CSC3002F Networks Assignment 1

Due Date:

17th March 2025, Monday 9AM

Group Members:

Kelvin Wei (WXXKEL001)
Liam de Saldanha (DSLLIA002)
Mark du Preez (DPRMAR021)

Table of Contents

Protocol Message Specification

Message Types

Commands:

- **`add_seeder:`**
    - Sent from **seeder** to **tracker** to register with the tracker.
    - Format:
        - Header:
            - `add_seeder`
        - Body:
            - `\n (tcp_address)`
- **`ping_tracker:`**
    - Sent from **seeder** to **tracker** to update ping time of seeder.
    - Format:
        - Header:
            - `ping_tracker`
        - Body:
            - `\n (tcp_address)`
- **`request_seeder_list:`**
    - Sent from **leecher** to **tracker** to request the seeder list of the tracker.
    - In response, the tracker sends a list object to the seeder of the following format:
        - `[[seeder_addr: tuple, last_check_time: datetime],...]`
    - Format:
        - Header:
            - `request_seeder_list`
- **`request_file_list:`**
    - Sent from **leecher** to **tracker** to request a current list of files available.
    - Format:
        - Header:
            - `request_file_list`
- **`request_connection:`**
    - Sent by **leecher** to **seeder** to request a connection to the seeder.
    - Format:
        - Header:
            - `request_connection`

- **`request_file_chunk:`**
    - Sent from **leecher** to **seeder** to ask the seeder to send a specific range of chunks of a file from a certain starting point in the file (send_after).
    - Format:
        - Header:
            - `request_file_chunk`
        - Body:
            - `\n [file_name, num_chunks, send_after]`

Data Transfer:

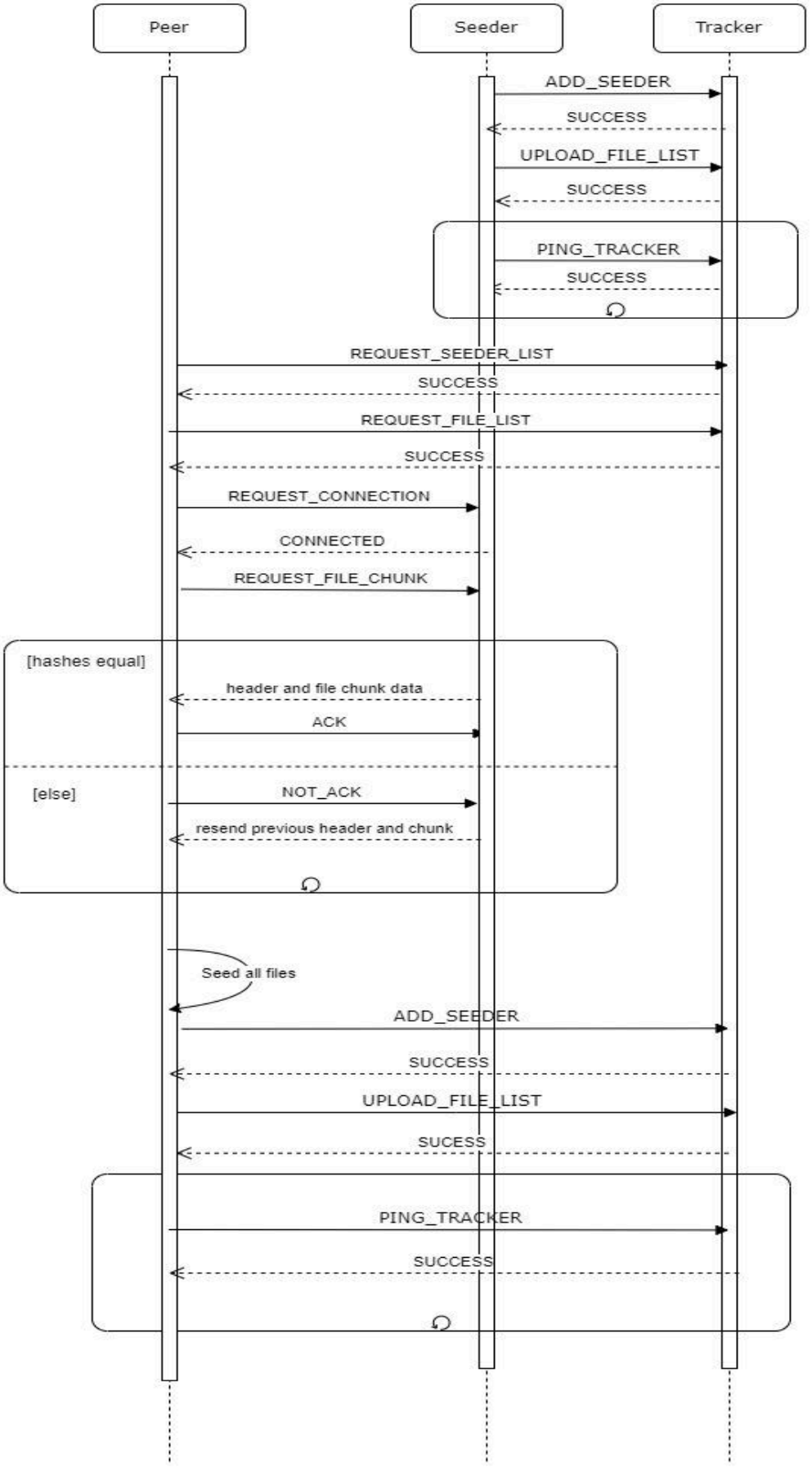
- **`upload_file_list:`**
    - Sent from **seeder** to **tracker** to add the seeder's file list to tracker's current file list.
    - Format:
        - Header:
            - `upload_file_list`
        - Body:
            - `\n [filename1, filename2, ..., filename]`

Control:

- **`connected:`**
    - Sent from **seeder** to **leecher** when a successful connection has been established.
    - Format:
        - Header:
            - `connected`
- **`away:`**
    - Sent from **seeder** to **leecher** to indicate that it cannot connect at the moment.
    - Format:
        - Header:
            - `away`
- **`exit_connection:`**
    - Sent by **leecher** to **seeder** to ask to leave the connection.
    - Format:
        - Header:
            - `exit_connection`

- **error:**
    - Sent by **tracker** to indicate that there was an error.
    - Format:
        - Header:
            - error
        - Body:
            - <error message>
- **success:**
    - Sent by **tracker** to indicate a successful request.
    - Format:
        - Header:
            - success
- **acknowledgement:**
    - Acknowledgement by **leecher** to **seeder** that file is received successfully.
    - Format:
        - Header:
            - acknowledgement
- **error_notacknol:**
    - Sent by **leecher** to **seeder** that file was not successfully received.
    - Format:
        - Header:
            - error_notacknol

Sequence Diagram



Visual Overview of Features

Tracker Startup

```
Using Default arguments:
TRACKER: (ip: 127.0.0.1, port: 12500)
Tracker is up
```

Seeder Startup

```
Using default parameters:
 seeder file path: './data/'
SEEDER: (ip: 127.0.0.1, port: 12501)
TRACKER: (ip: 127.0.0.1, port: 12500)
File Dict: {'DOWNLOAD ME!!!!.mp4': 20212443, 'NetworkApp2025.pdf': 87478, 'Senior Lab - Computer Science Building - UCT video venue finder.mp4': 13070983, 'VIDEOS.zip': 3
2834707}
Add to Tracker Result: success
Upload to Tracker Result: success
Ping Result: success
```

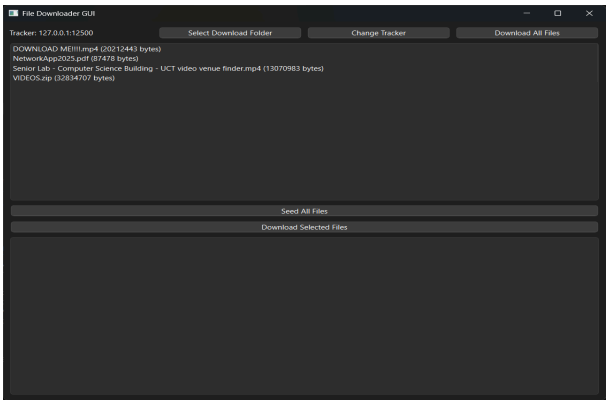
Tracker Receives Seeder

```
Connection Received from: ('127.0.0.1', 58674)
Request Message: add_seeder
Payload information: ['add_seeder', '["127.0.0.1", 12501]']
Added:  ['127.0.0.1', 12501] .... Request from: ('127.0.0.1', 58674)
Connection Received from: ('127.0.0.1', 58675)
Request Message: upload_file_list
Payload information: ['upload_file_list', '{"DOWNLOAD ME!!!!.mp4": 20212443, "NetworkApp2025.pdf": 87478, "Senior Lab - Computer Science Building - UCT video venue finder
.mp4": 13070983, "VIDEOS.zip": 32834707}']
Successfully updated file list
Connection Received from: ('127.0.0.1', 58676)
Request Message: ping_tracker
Payload information: ['ping_tracker', '["127.0.0.1", 12501]']
Ping Received from: ('127.0.0.1', 58676) ... to update ping time on ['127.0.0.1', 12501]
```

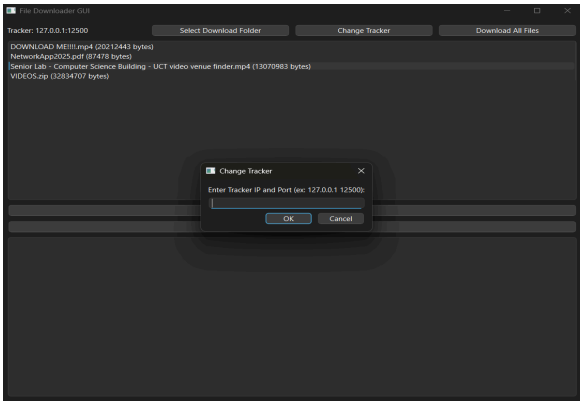
Seeder Pinging Tracker

```
Connection Received from: ('127.0.0.1', 59904)
Request Message: ping_tracker
Payload information: ['ping_tracker', '["127.0.0.1", 12501]']
Ping Received from: ('127.0.0.1', 59904) ... to update ping time on ['127.0.0.1', 12501]
```

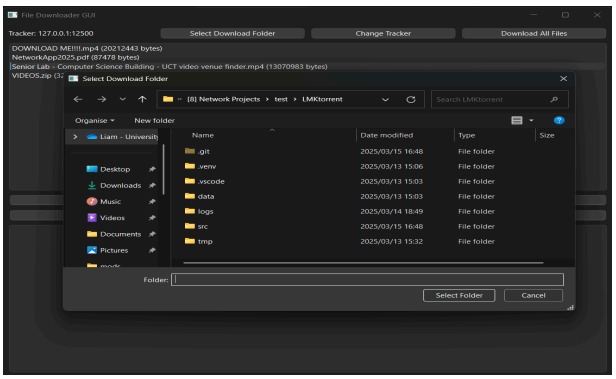
Leecher/Peer Startup



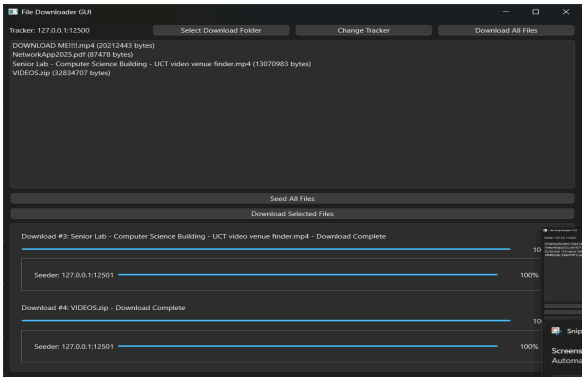
Change Tracker Functionality



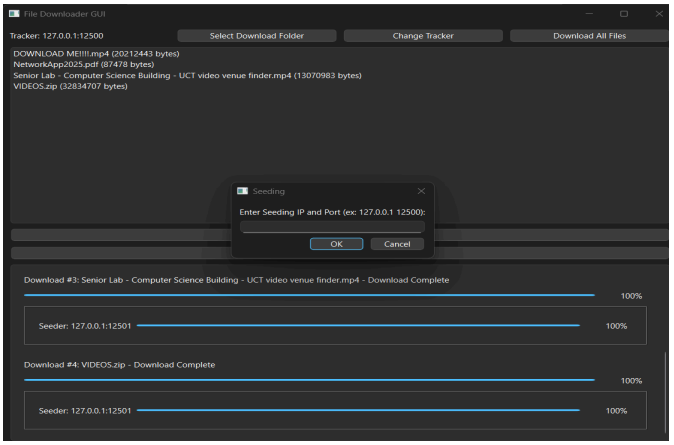
Select Download Folder



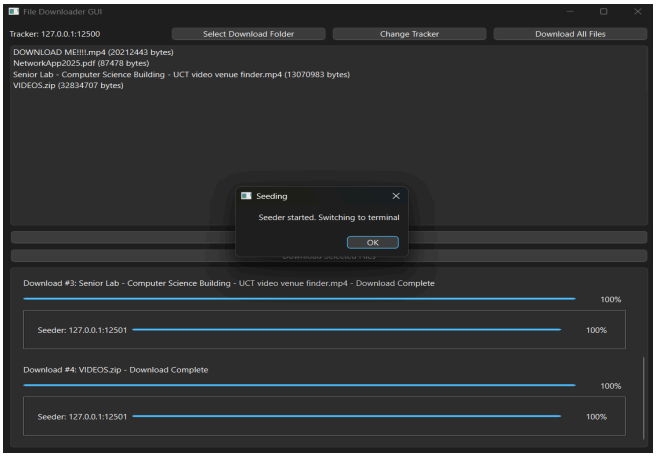
Download Files



First Transition of Leecher To Seeder



Second Transition of Leecher to Seeder



Third Transition of Leecher to Seeder

```
seeding
File Dict: {'DOWNLOAD ME!!!!.mp4': 20212443, 'NetworkApp2025.pdf': 87478, 'Senior Lab - Computer Science Building - UCT video venue finder.mp4': 13070983, 'VIDEOS.zip': 3
2834707}
Add to Tracker Result: success
Upload to Tracker Result: success
Ping Result: success
```

Features

File Integrity Verification

Ensures files are not corrupted through the download process. This is implemented through the use of checksum and hash validation.

GUI

Provides an interface for managing downloads ,seeding and tracking progress. This allows for a better user experience compared to CLI. The ability to track progress of the downloads provides users with reassurance.

Limiting seeders

Restricting the number of seeders a peer connects to. This allows us to optimise bandwidth and prevent network congestion and manage the seeder overhead.

Parallel Downloads (Required)

Allows multiple seeders to send a single peer different parts of the same file. Increases the efficiency of downloading and most importantly increases the speed of downloading. This allows for optimal use of bandwidth and data downloads.

Re-seeding (Required)

Peers are able to join in on the distribution of the files they download (reseeding). This allows for a sustainable environment where there are enough seeders and provides the backbone for implementing “Tit-for-tat” mechanisms.

Selective/All Downloading

Giving users the option to download specific files they may want or all of the files in the folder. This caters for those who only want certain files and those who want to download everything and then optionally reseed.

Implementation

The system requires a tracker, leecher and seeder to function. We expanded by adding a peer which has the ability to become a seeder. The tracker and seeder operate on the backend and provide data and connections to peers. The peer has a GUI for accessing said data and to provide a better user experience.

The tracker is assigned an ip address and port number. It initialises lists to maintain active seeders and current files available for download. The tracker establishes a UDP connection to listen for messages being sent by seeders or peers in a loop. When it receives a message it interprets it and responds appropriately. Some of the actions taken by the tracker include :

- Adding new seeders
- Updating available files
- Removing expired seeders
- Sending the file list
- Sending the seeder list

A seeder is given an ip address and port as well as the tracker's address and port number and directory which it will seed. When the seeder is initialised, it sends messages to update the tracker's list of available files to download, to add itself to the tracker's active seeder list and start a thread to ping the tracker that it is active. A TCP connection is established on which the seeder waits for a peer to connect. Once a peer connects the seeder receives messages containing the specifications of the files the peer wants to download and how the seeder should send them. The seeder constructs the message with the header and hash. A thread is created to send the file chunks. Once completed it waits for the next peer.

Peers are given the tracker address and the directory where they store their downloads. They begin as a Leecher and eventually have the option to transition to be a seeder. When they are initialised they get the active seeder list and files available for download. Their max seeder connection is also set. They then use the GUI to perform actions:

- Download selective files
- Download all files
- Change directory
- Seed all files

When a peer downloads a file it tries to connect to all active seeders by setting up a TCP connection with each (while staying within the max seeder connection range) and once it establishes how many seeders are willing to seed it. it provides a message to each seeder dictating which chunks and how much they should all seed to the peer. The peer runs a thread pool to send these requests simultaneously to each seeder allowing for parallel downloads. Once the chunk is received it checks if the hash is equal to ensure a reliable download otherwise a not-ACK is sent and the seeder resends the chunk . Once the peer has downloaded all files available the peer can transition into a seeder in which the user will be prompted for an IP address and port number on which it will seed. The GUI will close and the seeder will be redirected to the terminal.