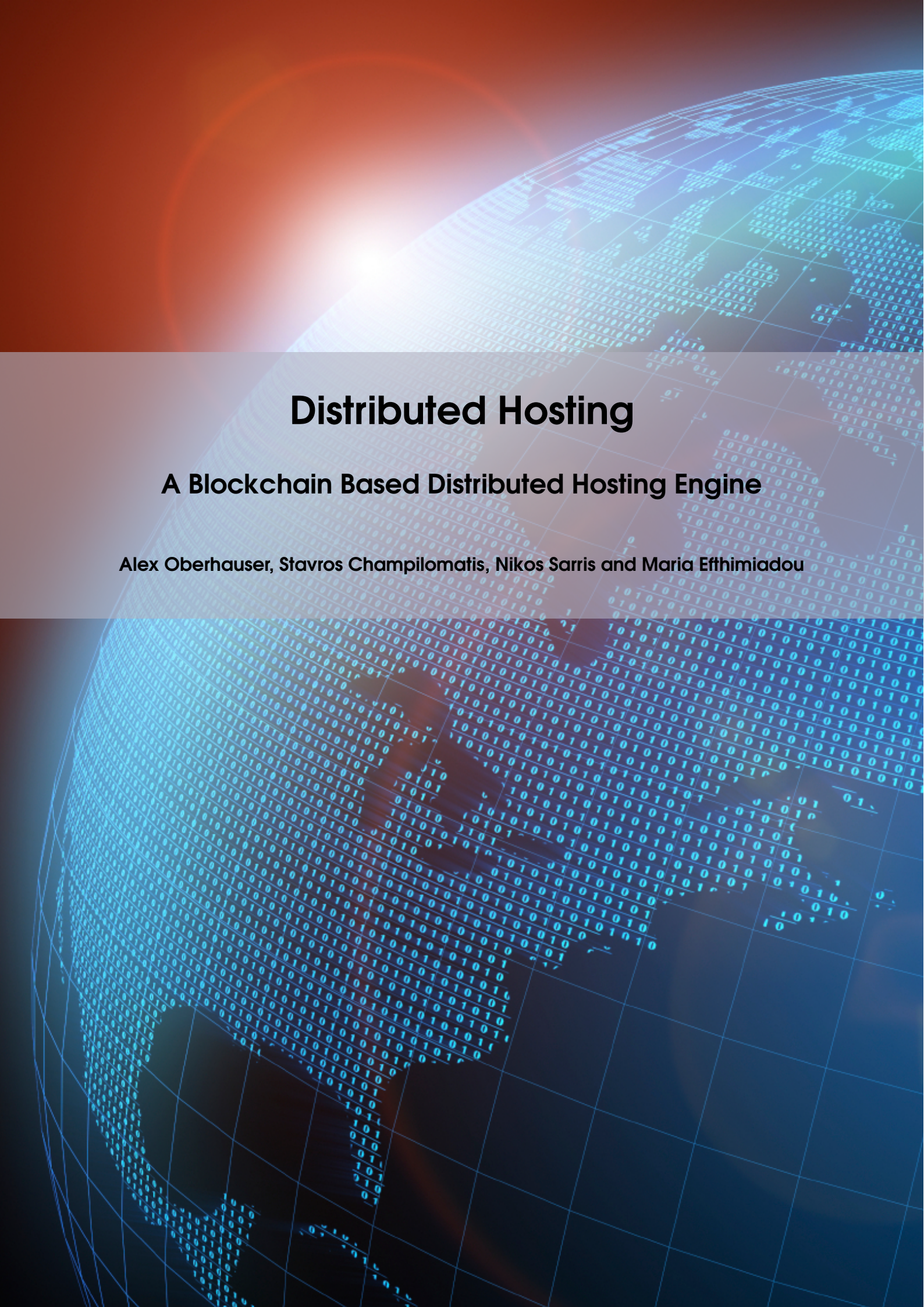


Distributed Hosting

A Blockchain Based Distributed Hosting Engine

Alex Oberhauser, Stavros Champilomatis, Nikos Sarris and Maria Efthimiadou

PUBLISHED BY THE *BlackGate* TEAM

HTTP://BLACKGATE.NETWORLD.TO

First released, March 2015

Contents

Overview

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 [**Preston-Werner2015**]. 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 [**nakamoto2008bitcoin**].

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.

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. Additional 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.

II Specifications

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.

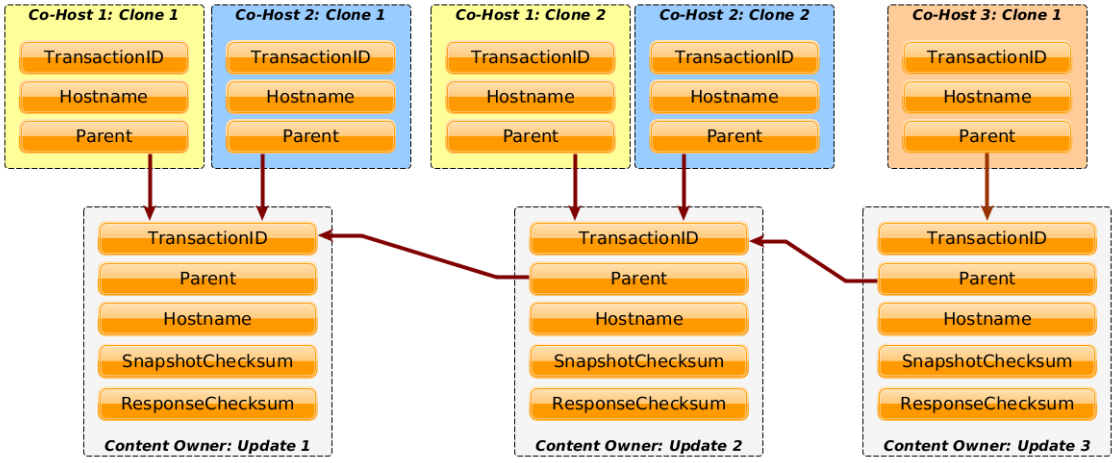


Figure 4.1: Transaction chain example for one page.

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	
TransactionID	Double SHA256 hash of this transaction.
Parent	The previous update *TransactionID*
ScriptSig	**TODO Future Work:** Proof here ownernship of the hostname and all requirements that come with this update.
Hostname	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	
Snapshot	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.
SnapshotChecksum	Checksum of the Snapshot content, used to verify the downloaded data.
ResponseChecksum	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.
Flags	**TODO Future Work:** 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.

TransactionID	Double SHA256 hash of this transaction.
Parent	The previous update *TransactionID*
ScriptSig	**TODO Future Work:** Proof here ownernship of the hostname and all requirements that come with this update.
Hostname	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.
Flags	Possible values are: *CLONE* when new update was received or *PURGE* 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)* [**castro1999practical**].

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.

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.

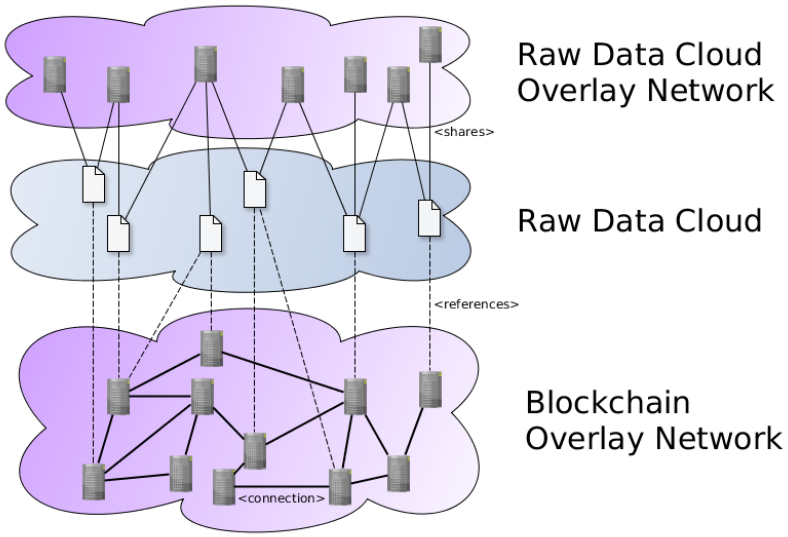


Figure 5.1: Relationship between blockchain and page content from a network perspective.

5.1 Blockchain Overlay Network

The blockchain overlay network consists of nodes that are responsible for blockchain exchanges. Each nodes keeps track of a set of blockchains, dependent on the surfing behaviour of the user. Each blockchain includes a list of known nodes that are used to keep track of updates insides this blockchain. As a matter of fact it is valid to say that the overall network consists of multiple overlay networks, one for each blockchain or page.

After extracting the original host and all co-hosts from a blockchain, the node keeps track about the performance of each found node and prioritises them. For the prioritisation quality of service properties, like availability or correctness of received information, are used. The highest ranking nodes are used to keep track about updates and propagate changes.

If a blockchain is known, new nodes can be found by extracting them from the propagated *CLONE* transactions. If the blockchain is not yet known, the node accesses the page for the first time, there are not fallback nodes if the original host is offline. For that specific node the page looks like if it is offline, also if there enough co-host to satisfy the request.

5.1.1 (REJECTED) Solution 1: Global Blockchain

One solution to this problem is to introduce a global blockchain that keeps track of all page blockchains. By using the sidechain approach [**back2014enabling**], the page blockchains could be linked against the global blockchain. Each time a new co-hosts starts hosting a page and propagates a CLONE transaction, in one of the sidechains, the global blockchain keeps track of this change and adds this host to a list of co- hosts related to the sidechain. Without going into detail how this could be specified and implemented, there are some problems with this approach.

Dynamic Nature

In difference to the Bitcoin and other blockchains our approach relies on an external state that can not be usefully modelled in transactions, through the high frequency of changes, and that is the availability of co-hosts. Co-hosts are only relevant if they are online at a given point in time. Keeping track of availability of nodes in a global blockchain is not only counter productive, but also infeasible. A solution would be to keep track of all co-hosts and let figure out the nodes what to node to contact. This makes it pretty inefficient if the sidechain grows.

Scalability issue of block generation

The Bitcoin mining approach to solve the scalability problem by hashing power and changing complexity based on the networks' computation power is not suitable in this case. The here presented blockchain has no underlying asset that has to be mined. This makes the proof of work a waste of energy and time. For the same reason proof of stake is also not suitable. The mentioned approaches in literature are suitable from a theoretical point of view, but are not able to scale, e.g. **castro1999practical** [**castro1999practical**].

5.1.2 (ACCEPTED) Solution 2: Distributed Hash Table lookup

An alternative solution is to avoid a global blockchain altogether and instead keep track of co-hosts, related to the original page host, inside a *Distributed Hash Table (DHT)*. For page

content, aka. snapshots, distributions a DHT is used to lookup which node has currently the needed snapshot. After that the BitTorrent protocol is used to download and then distributed the content further. With minimal or no modifications at all the same approach can be used to lookup co-hosts, that are related to an original page host.

Every node in all blockchains is also part of this DHT and can be hence used as entry point for lookups for all other blockchains. If a host does not have any blockchains at all, publicly known hosts can be used as entrypoints. These entrypoints should be maintained by the community.

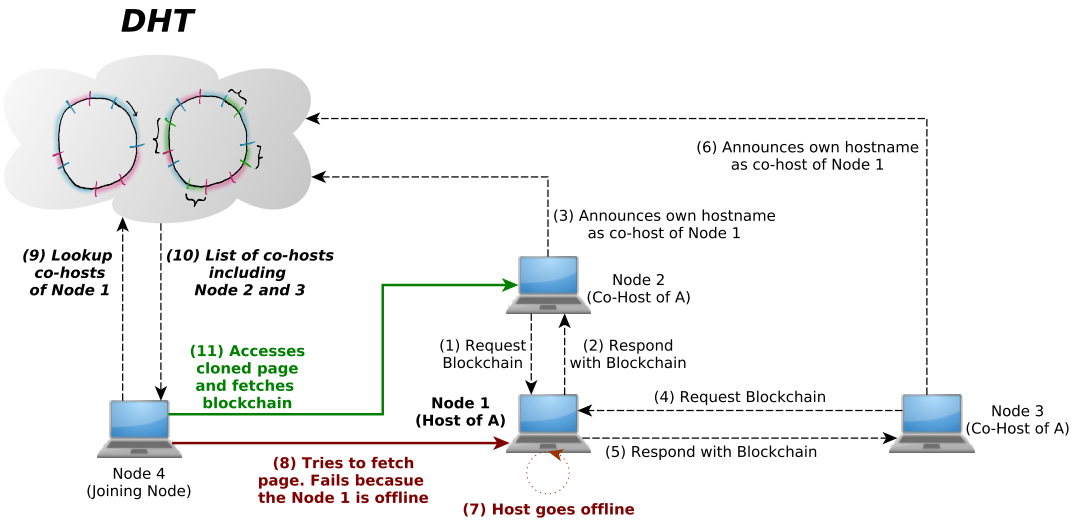


Figure 5.2: Proposed solution, based on a Distributed Hash Table (DHT) for the first lookup problem if the original host is offline.

Figure 5.2 explains that approach. *Node 1* is the content creator and hosts a page. *Node 2* and *3* access the page, fetch the blockchain and start hosting. After propagating a *CLONE* transaction to the network, they also announce themselves to the DHT. Each time they continue hosting, e.g. coming back online, an announce messages is send. *Node 4* wants to access the page of *Node 1*, but this node is currently offline. By looking up all, currently online co-hosts *Node 4* can choose in this example between *Node 2* or *3*, it chooses *Node 2*.

5.2 Data Overlay Network

5.3 Hosting Algorithm

5.4 Node Handling

All nodes assign priorities to its neighbor nodes in the overlay network. The priority, which is a subjective metric, represents ratio of the response messages, Transaction or a Block, requested from a specific host to the overall responses (from all the hosts).

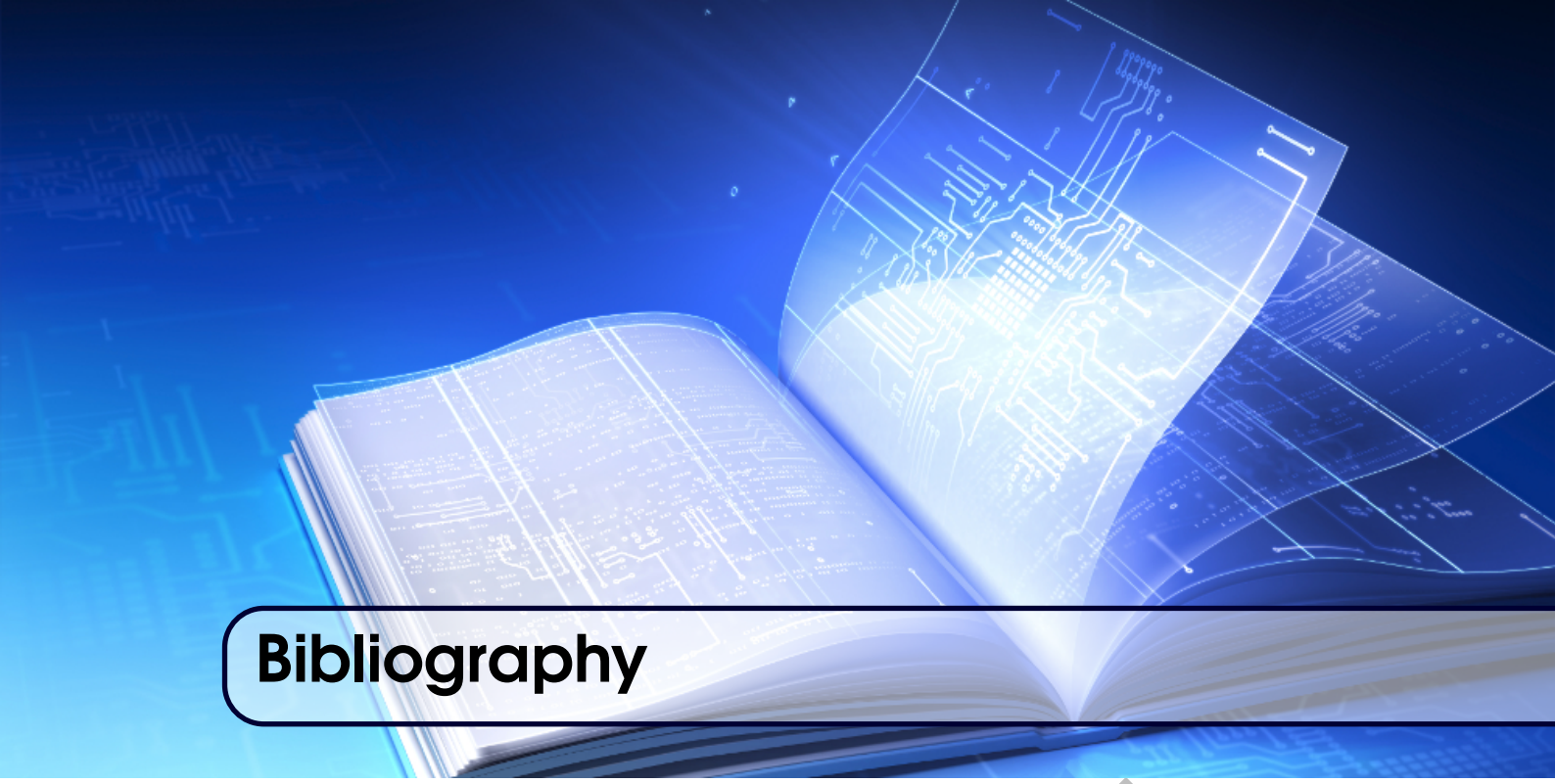
$$score = \frac{receivedTransaction + receivedBlock}{allResponses}$$

The host, which receives the response message, increments always the allResponses value but only the receivedTransaction or receivedBlock one for the host whose message

arrived first.

III

Appendix



Bibliography