swarm

swap, swear and swindle

incentive system for swarm

preliminaries

- content addressed immutable chunk store - all other levels derived - merkle chunk tree - manifest
- devp2p network - semipermanent peerpool
- R-kademlia key based routing with forwarding
- content-agnostic storage - implausible accountability

swap: swarm accounting protocol with swift automatic payments and payment by dept swap

- retrieval -> replication, autoscaling elastic cloud
- primary positive incentive: bandwidth compensation by rewarding retrieval
- pairwise accounting: service for service or delayed payment
- verifiable cheques - chequebook contract per node
- basically reputation system based on credit history
- price agreement and trade off trust for transaction cost

storage incentives

- genuine upload, distribution aka syncing = propagating content towards closest nodes
- shortcoming of retrieval incentives: delete unrequested chunks, need extra incentive, insured archiving
- existing proposals: positive incentive only: periodical outpayments pending on some proof of custody
    - storj/metadisk - proof of storage with pregenerated audits
    - ipfs/filecoin: mining with proof of retrieval
- lots of problems, does not scale - pros and cons

swap, swear and swindle

- swap - share with a peer
    - chain of local interactions: sell-on receipts, chunk forwarding (syncing)
- swear - secure ways of ensuring archival - swarm enforcement and registration - storage with enforced archiving rules
    - contract to register/suspend nodes, handle deposit, verify challenges, refutations
- swindle - swarm insurance driven litigation engine
    - litigation procedure - probing - challenge the challenge

benefits

- blockchain usage minimised, litigation last resort
- contracting (passing storage receipts) and syncing one and the same protocol, no spurious receipt traffic
- zero delay litigable agreements
- automatic adaptation to network growth and shrinking
- local pairwise accounting - part of swap
- flexible handling chunk TTL vs membership expiry

shortcomings

- issue with variable pricing and deposits
- expressing degrees of redundancy
- tragedy of commons - oversubscription
- receipt storage
- lack of price discovery/market

loss-tolerant merkle trees

- variable degree of redundancy entirely in owners hands - coding theory to the rescue
- systemic n out of m loss-tolerant encoding on each level of merkle tree
- still allows (partial) integrity validation, streaming, if parity chunks are used immediate repair is possible
- normal operation if no loss: parity chunks are not retrieved, optimal network traffic and replication.
- loss-tolerant merkle tree chunking offers flexible per-user setting of degree of redundancy, and relieves sw^3 scheme from variable deposit handling, as well as complex reserve deposit systems
- unexpected bonus: competitive retrieval (like ex-skype) allows trading off horizontal redundancy for vertical, ie., improved latency and resilience to unsaturated kademlia tables
- shortcomings: root chunk storage ?

tentative solutions

- tragedy of commons - oversubscription limited by compulsory syncing - how many chunks you store is actually globally limited

- it is possible to control prices by voting based on oversubscription rate (random subset of registered nodes vote on global price storage per chunk per epoch)

- receipt storage only issue for non-storer nodes, off-chain probing helps building trust, allow pact (instead of challenge, submit signed sync token), jointly responsible but only closest node needs to keep actual receipts

- receipt packaging for documents and collections, recursively ensured and redundantly coded

- toplevel storage maybe use a few permutations of 128 hashes and register on blockchain