Towards web3 infrastructure

Viktor Trón

November 30, 2016

- killer app for blockchain?
- peer to peer technologies with incentives
- base layer infrastructure for the third web
- what do dapps need? web3 dev stack

ethereum

whisper

swarm

IPFS

COLONY

BIGCHAINDB

ethereum

HTML 5

whisper

swarm

oraclize

Raiden Network
High Speed Asset Transfers for Ethereum

JAAK
A Smart Content Platform

PSS

IPFS

COLONY

BIGCHAIN DB

ethereum

SWATCH

HTML 5

whisper

swarm

SWORD

oraclize

Raiden Network

High Speed Asset Transfers for Ethereum

JAAK

A Smart Content Platform

1 Internode communication

2 multimedia live broadcast

3 Database services

Outline

1. Internode communication

2. multimedia live broadcast

3. Database services

How does information (dynamic content) move around?

Using the same routing and incentive system as storage and retrieval.

Chunks

under the hood swarm does not deal in files but in *chunks*.

Chunks

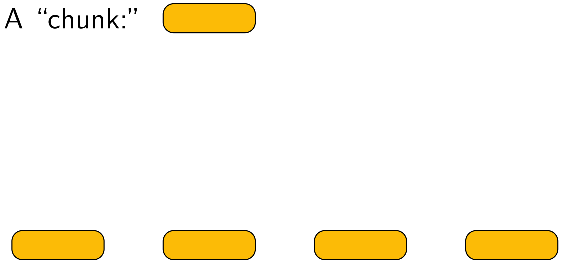
under the hood swarm does not deal in files but in *chunks*.

- all data is broken into pieces of size 4kB: “chunks”.

Chunks

under the hood swarm does not deal in files but in *chunks.*

- all data is broken into pieces of size 4kB: “chunks”.

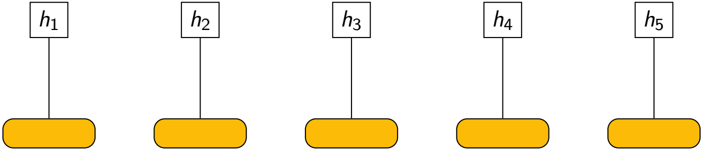


A “chunk:”

Chunks

under the hood swarm does not deal in files but in *chunks*.

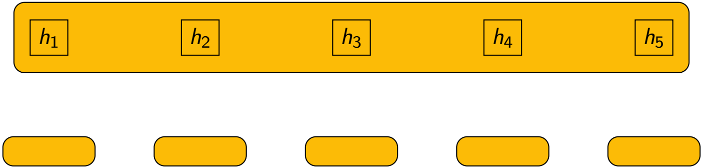
- all data is broken into pieces of size 4kB: “chunks”.
- chunks are hashed and the hash is used as their ID/address.



it's chunks all the way down...

under the hood swarm does not deal in files but in *chunks*.

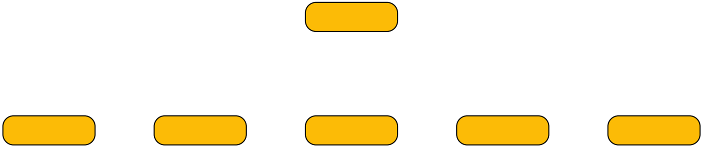
- all data is broken into pieces of size 4kB: "chunks".
- chunks are hashed and the hash is used as their ID/address.
- chunk hashes are also packaged into 4kB chunks...



it's chunks all the way down...

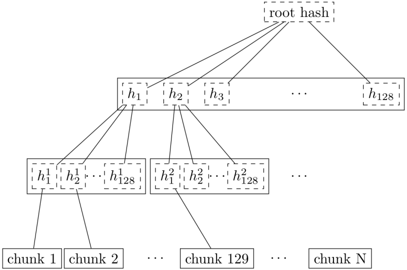
under the hood swarm does not deal in files but in *chunks.*

- all data is broken into pieces of size 4kB: "chunks".
- chunks are hashed and the hash is used as their ID/address.
- chunk hashes are also packaged into 4kB chunks...



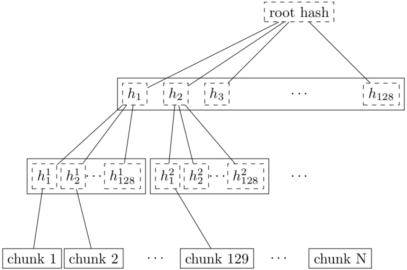
chunks are assembled in a **Merkle Tree**.

- files are retrievable using a single 32byte hash
- built-in integrity protection and random access



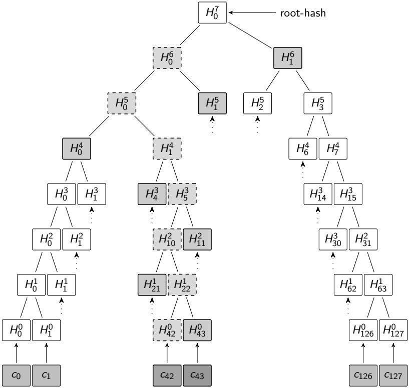
chunks are assembled in a **Merkle Tree**.

- files are retrievable using a single 32byte hash
- built-in integrity protection and random access
- merkle-proofs enable proof-of-custody schemes



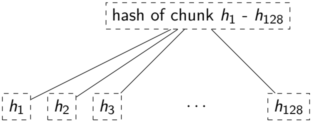
chunks are assembled in a **Merkle Tree**.

- files are retrievable using a single 32byte hash
- built-in integrity protection and random access
- merkle-proofs enable proof-of-custody schemes
- traversible using ASCII charactes due to branching factor of 128



chunks are assembled in a **Merkle Tree**.

- files are retrievable using a single 32byte hash
- built-in integrity protection and random access
- merkle-proofs enable proof-of-custody schemes
- traversible using ASCII charactes due to branching factor of 128



Chrome
Google
Vine
Instagram
LinkedIn
Pinterest
YouTu
Twitt

Internode communication

What kinds of interactions are we used to?

- status updates (public or restricted)
- chatroom, discussion forum, Q&A forum
- pager & fax, phonecall, videocall, voicemail
- audio-video broadcast, tv, radio, podcast (live or recorded)
- rss, subscription, pub/sub, notifications, newsletters
- file transfer, download
- datastreams, feeds, message bus

Internode communication

What kinds of interactions are we used to?

- status updates (public or restricted)
- chatroom, discussion forum, Q&A forum
- pager & fax, phonecall, videocall, voicemail
- audio-video broadcast, tv, radio, podcast (live or recorded)
- rss, subscription, pub/sub, notifications, newsletters
- file transfer, download
- datastreams, feeds, message bus

Question:

Why are these services provided by private entities and are not part of the basic public infrastructure? After all, everything is just pulling, pushing and storing data.

To replicate services we are used to, specify storage and delivery criteria.

- Who is it for?
- How should it be stored and transported?
- Encryption?
- What is the context?

To replicate services we are used to, specify storage and delivery criteria.

- Who is it for?
- How should it be stored and transported?
- Encryption?
- What is the context?

- Is it addressed to specific recipients?
- Should it be (re-)delivered to specific recipients?

To replicate services we are used to, specify storage and delivery criteria.

- Who is it for?
- How should it be stored and transported?
- Encryption?
- What is the context?

- Should it be stored (at content address) or is it ephemeral?
- Does it have high priority, is it urgent, is latency a factor?
- Should it be archived? is it insured? expiring?
- Is data access recorded/receipted?

To replicate services we are used to, specify storage and delivery criteria.

- Who is it for?
- How should it be stored and transported?
- Encryption?
- What is the context?

- Is it confidential? Private?

To replicate services we are used to, specify storage and delivery criteria.

- Who is it for?
- How should it be stored and transported?
- Encryption?
- What is the context?

- reaction to previous communication, content, topic? Comments, answers, corrections
- Existing asset (reference), streaming data, real time feed?
- How should the data be displayed? Timeline or thematic/threaded view

vision

comprehensive communications infrastructure

vision

comprehensive communications infrastructure

Tools at our disposal

- the recursive off kademlia network for deterministic message routing
- incentivised message relay (store requests sent towards non-content address must be paid for)
- deterministic routing and message delivery
- priority queues
- insured storage
- taking receipts
- multicast broadcast

PSS: **p**ostal **s**ervices **s**uite (bzz whispered)

Outline

1 Internode communication

2 multimedia live broadcast

3 Database services

How can this handle live multimedia sessions?

Multimedia live broadcast: multibitrate low-latency streaming

- leech - continuous data stream from peers
- non-multiplexed multi-bitrate stream offered
- RTSP/MPEG-DASH standard - available in most browsers using html5 video tag
- webRTC or FFMEG to capture and encode streams
- multicast tree - solves scalability of media server solutions
- p2p symmetry: the same technique for videoconference or even one-on-one AV session
- data goes to viewers via pairwise transmission channels
- peers sit on the multicast chain and get promoted, demoted depending on payment and latency/throughput

SWATCH: **s**treaming **w**ith **a**daptive **t**ransmission **ch**annels

Outline

1 Internode communication

2 multimedia live broadcast

3 Database services

Database services

Where is information (dynamic content) pulled from?

1. the blockchain, ethereum state & contract storage (expensive and slow)
2. local storage private to user, cookies (limited to data only client uses)
3. distributed database on swarm? (cheap and verifiable)

Manifests

We can take this one step further, be tying together various swarm assets under a new root-hash by generating a new tree: A **Manifest**

A Swarm Manifest...

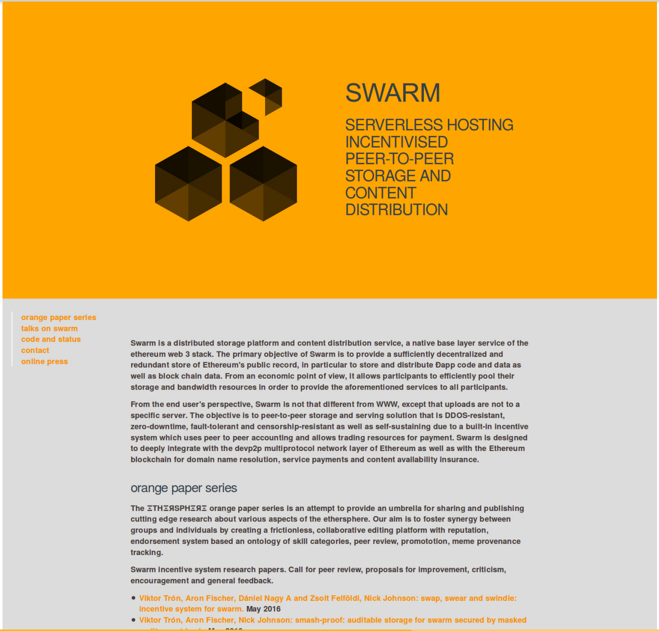
...is a Merkle tree whose leaves are root-hashes of other swarm assets (files, collections, manifests, chunks...)

The only difference between this and the chunk-tree of a file, is that it is not balanced and has metadata.

For example,

For example, the Swarm landing page

`swarm-gateways.net/bzz:/swarm/`



SWARM

SERVERLESS HOSTING INCENTIVISED PEER-TO-PEER STORAGE AND CONTENT DISTRIBUTION

orange paper series
talks on swarm
code and status
contact
online press

Swarm is a distributed storage platform and content distribution service, a native base layer service of the ethereum web 3 stack. The primary objective of Swarm is to provide a sufficiently decentralized and redundant store of Ethereum's public record, in particular to store and distribute Ðapp code and data as well as block chain data. From an economic point of view, it allows participants to efficiently pool their storage and bandwidth resources in order to provide the aforementioned services to all participants.

From the end user's perspective, Swarm is not that different from WWW, except that uploads are not to a specific server. The objective is to peer-to-peer storage and serving solution that is DDOS-resistant, zero-downtime, fault-tolerant and censorship-resistant as well as self-sustaining due to a built-in incentive system which uses peer to peer accounting and allows trading resources for payment. Swarm is designed to deeply integrate with the devp2p multiprotocol network layer of Ethereum as well as with the Ethereum blockchain for domain name resolution, service payments and content availability insurance.

orange paper series

The ΞTHΞЯSPHΞЯΞ orange paper series is an attempt to provide an umbrella for sharing and publishing cutting edge research about various aspects of the ethersphere. Our aim is to foster synergy between groups and individuals by creating a frictionless, collaborative editing platform with reputation, endorsement system based an ontology of skill categories, peer review, promototion, meme provenance tracking.

Swarm incentive system research papers. Call for peer review, proposals for improvement, criticism, encouragement and general feedback.

- Viktor Trón, Aron Fischer, Dániel Nagy A and Zsolt Felföldi, Nick Johnson: swap, swear and swindle: incentive system for swarm. May 2016
- Viktor Trón, Aron Fischer, Nick Johnson: smash-proof: auditable storage for swarm secured by masked

Manifests

...is loaded from this 4-entry manifest:

```
{"entries":[{
"path":"Swarm_files/",
"hash":"0294e48456a49fe7c02162c83b068075ff9ae6aaafb46439dba32da7de548379",
"contentType":"application/bzz-manifest+json",
"status":0},
{"path":"ethersphere/orange-papers/"...
{"path":"i"...
{"path":"talks/"...
{"path":"",
"hash":"6fac0b0c1f118f7f383792c0f01c80d1b2dc94f0e166d62ff4f999a926e9d94a",
"contentType":"text/html;charset=utf-8","status":0}]}
```

Manifests

...is loaded from this 4-entry manifest:

```
{"entries":[{
"path":"Swarm_files/",
"hash":"0294e48456a49fe7c02162c83b068075ff9ae6aaafb46439dba32da7de548379",
"contentType":"application/bzz-manifest+json",
"status":0},
{"path":"ethersphere/orange-papers/"...
{"path":"i"...
{"path":"talks/"...
{"path":"",
"hash":"6fac0b0c1f118f7f383792c0f01c80d1b2dc94f0e166d62ff4f999a926e9d94a",
"contentType":"text/html;charset=utf-8","status":0}]}
```

Manifests translate a URL path into swarm hashes (URL defines manifest merkle-tree traversal).

When combined with the Ethereum Name Service (ENS) to register a name for the manifest's own root hash, we can **serve any and all swarm data directly to your browser using human readable names**.

Example: Swarm File Manager

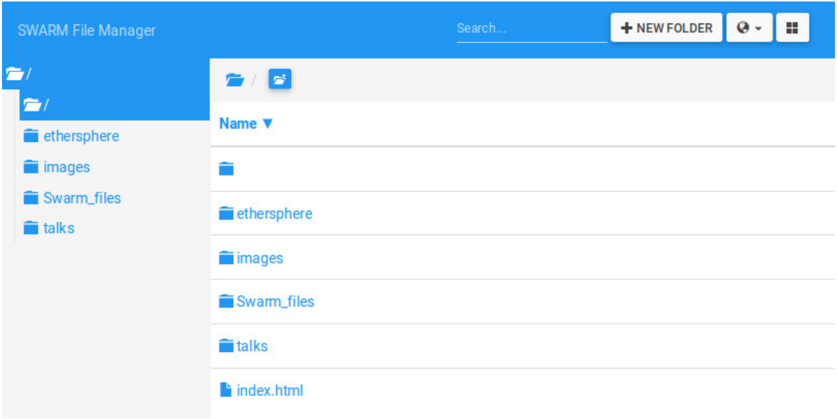
With manifests, you can navigate swarm just like you would navigate your own filesystem.

Example: Swarm File Manager

With manifests, you can navigate swarm just like you would navigate your own filesystem.
Let us open the swarm landing page in the swarm file manager:

Example: Swarm File Manager

With manifests, you can navigate swarm just like you would navigate your own filesystem.
Let us open the swarm landing page in the swarm file manager:



manifests

- only root hashes need to be registered (ENS) on blockchain
- site is integrity protected
- two-way translation possible from directories to routes on the domain

manifests enable

- filesystem API
- Dropbox, rsync, ...
- filesystem driver (FUSE)

extend manifests with metadata

- http headers
- copyright information
- access control
- payment triggers
- auto-play continuation
- subscription information
- database layout info

How are database services organised?

- structure - manifests
- security - blockchain proofs
- scalability - off-chain computation
- sustainability - incentives

How?

- manifests implement key-value store (Patricia Merkle Trie as oppose to traditional DHT)
- supports various indexes and iteration (range queries)
- conventions db table layouts in manifest metadata
- table and index roots anchored in Ethereum Name Service

How are database services organised?

- structure - manifests
- security - blockchain proofs
- scalability - off-chain computation
- sustainability - incentives

How?

- verifiable on the blockchain by challange
- verifiable authentication, record updates and notifications
- verifiable indexes, query resolution

How are database services organised?

- structure - manifests
- security - blockchain proofs
- scalability - off-chain computation
- sustainability - incentives

How?

- sql resolver (this reql of rethinkdb) sitting on top
- parallel processes walk the indexes and merge results
- index updates, derivative data (full text search indexes, aggregate statistics) supplied by a computational market
- query caching and accelerated retrieval for real-time low latency experience supplied by specialised nodes

How are database services organised?

- structure - manifests
- security - blockchain proofs
- scalability - off-chain computation
- sustainability - incentives

How?

- due to verifiable computations (truebit, ewasm), *swap, swear and swindle* is applicable
- positive and negative incentivisation
- secondary market for compensatory insurance

SWORD: **S**tate **W**ith **O**n-demand **R**etrieval of **D**ata

can we put the ethereum blockchain and state on swarm

- light client - flexible transition from remote, light, full and archival nodes
- solves the scalability problem of too big state data, receipts, contract storage, fast syncing
- decentralised blockchain explorer