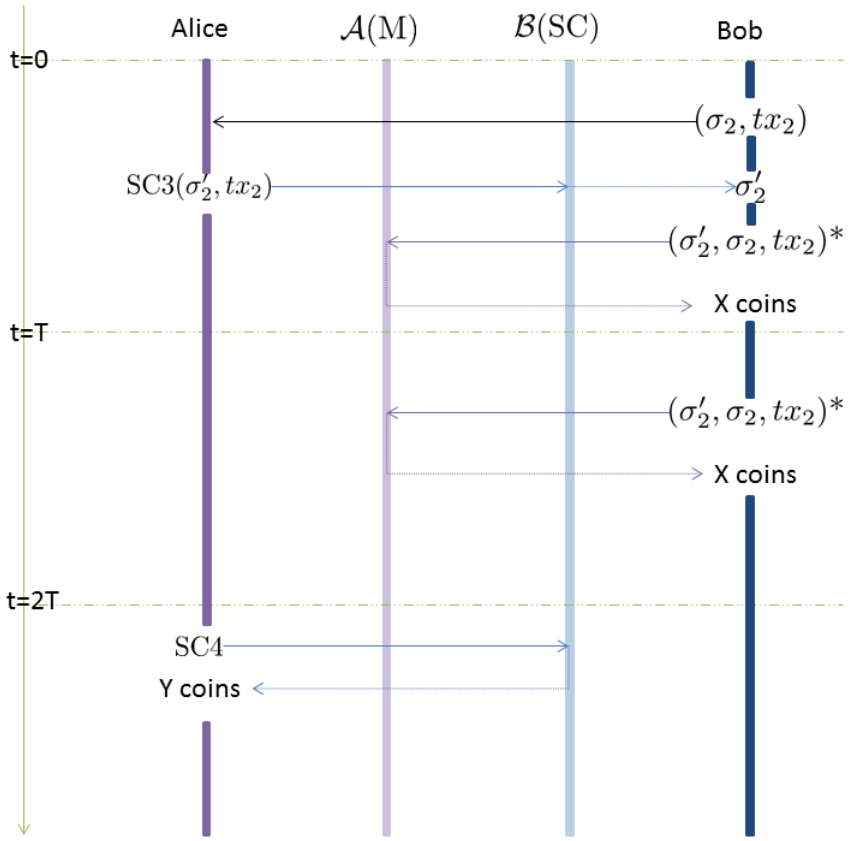

FIGURE 6.2 – Protocol flow resulting in a swap

Alice waits for $t \geq 2T$ and calls SC4 (blue arrow) to receive the Y coins (dashed blue arrow).

The protocol needs to ensure that for every outcome, swap or refund, for one user, there exists a sequence of operations the second user can do or could have done to have the same outcome, complete the swap or get a refund. Additionally, each user should get refunded if the other user aborts the swap or does nothing.

Swaps

Assuming $\mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B)$, i.e Bob received the X coins, we have :

$$\mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B) \implies OP5 \tag{6.1}$$

$$OP5 \implies OP4 \tag{6.2}$$

$$OP4 \implies unlock == True \tag{6.3}$$

$$\text{if } t \geq 2T \text{ and } OP6 \implies \mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A) \tag{6.4}$$

The step (3) relies on the fact that Bob can only learn σ_2' if Alice creates it, assuming $\mathcal{A}$ uses a secure signature scheme. The only way for Bob to access the Y coins is to use SC2 since $unlock == True$. He thus needs σ_1', which Alice has no reason to provide.

Assuming $\mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A)$, we have :

$$\mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A) \implies unlock == True \text{ from SC4} \tag{6.5}$$

$$unlock == True \implies OP4 \tag{6.6}$$

$$OP4 \implies OP5 \tag{6.7}$$

$$OP5 \implies \mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B) \tag{6.8}$$

Thus Bob can complete the swap. There is however one setting in which his part of the swap can fail : Alice has already taken the funds out of M, which is described in Figure 6.3. This is covered by the refund case.

Refunds

If Alice does no operation, Bob has to wait for $t \geq T$ to be refunded using $OP2$.

Alice, on the other end, can be refunded any time by using $OP1$. The problem is that Alice can refund herself and then try to get the funds on $\mathcal{B}$ using $OP2$ after $OP1$ while $t \leq T$, as described in Figure 6.3. This means that :

$$\text{Since } unlock == True \implies OP2 \text{ will fail when calling SC1} \tag{6.9}$$

But Alice does not have yet the funds from $\mathcal{B}$ and $T < t < 2T$.

$$\text{if } T < t < 2T \text{ and } OP7 \implies \mathrm{R}(\mathcal{B}, \mathsf{pk}\mathcal{B}_B) \tag{6.10}$$

6.4 Discussion

Cross-chain exchanges inherit the adversarial environment of each blockchain. This introduces multiple points of failure that we need to take into account to make the swaps atomic.

For example, Alice or Bob may have the means to launch an Eclipse attack [HKZG15]. One user would then have control over the other's network connections and decide which transactions reach the rest of the network. One user can also just have a better connectivity than the other and be sure his or her transactions would go through first, in the case of race. We can not solve such problems in the protocol without further assumptions thus we rely on the time locks the users set. They

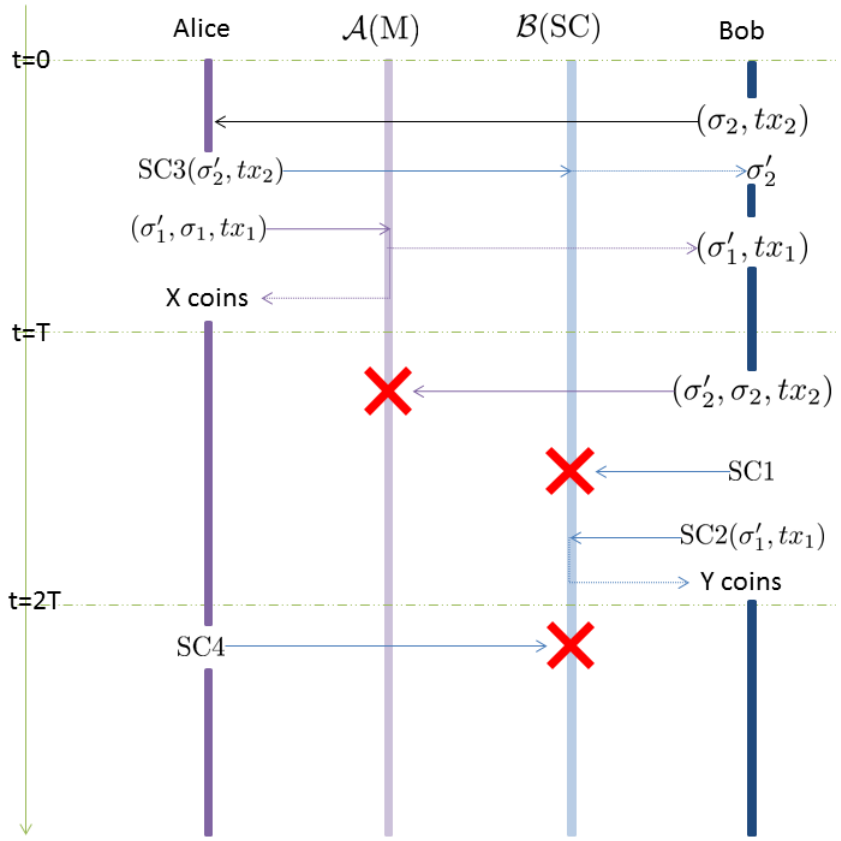

FIGURE 6.3 – Alice attack scenario

represent the intervals of time necessary for the users, in comparison to the block times, the confirmation times, to create, broadcast the relevant transactions and have them confirmed. In practice, they should also consider what is the current states of both blockchains as to not use them when they are more vulnerable. This could be the case if the mining hashing rate in Proof-of-Work, or the price of the coin in Proof-of-Stake crashes dramatically.

Another question is the pre-generation of the transactions that are in the smart contract M. This is relevant for security. It is better to require the signature of a *particular* message under a private key than *any* message under that key. This is also relevant for the fees of the transactions. Some blockchains do not take fees (Steem) or have very low fees (Litecoin) while other have a volatile fee market (Bitcoin, Ethereum). Alice can take care of this on $\mathcal{A}$ while Bob pays on $\mathcal{B}$. Still, if the fees were too low at creation, there is the risk that the transactions would take too long to appear in the blockchain(s).

6.5 Related works

The first atomic swap protocol emerged on a Bitcoin online forum and is based on HTLCs [Tie]. It has since been standardized on Bitcoin development resources [bit, BH]. This protocol is briefly presented in section 6.2.

To our knowledge, Herlihy [Her18] presented the first formal study of the underlying theory of atomic cross-chain swaps. It focuses on a swap of on-chain and off-chain assets between three parties. Using graph theory and the HTLCs, they explore the time locks constraints for the atomicity in that particular setting.

Many protocols focus on cross-chain communications and exchanges. Interledger [TS15] uses Hashed-Timelock Agreements (HTLAs) [dJS], which generalized the idea of HTLCs for payments systems with or without a private or public ledger. Those agreements are used to create secure multi-hop payments using conditional transfers. The parties involved decide where their trust lies, for example a HTLC between blockchains or a legal contract.

Cosmos [BKM18b, KB18], Polkadot [Woo14] and earlier versions of Interledger [TS15] rely on a set of validators to ensure cross-chain communication. Each round, a subset of those validators decides which cross-chain information to notarize on their chain, provided only a small protocol-defined portion of those validators is Byzantine. The problem is to guarantee that those systems are decentralized enough to be consider censorship resistant and secure while remaining scalable.

XClaim [ZHL+19] is a framework for achieving trustless cross-chain exchanges using cryptocurrency-backed assets based on a smart contract blockchain and an existing link between the other blockchains like a relay. It requires vault providers to guarantee the liquidity and ownership on the original chain and to have collateral on the smart contract blockchain. Bad behaviour is discouraged using slashing or proof-of-punishment on the smart contract chain. With these assumptions, XClaim builds a trustless and non interactive protocol for issuance, redemption and swapping of tokenized assets.

Our construction does not require collateral (because there is no vault to provide) or an existing link between the blockchains. We argue that our protocol uses the simpler assumptions to enable trustless exchange between users, in the same manner as HTLC based atomic cross-chain swaps.

6.6 Conclusion

In this work, we propose a new protocol for atomic cross-chain swaps. We extend the support of atomic swaps to blockchains without hash lock and time lock capabilities

without additional trust requirements. Users can now trade in a P2P way between blockchains with smart contracts and blockchain with only mutli-signatures transactions. We leave the full implementation and live experimentation of this protocol for future work.

Conclusions and perspectives

Contributions

In this manuscript we have proposed an functionning experimental platform (Chapter 4), which was used to benchmark the scalability of blockchains, based on metrics proposed in Chapter 3, using the Multichain implementation (a fork of Bitcoin). Using a realistic number of nodes, and realistic setting corresponding to countries in the context of the Orange Money application, we showed (Chapter 5) how to experimentally measure the *real* number of transactions per second, achieving a very reasonable 350 tx per sec in a 5 node context.

We have also discussed the possibility of using different blockchains and how it is possible to exchange currencies between them (Chapter 6).

Perspectives

The main experimental perspective of this work would be to extend our experimental testing to different kinds of blockchains. This should be reasonably feasable, since the design proposed in Chapter 4 lets us integrate many kinds of blockchains. Running the experiments is however quite a long task (spanning months, since many experiments must be run for tens of hours), which is the reason we decided to restrain ourselves to only one kind of blockchain.

Bibliographie

[ABB+18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric : a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018. [cited p. 24]

[B+02] Adam Back et al. Hashcash-a denial of service counter-measure. 2002. [cited p. 13]

[BG17] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv :1710.09437*, 2017. [cited p. 40]

[BH] Sean Bowe and Daira Hopwood. Hashed time-locked contract transactions. https://github.com/bitcoin/bips/blob/master/bip-0199.mediawiki, Mar. 2017. Accessed : 2019-07-03. [cited p. 85]

[bit] bitcoinwiki. Atomic swap. https://en.bitcoin.it/wiki/Atomic_swap, 2019. Accessed : 2019-07-03. [cited p. 85]

[BKM18a] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on bft consensus, 2018. [cited p. 40]

[BKM18b] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938, 2018. [cited p. 79, 85]

[BMC+15] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok : Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121, May 2015. [cited p. 20, 24]

[Bre00] Eric A. Brewer. Towards robust distributed systems (abstract). In Gil Neiger, editor, *Proceedings of the Nineteenth Annual ACM Symposium on Principles*

of Distributed Computing, July 16-19, 2000, Portland, Oregon, USA, page 7. ACM, 2000. [cited p. 11]

[BSA14] Alysson Bessani, JoÃ£o Sousa, and Eduardo Alchieri. State machine replication for the masses with bft-smart. pages 355–362, 06 2014. [cited p. 40]

[BSA+19] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Sok : Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland, October 21-23, 2019*, pages 183–198. ACM, 2019. [cited p. 20]

[BSAB+17] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. SoK : Consensus in the Age of Blockchains. https://arxiv.org/abs/1711.03936, 2017. [cited p. 24]

[CDE+16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016. [cited p. vii, 19, 23, 24, 31, 33, 43, 59]

[CDFZ17] Alexander Chepurnoy, Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. Twinscoin : A cryptocurrency via proof-of-work and proof-of-stake. *IACR Cryptology ePrint Archive*, 2017 :232, 2017. [cited p. 40]

[CL99] Miguel Castro and Barbara Loskov. Practical byzantine fault tolerance, 1999. [cited p. 22, 40]

[CV17] Christian Cachin and Marko Vukolic. Blockchain consensus protocols in the wild. *CoRR*, abs/1707.01873, 2017. [cited p. vii, 24]

[DFZ16] Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. 2-hop blockchain : Combining proof-of-work and proof-of-stake securely. *Cryptol. ePrint Arch., Tech. Rep*, 716 :2016, 2016. [cited p. 40]

[DGH+87] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In Fred B. Schneider, editor, *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10-12, 1987*, pages 1–12. ACM, 1987. [cited p. 12]

[DGKR18] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos : An adaptively-secure, semi-synchronous proof-of-stake blockchain.

In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018. [cited p. 29]

[dJS] Michiel de Jong and Evan Schwartz. Hashed-timelock agreements. https://github.com/interledger/rfcs/blob/master/0022-hashed-timelock-agreements/0022-hashed-timelock-agreements.md, Nov. 2017. Accessed : 2019-07-03. [cited p. 85]

[DLS88] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2) :288–323, 1988. [cited p. 10, 11]

[DMEN17] Patrick Dai, Neil Mahi, Jordan Earls, and Alex Norta. Smart-contract value-transfer protocols on a distributed mobile application platform, 03 2017. [cited p. 40]

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992. [cited p. 13]

[DPS17] Phil Daian, Rafael Pass, and Elaine Shi. Snow white : Robustly reconfigurable consensus and applications to provably secure proofs of stake. In *Iacr*, pages 1–64. 2017. [cited p. 29, 79]

[DSW16] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 13. ACM, 2016. [cited p. 22, 40]

[DW13] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sep. 2013. [cited p. 23, 29, 30, 32, 43]

[DW15a] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In Andrzej Pelc and Alexander A. Schwarzmann, editors, *Stabilization, Safety, and Security of Distributed Systems - 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings*, volume 9212 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015. [cited p. 22]

[DW15b] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In Andrzej Pelc and Alexander A. Schwarzmann, editors, *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18, Cham, 2015. Springer International Publishing. [cited p. 76]

[DWC+17] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. BLOCKBENCH : A framework for analyzing private blockchains. *CoRR*, abs/1703.04057, 2017. [cited p. 24, 41]

[EGSR16] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert Van Renesse. Bitcoin-ng : A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59, Santa Clara, CA, 2016. USENIX Association. [cited p. 22]

[eos] Eos.io technical white paper v2. [cited p. 40, 77]

[ES18] Ittay Eyal and Emin Gün Sirer. Majority is not enough : Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7) :95–102, 2018. [cited p. 79]

[ESSN14] Ittay "Eyal, editor="Christin Nicolas Sirer, Emin Gün", and Reihaneh" Safavi-Naini. "majority is not enough : Bitcoin mining is vulnerable". In *Financial Cryptography and Data Security*, pages 436–454, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. [cited p. 24]

[eth] Ethereum whitepaper. [cited p. 24, 40, 77]

[FB99] A. Fox and E.A. Brewer. Harvest, yield, and scalable tolerant systems. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*, pages 174–178, 1999. [cited p. 11]

[FLP85] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. volume 32, pages 374–382. ACM, 1985. [cited p. 11]

[FUN] INTERNATIONAL MONETARY FUND. Financial access survey 2020 trends and developments. [cited p. 4, 100, 105]

[GBE+18] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. Decentralization in bitcoin and ethereum networks. *CoRR*, abs/1801.03998, 2018. [cited p. 23]

[GHM+17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand : Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017. [cited p. 29, 40]

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol : Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. [cited p. 16]

[GKW+16] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016. [cited p. 16, 24, 29, 33, 43, 59, 79]

[GL02] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2) :51â59, jun 2002. [cited p. 11]

[GMSR+20] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok : Layer-two blockchain protocols. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 201–226, Cham, 2020. Springer International Publishing. [cited p. 20]

[GRKC15] Arthur Gervais, Hubert Ritzdorf, Ghassan O Karame, and Srdjan Capkun. Tampering with the delivery of blocks and transactions in bitcoin. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 692–705. ACM, 2015. [cited p. 24]

[Her16] Maurice Herlihy. Asynchronous consensus impossibility. In *Encyclopedia of Algorithms*, pages 152–155. 2016. [cited p. 11]

[Her18] Maurice Herlihy. Atomic cross-chain swaps. In Calvin Newport and Idit Keidar, editors, *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 245–254. ACM, 2018. [cited p. 85]

[HKZG15] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 129–144, 2015. [cited p. 24, 83]

[JJ99] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Secure information networks*, pages 258–272. Springer, 1999. [cited p. 13]

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1) :36–63, 2001. [cited p. 79]

[KB18] Jae Kwon and Ethan Buchman. Cosmos : A network of distributed ledgers, 2018. [cited p. 85]

[KGC+18] Harry A. Kalodner, Steven Goldfeder, Xiaoqi Chen, S. Matthew Weinberg, and Edward W. Felten. Arbitrum : Scalable, private smart contracts. In William

Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 1353–1370. USENIX Association, 2018. [cited p. 23]

[KJG+18] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger : A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 583–598. IEEE Computer Society, 2018. [cited p. 21]

[KKJG+18] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger : A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018. [cited p. vii]

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros : A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017. [cited p. 29, 79]

[Lee11] Charles Lee. Litecoin, 2011. [cited p. 33, 77]

[LLZ+18] Chenxing Li, Peilun Li, Dong Zhou, Wei Xu, Fan Long, and Andrew Yao. Scaling nakamoto consensus to thousands of transactions per second, 2018. [cited p. 21]

[LLZ+20] Chenxing Li, Peilun Li, Dong Zhou, Zhe Yang, Ming Wu, Guang Yang, Wei Xu, Fan Long, and Andrew Chi-Chih Yao. A decentralized blockchain with high throughput and fast confirmation. In Ada Gavrilovska and Erez Zadok, editors, *2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020*, pages 515–528. USENIX Association, 2020. [cited p. 21]

[LNE+19] Joshua Lind, Oded Naor, Ittay Eyal, Florian Kelbert, Emin Gün Sirer, and Peter R. Pietzuch. Teechain : a secure payment network with asynchronous blockchain access. In Tim Brecht and Carey Williamson, editors, *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*, pages 63–79. ACM, 2019. [cited p. 22]

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3) :382â401, jul 1982. [cited p. 10]

[LSZ+16] Daniel Larimer, Ned Scott, Valentine Zavgorodnev, Benjamin Johnson, James Calfee, and Michael Vandeberg. Steem : An incentivized, blockchain-based social media platform. *March. Self-published*, 2016. [cited p. 79]

[Mul] Multichain. [cited p. 38, 47]

[Nak] Satoshi Nakamoto. Cryptography mailing list : Bitcoin announcement. [cited p. 14, 19, 25]

[Nak08] Satoshi Nakamoto. Bitcoin : A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008. [cited p. vii, 1, 13, 14, 15, 16, 24, 28, 40, 76, 79]

[Ope] OpenZeppelin. Ecdsa solidity library. https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/cryptography/ECDSA.sol. Accessed : 2019-07-03. [cited p. 79]

[PB17] Joseph Poon and Vitalik Buterin. Plasma : Scalable autonomous smart contracts, Aug 2017. Accessed : 2017-08-10. [cited p. 22]

[PD16] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network, 2016. Accessed : 2016-07-07. [cited p. 22, 76]

[Poe14] Andrew Poelstra. Distributed consensus from proof of stake is impossible. *URL : https ://download. wpsoftware. net/bitcoin/old-pos. pdf*, 2014. [cited p. 29]

[PS17] Rafael Pass and Elaine Shi. Hybrid consensus : Efficient consensus in the permissionless model. In *31st International Symposium on Distributed Computing (DISC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. [cited p. 40]

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 643–673, Cham, 2017. Springer International Publishing. [cited p. 16]

[QZLG21] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the defi ecosystem with flash loans for fun and profit. In Nikita Borisov and Claudia Díaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2021. [cited p. 6]

[Rei96] Michael K Reiter. A secure group membership protocol. *IEEE Transactions on Software Engineering*, 22(1) :31–42, 1996. [cited p. 22]

[SCD+] Kazuko Shirono, Esha Chhabra, Bidisha Das, Yingjie Fan, and Hector Carcel Villanova. Is mobile money part of money ? understanding the trends and measurement. [cited p. 4, 99, 100, 104]

[SLZ16] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre : A fast and scalable cryptocurrency protocol. *IACR Cryptol. ePrint Arch.*, 2016 :1159, 2016. [cited p. 21]

[SZ15] Yonatan Sompolinsky and Aviv Zohar. *Secure High-Rate Transaction Processing in Bitcoin*, pages 507–527. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. [cited p. 21]

[Tie] TierNolan. bitcointalk : Alt chains and atomic transfers. https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949, May 2013. Accessed : 2019-07-03. [cited p. 76, 77, 85]

[Til20] Andrew Tilton. The what and why of digital currencies, 2020. [cited p. 17]

[TS15] Stefan Thomas and Evan Schwartz. A protocol for interledger payments. *https://interledger.org/interledger.pdf*, 2015. [cited p. 85]

[Wat16] Roger Wattenhofer. *The science of the blockchain*. Inverted Forest Publishing, 2016. [cited p. 10, 11]

[WG18] Karl Wuest and Arthur Gervais. Do you need a blockchain ? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018. [cited p. 17]

[Woo14] Gavin Wood. Polkadot : vision for a heterogeneous multi-chain framework, 2014. [cited p. 85]

[ZDBN19] Jean-Yves Zie, Jean-Christophe Deneuville, Jérémy Briffaut, and Benjamin Nguyen. Extending atomic cross-chain swaps. In Cristina Pérez-Solà, Guillermo Navarro-Arribas, Alex Biryukov, and Joaquín García-Alfaro, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings*, volume 11737 of *Lecture Notes in Computer Science*, pages 219–229. Springer, 2019. [cited p. viii, 3, 75]

[ZHL+19] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. Xclaim : Trustless, interoperable, cryptocurrency-backed assets. 03 2019. [cited p. 85]

Annexe

Annexe A

Mobile Money

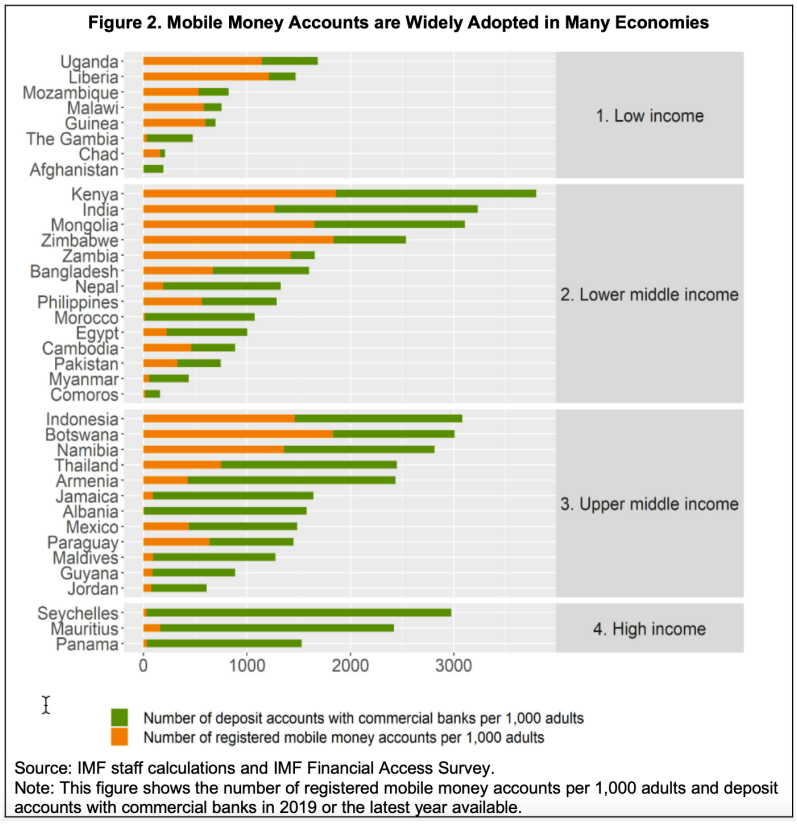

FIGURE A.1 – Mobile Money accounts[SCD+]

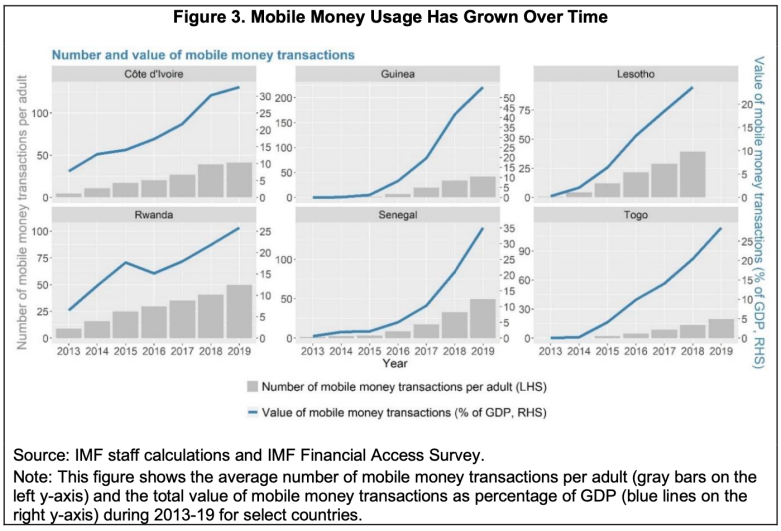

FIGURE A.2 – Mobile Money usage[SCD+]

TABLE A.1 – The number of registered mobile money accounts per 1,000 adults from [FUN]

		2015	2019
Europe and Central Asia	Armenia	118	428
	Romania	18	11
Middle East and North Africa	Egypt	63	224
	Qatar	98	228
Latin America and Caribbean	Guyana	22	87
	Mexico	84	437
East Asia and Pacific	Cambodia	42	459
	Fiji	991	2250
South Asia	India	73	1265
	Pakistan	120	328
Sub-Saharan Africa	Burkina Faso	224	1049
	Kenya	1129	1859

Annexe B

Codes listings

B.1 Benchmark script

```
#!/bin/bash

bench(){
  NBTESTPERSECOND=$1
  NBTEST=$2
  echo "$(date) - $(hostname) -
  Bench $NBTESTPERSECOND tx/s during $NBTEST test"
  >> /mnt/nfs/DATASTORE/multichaind-logs/bench-logs.log
  for j in `seq 1 $NBTEST`
  do
  for i in `seq 1 $NBTESTPERSECOND`
  do
  # tx size is close to 250 bytes
  multichain-cli sensors-data publish temperature
  datatofillthechain 10
  done
  sleep 1
  done
  sleep 60
}

bench 1 100
bench 5 100
bench 10 100
bench 25 100
bench 50 100
bench 100 100
bench 1000 100
```

B.2 Multichain mining-diversity

```
1  //...
2  int wSize=10;
3  //...
4  double wBlockTime[10];
5  //...
6
7  double wAvTimePerBlock=0;
8  for(int w=0;w<wSize;w++)
9  {
10   wAvTimePerBlock+=wBlockTime[w];
11 }
12 wAvTimePerBlock/=wSize;
13
14 canMine=MC_PTP_MINE;
15 if(mc_TimeNowAsDouble() < GetMinerAndExpectedMiningStartTime(
16 pwallet, &kMiner,&sMinerPool,
17 &dMiningStartTime,&dActiveMiners,&hLastBlockHash,
18 &nMemPoolSize,wAvTimePerBlock)
19 )
20 {
21   if( !fMineEmptyBlocks && not_setup_period
22   && (mempool.hashList->m_Count == 0))
23   {
24     nMiningStatus|=MC_MST_NO_TXS;
25   }
26   else
27   {
28     nMiningStatus|=MC_MST_SLEEPING;
29     //<=== no mining before
30     //GetMinerAndExpectedMiningStartTime(...)
31   }
32   canMine=0;
33 }
34 else
35 {//...
36
```

Table des figures

Liste des tableaux