Computer Science 2B

Practical Assignment 05

2023-08-22

Deadline: 2023-09-12 12h00 Marks: 100

This practical assignment must be uploaded to eve.uj.ac.za **before** 2023-09-12 12h00. Late or incorrect submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student.

Make use of proper coding conventions and documentation. Marks will be deducted if these are not present. Your submission should include a batch file.

The reminder page includes details for submission and queries. Please ensure that **ALL** submissions follow the guidelines. The reminder page can be found on the last page of this practical - read the reminder page carefully.

This practical will focus on peer-to-peer file sharing with UDP.

One of the most popular applications that use peer-to-peer communication is peer-to-peer file sharing (where examples are BitTorrent and Direct Connect). Instead of downloading a file from one host or server, it is downloaded from a peer that has the file. The file is seeded (or sent) from a peer to leechers that want the file (or a peer that does not have the complete file).

In this practical, we will be creating a **UDP-based** peer-to-peer file sharing system which consists of a Client that can either send files (Seeder mode) or receive files (Leecher mode). Once a client starts, it asks the user for which mode the client will be making use of. The user can then continue sending or receiving binary files depending on what mode it is in, until the user terminates the connection by closing the client. In order to achieve this, you require a client that has two modes:

Seeder Mode

In Seeder mode the following functionality needs to be provided:

1. The Seeder must provide the user with the ability to add a file to the list of available files that can be shared. **Suggestion** - You can use an add button that chooses files with a *FileChooser* and a *List* to display added files.

2. It must be able to send a **list of files** available (it can be a numbered list) to another peer that has connected to it.

3. It must be able to send the **chosen local file** to the peer that asked for it.

Leecher Mode

In Leecher mode the following functionality needs to be provided:

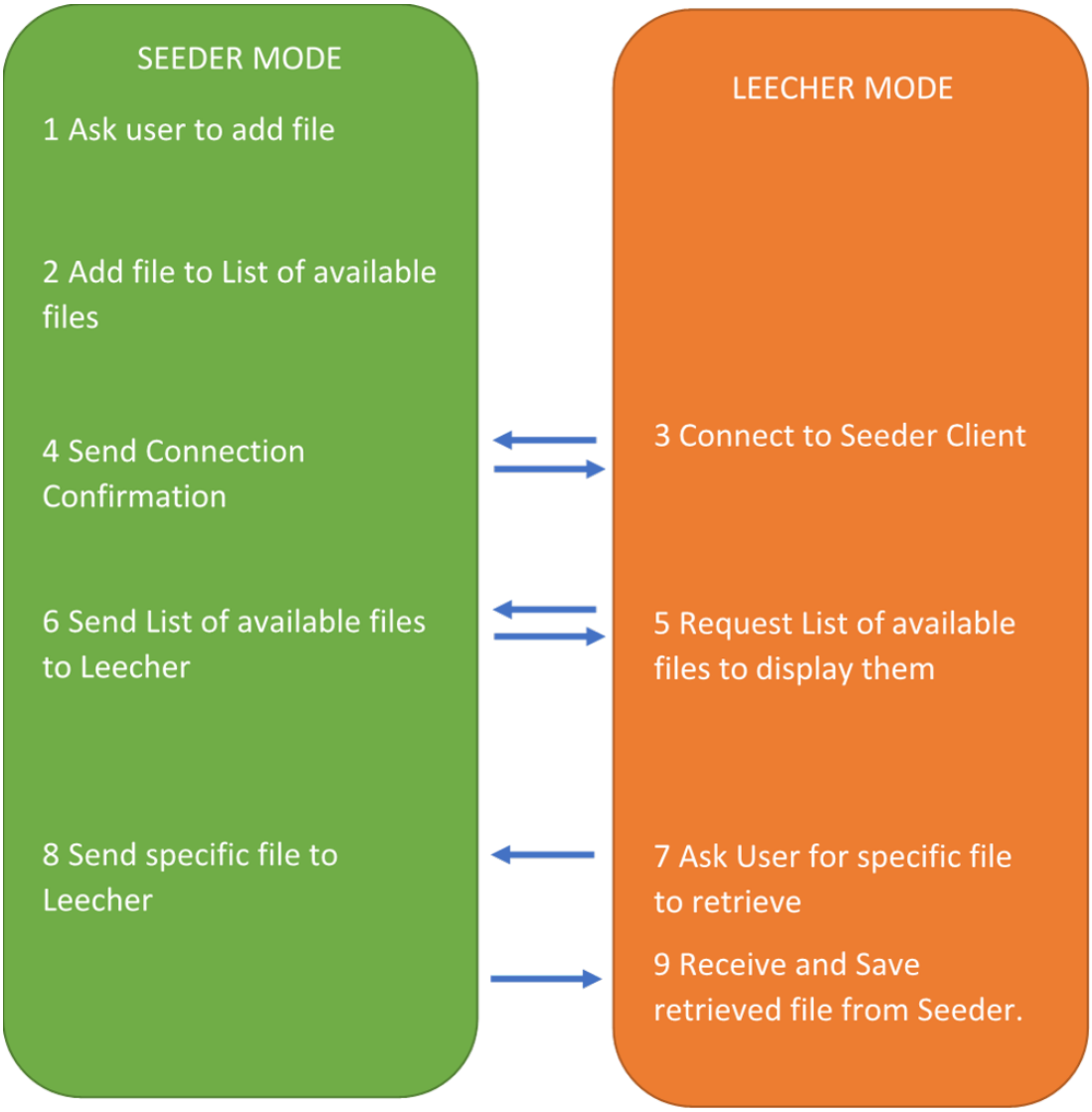
1. The Leecher must connect to a Seeder client (using a specific host address and port number). **Suggestion** - You can use 2 *TextField*s to capture the host address and port number from the user, along with a Connect button.

2. Ask the Seeder peer what files it has available and provide the user with the ability to choose a specific file. **Suggestion** - A *List* can be used to display the available files from a Seeder client.

3. Tell the Seeder peer, which file it wants (you can send the number, the user chose). **Suggestion** - A Retrieve button can be used to send the selected index to the Seeder client.

4. Receive and save the file from the Seeder peer.

Example


SEEDER MODE
1 Ask user to add file
2 Add file to List of available files
4 Send Connection Confirmation
6 Send List of available files to Leecher
8 Send specific file to Leecher
LEECHER MODE
3 Connect to Seeder Client
5 Request List of available files to display them
7 Ask User for specific file to retrieve
9 Receive and Save retrieved file from Seeder.

Marksheet

1. Client GUI for both Seeder and Leecher modes.	[10]
2. Seeder Mode: Send list of files available.	[10]
3. Seeder Mode: Send the chosen local file.	[20]
4. Leecher Mode: Ask peer what files it has available.	[10]
5. Leecher Mode: Receive a file from a remote peer.	[20]
6. Coding convention and correctness of solution.	[30]

NB

Submissions which **do not compile** will be capped at 40%

The awarding of marks is dependent on the student's ability to effectively justify and demonstrate understanding of the practical work presented.

Execution marks are awarded for a correctly functioning application and not for having some related code.

Reminder

Your submission must follow the naming convention as set out in the general learning guide:

`SURNAME_INITIALS_STUDENTNUMBER_SUBJECTCODE_YEAR_PRACTICALNUMBER`

Your submission must include the following folders:

- `bin` - *(Required)* - Should be empty at submission but will contain runnable binaries when your submission is compiled.
- `docs`- *(Required)* - Contains the batch file to compile your solution, and any additional documentation files. All documentation files must be in **PDF** format. Your details must be included at the top of any **PDF** files submitted. **Do not include generated JavaDoc.**
- `src`- *(Required)* - Contains all relevant source code. Source code must be placed in relevant sub-packages! Your details must be included at the top of the source code.
- `data` - *(Optional)* Contains all data files needed to run your solution.
- `lib` - *(Optional)* Contains all libraries needed to compile and run your solution.

Every submission **must** include a batch file that contains commands which will:

- Compile your Java application source code.
- Compile associated application JavaDoc.
- Run the application.

Do not include generated JavaDoc in your submission. All of the classes/methods which were created/updated need to have JavaDoc comments.

Note that only **one** main submission is marked. If you have already submitted once and want to upload a newer version, then submit a newer file with the same name as the uploaded file in order to overwrite it.

Bonus submissions should be uploaded separately and clearly named as the bonus submission - which will then be marked accordingly.

It is important to make use of **each practical opportunity** as preparation for the practical semester test (ST2). The practical assignments also contribute to the **Practical Component Mark** (PCM). There will be 9 practical assignments (P00-P08) this semester which will be released on a weekly basis except when tests are being written.

The process to **query** your practical assignment with an assistant is discussed in the learning guide as well as the first practical lecture.