

Protocol Foundations 002: Addressing

Mario Havel and Tim Beiko

https://ethereum.org/.../content//Ethereum_Whitepaper_Buterin_2014.pdf

Protocol or scheme Domain Path with the directory and the file name

Much like we assign addresses to buildings in the physical world, computers must keep track of the location for every piece of data they store. Similarly to physical constructions, smaller building blocks, called bits, are combined to create higher-level structures such as files or web pages. Unlike buildings, though, bits can be easily moved or copied. This means computers must use different addressing schemes for different use cases and ideally navigate frictionlessly between them.

At the lowest level, each bit has its own address in a computer's memory. The values of all variables in use are stored temporarily in physical parts of the device's hardware and applications access them based on these physical addresses. Only the hardware itself can process this form of addressing. Rarely, it gets exposed to programmers struggling to fix bugs. Most computer addresses are not legible to humans and only serve the machine's internal logic. Still, certain addresses need to be accessible to people and that's why computer addresses are often abstracted to **human-readable formats**.

Human-readable computer addresses usually correspond to higher levels of abstractions, such as files or web pages. Each file stored on a disk has its own address, which we commonly refer to as a **path**. The physical location on the disk is abstracted to a readable path in a form such as **/user/folder/doc.txt**. In a file path, slashes denote different directories, or grouping of files, starting from the root of the file system up until the name of the

file. This organization creates a tree-like hierarchy and allows users to organize files in directories. Files are accessed based on routes through directories. To analogize to the physical world again, this is akin to giving turn-by-turn directions to someone to get them to a specific location, rather than giving them an address itself.

An equivalent form of address is used when browsing data on other computers. On the internet, users mostly access files using the standardized format of their paths, known as **uniform resource locators: URLs**. In a form easily understandable by humans, a URL is a web address that directs us to specific content. Its basic form defines a communication protocol, domain, and path to the requested file.

The URL pictured above is easily understandable because it points to a **domain**, which is a human readable alias for machine level addresses. When accessing a web page using a domain-name-based URL, the human-readable address gets translated into its numerical counterpart, known as the **internet protocol (IP) address**. This number enables the computer to identify the location in the network and route data there. An IP address is assigned to every device connected to any kind of network and can be shared across multiple users, domains, or services. For example, one server with one IP address can host multiple websites with different domains. By analogy to the physical world, an IP address is like an apartment building with different tenants sharing a single street address.

Navigating through subdirectories can be practical when things are organized according to a clear hierarchy. But, as more and more locations come into existence and the number of branches to search through increases, this may not be efficient.

This system also allows different directories to hold identical files which is unnecessary for purposes like archiving. Using path location also becomes problematic when handling data whose location may change over time.

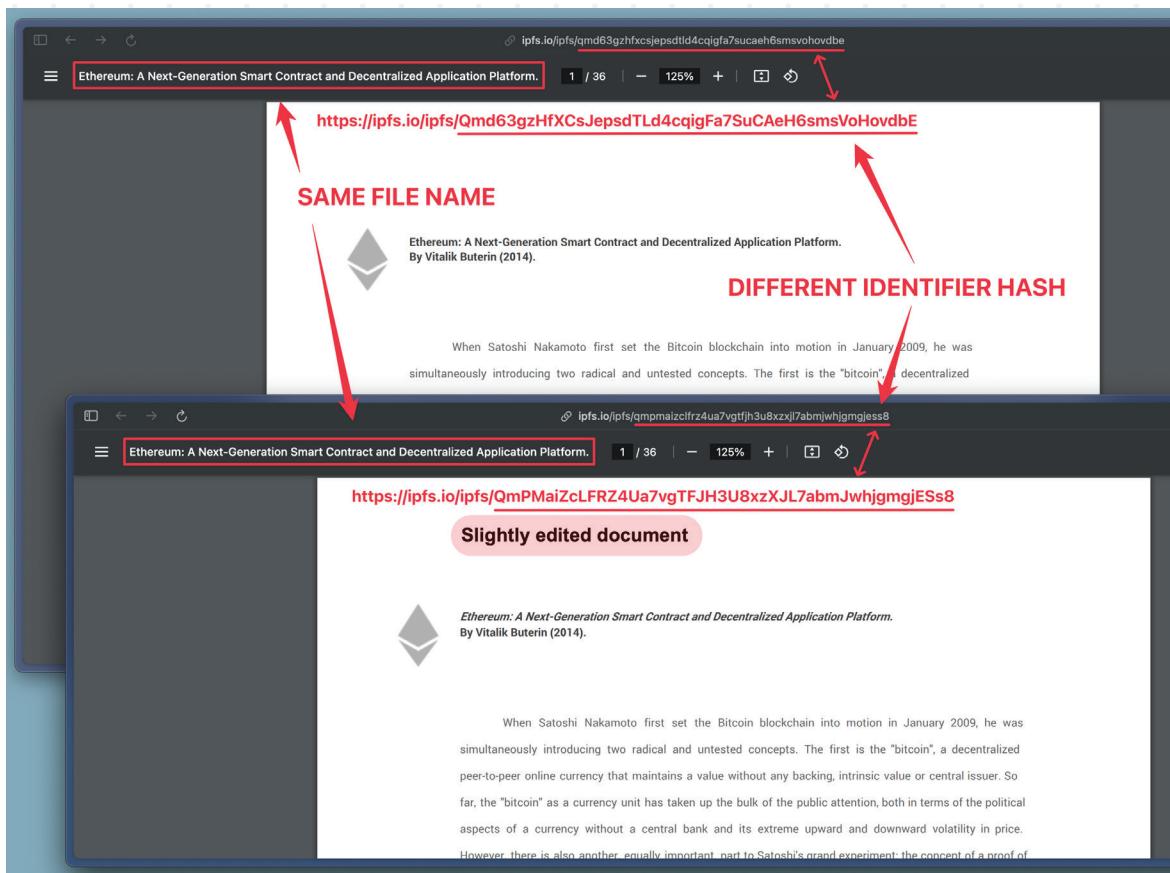
Luckily, another addressing scheme exists to address these drawbacks. **Content addressing** is a scheme which gives each file a unique address derived from its contents, rather than its location. This is especially useful for files that might be duplicated in multiple locations, some of which may become unavailable over time, but whose contents are stable. Specifically, content-addressable storage systems,

such as InterPlanetary File System (IPFS), retrieve files using an **identifier**. Identifiers are derived from the file's content using cryptographic hash functions. These take the file's data as an input and output a unique fingerprint. The same files will always result in the same fingerprint and therefore the same content identifier. Even the slightest change in the file's content will result in a completely new identifier and address.

Because the identifier is derived from the content, when changing even a single character in a file, its IPFS address also changes independently from its file name.

Content addressing is therefore ideal for files with fixed data, but variable locations. The uniqueness of identifiers allows for retrieving the desired file under the same address even if the file moves across different storage devices. In some cases, it's even possible to reconstruct a single

<https://ipfs.io/ipfs/Qmd63gzHfXCsJepsdTLd4cqigFa7SuCAeH6smsVoHovdbE>
<https://ipfs.io/ipfs/QmPMaiZcLFRZ4Ua7vgTFJH3U8xzXJL7abmJwhjgmgjESS8>



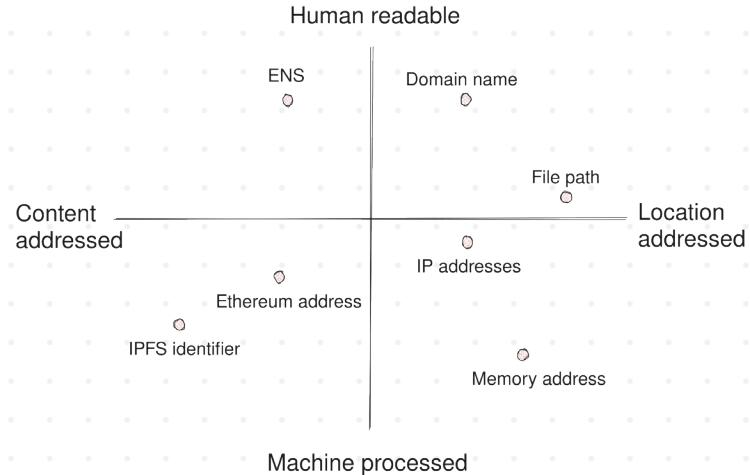
file from multiple sources, as the identifier guarantees the various bits received can be assembled into a single, coherent, output.

This feature makes content addressing the preferred basis for various modern decentralized protocols. One of the oldest examples is BitTorrent, which enabled peer-to-peer file sharing. Instead of data being hosted on a single server, torrents retrieve small chunks of a file shared by various peers in the network and reconstruct it locally.

A more modern example is IPFS, a decentralized storage network that enables cloud storage across a peer-to-peer network of hosts, not a single entity managing servers. Users of IPFS can seamlessly access files using static unique address, while the file can be communally hosted anywhere in the network, with varying levels of redundancy. Uploading the same file to the network then results in it having the same content identifier and adds redundancy to the protocol rather than wasted duplication.

While this is a valuable property, content addressing does come at the cost of losing human readability. In some cases, search engines have emerged to enable content discovery, such as ThePirateBay for torrents. In others, mapping schemes similar to the one between domains and IP addresses are more useful.

For example, blockchain addresses, which are derived from a specific private key, can be thought of as content addressing for that key. They are literally called **blockchain addresses!** In Ethereum's case, they are 20 byte hexadecimal values, prefixed by `0x`.¹ The Ethereum Name Service (ENS) allows users to register human readable names to point to an address (e.g., [summerofprotocols.eth](#) points to the aforementioned address), providing a human-readable alternative to the raw address.



To summarize, addressing is at the heart of how computers store, access, and exchange data. Addressing schemes used by computers aren't easily understandable to humans, and therefore higher-level addressing schemes are exposed to users, which the computer translates back to its internal logic.

Path- or URL-based addressing emphasizes the location of a piece of content, which results in a tree-like organizational structure. This is optimal for cases where the actual content in a specific location may change over time, but does not provide a way to access content without knowing its exact path.

Content addressing creates a unique identifier for a piece of data, allowing for its easy retrieval regardless of location. The downside there is that any change in the data results in a different identifier. Identifiers, because they are generated by hash functions, aren't as easy for humans to remember as file paths or domains.

MARIO HAVEL & TIM BEIKO are part of the Ethereum Foundation's Protocol Support team which helps facilitate Ethereum network upgrades as well as other protocol-related initiatives, including Summer of Protocols.
MARIO | [github.com/taxmeifyoucan](#)
TIM | [warpcast.com/tim](#)

1. For example, [0x00000000219ab540356cbb839cbe05303d7705fa](#)

ProtocolKit

summerofprotocols.com
hello@summerofprotocols.com

© 2023 Ethereum Foundation. All contributions are the property of their respective authors and are used by Ethereum Foundation under license. All contributions are licensed by their respective authors under CC BY-NC 4.0. After 2026-12-13, all contributions will be licensed by their respective authors under CC BY 4.0.

Learn more at: summerofprotocols.com/ccplus-license-2023

ISBN-13: 978-1-962872-53-9

Printed in the United States of America

Printing history: February 2024

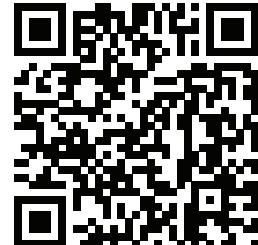
1 3 5 7 9 10 8 6 4 2

ISBN 978-1-962872-53-9



9 781962 872539

ProtocolKit



RETROSPECTUS



NEWSLETTER

