ethereum

vienna

Mist, Light Client
and Swarm

Agenda

General Introduction

Updates

Swarm: Decentralised Storage

Socialising

ethereum

vienna

Updates

Mist Beta

Not much extra functionality

No accounts exposed to dapp

Must be requested and manually confirmed

Example DApp "Stake Voice"

Mist Beta

DEMO

LES

Light Client is ready for testing

see 'Light Client Public Test' in wiki at
github zsfelfoldi/go-ethereum

Events don't work yet

Can use trusted headers for speedup

Requires **much less** space

LES

DEMO

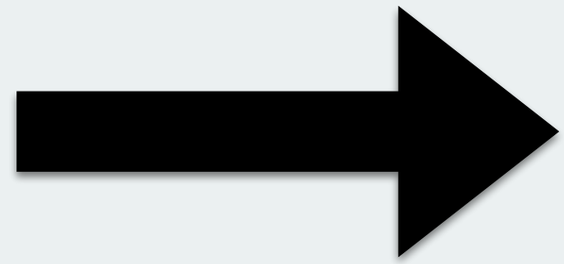
Workshops

Another Beginner Workshop in first half of September

Poll on the event page

ATM

Over there



but still doesn't work

(but something works)

Ethereum Fork

Geth 1.4.10 - 'Return of the ETH'

Chain split at #1920000

DAO refund contract

- can withdraw token from DAO (if authorised)
- releases 0.01 ether for every received token

Ethereum Fork

DAO Refund

- Multiple UIs available
- Some exchanges (polo) have conversion

Extra Balance + Child DAOs in curator multisig

should be resolved fairly soon

#childdaocommunity on thedao.slack.com

Ethereum Classic

Minority chain still alive

	Hard fork	Non-fork
Block	0x30b8c230	0xe5a0b897
Block Number	2022668	2020761
Difficulty	100.00%	22.45%
Total Difficulty	100.00%	12.40%
Block interval	14.5 sec	14.2 sec
Hash rate	3590.9 GH/s	828.4 GH/s

Ethereum Classic

ETC trading on multiple exchanges

Ethereum Classic

replay attack

tx valid on ETH and ETC if the balance checks out

split contract necessary to safely separate the funds

see ethereum blog

only if you care about both chains

some exchanges (polo) split automatically on deposit

Ethereum

1 year since genesis block (July 30th)

1 year since first tx today (August 7th, 3:00 am)

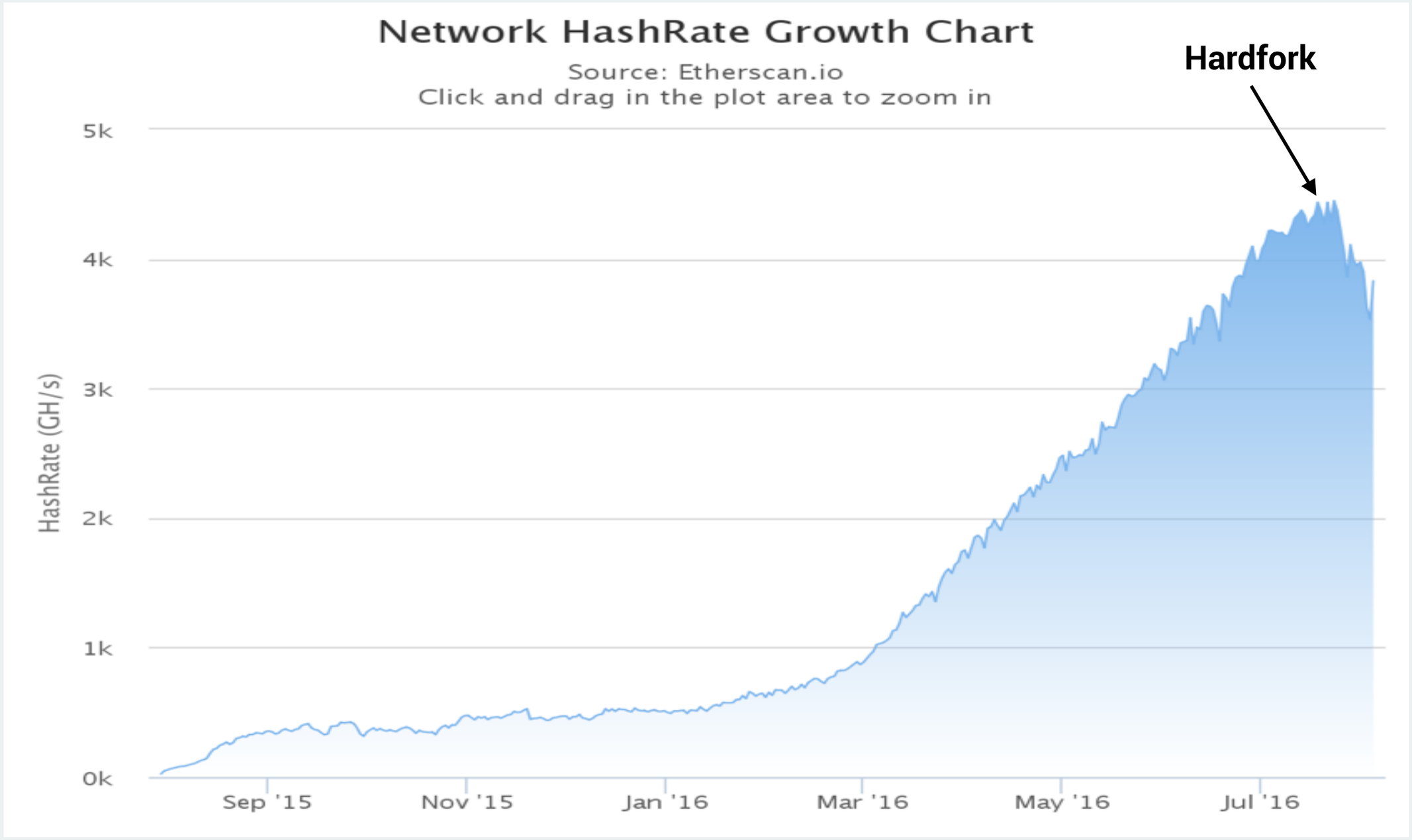
2 hardforks (1 unplanned) in the first year

8.1 million transactions

Ethereum


Transaction Chart
Source: Etherscan.io
Click and drag in the plot area to zoom in
80k
70k
60k
50k
40k
30k
20k
10k
0k
Transactions
Sep '15
Nov '15
Jan '16
Mar '16
May '16
Jul '16

Ethereum


Network HashRate Growth Chart
Source: Etherscan.io
Click and drag in the plot area to zoom in
Hardfork
HashRate (GH/s)
5k
4k
3k
2k
1k
0k
Sep '15
Nov '15
Jan '16
Mar '16
May '16
Jul '16

ethereum

vienna

Swarm

Why?

DApp Files have to be served in a decentralised way

Otherwise

Files might disappear

Server sends malicious code

Censorship

Goals

DDOS- resistant

zero-downtime

fault-tolerant

censorship-resistant

self- sustaining due to a built-in incentive system

dynamic scalability

Orange Papers

#1 swap, swear and swindle

incentive system for swarm

#2 smash-proof

auditable storage in swarm secured by masked audit secret hash

#3 state channels on swap networks (not released, Q3)

claims and obligations on and off the blockchain

Basics

based on devp2p

files are split into chunks of equal size

content addressed chunk store

3 core protocols

SWAP **SW**arm **A**ccounting **P**rotocol

SWEAR **SW**arm **E**nforcement **A**nd **R**egistration

SWINDLE **S**ecured **W**ith **IN**surance **D**eposit **L**itigation and **E**scrow

Basics

Every swarm node has an address (=enode?)

Same address space as chunks

Distance Measure possible

Chunks are requested from the closest node

Litigation mechanisms

MASH Masked Audit Secret Hash

SMASH Secured by Masked Audit Secret Hash

CRASH Collection-level Recursive Audit Secret Hash

SWAP

Pairwise accounting

Tally of sent and received services

= Sent and Received chunks

If balance goes to far in one direction => disconnect

Ether / Token can be transferred to restore balance

Price per chunk set at connection start

Persevered across connections

SWAP

Chequebook Contract

like a micropayment channel

sender generates cheques

Alternatively fully fledged payment channels

details will be published in paper #3

SWAP

Forwarders cache chunks

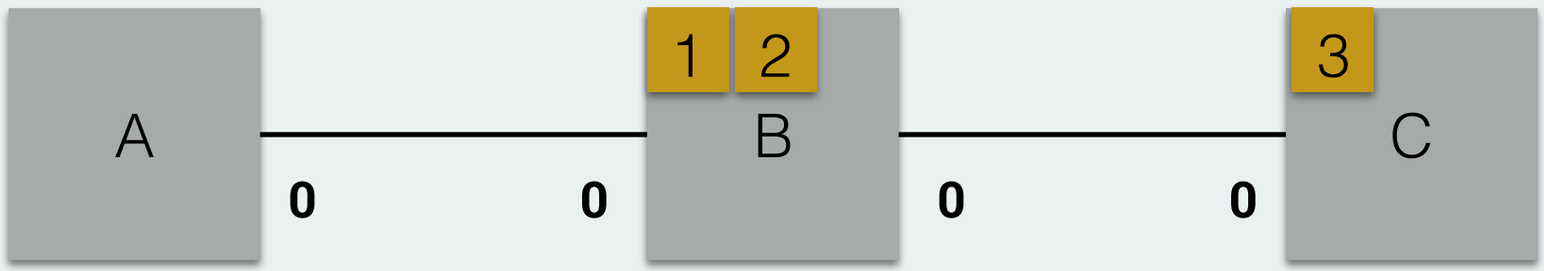
=> no need to repurchase popular content

prioritised by request frequency

Can keep popular content alive

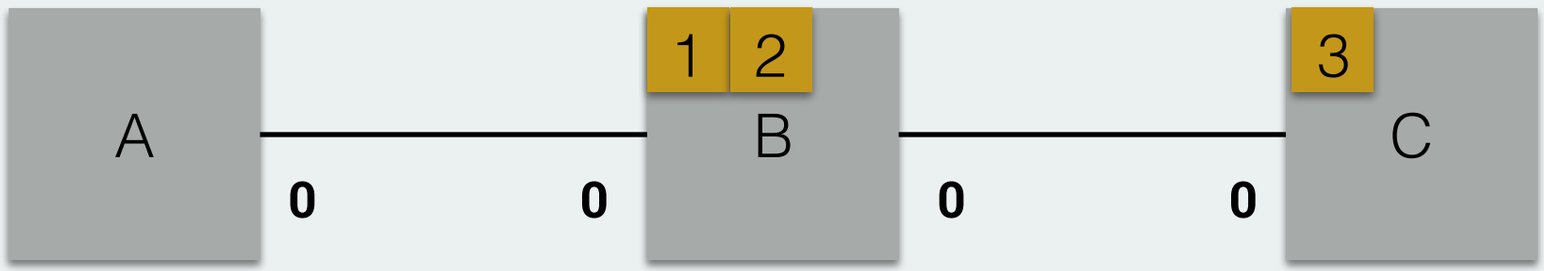
but different strategy required for other data

SWAP



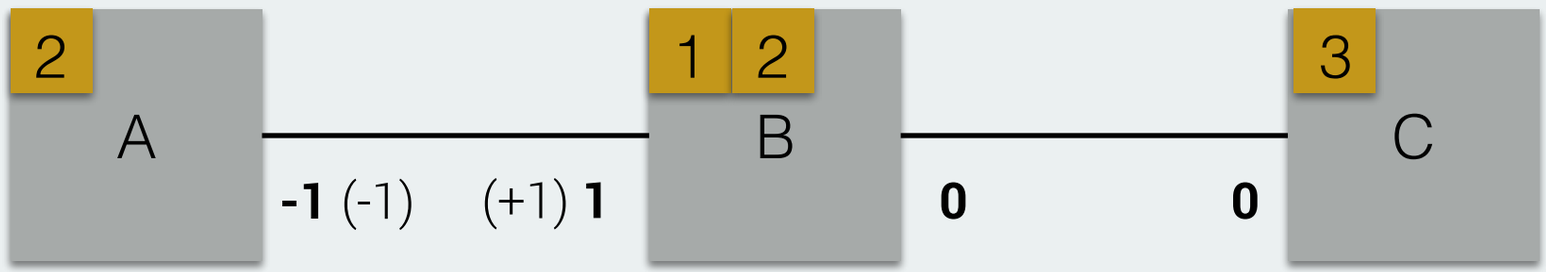
SWAP

A requests 2



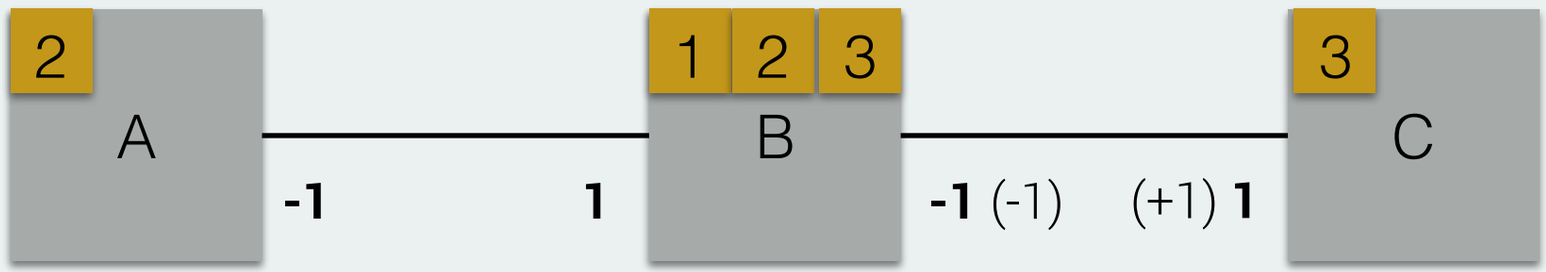
SWAP

A requests 2



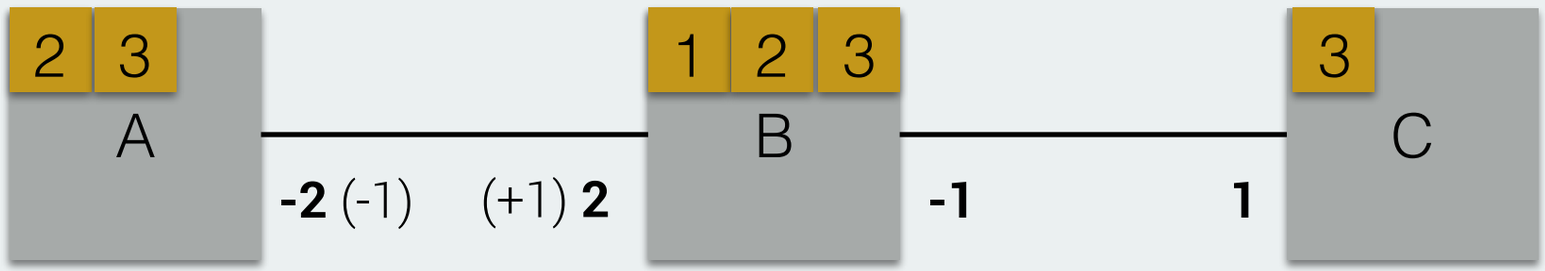
SWAP

A requests 3



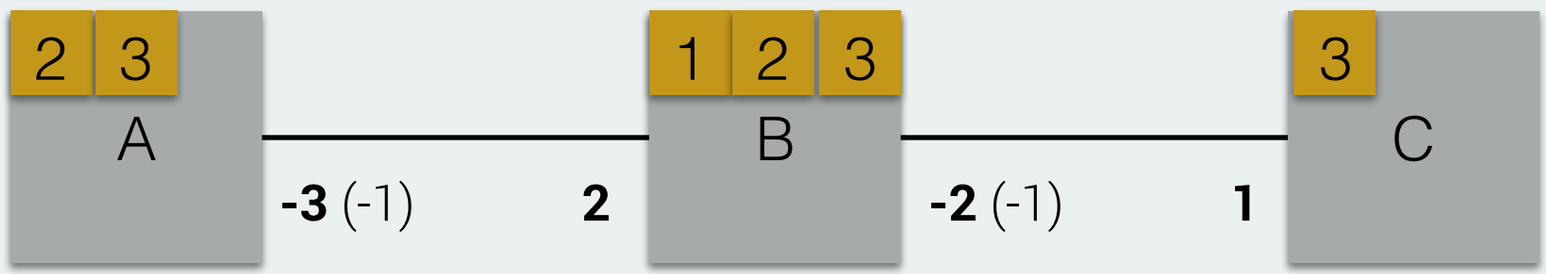
SWAP

A requests 3



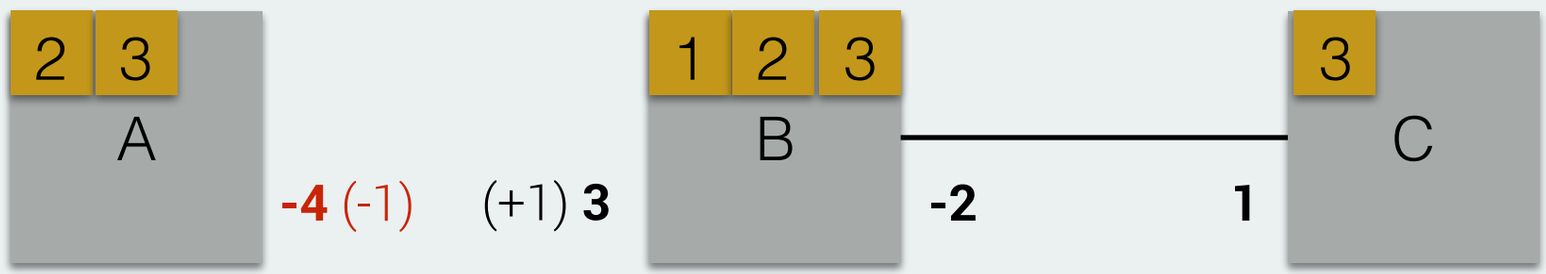
SWAP

A requests 4 (non existant)



SWAP

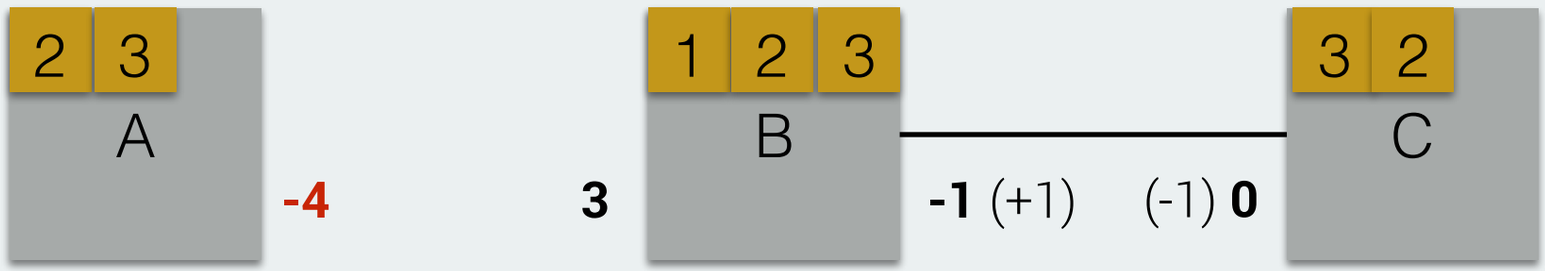
A requests 1



Threshold reached,
peer disconnected

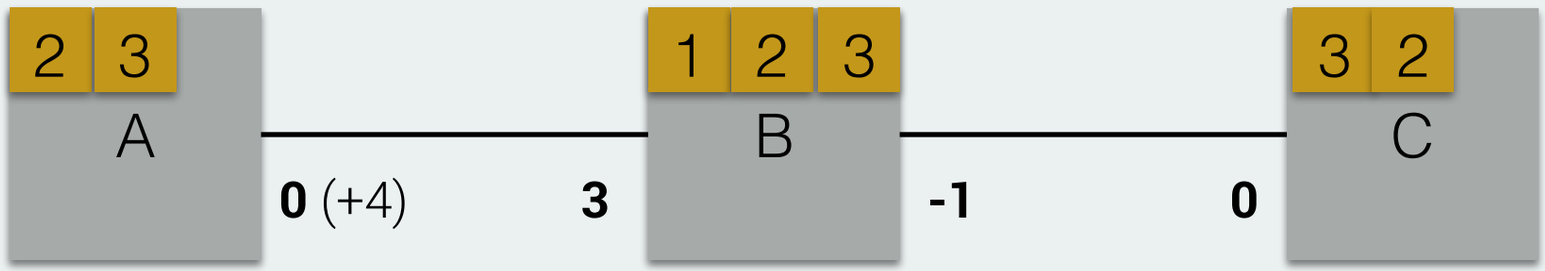
SWAP

C requests 2



SWAP

A pays for 4 chunks



Connection can be reinstated

Incentive Schemes

swap	swapping information syncing price negotiation auditing off-chain payments
swear	accountability with security deposits receipt generation on-chain enforcement
swindle	regular auditing policing conscientious litigation

SWEAR

Nodes may sell storage promises

Must register public key with SWEAR contract

Must send security deposit with it

Storer issues signed receipt

In case of lost data, deposit is mostly burned

SWINDLE

Node can put up a challenge for a chunk

Needs to present receipt to contract

Challenge valid if:

- receipt signed with key of a registered node
- receipt expiry date not passed
- sufficient funds for compensation

SWINDLE

Storer must reply with a refutation

Can be chunk data or **proof of custody**

Audits happen off-chain

=> Blockchain is a measure of last resort

If there are multiple results covering the same chunk

=> all storers lose their deposit if an audit fails

Forwarding

During syncing

chunk can be forwarded to a closer registered node

provable

Forwarders can play the blame game

Forwarding

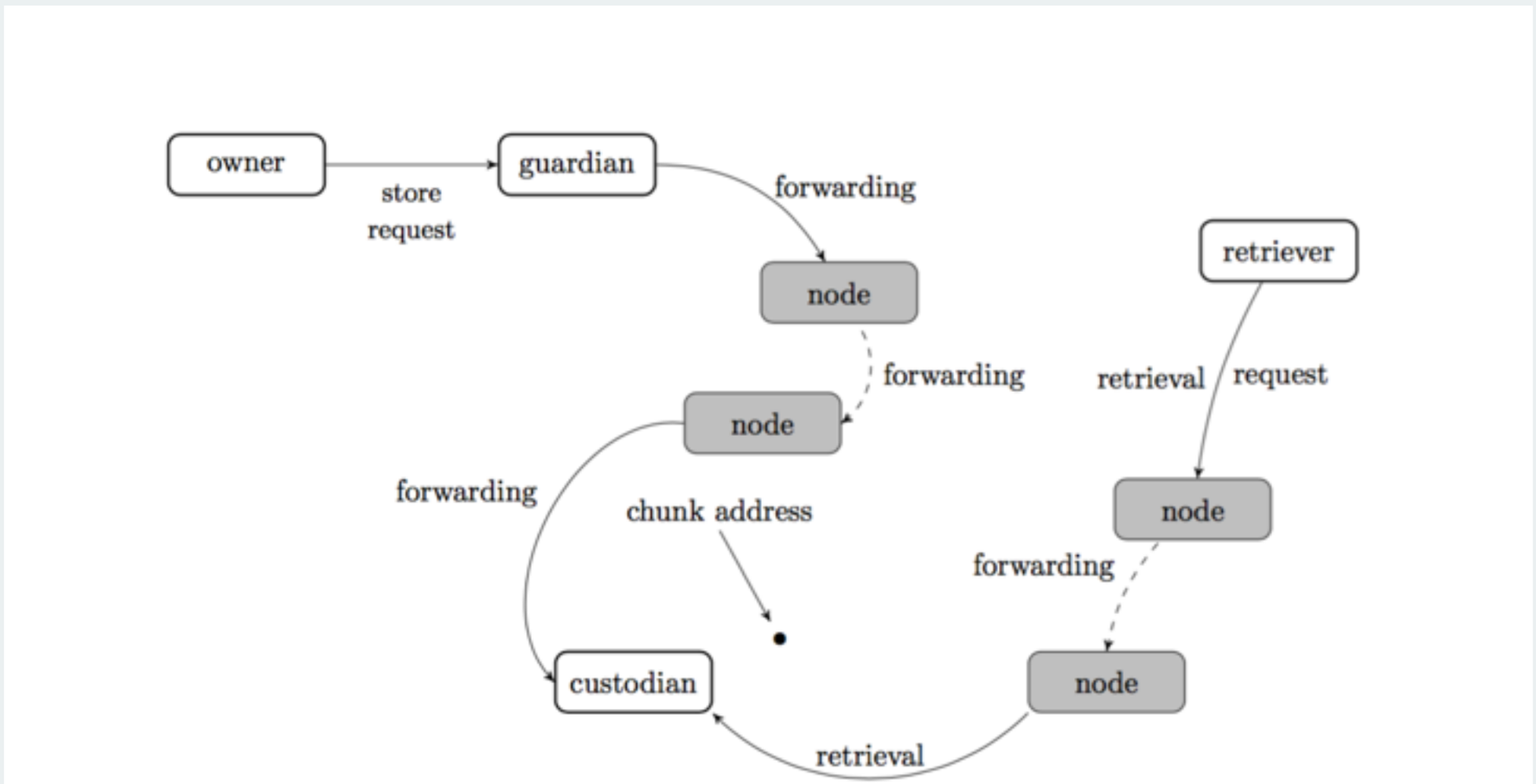
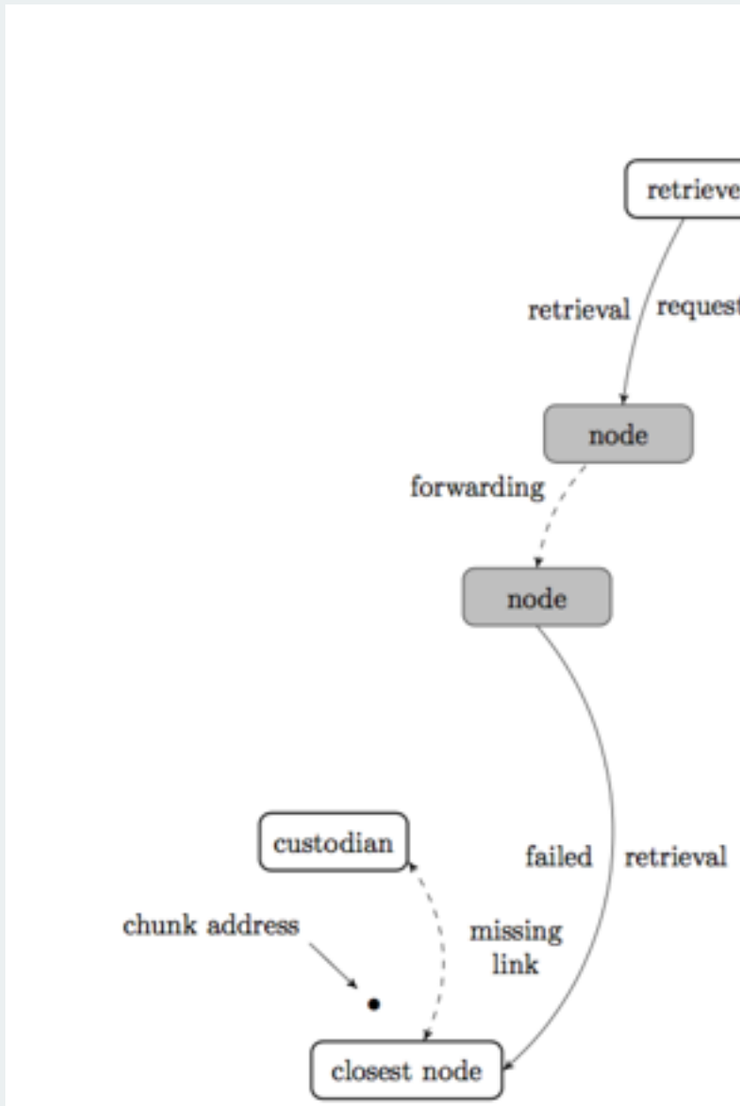
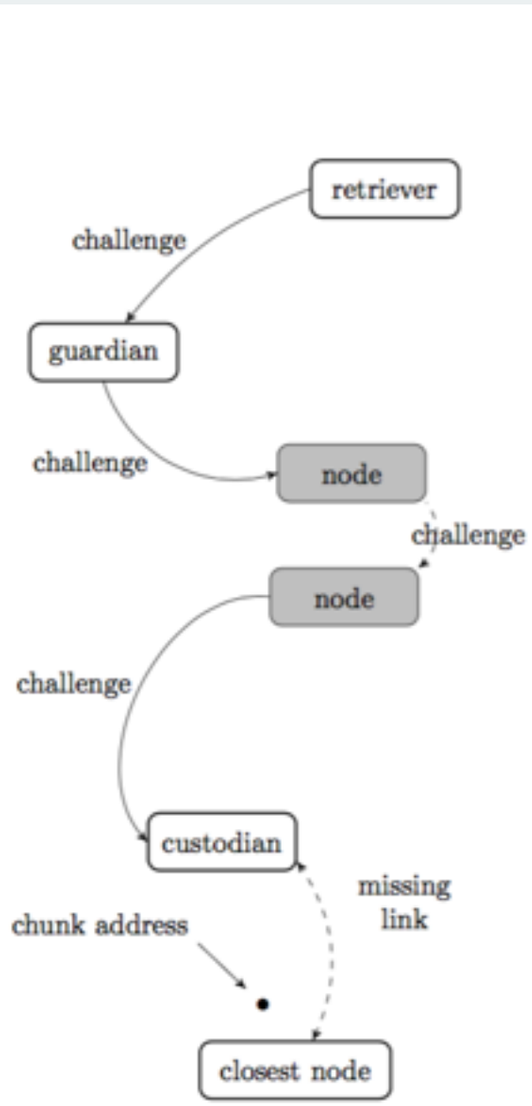


Fig. 2: The arrows represent local transactions between connected peers. In normal operation these transactions involve the farther nodes (1) sending store request (2) receiving delivery request (3) sending chunk (4) sending payment (5) receiving a receipt.

Forwarding


retriever
retrieval
request
node
forwarding
node
custodian
failed
retrieval
chunk address
missing
link
closest node


retriever
challenge
guardian
challenge
node
challenge
node
challenge
custodian
missing
link
chunk address
closest node

Erasure Codes

Redundancy encoded in document before uploaded

m-out-of-n CRS coding

m of n chunks can reconstruct the data

any client can detect lost chunks and resync them

Proof of Custody

Owner needs to reveal a seed

Storer generates secret from seed and chunk segment

Secret is masked (=hashed) and made public

=> masked audit secret hash (MASH)

MASH

Chunk split into segments

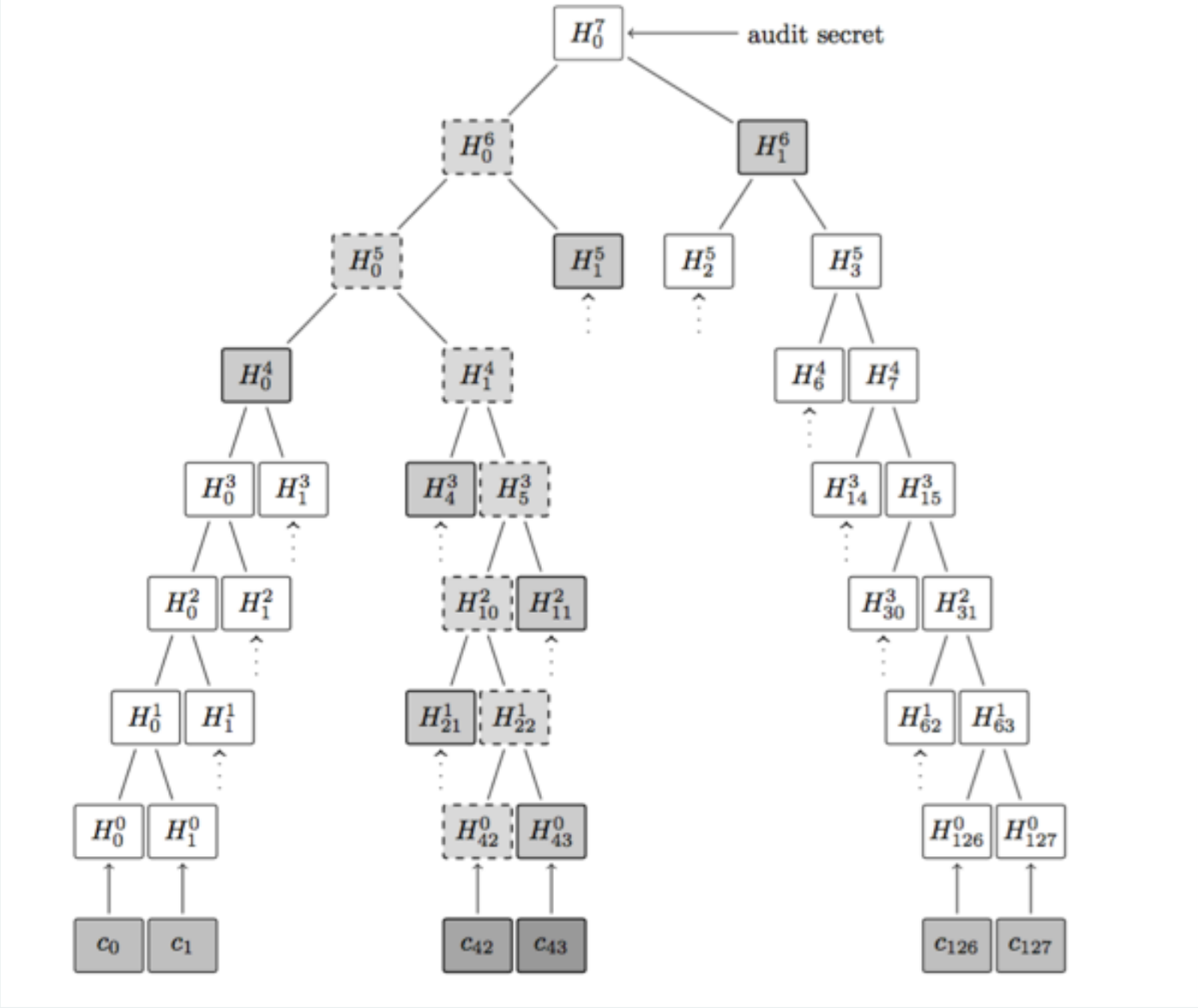
Target segment replaced by hash(segment || seed)

ASH = Root of binary merkle tree over segments

MASH = hash(ASH)

Smash Chunk Hash = Root over unmodified segments

MASH


H_0^7
audit secret
H_0^6
H_1^6
H_0^5
H_1^5
H_2^5
H_3^5
H_0^4
H_1^4
H_6^4
H_7^4
H_0^3
H_1^3
H_4^3
H_5^3
H_{14}^3
H_{15}^3
H_0^2
H_1^2
H_{10}^2
H_{11}^2
H_{30}^3
H_{31}^2
H_0^1
H_1^1
H_{21}^1
H_{22}^1
H_{62}^1
H_{63}^1
H_0^0
H_1^0
H_{42}^0
H_{43}^0
H_{126}^0
H_{127}^0
c_0
c_1
c_{42}
c_{43}
c_{126}
c_{127}

MASH

Masked secrets can also be stored in a merkle tree

=> MASH tree

Owner only remembers MASH root

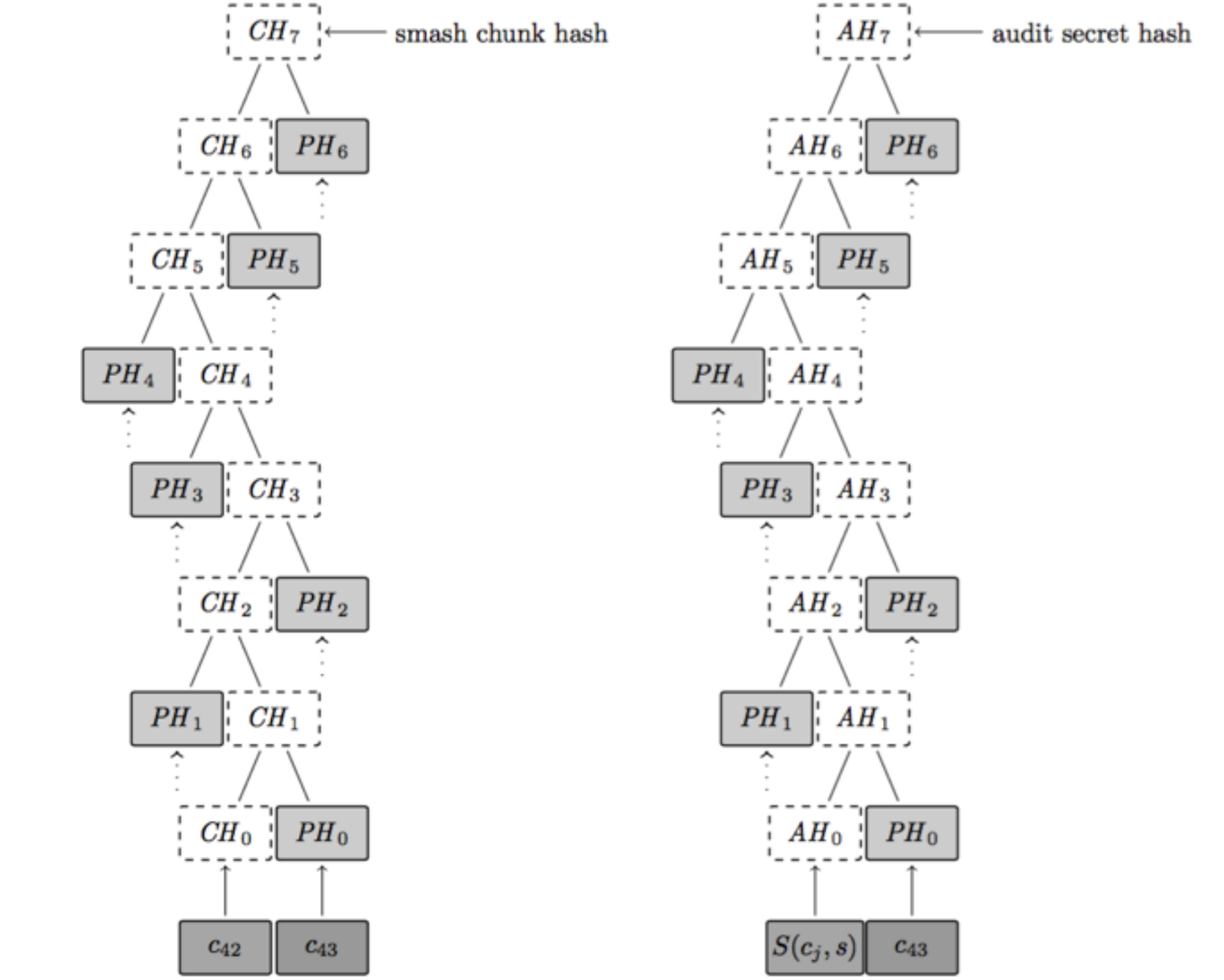
Response with merkle proof from secret to MASH root

=> MASH proof

smash proof = merkle proof of the relevant segment

necessary if the audit request was fraudulent

MASH


CH_7
smash chunk hash
CH_6
PH_6
CH_5
PH_5
PH_4
CH_4
PH_3
CH_3
CH_2
PH_2
PH_1
CH_1
CH_0
PH_0
c_{42}
c_{43}
AH_7
audit secret hash
AH_6
PH_6
AH_5
PH_5
PH_4
AH_4
PH_3
AH_3
AH_2
PH_2
PH_1
AH_1
AH_0
PH_0
$S(c_j, s)$
c_{43}

CRASH

Collection-level recursive audit hash

MASH for everything generates too much overhead

Hash for entire document collection from seed

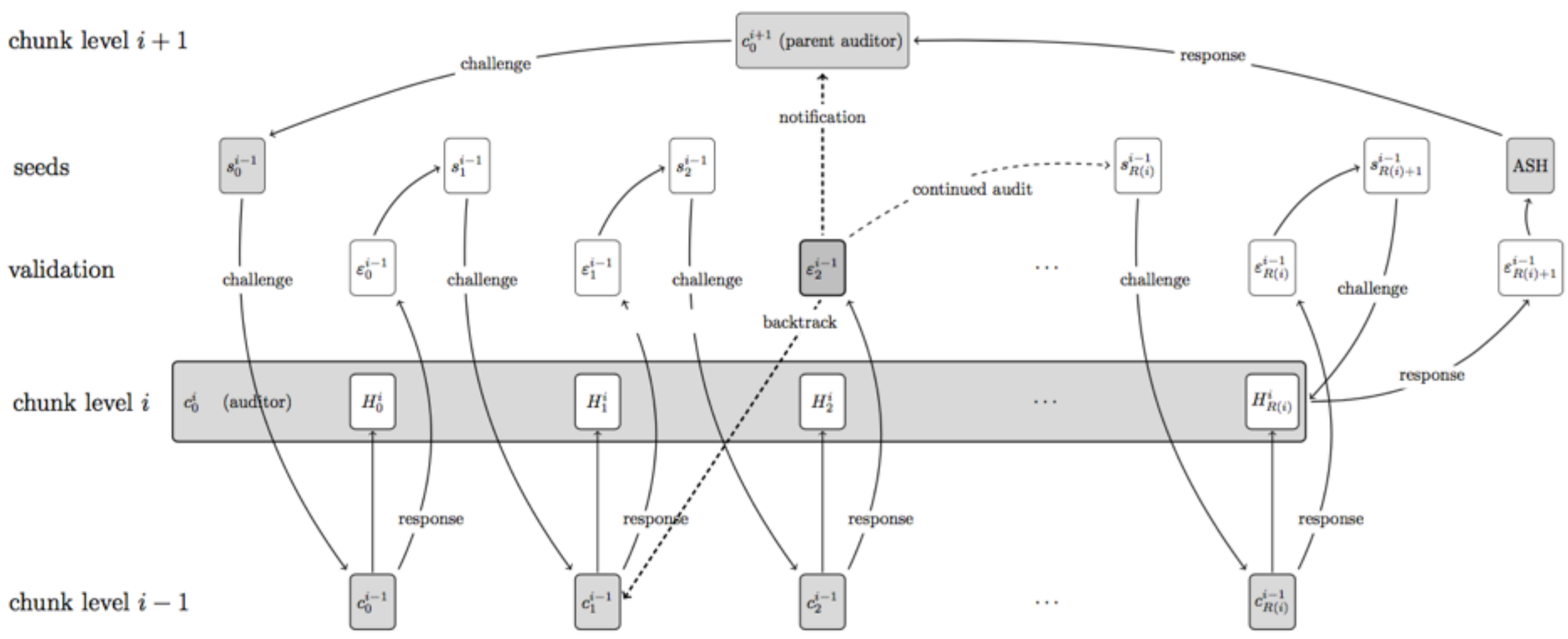
Fraudulent audits cannot be proven

=> not used for litigation

CRASH

Audit Manifest generated for document collection

Can be given to insurer



Progress

go-ethereum repository 'swarm branch'

hf code not merged yet (etc only)

working swarm gateway (orange papers in swarm)

SWAP seems to be (partly?) implemented

No SWEAR/SWINDLE implementation

No CRASH implementation

github.com/ahirner/ethereum