Cezar Mocan
CPSC 426

# Final project proposal

I am intending to implement a decentralized peer to peer file storing and sharing system, based on the Chord distributed hash table, with a keyword search functionality added. The three major functionalities of the client I am going to build are:

1. File search, either by keyword or by hash
2. File download, by connecting to the peer who owns the file and transfer it via TCP
3. Adding new files to the system

The way I see it now, the application is going to have two layers: the lower one is going to be the Chord layer, which is going to reflect the design specified in [1] as accurately as possible - create a namespace where both nodes and keys will reside, assign keys to nodes, perform lookups in O(log N) time by keeping finger and neighbour tables up to date for each node, handle new nodes or failing nodes, etc. This will provide an API to the upper layer, the application layer, with a simple functionality: getNode(K), which will return the address and port of the client which is in charge for the key K.

The application layer will be built on top of the Chord layer and has to take care of implementing the three functionalities of the project.

The major design choices for everything except the keyword search are well documented in the Chord paper. In order to implement it I am planning to make the application have two key-value stores instead of one which will both live in the same keyspace (a single Chord system) as follows:

1. file ID -> file contents
2. keyword -> list of file IDs whose names match that keyword

So a hash (SHA1) of each keyword present in the system will represent a key in the Chord structure, and the nodes will be able to query that in order to find out about the files whose titles match those keywords. Whenever a new file is added to the system, the title will be parsed and each word present in the title will represent a keyword for that file. The file ID will be updated in the lists for each of the keywords it represents.

The basic flow for each of the three operations supported by the system will go like this:

1. Search and Download go together:

- application layer takes the keywords entered by the user, creates a SHA1 for each and asks the Chord layer what nodes are responsible for the computer hashes
- the node initiates a connection with the other nodes, retrieves the lists of results and

displays them

- the user selects what he wants to download (a file name and a file ID)
- application layer asks chord again who is responsible for the selected file ID
- create a connection with the responsible node and download the file

2. Adding a new file to the system
   - compute the file ID for the selected file
   - parse the file name, get the keywords and ask Chord who is responsible for each of the keywords' hashes
   - connect to the nodes returned by Chord and update the lists for the given keywords, by adding the current file ID
   - ask Chord what node is responsible for the file ID
   - connect to that node and transfer the file to it

**Problems I see with my current proposal:**

1. I know this design is not ideal, since some keywords are a lot more popular than others, so a lot of data will be clustered on the few nodes whose responsibility is to manage the popular keywords, but since the data carried by a keyword is only a list of file IDs and not whole files it should not affect the load balancing of the system too much. If I get a better idea I will update the specifications of the system.

2. I have not designed yet the exact protocol of communication in the application layer, but I assume it will be somehow similar to the Peerster one. I will probably do its design after I'm done implementing the Chord layer, since the two are almost independent.

3. Keeping files in chunks on different nodes might be a better idea, but I chose this for simplicity. (I would need to extend the functionality of the key-value store even more in order to do this, and I'm not sure if it would improve efficiency).

**Wishlist:**

My initial idea was to create a P2P decentralized music (mp3) streaming system, similar to Spotify, on the architecture I described above, but I encountered 2 issues:

- I am not sure if streaming in general is viable in a completely decentralized system
- I have not found a way of streaming music using Qt. I would be thankful if you could point me to anything that could help. (either C++ and Qt or Java)

Initially I will implement the system I described above, and if I have time left I'll try to get streaming to work on top of that.

References:

[1] - http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf