Collective Memory: Long-term storage and Sharing of PDFs

Ryan Kwon '17
Williams College
CS 339

1. INTRODUCTION

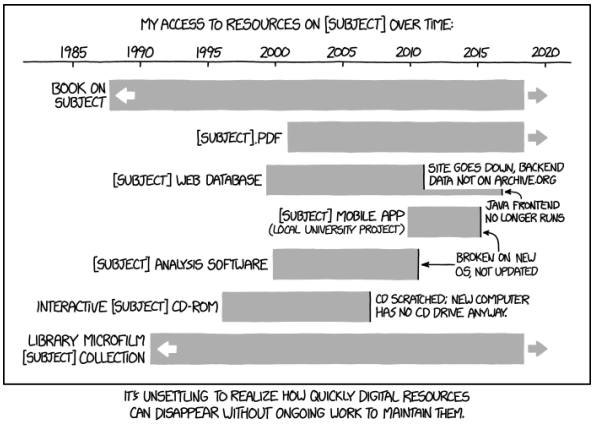

Figure 1. Scroll-over text: I spent a long time thinking about how to design a system for long-term organization and storage of subject-specific informational resources without needing ongoing work from the experts who created them, only to realized I'd just reinvented libraries.

Collective Memory was rationally justified after the fact by the above XKCD comic. Practically, our motivation is drawn primarily from the desire to build a cool decentralized system.

Collective Memory is a decentralized system with the goal of optimizing availability and longevity of uploaded files. This means that, once a file is uploaded and disseminated, the system is designed to make sure that file never goes away so long as the network has sufficient capacity to hold it. This goal creates a number of concerns: how do we ensure that the content that gets uploaded is "good"? How do we coordinate replication of files to make sure that we're robust to random and correlated node drops?

This checkpoint will discuss our progress so far.

2. Progress so far

- **Node/Network discovery Implemented:** We use multicast at a specified address and port to find network nodes. If no responses, we assume the network is dropping multicast packets (common in IPv4, though I think IPv6 supports multicast), so we fall back on a list of hard coded IP addresses and ports to ping. If none of them are reachable, we set up our own network.
- **Shepherd/Supernode model is integrated:** This is explained in a later section.

3. Things left to do

- **You should be able to specify an IP address and port to connect to.**
- **CLI interface**
- **Need to implement file-sending:** I think this should be a cinch. At that point, all the other things are pretty small.
- **Need to set up replication strategy:** This should be reasonably simple though. Most PDFs we're interested in will be fairly small (1-10 MBs likely), so we can make a reasonable guess on how many nodes will fail at once, and then replicate enough times such that we can always recover a copy if those nodes all fail.
- **Networks need to be able to reorganize themselves, so that they more logical sense:** Specifically, we aim to make subnetworks around Autonomous Systems (AS) where most of the logic will happen. This is logically simple, but I just need to write it up.

- **Need to think more on how to keep a network in the same AS from breaking up.**

- **GET and PROPOSE protocols need to be implemented**

4. Features we won't have

We won't have a "search" functionality. Although if you're ever exposed to a file, you should continue to have access to it while the relevant subnetworks are big enough to hold it. Uploaded file metadata and where they may be found are disseminated through a gossip-like protocol.

5. Node Protocol Commands

- GET: Get's a specified file given its metadata. The file is downloaded to a specific directory on the user's machine.

- PROPOSE: Propose that a file be uploaded. The proposed file goes to the shepherd, who needs to manually accept it. There is a cap on the number of files a node can propose.

6. Shepherd Model: Supernodes

The Shepherd is a supernode, gaining additional responsibilities and powers over other nodes in their AS/"flock." Every AS in a Collective Memory graph has a single shepherd (but if there are two Collective Memory networks in the same AS with no knowledge of each other, they will each have their own shepherd). For example, I'm currently the shepherd for the Williams network. Specifically, shepherds have these powers and responsibilities:

- Accept or reject "Proposed" files for upload from nodes in their AS.

- Replicating uploaded files among nodes in their AS for redundancy.

- Sit in the multicast address and accept new nodes in their AS into the network.

- Introducing nodes in their AS to each other, so that the shepherd can be replaced if she/he ever dies.

- Accept or reject files from another shepherd.

6.1. Motivation

There were three concerns that encouraged the shepherd model.

- **Efficiency**: Shepherds may be used to cache files, and their existence imposes a tree strucure on the Collective Memory network graph. Most nodes will only know about other nodes in their AS, as well as some shepherds.

- **Simplicity**: Coordinating efficient and effective replication among a bunch of equals sounds like a nightmare. Having a single node responsible for coordinating this is much simpler.

- **Security**: Promising longevity on random files uploaded by total randos is also a nightmare. Having a shepherd will hopefully reduce the number of bad files that gets uploaded to the network. Furthermore, having a bunch of nodes respond to you when you say "Hello! I'd like to join!" also struck me as a great DDoS resource. Having only the shepherds respond should drastically cut down on network hogging.

6.2. Subtleties

This section contains some extra subtleties on Shepherd behavior. This is mostly to remind myself. It will be rewritten in a more understandable format in the future.

- Isolated nodes cannot create their own nonsense by creating a network by themselves, making themselves a shepherd, and then joining the bigger network. File metadata will contain information on known shepherds when accepted. When "proposed" to another shepherd, the receiving shepherd will compare the list of shepherds extant at the time to their list of known shepherds, and reject if the file seems to have been accepted in intentionally isolated conditions.

- When a shepherd dies, nodes in its AS will become aware as their pings won't get responses. These pings are randomly distributed to occurr every 1-10 minutes. They'll then look at the multicast address to see if the new shepherd is there. If not, they become a shepherd and sit at the multicast address. If two nodes become shepherds of the same AS, they'll find each

other, and the shepherd with the lexicographically lower IP address resigns and becomes a normal node. This is also to help ensure that it's difficult to become shepherd. The position is life-long, and when lost, goes to the person with the highest IP address. Something that's hopefully not super-easy to just take up.

7. Thoughts

Also Jeannie, I want to comment that I'm really, really enjoying this project. I'm having a lot of fun building this system.