Keith Downer
Jonathan Kosasih
Justin Townsend

P2P Application Software Specification

The client will be opened as a command-line program. When opened, the user will see a startup message telling them to wait until the program finishes acquiring peers. If the program does not find any peers, it will remain on this screen until a peer connects to it, or the program is closed. Once a connection is established, the program will proceed to the chat interface, where it displays messages sent and received, as well as a prompt for the user to type and send messages.

Program Structure:

Scanning for peers
- Inform client that we are searching for peers.
- Create the list for storing peers.
- Acquire the ip address of the computer this program is running on.
- Send “HELLO” messages to all local IP Addresses for each port in port range.
- Get the current system time.
- Wait <timeout> seconds for at least one peer to respond.
- If a peer responds:
  - Add them to the list of peers.
  - Go to the next stage of the program.
- If no peer responds:
  - Inform the user that we found no peers and are now waiting for one to connect to us.
  - Listen for peer connections until a peer connects, or program is killed.
  - When a peer connection is received:
    - Send an acknowledgement message.
    - Add them to the peer list.
    - Go to the next stage of the program.

Main Message Interface
- Start an infinite loop:
  - Get the current system time
  - Get the time delta between the last loop
  - Loop over every peer in the peer list:
    - Subtract the time delta from their timeout value
    - If a peer’s timeout value has dropped below zero, set them as disconnected
  - Check the input stream for messages from the user to send.
  - If there is a message to send:
    - Send the message to all peers in the peer list
  - If the user typed “/quit”
    - Send a “LEAVE” message to all peers in the peer list
  - If <time> has passed since the last “ALIVE” broadcast:
    - Send an “ALIVE” message to all peers in the peer list.

Store the current time as the last “ALIVE” time
Listen for messages for a set amount of time (say 500ms)
If message is received:
Perform the specific actions for said message
Get new system time
Calculate how much time has passed since started listening
Go back to listening until reached the set amount of listen time

Specified Variables:

Peer List
a list of 4-tuples made up of (ADDRESS, CONNECTED, TIMEOUT)
ADDRESS: A tuple with an IP Address string and Port number
CONNECTED: A boolean specifying whether a peer is connected
TIMEOUT: A floating point value specifying how many seconds till a peer is considered disconnected

Timeout Value Constant
Number specifying how long (in seconds) the timeout value will be reset to.

Alive Broadcast Time
The amount of seconds between “ALIVE” message broadcasts.

Actions per message type received:

“HELLO”
store the sender’s IP and port in the peers list
Display a message to the user indicating “IP has connected”
Send an “ACK” message back to the sender

“ACK”
If scanning for peers, store the sender’s IP and port in the peers list
Otherwise restore this sender's timeout value to the maximum timeout value.

“ALIVE”
Restore the sender’s timeout value to the maximum timeout value.
Send an “ACK” message back to the sender

“MSG”
Display the sender’s IP and message to the user “<IP> message”

“LEAVE”
Remove the sender from peers list
Display to the user that “IP has disconnected”