Bern University of Applied Sciences

Informatics Seminar

InterPlanetary File System
IPFS

Author:
Martin Schmidli

Teacher:
Kai Brünnler

Bern, May 15, 2017

Contents

1 Abstract **2**

2 Introduction **3**
2.1 Origin . . . 3
2.2 Name . . . 3
2.3 Project state . . . 4
2.4 Knowledgebase . . . 4

3 Terms **5**
3.1 Content Adressing . . . 5
3.2 Multiformats . . . 6

4 Layer-Architecture **9**
4.1 Identities . . . 10
4.2 libp2p - network, routing, exchange . . . 12
4.3 Exchange - Bitswap . . . 16
4.4 IPLD - Merkle DAG . . . 19
4.5 State Mutability - IPNS . . . 24

5 Goals of IPFS **27**
5.1 Offline functionality . . . 27
5.2 Enhanced Protocol . . . 27
5.3 Permenancy . . . 28
5.4 Decentralization . . . 29
5.5 Security . . . 29

6 Today **30**
6.1 Usage . . . 30
6.2 Software & Private Cloud . . . 30
6.3 Competitors . . . 31
6.4 Outlook . . . 31

Chapter 1

Abstract

IPFS (InterPlanetary Filesystem) is an open source protocol which can be used to run a distributed filesystem. IPFS was invented to tackle the drawbacks of the Internet and the Internet protocol suite we are using today.

To understand why IPFS was invented, we first have to understand the issues we have today. This report will outline how IPFS works and what kind of issues it tries to resolve.

Chapter 2

Introduction

IPFS stands for InterPlanetary Filesystem. It's an open source Internet Protocol which can be used to run a distributed filesystem. The developers didn't invent IPFS from scratch. In its core, IPFS takes advantage of existing ideas like Kademlia DHT, BitTorrent and Git. The main intention behind IPFS is to tackle the issues we are having in today's Internet/Web. In the eyes of the inventors, the protocol should be seen as an upgrade or even as a replacement of the existing technologies like HTTP. Global data distribution should be simplified and be implemented in the protocol itself rather than prescribe a separate distribution mechanism [1].

All computers participating in an IPFS Network are called peers or nodes. IPFS is a peer 2 peer system, there are no central servers managing the network.

2.1 Origin

The Development of IPFS was started in 2014 by Juan Benet, former Stanford Student and founder of the company Protocol Labs. Protocol Labs and contributors of the Community are developing IPFS further [2].

2.2 Name

The name was chosen as a tribute to J. C. R. Licklider, a computer scientist who came up with an idea of an "intergalactic network" of computers in 1962 [3]. He imagined a global network of computers, able to talk to each other and exchange data. During his time working at DARPA (Defense Advanced Research Projects Agency) he influenced many people with his ideas. DARPA later started the ARPANET Project and laid the foundation stone for today's Internet. Many important technologies, for example, TCP/IP where invented or funded during this project [4].

2.3 Project state

As of 19.04.2017 the Specifications of the IPFS Protocol are still being developed and hasn't completed yet. The developers state, that the core parts of the specs have reached a reliable or stable state. No official RFC request have been submitted to the IETF. An Implementation of the Protocol, written in the programming language Go and some utilities have already been published. Implementation in other programming languages Javascript and Phyton are in developing [5].

Different sources state different facts how the system works. Many topics are still discussed intensively. This makes it very difficult to do a report about IPFS. The main data source of this report was the project's GitHub page and a white paper release by Juan Benett [25].

2.4 Knowledgebase

To understand the following chapters you should get yourself familiar with

- Hash functions
- Distributed Hashtables
- Bittorrent
- Public-key cryptography

Chapter 3

Terms

This chapter will get you familiar with some basic terms later used during this document.

3.1 Content Adressing

HTTP is addressing data with URLs. In IPFS the data is addressed by its contents hash.

3.1.1 HTTP

Data which is access by HTTP is addressed with its **hostname**, **port**, **path** and the **filename or search word** [6]. If the location of the file changes and no redirection was created, the link gets useless.

Example 3.1.1.1. URL scheme used for HTTP

http : // hostname [: port] / path [? searchwords]
https://raw.githubusercontent.com/github/gitignore/master/TeX.gitignore

3.1.2 IPFS

Data added to the IPFS network can be addressed by the hash of its content. The hash builds the link to data, document, picture... Every time we add data, the hash will be calculated. When the content of the file changes the hash/link will be changed as well. The older content will still be accessible with the old hash value. The link and the data are immutable.

Example 3.1.2.1. IPFS Link

/ipfs/QmbYzb3nScopAnfkoUpRWUFVv856uSWpSRc2KYM1FSBJxr

This scheme has been built similar than Unix filesystem paths. This design choice makes it possible to mount the IPFS filesystem directly into a Unix System. The IPFs filesystem can then be used by applications running on the Unix machine.

3.2 Multiformats

Multiformat is a collection of protocols/formats. They try to extend existing formats by adding self-describing components. Two multiformats which are used in IPFS are Multihash and Multiaddr. This chapter will give you a short introduction to these formats. To understand these formats is necessary as they are used everywhere in the IPFS project.

3.2.1 Hash functions - Multihash

Multihash is a protocol/format invented and maintained by the company Protocol Labs. Multihash defines a hash format, which is used in the IPFS project to encode Hashes. Every hash generated will be stored with two additional values; the Function code of the hash function, which was used to generate this hash and the length of the hash.

$$\text{Multihash Format} = \text{Hash Function Code} + \text{Hash Length} + \text{Hash}$$

The hash function codes were set by the developers. For your own projects, you can easily implement your own code hash function reference list.

codec	description	code
identity		0x00
sha1		0x11
sha2-256		0x12
sha2-512		0x13

Figure 3.1: Hashfunction code list

Multihash should allow the software using IPFS to upgrade the hash functions more easily [7]. For example, we hard coded SHA1 as a hash function into our program. If somebody would be able to break the SHA1 hash function we would have to replace the hash function. This would lead to longer hash values which might break our program. By using a more generic format the hash function can be switched very quickly.

By default, IPFS Nodes are using the sha2-256 hash function. If a multihash has been generated it will be encoded by Base58 [8].

Example 3.2.1.1. sha1

Functioncode:
0x11, when we look up the Multihash Function Table we can find **sha1**. This hash was generated by the sha1 hash function.

Length:
0x20 = 32, means 32 * 8 Bit = 256bit. The Hash is 256bit long.

Hash Digest:
Hash values

3.2.2 Peer Adressing - Multiaddr

Multiaddr is used to represent common network adresses in a different way.

Example 3.2.2.1. Normal representation of a websocket url:

ws://1.2.3.4:5678

Multiaddr address:

/ws/1.2.3.4/tcp/5678

Like in Multihash were every hash function has a function code, in Multiaddr every protocol has a function code.

```
code,   size,   name,   comment
4,      32,     ip4
6,      16,     tcp
17,     16,     udp
33,     16,     dccp
41,     128,    ip6
53,     V,      dns,    reserved
54,     V,      dns4
```

Figure 3.2: Multiaddr code list

With this change of the format and the substitution of the protocol string by a function code, the hole adress takes up less space in the binary format.

<1 byte ws code><4 byte ipv4 addr><1 byte tcp code><2 byte tcp port>

Chapter 4

Layer-Architecture

IPFS is made of a stack of different software modules/libraries.

Software-Modul	**Level**	**Layer**	**Function**
	7	Applications	Access the IPFS network
IPNS	6	Naming	creates immutable links
IPLD	5	Merkle DAG	creates a relationship between data blocks
libp2p	4	Exchange	exchange data blocks
	3	Routing	data & peer lookup
	2	Network	connect to other nodes
	1	Identities	peer identity and verfification

This chapter will guide you from the bottom of the stack to the top to explain in detail how IPFS works.

4.1 Identities

The identities module manages the node identity and the verification of other peers.

During the initialization of a node, a 2048-bit RSA key pair (public, private) is generated. A hash function is used to generate the hash of the public key. This generated hash is then used as Peer ID [9].

$$\text{Peer ID} = \text{multihash(public key)}$$

4.1.1 Trust

There is no central certification authority in place which can be used to check if another peer can be trusted. The whole system has been designed to be self-verifying.

Example 4.1.1.1. Communication

Peer A contacts another peer B. The Nodes exchange Node ID and Public Key. Peer A generates the hash of the public key of peer B. If this generated hash doesn't match the NodeID of peer B the connection will be terminated.

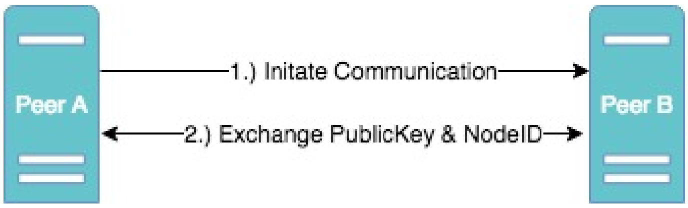


Figure 4.1: Node trust setup

Example 4.1.1.2. Attack scenario

Peer C tries to steal the identity of peer A. Peer C is using the public key and the Node ID from peer A. Peer B will establish a connection with peer C as the public key is valid and the Node ID matches. The sent data from B to C will be encrypted by Peer A's public key. As Peer C doesn't hold the private key, the data can't be decrypted.

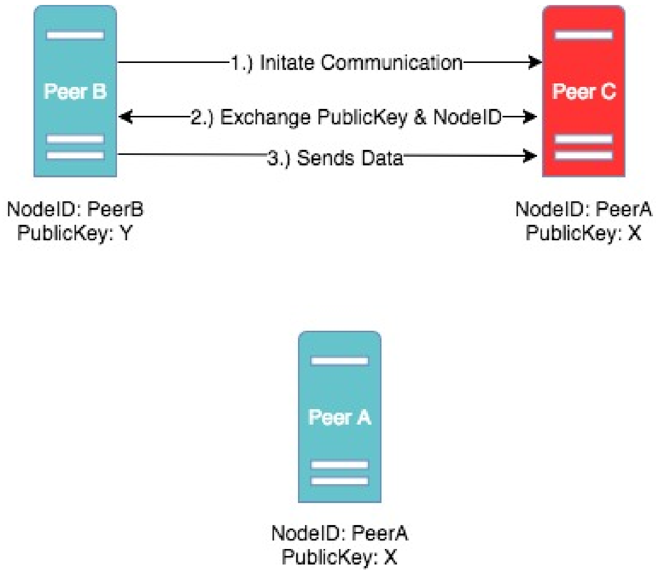


Figure 4.2: Node trust attack

4.2 libp2p - network, routing, exchange

During the development of IPFS, the team learned how hard it was to run IPFS on many different devices with different requirements to the network [10]. It was impossible for them to provide one single protocol to work everywhere. They started to redefine the network, routeing and exchange layer and created an independent library call libp2p. This library is responsible for the network connectivity between nodes, the data lookup on the network and the data block exchange among them. In its core libp2p basically is a collection of p2p protocols and defines interfaces to use them. There are many different protocols supported which can be used depending on the situation of the node and its network. The protocols are implemented like blocks and can be combined dependent on the needs of the application developers.

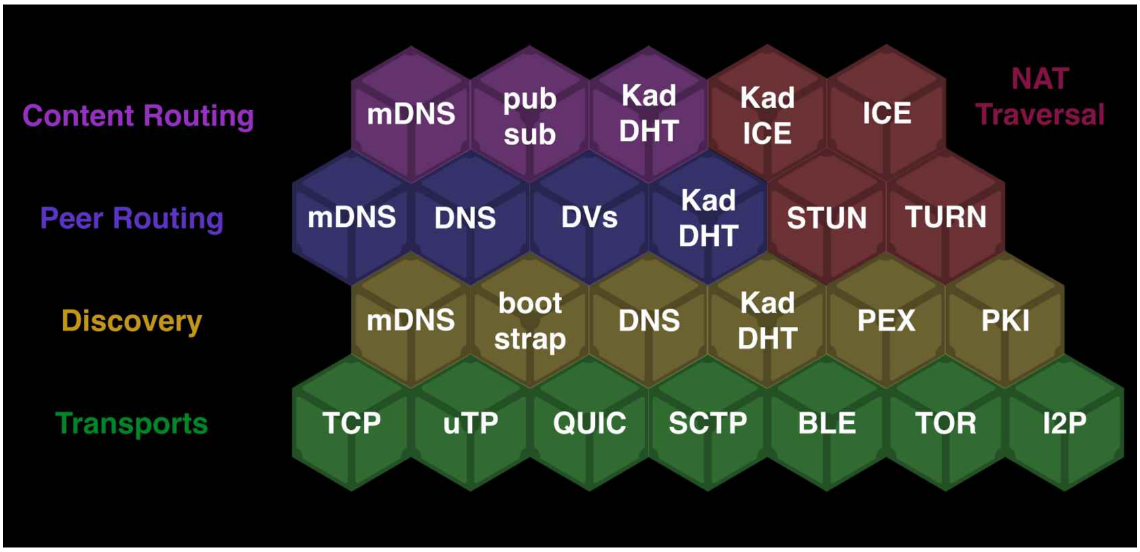


Figure 4.3: libp2p protocol collection

4.2.1 Network

The network layer establishes connections to other peers in the IPFS network.

libp2p has been designed to be transport independent. It can run without IP stack for example on NDN [11], XIA [12] and Bluetooth[13].

libp2p doesn't determine which kind of protocol should be used. For every layer, the developer is able to choose which protocols he would like to implement. With this choice, the developer decides how his application will work with other applications on the network.

For example, if the developer decides to route the data over the TOR (the onion router) network the application will not be able to exchange data with nodes within the same network. It's a tradeoff, the application will be more secure, as it benefits of the anonymity of the TOR network but you will give up performance.

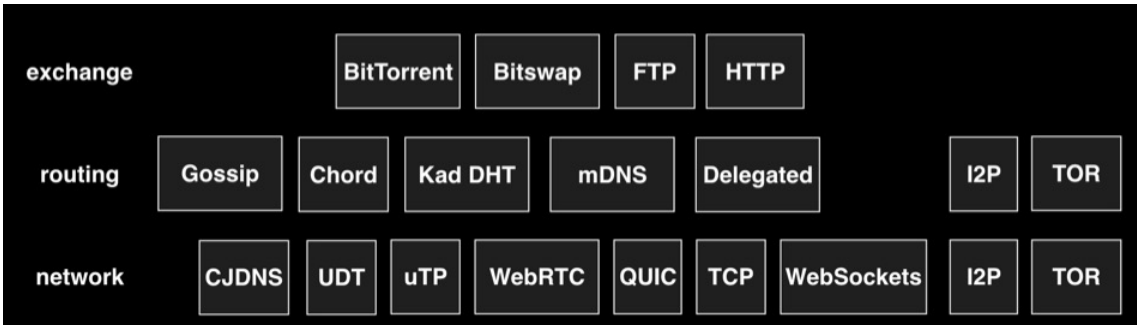


Figure 4.4: libp2p protocols.

The network layer tries to make the node as available as possible, over the internet and in the local lan.

Example 4.2.1.1. Interfaces where my peers listens for a connection

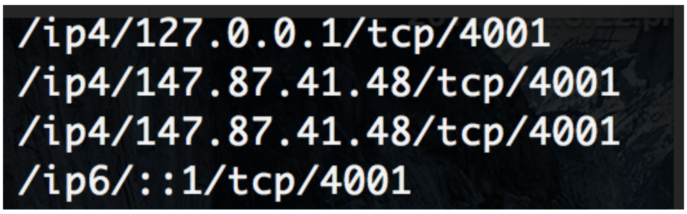


Figure 4.5: IPFS listen on interfaces

4.2.2 Routing

This Routing part of libp2p gives the IPFS node the ability to find the network address of other nodes and nodes which can serve specific data blocks.

The peer routing module takes in a key (hash) and responds with one or more Peer Info Objects. The Peer Info contains the PeerID and its multiaddr addresses. With these informations, the other node can be contacted.

The system is easily expandable. The protocols just need to fulfil the requirements of the Peer Routing interface. Currently, there are two Routing systems proposed and implemented.

Every node in a routing system can contact all the participating peers. If data should be separated from the public IPFS network a separate instance of a routing system needs to be established. For example: create your own DHT.

DHT

A DSHT (distributed sloppy hash table) based on S/Kademlia and Coral can be used to store Key (Hash) Value (Peer ID) pairs. A DHT is a Hashtable which is distributed on all the participating IPFS peers. Every peer will be responsible for a percentage of all blocks added to IPFS. If a node leaves the IPFS network the responsibility of his part of the DHT will be forwarded to another node.

If bigger data files > 256KB is uploaded to IPFS the file will be split up into smaller chunks. Every chunk is added to the DHT and referenced to the node. For every chunk a new entry **Block Hash → Node ID** is added to the DHT.

If a small file < 256KB is published, the data is directly stored on the DHT. A new entry **Block Hash → Data** is added to the DHT.

This process of publishing data is called providing [14].

If a node wants to find out where it can get the specific data blocks, it can use the DHT to determine the nodes which can serve those blocks.

Example 4.2.2.1. Add a large File to IPFS
In this example the folder structure "ipfsdemo/pictures" was added to the ipfs network. Inside the folder was placed a picture "boys_icefield.png". The size of the picture is around 4.5MB. The file was split up inside smaller chunks.

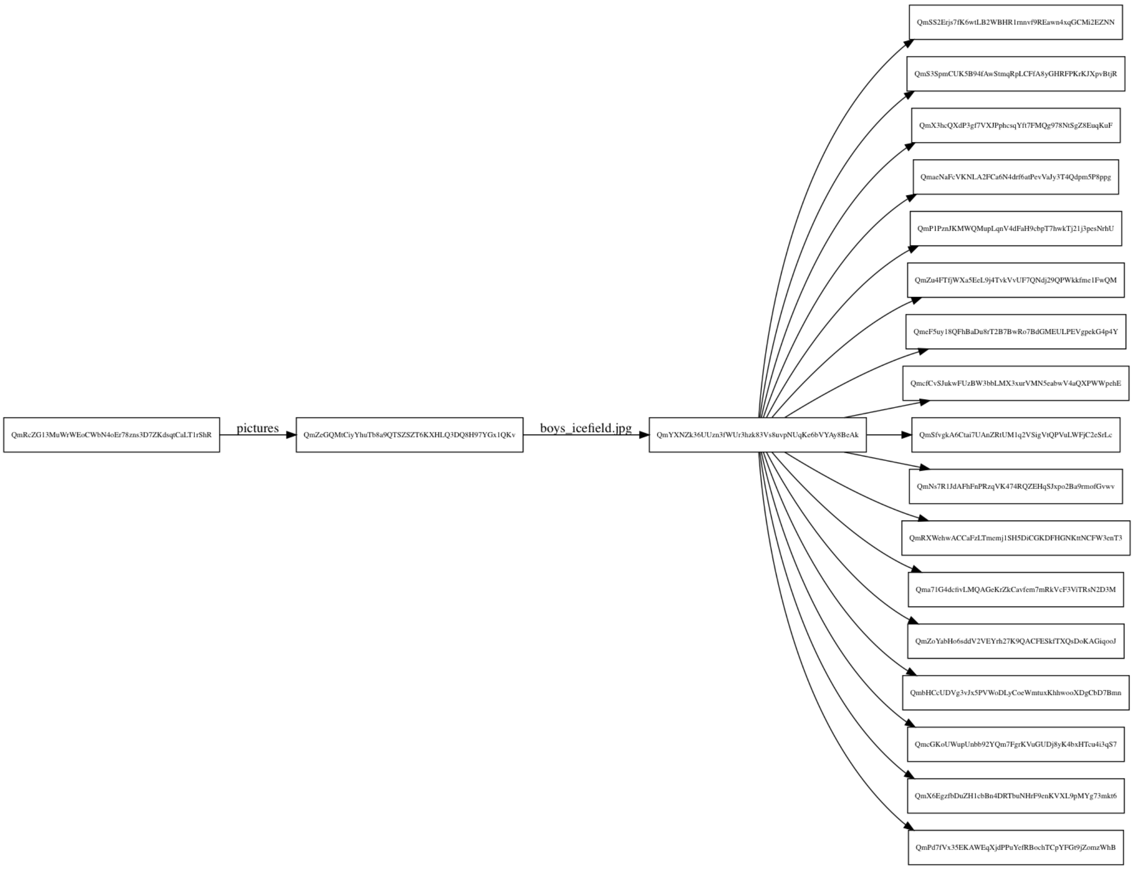


Figure 4.6: Add a big file

mDNS

Uses the mDns protocol to check if a node in the local network holds a specific key.

4.3 Exchange - Bitswap

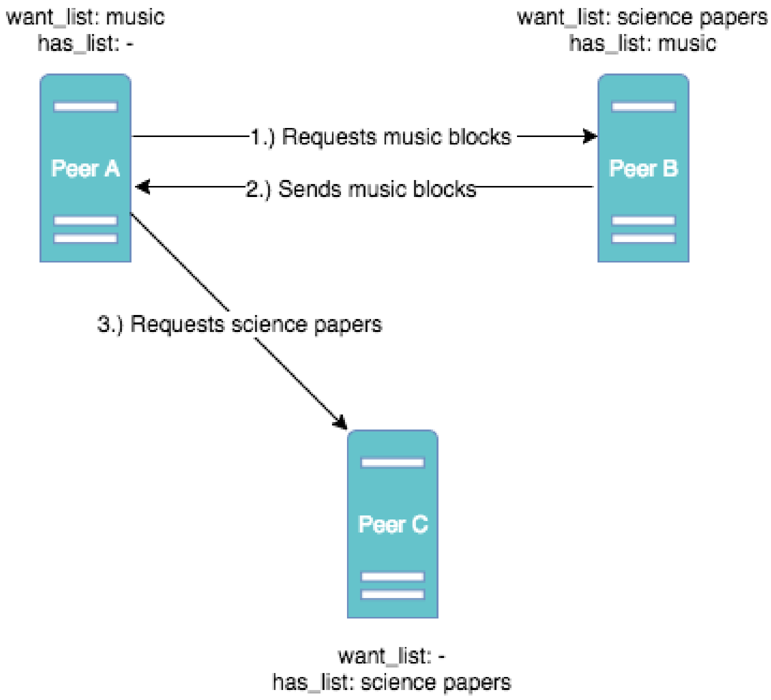
Many exchange protocols can be used with IPFS, for example, FTP, HTTP, BitTorrent or Bitswap.

Bitswap is a BitTorrent-inspired software module used to exchange data blocks between the nodes of the IPFS network. When a user requests a piece of data the specific blocks are added to the nodes ”want_list” list. Bitswap contacts other peers which own the data according to the Routing system (example. DHT) and sent them the ”want_list”. Peers keep track of the blocks which they own by putting them into their ”have_list’. If the receiver has blocks which another node requested, he will send the blocks back to the requester. ’When the blocks arrive, the requester will immediately send out a ”Cancel’ signal. Other nodes will stop sending those blocks. After the blocks arrived the requester move those blocks from the want_list to the have_list [15].

4.3.1 Exchange Strategies

When a bitswap node requests data (Peer A), the other nodes (Peer B) will only send the requested data if they receive blocks in return. Most of the time Nodes store very different data and a number of blocks which they could offer in return is very limited. Sometimes Node doesn’t own any data the other node is needing. This puts the requesting node (Node A) in a bad position as it has nothing he could offer in exchange. To resolve the situation, Node A will start to download the data Node B has on its want_list after all his own data requests are completed. The own needs have a higher priority.

Example 4.3.1.1. Let's make a short example of how this strategy works.
We have 3 Nodes, A, B and C. Node A wants blocks of a "music file" but currently the Node doesn't store any data. Node B want's blocks of a "science paper" file and owns blocks of the "music file"→ Node A has no blocks which Node B actually needs.



After Peer A got all the music blocks he wanted, he will start to look for science paper blocks.

4.3.2 Ledger

Bitswap nodes keep track of the amount of data they send and received to other nodes in a digital ledger. We speak of credit or debt between the nodes. It's an indicator of the trust and the connection between two nodes. When a new connection is established, the nodes exchange their ledger. If the ledger data doesn't match, for example, if one of the nodes maliciously changed the data, the ledger and the connection will be resetet. Depending on the Bitswap strategy good connections between nodes, which already exchange a lot of data could be preferred. Nodes with a high debt could be neglected.

Automatical Download Problematic

This exchange mechanism leads to data download even if the node doesn't actually need the data by itself. The data are cached and increase the availability of the blocks. This strategy is not yet implemented in bitswap. Today's implementation is syncing blocks without any conditions and doesn't fulfil the defined protocol specifications.

There are ongoing discussions if this feature should be enabled by default or if there should be a config option to enable it [16]. If this feature is enabled by default, IPFS nodes would download data without any knowledge of the user. This would lead to an extensive usage of the local storage and could lead to legal problems. To avoid the legal problematic, there needs to be a system in place to avoid the download of Copyright Protected Data and to implement Digital Right Management (DRM). If and how this will be implemented is currently discussed in the community [24]. Some possible approaches which are considered are

- White & Blacklist of certain blocks and objects
  Users can enable filters which kind of content they want to download
- Public trusted list of Copyright Protected content.
  Prohibits the sharing of files on this list

4.3.3 Data Availabiliy

In today's implementation of IPFS, data is not actively distributed in the IPFS Network. If a node requests data, it will be downloaded and therefore replicated. There is no higher instance which is controlling how many times a specific data block is physically stored somewhere. Example; If one data block is added from Node A to the IPFS network and this node goes down, the data is no longer available.

The developers try to solve this problem by designing a new software which is running on top of IPFS called IPFS-Cluster. The IPFS-Cluster Service will be running on the IPFS Node and makes sure, that data remains available. To cluster data, it will be "pinned" to a Node. Pinned data will not be removed localy if its running out of disk space.

Data will be replicated/pinned among IPFS-Cluster nodes according to a predefined replication factor. Example; Replication factor 2, data us store on 2 Nodes. If a node with pinned data goes down, the data automatically is replicated to other cluster nodes.

4.3.4 Difference to Bittorrent

Bittorrent nodes can only request blocks from one torrent. Data blocks are not shared among torrents.

4.4 IPLD - Merkle DAG

IPLD stands for Inter Planetary Linked Data.
The Merkle DAG which is part of IPLD gives a structure to the loose datablocks in the Routing System.

IPLD defines

- Merkle directed acyclic graph
  Used to create a relationship between the IPFS objects added to the Routingsystem for example DHT.
- Merkle-Links
  Edges /links between the objects of the DAG
- Merkle-Paths
  Unix-like path for traversing through the Merkle DAG
- IPLD Data Model
  Defines the data structure of the DAG objects, based on the JSON format

4.4.1 Merkle DAG & Merkle Links

A Merkle DAG is a Directed Acyclic Graph where the objects are connected with merke-links. The Merkel DAG has the following features:

- No circle is allowed
- Content is adressed by hashes → merkle-link

  Example: /ipfs/QmKLTg7BZ ... 63JtDuvs2k

  Using merkle-links provides some advantages

  - Accessed data can be verified/integrity checked by the hash.
  - Data structures can't be modified as a change results in a new hash.
- Objects of data will be stored only once in the DAG. If data is added twice it will be reference by its hash → Automatic deduplication

The Merkle DAG Datastructure is different than a Merkle Tree. The Merkle DAG doesnt need to be balanced, the added data is automatically deduplicated and all nodes can contain data whereas in a Merkle tree only leaf nodes can contain data.

4.4.2 IPLD Data objects

IPLD defines a basic datamodel based on JSON. The objects can be structured by the developer. This makes the DAG very flexibel. Various objects can be modelled to create complex datastructures.

Merkle Links

A merkle-link object contains of a key ”/” which defines a link to another object and the actual link value. The link value can be a multihash or the absolut path with the ”/ips” präfix.

```
{ "/" : "/ipfs/QmKLTg7BZ ... 63JtDuvs2k" }
```

Objects

IPLD objects are basic JSON objects. They can be linked with merkle-links.

```
{ "name": "Martin Schmidlli" }
```

Example 4.4.2.1. Example of linked objects

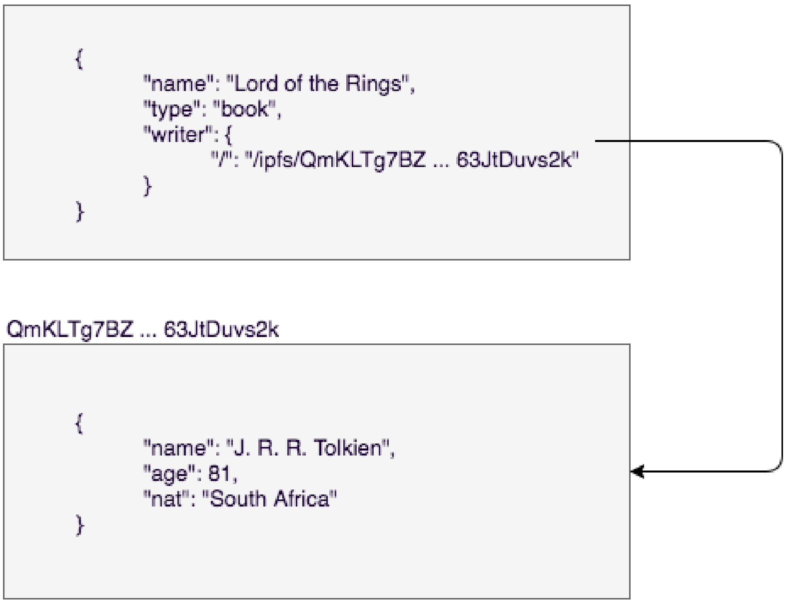


Figure 4.7: IPLD linked objects

Example 4.4.2.2. Add small Files <= 256kB

If the added data is smaller than 256kB, it's is stored directly into the DAG Object.

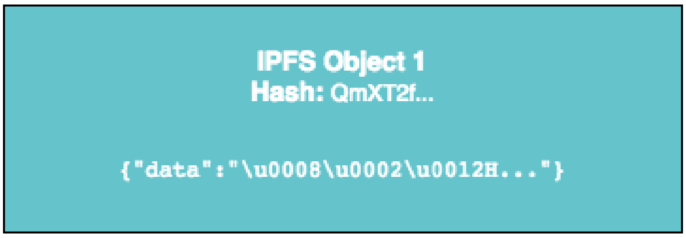


Figure 4.8: Small Object <= 256kB

Example 4.4.2.3. Add large Files > 256kB

If we add a file bigger than 256kB, the file will be split up into blocks. The generated hash will be the root node in the DAG. This root node will contain merkle-links to the subblocks/objects. The data segment is used to declare that this object represents a big file.

If a node requests a block, this block can come from any node owning one of these blocks even if it's not from the same "file" [21]. Multiple IPFS objects can share blocks. As IPFS is addressed by hash, the actual data block is stored once.[22].

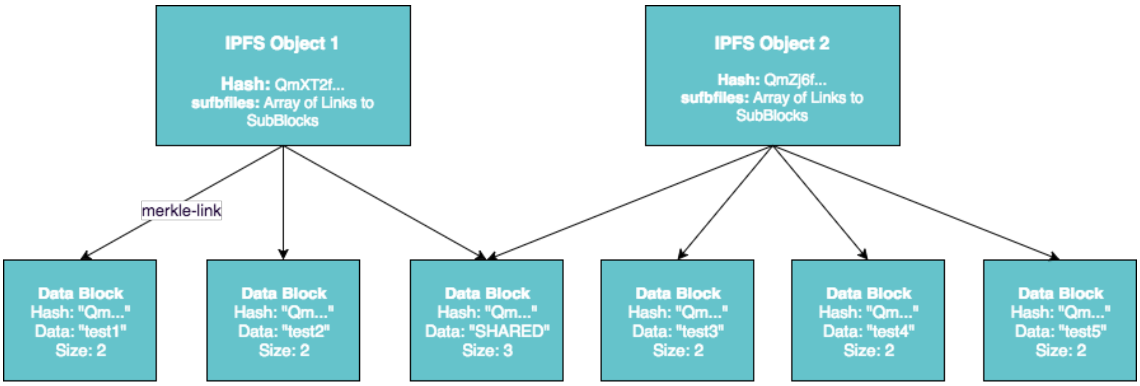


Figure 4.9: Abstract visualization how blocks are handeled

4.4.3 Merkle Paths

This is our basic object at the link /ipfs/QmKLTg7BZ ... 63JtDuvs2k

```
{ level1 : {
        level2: "Hello World"
}}
```

We can use merkle paths to reference to values or other object relative to the root.

/ipfs/QmKLTg7BZ ... 63JtDuvs2k/level1/level2

We can traverse through the DAG Object /ipfs/QmKLTg7BZ ... 63JtDuvs2k like in a unix filesystem and access values of level2.

Example 4.4.3.1. ipld cat

```
> ipld cat --json QmKLTg7BZ ... 63JtDuvs2k/level1/level2
"Hello World"
```

4.4.4 Encryption

Built-in file level encryption is one of the many missing features in IPFS today. The feature is listed in the IPFS whitepaper and will be added in a later release of the protocol. [19]. Data added to the IPFS network will automatically be encrypted. If private data is added to the ”public” IPFS network, the developers recommend to encrypt it manually [20].

4.4.5 Complex Datastructures

The basic DAG objects can be used to build more advanced data structures. Till today there are many implementations already developed:

- git
- Zcash (crypto currency)
- Bitcoin /crypto currency)
- Ethereum
- Unix Filesystem

Example 4.4.5.1. IPFS Git commit message


```
{
  "tree": {"/": "e4647147e940e2fab134e7f3d8a40c2022cb36f3"},
  "parents": [
    {"/": "b7d3ead1d80086940409206f5bd1a7a858ab6c95"},
    {"/": "ba8fbf7bc07818fa2892bd1a302081214b452afb"}
  ],
  "author": {
    "name": "Juan Batiz-Benet",
    "email": "juan@benet.ai",
    "time": "1435398707 -0700"
  },
  "committer": {
    "name": "Juan Batiz-Benet",
    "email": "juan@benet.ai",
    "time": "1435398707 -0700"
  },
  "message": "Merge pull request #7 from ipfs/iprs\n\n(WIP) records + merkledag specs"
}
```


Figure 4.10: DAG Objects - Git commit

Example 4.4.5.2. IPFS Bitcoin block


```
{
  "parent": {"/": "Qm000000002CPGAzmfdYPghgrFtYFB6pf1BqMvqfiPDam8"},
  "transactions": {"/": "QmTgzctfxxE8ZwBNGn744rL5R826EtZWzKvv2TF2dAcd9n"},
  "nonce": "UJPTFZnR2CPGAzmfdYPghgrFtYFB6pf1BqMvqfiPDam8"
}
```


Figure 4.11: DAG Objects - Bitcoin block

4.5 State Mutability - IPNS

All the data added to the IPFS Network is immutable. The data can't be changed anymore. New content generates new data and therefore a new hash key. If you change your data, people with the old hash key (link) will always be directed to the old version of the data. To solve this issue, the developers created the InterPlanatery Naming System (IPNS). IPNS creates self-certifying mazieres links, invented by David Mazières, to create a reference to the newest content.

Example 4.5.0.1. Mazieres Links
If we add data to ipfs an ipfs link gets created

/ipfs/QmU224/

A public/private keypair is needed to create one IPNS link.

1.) The hash of the public key will be generated
2.) The IPNS link is built with the public key hash
3.) If we resolve this IPNS link it should give us back the newest content, in this case: /ipfs/QmU224/

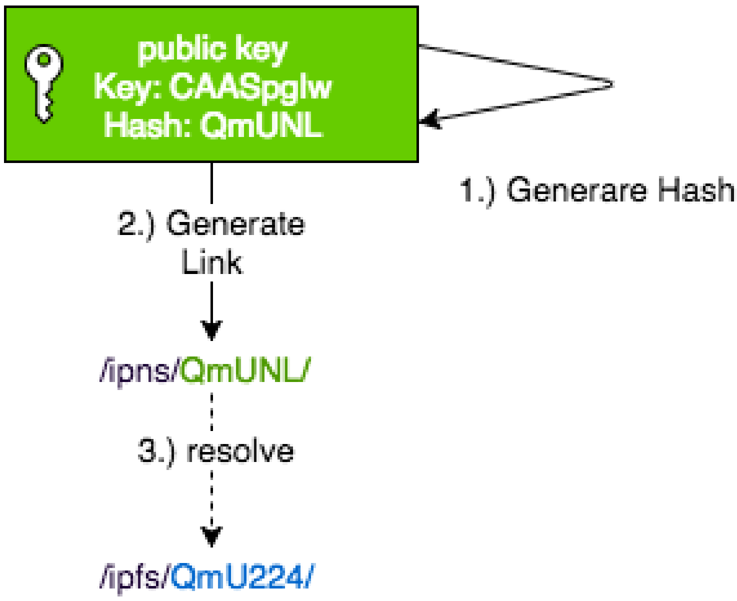


Figure 4.12: IPNS - Step 1

1.) To make the resolving process authenticated, the private key is used to sign a pointer object. The pointer object creates a link between the Public Key and the content. Three values are stored within this pointer object: The value (target hash), the key (public key hash) and a signature.
The pointer is another object in the IPFS Universe called IPRS Record. IPRS Records are stored on the Merkle DAG and can be used for authentication purposes.

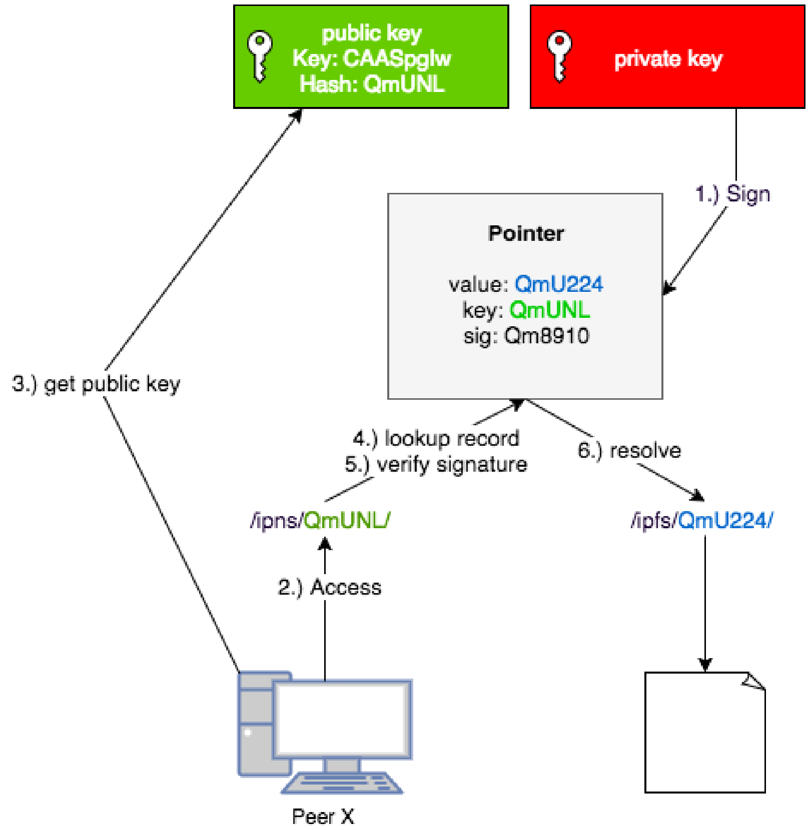


Figure 4.13: IPNS - Step 2

2.) Peer X wants to resolv the IPNS link
3.) Gets the public key
4.) The peer will look for a pointer record in the DAG with the specific hash of the public key.
5.) The signature will be verified with the public key
6.) The peer will find the target hash value. This value will then be used to build the final ipfs link.

If the content should be updated, only the target hash of the pointer needs to be changed. The ipns link stays the same.

4.5.1 DNS Links

So far we created a static link to the content we want to present to the world. The systems works, but those IPNS links are not very user-friendly. We can use the existing DNS System to create user-friendly links and map them to our IPNS links.

Example 4.5.1.1. Map DNS to IPNS
We can add a TXT Record to the DNS Zone of our domain. Example domain: ipfs.io [17]

dnslink="/ipns/QmWGb7PZmLb1TwsMkE1b8jVK4LGceMYMsWaSmviSucWPGG"

If we enter /ipns/ipfs.io the ipfs client will resolve the DNS TXT entry and fetch the data from the ipns link above.

Chapter 5

Goals of IPFS

The developers of IPFS state: The Internet/Web of today has many issues. The existing protocols we have today, have some major design issues or are not good enough anymore to satisfy the needs of today's Web and his Users [26]. In this chapter, I will list the main problems the developers and the community sees and try to analyse how IPFS could solve them.

5.1 Offline functionality

Imagine you are sitting with your colleagues at work. You all work together on a document. You are using a WebApplication to collaborate with each other. Suddenly the internet connection is lost. You are all sitting in the same local network but you are unable to share your version of the document with the others. All the data needs to be synced with the backbone service, for example, Google Docs. and then down again to the other clients / your colleagues. The Applications should be able to talk to each other and shouldn't be dependent on a service somewhere on the internet.

The library libp2p which is integrated into IPFS enables peers and their content to be detected without any backbone service. The IPFS nodes can detect who owns which kind of data and request them for example by using mDNS. Static Web applications can be served directly through IPFS. If new data is added to IPFS through the Application it will be available for other users as well.

5.2 Enhanced Protocol

The most used Protocol from the Internet Protocol Suite by far is HTTP. It follows the Server-Client communication model. The Client establishes a connection to one server, sends requests and get responses back from the server. Even if there are load balancing mechanisms in place it will be served by the by one host only.

The alternative communication model would be Peer to Peer short. p2p. The Clients establishes a connection to multiple "peers" which will serve the requested data. Every peer will deliver a fraction of the data. Those data "pieces" will be downloaded simultaneously.

IPFS establishes a peer 2 peer network. Data can be simultaneously downloaded from different sources. This "feature" makes IPFS more reliable against DDoS attacks because the data is distributed all around the world.

5.3 Permenancy

I guess every person which used an Internet Browser before has seen a message like "Error 404" or "Site not found".

Figure 5.1: Github 404 Error Page

The specific content you tried to reach has been deleted or has been moved by the provider of the site. All the links provided are now useless. Everyday content gets moved around or is deleted forever. Sometimes it's just a silly picture of a cat but mostly its data somebody actually would have needed.

Nevertheless, there are several companies already trying to solve this issue by archiving all the data available on the internet. The most well known would be "Internet Archive" with its "Wayback Machine" Web application. They are using web crawlers to get the data and store them. The User is able to use theWayback Machine to view the web page back then when the Internet Archive made a snapshot of it [27]. By October 2016 they managed to archive 273 billion web pages from over 361 million websites, which resulted in a data size of 15 Petabytes (15'000 Terabytes) [28].

All content added to IPFS is hashed and added permanently to the network. Once the IPFS cluster feature has been finalised the network will make sure the data stays online. The data can never be removed again as long as nodes are connected to the network serving this specific content. If some nodes break down the data get's rereplicated and stays available.

5.4 Decentralization

Big data hosting companies like Google or Amazon own datacenters worldwide. The developer can buy the service from them to cluster their application data and distribute it all over the world. This ensures the availability of your Application even in the case of a datacenter outage caused by a natural catastrophe or other technical issues. Those incidents aren't fictions and cost millions if they happen. The latest example, 28.02.2107 a site of the Amazon S3 System located in Northern Virginia gets unavailable. Thousands of web pages and web applications stopped working. An Amazon technician wanted to remove a small subset of servers. He entered the command wrong and more servers than planned were removed. This led to a 4h outage of the hole Site [29]. Estimated cost 150 Million US Dollar [30].

Clustering of data is not standard today. To make your data highly available is cost intensive. To achieve the same level of high availability cost effective without any kind of big service provider like Google is near impossible.

The second group of big data handlers are the so-called CDNs. These Content Delivery Networks host data worldwide and act as a proxy for data requests. Services of these providers are used to bring content closer to the customers to reduce latency and download time.

Data on the IPFS system is meant to be permanent. Once added to the network the IPFS cluster service will distribute the data over multiple nodes

Close peers which can serve the blocks are preferred. The overall latency is reduced and the data is downloaded faster. The experience for the user is much more pleasant. For us living in a country with good internet download speed, this feature will not change much. For other countries, in particular developing countries, it's a big deal.

5.5 Security

Todays data connections are secured. For exmaple by TLS. The data itself stored on the local machine is not encrypted. Files located at a data provider are encrypted but mostly the companies has the corresponding keys to them. Data can get monetized and missued by others.

IPFSs way around this is to encrypt the data not the connection. All data added to the ipfs network will get encrypted and will be only accessible bye the user himself. This feature is yet to be developed [19].

Chapter 6

Today

6.1 Usage

6.1.1 Webhosting

Static Webpages can be hosted easily on IPFS as they contain only static files like .css or .html. According to Juan Benet, more than 100'000 web pages have been uploaded to IPFS [31].

6.2 Software & Private Cloud

The IPFS protocol has been bundled with the popular software FreeNAS. Free NAS turns every computer into a NAS (Network attached Storage) [32]. By leveraging the IPFS network the data of the NAS can be shared and distributed.

6.2.1 Archive

The developers of IPFS started a new project to bring the content of Wikipedia on IPFS. This project was started due to the recent blockade of the Wikipedia Website in Turkey on the 29.04.2017 [33].

6.2.2 Blockchain

In the last years, the blockchain technology gains a lot of popularity. More and more projects have been introduced. Some of the well knowns are Bitcoin and Ethereum. Ethereum is a platform to run applications, so-called DAPPs, decentralised Applications. One Example of DAPPs are Smart Contracts. A Smart contract is a piece of software which stores some rules for negotiating the terms of a contract. If the rules are fulfilled, the program automatically executes the agreed terms [34]. The blockchain is the wrong place to store big data [35]. Some companies discovered IPFS to outsource the data. The data is placed on the IPFS Network and only the hash/link is stored in the blockchain.

6.3 Competitors

During the last years, the dev team created a very powerful yet easy to understand protocol which could have a massive impact on how we work today with the internet. Nevertheless, Protocol Labs aren't the first to come up with such an idea. There are other companies like Maidsafe [37] or SIA [36] trying to reinvent the web. They have a similar goal, us the unused disk capacity of a computer, decentralise and secure the data. In my opinion, IPFS takes it one step further by enabling developers to model a lot of different data structures like Git filesystems or entire blockchains. The other companies are more focused on private clouds. IPFS, compared to Maidsafe, lacks a lot of features like built-in encryption and anonymity. This is and will be possible with IPFS aswell, however, IPFS is still a young protocol and the features have yet to be developed.

6.4 Outlook

Like the competition did with SIA and Maidsafe, Protocol Labs is planning monetize IPFS. The new technology is called Filecoin. It will allow users to offer their disk space on their computer/server to the IPFS network and as a reward, they will gain Filecoin. Filecoin is an Altcoin and can then be traded to Bitcoin and from there to "real" money like US Dollar. If the user caches more data equals helps to make content available, he will receive more coins. This monetization should attract new user to IPFS and increase the available storage space worldwide.

It's hard to determine the future of IPFS. IPFS has no big spread. The protocol is still in the beta stage. A lot of big companies like the Wikimedia foundation are interessed to use it. Till the protocol doesn't reach a more stable stage/first release it will not be used in big software projects.

Bibliography

[1] https://github.com/ipfs/ipfs, 24.07.2017

[2] https://www.linkedin.com/in/jbenetcs/, 28.04.2017

[3] https://www.youtube.com/watch?v=HUVmypx9HGI 4:30, 22.04.2017

[4] http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet, 22.04.2017

[5] https://github.com/ipfs/specs, 29.04.2017

[6] https://www.w3.org/Addressing/HTTPAddressing.html, 23.04.2017

[7] https://github.com/jbenet/random-ideas/issues/1, 24.04.2017

[8] https://github.com/ipfs/faq/issues/22, 30.04.2017

[9] https://github.com/ipfs/faq/issues/238, 24.04.2017

[10] https://www.infoq.com/presentations/data-ipfs-ipld, min: 15

[11] https://named-data.net/ 14.05.2017

[12] https://www.cs.cmu.edu/~xia/ 14.05.2017

[13] https://github.com/libp2p/specs/blob/master/3-requirements.md

[14] https://github.com/ipfs/go-ipfs/issues/1396

[15] https://github.com/ipfs/specs/tree/master/bitswap, 29.04.2017

[16] https://github.com/ipfs/faq/issues/47, Q: but bitswap says..., 29.04.2017

[17] https://www.reddit.com/r/ipfs/comments/3m351b/discussion_permanent_content_dmca_and_illegal/?st=j24nd4on&sh=86fd55ce

[18] https://news.ycombinator.com/item?id=10229481 14.05.2017

[19] https://github.com/ipfs/faq/issues/22, 30.04.2017

[20] https://github.com/ipfs/faq/issues/116, 30.04.2017

[21] https://github.com/ipfs/faq/issues/4, 30.04.2017

[22] https://github.com/ipfs/examples/tree/master/examples/data, 29.04.2017

[23] https://medium.com/@ConsenSys/an-introduction-to-ipfs-9bba4860abd0, 29.04.2017

[24] https://github.com/ipfs/faq/issues/48, 29.04.2017

[25] https://github.com/ipfs/ipfs/blob/master/papers/ipfs-cap2pfs/ipfs-p2p-file-system.pdf?raw=true 2.05.2017

[26] https://ipfs.io/, Sector: The web of tomorrow needs IPFS today, 22.04.2017 23.04.2017

[27] https://en.wikipedia.org/wiki/Internet_Archive, 22.04.2017

[28] https://blog.archive.org/2016/10/23/defining-web-pages-web-sites-and-web-captures, 22.04.2017

[29] https://aws.amazon.com/message/41926/, 23.04.2017

[30] http://www.npr.org/sections/thetwo-way/2017/03/03/518322734/amazon-and-the-150-million-typo,

[31] https://www.infoq.com/presentations/data-ipfs-ipld 14.05.2017

[32] http://www.freenas.org/blog/announcing-freenas-10-alpha/ 14.05.2017

[33] https://ipfs.io/blog/24-uncensorable-wikipedia/ 14.05.2017

[34] https://en.wikipedia.org/wiki/Ethereum 14.05.2017

[35] http://www.coindesk.com/ethereum-meets-zcash-why-ipfs-plans-a-multi-blockchain-browser/ 4.05.2017

[36] http://sia.tech/ 14.05.2017

[37] https://maidsafe.net/ 14.05.2017

List of Figures

3.1 Hashfunction code list 6
3.2 Multiaddr code list 8

4.1 Node trust 10
4.2 Node trust attack 11
4.3 libp2p 12
4.4 libp2p 13
4.5 IPFS listen interfaces 13
4.6 Add big data 15
4.7 IPLD linked objects 20
4.8 Small Object 21
4.9 Big Object & Blocks 21
4.10 DAG Objects commit 23
4.11 DAG Bitcoin block 23
4.12 IPNS Step 1 24
4.13 IPNS Step 2 25

5.1 Github 404 Error Page 28