This document will describe the workflow of the server.

Purpose

The server is meant to manage the file sharing activities and be used a “negotiator” between clients.

The main bulk of the data transferred to support file sharing will be done peer to peer to avoid heavy traffic loads on the server. Nevertheless, the server must allow a safe (robust), reliable and secure way for clients to know who they should communicate with, and when.

Main loop

The server’s main loop will persist until it is explicitly requested to stop. By default, there is no reason for us to support this feature, so we can start by implementing its main loop inside a literal while(true) clause.

During its main loop the server will accept new clients, and will **only** accept new clients.

For each new client, a new thread will be started, dedicated for communication with this client.

Server

The server component will be responsible for the “global” state of the server application.

The communication channels will be as a member of the server object, which will allow us to easily check for the current amount of working channels, close channels if necessary, and pass channels the needed interfaces (through which they’ll be able to perform thread-safe actions).

Communication channels

Inside each communication channel the client and the server will be communicating back and forth – allowing the client to query for potential seeders, request a new file share, and more.

All non-thread-local actions must be done in a thread safe manner. This can be implemented a few ways:

1. Through interfaces passed to each communication channel, each implementing a specific action (for example – DBManager which governs data insertion/query).
2. Through a global ActionQueue which will receive actions from the communication channel. Responses will be implemented by using another ActionQueue which will reside inside the communication channel and will be used by the global ActionQueue to insert “response actions” after performing an action (for example – after data query was done, insert a “query answer action” to the appropriate communication channel ActionQueue).

Actions

Non-thread-safe actions (for example – DB queries) must be done via an external component, and not directly via the channels themselves.

We should implement an interface for the channels to communicate with this external component and perform actions through it.

Representation of program in a diagram:

