

PEER-REVIEWED RESEARCH

OPEN ACCESS

ISSN Online: 2516-3957

ISSN Print: 2516-3949

[https://doi.org/10.31585/jbba-9-2-\(6\)2026](https://doi.org/10.31585/jbba-9-2-(6)2026)

Decentralised Web Technologies: A Review with Implementation Case Study

Joseph W. Matiko

Dar es Salaam Institute of Technology, Tanzania

Correspondence: jwmatiko@gmail.com**Received:** 14 January 2026 **Accepted:** 11 May 2026 **Published:** 22 July 2026

Abstract

The World Wide Web is undergoing a paradigm shift from the centralised platforms of Web 2.0 to the user-sovereign Decentralised Web (Web 3.0). However, the current transitional phase, often termed Web 2.5, remains structurally vulnerable due to its persistent reliance on centralised infrastructure for frontend delivery and domain routing. Applying a structured selection methodology, this article presents a review of decentralised web technologies with an implementation case study, dissecting the ecosystem's three foundational pillars: Decentralised Domain Name Systems (DNS), Hosting, and Storage. Across these three categories, we provide a comparative analysis of 20 prominent protocols, evaluating mechanisms ranging from overlay networks such as the Ethereum Name Service (ENS) to content-addressed systems such as the InterPlanetary File System (IPFS) and permanent cryptographic ledgers such as Arweave. Furthermore, to synthesise these theoretical pillars, the study investigates the emerging On-Chain Web paradigm through a detailed practical case study of the Massa DeWeb architecture. By analysing its distinct approach to natively embedding full website logic, naming registries, and data assets directly within the ledger to eliminate intermediary gateways, we demonstrate the functional viability of a fully decentralised, serverless application stack that effectively resolves the fragility of hybrid network architectures.

Keywords: *Decentralised Web, Blockchain DNS, On-Chain Web, Smart Contracts, DeWeb***JEL Classifications:** *L86, O33, D85, L17, L96*

1. Introduction

The World Wide Web has evolved from the static repositories of Web 1.0 to the dynamic, albeit centralised, platforms of Web 2.0. To understand the current imperative for decentralisation, it is essential to trace this trajectory.

Web 1.0, also known as the Read-Only Web, spanned roughly from 1990 to 2004. This era was defined by static content served from file systems. It was built on open, decentralised protocols such as Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), and File Transfer Protocol (FTP), where users were primarily consumers of information. While decentralised in architecture, the barrier to entry for publishing was high, limiting the number of content creators.

Web 2.0, also known as the Read-Write Web, emerged around 2004, introducing dynamic, interactive applications that allowed users to generate content easily [1]. However, this convenience came at the cost of centralisation. Platforms like Facebook, Google, and Twitter (now X) aggregated vast amounts of user data into proprietary silos, effectively becoming the gatekeepers of the internet [2]. While users

could write content, they did not own the underlying data or the platform itself.

As the limitations of Web 2.0 became apparent, a transitional phase often termed Web 2.5 emerged. While “Web 2.5” is used informally in the literature, recent academic work has begun to formalise the term as describing hybrid architectures that connect decentralised backends (smart contracts) with traditional centralised frontends and DNS infrastructure [3]. While this allowed for early adoption of blockchain technologies, it introduced a critical fragility of relying on legacy infrastructure for access.

Dependence on centralised infrastructure has emerged as a significant vulnerability, even for protocols designed to be decentralised. The Web 2.5 model has recently resulted in catastrophic security breaches. Notably, the BadgerDAO protocol sustained a loss of approximately \$120 million following the compromise of a Cloudflare API key, which allowed attackers to inject malicious scripts into the frontend interface [4]. Similarly, Curve Finance users suffered losses exceeding \$570,000 due to a DNS hijacking attack that redirected the official domain to a malicious site [5]. These incidents underscore the critical observation that a truly

decentralised application cannot securely rely on centralised web servers or legacy DNS infrastructure for its delivery.

The Decentralised Web (Web 3.0) signifies a unified effort to revive the initial concept of a peer-to-peer internet. In contrast to the centralised model, where data resides in silos managed by intermediaries, the decentralised web leverages distributed ledger technologies (DLT) and cryptographic primitives to distribute control. The significance of this transition is encapsulated in the Read-Write-Own paradigm. Whereas Web 2.0 permits users to read and write content, Web 3.0 empowers users to own their data, identity, and digital assets independent of trusted third parties [6, 7]. This architecture results in enhanced resilience, censorship resistance, and data sovereignty.

To systematically address these developments, this paper provides a review of decentralised web technologies alongside a practical synthesis through an implementation case study. The remainder of this study is structured as follows: Section 2 establishes the methodology and justifies the selection criteria for the 20 protocols analysed across the web stack. Section 3 conducts a comprehensive comparative review of the three fundamental pillars of Web 3.0 infrastructure: Decentralised Domain Name Systems (DNS) replacing centralised routing authorities (Section 3.1), Decentralised Hosting shifting from location-based to content-based addressing (Section 3.2), and Decentralised Storage Networks (DSNs) guaranteeing data persistence through cryptographic models (Section 3.3). Section 4 synthesises these theoretical concepts through a practical, comprehensive case study on the Massa DeWeb, a novel On-Chain Web architecture that natively integrates all three pillars directly into the blockchain ledger to demonstrate a truly sovereign application stack. Finally, Section 5 concludes by summarising the implications of these technological shifts and the future trajectory of entirely serverless, sovereign applications.

2. Methodology

This review adopts a structured selection methodology based on explicit selection criteria for state-of-the-art technological surveys [8]. To ensure a rigorous and balanced evaluation of the Decentralised Web (Web 3.0) ecosystem, a multi-stage selection framework was developed.

The protocol selection process focuses on four primary inclusion criteria:

A. Academic and Industry Relevance – evidenced by formal citations, active developer ecosystems, and comprehensive technical documentation.

B. Technical Maturity – requiring operational mainnet deployments rather than theoretical whitepapers.

C. Architectural Diversity – ensuring the inclusion of distinct engineering paradigms (e.g., smart contract overlays versus sovereign distinct blockchains, location-based versus content-based addressing).

D. Pedagogical Value – incorporating both foundational pioneer protocols and state-of-the-art emerging architectures to highlight evolutionary trajectories.

To maintain analytical neutrality, systems were excluded if they lacked verifiable mainnet implementations, were fundamentally redundant in architecture relative to selected representatives, or had insufficient documentation to support rigorous technical evaluation. For instance, protocols such as Ceramic Network (which focuses primarily on mutable data streams and cross-chain identity) and Cosmos IBC/Polkadot (which represents Layer-0 interoperability frameworks) were deliberately excluded from the DNS comparative analysis, as they represent distinct architectural layers outside the primary scope of dedicated naming registries. The selected protocols are categorised into the three fundamental infrastructural pillars of the decentralised web: Naming Systems, Hosting, and Storage. Tables 1, 2, and 3 provide detailed inclusion justifications for the protocols selected within each respective category to ensure reproducibility and transparency.

2.1 Decentralised Domain Name Systems (DNS) Selection

Decentralised Domain Name Systems (DNS) represent a distributed, consensus-based naming infrastructure where the mapping of human-readable labels to network identifiers is maintained on a public ledger rather than a hierarchical database governed by a central authority [9]. To comprehensively review this pillar, naming protocols were selected to contrast general-purpose blockchain overlays with sovereign blockchains engineered explicitly for alternative root-zone decentralisation.

Table 1. Selected Decentralised DNS Protocols and Justification

Protocol	Selection Justification
Ethereum Name Service (ENS) [10]	Represents the industry-standard overlay architecture via L1 smart contracts and a rent-based economic model.
Namecoin [11]	Included for historical context as the inaugural implementation of decentralised DNS natively merged-mined with Bitcoin.
Handshake (HNS) [12]	Represents sovereign alternative root systems striving to decentralise the ICANN root zone using a custom UTXO chain.
Unstoppable Domains [13]	Selected to evaluate the perpetual ownership model and Layer 2 (Polygon) scalability implementations.
Stacks (BNS)[14]	Illustrates naming anchored to Bitcoin security (L2) with intrinsic integration of Decentralised Identifiers (DIDs).
Massa (MNS) [15]	Represents an emerging On-Chain Web paradigm that seamlessly integrates resolution natively into an L1 architecture.
Solana Name Service (SNS) [16]	Included to analyse high throughput and low-fee naming ecosystems scaling independently of Ethereum.

2.2 Decentralised Hosting and Content Delivery Selection

Decentralised Hosting is defined as a content-distribution architecture that utilises cryptographic hashing to identify and retrieve data based on its content (Content Addressing) rather than a physical server location (Location Addressing) [17]. For this evaluation, hosting platforms were identified based on their distinct methods of addressing mutability, peering incentives, and degrees of on-chain integration versus off-chain distributed hash tables (DHTs), as detailed in Table 2" to point to Table 2.

2.3 Decentralised Storage Networks Selection

Decentralised Data Storage provides the economic and cryptographic infrastructure required for data persistence, employing algorithmic incentive mechanisms to verify that nodes continuously store data [24]. The storage networks selected for this review were categorised by their distinct cryptographic persistence mechanisms, comparing temporary contract-based rental models against permanent archival solutions and ledger-based storage, as detailed in Table 3.

2.4 Practical Case Study Selection

To synthesise the theoretical analysis of decentralised web infrastructure, this review incorporates a practical case study demonstrating the functional integration of all three foundational pillars (Naming, Hosting, and Storage) into a cohesive architecture. The Massa DeWeb ecosystem was explicitly selected for this demonstration due to its unique embodiment of the "On-Chain Web" paradigm. Unlike conventional hybrid architectures (Web 2.5) that necessitate the orchestration of disparate, fragmented protocols – for instance, combining ENS (naming) with Filecoin (storage) and Fleek (middleware) – Massa natively integrates decentralised DNS (MNS), frontend hosting (serving HTML/JS), and data persistence directly within a single Layer 1 ledger state. This unified, self-contained architecture provides an ideal, highly controlled environment to empirically illustrate the deployment lifecycle of a fully serverless, trustless decentralised application without the confounding variables of cross-protocol interoperability.

Table 2. Selected Decentralised Hosting Protocols and Justification

Protocol	Selection Justification
IPFS [17]	The foundational standard for Content Identifiers (CIDs) and Merkle DAG-based peer-to-peer data distribution.
Swarm (EthSwarm) [18]	Provides an Ethereum-native alternative with unique 'postage stamp' economic incentives for data delivery.
Hypercore (Dat) [19]	Represents the evolution of append-only log structures for highly dynamic, mutable peer-to-peer data streams.

Internet Computer (ICP) [20]	Included for its novel 'canister' smart contract architecture, capable of directly serving HTTP requests at web speed.
Massa On-Chain [21]	Represents the frontier of embedding complete, immutable web assets directly into active ledger state keys.
Fluence [22]	Illustrates the decentralised compute layer, decoupling peer-to-peer execution from strict global consensus mechanisms.
Fleek [23]	Included as a representation of middleware abstraction layers that bridge complex backend Web3 hosting with Web2 developer UX.

3. State-of-the-Art Review

This section provides a comprehensive examination of the current landscape of decentralised web technologies. We conduct a detailed review of the leading protocols within each of the three foundational pillars of the decentralised web stack. This structured analysis aims to elucidate the diverse technical approaches, economic models, and trade-offs that characterise the state-of-the-art in this rapidly evolving ecosystem.

Table 3. Selected Decentralised Storage Networks and Justification

Protocol	Selection Justification
Filecoin [25]	The most prominent incentive layer for IPFS utilising novel Proof of Replication (PoRep) and Spacetime (PoSt).
Storj [26]	Selected for its enterprise-oriented architecture, leveraging erasure coding and an AWS S3-compatible gateway.
Sia [27]	Illustrates a purely trustless storage framework utilising on-chain file contracts without federated satellites.
Arweave [28]	Represents the permanent storage model (PermaWeb) driven by a distinct Blockweave graph and Proof of Access (PoA).
Massa (Ledger State) [21]	Included to evaluate the economic limitations and security advantages of tier-0 hot storage for critical web logic.
Web3.Storage [29]	Represents middleware aggregation protocols designed to streamline multi-network decentralised storage adoption.

The architecture of the decentralised web mirrors the structure of the traditional internet – comprising naming, routing, and persistence – but replaces central authorities with distributed

cryptographic primitives. We dissect these components as follows: Section 3.1 examines Decentralised DNS Protocols, contrasting overlay systems with alternative roots. Section 3.2 investigates Decentralised Hosting and Content Delivery, focusing on the shift from location-based to content-based addressing. Finally, Section 3.3 analyses Decentralised Storage Networks, categorising them into contract-based markets, permanent storage, and on-ledger solutions.

3.1 Decentralised DNS Protocols

The architectural landscape of Decentralised DNS fundamentally rearchitects the resolution of network identifiers, replacing the hierarchical governance of bodies such as ICANN with distributed ledger technology. By managing the mapping of human-readable identifiers (e.g., *alice.eth*) to machine-readable cryptographic addresses via immutable blockchain registries, these systems eliminate single points of failure and censorship. This ensures that domain ownership remains cryptographically verifiable and sovereign to the user [11].

Contemporary literature delineates a divergence in architectural methodologies, primarily distinguishing between Overlay systems and Alternative Root systems [9]. Overlay systems function as an application layer built upon established general-purpose blockchains. By utilising smart contracts, these protocols leverage the underlying security and consensus mechanisms of the parent chain to manage their namespace. Conversely, Alternative Root systems operate as sovereign blockchains with independent consensus mechanisms, aiming to decentralise the DNS root zone itself. The following subsections provide a comparative review of prominent protocols representing these diverse approaches.

3.1.1 Ethereum Name Service (ENS)

As the preminent protocol in this domain, ENS functions as a suite of smart contracts on the Ethereum blockchain. It maps human-readable names (e.g., *alice.eth*) to machine-readable identifiers. Recent analyses [10] highlight its governance model, wherein the ENS DAO dictates key parameters. ENS domains are ERC-721 compliant Non-Fungible Tokens (NFTs), facilitating seamless integration with the broader DeFi and NFT ecosystem. A critical design element is the rent model, which mandates annual renewal fees to deter domain squatting and maintain namespace vitality.

3.1.2 Namecoin

Historically significant as the inaugural implementation of decentralised DNS, Namecoin is a Bitcoin fork that introduced the *.bit* TLD. While it pioneered the resolution to Zooko's Triangle (creating names that are simultaneously secure, decentralised, and human-readable), its adoption has plateaued relative to smart-contract-based alternatives [11]. It utilises a UTXO-based model where names expire and necessitate renewal via transaction – a mechanism that has influenced subsequent protocols.

3.1.3 Handshake (HNS)

Handshake differentiates itself by attempting to decentralise the DNS root zone, rather than operating a TLD within a

specific blockchain ecosystem [12]. It employs a Vickrey auction mechanism for Top-Level Domains (TLDs) on its proprietary UTXO-based blockchain. The objective is to supplant the ICANN root file with a distributed ledger, enabling universal TLD ownership. This approach offers compatibility with existing DNS infrastructure through light clients and resolvers, aiming for a soft fork of the internet's naming infrastructure.

3.1.4 Unstoppable Domains (CNS/UNS)

Operating primarily on the Polygon network (an Ethereum Layer 2), Unstoppable Domains prioritises consumer accessibility and censorship resistance. A key differentiator identified in comparative studies [13] is its ownership model: domains are acquired via a one-time fee with no renewal obligations. While this mitigates the risk of domain loss due to lapsed payments, critics argue it exacerbates issues of domain squatting and namespace exhaustion.

3.1.5 Blockstack/Stacks (BNS)

The Bitcoin Name System (BNS) is constructed on the Stacks blockchain, which is anchored to Bitcoin. It introduces a distinct Proof-of-Transfer consensus mechanism. BNS names are intrinsically linked to decentralised identity (DID) standards, serving as a foundational layer for the Stacks ecosystem's user-owned internet [14].

3.1.6 Massa Name System (MNS)

Massa introduces a fully on-chain DNS solution integrated directly into its Layer 1 blockchain. Unlike overlay systems that map names to off-chain content hashes (such as IPFS), MNS maps *.massa* domains directly to smart contracts hosting the website's full logic and frontend assets on the Massa blockchain itself [15]. This On-Chain Web paradigm ensures that DNS resolution and content delivery are tightly coupled and equally decentralised. MNS domains are NFTs with a unique lifecycle that includes a 50% refundable domain release mechanism, incentivising the recycling of unused names [21].

3.1.7 Solana Name Service (SNS)

Solana Name Service (SNS) operates as a decentralised naming protocol on the Solana blockchain, providing human-readable names for wallet addresses and dApps. SNS leverages Solana's high throughput and low fees, making it suitable for scalable applications. Recent studies highlight its integration with Solana's DeFi and NFT ecosystems, but note that its adoption is still emerging compared to ENS [16].

3.1.8 Comparative Analysis of Decentralised DNS Protocols

Table 4 summarises the key architectural and economic distinctions between the discussed decentralised DNS protocols. The landscape is divided between overlay systems like ENS, Unstoppable Domains, and SNS, which leverage the security of existing blockchains, and sovereign chains like Handshake and Namecoin that attempt to redefine the root zone. Stacks (BNS) bridges these paradigms by anchoring to Bitcoin while operating a distinct DID ecosystem. ENS dominates due to its network effects and integration within the

Ethereum ecosystem, yet its rent-based model contrasts sharply with Unstoppable Domains' perpetual ownership model. While perpetual ownership appeals to consumers, it introduces long-term risks of namespace exhaustion and squatting. Alternatively, Solana Name Service (SNS) aims to resolve the cost friction of Ethereum by offering high-throughput, low-fee registrations natively tied to the Solana DeFi landscape.

Massa's MNS offers an alternative structural model with a 50% refundable domain release (burning the remaining 50% of the fee), designed to balance ownership security with economic incentives to recycle unused names. It also provides a mechanism for trading domains as NFTs and facilitating ownership transfers. While ENS remains the industry standard for identity, the future of decentralised DNS may depend on the success of alternative roots like Handshake in gaining browser adoption. Furthermore, the integration of DNS with on-chain hosting explores a fully decentralised web stack where the domain name and the referenced content operate within the same trust boundaries.

3.2 Decentralised Hosting and Content Delivery

The transition to decentralised hosting necessitates a fundamental departure from the traditional HTTP location-based model. Whereas conventional requests retrieve resources from designated physical server IP addresses (e.g., *192.168.1.1/file.jpg*), state-of-the-art decentralised protocols decouple information from its origin by identifying data solely through its cryptographic hash (Content Identifier or CID). This architectural shift allows resources to be retrieved from any participating node in the peer-to-peer network, significantly enhancing fault tolerance and reducing access latency through distributed caching [30].

However, this shift introduces a significant challenge regarding content updates: since the address is mathematically derived from the content itself, any modification to a file results in a new hash (Immutability). Consequently, decentralised hosting systems must implement mechanisms to map stable, mutable identifiers to these evolving immutable hashes. The following subsections examine how leading protocols address these challenges of addressing, routing, and mutability.

3.2.1 InterPlanetary File System (IPFS)

IPFS remains the preeminent protocol in this domain. It employs a Merkle Directed Acyclic Graph (DAG) to structure data, wherein each file is identified by a Content Identifier (CID). The CID is a self-describing hash that encapsulates the cryptographic algorithm used and the encoding format [17], [30]. To locate content, IPFS utilises a Distributed Hash Table (DHT) based on Kademlia. Data transfer is managed by Bitswap, a message-based protocol where peers exchange want-lists and data blocks.

To address the immutability challenge, IPFS provides InterPlanetary Name System (IPNS), which leverages public-key cryptography to create a mutable pointer. A user publishes a signed record pointing their stable Peer ID to the latest CID. Alternatively, DNSLink allows standard DNS domains to map to IPFS hashes, providing a bridge for Web 2.0 users. Despite

its resilience, IPFS does not guarantee persistence; content is garbage-collected if not explicitly pinned by a node.

3.2.2 Swarm (EthSwarm)

Swarm is engineered as the native storage and content delivery layer for the Ethereum ecosystem. Its core architecture is the Distributed Immutable Store of Chunks (DISC) [18]. Files are partitioned into 4KB chunks and distributed across the network using a Kademlia-based overlay. Swarm addresses updates through Single Owner Chunks (SOC), which allow a user to write data to a specific address derived from their Ethereum address and an identifier, effectively enabling mutable content atop immutable storage. Crucially, Swarm implements a stamps system where users attach postage (in BZZ tokens) to chunks, incentivising nodes to store and serve content.

3.2.3 Hypercore (Formerly Dat) and Pear

The Dat protocol, now evolved into the Hypercore protocol and the Pear runtime, offers a mutable, append-only log structure for peer-to-peer data sharing [19]. Unlike IPFS, which treats files as static blocks, Hypercore focuses on dynamic data streams. The public key of the log acts as the stable address, while new data blocks are appended and signed. This architecture natively supports versioning and real-time updates, rendering it highly suitable for dynamic applications. The recent Pear runtime leverages this to create serverless peer-to-peer applications.

3.2.4 Internet Computer (ICP)

The Internet Computer Protocol (ICP) reimagines the hosting stack by deploying canisters, advanced smart contracts that bundle both code and state (around 4GB of storage) [20]. Unlike Ethereum smart contracts which are limited in processing power and storage, canisters can serve HTTP requests directly to end users. This enables frontend interfaces to be hosted entirely on-chain, eliminating reliance on centralised cloud providers (AWS, Google Cloud). ICP achieves this through a chain key cryptography mechanism that splits execution across subnets, allowing the network to scale infinitely while maintaining web-speed latency for users.

3.2.5 Massa On-Chain

Massa takes a distinct approach by embedding web content directly into the blockchain's ledger state. Instead of referencing external hashes like IPFS, Massa smart contracts store HTML, CSS, and JS files within their datastore keys [21]. This allows the blockchain nodes to serve the frontend application directly, ensuring that the interface inherits the same security, immutability, and availability properties as the backend logic. This On-Chain Web paradigm mitigates the Web 2.5 vulnerability where decentralised protocols rely on centralised servers for their user interfaces.

3.2.6 Fluence

While protocols like IPFS and Swarm focus primarily on data availability, a comprehensive hosting architecture requires dynamic execution capabilities. Fluence addresses this by providing a decentralised, peer-to-peer compute paradigm that

decouples execution from mandatory global consensus [22]. It leverages Aqua, a distributed choreography language, and Marine, a WebAssembly (Wasm) execution environment, to coordinate serverless computations across a network of independent node providers. Instead of forcing every node to compute the same state (as in Ethereum smart contracts), Fluence allows developers to construct verifiable peer-to-peer service agreements. This architecture enables developers to host dynamic backend infrastructure and logic without relying on centralised cloud functions, effectively bridging the gap between static content delivery and complex off-chain execution.

3.2.7 Fleek

Developing directly atop native content-addressed networks and decentralised naming systems frequently imposes a steep learning curve. Fleek addresses this by functioning as an abstraction layer and a decentralised edge computing platform. It bridges complex Web3 protocols, such as IPFS, Filecoin, Arweave, and ENS, with traditional Web2 workflows [23]. Fleek mirrors the developer experience of platforms like Vercel or Netlify by providing automated Continuous Integration and Continuous Deployment (CI/CD) pipelines. When a developer pushes code to a Git repository, Fleek automatically builds the application, pins the static assets to IPFS to generate a CID, and can seamlessly update an associated IPNS, ENS, or DNSLink record to point to the new hashing structure. By abstracting the intricacies of content addressing, routing, and mutability handling, Fleek serves as critical middleware facilitating the widespread adoption of decentralised frontend hosting.

3.2.8 Comparative Analysis of Decentralised Hosting

Table 5 provides a comparison highlighting the shift from location-based addressing to content-based addressing and the varying degrees of on-chain integration.

IPFS excels in content addressing and deduplication but suffers from the persistence problem, often requiring centralised pinning services that reintroduce trust. Swarm and ICP attempt to solve this through economic incentives and canister models, respectively. Fluence introduces an alternative paradigm by wrapping hosting and compute into configurable peer-to-peer service agreements, decoupling execution from a single global consensus mechanism. Fleek provides an essential abstraction layer, acting as middleware to streamline the deployment of frontend applications across these underlying decentralised storage networks (like IPFS and Filecoin), effectively providing a Web2-like developer experience. However, ICP's architecture, while powerful, is complex and relies on a unique consensus mechanism. Massa's approach is structurally distinct, treating web content as an executable ledger state. This model ensures uniform availability and censorship resistance but is constrained by the high economic cost of on-chain storage. Consequently, it is optimal for critical frontend logic (e.g., HTML, JS) rather than heavy media hosting algorithms.

The ideal decentralised hosting solution likely involves a hybrid approach. High-value, critical application interfaces (the

frontend) are best served by fully on-chain solutions like Massa or ICP to prevent censorship and hijacking, potentially augmented by computation layers like Fluence for backend logic. Meanwhile, bulk static assets (images, videos) are more efficiently handled by off-chain networks like IPFS or Swarm. The convergence of these layers will define the next generation of dApp architecture.

3.3 Decentralised Storage Networks

While content-addressed networks provide robust routing, guaranteeing long-term data persistence requires dedicated economic layers. For instance, although protocols like IPFS excel at content distribution, they lack an intrinsic incentive mechanism for durability; nodes only retain data they explicitly pin or have recently accessed, leaving unpinned files vulnerable to garbage collection.

State-of-the-art Decentralised Storage Networks (DSNs) systematically resolve this deficiency by employing algorithmic incentive mechanisms – such as Proof of Spacetime or Proof of Replication – to replace trust in centralised cloud providers with verifiable cryptographic proofs [24]. By establishing permissionless marketplaces for redundant, censorship-resistant data archival, these networks and associated ecosystems are broadly categorised into Contract-based markets, Permanent storage solutions, Ledger-based storage, and aggregating Storage Middleware.

3.3.1 Contract-Based Storage Markets

Contract-based networks operate on a rental model similar to traditional cloud storage, but in a permissionless marketplace.

A. Filecoin

Built as the incentive layer for IPFS, Filecoin creates a marketplace where clients pay storage providers (miners) to store data for a fixed duration. Its security relies on two novel cryptographic proofs [25, 31]:

- (i) Proof of Replication (PoRep): A one-time proof demonstrating that a miner has received a specific piece of data and has encoded it in a way that is unique to their physical storage. This prevents Sybil attacks where a miner pretends to store multiple copies of the same data on the same disk.
- (ii) Proof of Spacetime (PoSt): A continuous proof that verifies the miner is still storing the data over the duration of the contract. Miners must submit these proofs regularly (Window PoSt) to the blockchain. Failure to do so results in slashing, where the miner loses a portion of their staked collateral.

B. Storj

Storj distinguishes itself with an architecture that prioritises compatibility and performance. Unlike Filecoin's blockchain-heavy approach, Storj uses a network of Satellites to manage metadata, node reputation, and payments, while the actual data is stored on Storage Nodes [26].

- (i) Erasure Coding: Instead of full replication, Storj splits files into 80 pieces using Reed-Solomon erasure coding. Only 29

pieces are needed to reconstruct the file. This ensures high availability even if many nodes go offline simultaneously.

(ii) S3 Compatibility: Storj offers a native S3-compatible gateway, allowing developers to switch from AWS S3 to Storj with minimal code changes, a significant factor in its adoption for enterprise backups and media streaming.

C. Sia

Sia focuses on a fully trustless implementation using smart contracts on its own blockchain. A File Contract is formed between a renter and a host, locking funds in a payment channel [27]. The host submits Storage Proofs (Merkle proofs) to the blockchain to unlock their payment. Sia employs sharding (splitting files into 30 segments, requiring 10 to recover) similar to Storj but manages the logic entirely on-chain without the federated Satellite components.

3.3.2 Permanent Storage

Arweave introduces a fundamentally different economic and technical model aimed at permanent data preservation, often referred to as the Permaweb [28]. Unlike a linear blockchain, Arweave utilises a Blockweave structure where each new block refers to the previous block and a randomly selected historical recall block. This architecture supports its Proof of Access (PoA) consensus mechanism, which requires miners to prove possession of the Recall Block to mine new blocks, thereby incentivising the storage of the network's entire history and ensuring data redundancy [32].

3.3.3 Ledger-Based Storage

While not a dedicated storage network like Filecoin or Arweave, Massa offers a unique storage capability via its ledger state. In Massa, data is stored directly in the active state of the blockchain, ensuring maximum availability and direct accessibility by smart contracts [21]. The files are stored on Massa permanently by locking a fixed amount of coins, and there is no recurring fee. The locked amount is fully refunded if the files are deleted, allowing the space to be reused. However, this storage is scarce and limited and is designed for critical web assets (HTML/JS for the On-Chain Web) and smart contract state, rather than bulk file storage. This creates a tiered architecture: Massa for the critical application logic and interface, and networks like Arweave or IPFS for heavy media assets.

3.3.4 Storage Middleware

Operating directly on raw decentralised storage protocols often requires managing complex cryptographic proofs, tokenomics, and data pinning infrastructure. Web3.Storage addresses this complexity by functioning as an abstraction layer and middleware aggregation protocol. It provides a streamlined interface that automatically routes data across IPFS for fast, location-independent retrieval, and Filecoin for provable, long-term cryptographic persistence [29]. By utilising technologies such as User-Controlled Authorization Networks (UCAN) for delegated access control, Web3.Storage allows developers to interact with disparate decentralised storage networks through a unified, Web2-style API. This abstraction removes the friction of acquiring native protocol tokens or running dedicated nodes, serving as a

critical bridge that accelerates enterprise and developer adoption of decentralised data infrastructure.

3.3.5 Comparative Analysis of Decentralised Storage Networks

Table 6 contrasts the leading decentralised storage networks and middleware based on their consensus mechanisms, economic models, and intended use cases.

The market is clearly segmented between rental and permanent storage. Filecoin, Sia, and Storj compete directly with AWS S3, offering lower costs through a rental model. Storj stands out for its S3 compatibility and performance focus, making it the most enterprise-ready. Conversely, Arweave creates a new market category for permanent data, essential for NFT metadata and historical records. Massa occupies a niche for hot storage of executable web code, which is too expensive for bulk data but critical for application integrity. Lastly, Web3.Storage represents the essential middleware layer, augmenting underlying networks by abstracting their complexity for developers.

Decentralised storage has matured significantly, moving from theoretical proofs to production-ready networks. The choice of protocol depends heavily on the use case: Storj for performance-critical cloud object storage, Filecoin for massive archival datasets, and Arweave for immutable cultural or legal records. Massa complements this ecosystem by addressing the specific need for hot, executable storage directly on the ledger, bridging the gap between static data and dynamic application logic. Crucially, the emergence of storage middleware such as Web3.Storage provides the necessary abstraction layer that automatically routes data to the most appropriate underlying network based on cost, speed, and persistence requirements, effectively masking the fragmented nature of the emerging storage stack.

4. On-Chain Web Architectures: A Comparative Case Study and Practical Implementation

To illustrate the practical implications and trade-offs of fully on-chain web architectures, this section first presents a comparative case study between the Internet Computer Protocol (ICP) and Massa. Following this comparative analysis, we provide a detailed practical examination of the Massa DeWeb architecture to demonstrate the deployment lifecycle of a fully serverless, trustless application.

4.1 Comparative Case Study: ICP vs Massa

Both protocols enable the hosting of frontend logic and assets directly on-chain, eliminating reliance on centralised cloud providers, but they differ significantly in technical design, economic models, and scalability.

4.1.1 Architectural Overview

ICP employs canisters – advanced smart contracts that bundle code and state – allowing web applications to be served directly from the blockchain. Canisters are funded with cycles, and storage costs scale with usage. ICP's consensus and subnet architecture enable horizontal scaling but introduce complexity and unique trust assumptions.

Massa embeds web content directly into smart contract storage, with domains resolved via the native Massa Name

System (MNS). The architecture is structurally simpler, embedding HTML/JS directly into the ledger's active datastore keys, but storage is fundamentally limited and best suited for critical application logic.

4.1.2 Storage Cost and Scalability

ICP's storage cost is determined by the cycles consumed per GB per year, using dynamic pricing based on network load. The protocol is designed for massive scalability, but predictable costs can fluctuate, becoming significant for large-scale applications over time.

Conversely, Massa's storage is paid via a deterministic deposit model. The storage cost is calculated based on the volume of data persisted on the ledger using the following equation:

$$\text{Storage Cost} = 0.001x(B + K + V) \text{ MAS}$$

where B represents the fixed overhead cost for creating a new ledger entry, K is the number of bytes of the key, and V is the number of bytes of the value (data) to be stored. Unlike traditional "rent" models (such as ICP's cycle burn or Filecoin's recurring fees), Massa locks the MAS tokens as a deposit. If the data is deleted, the storage space is freed, and the initial deposit is fully refunded to the user.

Quantitatively, persisting a 2MB frontend bundle on-chain using Massa is economically feasible (typically requiring a few dollars in native token lockup), making it highly suitable for high-integrity assets like DeFi interfaces. However, scaling either network to gigabytes of data for consumer-grade media hosting remains prohibitively expensive compared to off-chain networks like IPFS or Arweave.

4.1.3 Decentralisation, Trust, and Limitations

Both networks offer strong guarantees of availability and censorship resistance. ICP achieves decentralisation through a network of independent node providers and an algorithmic governance system. However, its reliance on boundary nodes to route HTTP traffic introduces unique trust vectors, and the protocol's sheer complexity results in a high barrier to entry for node operators.

Massa's approach is more minimal, allowing all standard nodes to serve content directly, which bolsters censorship resistance at the base layer. However, Massa's ecosystem is currently less mature, and its network is smaller. Ultimately, ICP provides greater scalability and flexibility at the cost of architectural complexity, while Massa offers direct simplicity constrained by hard ledger storage limits.

4.2 Practical Implementation: The Massa DeWeb Architecture and dApp Deployment

To instantiate the theoretical models discussed above, this section provides a practical examination of deploying a fully decentralised web application. We first outline the underlying Massa DeWeb architecture and subsequently detail the lifecycle of the *Massa Domain Market* dApp.

4.2.1 Massa DeWeb Architecture

The Massa DeWeb architecture diverges fundamentally from traditional web hosting by unifying the entire application stack

– DNS, hosting, and storage – within the blockchain's ledger. This ecosystem relies on three foundational pillars:

A. Massa Name System (MNS)

MNS is a decentralised naming service resolving the *.massa* TLD directly via the blockchain [33]. Domains are structured as tradable NFTs. To mitigate squatting, the protocol includes a distinct mechanism where owners can return unneeded domain, thereby reclaiming 50% of the initial reservation cost [21] while the remainder is burned.

B. On-Chain Storage

Website assets (HTML, CSS, JavaScript, and images) are persisted directly within the storage keys of a smart contract [21]. This ensures that the frontend interface possesses the same immutability and censorship resistance as the backend logic. This mechanism relies on the deposit-based economic model established in the comparative analysis, preventing ledger bloat while ensuring data permanence as long as the deposit remains locked.

C. Dual Resolution Strategy

Access to these decentralised sites is facilitated through a flexible resolution strategy. The primary, trustless method utilises a local DNS client (e.g., Massa Station) that intercepts requests (like *domains.massa*), queries the blockchain state directly, and serves assets locally [34]. To lower the barrier to entry, third-party public gateways (e.g., <https://domains.massa.host>) allow standard web browsers to access on-chain content without local infrastructure, albeit by reintroducing a degree of trust in the gateway operator.

Figure 1 illustrates the high-level architecture of the Massa DeWeb ecosystem, delineating the interaction flow between the user, the resolution layers, and the blockchain ledger. The diagram highlights the dual access mechanism: the Trustless Resolution, where a local Massa Station node directly queries the blockchain state for maximum security and privacy, and the Gateway Resolution, which bridges standard Web 2.0 browsers to the decentralised network via a proxy. Central to this architecture is the Massa Blockchain, which houses both the MNS Registry (mapping *.massa* domains to addresses) and the Smart Contract Storage (containing the actual website assets), pointing to another smart contract that is used for data storage. This visual representation underscores how the domain resolution, content retrieval, and data storage processes are unified within a single ledger, contrasting with hybrid models that rely on external storage networks.

4.2.2 Implementation: Massa Domain Market dApp

To illustrate the practical deployment lifecycle within this architecture, we examine the *Massa Domain Market*, a fully decentralised application enabling users to trade *.massa* domains and view marketplace statistics.

Establishing a DeWeb presence commences with securing a *.massa* domain via the MNS smart contract [33]. For this application, the domain *domains.massa* was purchased by a Massa Community member, transferred to the Author, and

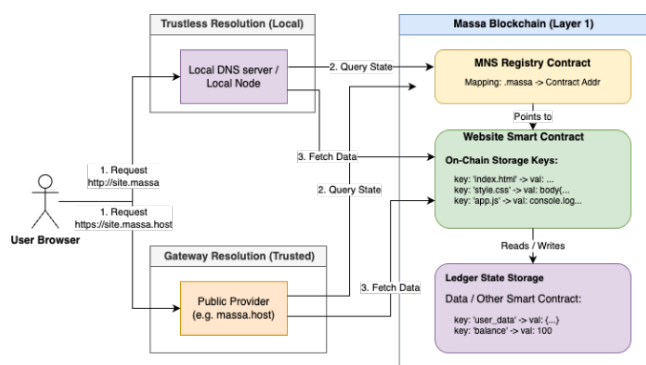


Figure 1. Massa DeWeb architecture

mapped to the application's specific smart contract address immutably in the on-chain registry.

The application is composed of two integrated components:

A. Frontend

The user interface is engineered using React, Vite, and TypeScript. Interaction with the blockchain is mediated by the *massa-web3* library, enabling the browser to communicate directly with Massa nodes via JSON-RPC, effectively eliminating the traditional centralised backend server. The user workflow for selling a domain is streamlined as follows: the user connects their wallet to retrieve a list of owned addresses (Figure 2), selects the specific domain to be listed by clicking the Sell button, opens a selling window (Figure 3), and finally signs the transaction to confirm the listing (Figure 4).

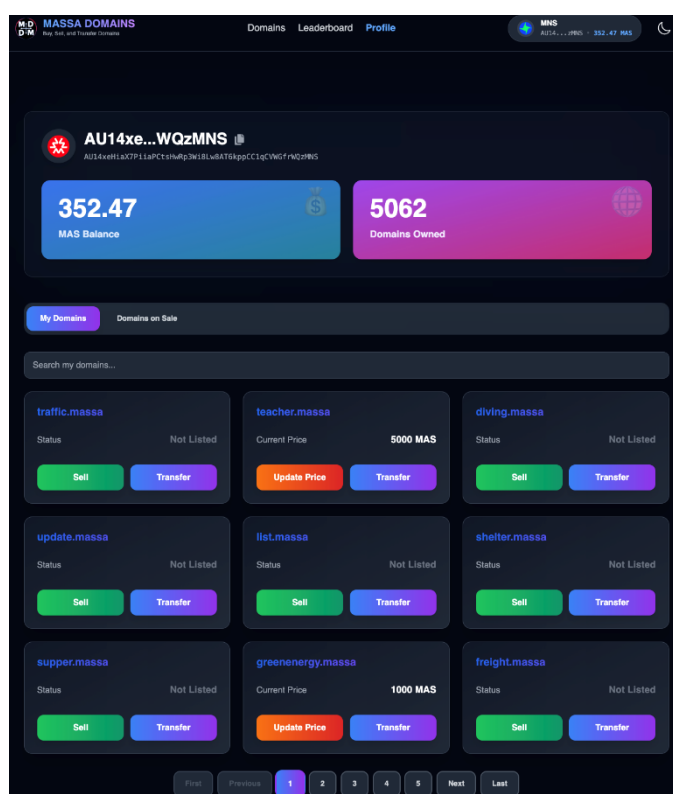


Figure 2. Sample screenshot of listing all domains owned by a given address.

B. Backend (Smart Contract)

The application logic and state are encapsulated entirely within a smart contract deployed on the Massa blockchain. This contract functions as an immutable backend, managing two primary categories of data:

(i) Listing Data – The contract persistently stores the essential details for every domain listed for sale. It maps the domain identifier to the owner's address and the designated sale price. To submit a sale offer, the user signs a transaction invoking the *sellDomain* function with specific parameters: the owner's address (e.g., *AU14xeHia...*), the domain identification number

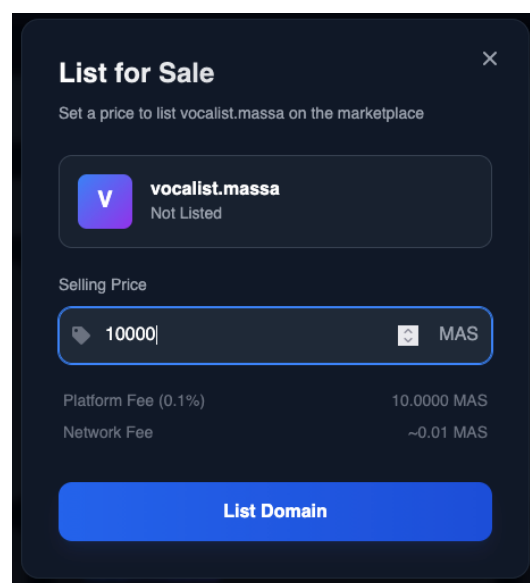


Figure 3. The sell window showing the domain to be sold and the sell price and associated fee.

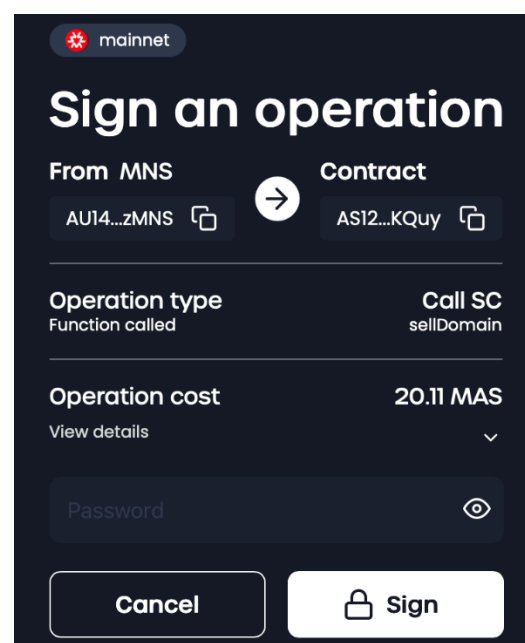


Figure 4. The wallet signing window indicating that you must own the domain to sell it by signing the transaction with your wallet's private key.

(e.g., 345), and the sale price (e.g., 1200 MAS). This data is serialised and stored on the blockchain as a key-value pair.

(ii) Statistics and Housekeeping – The contract maintains aggregate metrics for the marketplace to populate the dashboard (Figure 5). These metrics include the total number of transactions (sales and purchases), overall sales volume, total fees collected, and historical price data, such as the lowest and highest recorded sale prices.

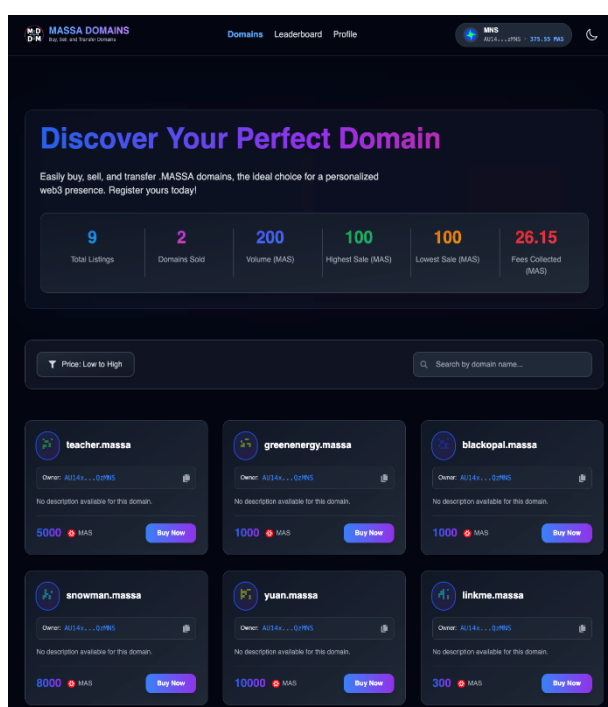


Figure 5. Landing page showing some statistics and list of domains on sale.

4.2.3 Deployment and Gateway Setup

Deployment is done by uploading the compiled artefacts (HTML, CSS, JS) directly to the smart contract's storage via the DeWeb CLI. This process involves the following steps:

Step 1. Build Generation: The developer initiates a standard build process (e.g., `npm run build`) to generate the necessary static assets, including `index.html`, `stylesheets`, `JavaScript bundles`, and `media files`.

Step 2. On-Chain Upload: The `@massalabs/deweb-cli` tool is utilised to upload the application. The command below fragments the files and transmits them as a series of transactions, persisting the data within the smart contract's datastore keys [21]:

```
px @massalabs/deweb-cli upload -w <wallet_private_key> -d
domains.massa <path_to_build_folder>
```

Upon completion, the CLI outputs the specific smart contract address where the web application is hosted. To lower the technical barrier to entry, developers may alternatively utilise the Simple Dweb Uploader tool [35]. This web-based interface

provides a streamlined, graphical method for uploading applications without requiring command-line interaction.

Step 3. Domain Association: To render the application accessible via a human-readable name, the deployed smart contract address must be associated with a Massa domain. This involves updating the target address of the domain (e.g., `domains.massa`) in the registry to point to the new application contract.

Step 4. Local Server Setup: For development and testing purposes, setting up a local server is recommended to serve the DeWeb application locally. Developers can download the massa deweb server binary from the official GitHub releases page [36]. Configuration involves creating a `config.yaml` file to specify the node URL (e.g., `https://mainnet.massa.net/api/v2`) and the listening port. Once running, the server acts as a local gateway, allowing the developer to browse their on-chain application via `http://localhost:8080` (or the configured port).

Step 5. Provider Setup: To ensure the application is accessible to users without a local node, a public gateway must be configured. This involves setting up a third-party provider instance that acts as a bridge between standard web browsers and the Massa blockchain. Technically, this requires deploying the massa deweb server [36] on a public-facing server (e.g., a VPS). The configuration necessitates setting up a wildcard DNS record (e.g., `*.massa.host`) pointing to the server's IP address. Additionally, a reverse proxy such as Nginx is typically employed to handle SSL/TLS termination and route incoming HTTP requests to the deweb server's listening port. The server's `config.yaml` must be updated to define the domain parameter, ensuring that requests like `https://domains.massa.host` are correctly resolved to the `domains.massa` smart contract address on the ledger.

4.2.4 Updates, Immutability, and Limitations

Website updates do not necessitate domain migration. The developer rebuilds the application and re-executes the CLI upload command, which manages differential updates to the smart contract's keys. For applications requiring absolute permanence (e.g., DeFi interfaces), Massa introduces an *Immutability* feature [37]. Developers can irrevocably lock the website's smart contract, preventing any future modifications to the code or stored assets, ensuring trustlessness.

While Massa DeWeb offers a robust decentralised architecture, users must remain cognisant of privacy limitations contingent on their access method [38]. The most secure configuration involves running a local Massa node, ensuring zero third-party involvement. Conversely, relying on third-party public gateways introduces privacy risks, such as browsing activity tracking and potential *last mile* content manipulation, highlighting a persisting trade-off between absolute decentralisation and user convenience.

5. Conclusion

The transition from the centralised Web 2.0 to the decentralised Web 3.0 is propelled by the imperative for digital sovereignty, censorship resistance, and the Read-Write-Own paradigm. This review has elucidated the rapid evolution of the ecosystem's three core pillars: DNS, Hosting, and Storage,

while highlighting the critical vulnerabilities inherent in current Web 2.5 hybrid architectures.

In the domain of Decentralised DNS, the landscape is divided between overlay networks like ENS, which leverage existing blockchain security, and sovereign roots like Handshake that aim to decentralise the internet's root zone. Similarly, Decentralised Hosting and Storage solutions have matured into distinct categories, with IPFS and Filecoin dominating contract-based storage, Arweave pioneering permanent data preservation, and protocols like Swarm and Hypercore enabling dynamic peer-to-peer applications.

Despite these advancements, the reliance on centralised gateways for frontend delivery remains a significant bottleneck. The case study of the Massa DeWeb demonstrates a significant evolution in addressing this issue. By enabling a fully on-chain web where DNS, storage, and logic are tightly coupled within the blockchain ledger, Massa eliminates the dependency on external intermediaries. The successful development and deployment of the Massa Domain marketplace application [39] illustrates that a fully decentralised, user-owned web is practically achievable. As these technologies converge, they promise to reshape the digital landscape, moving beyond hybrid compromises to a truly serverless and sovereign internet.

Competing Interests:

The author declares the following competing interests: The Massa Domain Market application described in the case study was developed and deployed by the author, who also received the transfer of the domains.massa domain from a Massa community member. The author has no employment, advisory, or financial relationship with the Massa protocol team, but acknowledges reputational interest in the success of the application.

Ethical approval:

Not applicable.

Author's contribution:

The Author designed and coordinated this research and prepared the manuscript in its entirety.

Funding:

None declared.

Disclosure:

The author used artificial intelligence (AI) tools for limited language editing, structural refinement, and formatting assistance during the preparation of this manuscript. All research, analysis, interpretations, and conclusions are the sole responsibility of the author.

Acknowledgements:

The author sincerely thanks colleagues for their valuable insights and constructive discussions, and acknowledges the blockchain community for its open-source contributions and collaborative efforts that have advanced decentralised web technologies.

References:

- [1] T. O'Reilly, "What is Web 2.0: Design patterns and business models for the next generation of software", *Communications and Strategies*, no. 65, pp. 17–37, 2007.
- [2] S. Zuboff, "Big other: Surveillance capitalism and the prospects of an information civilization", *Journal of Information Technology*, vol. 30, no. 1, pp. 75–89, 2015. (for Web 2.0 platform centralisation)
- [3] H. Gawel, "Web 2.5 as a safe shift from Web 2.0 to Web 3.0: A definition of Web 2.5 in informatological approach", Cambridge Open Engage, 2022. doi:10.33774/coe-2022-sj2w3.
- [4] Rekt News, "BadgerDAO – Rekt", Rekt.news, Dec. 02, 2021. [Online]. Available: <https://rekt.news/badger-rekt/>.
- [5] Rekt News, "Curve finance – Rekt", Rekt.news, Aug. 09, 2022. [Online]. Available: <https://rekt.news/curve-finance-rekt/>.
- [6] A. V. Sambra et al., "Solid: A platform for decentralized social applications based on linked data", 2016. [Online]. Available: <https://api.semanticscholar.org/CorpusID:49564404>. [Accessed: May 7, 2026].
- [7] C. Dixon, *Read Write Own: Building the Next Era of the Internet*. New York: Random House, 2024.
- [8] B. Kitchenham and S. Charters, Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report, 2007.
- [9] K. Toyoda, P. T. Mathiopoulos, and A. N. Ota, "A survey of blockchain-based internet services: Domain name system, web hosting, and content delivery", *IEEE Access*, vol. 9, pp. 156071–156091, 2021.
- [10] Ethereum, "Ethereum name service", [Online]. Available: <https://docs.ens.domains/learn/protocol/> [Accessed: May 7, 2026].
- [11] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of namecoin", Proceedings of the 14th Workshop on the Economics of Information Security (WEIS), 2015.
- [12] J. Poon and T. Dryja, "Handshake: A decentralized certificate authority and naming", Handshake Whitepaper, 2020.
- [13] Unstoppable Domains, "Unstoppable domains architecture", 2025. [Online]. Available: <https://unstoppabledomains.com/learn> [Accessed: May 7, 2026].
- [14] M. Ali et al., "Trust-to-trust design of a new internet", Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI '19), 2019.
- [15] Massa Labs, "Enabling the on-chain web: The Massa decentralized DNS", Massa Labs, 2023. [Online]. Available: <https://massalabs.medium.com/enabling-the-on-chain-web-the-massa-decentralized-dns-890476865074>. [Accessed: Jan. 3, 2026].
- [16] A. Yakovenko, "Solana: A new architecture for a high performance blockchain", Whitepaper, 2018. [Online]. Available: <https://solana.com/solana-whitepaper.pdf> [Accessed: May 7, 2026].
- [17] J. Benet, "IPFS – Content addressed, versioned, P2P file system", 2014 [Online]. Available: <https://arxiv.org/abs/1407.3561> [Accessed: May 7, 2026].
- [18] V. Tron, "The book of swarm", Swarm Foundation, 2021. [Online]. Available: <https://papers.ethswarm.org/p/book-of-swarm/> [Accessed: May 7, 2026].
- [19] Holepunch, "Pear runtime documentation", 2025. [Online]. Available: <https://docs.pears.com/>. [Accessed: May 7, 2026].
- [20] DFINITY Foundation, "The internet computer in a nutshell", 2025. [Online]. Available: <https://internet>

- computer.org/presentations/Internet%20Computer%20in%20a%20nutshell%20EN.pdf [Accessed: May 7, 2026].
- [21] Massa Labs, “Massa DeWeb documentation”, 2025. [Online]. Available: <https://docs.massa.net/docs/deweb/home>. [Accessed: May 7, 2026].
- [22] Fluence Labs, “Fluence project”. [Online]. Available: <https://github.com/fluencelabs> [Accessed: May 7, 2026].
- [23] Fleek, “Fleek network”. [Online]. Available: <https://github.com/fleek-network/fleek-network-docs> [Accessed: May 7, 2026].
- [24] Q. Wang et al., “A comprehensive survey on decentralized storage systems”, *IEEE Access*, vol. 11, pp. 12345–12367, 2023.
- [25] Protocol Labs, “Filecoin: A decentralized storage network”, Technical Report, 2020. [Online]. Available: <https://filecoin.io/filecoin.pdf> [Accessed: Jan. 3, 2026].
- [26] Storj Labs, “Storj v3: A decentralized cloud storage network”, Whitepaper, 2019. [Online]. Available: <https://www.storj.io/storjv3.pdf> [Accessed: May 7, 2026].
- [27] D. Vorick et al., “Sia: Simple decentralized storage”, Nebulous Labs, 2020. [Online]. Available: <https://www.securities.io/wp-content/uploads/2022/04/sia.pdf> [Accessed: May 7, 2026].
- [28] S. Williams et al., “Arweave: The permanent information storage protocol”, Arweave Yellow Paper, 2020. [Online]. Available: <https://www.arweave.org/files/arweave-lightpaper.pdf> [Accessed: May 7, 2026].
- [29] Web3.Storage, “Web3.Storage: Middleware for decentralized storage networks”, 2025. [Online]. Available: <https://web3.storage/docs/> [Accessed: May 7, 2026].
- [30] D. Trautwein et al., “Design and evaluation of IPFS: A storage layer for the decentralized web”, Proceedings of the ACM SIGCOMM 2022 Conference, 2022, pp. 739–752.
- [31] J. Benet, B. Fisch, and N. Greco, “Proof of replication”, Protocol Labs, 2018. [Online]. Available: <https://filecoin.io/proof-of-replication.pdf> [Accessed: May 7, 2026].
- [32] S. Williams et al., “Arweave: A protocol for economically sustainable information permanence”, Yellow Paper. [Online]. Available: <https://www.arweave.org/yellow-paper.pdf> [Accessed: May 7, 2026].
- [33] Massa Labs, “Massa Name System (MNS) App”, 2025. [Online]. Available: <https://mns.massa.net/>. [Accessed: Jan. 5, 2026].
- [34] Massa Labs, “Massa station documentation”, 2025. [Online]. Available: <https://docs.massa.net/docs/massa-station/home>. [Accessed: Jan. 3, 2026].
- [35] Massa Labs, “Simple Dweb uploader”, 2025. [Online]. Available: <https://uploader.massa.net/>. [Accessed: May 7, 2026].
- [36] Massa Labs, “Massa DeWeb Server”, 2025. [Online]. Available: <https://github.com/massalabs/DeWeb/>. [Accessed: May 7, 2026].
- [37] Massa Labs, “Massa smart contract immutability”, 2025. [Online]. Available: <https://docs.massa.net/docs/smart-contracts/immutability>. [Accessed: May 7, 2026].
- [38] Massa Labs, “Massa DeWeb privacy and security”, 2025. [Online]. Available: <https://docs.massa.net/docs/deweb/getting-started/privacy>. [Accessed: May 7, 2026].
- [39] Nafsi Labs, “Massa domain market”, 2025. [Online]. Available: <https://domains.massa.host>. [Accessed: May 7, 2026].

Table 4. Comparisons of Decentralised DNS Protocols

Feature	Ethereum Name Service (ENS)	Namecoin	Handshake (HNS)	Unstoppable Domains	Stacks (BNS)	Massa (MNS)	Solana Name Service (SNS)
Architecture	Ethereum Smart Contracts (L1)	Bitcoin Fork (Merge-mined)	Custom UTXO Blockchain	Polygon Smart Contracts (L2)	Stacks Blockchain (Bitcoin L2)	Massa Smart Contracts (L1)	Solana Smart Contracts (L1)
Name Allocation	Instant Registration / Auction	First-Come-First-Served (FCFS)	Vickrey Auction (for TLDs)	FCFS/Premium Tiering	Bidding/Auction	FCFS/Auction	FCFS/Auction
Ownership Model	Annual Renewal Fees (Rent)	Expiration (Update Required)	Biennial Renewal (Heartbeat)	One-time Purchase (Perpetual)	Annual Renewal Fees	One-time + Refundable Release	One-time Space Rent (Perpetual)
Resolution	Smart Contract Registry	Local/Browser Extension	Decentralised Root Zone	Smart Contract Registry	BNS Smart Contracts	Native Browser/Station	Smart Contract Registry
Key Features	ERC-721 NFT, High Integrability	First Implementation, Censorship-Resistant	Root Zone Decentralisation	No Gas Fees (Polygon), Easy UX	Bitcoin Security, DID Integration	Fully On-Chain Web, Recyclable	High Throughput, Low Fees
Limitations	Gas Fees (L1), Rent Model	Low Adoption, Squatting	Requires Resolver/Browser Support	Squatting Risk (No Expiry), Centralised Launch	Network Latency (Bitcoin Block Times)	Ecosystem Maturity	Ecosystem Specific

Table 5. Comparisons of Web and Decentralised Hosting Architectures

Feature	Traditional Web (HTTP)	IPFS	Swarm (EthSwarm)	Internet Computer (ICP)	Massa (On-Chain)	Fluence (Compute)	Fleek (Middleware)
Addressing	Location (IP/URL)	Content (CID/Hash)	Content (Chunk Hash)	Canister ID (Principal)	Domain/Contract Address	Service ID (Peer-to-Peer)	URL/DNSLink
Storage Location	Centralised Servers	P2P Nodes (Off-chain)	P2P Disc (Off-chain)	On-Chain (Canister State)	On-Chain (Ledger State)	P2P Compute Network	IPFS/Arweave/Filecoin
Persistence	Server Owner Dependent	No Guarantee (Pinning req.)	Incentivised (Postage Stamps)	Guaranteed (Cycle Balance)	Guaranteed (Ledger State)	Provider Agreements	Pinning Service

Latency	Low (Optimised CDNs)	Variable (DHT Lookup)	Variable (Kademlia Routing)	Low (Web Speed Consensus)	Low (Direct State Read)	Variable (P2P Execution)	Low (CDN Backed)
Dependency	DNS + Hosting Provider	Pinning Service/Gateway	BZZ Token/Gateway	Cycles/Boundary Nodes	Massa Node/Station	Compute Providers	Fleek Platform

Table 6. Comparisons of Decentralised Storage Networks

Feature	Filecoin	Arweave	Storj	Sia	Massa (Ledger State)	Web3.Storage (Middleware)
Consensus	Expected Consensus (EC)	Proof of Access (PoA)	Proof of Work (Eth/Payment)	Proof of Work (Blake2b)	Proof of Stake (Blockclique)	Protocol Agnostic/Delegated
Data Structure	IPFS DAG/Sector Chains	Blockweave (Graph)	Erasure Coded Shards	Erasure Coded Shards	Ledger State (Key-Value)	Abstraction API over IPFS/Filecoin
Economic Model	Rent (Pay-per-time)	Endowment (One-time pay)	Rent (Pay-per-usage)	Rent (Payment Channels)	Storage Rent/Byte Cost	Abstracted/Service Tier
Persistence	Contract Duration	Permanent (>200 years)	Contract Duration	Contract Duration	Indefinite (while funded)	Managed (via Filecoin)
Primary Use Case	Cold Storage/Archives	History/Immutable Web	Cloud Object Storage (S3)	Private Cloud Backup	Hosting/Smart Contracts	Unified Developer API/Aggregation