Blockchains and the Web

https://ipfs.io
/ipns/ipfs.io

W3C

@juanbenet
2016-06-29
0. IPFS
1. multi formats
2. IPLD
3. libp2p
0. IPFS
1. multi formats
2. IPLD
3. libp2p
WEB 1.0
<table>
<thead>
<tr>
<th>Secure, high perf p2p networking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure data structures + merkle web</td>
</tr>
<tr>
<td>Secure consensus + transaction ledger</td>
</tr>
<tr>
<td>Smart contracts + MPC</td>
</tr>
<tr>
<td>Verifiable, decentralized applications</td>
</tr>
</tbody>
</table>

IPLD
libp2p
IPFS
a new hypermedia distribution protocol
(a new web transport protocol)
The web has problems:

- **Bad in mobile and IoT**
- **Huge inefficiencies**
- **Links break**
- **Bad security model**
- **Censorship**
- **No offline use**
large open source project
400+ contributors
70+ contribute weekly
The IPFS Stack

**Defining the Data**
- IPNS
- IPLD

**Using the Data**
- applications
- naming
- merkledag

**Moving the Data**
- exchange
- routing
- network

libp2p
dns name  /dns/example.com/foo/bar/baz.png
key name  /ipns/QmYJPtosPTfoC/foo/bar/baz.png
content addr  /ipfs/QmW98pJrc6FZ6/foo/bar/baz.png

fs:/ipfs/QmW98pJrc6FZ6/foo/bar/baz.png
ipfs:/ipfs/QmW98pJrc6FZ6/foo/bar/baz.png
distributed webapps

- app code stored + distributed with ipfs
- app data stored + distributed with ipfs
- browsers can connect to each other
- no origin servers!
- no central point of failure
- everything end-to-end encrypted
- app "lives on the network"

examples: forums, chat, messaging, cms, blogs, github, ...
A Free Market for all.
No Fees. No Restrictions.

Hymettus Honey (From Greece)
used for package managers

- distributed / peer-to-peer
- cryptographically verified links
- digitally signed links
- "everyone is a mirror"
- save lots of bandwidth
- versioning built in

```bash
npm i <insert-module-name> --registry=<IPFS bridge>
```
npm on ipfs

[github.com/diasdavid/registry-mirror](https://github.com/diasdavid/registry-mirror)

works with vanilla npm

```bash
npm i <insert-module-name> --registry=<IPFS bridge>
```
npm on ipfs

github.com/diasdavid/registry-mirror
works with vanilla npm

npm i <insert-module-name> --registry=<IPFS bridge>

gx

github.com/whyrusleeping/gx
extensible pkg mgr
gx-go for Golang!
used for secure documents

- content addressed hash links
- digitally signed links
- trustless ledgers
- permanent links
- secure document web

already in use at:
- **banks**
- legal archives
- blockchain companies
- smart contract apps
Orbit

github.com/haadcode/orbit
p2p chat on IPFS
0. **IPFS**

1. multi formats

2. IPLD

3. libp2p
multiformats - self describing values
protocol agility, interop, avoid lock in

multihash
multiaddr
multibase
multicodes
multistream
multikey
multiformats - self describing values

protocol agility, interop, avoid lock in

multihash - cryptographic hashes
multiaddr
multibase
multicodec
multistream
multikey
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Length</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>sha2 256</td>
<td>256</td>
<td>0x08e11fc41466fcda0af7dee0905605d9e4aada4961542da952c8bb93080cc6f9</td>
</tr>
<tr>
<td>sha2 512</td>
<td>256</td>
<td>0x95a1b32bd70332e24f63f3802aae5f5e1fa4622cc72750e0073b5b6dcf6fceanf7</td>
</tr>
<tr>
<td>sha3</td>
<td>256</td>
<td>0xcaadb37a46daeda4e0d5e61574a9aaca211d513806a026e6cc4461f7ba7867f9</td>
</tr>
<tr>
<td>blake2b</td>
<td>256</td>
<td>0x08fbea061a5dea457d69fe5c12575c1d9d30c49f575936f6e1c6d4ea0ab078df</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Size</td>
<td>Hash 1</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>sha2-256</td>
<td>256</td>
<td>1120e11fc41466fceda0af7dee0905605d9</td>
</tr>
<tr>
<td>sha2-512</td>
<td>256</td>
<td>122095a1b32bd70332e24f63f3802aae5f5e</td>
</tr>
<tr>
<td>sha3</td>
<td>256</td>
<td>1420caadb37a46daeda4e0d5e61574a9aaca</td>
</tr>
<tr>
<td>blake2b</td>
<td>256</td>
<td>4020fbea061a5dea457d69fe5c12575c1d</td>
</tr>
</tbody>
</table>
fn code length

11 20

08e11fc41466fcdca0af7dee0905605d9
e4aada4961542da952c8bb93080cc6f9
Multihash - cryptographic hashes

- self describing
- in the value itself (not out of band)
- as small as possible
- no assumptions
- no lock in
- interop of hash functions
Because aesthetically I prefer the code first. You already have to write your stream parsing code to understand that a single byte already means "a length in bytes more to skip". Reversing these doesn’t buy you much.

Implementations:

- go-multihash
- node-multihash
- clj-multihash
- rust-multihash
- haskell-multihash
- python-multihash
- elixir-multihash, elixir-multihashing
- swift-multihash
- ruby-multihash
- scala-multihash

**table for Multihash v1.0.0-RC (semver)**

The current multihash table is here:

<table>
<thead>
<tr>
<th>code</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Øx11</td>
<td>sha1</td>
</tr>
<tr>
<td>Øx12</td>
<td>sha2-256</td>
</tr>
</tbody>
</table>
multiformats - self describing values
protocol agility, interop, avoid lock in

multihash - cryptographic hashes
multiaddr - network addresses
multibase - base encodings
multicodec - serialization codecs
multistream - stream wire protocols
multikey - cryptographic keys and artifacts
multiaddr - network addresses

/ip6/::1/tcp/80/http
/ip4/1.2.3.4/udp/5001/sctp/sip
/ip4/1.2.3.4/udp/5002/utp/bittorrent
/ip4/1.2.3.4/udp/5003/quic/ipfs
/onion/3g2upl4pq6kufc4m/80/http
multiformats - self describing values

protocol agility, interop, avoid lock in

multihash - cryptographic hashes
multiaddr - network addresses
multibase - base encodings
multicodec - serialization codecs
multistream - stream wire protocols
multikey - cryptographic keys and artifacts
multiformats - self describing values

protocol agility, interop, avoid lock in

<table>
<thead>
<tr>
<th>Multiformats</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>multihash</td>
<td>- In Value (not OOB)</td>
</tr>
<tr>
<td>multiaddr</td>
<td>- Small, Binary (perf)</td>
</tr>
<tr>
<td>multibase</td>
<td>- Human Readable</td>
</tr>
<tr>
<td>multicodec</td>
<td>- Stable</td>
</tr>
<tr>
<td>multistream</td>
<td>- Starting Standard Now</td>
</tr>
<tr>
<td>multikey</td>
<td>- Impls in many langs</td>
</tr>
</tbody>
</table>
distributed data structures

authenticated data structures

hash linked data structures
IPFS is like a forest of linked merkle-trees
IPLD

a common hash-chain format for distributed data structures
IPLD

- **merkle-links** secure, immutable
- **merkle-paths** /ipfs/Qmabc...xyz/foo/bar.jpg
- **canonical** hashing safe
- **universal** nestable URIs
- **serialization** CBOR, JSON, YAML, XML, PB
- **linked data** JSON-LD, RDF compatible
CBOR
RFC 7049 Concise Binary Object Representation

“The Concise Binary Object Representation (CBOR) is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation.”

JSON data model
CBOR is based on the wildly successful JSON data model: numbers, strings, arrays, maps (called objects in JSON), and a few values such as false, true, and null.

No Schema needed
One of the major practical wins of JSON is that successful data interchange is possible without casting a schema in concrete. This works much better in a world where both ends of a communication relationship may be evolving at different rates.

Embracing binary
Some applications that would like to use JSON need to transport binary data, such as encryption keys, graphic data, or sensor values. In JSON, these data need to be encoded (usually in base64 format), adding complexity and bulk.

Concise encoding
Some applications also benefit from CBOR itself being encoded in binary. This saves bulk and allows faster processing. One of the major motivators for the development of CBOR was the

Stable format
CBOR is defined in an Internet Standards Document, RFC 7049. The format has been designed to be stable for decades.

Extensible
To be able to grow with its applications and to incorporate future developments, a format specification needs to be extensible. CBOR defines tags as a mechanism to identify data that warrants additional information beyond the basic data model. Both future RFCs and third parties can provide these tags without making the data invalid.
a common hash-chain format for distributed data structures

\o/ Ready for Standardization! \o/
> var ipld = require('ipld')

> var obj1 = { "data": "Hello " }
> var ipld = require('ipld')

> var obj1 = { "data": "Hello " }

> var obj1Data = ipld.marshal(obj1)
> obj1Data.toString('base64')
oWRkYXRhZkhlbGxvIA==
> var ipld = require('ipld')

> var obj1 = { "data": "Hello " }

> var obj1Data = ipld.marshal(obj)
> obj1Data.toString('base64')
oWRkYXRhZkhlbGxvIA==

> var obj1Hash = ipld.multihash(obj1Data)
> obj1Hash
QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg
> var ipld = require('ipld')

> var obj1 = { "data": "Hello " }
> var obj1Data = ipld.marshal(obj1)
> var obj1Hash = ipld.multihash(obj1Data)
> var ipld = require('ipld')

> var obj1 = { "data": "Hello " }
> var obj1Data = ipld.marshal(obj1)
> var obj1Hash = ipld.multihash(obj1Data)
```javascript
> var ipld = require('ipld')

> var obj1 = { "data": "Hello " }
> var obj1Data = ipld.marshal(obj1)
> var obj1Hash = ipld.multihash(obj1Data)

> var obj2 = { "data": "World\n" }
> var obj2Data = ipld.marshal(obj2)
> var obj2Hash = ipld.multihash(obj2Data)

> obj2Hash
QmSVuc2kJbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV
```
> var obj3 = {
  "files": [
    { "": "QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg" },
    { "": "QmSVuc2kjbCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV" }
  ]
}
> var obj3Data = ipdl.marshall(obj3)
> var obj3Hash = ipdl.multihash(obj3Data)

> obj3Hash
QmdhMzs1tkLYwC3jimzUABEt1xzkrokkanywe1y1QFcAhw
```javascript
> var obj3 = {
  "files": [
    { "/": "QmUUuADWvRG23zyzBQVv43etRqmbGCRNgZYu9qVZ88Bg" },
    { "/": "QmSVuc2kbtCF9ur8fnyKUKvSyLZMTBVbZugJWChydAHV" }
  ]
}
> var obj3Data = ipld.marshal(obj3)
> var obj3Hash = ipld.multihash(obj3Data)

> obj3Hash
QmdhMzs1tkLYwC3jimzUABEL1xzkrokkkanywe1y1QFcahw
```
> var obj3 = {
  "files": [
    { "/": "QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg" },
    { "/": "QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV" },
  ]
}

> var obj3Data = ipld.marshall(obj3)
> var obj3Hash = ipld.multihash(obj3Data)
> var ipfs = require('ipfs')
> ipfs.add(obj1)
> ipfs.add(obj2)
> ipfs.add(obj3)
> var ipfs = require('ipfs')
> ipfs.add(obj1)
> ipfs.add(obj2)
> ipfs.add(obj3)

> ipfs.resolve("QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg")
{ "data": "Hello " }

> ipfs.resolve("QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV")
{ "data": "World\n" }
> var ipfs = require('ipfs')
> ipfs.add(obj1)
> ipfs.add(obj2)
> ipfs.add(obj3)

> ipfs.resolve("QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg")
{ "data": "Hello " }

> ipfs.resolve("QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV")
{ "data": "World\n" }

> ipfs.resolve("QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg/data")
"Hello "

> ipfs.resolve("QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV/data")
"World "
```javascript
var ipfs = require('ipfs')

ipfs.add(obj1)
ipfs.add(obj2)
ipfs.add(obj3)

ipfs.resolve("QmdhMzs1tkLYwC3jimzUABEt1xzkrokkywe1y1QFcAhw")
{
  "files": [
    { "/": "QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgzYu9qvZ88Bg" },
    { "/": "QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV" }
  ]
}
```
> var ipfs = require('ipfs')
> ipfs.add(obj1)
> ipfs.add(obj2)
> ipfs.add(obj3)
>
> ipfs.resolve("QmdhMzs1tkLYwC3jimzUABEt1xzkrokkanywe1y1QFcAhw")
> {
>   "files": [
>     { "/": "QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg" },
>     { "/": "QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV" }
>   ]
> }
>
> ipfs.resolve("QmdhMzs1tkLYwC3jimzUABEt1xzkrokkanywe1y1QFcAhw/files")
> [  
>   { "/": "QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg" },
>   { "/": "QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV" }
> ]
var ipfs = require('ipfs')

ipfs.add(obj1)
ipfs.add(obj2)
ipfs.add(obj3)

ipfs.resolve("QmdhMzs1tkLYwC3jimzUABEt1xzkrokkywe1y1QFcAhw/files/0")
{ "data": "Hello " }

ipfs.resolve("QmdhMzs1tkLYwC3jimzUABEt1xzkrokkywe1y1QFcAhw/files/1")
{ "data": "World\n" }

ipfs.resolve("QmdhMzs1tkLYwC3jimzUABEt1xzkrokkywe1y1QFcAhw/files/0/data")
"Hello "

ipfs.resolve("QmdhMzs1tkLYwC3jimzUABEt1xzkrokkywe1y1QFcAhw/files/1/data")
"World "

obj1
obj2
obj3
function catFile(link) {
    var obj = ipfs.resolve(link);
    var out = obj.data || "";
    for (var file of (obj.files || [])) {
        out += catFile(file);
    }
    return out;
}
function catFile(link) {
    var obj = ipfs.resolve(link)
    var out = obj.data || ""
    for (var file of (obj.files || [])) {
        out += catFile(file)
    }
    return out
}

> catFile("QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg")
"Hello "

> catFile("QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV")
"World\n"
```javascript
function catFile(link) {
  var obj = ipfs.resolve(link);
  var out = obj.data || ""
  for (var file of (obj.files || [])) {
    out += catFile(file);
  }
  return out;
}

> catFile("QmUUuaDDWvRG23zyzBQVv43etRqmbGCRNhgZYu9qvZ88Bg")
"Hello"

> catFile("QmSVuc2kjbtCFQ9ur8fnyKUKvSyLZMTBVbZugJWChydAHV")
"World\n"

> catFile("QmdhMzs1tkLYwC3jimzUABEt1xzkrokkanywe1y1QFcAhw")
"Hello World\n"
```
0. IPFS
1. multi formats
2. IPLD
3. libp2p
libp2p is a collection of peer-to-peer protocols for finding peers, and connecting to them, for finding content, and transferring it.
libp2p: a collection of peer-to-peer protocols

Content Routing
- mDNS
- pub sub
- Kad DHT
- Kad ICE
- ICE

Peer Routing
- mDNS
- DNS
- DVs
- Kad DHT
- STUN
- TURN

Discovery
- mDNS
- boot strap
- DNS
- Kad DHT
- PEX
- PKI

Transports
- TCP
- uTP
- QUIC
- SCTP
- BLE
- TOR
- I2P

NAT Traversal
libp2p

Description

libp2p is a networking stack and library modularized out of The IPFS Project, and launched separately for other tools to use.

libp2p is the product of a long and extensive quest of understanding—a deep dive into the Internet's network stack, and fluent peer-to-peer protocols from the past. Building large scale peer-to-peer systems has been complex and difficult in the past 15 years, and libp2p is a way to fix that. It is a “network stack” — a protocol suite — that cleanly separates concerns, and enables sophisticated applications to easily use the protocols they otherwise need, without giving up interoperability and observability. libp2p grew out of IPFS, but it is built so that the code of people can use it, for all of different projects.

We will be writing a set of docs, tests, and tools to explain what it is, why it is tremendously useful, and how it can help your existing and new projects. But, in the meantime, check out:

- The IPFS Network Stack, which grew into libp2p
- go-libp2p-implementation
- js-libp2p-implementation
Orbit

github.com/haadcode/orbit

p2p chat on IPFS
Blockchains and the Web

https://ipfs.io
/ipns/ipfs.io

@juanbenet
2016-06-29
IPLD

Routing

HTTP SSH TCP uTP TCP RLPx TLS?

Network

Exchange

Git Repl. (sync heads) BitTorrent (Tit for Tat) Bitcoin Gossip Trackers MainlineDHT

Application

Git BitTorrent Bitcoin Ethereum BigchainDB

Naming

DNS Blockstack etc ETH Names

Diagram with various technologies and their relationships.