UMEÅ UNIVERSITY
Department of Computing Science

March 4, 2019

Emergent Routing Policies in the Lightning Network

Abstract

In payment channel network, such as the lightning network, the routing nodes receives a fee as compensation for displaced liquidity, time value of money and operational costs. Currently this fee is mostly manually set procuring an unoptimized profit to the node operator. TODO: TYPES OF POLICIES HERE, NOT ONLT PRICE BIT PREFERENTIAL ATTACHMENT, RECEIVED CHANNELS AND OPTIMAL STRATEGY AS PRODUCT OF OTHERS STRATEGIES; HENCE GAME THEORY.

A simulation study was performed where many policies was pitted against each other to find emergent equilibria under competitive market pressure. Were such equilibrium may lie have further consequences for the network in form of total throughput, routing cost and robustness.

Name	John-John Markstedt
Cinnober Supervisor	Oskar Janson
University Supervisor	Jerry Eriksson
Examinator	Henrik Björklund

Acknowledgements

Curabitur ac lorem. Vivamus non justo in dui mattis posuere. Etiam accumsan ligula id pede. Maecenas tincidunt diam nec velit. Praesent convallis sapien ac est. Aliquam ullamcorper euismod nulla. Integer mollis enim vel tortor. Nulla sodales placerat nunc. Sed tempus rutrum wisi. Duis accumsan gravida purus. Nunc nunc. Etiam facilisis dui eu sem. Vestibulum semper. Praesent eu eros. Vestibulum tellus nisl, dapibus id, vestibulum sit amet, placerat ac, mauris. Maecenas et elit ut erat placerat dictum. Nam feugiat, turpis et sodales volutpat, wisi quam rhoncus neque, vitae aliquam ipsum sapien vel enim. Maecenas suscipit cursus mi.

Quisque consectetuer. In suscipit mauris a dolor pellentesque consectetuer. Mauris convallis neque non erat. In lacinia. Pellentesque leo eros, sagittis quis, fermentum quis, tincidunt ut, sapien. Maecenas sem. Curabitur eros odio, interdum eu, feugiat eu, porta ac, nisl. Curabitur nunc. Etiam fermentum convallis velit. Pellentesque laoreet lacus. Quisque sed elit. Nam quis tellus. Aliquam tellus arcu, adipiscing non, tincidunt eleifend, adipiscing quis, augue. Vivamus elementum placerat enim. Suspendisse ut tortor. Integer faucibus adipiscing felis. Aenean consectetuer mattis lectus. Morbi malesuada faucibus dolor. Nam lacus. Etiam arcu libero, malesuada vitae, aliquam vitae, blandit tristique, nisl.

Contents

Chapter 1 | Problem Description

1.1 Background

(When a majority of the text is written, this will be filled in.)

1.2 Aim

This thesis aim to address following three questions:

1. What is the fair discount for a transaction that balances the channel, as opposed to an unbalancing one?
2. When should a new channel be opened, and what node should the new channel connect to in order to incentivize routing through the node?
3. When should a manual rebalancing of the channels occur, as opposed to incentivized rebalancing using discounts?

1.3 Related work

Little to nothing in the literature that addresses the Lightning Network Routing Policies has been found. Rene Pickard have made some early attempts to find suitable channel parties with traditional graph heuristics but is yet incomplete and unpublished[How source to unpublished?]. Further conversation with people in the community confirms that this problem set is known yet largely unexplored.

The actual routing in LN has been researched[source] and implemented[source]. Further efficient sharing of routing tables have been proposed[31].

A similar problem at first glance, the estimation of the on-chain Bitcoin Network Fee has been covered extensively. However it's not applicable, nor is it even similar as a problem set. The Bitcoin Fee problem is far simpler as the competing transaction fees(mem-pool) are known at all times and estimating a fee too low can be corrected by *fee bumping*[1] or *child-pay-for-parent*[2] implementations.

As the Lightning Network is indeed a network and many problems is yet another incarnation of already solved graph problems. A nodes position in a network has been studied and specifically the measurement of Betweenness Centrality, as introduced by Freeman in 1977[23], lays a sound basis for this thesis. Calculating paths in graphs is as old as the computing field itself with Floyd-Warshall, Johnson[35] and Dijkstras algorithms all being appropriate here. The attributes of the network as policies emerges have consequences on throughput and robustness have also been extensively researched in the field of topology. Especially that of scale free networks by Barabasi and Albert[21] is of concern in here.

Emergence of equilibria as the product of policies of competing actors is neither new or rare in the fields of Evolutionary Biology and Economics. Ideas as the Standard Price Theory, Evolutionary Stable Strategies and more widely Game Theory has aided in the formulation of the thesis and construction of the simulations.

Lastly, bitcoin and off-chain proposals have opened a plethora of possible research topics in which fascinating research is published every day.

1.4 Structure

This thesis is divided into 8 chapters, including this one. The first four chapters gives a wide overview of Bitcoin, the Lightning Network. Chapters 6 and 7 aims to answer the thesis question by formulating, evaluating and simulating routing policies. The last chapter 8 discusses the result and conclusions are drawn and elaborating on future work.

1. **Introduction** introduces the thesis.
2. **Macroeconomics** gives a wide background to currencies and systems built on trust.
3. **The Bitcoin Network** chapter introduces the basic bitcoin architecture, answers how trust is solved and describes bitcoin script as it allows further development as a base protocol.

[1] Broadcasting a new transaction spending the same UTXO with a higher fee.

[2] Using the unconfirmed parent UTXO in a new transaction, a miner must include the parent tx to be able to mine the child tx and receive the child tx fee.

4. **Scaling Bitcoin** describes historical attempts to scale bitcoin and the merit of a wide array of scaling solutions.

5. **The Lightning Network** chapter introduces the most fundamental parts of the Lightning protocol.

6. **Evaluation** reduces the Lightning Network to a manageable problem set, formulates policies and measurements how these policies may be evaluated.

7. **Results** shows the simulated results of the policies and their effect on network as a whole in terms of throughput, efficiency and robustness.

8. **Discussion and Conclusions** Processes the results, draws conclusions and ties them into the wider ecosystem. The consequences and viability of the Lightning Network is elaborated and future work discussed.

Chapter 2 | Systems of Trust

Bitcoin was not created in a vacuum and to understand the design of Bitcoin and by extension the Lightning Network one would benefit from understanding money and systems of trust. Nakamoto encoded the string "the times 03/jan/2009 chancellor on brink of second bailout for banks"[12][11] in the coinbase transaction of the genesis block as an alleged comment on the current banking system as well as a time stamp. He later cemented this view with multiple forum posts while he was still active[47][48].

2.1 Origin of money

On the fringes of our specie's history barter became the first medium in which trade became viable, removing the risk from otherwise delayed reciprocity. It is possible for trade to be mutually beneficial as Nick Szabo so elegantly puts it:

> "individuals, clans, and tribes all vary in their preferences, vary in their ability to satisfy these preferences, and vary in the beliefs they have about these skills and preferences and the objects that are consequent of them, there are always gains to be made from trade."[58]

Barter is by itself quite limited, consider a System **S** consisting of **n** commodities, the amount of possible exchanges between commodities in **S** is **n x n**. When **n** eventually increases, the exchange pairs explodes exponentially. This makes it difficult to assess fair pricing and decreases the coincidence of a trade[1]. The economist Carl Menger described it as inevitable for money to evolve from a sufficient volume of commodity barter[43]. If the same System **S** is considered with money - the possible exchange pairs would be reduced to only **n** pairs.

2.1.1 Characteristics of money

Out of these commodities money emerged and history has provided many peculiar forms of money. The Rai stones of Yap islands, The Wampum shells in North America[2][58] and Aggry beads in Africa to name a few. Although many different types of commodities have acted as money during certain time in history there are some characteristics that seem favorable and recur. The Federal Reserve Bank of Saint Louis lists them as[50][3]:

- **Durability**. The ability to remain intact over time.
- **Portability**. The ability to move across physical distance.
- **Divisibility**. The ability to be divided across scale, to be used in any size of value transaction.
- **Uniformity**. The ability to use one unit interchangeably with another. Also referred to as **fungibility**.
- **Limited supply**. The ability to be scarce over time. Usually quantified by stock-to-flow. The amount of new supply in relation to already existing supply.
- **Acceptability**. The likelihood of being accepted in trade by others.

Many of the previous mentioned monies have expressed many of these characteristics well. Changes in these characteristics have also led to their downfall. Rai stones are carved limestone which are not native to the Yap islands, when outsiders started to bring in these on large ships it quickly changed their stock-to-flow ratio for the worst[55]. Aggry beads met the same fate when Europeans, with efficient means to produce them, started to export them to West Africa and modernized shell fishing ruined the Wampum shells function as money[58].

Rare metals have held these attributes since the beginning of history and in many ways still hold them today. Gold especially stands out with it's very low stock-to-flow ratio.

[1] The coincidence of finding another party who is looking to exchange the same commodity pair you are but in reverse order.

[2] The Wampum shells were actually legal tender as recently as 1710 in North Carolina and long into the 1600s in New England.

[3] Note that only the characteristics are from the Federal Reserve, the descriptions are not.

2.2 Bearer promissory notes

Although gold emerged as the commodity best suited as money it still had two major problems:

1. Gold is unsuitable for small transactions due to the limit in divisibility.
2. Gold is expensive to carry around and protect.

Private and central banks began to issue bearer promissory notes to the owners of the underlying asset it stored(e.g figure 1). The asset could then be retrieved for the note on demand. The notes themselves could then be traded instead of the underlying gold. This solved both the above problems, notes are easy to carry, conceal and could be minted in very small amounts.

Figure 1: *The private bank 'Stockholms Enskilda Bank'(SEB) issued bearer promissory note. It promises to pay 10 crowns to the bearer upon demand in gold. The exchange rate as per the Scandinavian Convention(Sweden, Denmark 27 may 1873, Norway 1 april 1877)[40] was 1 kilogram gold per 2480 crowns or 1 crown per 0.403g gold[39].*

By the late 19th century almost all major currencies was under the gold standard as seen in table 1.

Nation	Currency	Period	Years
France	Franc	1814-1914	100
Netherlands	Guilder	1816-1914	98
G.Britain	P. Sterling	1821-1914	93
Switzerland	Franc	1850-1936	86
Belgium	Franc	1832-1914	82
Sweden	Kronor	1873-1931	58
Germany	Mark	1875-1914	39
Italy	Lira	1883-1914	31

Table 1: *Major European nations periods under the gold standard as composed by Ferdinand Lips[41] from 1975 Pickís Currency Yearbook data.*

2.2.1 Bank runs

The introduction of the promissory note solved the two previously mentioned problems but also introduced trust.[4]

Although the gold deposits could be regularly audited by a third party there is little to no guarantee that the bank in question wouldn't issue more notes than gold it holds in reserve. Banks utilizing this sort of practice could go on for a very long time before getting caught. Consider a bank that issue twice as many notes as it holds gold in reserve. It would require 50% of it's notes to be demanded before the bank would default.

There has been many so called bank runs in history. The public gets suspicious about the bank and the trust disappears - leading to massive withdrawals in a short time span. Even if a bank is technically solvent, having assets tied up in different ventures, they could fail to deliver on the notes promises. This can also be viewed as a negative spiral, people start to withdraw, increasing the risk of default, more people start to withdraw due to the new higher risk. R.H. Patterson elaborates in detail the bank run in Great Britain in 1866 leading to the default of Overend, Gurney and Company and the behavior under panic:

> "When a Panic occurs, a much more serious home-drain is produced upon the bank. At such time cheques fall somewhat into disrepute, so that merchants in some cases require payment in cash. The public also, to some extent, take to hoarding[...]. But a very large part of the drain upon the Bank of England in the form of hoarding ,[...],is made by other banks: for these banks being liable to unusual demands on the part of their customers, have to keep in hand a larger stock of money than usual"[51]

Overall most larger institutions stayed solvent and bank runs is more an exception than a rule during this time period. Since all major currencies were denominated in gold the threshold for global trade sank. During the gold standard era the economy boomed with trade and is often known under the French term 'La Belle Epoque' or 'Beautiful era'.

[4] Note that this a much wider problem than for only promissory notes. This trust model is what makes banks in the first place.

2.3 Elasticity of money

It all came crashing down with the outbreak of the first world war. Nations central banks began to issue more notes than gold in reserve to fund the war effort. It effectively ended the gold standard and countries not affected by the war followed shortly after.

As response to the Great Depression U.S President Franklin Roosevelt issued Executive order 6102 confiscating all gold coins, bullion and certificates and banned trade in gold[53]. The U.S Dollar was devalued the following year from $20.67 per troy ounce to $35 under the Gold Reserve Act[54] to enable spending their way out of recession.

A new economic era began following ideas of Maynard Keynes of economic interventionism and monetary policies set to aid growth. Keynes rebutted the classical idea that 'supply creates its own demand' and that aggregated demand and aggregated supply may get stuck in an equilibrium with high unemployment[37]. This would motivate government intervention to increase the aggregated demand. While demand may be altered by a government in multiple ways the far most effective one is by altering the money supply[5]. Although the dollar was still technically redeemable for gold the elastic money supply made it possible to steer the economy by means of monetary policy.

The dollar was re-pegged multiple times between 1968-1971 as part of the Nixon chock[22]. In 1971 Nixon ended the Bretton Woods agreement[22], the international agreement of exchange rates between currencies, rendering the dollar and all currencies pegged to the Dollar true fiat currencies. After the initial shock the free floating currencies failed to keep pace with gold due to money supply increases causing inflation(Se figure 2).

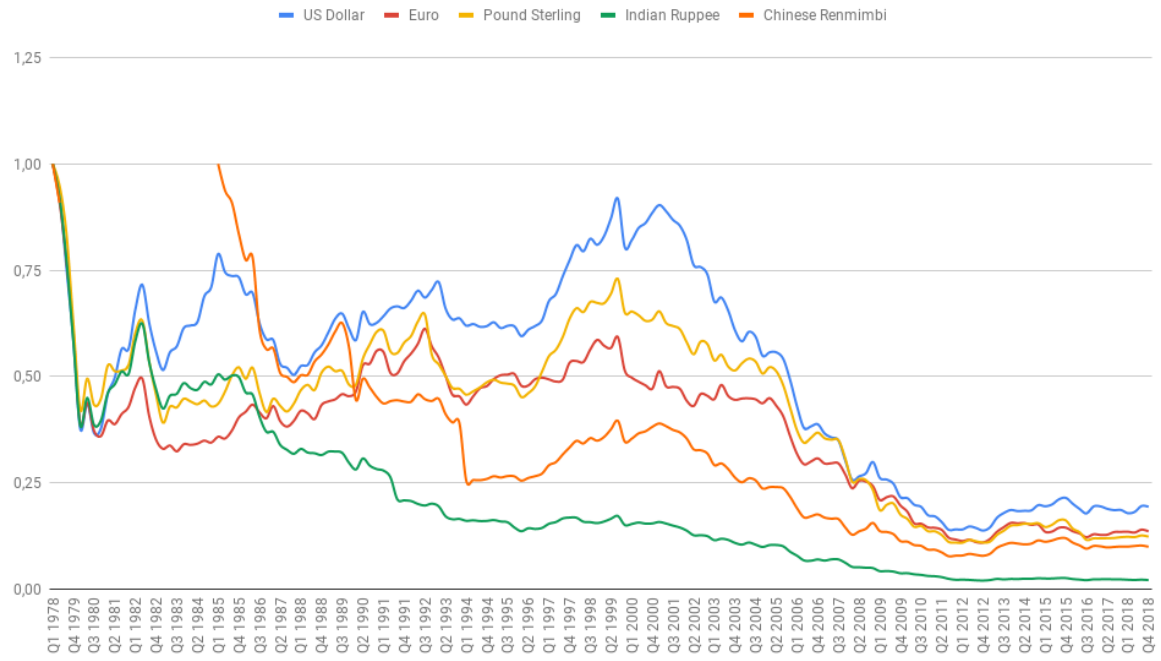

Figure 2: *The relation of major currencies against its value in gold since the end of the Bretton Woods agreement to today. The graph is composed with data from the World Gold Council[14].*

[5]The increased supply can be used directly by government but is usually used by lowering the funds rate, increasing investments and thus demand. Quantitative easing is another term used to describing this method.

Nation	Currency	Period	Highest monthly Inf.	Avg. daily Inf.	Type of index
Hungary	Pengő	Aug. 1945 - Jul. 1946	4.19×10^{16}%	207%	Consumer
Zimbabwe	Dollar	Mar.2007 - Mid-Nov.2008	7.96×10^{10}%	98.0%	Implied Exchange
Yugoslavia	Dinar	3 Apr.1992 - Jan.1994	313,000,000%	64.6%	Consumer
Germany	Papiermark	Aug. 1922 - Dec.1923	29,500%	20.9%	Wholesale
China	Yuan	Oct.1947 - Mid-May.1949	5,070%	14.1%	Wholesale
Venezuela(§)	Bolívar	Jan. 2018 - ongoing	-	4.1%	-

Table 2: *Hyperinflation of selected fiat currencies. From the Hanke-Krus Hyperinflation Table [33]. §The Venezuela numbers are only a projection by the International Monetary Fund and the dates and rate will most likely change with more accurate data[32] and when the inflationary period is over.*

2.3.1 Hyperinflation

The major currencies seen in figure 2 is very far from worst in their ability to hold value. Fiat currencies are highly dependent on it's authority and their willingness to increase the money supply and history supplies us with some catastrophes as seen in table 2.

During the last 250 years hyperinflation[6] have occured at least 54 times[33][32]. Many countries going through multiple inflationary periods, e.g. China 43-45, 47-49 and Georgia 92, 93-94. Hyperinflation devastates all savings and effectually letting savings subsidize the newly printed currency. The inflation narrows the time horizon of trade, since capital will go worthless in a short while, making it very difficult to formalize any long term investment or running a normal economy.

2.4 Trust

The economic system built upon fiat currencies are based on the trust in the issuing body not to increase the money supply excessively. It's almost been common knowledge that value is lost in currencies, moving money away from currency and bonds into stock and real estate. Modern currencies have been decoupled from the underlying reasons of its nascence, leading up to the failures of 08 coinciding with Nakamoto's publication completing full circle.

2.4.1 Relevancy

In many ways it helps to think of bitcoin as both a bearer instrument and as a monetary policy and how the system is designed as such. This section is an incomplete view on it's own and for further reading these recommendations are good starting points:

- Nick Szabo's blog Enumerated and above cited Shelling out[58][56].
- Saifedean Ammous's The bitcoin standard.[55]
- Waren Weber's paper A bitcoin standard.[59]

[6] There is no official definition of what constitutes hyper inflation but a rule of thumb seem to be over 50% annual inflation.

Chapter 3 | The Bitcoin Network

Bitcoin is a peer-to-peer electronic cash system without any trusted third party and was described by Satoshi Nakamoto in Oktober 2008[46]. The name Bitcoin refers to both the operating network of nodes and the system's native currency. The network and protocol is usually refereed to with a large B, Bitcoin, and the currency with a small b, bitcoin.[1]

3.1 Digital Signatures

Part of the solution is provided by one way asymmetric encryption utilizing the `secp256k1` elliptic curve. Asymmetric encryption or public-key encryption allows the generation of key pairs with the ability to derive a public key from a private key without the ability to derive the private key from the public key and the ability to prove ownership of the private key through signatures without disclosing the private key. Ownership and expenditure of bitcoin is proved by signing a previous transaction output payable to the public key derived bitcoin address[2] with the accompanying private key.

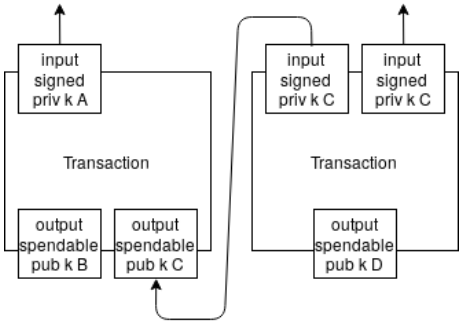

Figure 1: *A simplified illustration of two transactions.*

Bitcoin is essentially a long ledger of inputs and outputs. Note that the whole output must be spent in the same transaction and usually there is an output leading back to an address the spender controls with the 'change'. The sum of all the inputs and the sum of all the outputs usually does not line up, the differential is the implicit miners fee the miner receives if the transaction is included in a block.

3.2 The Double-Spending problem

While digital signatures are able prove ownership of bitcoin. There is nothing to stop the owner of an amount of bitcoin to sign two different transactions spending the same transaction output and broadcasting them to different parts of the network. This is called the double-spending problem which is a variation of the more general Byzantine's General Problem.

The network reaches a coherent global view over the transaction history by a method known as *Proof-of-work*. The Bitcoin *Proof-of-work* algorithm is very similar to that of hashcash as it uses a hash-based cost function which requires a probabilistic amount of work and the proof of the work fixed cost to verify[20][3].

The *Proof-of-work* algoithm is directly tied into the bitcoin transaction history data structure composite of a chain of blocks as seen in figure 2.

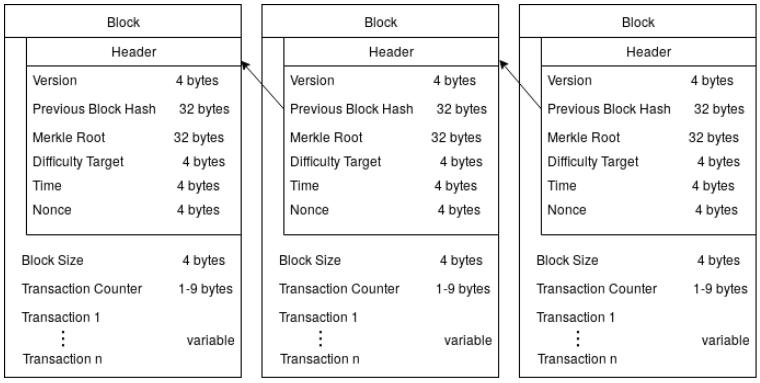

Figure 2: *Three blocks linked by the hash of the previous block's header. The content in the header is immutable and can't be replaced without redoing Proof-of-work.*

[1] While it's convention to not capitalize currencies in the English language, it's not been widely followed in mainstream or semi-mainstream outlets. Also the plural form of currencies is equivalent to the singular. The convention will be followed in this report.

[2] A bitcoin address is derived from the public key by hashing the public key with both `SHA256` and `RIPEMD160`

[3] It's similar to hashcash in both being hash-based with a cost function that is non-interactive, publicly auditable, trapdoor-free and have an unbounded probabilistic cost[20]. Bitcoin uses the double `SHA256` hash function.

The 'work' part of the algorithm is to find a sufficiently small hash of the previous block's header. A hash is valid if it's smaller than a number imposed by a *difficulty target*. There is no way to reverse engineer a sufficient hash and the algorithm repeatedly hashes the block header, changing the nonce, until a hash is found. Consensus is reached by regarding the longest chain of work to be the valid chain. The blocks are in a sense chained through these hashed headers and each additional block adds cumulative work on the blocks below reducing the chance of ever being reverted. The difficulty target adjusts every 2015th[4] block[10] as part of the consensus algorithm and forces the average time between blocks to always be ten minutes. Running the bitcoin *Proof-of-work* algorithm is known as *mining* as per the similarities with the finding and minting of gold coins.

3.2.1 Merkle Tree

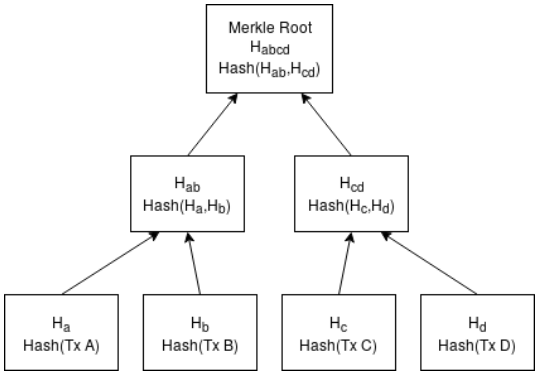

Figure 3: *A merkle tree resulting from four transactions A, B, C, D.*

The content in the block header is practically immutable since a change would make the hash invalid. The transactions are not in the header but are still immutable by the use of a Merkle tree(see fig 2). A Merkle tree is a binary tree in which the parent is the hash of its two children. The root hash of the tree is included in the header and if any transaction were to change it would also cause the root hash to change rendering each transaction immutable by implication. Bitcoin uses the double `SHA256` as the Merkle tree hash function.

3.2.2 Incentive

The miners are incentivized by receiving the transaction fees from the transactions included in mined block as well as a subsidy. The subsidy is on the form of a coinbase transaction which has only outputs, this is how new bitcoin is minted. Every fourth year the subsidy halves and by 2140 will be zero. The miner must follow the consensus rules and mined blocks not following the rules will not be propagated through the network causing the miner to loose out on the block reward.

A miner, or a miner cartel, controlling more than 50% of the hashing power could technically double-spend transactions by mining blocks on a secret chain **A**, instead of the public chain **B**, without broadcasting them to the greater network and simultaneously spending outputs **OB** on chain **B**. When the chain **A** is broadcast and if it contains more work than **B** the outputs **OB** would be invalidated and never have transpired per the consensus rules. A huge problem with such an attack is that the confidence in the network would decrease causing the value of the 'stolen' bitcoin outputs **OA** to be close the worthless. The attacker would still own a lot of specialized hardware, only capable of running the `SHA256` hashing algorithm, that is worthless without utility outside the network. In a sense such an attack would cause mutual destruction for both the network and the attacker.

3.3 Script

Transactions utilize a forth-like polish-notation stack based execution scripting language[18]. Each transaction output consist of a locking script[5]. In order to spend an output a valid unlocking script must be provided in the input to the spending transaction that solves the locking script[6]. Validation is performed by executing the unlocking script and locking script sequentially as seen in figure 4.

```
 OP_1    OP_2 OP_ADD OP_3 OP_EQUAL
unlock             lock
```

Figure 4: *A simple predicate. Evaluated from left to right (1 2 +) -> 3 and (3 3 =) -> true. This lock script can obviously be solved by anyone and should not be used with real bitcoin.*

The two scripts added together forms a predicate. If the predicate is true the transaction is valid. Nakamoto considered naming the language

[4] This is off by one bug, 2016 blocks is two weeks

[5] Also refereed to as scriptPubKey. Locking script is a broader term and technically scriptPubKey scripts $\subseteq$ locking scripts.

[6] scriptSig, witness $\subseteq$ unlocking scripts.

predicate but went with script to be more inclusive to a broader audience[49].

3.3.1 Pay to Public Key Hash

The Pay-to-Public-key-hash or P2PKH for short was the standard script for a long time. It allows only the owner of the private key to the public key to spend the output.

```
<Sig><Pub_Key>
    unlock

OP_DUP OP_HASH160 <Pub_Key_Hash>
OP_EQUALVERIFY OP_CHECKSIG
    lock
```

Figure 5: *The P2PKH locking and unlocking scripts.*

The P2PKH pushes the signature produced by the private key and the public key to the stack. The public key gets duplicated and the top one gets hashed. The `OP_HASH160` is a double hash and first hashes the key with `SHA256` and then `RIPEMD160`. This is done to hide the public key until expenditure which may be useful if, for example, the ECDSA would break and private keys could be reversed. In the early days of the network this obfuscation technique wasn't used, for example the first transaction between Nakamoto and Finney did not hide the public key[2]. The `OP_EQUALVERIFY` pops the two public key hashes and terminates the script with false if they are not equal. The `OP_CHECKSIG` verifies that the signature is indeed a match with the public key and returns true.

3.3.2 Pay to Script Hash

The Pay-to-Script-Hash or P2SH is a more flexible transaction than the P2PKH and was introduced with BIP016[16]. It allows the locking script to only be the hash of the redeemable script. If a long script with many public keys is considered as seen in figure 6.

```
0 <Sig A> <Sig B>
    unlock

2 <Pk_A><Pk_B><Pk_C> 3 OP_CHECKMULTISIG
    lock
```

Figure 6: *Showing a multisig transaction which require to two signatures out of three to be spent. Note the `0` in the unlock script, it's there due to a bug with `OP_CHECKMULTISIG` which pops an extra item from the stack. If not for the `0` it would try to pop an empty stack. The dummy value must be a `0` with BIP 147 compliant node implementations[38].*

This multisig transaction(in figure 6) could be rewritten as a P2SH by hashing the locking script with `SHA256` and `RIPEMD160` and utilizing it as seen in figure 7.

```
0 <Sig A> <Sig B> <redeem script>
    unlock

OP_HASH160 <redeem script hash>
OP_CHECKVERIFY
    lock
```

Figure 7: *The P2SH transaction which is essentially equivalent to the multisig in figure 6. Note that 'redeem script' corresponds to the lock script in figure6.*

Converting to a P2SH transaction does two things:

- It shifts the fee burden from the sender to the recipient. The locking script is shorter, the unlocking script is longer.
- All unspent UTXOs must be kept in the RAM of the node, shifting the burden does reduces the node RAM load.

P2SH also allows the script to be used as an address as per BIP013[15]. By simply sending to the script address could fund any type of complex transaction with no added complexity to the sender.

Even though script is not Turing complete it allows many quite advanced systems on top of it, transactions or output scripts is sometimes referred to as 'smart contracts'[7].

[7] First formally defined by Nick Szabo in 1994[57]

```
IF
  IF
    2
  ELSE
    <30 days> CHECKSEQUENCEVERIFY DROP
    <Pubkey D> CHECKSIGVERIFY
    1
  ENDIF
  <Pk A><Pk B><Pk C> 3 CHECKMULTISIG
ELSE
  <90 days> CHECKSEQUENCEVERIFY DROP
  <Pubkey D> CHECKSIG
ENDIF
```

Figure 8: *A more complex multisignature locking script involving timelocks.*

One example of a more advanced script is this multisignature locking script with timelocks seen in figure 8. The script consists of three clauses, the clause executed is determined by the unlocking script since the condition to `IF` is the top item on the stack, not the following item as in most languages. The output could be spent by either clause being triggered:

- 2 out 3 of the A, B, C signatures.
- 30 days passed, D signature and 1 out of A, B, C signatures.
- 90 days passed and D signature.

3.3.3 Relative lock time

The script(figure 8) uses a construction known as a relative time lock and uses the `OP_CHECKSEQUENCEVERIFY` opcode in the bitcoin scripting language[27]. The relative lock time will only become true after the defined time since the parent transaction were included in a block has passed. To avoid the block miners to become oracles of time, the median time of the eleven last blocks are used as the lock time(BIP113[36]). The `OP_CHECKSEQUENCEVERIFY` was activated by BIP112[24]. The block time is therefor approximately an hour behind real time.

The Lightning Network utilize many complex transactions and more will be seen in chapter 5.

3.4 Network

The Bitcoin network consists of nodes complying with the consensus protocol. In the beginning only the reference implementation available but with the growth of popularity others have implemented independent implementations[10][1][4].

The mining component of nodes has mostly been decoupled from ordinary nodes with the advent of ASIC[8]. Mining nodes still do everything as an ordinary node does as it needs the transaction history to construct valid block. Non mining nodes validates and propagates blocks and transactions across the network. It allows some benefits in constructing transactions but mainly allows the operator to validate that the network operates as expected instead of trusting that it does.

Construction of transactions can be possible without running a node and many bitcoin wallets does not operate as a node.

[8] Application Specific Integrated Circuit, hardware that are built only to run `SHA256`

Chapter 4 | Scaling bitcoin

There is no secret that the bitcoin blockchain will not be able to scale on its own. When Nakamoto first published the whitepaper on the cryptographic mailing list his first response was in fact that such a solution won't scale very well[26].

Andreas Antonopoulos has drawn similarities between the scaling of bitcoin with the scaling of the internet:

> "Bitcoin is failing to scale. If we're really lucky, bitcoin will continue to fail to scale gracefully for 25 years, just like the internet."[19]

His quote in many ways are spot on as Usenet used a *Store-and-forward* method for content to reach every server in the network. *Store-and-forward* does not scale very well and is somewhat similar to the base Bitcoin protocol. The internet moved away from *Store-and-forward* to more direct routing and Bitcoin is too, allowing transactions to happen off-chain.

In this chapter the various scaling alternatives will be discussed.

4.1 Block Size limit

In 2010 Nakamoto reduced the block size limit from 32 to 1 Megabyte as part of two commits. The first commit introduces the new `BLOCK_SIZE_LIMIT` constant[44] and the other commit enforcing the blocks to be under the limit[45]. Nakamoto did not give any explanation as to the reason of this limit. As there was only a few transactions per block at this time a limit reduced the worst case scenario for a DoS attacker filling up the blocks with dummy transactions.

Increasing the amount of transactions would also increase the imposing cost for the ecosystem by forms of bandwidth, processing, storage and the speed by which blocks propagate through the network. By imposing a block limit the cost of operating a node would be kept low allowing more peers to be able to participate in the network.

4.2 Increasing block size limit

There has been many proposals to increase the block size. Including a handful of BIPs:

- **BIP 100** allows the miner to vote on the block limit by encoding a proposed value in the coinbase unlocking script. A 75% supermajority may increase the block size limit every 2016 blocks. Each period change is limited to only increase by 5% from the previous period.[30]
- **BIP 101** proposes an immediate increase to 8 megabyte blocks and continue to double every second year. In the two year periods the size would increase at a linear pace based on the block timestamp.[17]
- **BIP 102** is a simple one time increase to 2 megabytes. The BIP would be triggered if 95% of the latest blocks signaled support for the upgrade.[29]
- **BIP 103** aims to increase the block size in accordance with the technology. In practice however it increases the block size with 4.4% every 97 day, 17.7% annually. [61]

Most of these BIPs are implemented as a hard forks. Older nodes would regard the bigger blocks as being over the limit and thus invalid. If poorly executed, without a great majority upgrading the software in time, the network would be forked into two networks with diverging transaction histories.

4.2.1 Contentious Hard Forks

No proposal to increase the block size has yet received enough support to be activated. This has caused some controversy among community members and multiple have attempted to increase the limit by force.

One of the earliest attempts to increase the block size was when node Bitcoin XT implemented BIP 101 mentioned above. It failed to reach the miner consensus it required to activate. The BIP 101 was then removed and replaced by a 2 megabyte block limit which forked bitcoin into a new network known as bitcoin classic. Mike Hearn, co-creator of Bitcoin XT along with Gavin Andersen, has made his intention clear arguing that the block size must

increase for bitcoin to reach any adoption beyond a fringe niche[34].

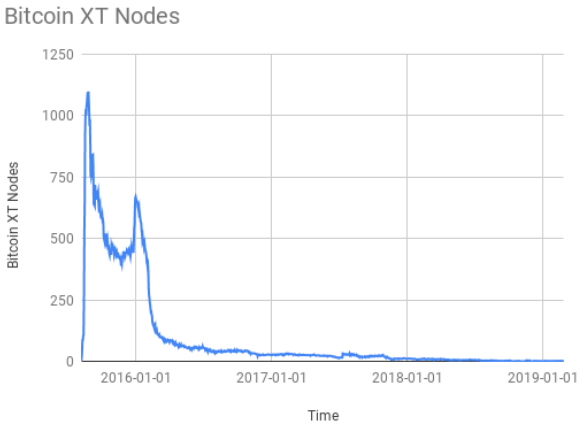

Figure 1: *Historic Bitcoin XT node count. As of 2019-02-25 only 3 nodes are still active. Data retrieved from coin.dance[13].*

As figure 1 attests, bitcoin XT and classic failed to receive support and is now dead.

4.2.2 The bitcoin cash saga

- SegWit2x

(THIS SECTION IS OBVOUSLY NOT DONE)

4.3 Removing decentralization

4.4 Side channel

4.5 Off-chain scaling

4.6 Rational for a small blocks

Chapter 5 | The Lightning Network

The Lightning Network is a network of micro-payment channels on top of the Bitcoin Network. The network was first conceived and defined in 2014[52], became viable with the segregated witness upgrade[42] and got activated through a *user-activated-soft-fork* in mid 2017[28]. The upgrade moves(segregates) the unlocking script(witness) from the transaction which fixes transaction malleability since multiple unlocking scripts may be valid for a single locking script.

The network protocol have been defined[9] with four independent actors implementing protocol compliant nodes[6][7][5][8].

5.1 Payment channels

The Lightning network consists of many payment channels between both routing and non-routing nodes. A payment channels is a essentially a transaction containing funds locked up on the bitcoin network. It requires signatures from both parties to be able to spend the transaction and cease the channel.

Initially each party have a signed transaction splitting the funds to their original committed amount[1]. The parties can then agree on a different balance signing a new transaction invalidating the previous spending transaction essentially allowing for infinite transactions inside the channel without any burden on the bitcoin network.

5.1.1 Funding Transaction

A payment channel is funded by a funding transaction defined in the lightning paper by following steps[52]:

1. Create the parent (Funding Transaction)
2. Create the children (Commitment Transactions and all spends from the commitment transactions)
3. Sign the children
4. Exchange the signatures for the children
5. Sign the parent
6. Exchange the signatures for the parent
7. Broadcast the parent on the blockchain

It is critical that the signatures of the spending transaction is exchanged before the funding transaction otherwise the funds could be held hostage by an uncooperative partner[2]. By exchanging the spending transaction first either party could broadcast it to retrieve the initial funds.

5.1.2 Commitment Transaction

In order for a channel to be useful the creation of a new commitment transaction[3], updating the balance, would need to invalidate the previous one. The previous transaction is after all still a valid bitcoin transaction. This creates a problem since each party would benefit from different commitment transactions being broadcast to the network.

The solution is to construct the commitment transaction similar to that of a fidelity bond where one party would be penalized if violating the agreement. In this case violation is broadcasting any but the latest commitment transactions. To be able to ascribe blame to the violating party the commitment transactions for the same channel state must be unique as otherwise one would not know which party broadcast the transaction to the network. Only uniqueness is however not enough since there is no way to enforce the penalty for violation.

The enforcement is possible if revocation is built into the commitment transaction, where the other party could revoke a faulty broadcast. This became possible with the addition of relative block maturity introduced with BIP68(see section 3.3.3). The commitment transaction has two output paths, one is the other party's funds spendable immediately. The other output has two redeemable paths. Either:

1. The broadcasting party can spend the funds after a defined relative block time has passed.

[1] In the RFC each channel is funded by only one party so the initial split is always all to one party.

[2] This is possible with the `SIGHASH_NOINPUT` transaction defined in BIP118[25] which decouples the signature from the specific transaction hash of the output. This is not yet activated and only single source funding is currently available

[3] It may be easier to think of this type of transaction outputs as smart contracts, as in this context it's the same thing.

Essentially leaving time window for dispute for the violated party to enforce the penalty(Revocable delivery).

2. The non broadcasting party can refute that the commitment transaction is not last signed spending spending the output as penalty.

Remember that the commitment transactions is reversed for the other party. This type of output which enable revocation is called a Revocable Sequence Maturity Contract(RSMC).

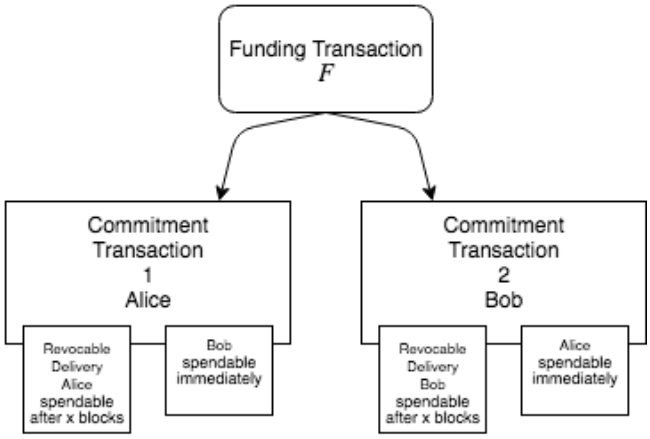

Figure 1: *Commitment transactions.*

5.1.3 Revocation of old commitment transaction

In order to update the channel balance a new pair of commitment transactions is created. The previous commitment transactions is invalidated by both signing a Breach Remedy Transaction(BRT) spending from their own previous commitment transactions RSMC output and handing it over to the other party. If a previous commitment transaction were to be broadcast the BRT would take precedence since the revocable delivery takes a certain amount of time before it is valid. The Breach Remedy Transaction will send all funds to the non faulty party, this is the penalty for violating the contract.

It is necessary for each party to monitor the blockchain for old commitment transactions and to broadcast the Breach Remedy Transaction in case it happens. If the Breach Remedy Transaction is not broadcast in the designated time the revocable delivery output of the previous commitment transaction will become valid and may be spent, settling the incorrect channel balance is settled and valid on the blockchain.

5.1.4 Alice and Bob opens a channel

Alice and Bob wants to open a channel. They agree on funding the channel with **0.5฿** each. They construct a funding transaction F which require both Alice and Bobs signatures to be spent. Alice constructs a CT_{A1} with two outputs $RSMC_{A1}$ and SO_{B1} and similarly Bob constructs CT_{B1} with outputs $RSMC_{B1}$ and SO_{A1}. Alice signs CT_{B1} and Bob signs CT_{A1}[4]. Alice and Bob exchange signatures for F and either of them broadcast F.

Alice wants to buy a cookie from Bob for **0.2฿** and Bob agrees to sell. They each constructs new commitment transactions CT_{A2}, CT_{B2}, each with new $RSMC_{A2}$, $RSMC_{B2}$ and SO_{B2}, SO_{A2} that reflects the new balance **0.7฿** to Bob and **0.3฿** to Alice without signing them. Before Bob can hand Alice the cookie the old CT_{A1}, CT_{B1} is invalidated by Alice signing a BRT_{B1} spending from $RSMC_{A1}$. Although Bob has nothing to gain by broadcasting CT_{B1} as he would net **0.2฿** less - Bob signs BRT_{A1} spending from $RSMC_{B1}$. Alice signs CT_{B2} and Bob signs CT_{A2}. The channel balance is updated.

If Alice tries to undo the cookie purchase by broadcasting CT_{A1} giving **0.5฿** each. Bob simply broadcasts the BRT_{B1} spending from Alice's share of the CT_{A1} and Bob receives **1฿** and Alice **0฿**.

As long as the channel is open Alice and Bob may settle any imbalances between them with a new commitment transaction.

5.1.5 Contracts and implementation

Signing and sending BRTs is unnecessarily inefficient if a HD wallet is used to generate keys[60]. HD wallets have a master key which can derive new keys in a tree structure. Each key in the tree can derive its children and no child can derive its parent. This works both for the public and the private keys independent of each other. A new key is used for each CT and if that is deep in the tree, say a millionth layer down, and can simply be disclosed to the other party whenever the CT is invalidated. The public master key may be shared with

[4] A note on notation - The lowered A,B indicates whether the output is spendable by Alice or Bob. The number denote the channel state the output corresponds to. SO stands for Spendable Output.

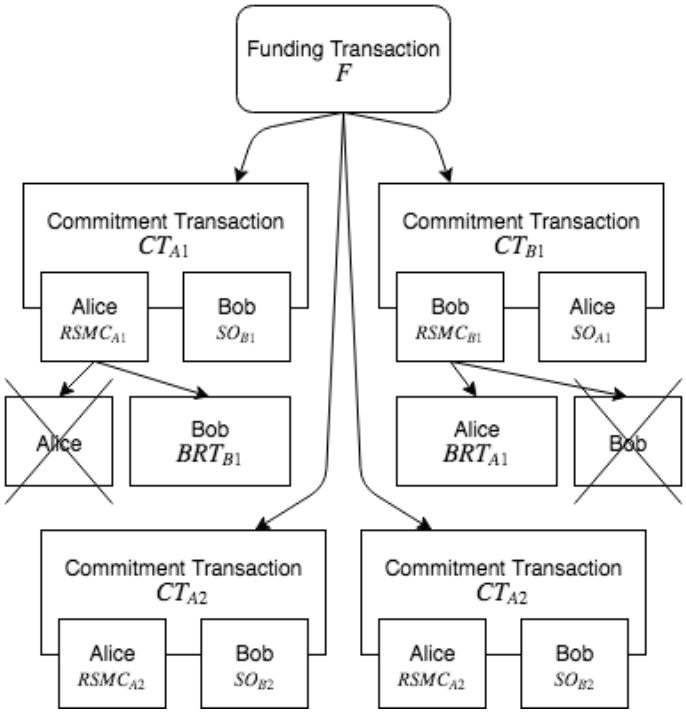

Figure 2: *Showing the channel balance state between Alice and Bob after the second Commitment transaction has been made.*

the other party so all public keys can be derived for transaction construction. With each new invalidated CT the previous CT's parent key may be used. So each party only needs to store the latest disclosed key as the other child keys can be derived from it.

```
OP_IF
  <time> OP_CHECKSEQUENCEVERIFY OP_DROP
  OP_HASH160 <A pk> OP_EQUALVERIFY OP_CHECKSIG
OP_ELSE
  2 <A r pk><B r pk> 2 OP_CHECKMULTISIGVERIFY
OP_ENDIF
```

Figure 3: *A Revocable Sequence Maturity Contract.*

If we consider a Revocable Sequence Maturity Contract seen in figure 3, it is constructed by Alice and she used Bob's master public key to derive Bob's revocation public key in some prearranged manner. To invalidate the CT Alice need only to disclose the private key corresponding to her revocation public key to Bob for Bob to be able to revoke a violation.

5.1.6 Closing channels

It's possible to close a channel by broadcasting either of the latest CTs. However the broadcasting needs to wait the designated time period, losing out on the time value of money. Instead both parties may agree to construct a new Exercise Settlement Transaction ES with direct spendable outputs instead of a $RSMC$ output. Signing the ES would close the channel and allowing both parities immediate access to the funds.

5.2 Network wide payments

By combining channels into a network, a payment could potentially ripple through multiple channels and may allow parties without a direct channel to transact. Such a multi-hop payment would need the same trust-less custodial properties for the routing middle nodes as between direct channel parties shown above. This is solved by utilizing a Hashed Timelock Contract(HTLC) for each of the channels in the route.

5.2.1 Hashed Timelock Contract

A $HTLC$ is an extra output added to a CT which enforces a payment from Alice to party Bob if and only if Bob can produce and disclose a preimage R of a hash in a designated time period.

```
OP_IF
  OP_HASH160 <H_R> OP_EQUALVERIFY
  2 <A PK_2><B PK_2> OP_CHECKMULTISIGVERIFY
OP_ELSE
  <time> OP_CHECKSEQUENCEVERIFY OP_DROP
  2 <A PK_1><B PK_1> 2 OP_CHECKMULTISIGVERIFY
OP_ENDIF
```

Figure 4: *A Hashed Timelock Contract. It's not yet viable for payments in this state.*

A $HTLC$ can be seen in figure 4 and contains two validation paths.

1. If Bob have the knowledge of R that produces H_R he can validate the first path. (Execution)
2. If Bob fails to prove knowledge of preimage R, Alice may validate the second path after the designated time period has passed. (Timeout)

5.2.2 Revocation of HTLCs

$HTLC$s have a similar problem as with direct payment channels, if an old CT is broadcast both validation paths may be valid. In the same manner as before, the $HTLC$ is made revocable by the construction of two different contracts for each party $HTLC_{A1}$ and $HTLC_{B1}$. So here again the possibility to ascribe blame exists. Adding revocable paths to $HTLC_{A1}$ and $HTCL_{B1}$ is necessary[5] but not enough since the execution path is also valid. A period for dispute must still be had as the disclosure may be correct but may be from an old commitment transaction. The same is true for the timeout path as Bob may not have knowledge of R in an old commitment transaction.

Instead the timeout / execution path to oneself is spent into new transactions with a dispute period and ability for the other party to revoke an incorrect broadcast.

```
OP_DUP OP_HASH160 <H_RPK> OP_EQUAL
OP_IF
  OP_CHECKSIG
OP_ELSE
  <PK_B> OP_SWAP OP_SIZE 32 OP_EQUAL
  OP_NOTIF
    OP_DROP 2 OP_SWAP <PK_A> 2 OP_CHECKMULTISIG
  OP_ELSE
    OP_HASH160 <R_preimage> OP_EQUALVERIFY
  OP_ENDIF
OP_ENDIF
```

Figure 5: *The $HTLC_{A1}$ as offered by Alice*

The $HTLC_{A1}$ seen in figure 5 can only be broadcast by Alice and contains three validation paths:

1. Bob can revoke if it's not the last CT constructing a BRT spending the output.
2. The output is spent by a pre-signed transaction to a new transaction ST with a dispute period.
3. Bob can disclose knowledge of R and spend the output.

The order of the two last clauses is reversed for Bob's $HTLC_{B1}$ as if Bob can prove knowledge of R the output is spent to a new transaction ST for disputation.

[5]It's necessary to include this path as an uncooperative party may never broadcast the execution/timeout transaction in which revocation is possible.

```
OP_DUP OP_HASH160 <H_RPK> OP_EQUAL
OP_IF
  OP_CHECKSIG
OP_ELSE
  <PK_B> OP_SWAP OP_SIZE 32 OP_EQUAL
  OP_IF
    OP_HASH160 <R_preimage> OP_EQUALVERIFY
    OP_DROP 2 OP_SWAP <PK_A> 2 OP_CHECKMULTISIG
  OP_ELSE
    OP_DROP <time> OP_CHECKLOCKTIMEVERIFY OP_DROP
  OP_ENDIF
OP_ENDIF
```

Figure 6: *The $HTLC_{B1}$ as offered by Bob*

The $HTLC_{B1}$ seen in figure 6 thus contain:

1. Alice can revoke if it's not the last CT constructing a BRT spending the output.
2. Bob can prove knowledge of R the output is spent to a new transaction ST with a dispute period.
3. After the designated time Alice can spend the output.

The new transactions by which both the second clauses spends to is constructed and pre-signed before the $HTLC$s is exchanged simply adds time for revocation as seen for $HTLC_{B1}$ in figure 7.

```
OP_IF
  <PK_RA>
OP_ELSE
  <time> OP_CHECKSEQUENCEVERIFY OP_DROP
  <PK_B>
OP_ENDIF
OP_CHECKSIG
```

Figure 7: *Showing the transaction $HTLC_B$ spends to, the keys are reversed for $HTLC_A$.*

A $HTLC$ is terminated off-chain by creating a new commitment transaction, without $HTLC$ outputs, reflecting new the balance state as both parties knows what would transpire if the transaction were broadcast. In the creation of the new CT both parties disclose the both the private keys corresponding to the two revocation outputs as a penalty if and old CT were broadcast. Hence most transactions will be kept off chain and only held in case of an uncooperative party.

In this manner Hashed Timelock Contracts may be used without the need for either of the channel parties to trust one another.

Note that key generation and disclosure may be used for $HTLC$ in the same manner previously disused in section 5.1.5.

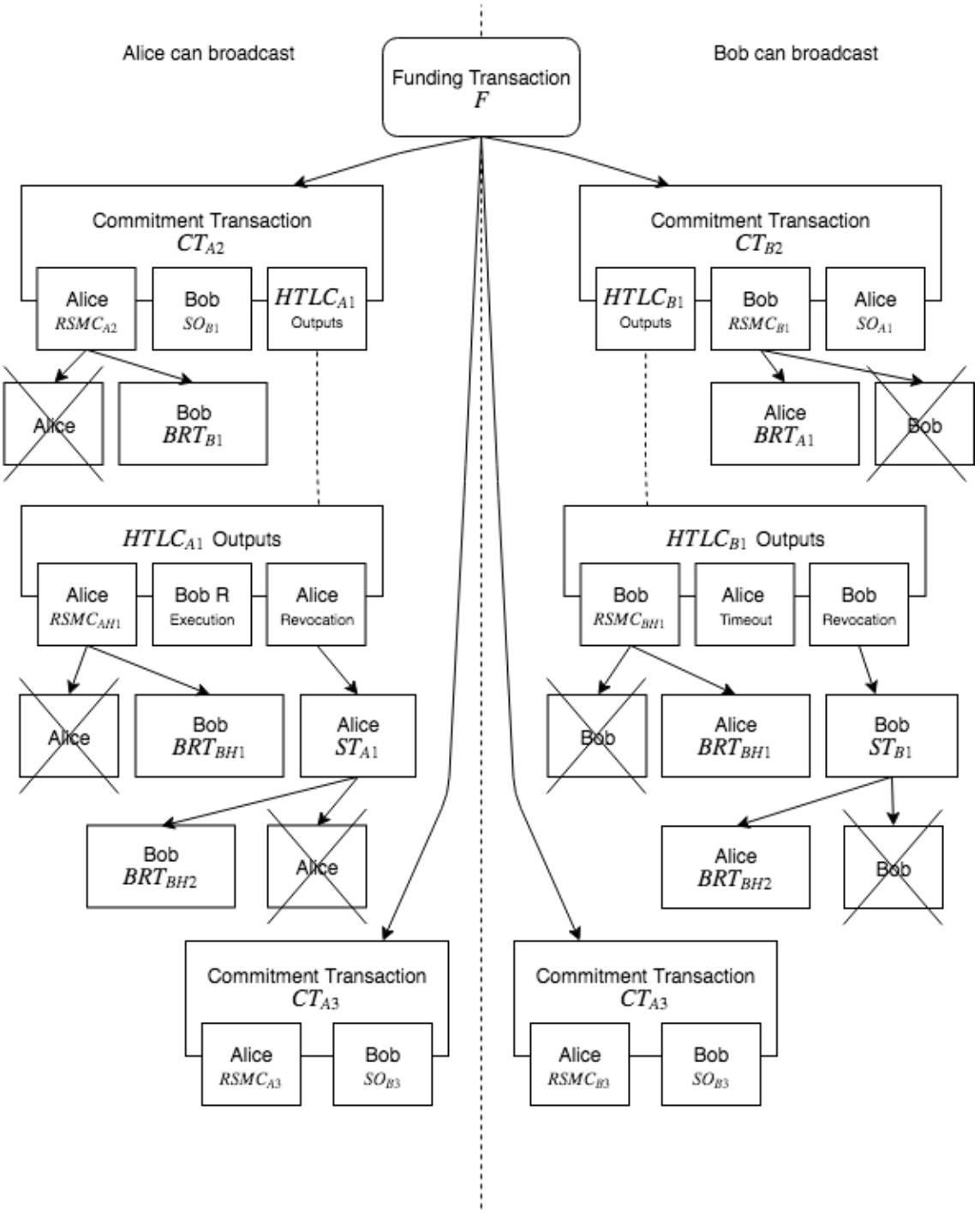

Figure 8: *Shows the state of transactions after Alice and Bob have agreed to a new Commitment Transaction CT_{A3}, CT_{B3}. The transactions to the left side can be broadcast by Alice and the ones to the right by Bob. With the signing of the new CTs Bob and Alice disclose keys for BRT_{AH1}, BRT_{AH2}, BRT_{BH1}, BRT_{AH2} as penalty if an old commitment transaction is broadcast.*

5.3 Multi-hop payments

With properly funded channels and Hashed Timelock Contracts in place network wide payments utilizing multiple channels is possible. The network uses a Gossip protocol[6] were public channels are broadcast. Each node may then find a path through multiple channels and construct a chain of $HTLC$s with decreasing time locks.

Alice wants to pay David but have no direct path to Carol. She finds a route via Bob and Carol to David as seen in figure 9.

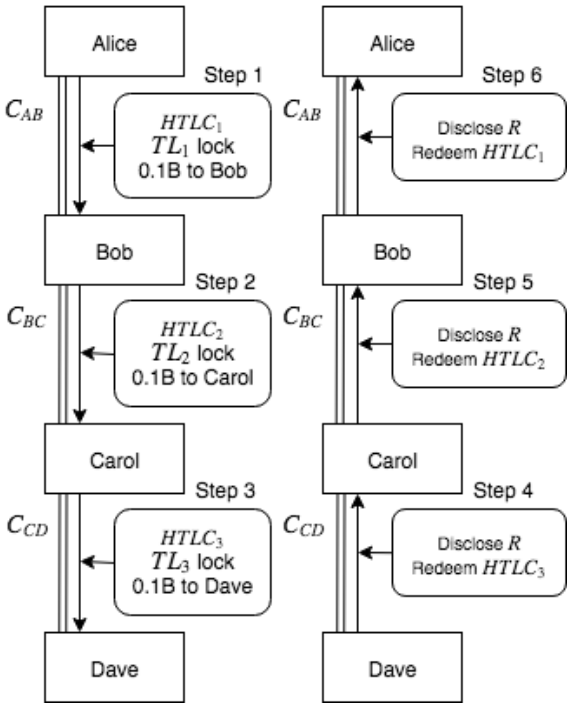

Figure 9: *Shows a multihop payment from Alice to Dave across three channels C_{AB}, C_{BC}, C_{CD}. It's critical that the HTLCs is constructed in the correct order and with decreasing timesteps $TL_1 > TL_2 > TL_3$.*

Alice and David agree on size of the payment and David constructs R and shares H_R with Alice. Alice then uses H_R to construct a $HTLC_1$. After Alice have created $HTLC_1$ with TL_1, Bob is comfortable creating $HTLC_2$ with a $TL_2 < TL_1$ as the only way Carol could redeem $HTLC_2$ is by disclosing R to Bob. Bob in turn may then disclose R to Alice and redeem $HTLC_1$. If the timelocks was improperly set as $TL_2 > TL_1$ the $HTLC_1$ may be invalid by the time Carol redeem $HTLC_2$ leaving Bob paying Carol but without getting payed by Alice. [RFC source of actual time step here?]

[6] as per BOLT#7 routing-gossip in Lightning RFC[9]

5.4 Routing fee

The routing nodes receives a fee as compensation for displaced channel liquidity, the time value of money and operational cost of bandwidth, computation and electricity. The fee is structured with a base fee F_B and a fee F_R proportional to satoshis forwarded AF. The base fee is always paid independent of payment size while the proportional part is the product of size and fee. The base fee is denominated in microsatoshi (**µS**) and the proportional in microsatoshi per satoshi forwarded (**µS** / **S**). The fee for a payment P is thus:

$$F_P = F_B + (AF * F_R/1000000)$$

This fee structured is problematic as it doesn't discriminate for channel balance. As this fee structure is very integral to this thesis a proposal to change the protocol regarding this fee structure is made. The proposal is explained in the following chapter and the full proposal is appended in Appendix A.

Chapter 6 | Evaluation

(DONT MIND THIS CHAPTER AS MOSTLY SOFTWARE FOR THIS PART IS CURRENTLY WRITTEN AND CONTAINS ONLY THE BARE ESSENTAIL THOUGHS)

With the eventual growth of the Lightning Network a fee market for channel liquidity will emerge. How routing nodes

6.1 Lightning network as a graph

6.2 Fee as a competitive equilibrium

The

The conventional long run demand and supply curve apply to the fee market.

If a demand is running DD' and Supply SS' and

Suppose the average fee price is set at F_1 with demand F_1D_1 and supply F_1S_1.

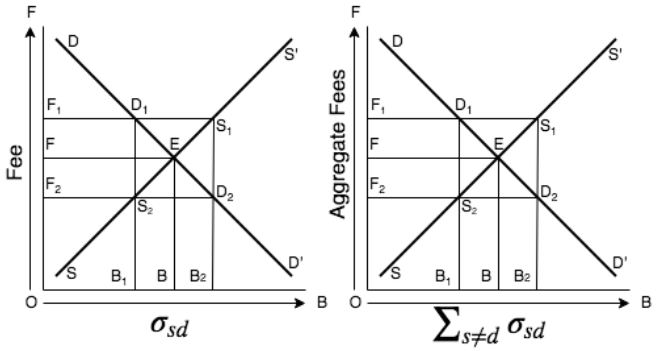

Figure 1

The cost of procuring a shortest path between s and d might be unequal for two different routing nodes dependent on already existing paths and set policies.

6.3 Pricing function

Betweenness centrality

$$g(v) = \sum_{s \neq v \neq d} \frac{\sigma_{sd}(v)}{\sigma_{sd}}$$

Steps to find the maximizing the fee profit for outgoing channel C assuming pairs have a uniform probability of conducting a payment.

1. Calculate the shortest path from all-to-all vertices without going through C with either Floyd-Warshall or Johnson.
2. Calculate shortest path from C source to all explicitly going through C with Dijkstra.
3. Compare difference between all-to-all cost with all-to-V-to-all, producing a cumulative summation over the difference.
4. Maximizing fee * cumulative difference.

Where the use of Johnson over Floyd-Warshall depends on the sparseness of G as

$$O(V^3)$$

$$O(V * E * log(V))$$

6.4 Topology

Network topology

$$P(k) \propto k^{-\gamma}$$

$$p_i = \frac{k_i}{\sum_j k_j}$$

Lightning network consists of nodes playing game

Non routing nodes, cheap as possible and possible as private as possible(?)

Barabási–Albert model

Does it become scale free

does any channel opening algorithm perform better than another and essentially force the network into a scale free hub-spoke network instead of a mesh network?

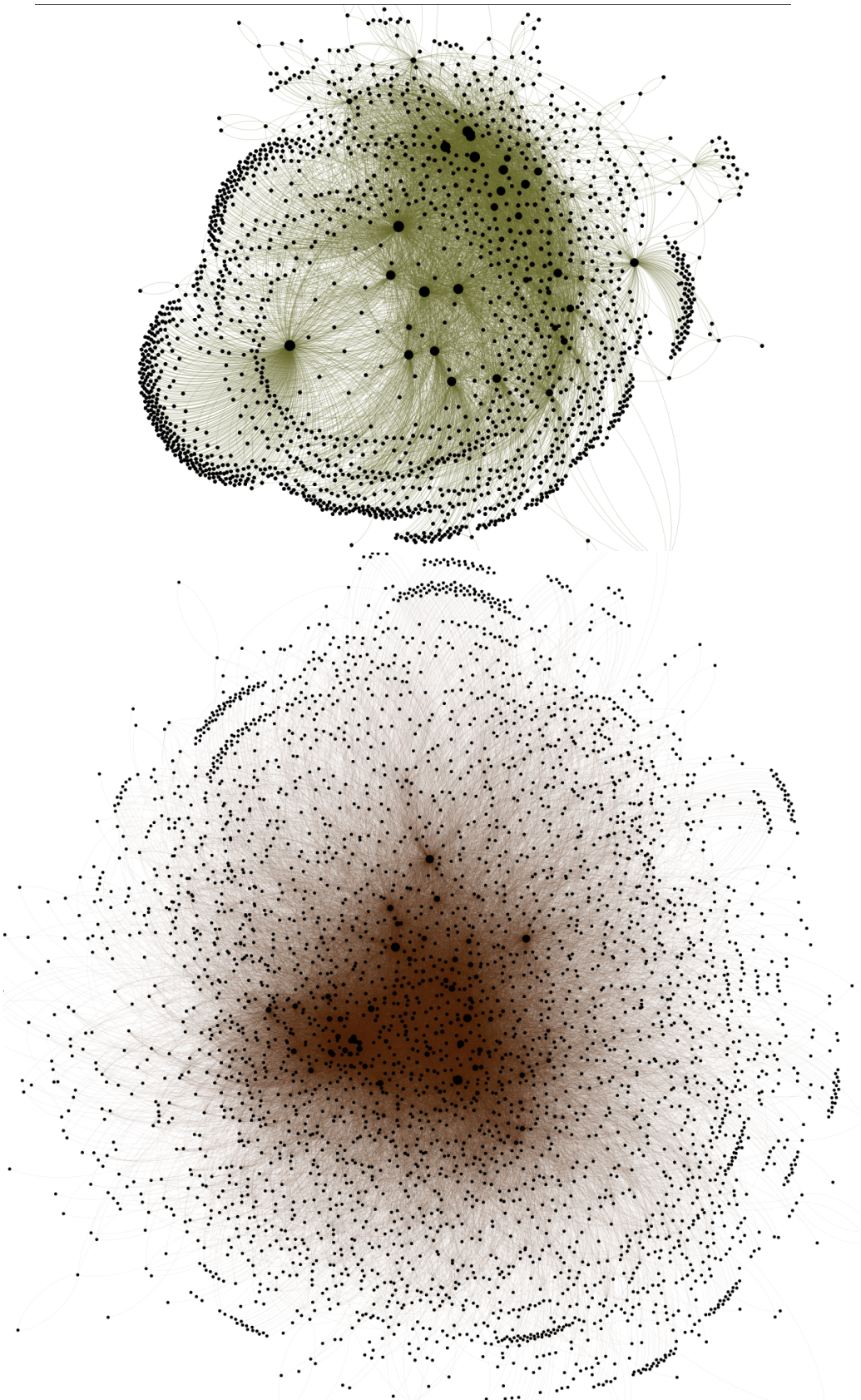

Figure 2: *Shows testnet above and mainnet below as of 2019-02-25. Node size corresponds to degree, larger degrees to larger size. Positioning is a mix between Force Atlas and Fruchterman Raingold. Network data retrieved by running a c-lightning node and visualized with Gephi[3].*

Chapter 7 | Results

Chapter 8 | Discussion & Conclusions

8.1 Complexity

When the network grow it will eventually become to cumbersome for every node to Floyd-Warshall the whole network as this simulation assumes. Approx. will propably be used. Ant routing tables have been suggested but might be problem with lying about connectedness.

References

[1] btcd: a full node bitcoin implementation. https://github.com/btcsuite/btcd.

[2] The first transaction, satoshi nakamoto -> hal finney in block 170. https://www.blockchain.com/btc/tx/f4184fc596403b9d638783cf57adfe4c75c605f6356fbc91338530e9831e9e16. (Accessed on 2019-02-06).

[3] Gephi. https://github.com/gephi/gephi.

[4] Light weight bitcoin node implementation. https://github.com/lightninglabs/neutrino.

[5] Lightning node c-lightning developed by blockstream. https://github.com/ElementsProject/lightning.

[6] Lightning node daemon. https://github.com/lightningnetwork/lnd.

[7] Lightning node eclair developed by acinq. https://github.com/ACINQ/eclair.

[8] Lightning node lit developed under mit digital currency initative. https://github.com/mit-dci/lit.

[9] Lightning specification, request for comments. https://github.com/lightningnetwork/lightning-rfc.

[10] Bitcoin reference implementation, 2009.

[11] The coinbase transaction of the genesis block. https://www.blockchain.com/btc/tx/4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b?show_adv=true, 2009. (Accessed on 2019-01-30).

[12] Original bitcoin repository with the genisis block function, 2009.

[13] coin.dance. https://coin.dance/, 2019.

[14] The world gold council. https://gold.org, 2019.

[15] Gavin Andersen. Bip 0013 address format for pay-to-script-hash. https://github.com/bitcoin/bips/blob/master/bip-0013.mediawiki, January 2012. (Accessed on 2019-02-07).

[16] Gavin Andersen. Bip 0016 pay to script hash. https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki, January 2012. (Accessed on 2019-02-07).

[17] Gavin Andersen. Bip 0101 increase maximum block size. https://github.com/bitcoin/bips/blob/master/bip-0101.mediawiki, June 2015. (Accessed on 2019-02-19).

[18] Andreas Antonopoulos. *Mastering Bitcoin*. O'Reilly Media, first edition, July 2014.

[19] Andreas Antonopoulos. *The Internet of Money*. Merkle Bloom LLC, first edition, August 2016.

[20] Adam Back. Hashcash - a denial of service counter-measure. 2002.

[21] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286, October 1999.

[22] Michael D. Bordo. The imbalances of the bretton woods system 1965 to 1973: U.s. inflation, the elephants in the room.

[23] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. November 2007.

[24] BtcDrak, Mark Freienbach, and Eric Lombrozo. Bip 0112 checksequenceverify.

[25] Christian Decker. Bip 0013 sighash noinput. `https://github.com/bitcoin/bips/blob/master/bip-0118.mediawiki`, February 2017.
(Accessed on 2019-02-13).

[26] James A. Donald. Re: Bitcoin p2p e-cash paper. `https://www.mail-archive.com/cryptography@metzdowd.com/msg09963.html`, 2008.

[27] Mark Freienbach, BtcDrak, Nicolas Dorier, and kinoshitajona. Bip 0068 relative lock-time using consensus-enforced sequence numbers. `https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki`, May 2015.
(Accessed on 2019-02-13).

[28] Shaolin Fry. Bip 0148 mandatory activation of segwit deployment. `https://github.com/bitcoin/bips/blob/master/bip-0148.mediawiki`, 2017. (Accessed on 2019-02-13).

[29] Jeff Garzik. Bip 0102 block size increase to 2mb.

[30] Jeff Garzik, Tom Harding, and Dagur Valberg Johannsson. Bip 0100 dynamic maximum block size by miner vote.

[31] Cyril Grunspan and Ricardo Perez-Marco. Ant routing algorithm for the lightning network.

[32] Steve H. Hanke. Venezuela hyperinflation.

[33] Steve H. Hanke and Nicholas Krus. *World Hyperinflations*. August 2012.

[34] Mike Hearn. Why is bitcoin forking. `https://medium.com/faith-and-future/why-is-bitcoin-forking-d647312d22c1`, august 2015.

[35] Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the Association for Computing Machinery*, January 1977.

[36] Thomas Kerin and Mark Freienbach. Bip 0113 median time-past as endpoint for lock-time calculations.

[37] John Maynard Keynes. *The General Theory of Employment, Interest, and Money*. 1936.

[38] Johnson Lau. Bip 0147 dealing with dummy stack element malleability. `https://github.com/bitcoin/bips/blob/master/bip-0147.mediawiki`.
(Accessed on 2019-02-07).

[39] Nils Linder. *Nordisk Familjebok*. C. E. Gernandt, second edition, 1911.

[40] Nils Linder. *Nordisk Familjebok*. C. E. Gernandt, second edition, 1913.

[41] Ferdinand Lips. *Gold Wars The Battle Against Sound Money As Seen From A Swiss Perspective*. The Foundation for the Advancement of Monetary Education, 2001.

[42] Johnson; Wuille Pieter Lombrozo, Eric; Lau. Bip 0141 segregated witness. `https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki`, December 2015. (Accessed on 2019-02-13).

[43] Carl Menger. On the origins of money. *Economic Journal*, pages 239–55, 1892.

[44] Satoshi Nakamoto. fix openssl linkage problems,. https://github.com/bitcoin/bitcoin/commit/a30b56ebe76ffff9f9cc8a6667186179413c6349. (Accessed on 2019-02-19).

[45] Satoshi Nakamoto. only accept transactions sent by ip address if -allowreceivebyip is s.... https://github.com/bitcoin/bitcoin/commit/172f006020965ae8763a0610845c051ed1e3b522. (Accessed on 2019-02-19).

[46] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[47] Satoshi nakamoto. Bitcoin talk forum post. https://bitcointalk.org/index.php?topic=57.msg415#msg415, 2010. (Accessed on 2019-02-04).

[48] Satoshi Nakamoto. Bitcoin talk forum post. https://bitcointalk.org/index.php?topic=13.msg46#msg46, 2010. (Accessed on 2019-02-04).

[49] Satoshi Nakamoto. Bitcoin talk forum post. https://bitcointalk.org/index.php?topic=195.msg1611#msg1611, 2010. (Accessed on 2019-02-11).

[50] Federal Reserve Bank of Saint Louis. Functions of money.

[51] R. H. Patterson. On our home monetary drains, and the crisis of 1866. June 1870.

[52] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. 2016.

[53] Franklin D. Roosevelt. Executive order 6102. https://en.wikipedia.org/wiki/Executive_Order_6102#/media/File:Executive_Order_6102.jpg.

[54] Franklin D. Roosevelt. The gold reserve act. http://legisworks.org/congress/73/publaw-87.pdf, January 1934.

[55] Ammous. Saifeadean. *The Bitcoin Standard*. Wiley, 2018.

[56] Nick Szabo. Unenumerated. http://unenumerated.blogspot.com/.

[57] Nick Szabo. Smart contracts. 1994.

[58] Nick Szabo. Shelling out. 2002.

[59] Warren E. Weber. A bitcoin standard: Lessons from the gold standard. October 2015.

[60] Pieter Wuille. Bip 0032 hierarchical deterministic wallets.

[61] Pieter Wuille. Bip 0103 block size following technological growth.