

# Distributed Hosting

A Blockchain Based Distributed Hosting Engine

Alex Oberhauser, Stavros Champilomatis and Maria Efthimiadou

DRAFT

Copyright © 2015 Alex Oberhauser, Stavros Champilomatis and Maria Efthimiadou

PUBLISHED BY THE *Distributed Hosting TEAM*

[HTTP://BLACKGATE.NETWORLD.TD](http://BLACKGATE.NETWORLD.TD)

This work is licensed under the Creative Commons Attribution 4.0 International License.  
To view this license visit <http://creativecommons.org/licenses/by/4.0/>.

*First released, March 2015*



# Contents

I

## Overview

<b>1</b>	<b>Introduction</b>	7
1.1	Terminology	7
1.2	Hosting Paradigm Shift	8
1.3	Use Cases	8
1.3.1	Simple Blog Hosting	8
1.3.2	Shared Access with Optional Paid Service	8
<b>2</b>	<b>Empowering Technology</b>	9
2.1	Blockchain	9
2.2	Tor Hidden Services	9
<b>3</b>	<b>Ecosystem</b>	11
3.1	Hosting Provider	11
3.2	Complementary Services	11

II

## Specifications

<b>4</b>	<b>Blockchain Specification</b>	<b>15</b>
<b>4.1</b>	<b>Transaction Definition</b>	<b>15</b>
4.1.1	Transaction Definition .....	15
4.1.2	CLONE Transaction Definition .....	16
<b>4.2</b>	<b>Block Definition</b>	<b>17</b>
<b>4.3</b>	<b>Mining Process</b>	<b>17</b>
<b>5</b>	<b>Protocol Specification</b>	<b>19</b>
<b>5.1</b>	<b>Blockchain Overlay Network</b>	<b>20</b>
<b>5.2</b>	<b>Data Overlay Network</b>	<b>20</b>
<b>5.3</b>	<b>Hosting Algorithm</b>	<b>20</b>

III

## Appendix

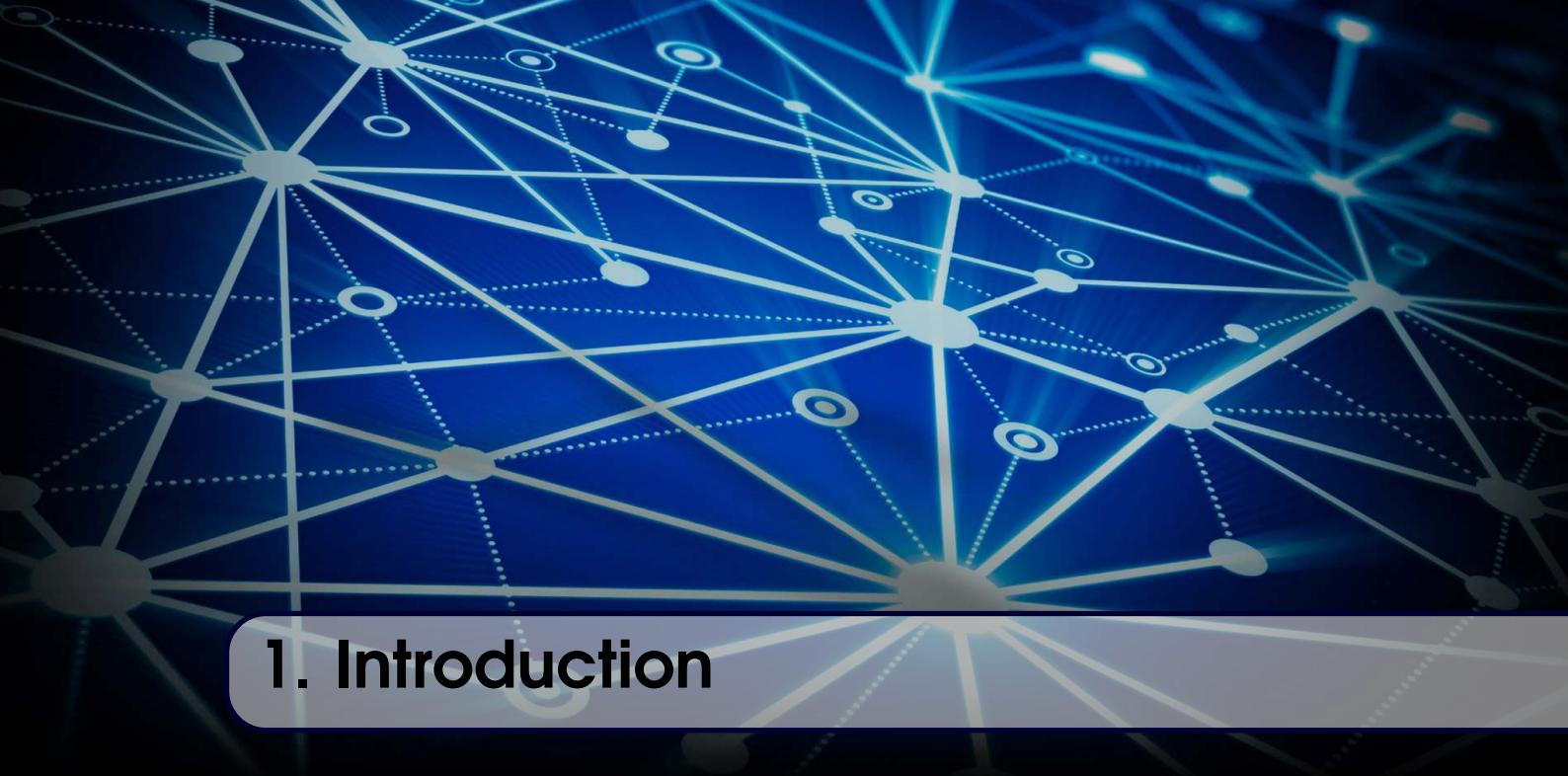
<b>Bibliography</b>	<b>23</b>
---------------------	-----------

DRAFT

# Overview

<b>1</b>	<b>Introduction</b>	7
1.1	Terminology	
1.2	Hosting Paradigm Shift	
1.3	Use Cases	
<b>2</b>	<b>Empowering Technology</b>	9
2.1	Blockchain	
2.2	Tor Hidden Services	
<b>3</b>	<b>Ecosystem</b>	11
3.1	Hosting Provider	
3.2	Complementary Services	

DRAFT



# 1. Introduction

This part gives a motivation and a general overview about the *Distributed Hosting Engine*. It introduces some concepts that will be explained in more detail in the next part. If you are looking for a detailed specification you can directly go to Part II.

The here presented *Distributed Hosting Engine* is designed with the following properties in mind:

**Self-Managed Overlay Network** Hosted pages are distributed between nodes, dependent on different metrics. This increases not only performance, but makes the network also robust against malicious nodes, failures and other expected or unexpected changes.

**Hosting from Everywhere** By design pages can be hosted behind NATs (no public IP) and Firewalls, making the participation of mobile nodes possible.

**Privacy Protecting** The blockchain, nor the protocol, does expose IPs, locations or any other privacy critical data.

**Distributed** There is no central authority, nor should there be any central infrastructure.

## 1.1 Terminology

**Blockchain**

**Block**

**Genesis Block**

**Transaction**

**Protocol**

**Content Creator**

**Co-Host**

**Page**

**Snapshot**

**Node**

## Overlay Network

### 1.2 Hosting Paradigm Shift

This Whitepaper specifies a new protocol and blockchain that will change the hosting of semi-static pages - like blogs, company, private and news pages. Instead of retrieving pages from a specific location, they are hosted directly on end-user devices. The distributed hosting algorithm and protocol, specified here, takes care of the optimal location for each page. By taking into account different metrics, like response times, availability and/or relevance, the perfect place for each page will be found after some time. This not only increases the performance for single page accesses, but reduced also network load and hence increases the overall network health.

Additional to the increased performance, the underlying blockchain assures the correctness and validity of each page, also, or especially, if hosted by a random node in the network. This validation mechanism is then used to create a reputation system, allowing blocking of malicious nodes that have a reputation score that is lower than a threshold value.

This mechanism, together with the page distribution algorithm, makes the network self-managing, self-healing and robust against changes.

By design the *Distributed Hosting Engine* is censorship resistance, but only under the assumption that there are enough nodes that are willing to host the page. This moves responsibility what could be hosted from a central authority to the collective.

From an economical point of view, the here proposed approach, allows the creation of an interoperable hosting ecosystem. Each hosting provider acts as clone and has no direct access to the page, but can support the user with additional (paid or free) services and hosting space. By design a switch from one hosting provider to another one does not affect the hosting of the page, nor the page itself. Technical there is not even a migration happening, only a switch from one complementary service to another one.

### 1.3 Use Cases

#### 1.3.1 Simple Blog Hosting

Jane Doe creates locally a new blog with the help of the jekyll static site generator [3]. The *Distributed Hosting Engine* software takes in the background care of the generation of a new blockchain for this page and the distribution to a bunch of neighbour nodes. After some time, when the page was successfully distributed, Jane switches off her laptop. Even though offline, John can access her blog, because he has previously cloned the page content and has the complete blockchain locally.

#### 1.3.2 Shared Access with Optional Paid Service

John and Jane Doe share access to the same blog. Both have access to the same, related blockchain of this page. John is not familiar with website programming, so he decides to pay for a service in order to have a higher level frontend that allows him to generate pages via a wizard. Jane on the other side is a senior web developer and decides not to pay for any service, but to develop the page by her own without using an external service and hence for free.



## 2. Empowering Technology

This chapter explains the technologies that make the *Distributed Hosting Engine* possible. Most notable the concept of *Blockchains* for the distributed management and self-organization of pages and *Tor Hidden Services* that not only protect the privacy of all involved parties, but that also allows to host pages behind NATs and Firewalls.

### 2.1 Blockchain

*Blockchain* is a rather new concept, with a lot of potential. A *Blockchain* is a distribute ledgers that prevents double spending without a central authority. The first time the *Blockchain* was mentioned and implemented was for Bitcoin [2].

### 2.2 Tor Hidden Services

Tor hidden services are generally used to host pages anonymously and protect visitors and hosters alike. In the scope of this whitepaper tor hidden services are used to enable hosting from everywhere, no matter if the device is behind a NAT or a Firewall. This allows to host pages from public wireless hotspots, from the office or from home. Without the possibility to abstract away from public accessible IPs and the capability to circumvent Firewalls it would not be possible to develop real distributed hosting on end-user devices.

DRAFT



## 3. Ecosystem

### 3.1 Hosting Provider

**TODO:** Explain here the role change of hosting providers. For example: A hosting provider does provide service on top of the blockchain, like page generators and frontends for page publications and updates. If a hosting provider decides to host a page, it is only one clone out of many without special role.

By introducing the *Distributed Hosting Engine* the role of hosting providers will change. Instead of being the only entity that hosts an instance of a page, they will be part of an overall network of hosters. The hosting responsibility is shared between always-on devices, like servers and partially-on devices, like laptops or desktop computers. Additionally a single hosting provider will not be able to control the distribution of the page, because that is determined by the collective of nodes in the network. That said, hosting providers are playing an important role in the overall ecosystem. They not only guarantee uptime of pages by hosting them 24/7, but can also provide additional services that abstract away from the underlying blockchain and page development. Especially the former point is important in the early stage of the network and/or when a new page joins the network. A bigger network with pages that had time to distribute in an optimal way makes dedicated hosting servers unnecessary.

### 3.2 Complementary Services

**TODO:** Figure out what alternatives services could be provided by the ecosystem. This includes also new business ideas for new types of startups.

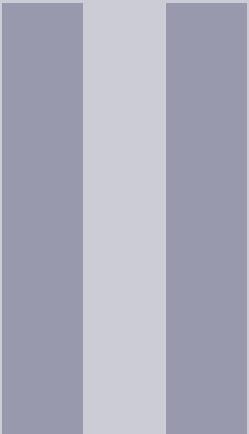
**Hosting Space** As mentioned in the previous section, in the early stage there is a need for 24/7 online hosting servers in order to guarantee uptime. These *Clones* are not

different than any other clone, except that they are always online.

**Page Generation** The countless simple page generation services that exists today can be used to support non-technical users to host their pages.

**Host Wallets** Like Bitcoin Wallets this *Host Wallets* manage all pages related to one users. Not really recommended for security reasons, because private keys are controlled by the *Host Wallet* provider.

DRAFT



# Specifications

<b>4</b>	<b>Blockchain Specification .....</b>	<b>15</b>
4.1	Transaction Definition	
4.2	Block Definition	
4.3	Mining Process	
<b>5</b>	<b>Protocol Specification .....</b>	<b>19</b>
5.1	Blockchain Overlay Network	
5.2	Data Overlay Network	
5.3	Hosting Algorithm	

DRAFT

## 4. Blockchain Specification

### 4.1 Transaction Definition

This section defines transactions and their relationship to each other and to external data. It also defines a (de-)serialization format to and from the binary format that is transferred over the wire. In Figure 4.1 a chain of transactions, with relevant fields, is shown. This example includes three hosts that have cloned the page and in total three updates.

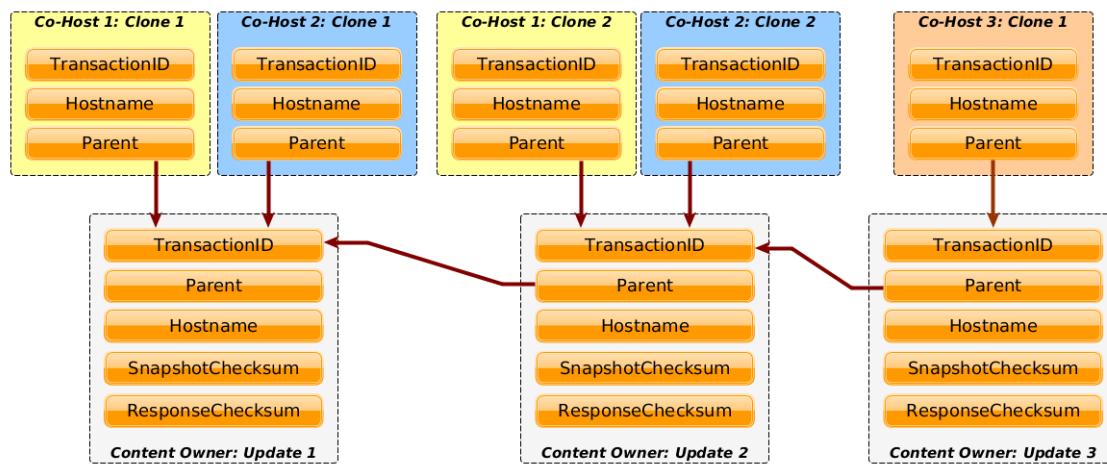


Figure 4.1: Transaction chain example for one page.

#### 4.1.1 Transaction Definition

Update transactions can only be created by the page creator. Additional to the proof of ownership they also include a reference to the new content, with related checksums.

TRANSACTION HEADER	
<b>TransactionID</b>	Double SHA256 hash of this transaction.
<b>Parent</b>	The previous update <i>TransactionID</i>
<b>ScriptSig</b>	<b>TODO Future Work:</b> Proof here ownership of the hostname and all requirements that come with this update.
<b>Hostname</b>	The tor hidden service onion hostname of this page. Tor hidden services are needed to be able to host page behind firewalls and NATs. Additional they protect the identity of the content owner and more importantly of clone hosts.
CONTENT SECTION	
<b>Snapshot</b>	The hash value of the content. This hash value is used to find all peers in the Distributed Hash Table (DHT) that share currently this snapshot. The DHT is bootstrapped by using all active hostname in the blockchain. For increased compatibility the MAGNET URI scheme is used.
<b>SnapshotChecksum</b>	Checksum of the Snapshot content, used to verify the downloaded data.
<b>ResponseChecksum</b>	Checksums for all endpoints of this page. Makes each page verifiable and hence could be served by a random node. If the returned content does not match the here checksum the response is invalid.
<b>Flags</b>	<b>TODO Future Work:</b> Could be used for differentiation between incremental and full updates, but also indicating that this page is not hosted anymore and could be deleted.

#### 4.1.2 CLONE Transaction Definition

**TODO:** This transaction can be merge with the UPDATE transaction. They do not differ too much, except that they have different Flags content and that the CLONE transaction does not include all fields.

Clone transactions are created and propagated by all hosts that start hosting a page or if they receive a new update transaction. This transactions always link back to exactly one update transaction.

<b>TransactionID</b>	Double SHA256 hash of this transaction.
<b>Parent</b>	The previous update <i>TransactionID</i>
<b>ScriptSig</b>	<b>TODO Future Work:</b> Proof here ownership of the hostname and all requirements that come with this update.
<b>Hostname</b>	The tor hidden service onion hostname of this page. Tor hidden services are needed to be able to host page behind firewalls and NATs. Additional they protect the identity of the content owner and more importantly of clone hosts.
<b>Flags</b>	Possible values are: <i>CLONE</i> when new update was received or <i>PURGE</i> if the page is not hosted anymore.

## 4.2 Block Definition

**TODO:** Define here the block as transmitted via the network and how it will be stored (slightly different).

## 4.3 Mining Process

**Note:** *This section is work in progress. The resulting description is a first naive solution. Currently in discussion are multi- entity ownerships and a consensus network between this small group of page owners, e.g. with Practical Byzantine Fault Tolerance (PBFT) [1].*

By definition each page in the hosting space has an own blockchain. In the *Genesis Block* the *Content Creator* proofs ownership of this blockchain and hence the related page. The assumption here is that only the *Content Creator* can update the page and that (s)he is the only entity that has not intention to compromise the own page.

DRAFT

DRAFT



## 5. Protocol Specification

In Figure 5.1 the relationship between the page content (html, css, javascript, images...), in the middle cloud, the data (above) and blockchain overlay network (below) is shown. The network that manages the page content has not to be the same as the blockchain network, but they maybe nearly identical or overlapping. The page content is not part of the blockchain, but referenced from it. This allows to have no page content size limit and the data exchange could be optimized.

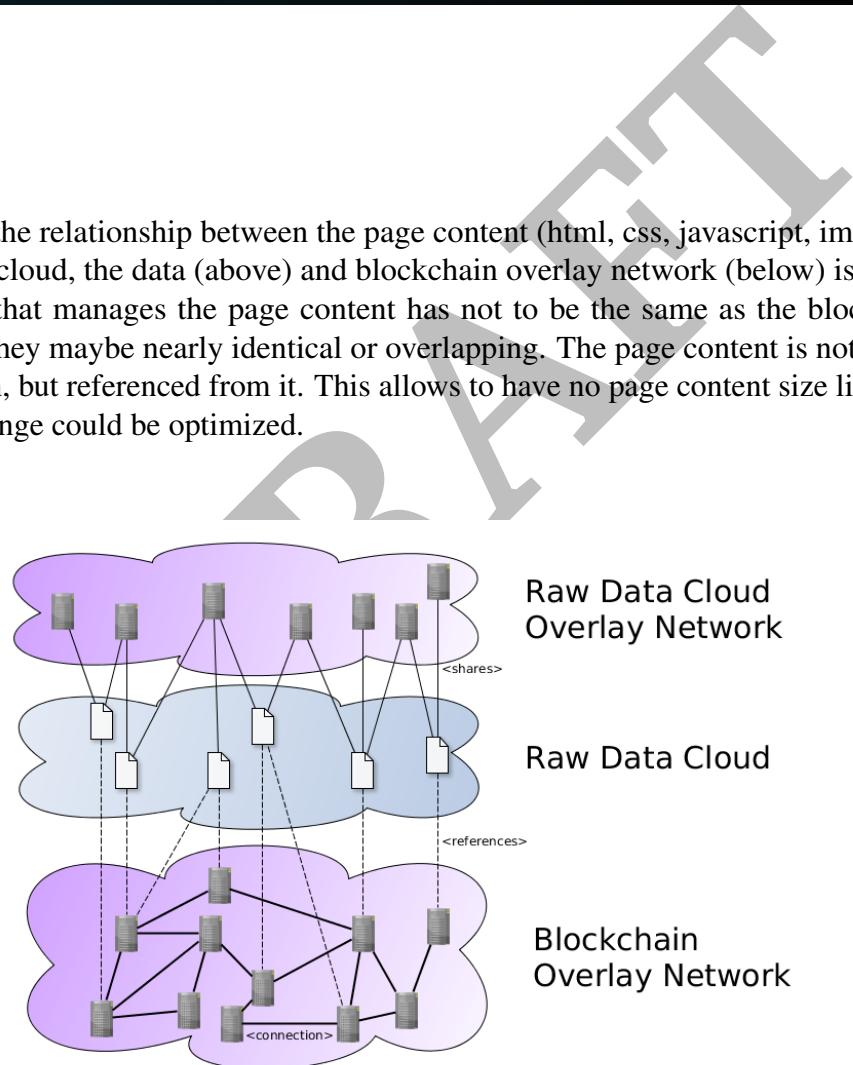
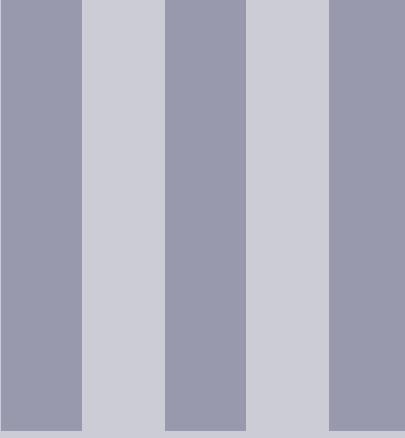


Figure 5.1: Relationship between blockchain and page content from a network perspective.

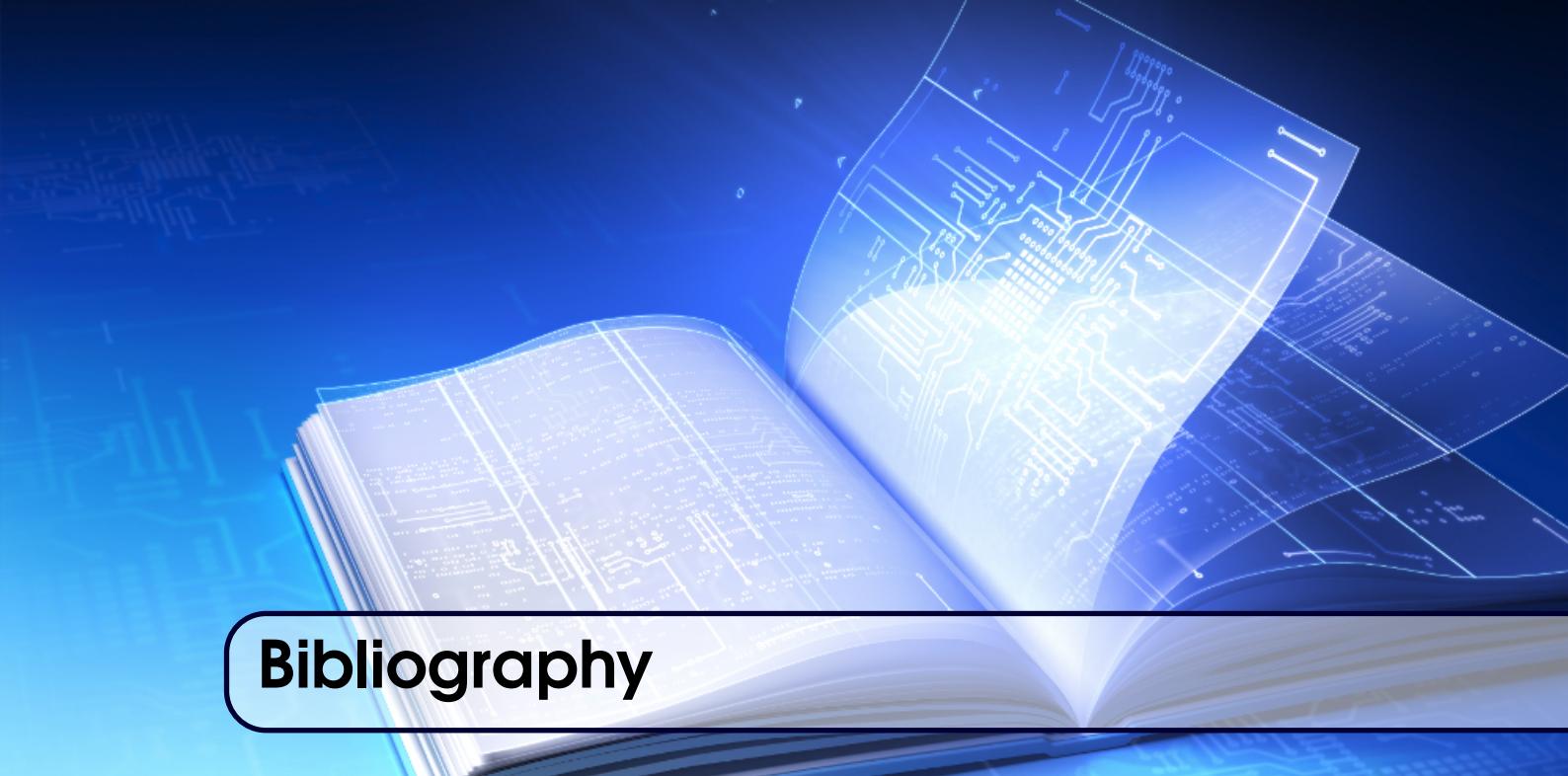
**5.1 Blockchain Overlay Network****5.2 Data Overlay Network****5.3 Hosting Algorithm**

DRAFT



# Appendix

DRAFT



## Bibliography

- [C+99] Miguel Castro, Barbara Liskov, et al. “Practical Byzantine fault tolerance”. In: *OSDI*. Volume 99. 1999, pages 173–186 (cited on page 17).
- [Nak08] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: *Consulted 1* (2008), page 2012 (cited on page 9).
- [Pre15] Tom Preston-Werner. *Jekyll - A simple, blog-aware, static site generator*. 2015. URL: <http://jekyllrb.com/> (cited on page 8).

DRAFT