Nexusless: Building a Peer-to-Peer Content Delivery Network

Nexusless Research Team
Revision 2, February 2018

Abstract

Building on the recent developments in the fields of data structures, networking, and cryptography, this paper explores the concept of a distributed, peer-to-peer content delivery network (CDN) that enables fast, secure, and efficient serving of websites, applications, streaming content and other data within the existing web infrastructure. Instead of serving content from specialized machines located in remote server farms, Nexusless envisages storing and delivering web content from participating consumer devices that act like miniature points or presence, typically in close physical proximity to any given end user. Apart from providing any laptop or smartphone owner with the possibility to monetize the unused computing capacity of their device, such design potentially allows reducing network load and latency, as well as ensuring maximum uptime.

Nexusless is based on a distributed filesystem with content-addressing, inspired by the IPFS (in turn derived from the sKademlia DHT and the Git versioning system). Containers for mutable states and permanent names are utilized as well, in order to provide stable addresses for evolving data and allow interfacing with the existing web infrastructure, foremost the DNS.

A blockchain-based incentive layer is added on top of the storage-delivery framework, allowing any participant to earn income for acting as points of presence for the network. The blockchain makes it possible to handle transactions between content originators and the edge nodes without requiring trusted intermediaries. A probabilistic consensus mechanism is introduced based on proofs of retrievability and proofs of delivery, to make sure that new blocks are added by network participants which perform the largest amount of *useful* work. The demand for high volumes of micro-transactions is solved by adding a mesh of payment channels, similarly to the implementation of the Lightning Network.

The paper also discusses management protocols and other measures needed to ensure practicability as well as key performance metrics, including low latency (optimizing network paths), high reliability (eliminating single points of failure), and data security (encryption and content-addressing).

Note: the current paper should be regarded as work in progress; the authors will be issuing updated versions as research and testing continues.

Contents

1. Introduction

A content delivery network (CDN) is a geographically spread group of data centers set up to achieve speed optimizations and high availability of web services (websites, applications, media, etc.) to end users across a wide region of the world or globally [1]. A large share of modern web traffic is served by CDNs [2], either specialized providers such as Akamai [3] and Stackpath [4] or systems built in-house by large corporations, such as Google's Edge Network [5]. To put things into perspective, Akamai alone claims to serve as much as 30% of the world's internet traffic on a regular basis [6].

The market for content delivery services is expected to grow at an average annual rate of over 32% in the coming years, reaching $30 billion by 2022 [7], fueled by both growing internet usage across the globe and the steady increase of the quality (and therefore file size) of transferred media – from images to videos to, eventually, VR streams. This paper presents a conceptually new type of CDN based on a highly distributed global peer-to-peer (p2p) network, based on the latest developments in software and hardware and as a response to the challenges facing the CDN sector and the modern web as a whole.

1.1. Motivation

While considerable demand growth for content delivery services is expected in the upcoming years, there is a number of issues that, collectively, leave much room for improvement and ask for conceptually new implementations:

- High prices: although there are emerging offers geared towards small-budget projects and individual webmasters [8], most CDN services are both too expensive for emerging websites and require four-five-digit monthly expenditures from established ones. The prices are set arbitrarily by the service providers and are only indirectly market-based.

- Need for (real) speed: the exact content delivery speeds may diverge significantly from the advertised figures, with no real accountability on behalf of the CDN providers.

- Bandwidth and scalability: even the most efficient data centers need to connect to the larger web at some point; these junctions can potentially become choke points as content demand and hardware performance grow faster than the network throughput. This also leads to increasing vulnerability to physical cataclysms, infrastructure failures [9] and targeted attacks [10].

- Concentration of power: as CDN providers grow in size, they can afford to deploy more and more points or presence, thus gaining a competitive edge to repeat this cycle. This means that the content traffic of the World Wide Web might be increasingly handled by a shrinking group of conglomerates; the least detrimental way such situation could stabilize is into an oligopoly, but consumer bargaining power would be greatly reduced in any case.

- Resource consumption: while its rate of growth has fallen in the past years owing to data center infrastructure and server efficiency improvements, continuing increases in electricity demands from the world's server farms are still expected.

- Limits to locality: despite the growth in the number of points of presence available for the largest CDN market players, there are economical limitations to how local the data centers can become. In addition to that, the emerging and frontier markets naturally get less attention and investment, which is exacerbated by high latency in existing local networks.

One way to meet the above challenges is to build more of the same – setting up additional large, power-hungry data centers connected to the Web by ever-higher bandwidth; another potential approach is to make better use of the hardware that is already all around us – the 2 billion (and counting) consumer devices such as smartphones, tablets, laptops and desktops everyone has on their desks and carries around in their pockets. Despite the continuing trend of more time spent on various devices, people still use their smartphones just 20% of the day [11] – the remaining time is idle, unused processing cycles.

Such CDN will be decentralized by design, driven by peers instead of asymmetrically powerful specialized agents. An incentive layer would contribute further to a greater equality of the web participants, as any device owner is able to connect to the network and become a point of presence. The next section provides a brief overview of the main concepts and existing research on peer-to-peer content delivery networks.

1.2. Overview of Concepts and Literature

It can be argued that we currently find ourselves at a critical point in time when the evolution of peer-to-peer file sharing protocols as well as cryptographic distributed trust-less ledgers such as blockchain converges towards a possible common denominator; coupled with the rapid recent growth in performance of personal computing devices, these trends present the potential for forming principally new models of how the Web might function in the near future.

Below we are going to look at the key concepts related to the idea of a distributed peer-to-peer content delivery network; the rest of the paper will build on these concepts to arrive at a design for a competitive and economically viable new-generation CDN.

One of the most successful attempts to date at moving large volumes of content online using peer-to-peer networks is the BitTorrent protocol [12]: used by millions of people on a regular basis and estimated to stand behind more than 40% of the entire web traffic. The protocol is based on the use of distributed hash tables (DHTs) such as Kademlia [13], [14] in order to coordinate extensive metadata in trustless networks of peers (swarms) and exhibits several important features that make it so useful and resilient, namely block-wise transfer of files, content addressing, load balancing, and incentives for sharing rather than leeching.

The torrent network exists as a separate entity on par with the World Wide Web, and so requires specialized software for accessing. IPFS (short for InterPlanetary FileSystem) builds on the ideas of peer-to-peer file sharing protocols, crossing them with version control systems such as Git and self-certified filesystems to arrive at a concept of a global, versioned, fully distributed content-addressed block storage model with no need for trust and no single point of failure [15].

Filecoin adds a protocol layer on top of the IPFS to create incentivized markets for distributed storage and retrieval of any web-based content using a blockchain with proof-of-space (POS) derived proofing [16]. The work discusses practical issues of fault tolerance and protocols aimed at maximizing the robustness of such network. Another notable example of a peer-to-peer web content delivery concept is the Orchid project [17]; the authors propose "bandwidth mining" as a mechanism to incentivize participants in a free for all distributed layer on top of the existing Internet infrastructure. The primary goals of the Orchid network are bypassing any surveillance and censorship, two phenomena that affect the majority of the modern Web population.

We will address the above ideas throughout the current work and discuss how they can be evolved further to adapt for a particular practical application within the existing Internet framework.

1.3. Roadmap

The rest of the paper is organized as follows: Part 2 explains the functioning of the modern CDNs and presents a design for the Nexusless distributed CDN. Part 3 introduces the incentive layer in the form of storage and retrieval markets, while Part 4 discusses the most important properties such network should possess in order to be useful. Finally, Part 5 proceeds with further discussion and possible directions for future research.

2. Nexusless Content Delivery Network Design

This section presents the necessary components of a decentralized CDN as well as the key principles that guide their interaction to form a usable, useful web service. Before discussing the architecture of the network, however, it is important outline the important features of a well-functioning CDN.

2.1. Necessary Properties

Content delivery networks exist as a seamless layer (from the end user's standpoint) to serve websites and applications as efficiently as possible. In order to meet the needs of both the clients and the users, a CDN must therefore possess the following crucial properties [1]: performance, reliability, scalability, and responsiveness.

Performance can be expressed in terms of delivery speed from the standpoint of the end users. When it comes to rendering content, a useful indicator of performance is the round-trip time, expressed as the number of milliseconds it takes for the viewing browser to send a request and receive a response from the server. This is an important metric because no rendering can begin before receiving the response, and users will tend to abandon a page or service if it doesn't load after several seconds [18] – negatively impacting engagement, conversions, and ultimately the bottom line. There are multiple ways to minimize latency, including reducing the physical distance a signal has to travel between the user and the server (as well as the number of intermediate nodes it has to traverse), optimizing overall traffic and choosing the most efficient transmission mediums.

Reliability means being able to access the CDN-backed website or service at all times. The most important measure to maximize overall uptime is increased redundancy at all levels: duplicating content across data centers, adding hardware and software redundancies, setting up automatic systems to route traffic around downed servers, as well as simultaneously working with multiple network carriers. Lowering the probability of downtime thus implies adding more redundancy, or eliminating single points of failure, which, due to the nature of how CDNs are constructed, often implies adding more points of presence, thus increasing the overall costs of maintaining the network; such extra costs are then typically transferred to consumers.

Scalability implies the ability of a content delivery network to handle arbitrary amounts of traffic at any given moment. Internet traffic is typically not homogenous with respect to the time of day and other factors like external events. This means that points of presence should deploy significant extra processing and networking capacity to cover the entire demand and ensure glitch-free service even at moments of sharp

traffic spikes (being not able to withstand surges in traffic renders a CDN half-useless). In other words, high scalability requires every data center to maintain large unused capacity at most times. Another important related consideration is resilience to capacity-based attacks [19] such as denial-of-service (DoS and especially DDoS) which can be thought of as intentional equivalents of unexpected traffic spikes.

Responsiveness measures how quickly content and configuration changes propagate throughout the entire CDN. Maintaining high responsiveness tends to get harder as the network becomes more complex and physically dispersed. Failure to ensure sufficiently fast propagation can result e.g. in end users in different locations getting different versions of content, which may have negative impact on the client's overall reputation or even be a critical factor in their operations.

As networks scale, the inter-relation between these qualities becomes more apparent; this tradeoff problem has been formulated in the so-called Brewer's CAP theorem [20], stating that it is not possible to simultaneously ensure perfect *consistency* (C) and *availability* (A) in the event of a *partition* (P) failure. *Consistency* in this context can be mapped to *responsiveness* in the above points, while *availability* can be thought of as encompassing both *performance* and *reliability*. In other words, as CDNs grow in size and complexity they can experience increasing pressure to relax at least one of the above stated requirements, potentially impacting end user satisfaction with the service.

2.2. Traditional CDN Architecture

Although the precise details and solutions may vary, most existing content delivery networks can be generalized into the following scheme:

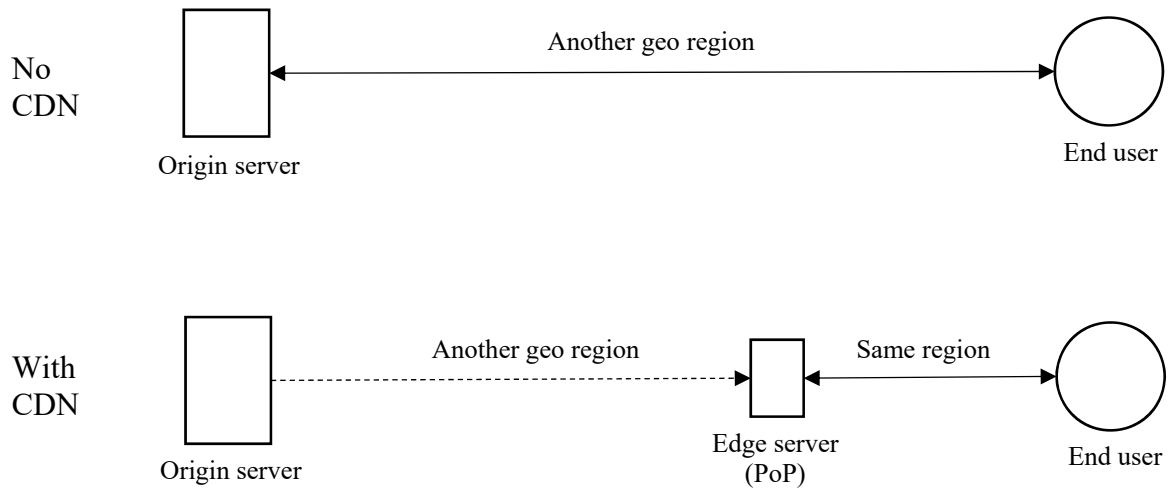

Figure 1: Content delivery network - basic principle

In the case of no CDN the end user's browser sends a request and receives the requested files directly to/from the server which hosts the original website (which might be located across the globe from the user). What a content delivery network does is deploy multiple datacenters across the globe and keep the cached versions of the page on edge servers in these points of presence (PoP); the end user can thus get the requested content from a more adjacent (and optimized) machine instead of communicating directly with the origin server (Figure 1). The dashed line between the origin and the edge displays the route of the cached pages while the solid lines indicate traffic to the final user.

This setup leads to several benefits for both the page owner (the client) and the user, including higher page/content load speed (*performance* from section 2.1), higher availability due to data redundancy (*reliability* from section 2.1), decreased load on the origin server, and potentially enhanced security since the edge network can act as a reverse proxy for the origin, implementing various measures to mitigate attack risks.

However, such scheme also faces several challenges, including costly scaling as additional servers and entire data centers need to be deployed for network growth as well as other issues presented in section 1.1. In order to mitigate these and simultaneously provide a more egalitarian environment for the further growth of the Web, a principally new design can be constructed using peers themselves as miniature points of presence.

2.3. Nexusless Distributed CDN Layout

Holding other parameters constant, moving closer to the nearest PoP should result in higher speed as signals need to travel less to reach their destination; along the same line of reasoning, having more PoPs should generally lead to increased availability as network redundancy increases as well as contribute to more efficient bandwidth utilization as traffic is diverted from the junctions which connect the PoP with the web. Adding more nodes and maximizing proximity in the traditional design discussed in section 2.2 always requires additional investments and expenditures in an almost linear fashion (excluding network-wide overheads and modest economies of scale).

The proposed content delivery network architecture (hereafter also: Nexusless) extrapolates the proximity and availability arguments above without incurring proportional cost increases by merging, to a degree, the edge servers with the end users:

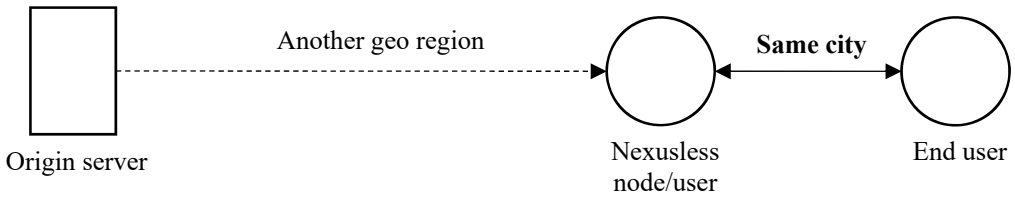

Figure 2: Nexusless peer-to-peer CDN principle

The concept of Nexusless implies that any user can also become a server by installing a dedicated piece of software that enables it to store cached pages from the origin server (or other network nodes) and deliver them to end users, who themselves may or may not be Nexusless nodes serving content to yet other users. Edge nodes compete for content delivery, aligning the motivations of the network participants with the expectations of both clients and end users.

Any modern consumer-grade device (desktop computers, laptops, tablets, mobile phones) can become a node of the network and pledge some of its resources to serving content. Because of that Nexusless CDN can be comprised of *millions* instead of hundreds of PoP-like entities. As more and more user-nodes are added to the network, a decentralized, peer-to-peer "swarm" emerges, possessing several interesting useful properties:

- *Extreme proximity to end users*: the number of nodes will tend to naturally be higher in places with higher populations of device users, meaning that regions with more demand for web content will also tend to get more capacity.

- *Further load reductions* from potential network choke points as most of the interchange will be done between the peers themselves.

- *Higher service availability* through balanced redundancies across arbitrarily many local devices. While the robustness of a single node is considerably lower than that of a traditional CDN's specialized edge server, the dramatic increase in the number of participating nodes has the potential to more than offset lower individual uptime.

- *Lower storage/retrieval costs* as the network makes use of existing devices instead of installing new hardware. Scaling such a network implies significantly lower expenditures because no new datacenters need to be added.

- *Incentivization potential* through the introduction of a decentralized, blockchain-based marketplace for content storage and retrieval, allowing network users to earn income by becoming edge nodes.

Nexusless makes use of specialized protocols to ensure useful and economically viable service for both the clients (website/service operators), participating users (edge nodes) and end users (who might or might not be part of the network). The protocols will be discussed in the next section of the paper.

2.4. Content Addressing

As anyone with a moderately powerful consumer-grade computing device can behave as a node in the Nexusless network and store/retrieve other people's files, the challenge of ensuring data integrity becomes paramount to the functioning of the entire system. In order to simplify content verification, prevent disappearance of important files, as well as utilize computing and network resources more efficiently, it is proposed to use content-addressed block storage and exchange model based on the principles implemented in the IPFS [15].

Such framework builds on the principles tested in DHT-based block exchange protocols (such as Bittorrent), versioning systems (Git), as well as self-certified filesystems or SCF [21]. Files are stored as blocks on peer machines, with each block being addressed using its hash (a fixed-length string of seemingly random digits created by using a cryptographic hash function such as SHA-2); this implies that each file receives a unique "name" by which it can be found at any time, and since cryptographic hash functions are designed to produce a non-correlated outputs for any two distinct inputs, any changes in the content of the file will lead to it producing a significantly different hash, therefore preventing any tampering.

When a new version of a file is created, it incorporates the name (hash) of the previous version; the same principle applies to linking to other objects within the file-space. Such chaining of content leads to the emergence of a network-wide directed acyclic graph[1] (DAG) similar to those used in the Git versioning system. Each network node that stores content is assigned a unique key-based ID as proposed in the S/Kademlia modification [22], making the construct more resistant to various attack types (see section 4.2 for further discussion on the topic of security). Nodes can communicate with other peers using a DHT and exchange content blocks based on a system that rewards replication and distribution.

[1] A Merkle DAG is a mathematical construct which can be defined as a collection of vertices connected by unidirectional edges, in the absence of directed cycles. A special case of the DAG, called a Merkle tree, is widely used for data verification purposes in the most popular cryptographic currencies, including Bitcoin and Ethereum [51].

This hybrid system solves the content genuineness issue by making sure each name-address can only point to a single, unique set of bits. Since content becomes immutable, updating in-place becomes possible through the use of SCF-based self-certified names that allow having mutable states (versions) of a file available from the same network path. In other words, names act as "containers" for the subtrees that represent the version evolution of individual files – for example, updating a website's `index.html` homepage file can be viewed as substituting one immutable version of the file with another within that same "name-container".

Apart from providing authenticity, the above described system also allows avoiding dead links (as content addresses become permanent) and saving on network bandwidth and computing resources through data deduplication.

The following section explores a protocol layer which builds upon the described filesystem and namespace to provide content delivery service that meets the needs of all involved parties.

2.5. Network Protocols

The Nexusless content delivery network emerges from the interactions of several types of agents: the clients (can be identified with origin servers) who wish to distribute their content and services via the CDN, the edge nodes which store and deliver the content, as well as the end users who wish to consume content and services; the latter might not (and *need* not) even be aware of the exact way those are delivered as long as it is done in a timely, consistent and efficient manner (see section 2.1 for a discussion of the necessary properties of a CDN).

Clients let the network store multiple cached copies of their data on the edge nodes, which is then retrieved on demand by the end users. This means that, as opposed to e.g. the Filecoin design [16] where clients are the party which both stores *and* retrieves data from the network, in this case the client and user roles are distinctly separate as each group pursues their own specific goals.

Disregarding the incentive layer at this stage, we can draft Nexusless protocols within the storage and delivery cycles from the perspectives of each connected party (Table 1).

Table 1: Nexusless network protocol draft

Client (~ Origin Server)

- at any time:
  - place *store* bid orders on the market
- upon transaction request from edge node E for file f:
  - send store request to E
  - sign and send f to E
- at each interval t:
  - receive proofs of retrievability (POR) for already stored data
  - if some POR are missing or incomplete, place new *store* orders

Edge Node

- at each interval t:
  - refresh storage pledges by removing any occupied/expired sectors and adding any new ones
  - generate recurring proofs of retrievability for already stored data
  - on the storage market, find *store* bid orders matching the pledged storage space and device's physical location
  - initiate transactions with matching clients
- upon *store* request from client C for file f:
  - check that f is properly signed and is of the size specified in the bid order
  - download f into storage
  - generate proof of storage for f
- at any time:
  - listen to *deliver* requests from the network
- upon *deliver* request for piece f from the network:
  - retrieve f from storage and send it to end user U
  - generate and broadcast proof of delivery

Network (also Market)

- at any time:
  - aggregate incoming *store* bids from clients
  - listen to incoming *deliver* requests from end users
- upon *deliver* request for piece f from the end users:
  - broadcast request to edge nodes
  - receive proofs of delivery from edge nodes
- at each interval t:
  - receive POR from edge nodes
- at each interval t for each new block b:
  - make sure all *store* and *deliver* transactions are valid
  - make sure all proofs are valid
  - add valid transactions to b
  - finalize and broadcast b

It is worth noting that some of the actions above are carried out at equal time intervals, while others happen asynchronously at any possible point in time. In an ideal setting, the entire protocol could be made asynchronous thus making the network more responsive, yet since minimum latency is only strictly required in the delivery cycle (holding other things constant, end users want web services to work as quickly as possible), the storage cycle can function on a timed interval basis without losing most of the utility to the client side. The length t of the synchronization intervals can be tuned to balance the network load with the up-to-dateness of content and response speed of the storage operations. However, if it is feasible to make the value of t sufficiently small, the delivery network (or at least parts of it that do not require speed maximization, like gradual consumption of large-volume, streamed content) can also run on timed cycles.

Two strategies can be used to ensure that each storage and delivery transaction is valid and useful: either having a trusted third party for verification and allocation or using a trustless distributed ledger like blockchain. The latter has an obvious advantage of not requiring the existence of a centralized prover/marketplace and thus increasing transparency and reducing costs, yet in the case of time-critical applications like CDN will need additional adjustments to be able to function without significant fault probabilities (this will be discussed further in section 4). The next section will present an outline of a decentralized market for content storage and delivery that functions on top of the Nexusless network.

3. Content Storage and Delivery Markets

In order to motivate active participation (as an edge node) in the Nexusless network, an incentive scheme can be constructed whereby peers who provide storage and delivery of clients' content get paid for lending their spare processing power, memory, and bandwidth. Since both supply (number of online edge nodes) as well as demand (end user traffic) tend to fluctuate significantly with time, a market-based system is preferable to fixed pricing. Instead of organizing such marketplace around a central authority which validates every transaction, this paper proposes a decentralized layer using a near-permissionless (public) blockchain.

The main advantage of using a distributed ledger for the storage/delivery markets is its fit with the concept of a server-less, peer-to-peer CDN: it avoids introducing a centralized authority and trust asymmetry into a system that is designed to be open and trust-less from the onset. If a blockchain is to be used for storing transactions in the Nexusless network, a suitable proofing method and consensus protocol are required for adding new blocks to the public ledger.

3.1. Proofing Methods

To achieve consensus on a permissionless blockchain there needs to be a mechanism for choosing which nodes get to write new blocks in each time period. Ideally, such proofing mechanism should not only be relevant, but also *useful*, i.e. non-wasteful in terms of resource consumption to ensure network scalability.

Since the operation of the peer-to-peer CDN under discussion consists of two distinct cycles – data storage and data delivery – a way of making sure each action is performed by the edge nodes is required. In other words, a given edge node E_i should be able to prove (and the client C or another node $E_j$to verify) that E_i is storing a given file f as well as to prove that E_i has delivered f to an end user at a given time.

In can be argued that since storing the content is not the end goal of a CDN, a reasonable proofing scheme should mostly be concerned with the proofs of actually delivering the content, as many more acts of delivery are to be expected than acts of storage (i.e. a single file could be stored as redundant copies on hundreds of devices but served millions of times to end users if a particular web page or piece of content is popular enough). On the other hand, storage cannot be made completely proof-less (and therefore free) because not all files can be expected to be small, e.g. images, videos, and other bulky content.

To be able to compensate the edge nodes for committing their storage capacity, a proof-of-storage (POS) scheme is thus needed in Nexusless. Proofs of retrievability (POR) proposed by Shacham and Waters [23] fit well with the requirements of the storage cycle, the main advantages being a guaranteed ability to prove that E_i is indeed storing the entire f, as well as small challenge query-response message sizes (measured in double-digit bytes).

Another useful property of POR, as compared with the earlier POS such as [24], is that after running a certain number of challenges the verifier is able to reconstruct the original file even in case the node refuses to hand it over (hence the name). Since each E_i possesses a unique ID on the network (see section 2.4), Sybil attacks[2] can be avoided by tying proofs to node IDs; a possibility of "outsourcing" and colluded deduplication[3] might also require introducing a time bounds into the proofing process as detailed in [25] in order to distinguish between genuinely stored files and files that were retrieved elsewhere "on demand" at the moment of running the challenge.

As noted above, a well-organized CDN is expected to produce more instances of data delivery than data storage (otherwise some stored data will not be delivered at all), which means that a proofing mechanism for end-user delivery is paramount to ensuring that an incentive layer is operating efficiently. The additional challenge in this case is that the end users should not be forced to handle any specialized software for accessing content from the CDN, apart from a standard web browser; moreover, the delivery is paid for by the content owners C instead of the end users (in contrast with the more symmetric requirements of purely storage-oriented systems like the Filecoin [16] or Storj [26]).

The latter implies that the edge nodes need to prove content delivery to clients instead of the end users who actually receive the content. A client therefore cannot ask an end user directly for an evidence of the file having actually been downloaded to their machine (i.e. successfully delivered, such as via the POR described above); instead, such evidence must be constructed by the edge nodes.

One solution is to require clients to insert tracking code snippets into their websites, e.g. JavaScript similar to that used by real user monitoring (RUM) services like Pingdom [27]. Such code can count each instance of the page in question being accessed by an end user, yet would introduce additional effort on the part of the client as well as add to the already dense tangle of monitoring scripts populating most web pages today [28]. There will also arise a question of the recipient of such data – in the absence of a central trusted party, visitor statistics will have to be forwarded to random network participants to ensure equality.

---

[2] A Sybil attack implies a single peer assuming multiple forged identities with a malicious goal to exploit or undermine the network.

[3] A group of ill-intent edge nodes agreeing between themselves to collectively store only a single copy of the data while claiming to store an arbitrary number of copies.

A less invasive way is to employ the *synthetic monitoring technique*, whereby network nodes periodically send HTTP requests to each other, simulating real-user interactions with the served web page. While efficiently adhering to the above limitations, this method generates extra traffic and, more importantly, provides only an approximation of a given node's trustworthiness in terms of content delivery – the proofs solely verify that E_i is *capable* of serving f at given points in time, not that it has *actually served* it a specific number of times to real end-users.

However, as the number of deliveries grows, the results of synthetic monitoring become more and more characteristic of the actual delivery patterns, as follows from the law of large numbers, which states that the sample mean $\bar{X}_n$ converges to the true mean μ as the sample size n increases:

$$\lim_{n\to\infty} \Pr(|\bar{X}_n - \mu| > \varepsilon) = 0$$

In other words, the higher the number of checks and the higher their proportion in the overall traffic, the more confidence can be had about the actual delivery numbers; in practice, a balance must be struck between the extra network load and the desired monitoring precision by dynamically adjusting the frequency of synthetic requests relative to the popularity of particular data. This method is conceptually similar to how the output of large industrial production lines is checked for defects, or how polling is used to infer useful conclusions about the entire population.

After successfully receiving f from the edge node under audit, its peer can compare the hash of the data with the network's Merkle DAG (see section 2.4 for details on content addressing) to ensure that it hasn't been tampered with in any way. Nodes with poor audit results are to be penalized by reduced participation priority (leading to lower earnings) or, ultimately, by being excluded from the network completely.

With both proofing mechanisms provided by peer edge nodes, the network can audit itself without the need to use specialized software on client or user side; a reputation system with each node storing a ranking of its closest peers can be suggested as a natural extension of the setup, improving the overall quality of the network by rewarding well-behaved nodes and punishing non-cooperative or outright malicious participants. The full storage/delivery cycles, including audits, can be presented graphically as follows:

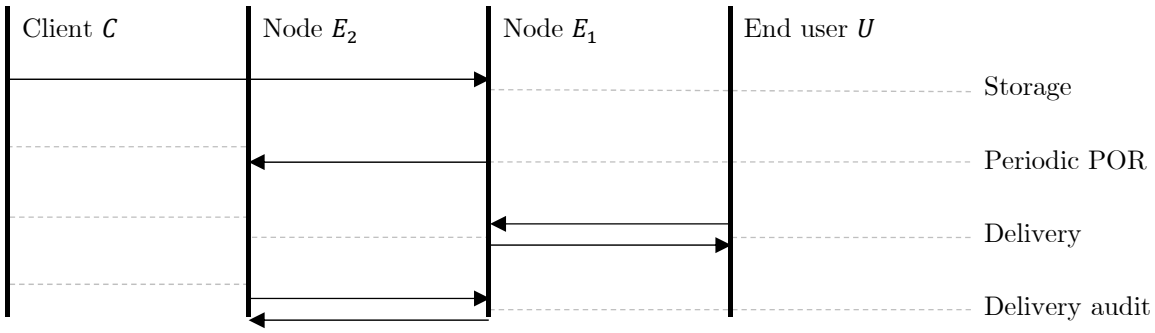

Figure 3: Storage and delivery cycles with audits. Thick vertical lines represent parts of the network; time progresses from top to bottom, with each horizontal line representing a specific action by network participants.

It is also worth mentioning that a sustainable delivery-proofing process should be set up in such a way that the nodes under audit are not able to distinguish between genuine end user requests and audit requests from peer nodes; moreover, participating nodes should be incentivized to periodically audit their peers as described above. This can be done either by directly rewarding audits or by including them as a positively correlated factor in electing nodes for the network consensus process, discussed below.

3.2. Consensus Protocol

Based on the expectations and network layout described in part 2, there are several properties that a sustainable consensus method for a decentralized p2p CDN must possess:

- *Byzantine tolerance*: while being a generic requirement for any workable consensus protocol, it should be mentioned nonetheless, given the permissionless state of the underlying blockchain – the system must be resilient to a certain number of faulty nodes (including wilful manipulation) within the theoretical limits of $n \geq 3m + 1$ where n is the total number of nodes and m is the number of faulty nodes [29].

- *Speed*: it should not slow down the content delivery cycle in any way, since page load speed is a critical requirement by the end users of a CDN.

- *Low frictions*: given the potentially high number of microtransactions arising from each act of delivering the content to end users, transaction costs should be minimized if the system is to make economic sense.

- *Cheap validation:* if the network is to maintain its decentralization in the long term, it should be feasible for most of the participants to validate transactions and blocks; in other words, consumer-grade devices should be able to handle the volumes of data, bandwidth requirements, and computational tasks related to validation.

- *External resources*: consensus protocols relying solely on the internal states of the system have been demonstrated as notoriously difficult to secure [30].

- *Scalability*: since the network is expected to be able to support any number of content delivery instances (page loads), it is important for the proofing protocol to support large numbers of transactions per second.

- *Energy efficiency*: this requirement is closely related to the previous two, since high electricity usage acts as an obstacle to both scaling and adoption rates.

Based on the above criteria alone, it is possible to rule out several existing alternatives, including directly using the Bitcoin blockchain or its forks for recording all transactions due to capacity, speed and scalability limitations as well as high transaction costs [31]. The Filecoin protocol, which focuses primarily on storage-based proofing [16], can also be argued to provide insufficient robustness for the needs of a CDN at the moment.

Given the Nexusless network's orientation on utility (delivering content to Internet users) rather than storage and transfer of financial value, it is reasonable to consider consensus methods that rely on utility creation rather than expending useless work (inverting hashes as in Bitcoin [32] and its derivatives) or hoarding tokens (as in pure proofs-of-stake). An election mechanism can be introduced that fairly and randomly chooses the creators of the next block based on their network power, which is in turn proportional to the network value produced by the particular node. Both storage and delivery cycles can be seen as the sources of value, implying that the nodes which consistently store and deliver more content (based on auditable evidence described in section 3.1) will have higher network power.

The inclusion of content delivery cycle activity in the network power calculation also serves to prevent aggregation of power by a few nodes by investing into significant storage capacity (e.g. entire data centers) in far-off regions with cheap electric power, as it is currently the case with Bitcoin's ASIC miners [33]. Since demand for content delivery is expected, on average, to naturally gravitate towards large population centers (such as country capitals and significant regional cities) where electricity, land, and workforce tend to be more expensive, the advantage gained by large players in location-agnostic blockchains is significantly diluted in favour of the nodes which are situated as close as possible to the end users.

Relative network power at time t of an edge node E_i on the Nexusless blockchain with n participants can therefore be expressed as a combination of its share of verified occupied storage s and outbound traffic d:

$$P_i^t = \omega \frac{s_i^t}{\sum_{j=1}^{n} s_j^t} + (1 - \omega) \frac{d_i^t}{\sum_{j=1}^{n} d_j^t}$$

The coefficient ω is introduced to be able to adjust the relative weights (i.e. importance) of the storage and delivery cycles as the network evolves.

The process of electing nodes that add blocks at each time interval can therefore be modelled as a probabilistic consensus based on the entropy ε extracted from the blockchain itself (e.g. hashes of previous blocks concatenated with timestamps):

$$P_i^t \geq \frac{\mathrm{H}(\varepsilon_t^i)}{2^h}$$

The superscripted ε_t^i above is used to denote public entropy at time t signed by the node i. If h is chosen as the (constant) length of the output of the cryptographic hash function H, the right side of the expression above returns a normalized quasi-random variable (uniformly distributed between 0 and 1); this means the expression is true with the probability P_i^t.

In addition to being deterministic at each period t, such consensus mechanism possesses several important properties: it is *secret* and *publicly verifiable* at the same time, given the fact that ε_t^i is signed by a secret key known only to node i yet provable using their public key.

Despite satisfying most of the requirements listed at the beginning of the current section, the proposed consensus does not fully address the issues of speed and scalability crucial to the operation of a CDN: while storage transactions are not time-critical, the content delivery cycle is not able to tolerate waiting for multiple confirmations from the blockchain before initiating data transfer to the end user. This calls for an additional layer of infrastructure to support near-instantaneous payment channels for fast content delivery.

3.3. Off-Chain Microtransaction Layer

Given the expected number of data delivery instances (many orders of magnitude above the current transaction rates for Bitcoin [34]) as well as the expected equilibrium price per such transaction (small fractions of 1 USD, based on existing CDN pricing plans, e.g. [35]) a blockchain-based peer-to-peer content delivery

network requires a subsystem for securely processing large quantities of micropayments between untrusted parties. Another requirement for this structural layer is extremely low latency (i.e. high transaction speed), given the time-critical nature of web content delivery process.

While the current permissionless blockchain technologies clearly cannot meet the above requirements, new solutions are emerging that specifically address the speed and scalability issues; one of the most promising protocols is the Lightning Network, detailed in [36] and currently in the Testnet phase [37] with some tentative reports of physical purchases [38].

The system works by constructing a mesh of bidirectional payment channels, which allow updating the balance between the two concerned parties at any time, with near-arbitrary frequency. *Revocable Delivery Transactions* and *Breach Remedy Transactions* are used to provide motivation for both channel participants to cooperate, as all committed funds are given to the other party in case of violation of the terms of the channel. Multi-hop channels are made possible by the introduction of Hashed Timelock Contracts (HTLCs) with decrementing time-locks which allow for participation of intermediate nodes without indirect counterparty risks [36]. The entire construction is supported by the public blockchain as a source of programmatic arbitration to ensure trust-less resolution of any disputes.

It is worth noting that the said microtransaction layer is not an entirely separate system acting on top of the blockchain, rather an algorithmic extension of the blockchain which allows significantly reducing the proportion of transactions actually broadcasted to the public network.

In the context of a peer-to-peer CDN, the payment channel network is used to allow the clients (i.e. the origin server side) to compensate the edge nodes for each act of storage and delivery via micropayments instead of having to broadcast every single transaction on the blockchain. This greatly off-loads the blockchain itself and vastly improves the granularity of peer-to-peer compensation.

4. Ensuring Quality and Robustness

If the Nexusless network is to function sustainably without outside interventions, it must meet the needs of all involved parties, namely the clients (webmasters, content creators and service operators using the CDN to disseminate their content), the edge node operators (device owners who store and deliver content for the clients) as well as the end users consuming the content (who might or might not be edge node operators). Based on the challenges of the traditional CDNs presented in section 1.1 and the required properties outlined in section 2.1 we are going to discuss measures that can help ensure economic viability of the proposed network design.

4.1. Optimizing Operational Capabilities

As its name suggests, a CDN should be able to *deliver* specific content to any network user, which implies that *reliability* can be regarded as the cornerstone of any such construct – without the end users actually receiving the requested data the entire idea loses its further meaning.

Despite the fact that many large specialized content delivery networks have moved to a "100% uptime guarantee + service credits for downtime" in their Service Level Agreements [39] [40] [41], essentially finding a legal loophole for flexible management of availability promises, certain internal CDNs of large diversified multinationals do offer specific figures to serve as proxy benchmarks: Microsoft's SLA cites a 99.9% uptime guarantee [42], while Google's contains a reference to a 99.95% level [43]. To further put this into perspective and add actual observed figures, Google claims to achieve 99.978% availability for its key services [44], which can be reasonably assumed as web user expectation benchmark for established internet businesses.

Various measures exist for ensuring maximum availability, including redundancy, backups, partitioning, automation, fault tolerance [45]. While the latter is discussed separately in the following section, we elaborate on the remaining measures for the case of a distributed, peer-to-peer CDN.

Since the technical characteristics and individual availability of each single device/node in the Nexusless network are significantly inferior to the ones used in traditional commercial CDNs, they need to be offset, among other factors, by the number of devices as well as easier deployment of new nodes. Indeed, both parameters are expected to be much higher in a free-to-join network in which anyone with a consumer-grade computing device can store and server the content.

Low expected availability of individual nodes is due to a combination of lower fault tolerance and the fact that the majority of such participants will need to be able to set specific resource limits (processing power, storage space, memory, bandwidth, battery life per day) for the application running the node, in order for it to be practical i.e. not impair the device's usability.

This can be modeled as follows: take a random sample of n nodes in a specific geographic area (i.e. n is low relative to the number of nodes in the entire network N) and assume some average probability p of any given node being operational at a given time; in other words, each instance of checking whether a node is online can be viewed as a Bernoulli trial, returning either "yes" or "no" as the outcome. The uptime u of this subnetwork, or the overall probability that at least s edge nodes will be available at any given time, can be defined as the complement of the cumulative binomial distribution function (s successes out of n trials):

$$u = 1 - \sum_{i=0}^{s-1} \binom{n}{i} p^i (1-p)^{n-i}$$

A noteworthy property of this function is that for a certain fixed s the value of u has a relatively narrow window with respect to n where it is sizably non-extremal, i.e. is between 0 and 1; in other words, after a certain n the value of u quickly increases from zero to unity. This means that, all other factors held constant, for a given required minimum number of available edge nodes, the number of additional nodes needed to achieve near-100% collective uptime is small relative to the total number of nodes:

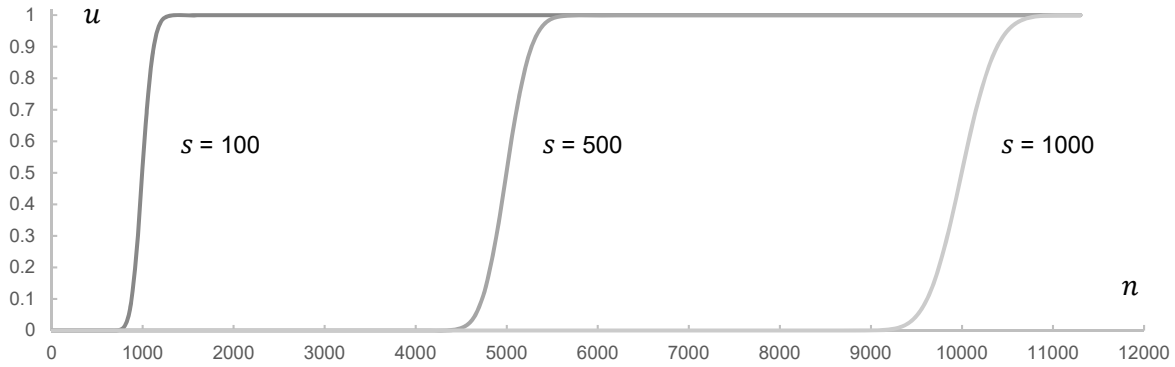

Figure 4: Probability of at least s nodes being available, depending on the total number of nodes. Can also be interpreted as network uptime depending on the total number of nodes (x-axis), given three specific values for the minimum number of available nodes (in each case the individual availability p is fixed at 0.1).

This is both good news and bad news: to ensure that a given number of edge nodes is always online, the network needs to add a relatively small number of new devices even if each individual device is expected to contribute only a small fraction of its operating

resources; at the same time, until the total number of nodes rises past a certain threshold the probability of a given number of devices being simultaneously online stays negligible. As a result, a distributed peer-to-peer CDN should aim to include the highest possible number of edge nodes at any given location if it wants to meet the goal of stable uptime.

Moreover, popular content should be duplicated at a large fraction of all nodes to ensure that a certain number of devices with this data is online at any given time. Additional autonomous management is required to adjust the degree of content replication to account for time-patterns in both content demand and node supply, as well as low-probability traffic spikes. This calls for adaptive fault tolerance mechanisms that dynamically choose a suitable redundancy strategy under changing operating environment [46]. Part of these issues can also be auto-regulated by a market-based incentive layer, described in more detail section 3.

Apart from the service being available at all, *load speed* is the CDN feature that is the most salient to the end user, which means that this is also what will be demanded by the clients. The Nexusless network can use its scale to tackle both major components of this parameter: latency and data transfer speeds. If a sufficient number of edge nodes can be guaranteed to stay online at any given time, a situation can be reached where each end user is always in close proximity to many participating devices active in the same locality (city, village, or even a part thereof). This implies that each end user can be served by the closest edge node(s), minimizing most factors affecting round-trip time: physical distance, the number of intermediate nodes, and network congestion. The mechanism for choosing which node to retrieve a piece of data from can be optimized for minimum latency and high reliability using e.g. replication with cancelling [47].

In addition to the above, modern consumer-grade computing devices can afford to store a certain share of the cached pages/content (e.g. the most demanded ones) directly in the RAM, making retrieval even faster than with solid-state hard drives.

4.2. Fault Tolerance and Attack Resistance

A highly distributed content delivery system with low reliability of diverse individual nodes such as Nexusless can be classified as a distributed computing system operating mostly in hard-real-time mode [48]. In this system it can be useful to view individual connected devices as atomic nodes, disregarding their internal structure; such approach helps to focus on the network-level robustness and the protocols aimed at maximizing network performance, given to low-to-none control over the performance of each given node.

While the fundamental prerequisite to fault tolerance – the absence of a single point of failure via replication and redundancy – is at the core of the concept of a CDN [1] and is therefore already being implemented in any such system, a switch from dedicated server farms to consumer device nodes requires additional measures for ensuring reliability and security. This is due to a multitude of factors, including lower natural uptime, limited computing resources, higher opacity as well as substantially decreased reliability of individual edge nodes.

In other words, a distributed peer-to-peer network must be able to tolerate a certain share of faulty nodes (including those functioning incorrectly and not functioning at all) as it is highly likely that at least some faulty nodes will indeed appear in the network once participation is practically free for everyone. Below we discuss some of the important considerations by examining the issue from information and transport dimensions as well as proposing measures to increase attack resistance.

Data faults encompass any errors affecting the actual bytes that are to be delivered to the end user; this includes both data corruption and loss during storage as well as retrieval cycles (see Table 1). In the Nexusless network, data genuineness is ensured by using content addressing: each data block can be found by its unique hash which changes unpredictably if any single bit of the underlying content is changed (see section 2.4). Data availability is solved by high degree of replication (see section 4.1) in combination with periodic proofs of storage run against the existing data on active edge nodes.

Transport faults relate to problems with transmitting the data between network participants – i.e. along the paths of types $C \rightarrow E$, $E \rightarrow E$ and $E \rightarrow U$ (notation preserved from Table 1). These may occur due to problems with network links or nodes becoming unresponsive or faulty before or in the middle of a transmission. This is handled by the network protocols described in section 2.5 as well as the systems used for peer-to-peer communication (section 2.4).

Attacks can be defined as network faults caused by conscious actions, in other words faults that would not manifest with significant probability in the absence of malicious agents. Such actions can be classified by their intention into attacks aiming at reducing (or nullifying) the capabilities of the network, attacks aimed at delivering false data to end users (e.g. substituting the content of a web page), and those having the goal of extracting unwarranted profits from the network (in the presence of an incentive layer).

The former group includes a wide range of attack types, most importantly Sybil, Eclipse (an attacker gaining control over a part of the network) as well as denial of service. The risk of such attacks is greatly reduced by the use of the enhanced version of the Kademlia protocol in the underlying file system (see section 2.4). Profit-extraction attacks are to be addressed by proofing and auditing within the

storage/retrieval markets, which is discussed in mode detail in the next part of the current paper.

The features described above help ensure high partition tolerance as well as reliability for the discussed peer-to-peer content delivery network; in the following section we explore how the system's economic viability can be further increased by optimizing resource use.

4.3. Costs and Resource Consumption

Apart from the need of any stable infrastructure solution to be economically viable, resource consumption of the Nexusless network is an important consideration since it is largely born by the edge nodes, i.e. private participants using their personal devices. As the most significant variable input is electricity, two key energy demands need to be considered: running the storage/retrieval cycles as well as achieving consensus in the decentralized incentive layer. The former component is not to be neglected, given that the global content delivery infrastructure consumes a significant share of the total ICT (information and communication technologies) sector's electricity outlays [49].

The latter is worth considering because, despite being a relatively new technology, blockchain has already received attention in the 2017 report of the International Energy Agency, mainly due to the growing power consumption of the bitcoin network [50]; there are, however, two key distinctions in the Nexusless concept with respect to the largest cryptographic currency: locality and proofing. Since bitcoin uses the power-intensive, location-independent hashcash-based proof-of-work mechanism for achieving consensus [32], proofing nodes are incentivized to minimize their input costs by seeking out regions with the lowest electricity prices – which in turn has led to a sizeable fraction of such activity clustering in remote regions of China, running on cheap coal or hydro power [33]. In contrast to that, a peer-to-peer CDN rewards proximity to end users, which implies locating edge nodes in places with the highest internet traffic demand, e.g. large urban centers (see also section 3.2).

In any case, serving content and participating in the incentive layer requires dedicating a certain amount of resources, with the effect being more noticeable on average on mobile, battery-powered devices. This means that edge node operators should be able to set custom limits for electricity consumption and/or percentage of battery drain, for each device connected to the Nexusless network.

Other resources consumed by Nexusless edge nodes include bandwidth and hardware (through faster depreciation due to higher load). While the latter is outside of the scope of the current paper and requires additional research and testing, the former needs to be considered from the onset since any device on a metered connection can potentially incur prohibitive expenditures for content delivery. This can be solved by

both giving edge node users control over how much outbound traffic the application is allowed to consume per unit of time (per connection type), as well as utilizing algorithms and smart defaults to limit upload activity on specific networks. In addition to that, individually adjustable limits for RAM, CPU load, as well as storage space consumption per unit of time are highly recommended to make it attractive for the widest range of users to join as edge node operators.

5. Future Potential

The Nexusless CDN combines widely used concepts, including Distributed Hash Table based peer-to-peer block exchanges and version management, with the most recent developments in cryptography, such as blockchains with probabilistic consensus based on proofs of retrievability as well as networks of bidirectional micropayment channels, to arrive at a practical solution to decentralized delivery of web content. The resulting framework not only meets the expectations from any usable CDN, but also allows any owner of a personal computing device to participate as a storage/delivery node and get paid for their services.

5.1. Possible Extensions

Despite being a significant step from the current status quo, the Nexusless network is not only an end goal but also a possible basis for a new generation of inherently more decentralized web services, contributing to a fairer Internet where any individual can monetize their available computing and networking resources in an ecosystem of open and robust public marketplaces.

While the edge nodes are quite symmetric in most respects in the original Nexusless design, specialization might become practical as more participants join the network. This potentially allows for new useful collective capabilities, resulting in additional services for the end users, such as for example:

- *Online storage* – if the originator of the content and the end user are the same entity, the respective storage and delivery cycles effectively represent a personal web drive, with the specializing edge node(s) being compensated for holding and retrieving versioned files for their owner.

- *Web hosting* – while most edge nodes cannot be expected to be permanently available and be able to process dynamic queries (requirements needed to make them practical as actual hosts or origin servers), some might choose to do so given their hardware capabilities and uptime constraints.

- *Site builders* – closely related to the previous example, this one implies the emergence of applications which use the Nexusless network as a decentralized hosting and content delivery platform, while offering value-added services related to creating websites, such as interactive visual composers and content management systems.

- *Universal wallets* – in order to facilitate adoption, specialized wallets will need to accept fiat currency and use outstanding balances to purchase services within the Nexusless network (mediated by the network tokens); however, as the user base grows, it is possible to envisage some of those wallets becoming much more versatile and extending the direct use of the Nexusless tokens to payments for other web services, such as paywalled content, freelancing, e-commerce, etc. The microtransaction layer described in section 3.3 greatly extends the range of such practical applications.

Extensions described above can be developed by independent teams with free access to the underlying network's facilities, resulting in a dynamic ecosystem of competing applications. Those can be free and open-source or profit-oriented, similarly to how the web server or content management system markets operate today.

5.2. Topics for Further Research

Certain areas require additional development and enhancements before they become practical, including, foremost, the following:

- Refining proofs of delivery to either find the right balance between additional network load vs audit frequency (and thus precision of inferences about the total number of deliveries per node) – or replacing them with a mechanism where proofers are able to get the relevant data directly from end users.

- Testing algorithms for efficient routing of requests by the content delivery network to find the closest and fastest edge node(s) for each specific request.

- Maximizing positive incentives in the blockchain consensus mechanism and checking its resilience against malicious actions.

- Fine-tuning algorithms for determining the optimal number of replicate stored copies of a data instance, given information about edge node activity and uptime.

- Exploring the actual performance and limitations of consumer-grade devices while functioning as active network nodes (power consumption, bandwidth, RAM, CPU load).

Ongoing research is thus focused on making the various components of the Nexusless network function together in a useful and secure fashion. This paper will see new version releases as the work progresses.

References

[1] S. Hull, Content Delivery Networks: Web Switching for Security, Availability, and Speed, McGraw-Hill Professional, 2002.

[2] "Cisco Visual Networking Index: Forecast and Methodology, 2016–2021," 15 September 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. [Accessed November 2017].

[3] "Akamai Technologies," [Online]. Available: https://www.akamai.com/. [Accessed December 2017].

[4] "Stackpath," [Online]. Available: https://www.stackpath.com/. [Accessed December 2017].

[5] "Google Edge Network," [Online]. Available: https://peering.google.com/. [Accessed November 2017].

[6] "CDN Architecture," Akamai Technologies, [Online]. Available: https://www.akamai.com/us/en/resources/cdn-architecture.jsp. [Accessed November 2017].

[7] "Content Delivery Market Report," MarketsandMarkets Research, 2017. [Online]. Available: https://www.marketsandmarkets.com/PressReleases/cdn.asp. [Accessed December 2017].

[8] "CDN Pricing for Entrepreneurs," MaxCDN, [Online]. Available: https://www.maxcdn.com/pricing/entrepreneur/. [Accessed November 2017].

[9] "Incident Report on Memory Leak Caused by Cloudflare Parser Bug," Cloudflare Blog, 23 February 2017. [Online]. Available: https://blog.cloudflare.com/incident-report-on-memory-leak-caused-by-cloudflare-parser-bug/. [Accessed December 2017].

[10] G. Wassermann, "Forwarding Loop Attacks in Content Delivery Networks May Result in Denial of Service," CERT Carnegie Mellon University, 26 February 2016. [Online]. Available: https://www.kb.cert.org/vuls/id/938151. [Accessed November 2017].

[11] S. Khalaf and L. Kesiraju, "U.S. Consumers Time-Spent on Mobile Crosses 5 Hours a Day," Flurry Analytics, 2 March 2017. [Online]. Available: http://flurrymobile.tumblr.com/post/157921590345/us-consumers-time-spent-on-mobile-crosses-5. [Accessed December 2017].

[12] "About BitTorrent," BitTorrent, [Online]. Available: http://www.bittorrent.com/company/about. [Accessed November 2017].

[13] P. Maymounkov and D. Mazieres, "Kademlia: a Peer-to-peer Information System Based on the XOR Metric," in *First International Workshop on Peer-to-Peer Systems*, 2002.

[14] A. Loewenstern and A. Norberg, "DHT Protocol," Bittorrent.org, 31 January 2008. [Online]. Available: http://www.bittorrent.org/beps/bep_0005.html. [Accessed November 2017].

[15] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," *Protocol Labs white paper,* 2016.

[16] J. G. N. Benet, "Filecoin – a Decentralized Storage Network," Protocol Labs white paper, 2017.

[17] D. Salamon, G. Simonsson, B. Vohaska and B. Fox, "Orchid: Enabling Decentralized Network Formation and Probabilistic Micro-Payments," 2017.

[18] "The Need for Mobile Speed: How Mobile Latency Impacts Publisher Revenue," DoubleClick by Google, 2016. [Online]. Available: https://www.doubleclickbygoogle.com/articles/mobile-speed-matters/. [Accessed December 2017].

[19] M. McDowell, "Understanding Denial-of-Service Attacks," United States Computer Emergency Readiness Team, 4 November 2009. [Online]. Available: https://www.us-cert.gov/ncas/tips/ST04-015. [Accessed November 2017].

[20] S. Gilbert and N. Lynch, "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services," *ACM SIGACT News,* vol. 33, no. 2, 2002.

[21] D. Mazieres and F. Kaashoek, "Escaping the Evils of Centralized Control With Self-Certifying Pathnames," in *Proceedings of the 8th ACM SIGOPS European Workshop on Support for Composing Distributed Applications*, Sintra, Portugal, 1998.

[22] I. Baumgart and S. Mies, "S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing," in *Proceedings of the 13th International Conference on Parallel and Distributed Systems*, 2007.

[23] H. Shacham and B. Waters, "Compact Proofs of Retrievability," in *Advances in Cryptology - ASIACRYPT*, 2008.

[24] Y. Deswarte, J. Quisquater and A. Saïdane, "Remote Integrity Checking," in *Integrity and Internal Control in Information Systems VI*, Boston, MA, United States, Springer, 2004.

[25] N. Greco and J. Benet, "Proof of Replication," Protocol Labs, 2017.

[26] S. Wilkinson, T. Boshevski, J. Brandoff, J. Prestwich, G. Hall, P. Gerbes, P. Hutchins and C. Pollard, "Storj, a Peer-to-Peer Cloud Storage Network," 2016.

[27] "Performance & Real User Monitoring Tools (RUM)," Pingdom, [Online]. Available: https://www.pingdom.com/product/performance-monitoring. [Accessed January 2018].

[28] L. Matsakis, "Over 400 of the World's Most Popular Websites Record Your Every Keystroke," Vice Media, 20 November 2017. [Online]. Available: https://motherboard.vice.com/en_us/article/59yexk/princeton-study-session-replay-scripts-tracking-you. [Accessed January 2018].

[29] M. Pease, R. Shostak and L. Lamport, "Reaching Agreement in the Presence of Faults," *Journal of the AssoctaUon for Compmmg Machinery,* vol. 27, no. 2, 1980.

[30] V. Buterin, "Slasher Ghost, and Other Developments in Proof of Stake," Ethereum Foundation, 3 October 2014. [Online]. Available: https://blog.ethereum.org/2014/10/03/slasher-ghost-developments-proof-stake/. [Accessed December 2017].

[31] K. Croman, C. Decker, I. Eyal, A. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Sirer and D. Song, "On Scaling Decentralized Blockchains," *Position Paper,* 2016.

[32] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[33] J. Wong and J. Simon, "Inside One of the World's Largest Bitcoin Mines," Quartz, 17 August 2017. [Online]. Available: https://qz.com/1055126/photos-china-has-one-of-worlds-largest-bitcoin-mines/. [Accessed December 2017].

[34] "Bitcoin Transaction Rate," Blockchain.info, [Online]. Available: https://blockchain.info/charts/transactions-per-second. [Accessed January 2018].

[35] "Amazon CloudFront Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/cloudfront/pricing/. [Accessed January 2018].

[36] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," 2016.

[37] "Lightning Network Explorer (Testnet)," Lightning Network, [Online]. Available: https://explorer.acinq.co/. [Accessed January 2018].

[38] J. Buck, "Lightning Network's Pizza Day? First Ever Physical Purchase On Lightning Network," CoinTelegraph, 22 January 2018. [Online]. Available: https://cointelegraph.com/news/lightning-networks-pizza-day-first-ever-physical-purchase-on-lightning-network. [Accessed January 2018].

[39] "Cloud Architecture," Akamai, [Online]. Available: https://www.akamai.com/us/en/resources/cloud-architecture.jsp. [Accessed November 2017].

[40] "Legal Information," MaxCDN, 18 November 2013. [Online]. Available: https://www.maxcdn.com/legal/. [Accessed November 2017].

[41] "Business Service Legel Agreement," CloudFlare, [Online]. Available: https://www.cloudflare.com/business-sla/. [Accessed November 2017].

[42] "SLA for Content Delivery Network," Microsoft, May 2015. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/cdn/v1_0/. [Accessed November 2017].

[43] "Google Cloud CDN Service Level Agreement," Google, 28 June 2016. [Online]. Available: https://cloud.google.com/cdn/sla. [Accessed November 2017].

[44] "Reliability - Google Cloud Help," Google, [Online]. Available: https://support.google.com/googlecloud/answer/6056635?hl=en. [Accessed November 2017].

[45] M. Hawkins and F. Piedad, High Availability: Design, Techniques, and Processes, Prentice Hall, 2000.

[46] O. González, H. Shrikumar, J. A. Stankovic and K. Ramamritham, "Adaptive Fault Tolerance and Graceful Degradation Under Dynamic Hard Real-time Scheduling," *Computer Science Department Faculty Publication Series,* vol. 188, 1997.

[47] Z. Qiu, J. Perez and P. Harrison, "Tackling Latency via Replication in Distributed Systems," in *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, Delft, The Netherlands, 2016.

[48] K. Kim and T. Lawrence, "Adaptive Fault-Tolerance in Complex Real-Time Distributed Computer System Applications," *Computer Communications,* vol. 15, no. 4, 1992.

[49] A. Bianco, R. Mashayekhi and M. Meo, "Energy Consumption for Data Distribution in Content Delivery Networks," in *IEEE International Conference on Communications (ICC)* , Kuala Lumpur, Malaysia, 2016.

[50] "Digitalization and Energy," International Energy Agency, 2017.

[51] N. Koblitz and A. Menezes, "Cryptocash, Cryptocurrencies, and Cryptocontracts," *Designs, Codes and Cryptography,* no. 78.