Methodology:

For the testing purpose we declare that host.java is super node. player1.java is all the node that can directly connect to super node. Player2.java is second layer node that can connects to host node.

Player1.java can ran multiple times as instances as new peer. Each time they will be randomly assigned a new port number and file contents. For the simple and clear demonstration, use local txt file as its file that can be shared.

Connect to the Gnutella network by sending out a ping. The ping will be forwarded by host and player2. Each ping will get a corresponding pong routed back on the same route as the ping was sent.

The TTL is decremented and the Hops is incremented after each step. If at anytime TTL(0) != TTL(i) + Hops(i) the node can discard the descriptor as an error. When TTL gets to zero the packet is not forwarded to anymore nodes from the Node 4 (i.e. the packet is dead).

Communication:

UDP send data by byte array, but for better representation, use string displayed in console as well as each command descriptor.

Use thread to do the ping and pong.

Identity and nodes storage:

Each port number is like an ID that represents each node (player)'s identity.

Files:

printout the txt file content instead of download is more convenient to represent file sharing capability. However, my codes include java file that contains file downloading.

Ping + Query (HIT) + TTL:

The default number of files for each node is 1. so only passing file name by pong is enough to test the system; Know source port; Pong back; add to nodesTable and get ready for Ping back; know the shared files; bring shared files and ping to my peers from nodesTable with TTL values.

Query (HIT) + UI:

Use scanner to take user input. Use thread to keep track of user input and return what user what. Keep running until user press 3 to quit this thread. User is able to print out existing files (press 1 followed by file name), see available files (press 2). After data communication, file list is updated.

Pong:

Know source port; update nodesTable by portnumber as ID and timeout time.

Module Design

For nodes:

- LOCAL FILE SETUP

- USER INPUT COMMAND INTERATION
- create a new UDP socket to receive packets, including assign random port numbers
- set data structure KEEP TRACK OF ALL PEERS
- launch CHECK TIME OUT thread
- while loop forever, waiting for UDP packet and ready to process each message with its command descriptor.
- Process includes command descriptor, TTL values, and the above mentioned PING, PONG, QUERY (HIT).

For health check function thread:

- periodically check peers node in nodes table with its timeout time and current time. Compare and update nodes table including remove them and modify data
- to avoid java.util.ConcurrentModificationException, do not remove table entry right after the timeout is met. Instead, create arraylist to store all the outdated entry during hashtable iteration. After interation before sleep period, remove all entries that in arraylist.

For user interaction:

- While true, keep listening user input and ask for user input
- Responds to user input as needed, including printFile, show existing files.

For communication thread:

- Ask for source port number, target source number, file name, and TTL value
- Setup UDP and message

Meet Grading criteria:

- A node joins the network with a PING to announce self
- Receivers can forward the Ping to their neighbors based on TTL
- Receivers can Pong back to announce self
- Each node PING neighbors periodically and PONG back
- Choose a new neighbor when an existing one disappears
- Download (and display) the file from a node which has the requested files.
- Creates a query to get peers sharing files.
- in my case, all nodes are on local IP. so no need to including IP in each PONG

Screen Shot and Demo:

Test: test for user interaction (QUERY, QUERYHIT) and file sharing.

The following images are console of Player 1, Player 2, host respectively. At the beginning, if user press 1, user can only see their own file. After communication protocol, the file are shared and they can see peers' shared file. After press 1, user can enter file name known from last step, press enter to print out the file contents.

How to run: any order

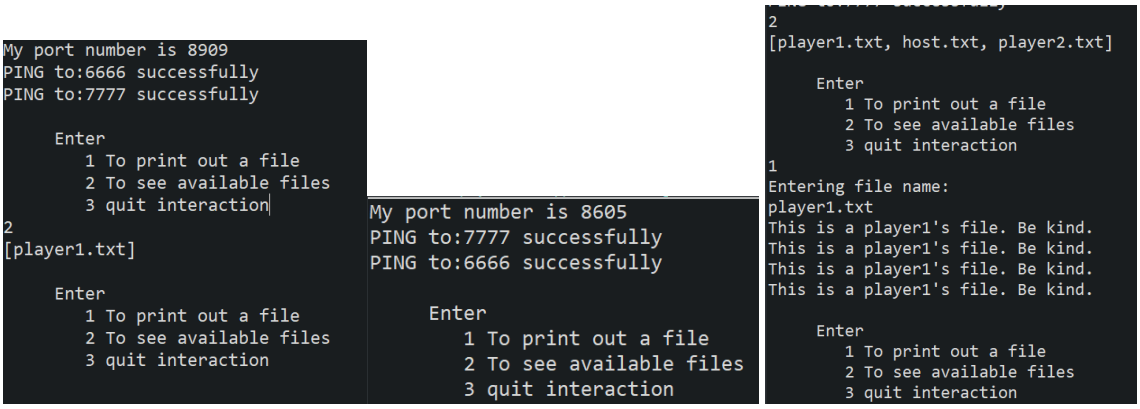


Figure: Player 1

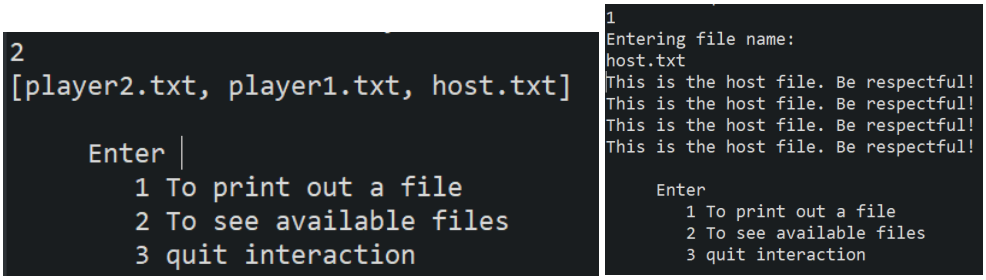


Figure: Player 2

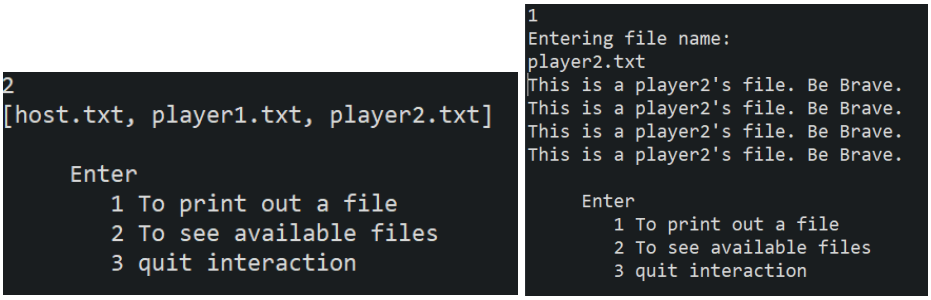


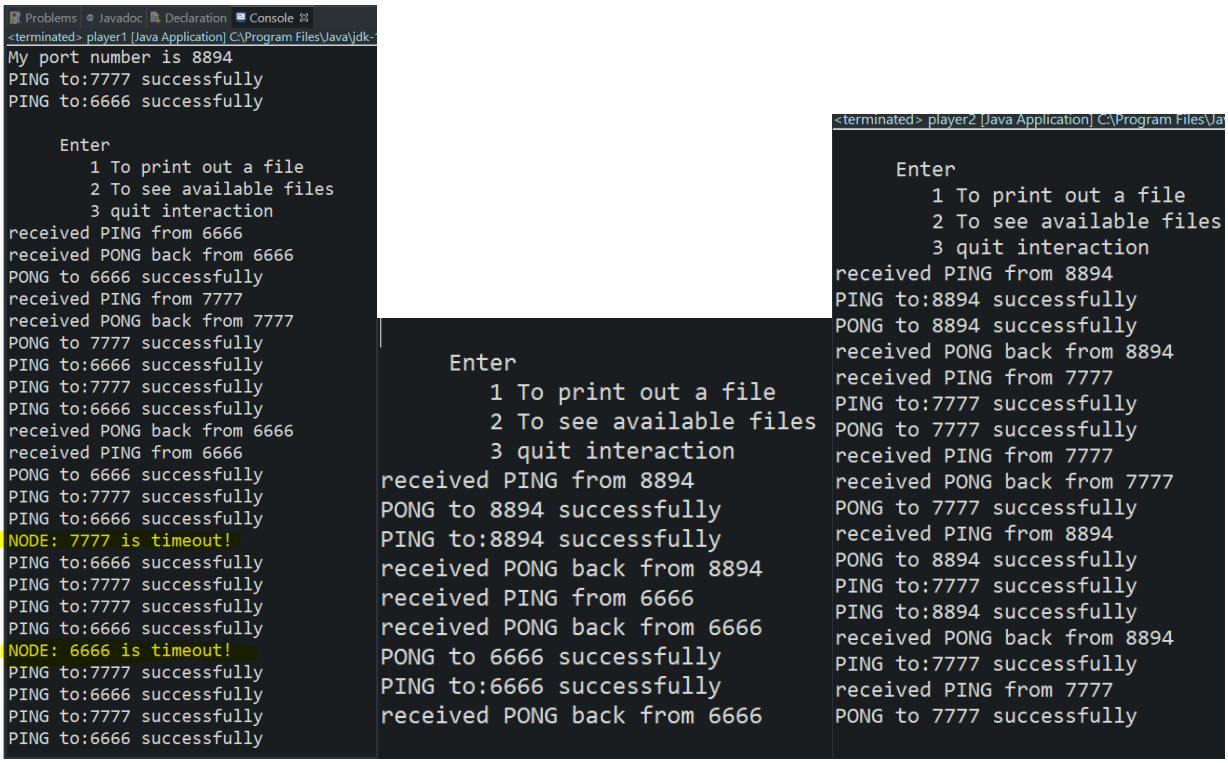
Figure: host

Test: test for PING, PONG, checking peers' timeout and add new node as peers.

The following images are console of Player 1, Player 2, host respectively.

we can see that after 20s, players 1 noticed host and player2 are timeout. It looks stuffed however for the testing purpose.

How to run: run host, player2, then player 1, and terminate host and player2.


Problems Javadoc Declaration Console
<terminated> player1 [Java Application] C:\Program Files\Java\jdk-
My port number is 8894
PING to:7777 successfully
PING to:6666 successfully
Enter
1 To print out a file
2 To see available files
3 quit interaction
received PING from 6666
received PONG back from 6666
PONG to 6666 successfully
received PING from 7777
received PONG back from 7777
PONG to 7777 successfully
PING to:6666 successfully
PING to:7777 successfully
PING to:6666 successfully
received PONG back from 6666
received PING from 6666
PONG to 6666 successfully
PING to:7777 successfully
PING to:6666 successfully
NODE: 7777 is timeout!
PING to:6666 successfully
PING to:7777 successfully
PING to:7777 successfully
PING to:6666 successfully
NODE: 6666 is timeout!
PING to:7777 successfully
PING to:6666 successfully
PING to:7777 successfully
PING to:6666 successfully
Enter
1 To print out a file
2 To see available files
3 quit interaction
received PING from 8894
PONG to 8894 successfully
PING to:8894 successfully
received PONG back from 8894
received PING from 6666
received PONG back from 6666
PONG to 6666 successfully
PING to:6666 successfully
received PONG back from 6666
<terminated> player2 [Java Application] C:\Program Files\Ja
Enter
1 To print out a file
2 To see available files
3 quit interaction
received PING from 8894
PING to:8894 successfully
PONG to 8894 successfully
received PONG back from 8894
received PING from 7777
PING to:7777 successfully
PONG to 7777 successfully
received PING from 7777
received PONG back from 7777
PONG to 7777 successfully
received PING from 8894
PONG to 8894 successfully
PING to:7777 successfully
PING to:8894 successfully
received PONG back from 8894
PING to:7777 successfully
received PING from 7777
PONG to 7777 successfully