LAB 5: P2P FILE SHARING APP

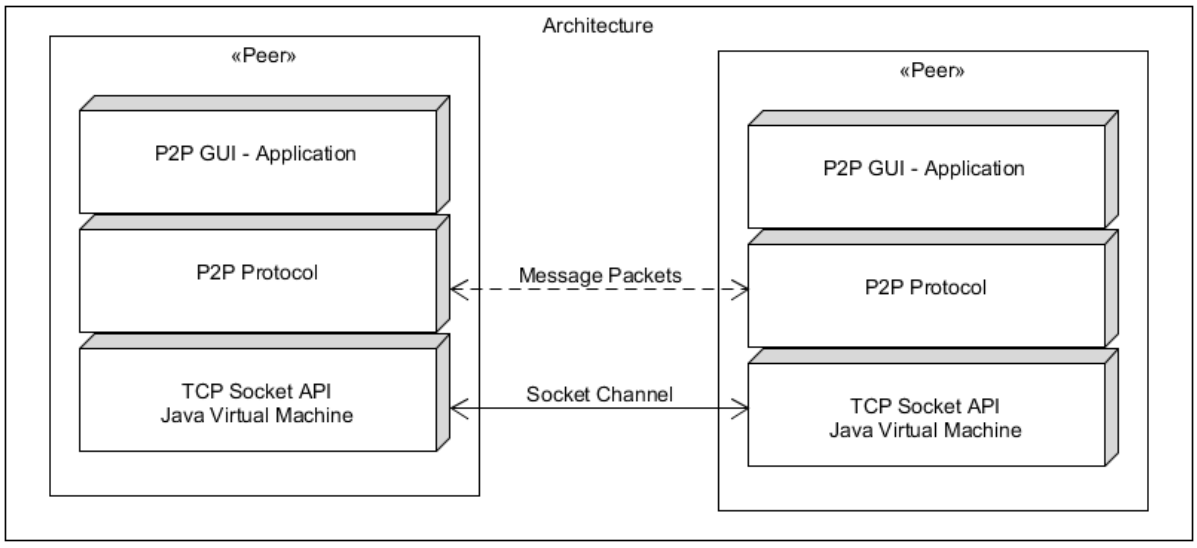
In this lab, you will learn how to develop a peer-to-peer application as well as protocol to communicate effectively among the peer nodes. Download the source code from Moodle and import it into your Eclipse IDE. There are altogether five packages in the project:

- **edu.rosehulman.p2p.ptotocol** – Contains the protocol/interface specification for the peers
- **edu.rosehulman.p2p.impl** – Implements the protocol interfaces specified in the protocol package.
- **edu.rosehulman.p2p.impl.handlers** – Implements classes that perform extra work in the process of receiving or sending a network packet.
- **edu.rosehulman.p2p.impl.notification** – Defines interfaces for receiving protocol-related notification (for example in GUI).
- **edu.rosehulman.p2p.app** – Implements the entry point of the application and GUI classes.

Follow your instructor's demo to understand how it works.

Architecture

A peer in P2P application acts as both client and server. The following diagram depicts the overall architecture of the application. It allows several peers to communicate with each other through TCP sockets. A P2P protocol is needed for information exchange and message processing, which is discussed next.



Request for Comments (RFC) – The P2P1.0 Protocol

The P2P1.0 protocol is used for peer-to-peer file sharing. Here is the general packet format that gets exchanged between two peers (excluding the surrounding lines):

<Protocol-Version> <Space> <Command> <Space> <Object> <CRLF>
<Key1> : <Value1> <CRLF>
<Key2> : <Value2> <CRLF>
…
<KeyN> : <ValueN> <CRLF>
<CRLF>
<Payload-Data>

Here is the explanation of the fields:

Protocol-Version: P2P1.0, for version 1.0 of the protocol
Space: Regular space character
Command: The request for action, e.g. GET, PUT, ATTACH, etc. See **edu…protocol.IProtocol** interface.
Payload-Data: The extra data that the packet contains in addition to the regular header, e.g. a file.
CRLF: Carriage return and line feed together, i.e., the "\r\n" string

In the rest of the document we will discuss the sequence as well as the format of message exchange.

Establishing and Closing Connection

There are four commands that are used to establish and close connection in the protocol: ATTACH, ATTACH_OK, ATTACH_NOK, and DETACH. Here is the general format of these messages:

P2P1.0 **ATTACH** <Receiver-IP:Port> <CRLF>
Host: <Sender-IP> <CRLF>
Port: <Sender-Port> <CRLF>
Sequence: <Sender-Generated-Seq-No> <CRLF>
<CRLF>

The **ATTACH_OK** and other packets are only different in the command field (highlighted above).

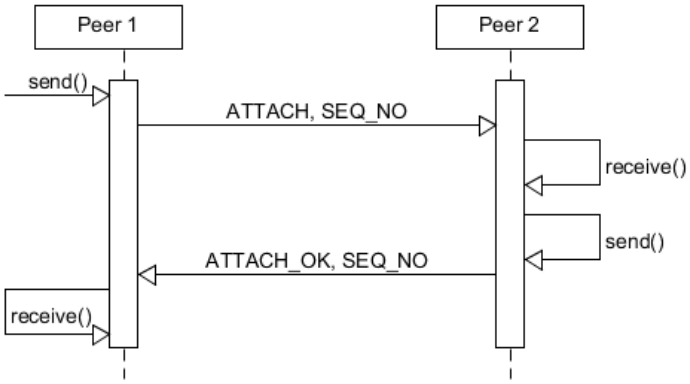
Accepting Connection

Here is the sequence to establish connection, (representing a packet as <command, sequence-no> :

Peer 1: Send <ATTACH, 1>
Peer 2: Receive <ATTACH, 1>
Peer 2: Send <ATTACH_OK, 1>
Peer 1: Receive <ATTACH_OK, 1>

Note that the same sequence number is used to represent the packet that originated the connection request.

Here is the sequence diagram depicting the message exchange between two peers:



Rejecting Connection

Here is the sequence showing how a connection may be rejected:

Peer 1: Send <ATTACH, 2>
Peer 2: Receive <ATTACH, 2> (Protocol ver. mismatch, header fields missing, or some other problems)
Peer 2: Send <ATTACH_NOK, 2>
Peer 1: Receive <ATTACH_NOK, 2>

Closing Connection

If a peer is already connected, it can be sent a packet to terminate the connection.

Peer 1: Send <DETACH, 2> (Followed by socket close operation)
Peer 2: (Socket close operation, **the peers may throw an exception on the console, which is expected.**)

Note that the socket close operation may throw an exception while reading the DETACH packet, which is ok.

Take a look at **edu…impl.P2PMediator** to see how these packets are created.

Listing Files in a Remote Peer Directory

Sending LIST Request

The LIST file request has the following format:

P2P1.0 **LIST** <Receiver-IP:Port> <CRLF>
Host: <Sender-IP> <CRLF>
Port: <Sender-Port> <CRLF>
Sequence: <Sender-Generated-Seq-No> <CRLF>
<CRLF>

The LISTING Response

The peer receiving the list file request respond with the following packet:

P2P1.0 **LISTING** <Receiver-IP:Port> <CRLF>
Host: <Sender-IP> <CRLF>
Port: <Sender-Port> <CRLF>
Sequence: <Sender-Generated-Seq-No> <CRLF>
Payload-Size: <Size-Of-Payload> <CRLF>
<CRLF>
abc.txt <CRLF>
xyz.gif <CRLF>

Note that the size of payload here is the size of the last two lines of the packet, which is the file listing in the remote directory.

Lab Assignment

Understanding the Protocol and Handlers

1. Write down a few line description of each of the following interfaces in the **protocol** package. Please specify their responsibilities in your description. Please look at their concrete implementation in the **impl** package:

IConnectionMonitor, IHandler, IHost, IP2PMediator, IPacket, IProtocol, IRequestHandler, IResponseHandler, IStreamMonitor, and AbstractHandler.

The class diagram of the application can be found at the very end of this document and also on Moodle.

Understanding the Notification Mechanism

2. Write down a few line description of each of the following interfaces in the **notification** package. Please specify their responsibilities in your description. Make sure to look at their concrete implementation in **app.P2pGUI** and **impl.P2PMediator**:

IActivityListener, IConnectionListener, IDownloadListener, IListingListener, and IRequestLogListener.

3. Explain using a C&C diagram how the notification mechanism works in the application.

Extending the Application

4. You will extend P2P1.0 protocol to support the following new feature:

Design and implement a "FIND" command that will search (depth-limited, breadth-first search) **n** peers deep in the network for the file specified in the Network File Searching panel of the application GUI. The **n** must be read from the Configuration Window of the application (modify it).

You must provide a snapshots of application working normally. Here are some of the scenarios that you must test:

Scenario A

a.1 Configure 5 peers in a ring (peer1 -> peer2 -> peer3 -> peer4 ->peer5 -> peer1)

a.2 Set the network search depth to 2.

a.3 Have the file hosted only by peer3 and peer4 (name the file "FindMe.txt").

a.4 Start the search at peer1, your snapshot must show that the files were found in both peer3 and peer4

Scenario B

b.1 Configure 6 peers in a ring (peer1 -> peer2 -> peer3 -> peer4 -> peer5 -> peer6 -> peer1)

b.2 Set the network search depth to 2.

b.3 Have the file hosted only by peer3, peer4 and peer5 (again, name the file "FindMe.txt").

b.4 Start the search at peer1, your snapshot must show that the file was not found in peer 4 but was found in peer3 and peer5.

Deliverables (Two Files)

- Create a **pdf** file that contains answers to Q1-3.
- Extend the protocol documentation to include the documentation of the FIND feature and append to the pdf.
- Append the snapshots of the two test scenarios to the pdf.
- **Turn in the pdf on Moodle.**
- **Bundle the project into a zip file and turn it in on Moodle separately.**

Bonus (2 Course Points)

For **2 extra course points**, show the network graph (undirected) of the connected hosts on the Network Graph panel of the GUI. Each node in the graph must show the last part (typically 3 digits) of the IP address followed by the port number as the node label (e.g, 112:9001). The graph must update itself as the peer attach or detach from the network. In your **pdf**, add a separate Bonus section and present the following:

1. Explain the protocol showing all the packets used.
2. Show the sequence (a well-thought-out sequence) in which the graph related packets get send and received by the hosts.
3. Take snapshots that show a binary tree like network of depth 3
4. Take a snapshot that show a ring network of 5 hosts.
5. Show that the graph is dynamically updated (using snapshots) by deleting a node in the tree. It should always be the case that **the detaching node must inform necessary neighbors** to connect with each other so that the network graph is strongly connected (i.e., all the other nodes including the neighbors are reachable from one another after the node is detached).
6. Repeat the process for the ring network of 5 hosts and show your snapshots.

Class Diagram