Daniel Wojtowicz
A20349975
CS 550: Advanced Operating Systems
March 26th, 2018

Danutella Manual

Table of Contents

I. Introduction: What is Danutella

Danutella is a peer to peer file sharing service that works like Gnutella but with consistency. The focus of the program is for users to share their files with each other almost directly with simple operations. The main feature of this is that the files are consistent with one another most of the time. That way, people know they're looking at the same thing.

For more information about how Danutella works on the inside, check the Design document as well as the provided source code in the src directory.

II. System Requirements

To get started and use Danutella, you must ensure that you have the following components:

- An Operating System that supports Java
- Java JDK version 8 or higher
- Apache Ant

III. Building From Source

This software package comes with the source code, 10 sample server directories, and an Apache Ant build.xml file. In order to build the program, please follow these steps:

1. Make sure that the danutella directory is extracted to your desired location.
2. Open a Terminal and navigate to the extracted danutella directory. This can easily be done in certain version of Linux such as Ubuntu by right clicking in the Files manager and selecting “Open in Terminal”.
3. Final step: inside the terminal, type **`ant`** and press ENTER.

IV. Running the Peers

Push Mode

Once the build is completed, your installation directory becomes equipped with 10 peers labeled peer0 through peer9.

1. Open each peer folder in a separate terminal. Here is an example of what it could look like:

Figure 1: The terminals are opened with a paint file underneath that accentuates the connections between the peers. This is graphic underneath is optional!

2. In each of the terminals, start the Peer by typing java -jar Peer.jar push

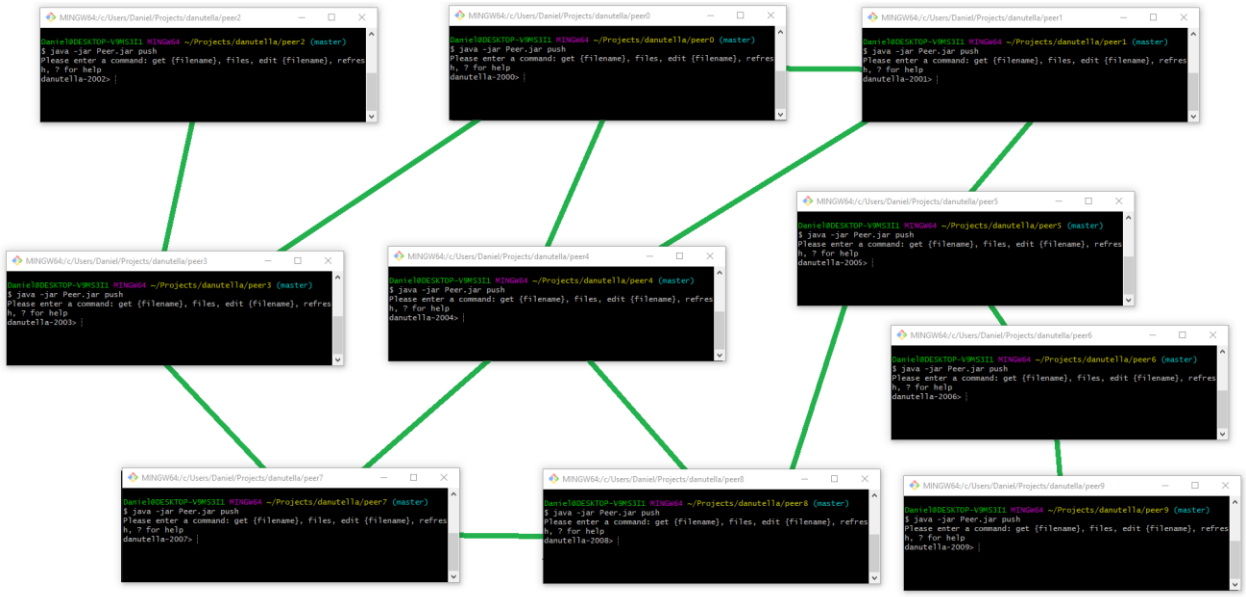

Figure 2: All peers initialized in push mode

3. Once a peer is running, it can do the following commands via user input
    - **`get {filename}`** – Sends a query message out that searches for the file with the given name. If a neighbor or someone on a network has it, this peer will download that file
    - **`files`** – lists all files this peer has downloaded. Shows file statistics such as next expiration time, owner (+++ for this is the owner), and current version number.
    - **`edit {filename}`** – simulates a file edit. This sends an invalidation message over the network in push mode but not in pull mode
    - **`refresh`** – redownloads all invalid files
    - **`?`** – shows the list of available options
    - **`exit`** – shuts down the peer
4. The following order of operations will result in output shown in the output document of this software package:
    a. First, in p0's terminal, enter **get p9-1k.txt**. This will result in a download of the file from peer9. Typing **files** will show that you now have the file.

b. Next, Go to peer 9 and enter **edit p9-1k.txt**. This will simulate an edit and invalidate the file across the network.
c. Back in peer0, enter **files** again. You will now see that the file is indeed invalid.
d. In peer0's terminal, enter **refresh**. This will re-download the file with the new version from p9-5k.txt.

Pull Mode

The previous section explained how to run the Peers in Push mode. To run them in Pull mode instead, simply replace the word push with pull: **java -jar Peer.jar pull**

The same commands are available in Pull mode as are in Push mode. The only difference is that editing a file will not send an invalidation message. Instead, periodically (default 15 seconds) a peer will check directly with the origin server for the latest version of the file.

The following order of operations will result in output shown in the second part of the output document of this software package:

1. First, in p0's terminal, enter **get p9-1k.txt**. This will result in a download of the file from peer9. Typing **files** will show that you now have the file and that it is valid.
2. Waiting the TTR time will result in a TTR-Expired. Typing **files** after some seconds, you will see that the file is now labeled TTR-Expired
3. Waiting some time until the Lazy Poll occurs, you will see that typing **files** again will have the file be Valid again with a new TTR.
4. Next, Go to peer 9 and enter **edit p9-1k.txt**. This will simulate an edit.
5. Waiting a short duration of time and entering **files** will show that the file is INVALID.
6. In peer0's terminal, enter **refresh**. This will re-download the file with the new version from p9-5k.txt, once again making the file valid when entering **files**.