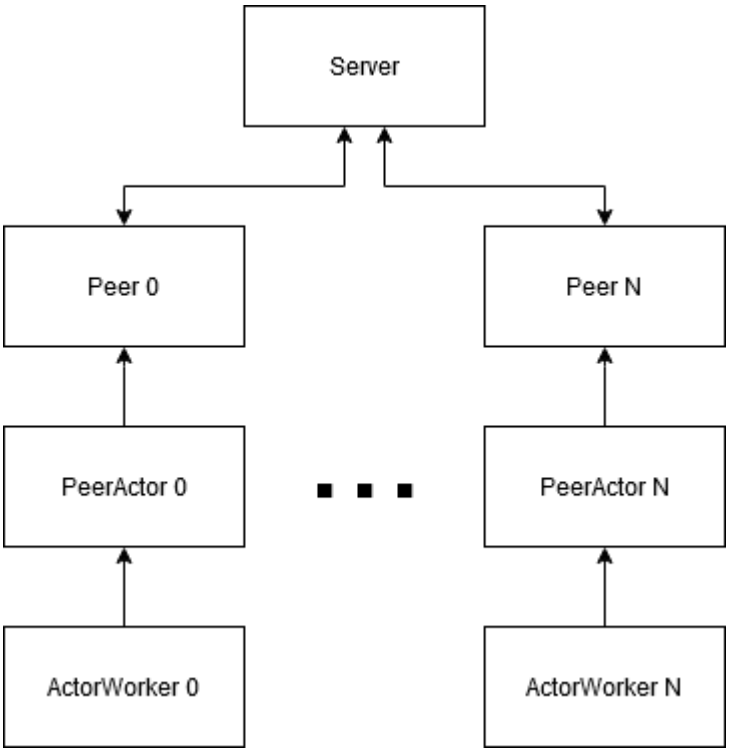
Home Exam 1

Analysis



Hashing

For the hashing algorithm, I chose Bernstein's djb2a algorithm. It is a tweaked version of his initial djb2 algorithm using XOR instead of +.

Server

The server keeps tabs on which peer has which file through the use of `PeerRecord`s. Each record corresponds to a specific peer, and the files that peer has reported to have. The server itself is a monitor object, and controls access to the list of `PeerRecord`s in an interference-free way.

Peer

The peers have a list of files that they are able to supply themselves, as well as a list of files they have downloaded from other peers. The list of downloaded files will always be a subset of the list of available files, since downloaded files are made available for download.

Main.m

Utilizing the Server and Peer types and builders, we define a `PeerActor` type which will carry out some actions on behalf of a peer. `ActorWorker`s are used to run `PeerActor`s in a separate process/thread to enable the peers to run concurrently.

We have 3 main types of `PeerActor` implementations:

- `HasFiles`: starts with no files whatsoever, and gradually over time gets more files that are then made available to the other peers.
- `WantsFiles`: starts with no files, and asks for files from the other peers. Requests files from the server by name, and then picks a random peer from the received list to download the file from.
- `LosesFiles`: starts with all files, but loses them over time, making them unavailable from the peer in question.

Tests

I wrote a simple test suite as well as some tests to ensure that I had implemented the hashing algorithm correctly. I had implemented it incorrectly to begin with, so these tests proved invaluable to ensure I could confirm that I had implemented the algorithm correctly in the end.

Caveats

When a peer downloads a file, the object reference of the file is sent instead of an actual file. This means peers will over time accrue files that are not located on the same node as the peer. I attempted to remedy this by cloning the files before they were sent, but I encountered a lot of issues where files ended up being `nil` as a result of this, and I simply had to give up at some point due to the grief the issue brought me. I am fairly certain I simply made an error somewhere, but while removing the pieces of code that enabled this cloning behavior I did not find out where I had made an error.

Since I use `PeerActor`s to control the peers, the `Peer` type has a lot of extra operations that are only meant to be used by a `PeerActor`. This is somewhat noisy, but only a minor issue. Had I thought this more through, I would've instead made the `PeerActor`s the actual `Peer` type, and had common functionality used by the peers in a sort of container-object.

PlanetLab setup

I used the following PlanetLab nodes to test my Nopester in a distributed environment:

- planet1.elte.hu
- planetlab3.cs.ubc.ca
- ple41.planet-lab.eu
- csl12.openspace.nl
- mars.planetlab.haw-hamburg.de