Hey, I have not posted on here for a while. I would like to introduce a project we have been working on. i think that some of the distributed systems problems we've encountered will ring bells with other projects in this space. We'd love to hear from anyone working on distributed storage, or on group governance, who is interested in a collaboration or exchange of ideas.

Cobox - a multi-writer distributed filesystem

Cobox is a research-driven project, responding to the identified need of small organisations for a collaborative file store with a focus on privacy, permissions and data integrity, without reliance on a single provider.

Cobox is built on the hyper-* stack, and extends on hyperdrive. A hyperdrive is a filesystem abstraction based on append-only logs. You can think of it as like a modifiable Bittorrent archive, where changes to files are replicated by peers in real-time.

For those more familiar with IPFS, this is somewhat like IPNS - we refer to the archive by the public key used to sign updates to the content, rather than by the content itself.

Kappa architecture

Hyperdrive works great when a single author wants to update an archive which has multiple readers - such as when publishing a website using 'Beaker browser' - which bears some similarity to using IPNS for hosting a website.

But what we want is a collaborative filesystem, where multiple peers or devices can make changes. It is not possible for multiple writers to sign changes to the same feed, because we may end up with incompatible 'forks'.

Cobox gets around this by presenting a set of hyperdrive archives, one from each peer or device, as a single archive. Your own hyperdrive contains only the files which you have personally modified, but what you 'see' is an aggregation of everybody's hyperdrives, taking the most recent version of each file. The filesystem is mounted using FUSE and behaves just like a normal directory - the notion of multiple archives is 'behind the scenes'.

This is made possible thanks to some great work on kappa architecture by the developers who wrote 'cabal'. Cabal is a distributed chat app - like IRC but with no servers. It also uses this technique of aggregating a set of independent feeds and presenting them as a chronological conversation.

Dynamic, encrypted backups

Another feature we wanted was the possibility to have certain peers replicating the filesystem without having access to it. This way, dynamic backups could be made by other organisations without issues around privacy. Normal hyperdrives are encrypted over the wire, but not on disk. So we've added on-disk encryption, and the result is that groups using Cobox are able to provide backups for others. Making it a kind of self-sustaining network without a single provider.

Conflict resolution using Vector Clocks

I wanted to mention conflict resolution in this post because it is a general problem in distributed systems which probably others have experience with.

It is an inherent issue with high latency, offline-first, peer-to-peer systems, that there is no 'single source of truth' by which we can infer the chronology or 'correctness' of a given event. In our case, if two peers both make a modification do a file, how to we decide which to take as the 'newest', current state of the file.

Cobox is not designed for actively collaborating on files, but more for archiving. For active collaboration there are much more sophisticated version control systems like Git. However we anyway needed some kind of strategy for these cases.

The thing which makes this problem hard is that there is not really a 'right answer' by which to compare a given technique. Using timestamps is inherently problematic due to system clocks not being synchronised. But even if they were, the most recent change might not be the best.

If a peer has been offline for sometime, they might make a change to a copy of a file which is not the latest version. They might have the 'newest' change, but by publishing it they will revert somebody else's change. Similarly in Cabal, the latest submission to the conversation might not be responding to the comments preceding it if the peer had not seen those previous event.

So we are using vector clocks, where time is measured logically in terms of 'number of events experienced'. Each peer maintains a 'vector' composed of their subjective clock for themselves and each other peer.

[

vector clock diagram

800×441 47.9 KB

](https://ethresear.ch/uploads/default/original/2X/8/818b85a7d854b97ddc51de6dcaadab38ac4c4541.png)

When a peer experiences an event (a file system change in our case) they increment their own clock value by one, and publish their version of the vector.

When we read another peer's vector (their personal record of how many events each peer has knowledge of) we merge it together with our own vector by taking the maximum known value for each peer.

When we need to decide which proposed version of a file to assume as the 'current state', we choose the proposal from the peer who has the highest value on our vector clock - the peer who we believe has the most knowledge of previous events is the winner. This gives us a notion of time based on causality, and is robust to situations where there is not ubiquitous connectivity between peers.

A recent development in this area is described in the 2019 paper 'The Bloom Clock' by Lum Ramabaja. This addresses the scalability issue of storing large vector clocks, replacing them with a probabilistic model based on Bloom filters. In our case we decided not to introduce Bloom filters, since Cobox is designed for small, closed groups. But for use-cases with potentially many peers, 'Bloom Clocks' could be a great conflict-resolution system.

Addressing the problem of conflict resolution allows us to benefit from a major advantage that the peer-to-peer model gives us over other types of network filesystem. Access times are fast, regardless of network connectivitiy, and the system continues to function when offine.

Hardware device

Cobox's final ingredient is a hardware device which serves to guarantee uptime and allow groups to provide dynamic backups for other groups. We are using the Olimex Lime2, a low power ARM computer with a SATA connector, which is completely open-hardware.

The device acts as a 'seeder', replicating archives just as any other peer, but without having the encryption keys.

Agreements

Our organisational model focusses on trust-based 'agreements' for providing backups, rather than smart contracts or legal contracts. We'd love to hear your opinions about this!

–Peg from Magma Collective.

References

- Cobox MVP Report
- Cabal - an experimental p2p chat platform
- Hypercore protocol
- kappa-core database indexing
- The Olimex Lime2
- Lum Ramabaja, 'The Bloom Clock' 2019