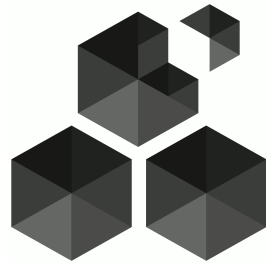


# Berlin Ethereum Meetup



swarm

25<sup>th</sup> May 2016 presentation by  
Viktor Trón, Dániel A. Nagy & Aron Fischer

Swarm

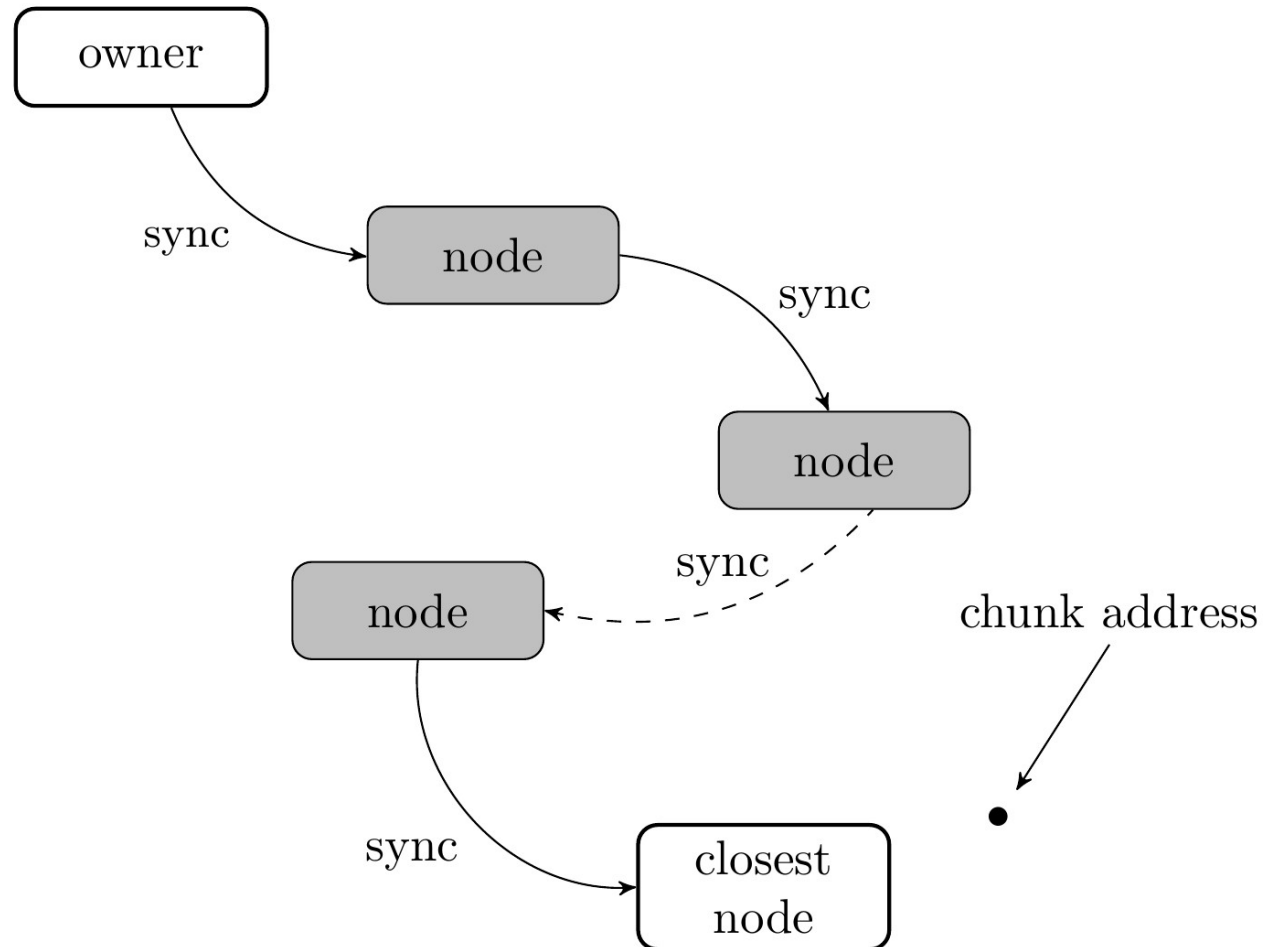
# A (very quick) introduction to Swarm

# A (very quick) introduction to Swarm

1. Get data into the swarm

# A (very quick) introduction to Swarm

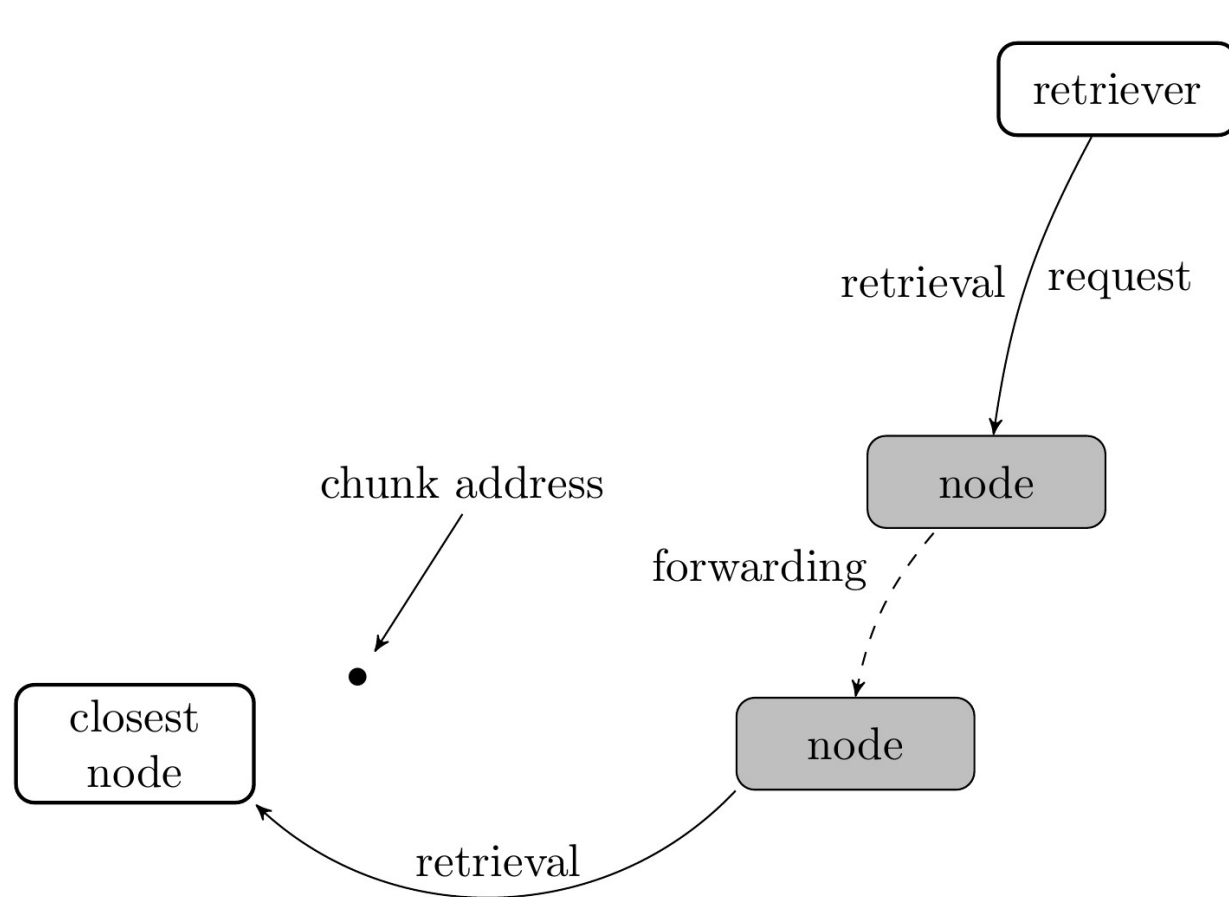
- Data is chopped up into chunks.
- Chunks are forwarded node-to-node (sync)
- Chunks end up with node whose address is closest to chunk hash



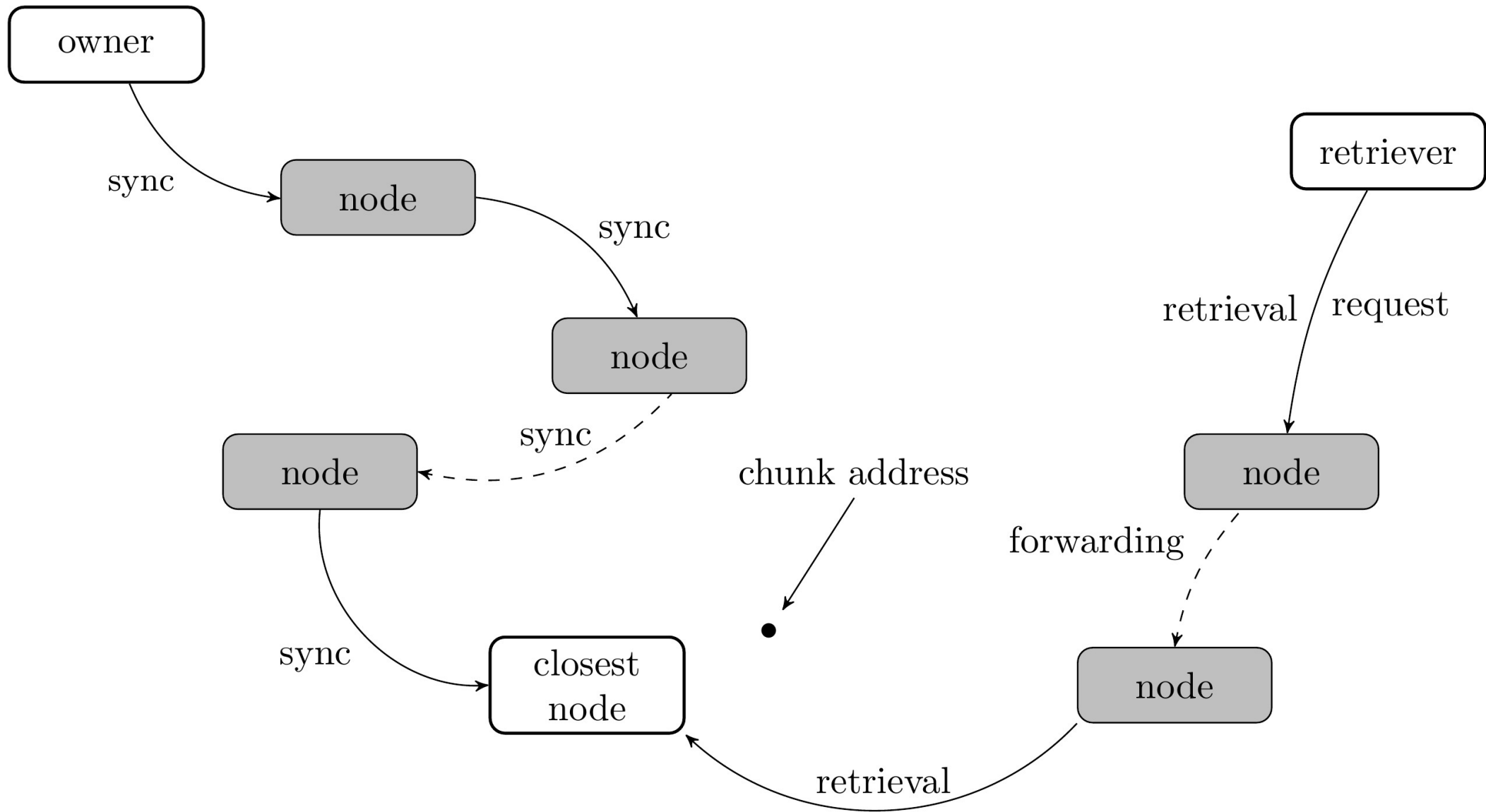
# A (very quick) introduction to Swarm

2. Get data out of the swarm

# A (very quick) introduction to Swarm



- Retriever makes a request to a closer connected node
- Nodes pass on requests
- Forwarding ends when chunk is found. Chunk is passed back.







# Incentivisation in SWARM

A brief history of:  
**Web Hosting and Incentivisation**

# A brief history of: Web Hosting and Incentivisation

Web 1.0

# A brief history of: Web Hosting and Incentivisation

## Web 1.0

- Start a web server
- Upload content

# A brief history of: Web Hosting and Incentivisation

## Web 1.0

- Start a web server
- Upload content

### 1. Content is unpopular

- pay costs of maintaining webserver

# A brief history of: Web Hosting and Incentivisation

## Web 1.0

- Start a web server
- Upload content

1. Content is unpopular
  - pay costs of maintaining webserver
2. Content becomes popular

# A brief history of: Web Hosting and Incentivisation

## Web 1.0

- Start a web server
- Upload content

### 1. Content is unpopular

- pay costs of maintaining webserver

### 2. Content becomes popular

- bandwidth costs skyrocket
- server crashes / goes offline



# A brief history of: Web Hosting and Incentivisation

## Web 1.0

- Start a web server
- Upload content

### 1. Content is unpopular

- pay costs of maintaining webserver

### 2. Content becomes popular

- bandwidth costs skyrocket
- server crashes / goes offline

...but at least you owned your own content.

A brief history of:  
**Web Hosting and Incentivisation**

Web 2.0

# A brief history of: Web Hosting and Incentivisation

## Web 2.0

- Upload content to the 'cloud'
  - cheap/free
  - scalable

# A brief history of: Web Hosting and Incentivisation

## Web 2.0

- Upload content to the 'cloud'
  - cheap/free
  - scalable

But...

# A brief history of: Web Hosting and Incentivisation

## Web 2.0

- Upload content to the 'cloud'
  - cheap/free
  - scalable

## But...

- Content owned by the service providers
- All users are tracked and spied on; providers profit off the data.
- Centralised control: surveillance and censorship.

A brief history of:  
**Web Hosting and Incentivisation**

Peer to Peer Networks?

# A brief history of: Web Hosting and Incentivisation

## Peer to Peer Networks

eg. Bittorrent

- Content is distributed among peers
- Distribution scales automatically
- Hashing ensures data integrity
- No central point of failure / no servers

# A brief history of: Web Hosting and Incentivisation

## Peer to Peer Networks

eg. Bittorrent

- Content is distributed among peers
- Distribution scales automatically
- Hashing ensures data integrity
- No central point of failure / no servers

But:

Downloads start slowly (high latency)

No incentive to provide content: “seeding”





# Swarm Incentive System

# Swarm Incentive System

Bandwidth

Storage

# Swarm Incentive System

## Bandwidth

- accounting for bandwidth used in the p2p setting
- compensating nodes based on the bandwidth they provide

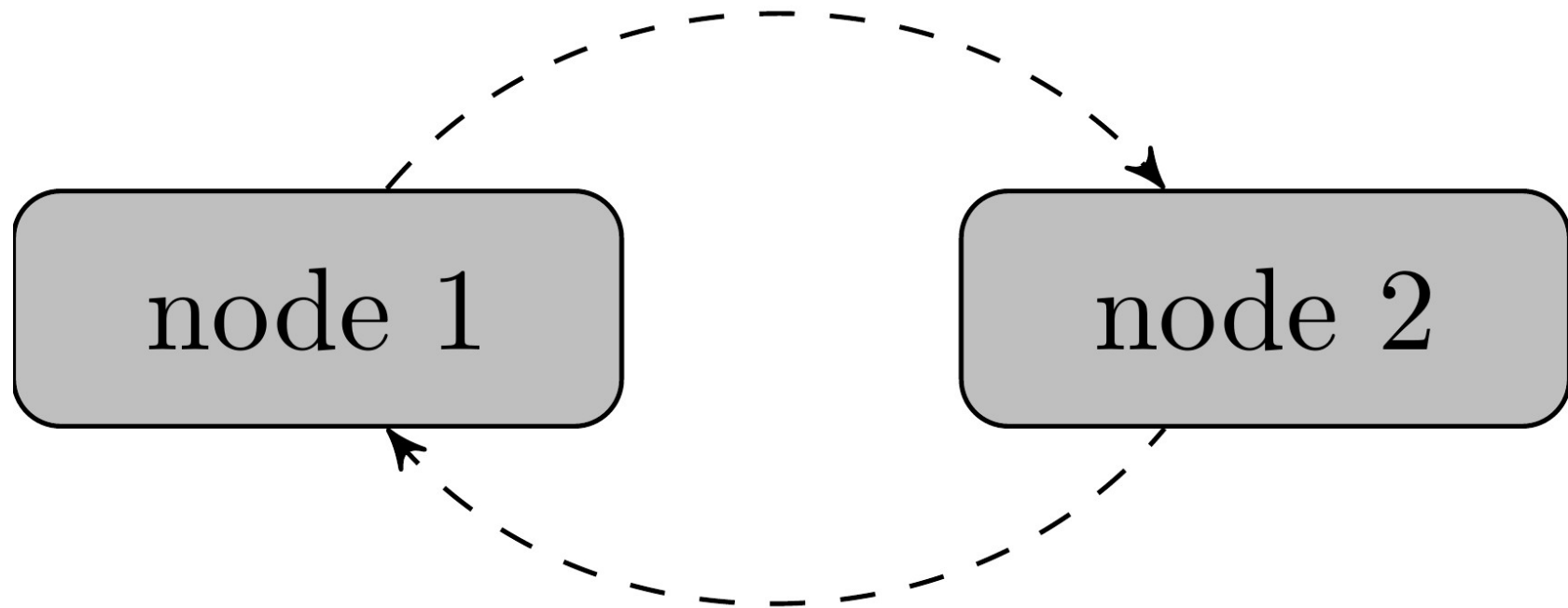
## Storage

- allow for long-term storage of data in the swarm
- provide proper compensation to nodes for storing data

# Swarm Incentive System

## Bandwidth

Bandwidth accounting is per-peer

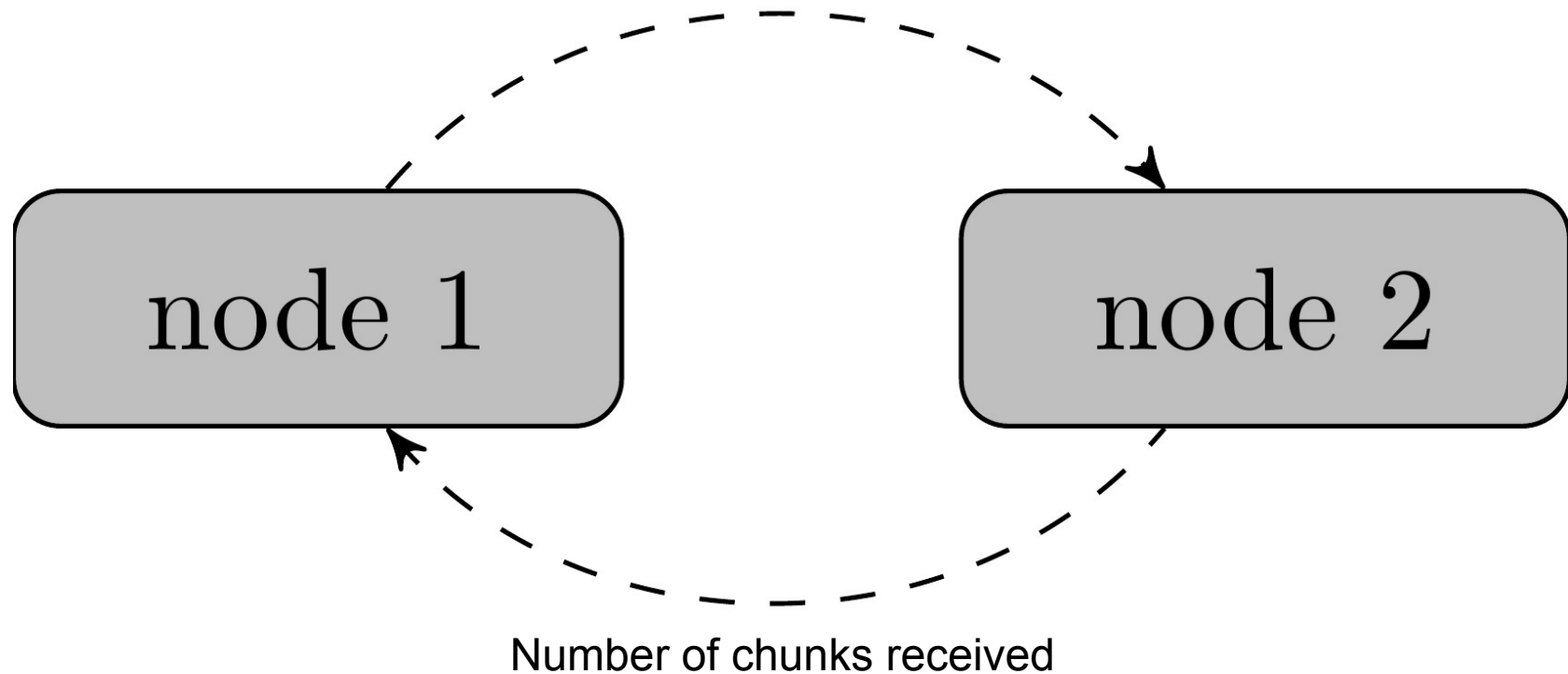


# Swarm Incentive System

## Bandwidth

Bandwidth accounting is per-peer

Number of chunks supplied



# SWAP – Swarm Accounting Protocol

# SWAP – Swarm Accounting Protocol

- Keeps track of number of chunks provided/received per peer
- Can trade chunk-for-chunk or chunk-for-payment
- Payments are made using the swarm chequebook contract on the blockchain  
(cheques are cumulative: you only ever have to cash the last one, thus saving transaction costs)



# SWAP – Swarm Accounting Protocol

Big picture:

- If you download a lot of content, you pay your peers for providing it.
- If you host popular content, you will earn fees from your peers for making the content available.

# SWAP – Swarm Accounting Protocol

Big picture:

- If you download a lot of content, you pay your peers for providing it.
- If you host popular content, you will earn fees from your peers for making the content available.
- Swarm is auto-scaling.
  - interplay of routing protocol and per-chunk payment between peers means that popular content will be widely distributed thereby increasing available bandwidth while decreasing latency

# Swarm Incentive System

Storage

# Swarm Incentive System

## Storage

The Problem:

I want to deploy my content only once:  
“upload and disappear”.

I want to make sure the content remains available  
years into the future even if it is not popular content.

# Swarm Incentive System

## Storage

The Problem:

I want to deploy my content only once:  
“upload and disappear”.

I want to make sure the content remains available  
years into the future even if it is not popular content.

Solution:

Pay certain nodes to keep your data.

# Swarm Incentive System

## Storage

The Problem:

I want to deploy my content only once:  
“upload and disappear”.

I want to make sure the content remains available  
years into the future even if it is not popular content.

Solution:

Pay certain nodes to keep your data.

-Nodes that sell such promises-to-store must have a  
deposit locked on the blockchain.

-Nodes that loose content, loose their deposit.

# Swear and Swindle

# SWEAR – Swarm Ensured Archival

- Nodes register with the SWEAR contract and pay a deposit.



# SWEAR – Secure Ways of Ensuring Archival

- Nodes register with the SWEAR contract and pay a deposit.
- Registered nodes can sell receipts for chunks received.

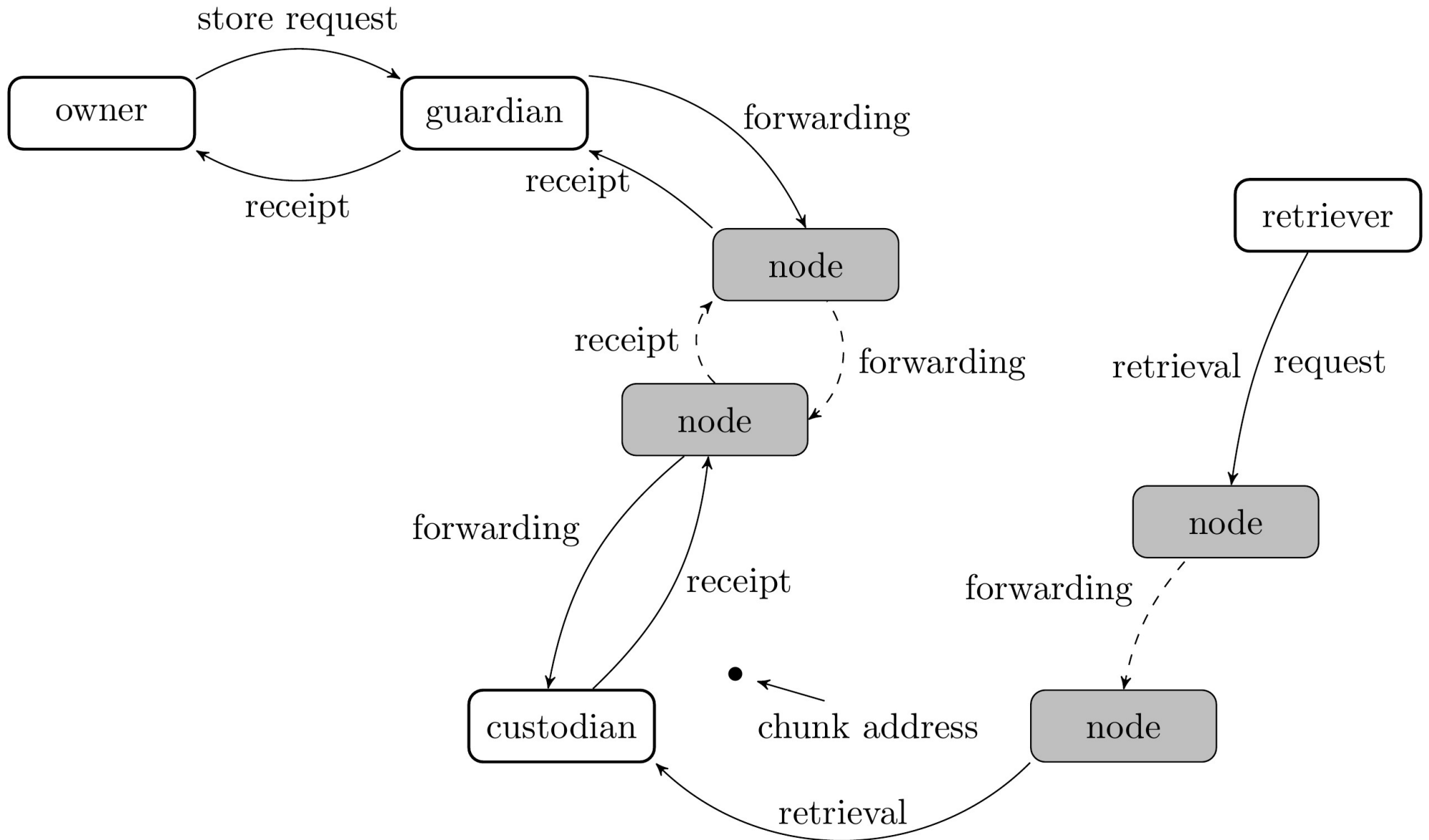
# SWEAR – Swarm Enforcement and Registration

- Nodes register with the SWEAR contract and pay a deposit.
- Registered nodes can sell receipts for chunks received.
- Receipts are promises that the data remains available in the swarm.

# SWEAR

- Nodes register with the SWEAR contract and pay a deposit.
- Registered nodes can sell receipts for chunks received.
- Receipts are promises that the data remains available in the swarm.
- “Upload and Disappear” made possible by the system of ‘guardians’

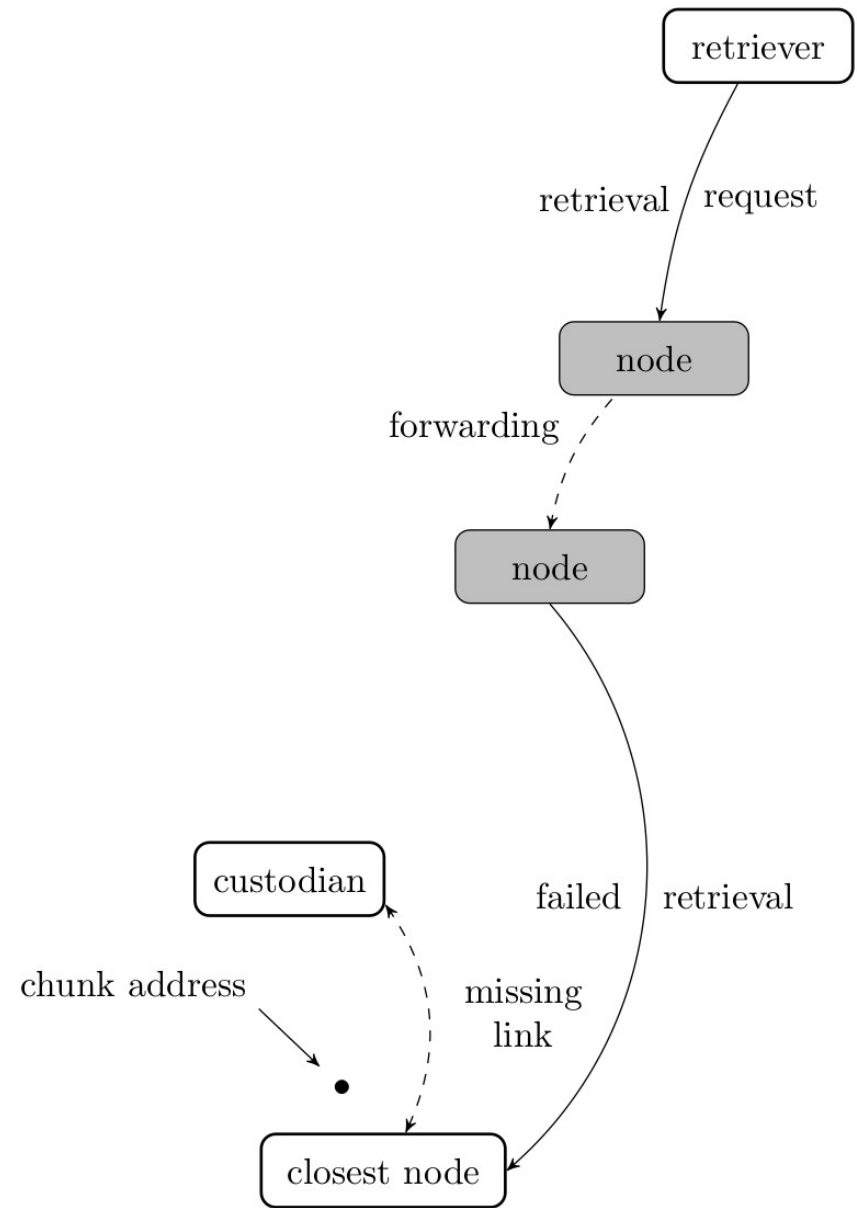
# Storing content in the swarm:



What if the data cannot be found?

# Example:

Chunk is not actually 'lost' -  
It is still in the swarm,  
but lookup fails because  
chunk never reached  
the closest node.



**SWINDLE**

**SWINDLE – Secured with Insurance Deposit  
Litigation and Escrow**



# SWINDLE

- Issue 'challenges' to the guardian to show proof-of-custody of the chunk shown in the receipt.

# SWINDLE

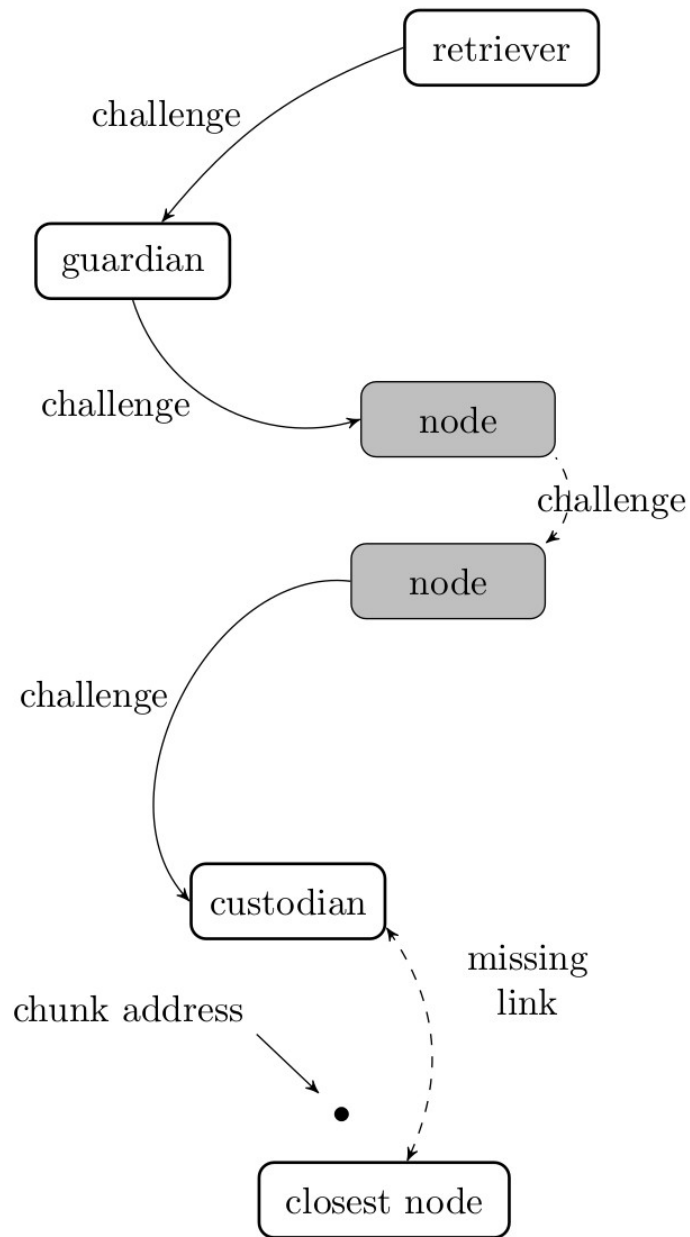
- Issue 'challenges' to the guardian to show proof-of-custody of the chunk shown in the receipt.
- Guardian can defend themselves by showing proof-of-custody or guardian will forward a challenge to the next node.

# SWINDLE

- Issue 'challenges' to the guardian to show proof-of-custody of the chunk shown in the receipt.
- Guardian can defend themselves by showing proof-of-custody or guardian will forward a challenge to the next node.
- Chain of receipts ends up with either
  - 1) A node storing the chunk (custodian)

# SWINDLE

- Issue 'challenges' to the guardian to show proof-of-custody of the chunk shown in the receipt.
- Guardian can defend themselves by showing proof-of-custody or guardian will forward a challenge to the next node.
- Chain of receipts ends up with either
  - 1) A node storing the chunk (custodian)
  - 2) A node that should have the chunk but lost it.



→ Retriever challenges guardian

→ Guardian challenges the node that it bought a receipt from.

→ Nodes forward challenges until the custodian is found.

# Results of Litigation

# Results of Litigation

- 1) The Custodian is found; the missing link is identified; the swarm is repaired

# Results of Litigation

- 1) The Custodian is found; the missing link is identified; the swarm is repaired
- 2) The Chunk is indeed lost and the offending node is punished (loss of deposit)





# Dealing with Data Loss

# Preparing your data with Erasure Codes

# Preparing your data with Erasure Codes

**Idea:** When preparing your file for the swarm – i.e. when generating the swarm chunk merkle tree – generate extra ‘redundancy chunks’ so that all data can be recovered even if individual chunks are lost.

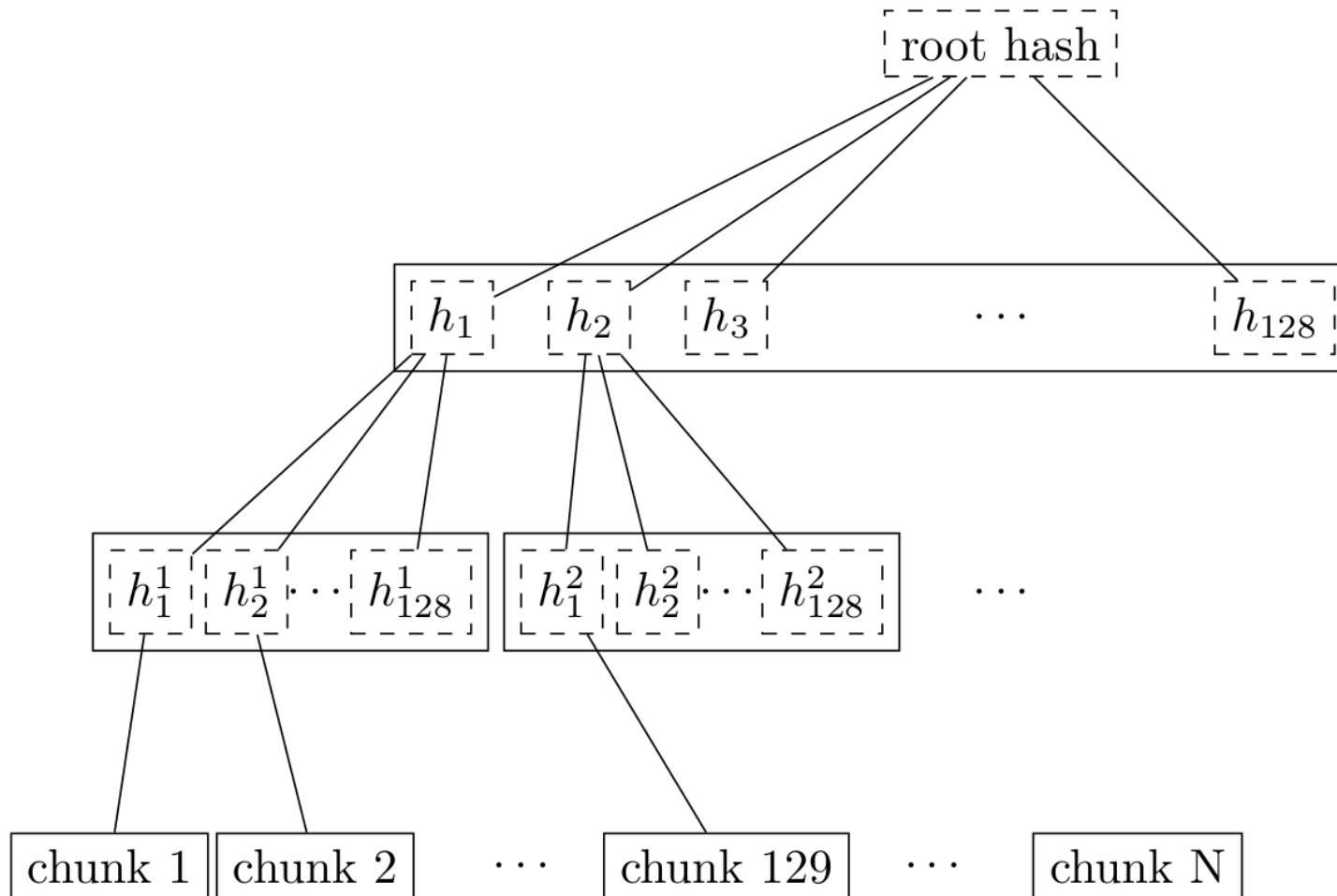
# Preparing your data with Erasure Codes

**Idea:** When preparing your file for the swarm – i.e. when generating the swarm chunk merkle tree – generate extra ‘redundancy chunks’ so that all data can be recovered even if individual chunks are lost.

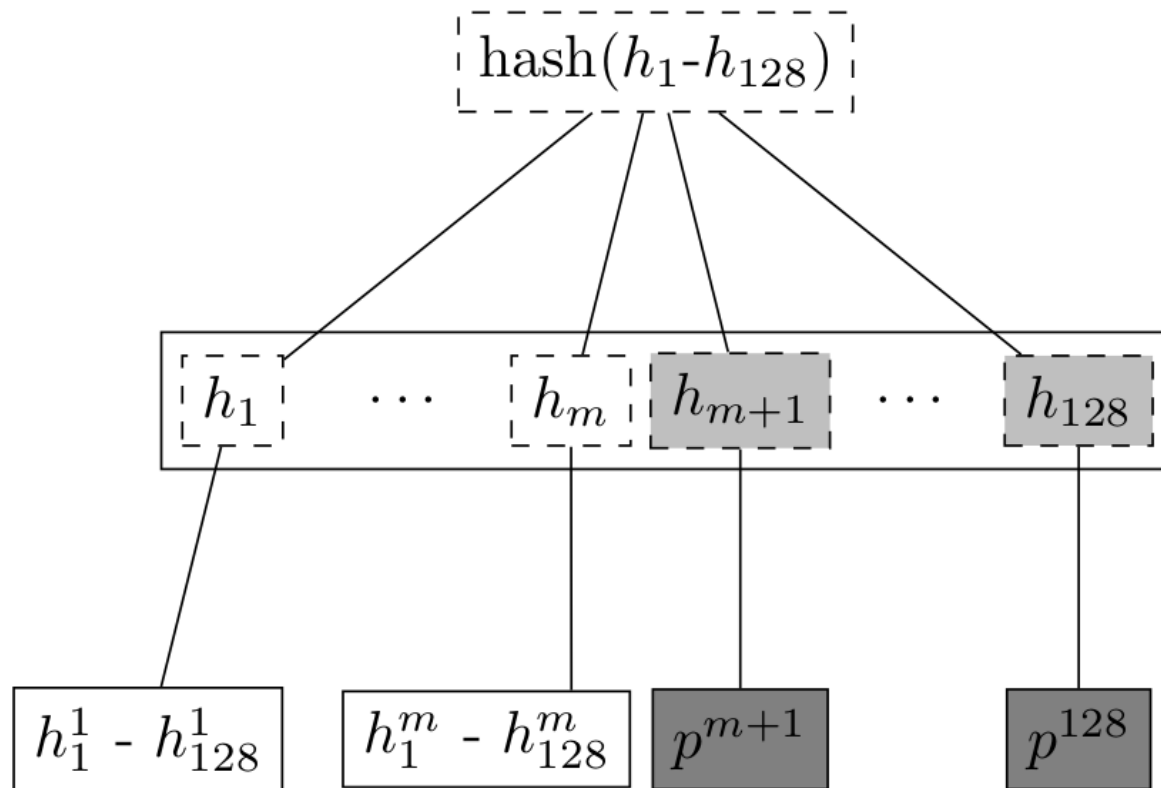
## Benefits:

- Owner can set their own redundancy parameters
- Swarm can repair itself following data loss

# Ordinary Swarm Chunk Merkle Tree



# Adding Parity Chunks via Erasure Coding



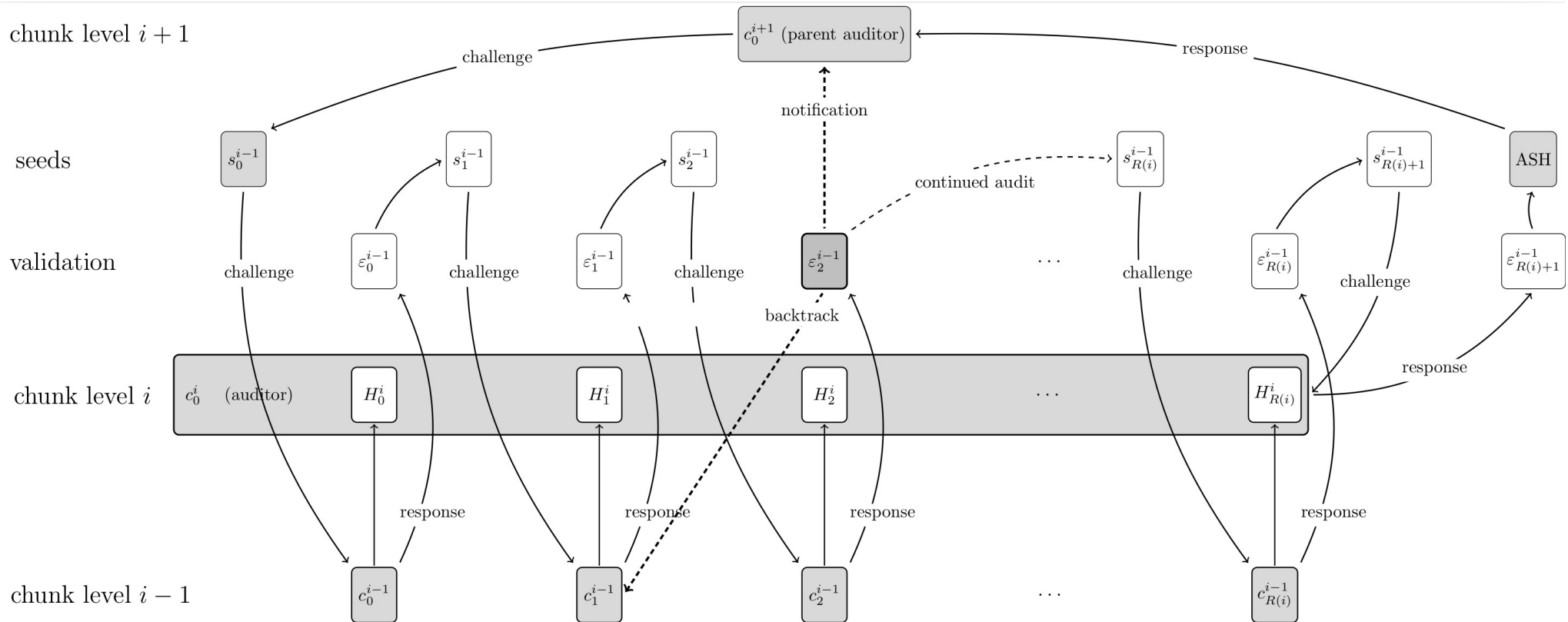
## Potential Benefits:

- All chunks in the tree are equally important for retrieval.
- **Any** node can repair swarm if data loss is discovered.
- Requesting **all** chunks (data + parity) can greatly reduce latency. This could lead to more responsive dapps.



- But Erasure coding is not enough, especially for large data sets you have to be able to monitor and repair.
- Swarm includes an audit system able to identify missing chunks.

- But Erasure coding is not enough, especially for large data sets you have to be able to monitor and repair.
- Swarm includes an audit system able to identify missing chunks.



- But Erasure coding is not enough, especially for large data sets you have to be able to monitor and repair.
- Swarm includes an audit system able to identify missing chunks.

- But Erasure coding is not enough, especially for large data sets you have to be able to monitor and repair.
- Swarm includes an audit system able to identify missing chunks.
- The same auditing system is used as a condition to periodically release payments for long-term storage agreements.

Idea: “each new payment requires a proof (audit) that the data is really still available”

**SWAP • SWEAR • SWINDLE**

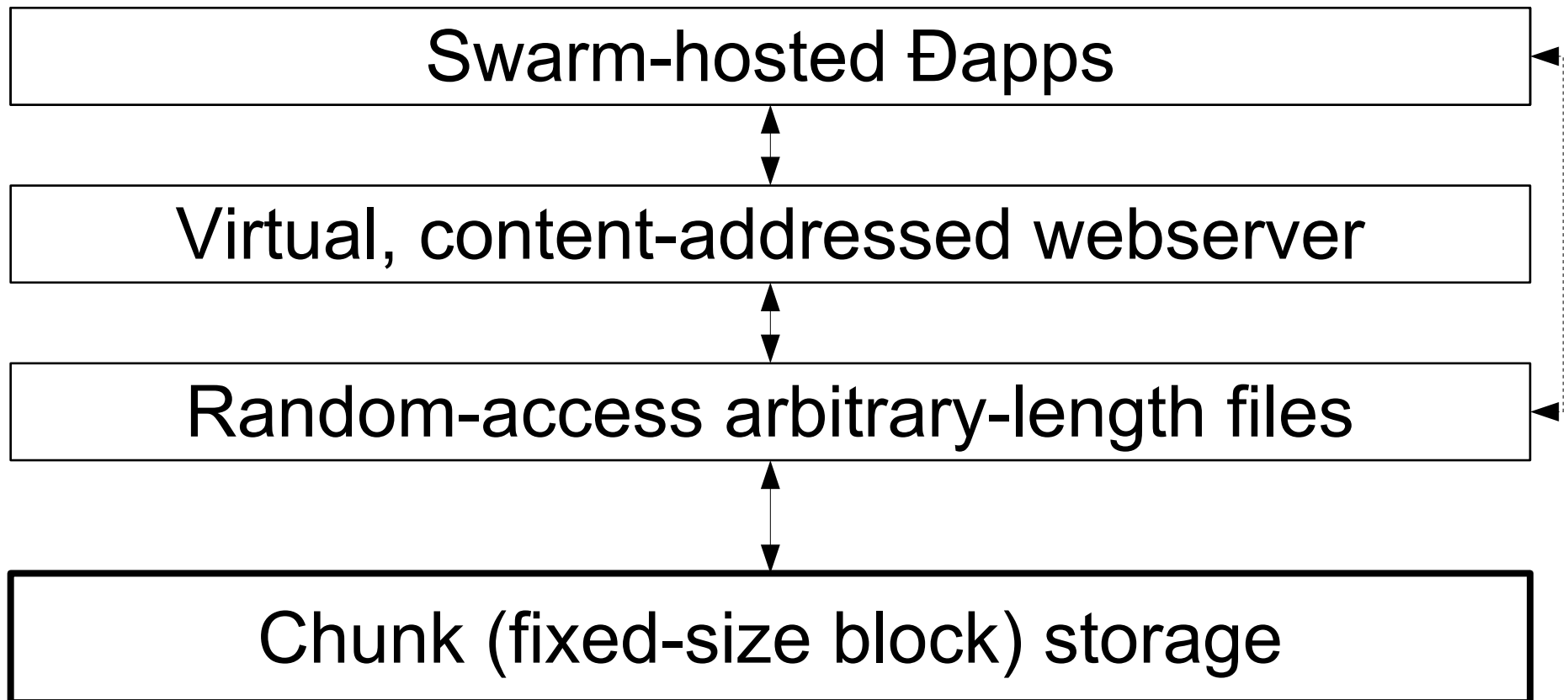


# The Web3 Experience



# swarm: Basic architecture

Well-separated layers connected by simple APIs:







# swarm: Web3 user experience

- Familiar: hypertext with multimedia in a browser
  - Interactive, responsive, intuitive
- Personalization and identity management
  - Selectable personae, identities
  - Part of browser, not application
- Legal and financial interactions
  - Binding agreements
  - Payment with provable receipts
  - Rate-limits, confirmations with passwords, etc.



# Swarm: Dapp mechanics

- Current root hash registered on block chain
- Most static and dynamic data in Swarm
- Global state changes on block chain
- Local state changes stored locally
  - Optionally backed up in swarm and/or block chain
- Business logic gets executed locally
  - But verified globally by means of Ethereum



# Swarm: Dapp example #1

## **distributed photo album**

- Web-app & data hosted in swarm
  - Root hash of collections published on block chain
  - Long-term incentives make sure it is not gc'd
  - Short-term incentives drive publishing costs down
  - High performance irrespective of popularity
- No concurrent editing
  - Each collection is only edited by one contributor
  - All editing is done by the editor's computer
- No comments or ratings



# Swarm: Dapp example #2

## **distributed social network**

- Personalization: list of followed contributors
  - E.g. friends contributing comments, likes, etc.
  - Their number is limited
- Content is rendered by traversing this list
  - For each post, friend list is scanned for comments to this post
  - Single root hash on the block chain for each participant, changes with editing, publishing, commenting
- Requests for following through block chain



# Swarm: Dapp example #3 distributed map/encyclopedia

- No “official truth”, forking is cheap
  - Alternative perspectives face no prohibitive costs
  - Continued “rebasing” keeps all versions up-to-date
  - No edit wars, no blackouts
- Groups or individuals can have own versions
  - Registered on the block chain by root hash
  - Requires some editor work, but not much
- Few versions of individual records
  - Opinions on any single topic tend to cluster around a few alternatives

What's next? (Roadmap)

SWARM