What is Holochain?

For web devs

David Meister
What is Holochain?

Holochain

A p2p model for data sharing, storage and validation.
A FLOSS implementation of this model written in Rust + WASM.

Holo (a whooole other talk…)

A startup backing the development of Holochain with the goal of bringing it to the web.
The combination of:

● An implementation of Holochain
● Holochain applications
● A mutual credit economy
● Network infrastructure
● Hardware
What is a client/server web app?

User’s perspective (front-end)

A combination of HTML/JSON/data, JS/WASM, CSS & static assets.

A web browser environment/runtime to fetch, cache, render, execute and interact with these assets.

Server’s perspective (back-end)

Storage, retrieval and generation of HTML/JSON/data, JS/WASM, CSS & static assets.

Authentication (who are you?) and authorization (what can you read/write?).

Validating the shape/content of data reads/writes.

Background processes: sending emails, webhooks, heavy computation, etc.
# Socioeconomic Considerations of the Web

A very *un*comprehensive list...

## Pros
- Global (mostly) and instant access to unprecedented volumes of information
- New social and economic/business structures
- New forms of art, expression, community, etc.

## Cons
- Information overload
- "Someone" pays for client devices
- "Someone" pays for servers
- "Someone" pays for network infrastructure
- Whoever controls the devices, servers and networks dominates society
  - Bezos net worth 100+ billion
  - Zuckerberg net worth 50+ billion
  - Internet censorship/surveillance: Venezuela, China, Russia, Australia, America, Alexa, etc.
Originally there was little or no interactivity on the internet, so clients were fat, (FTP) servers were skinny.

Authors and users started to demand N-way interactions such as commerce (checkout) and blogs (comments).

JavaScript alone has never been able to handle this demand so servers naturally got fatter over time.

The fundamental problems with servers were NOT fixed and most intensified:

- Centralisation: silos/monopolies, data migrations, censorship, privacy/security concerns, legal issues
- Scale: at some point the only option is to buy more servers, more interaction usually means more $$$
- Widening innovation gap: web clients/frameworks iterate faster and more diversely than servers/dbs

At some point Single Page Applications (SPAs) become Serious Business again.

Our industry is currently reassessing what role servers *should* and *need* to play in our tech stack.
There are a few things that web browsers cannot do in the current web infrastructure paradigm.

- Security, identity, validating and enforcing rules (i.e. auth & cryptography)
- Data storage (i.e. consistent reading/writing today and tomorrow)
- Global consensus algorithms and long running processes (e.g. FIFO queues)
- (potentially) SEO, a11y and performance concerns

Everything else has a feasible option available to the browser, or an option “coming soon”.

- HTML/DOM generation/manipulation (e.g. react, angular, vue, etc.)
- Data querying/manipulation (e.g. in browser databases, datascript, etc.)
- Routing
- Interaction
- Internationalisation
- Graphics
- Etc.
A very *un*comprehensive list…

Pros
- Global (mostly) and instant access to unprecedented volumes of information
- New social and economic/business structures
- New forms of art, expression, community, etc.

Cons
- Information overload
- "Someone" pays for client devices
- "Someone" pays for servers
- "Someone" pays for network infrastructure
- **Whoever controls the devices, servers and networks dominates society**

Holochain asks “what if users were their own server?”
Where does Holochain sit?

A p2p answer to some of the reasons we need servers today:

- Data storage and sharing through application specific DHT (e.g. Bittorrent on a private network)
- Cryptographically secure data validation through WASM
- An agent (user) centric security model for rule enforcement
- Direct messaging between agents (users) to coordinate data to be shared

Holochain does NOT help with many of the things we do on servers for historical/convenience reasons. Holochain does NOT offer consensus algorithms (e.g. FIFO queues, blockchains, etc.).

**SPAs with microservice infrastructure and key/value or graph style databases are easiest to port.**

As little as possible should be done in Holochain by design. You can and should use your favourite language/framework for 95% of your application. Everything in Holochain code is implied to be on the critical security path. It should be easy to see what to audit.

Fat UI, skinny Holochain.
Validation vs. consensus

Validation is a true/false test applied to any data that anyone can check independently at any time. User’s can’t force other users to accept or agree to something that doesn’t validate.

Consensus is a set of rules that everyone runs in real time to control who can write what to a data set. Users can censor/control what is written for everyone if they “win” the consensus game.

“This must be a number"
= validation because anyone can decide at any time if something is a number

“The next person to find a hash of the last 1mb of data starting with 3 zeros can write the next 1mb of data”
= consensus because everyone must endlessly sync their hashes and data in real time to participate

In Holochain everyone does their own validation before writing/sharing any data.

Every application defines its own rules, just like normal server-side code.
ARCHITECTURE
Holochain uses a Distributed Hash Table (DHT) used most famously in BitTorrent.

This means that data is content addressed rather than location addressed.

Web URLs are locations: https://example.com/foo is a location that maps to responses from some server(s). The structure of a URL already implies control structures. Who decides where “example.com” and “/foo” go?

Content addressed data uses its hash to look it up. In a way the content is its own URL in the DHT.

The string “foo” hashes to “2C26B46B68FFC68FF99B453C1D30413413422D706483BFA0F98A5E886266E7AE”.

Later we use the hash to retrieve “foo” from the network.
The benefits are more obvious when the content is much longer than the hash!

Content is redundantly stored to guard against data loss and uptime issues. For example, we can send “foo” to 10 different users. As long as at least 1 user responds with “foo” we are OK. If network issues cause only 5 holders of “foo” to be online, the data is copied until we are back to 10 live copies.
For data validation and security everyone has their own copy of any data they need and run their own validation.

There is a private network + DHT for each application. The hash of the validation code is the key to the network.

Anyone who tries to run bad validation logic will accept bad data but not cause anyone else to use it.

To make this work validation logic is always pure and deterministic based on content.

The same fundamental principles as a board game.
The current state of the board is the data to be validated, the game rules are known by everyone playing. The only way to cheat is if everyone allows it because anyone can refuse to play from an invalid board state.

The goal is NOT for everyone to see and participate in all moves for all games at once. The goal is for honest participants to be able to protect themselves against fraud in their own games.

For example, two chess players could invent “house rules” between themselves. They could NOT send the results of that game to a chess tournament, or influence the outcome of other games.
Unlike (most?) other p2p systems there is no expectation of anonymity in general.

Many Holochain apps rely on **authentication of real world identity** in order to securely **authorize** validation rules.

For example, I could create a simple blog backend where:

- The super user is hard-coded using a private key I own
- Any user that can provide a valid JWT signed by my private key can edit/contribute
- Any user that can provide a valid JWT signed by Auth0 (SaaS auth) can comment

Somewhat arbitrary and contrived but it shows:

- I can secure a blog without a server using standard crypto keys/signing
- I can also secure a blog with a server (Auth0) to create hybrid models
- Anyone who can verify signatures can **decide for themselves** which posts, comments and edits are “valid”
- The relationship between the UI (**provide** evidence e.g. “auth token”) and Holochain (**validate** evidence)
Source chains for own data

PRIVATE SOURCE CHAIN

HOLOCHAIN DNA

HEADER 0: GENESIS

AGENT ID

PRIVATE SOURCE CHAIN

DHT

HOLOCHAIN APP
### Centralised Client-Server Architecture

- **Server**
  - Back end logic
  - Data & Backups

- **Client**
  - Front end code and data

- **Server** acts as middleman and controller for all interactions between clients.
- **All data stored centrally (may be internally distributed)**

### Holochain Architecture

- **Holochain Node**
  - Application Logic
  - Backups of others’ source chains (DHT)
  - User data (Source chain)

- **Holochain nodes interact directly with each other**
- **User data (source chains) are stored on the user’s own computer**
- **Holo hosts are Holochain nodes that can serve data to a client browser**
Holochain vs. blockchain (and other consensus systems)

**Holochain Architecture**

- Holochain nodes interact directly with each other.
- User data (source chains) are stored on the user’s own computer.
- Holo hosts are Holochain nodes that can serve data to a client browser.

**Blockchain Architecture**

- Consensus-based DLTs act as a middleman between users.
- Non-critical application logic and data may be stored in centralised systems to reduce costs.
STATE OF PLAY
Our current tech stack is...

**Rust** for the **core** Holochain workflows (triggering validation, etc.).

**WASM** for **application** logic.

This means that application developers need to write **WASM callbacks**...
Which is very low-level and painful to do by hand.
So we support a **Rust HDK** that allows application logic to be written in Rust that compiles to WASM.

Application logic is **callback + API** based.
Core implements an internal API that holochain developers can use to trigger reads/writes/validation/etc.
Holochain app developers implement an external facing API that UIs can call.

Holochain is designed to be as **device/environment agnostic** as possible...
The storage, network and exposed APIs are all implemented by environment specific **conductors**.
There is currently a **nodejs** conductor and a **Rust/cli** conductor, both use **websockets** to communicate with UIs.
Officially supported core and conductors are tested through CI tools and pass tests on all platforms. Officially supported platforms right now are Debian, Ubuntu LTS, Windows, & Mac OS X. Official support for mobile devices is roadmapped but desktop native/web (Holo) is current priority.

Overall maturity of the codebase is **early alpha**. As of this month we have adopted a (roughly) **weekly release cycle** that app developers can pin versions against.

An indicative list of things we have NOT done:

- Performance benchmarking
- Sophisticated networking
- Security audits
- Simple onboarding process
- Comprehensive documentation
- CRDTs (resolving “merge conflicts” for live application data)
- Sophisticated data management (file streaming, large BLOBs, etc.)
# Holochain street cred

**Holochain Development**

<table>
<thead>
<tr>
<th>Holochain Github Repos</th>
<th>Holo Github Repos</th>
<th>Community</th>
<th>Exchanges/Markets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Holochain</strong> 83 repos</td>
<td><strong>Holo</strong> 41 public repos (55 in total, 35 active repos)</td>
<td><strong>Telegram: 13K</strong></td>
<td><strong>HOT Token Market Cap</strong></td>
</tr>
<tr>
<td><strong>Holochain Rust (as 3/12/19):</strong></td>
<td></td>
<td><strong>Mattermost: 4.6K</strong></td>
<td><strong>$0.001093</strong></td>
</tr>
<tr>
<td>8,789+ commits, 56 releases, 718 closed pull requests, 27 open pull requests</td>
<td></td>
<td><strong>Reddit: 5.6K</strong></td>
<td><strong>$145,563,556</strong> $1,788,156</td>
</tr>
</tbody>
</table>

**# Developers**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Holo</th>
<th>Holochain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter</td>
<td>18.9K</td>
<td>24.3K</td>
</tr>
<tr>
<td>Youtube</td>
<td>3.4K</td>
<td>1.8K</td>
</tr>
<tr>
<td>Medium</td>
<td>3.5K</td>
<td>1.9K</td>
</tr>
</tbody>
</table>

**# Developers**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Top 10 Exchanges</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>binance.com</td>
</tr>
<tr>
<td></td>
<td>Other Exchanges</td>
</tr>
<tr>
<td></td>
<td>asheswip.io/trade, atomicwallet.io, bitprime.co.nz, bitsonic.co.kr, cashierest.com, chainrift.com, coinex.com, coinzo.com, fatbtc.com, hotbit.io, idex.market/eth/hot, joyso.io, liqui.io, sobit.one</td>
</tr>
</tbody>
</table>

**Trading Platforms**

<table>
<thead>
<tr>
<th>Platform</th>
<th>binuniverse.org, blockfolio.com, delta.app,</th>
</tr>
</thead>
</table>

**Complete List:**

https://telegra.ph/Holo-Exchanges-11-25
How to get involved

Everything is still early...

Best ways to get your feet wet:

- Join chat - https://chat.holochain.org
- Learn about Rust/WASM
  - good general FOSS skills anyway, part of web standards, backed by Mozilla etc.
  - A little Rust goes a long way, remember that 95% of your app is still regular web dev!
- Learn about existing SPA + microservice patterns
- Read the documentation we do have
- Come to a meetup (next Melbourne event is 1st April)
- Try downloading some sample apps/binaries and give feedback on what doesn’t work for you