WESTERN UNIVERSITY - CANADA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SE 3314B – COMPUTER NETWORKS APPLICATIONS

Assignment 3: Peer-to-Peer DHT Search

Due Date: April 10, 2023

This assignment is to be done individually. Cheating is prohibited in this assignment, that is not to show your code to any other student, do not look at any other student's code, and do not put your code in any public domain. However, getting help is easy and available, just ask the instructor or the TAs. At the same time, you are encouraged to discuss your problems (if any) with other students in general terms, but the actual program design and code must be your own.

Please note that the consequences of cheating are much, much worse than getting a low mark for that particular item of work. The very best case is that you get a zero for the whole thing. Keep in mind that turning in your own work would have at least gotten you a few marks, better than a zero.

Code submissions will be graded manually, and the code similarity check will be made using the similarity-detection software system.

1. Objectives

In this assignment you are to build a peer-to-peer (p2p) Kademlia network and perform a search for an image on the p2p DHT based-network. Particularly, you will implement a KAD peer node similar to the one you implemented for assignment 2. This peer node will also support client image query to search for images. Again, please before you continue working on this assignment, you must refresh your knowledge, on how the DHT and Kademlia network built and works.

2. Task 1: A Peer Node

Your first task is to write a KADpeer node. If you've implemented assignment 2 and have decided to build this assignment on top of your working code for assignment 2, you're done with the first task of this assignment. If you have not implemented assignment 2, no problem you still have the chance to practice socket-programming in this assignment by exploring, studying, and using the given code for KADpeer node that is included as a sample solution for assignment2. It is also expected, but not necessarily, to re-use some of the methods that is built for the previous two assignments.

3. Task 2: Keys assignment and peer ID re-generations

To make this assignment short, we will assign the responsibilities of the keys manually into the KAD peers. We will assign one key (image file) to each peer within a KAD network of 5 peers only. To do so, we will have the following assumptions:

- The peer ID is the hash value of the peer's IP and port numbers
- The key ID is the hash value of the image name.
- The port numbers of the image sockets are any random number greater than 3000 and less than 5000.

- Each peer will have an image with a keyID that is closed to the peer ID. To do so, we will assign the port numbers of the peers (to be used for the peer sockets) as follows.
  - peer1 → port 2001
  - peer2 → port 2055
  - peer3 → port 2077
  - peer4 → port 2044
  - peer5 → port 2005
- We will create five folders and add one image in each folder as per the following assignment:
  - Folder named "peer1" has the image "Canna.gif"
  - Folder named "peer2" has the image "Flicker.jpeg"
  - Folder named "peer3" has the image "CherryBlossom.gif"
  - Folder named "peer4" has the image "Parrot.jpeg"
  - Folder named "peer5" has the image "Cardinal.jpeg"

Once the new peer (the sender peer) wishes to connect to any KAD peer (the receiver peer), it will recognize the image file it has in its main folder and then generate an image ID for it. These images, from all peers, are the keys that we need to share within the DHT Kademlia network. To do so, each peer will maintain a **local keys list** to record the IDs of its local images (one image in our case), each element in this list is a pair of image ID and the corresponding image name. As we did in assignment2, we will also use the cryptographic hash function SHA-1 to generate the images IDs. The code of the SHA-1 is given with the Singleton.js file along with the sample solution of assignmnet2, feel free to call the method `getKeyID()` whenever you need to generate an ID for a given image. For example, if the image name is "CherryBlossom.gif", then the ID of this image is equal to the output of `singleton.getKeyID("CherryBlossom.gif")`.

4. Task 4: Client Image Query

Your next task is to integrate the server functionality described in assignment 1 (that accept image query request and then respond by sending the requested image) with the functionality of the KADpeer program from assignment 2 in a new program (server) which we will call it **KADpeerDB**. This new server will use two different ports: one to handle KAD network traffic and another to handle image query traffic. You get the first when you instantiate a peer object, which will accept peer join requests. The second comes with the instantiation of the image object, which will accept image query requests. We will call the former the ***peer socket*** and the latter the ***image socket*** henceforth. We will use one image socket for both client query and peer image-search reply (next task).

When a client queries for an image (using **GetImage** client program), the server peer first searches its own database (or rather, its local keys list) for the requested file name. If the image is found, it is returned to the client using the ITP response packet format (see Figure 2) and the connection is then closed. If the requested image is not found, the server peer (we well call it the originating peer) will search for this image in the KAD network on behalf of the client. We will discuss how to handle KAD search in the next task. To keep things simpler for you, you always search for images that exist in the KAD network.

At this point, you should test your code and verify that your **GetImage** client and **KADpeerDB** server work as in assignment 1 to serve up image files that are local to the server.

5. Task 5: KAD Search

When a peer N cannot find the requested key K locally, N will try to find the closest peer P in KAD network that contains the value associated with the key K. To do so, a search packet is sent (using the ***peer socket***) to the closest peer in N's DHT table. The method is using search/query packet format described in Figure1 below. When a search packet arrives at a peer, the recipient peer must also check whether it has the requested image. If it does not have a copy of the image, the peer forwards the search packet further to the closest peer to the key in its DHT table, and the process continues. When a queried image is found, say in peer M, then M creates a new socket and connects to the originating peer at the address and port number listed in the search packet (again using the ***peer socket***). Once the image transfer is completed, the connection is closed by the peer initiating the transfer. The originating peer then forwards the received image to the client requesting it using the ***image socket*** and closes the connection to the client.

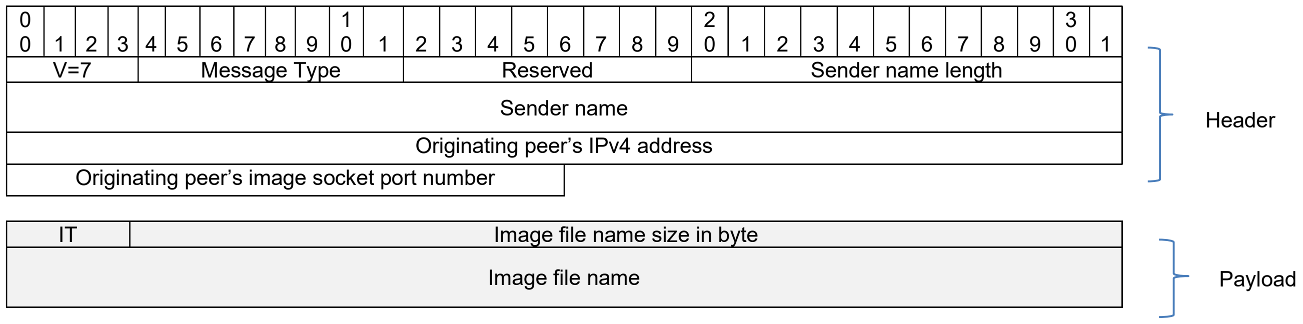


Figure 1: The kadPTP search request packet format

Here are the descriptions of the kadPTP search request packet fields:

- (V) is a 4-bit kadPTP version field. You must set this to 7.
- The message type is one byte to represent the message type; the value 3 means '**Search'** type.
- Reserved is an 8-bit field, not used in this assignment.
- The Sender name field holds the message sender name, the size of this field is dictated by the Sender name length field above.
- After the sender's name, the search packet must carry the originating peer's IP address and its image socket's port number.
- Again, the port number in the search message is that of the image socket, NOT the peer socket. The peer initiating an image search sends a copy of this search packet to the closest peer to the key in its DHT table.
- (IT) Image Type is a 4-bit field, part of the payload, indicates the type of the images as follows: 1 - BMP, 2 - JPEG, 3 - GIF, 4 - PNG, 5 - TIFF, and 15 – RAW.
- The file name size is a 28-bit field hold the number of bytes needed to store the image file name.
- The image file name, without the file extension is a field to store the name of the requested image. The size of this field is equal to the length of the image file name.

6. Task 6: Receive and display the requested image

Once the originating peer received (or already has) the requested image, it transfers the image to the client using the same protocol developed in assignment 1, you MUST precede the image with the ITP response packet format as follows:

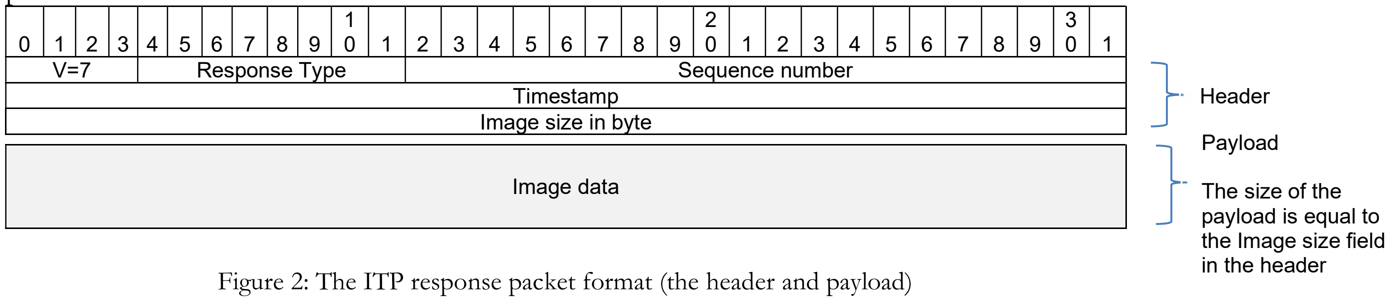


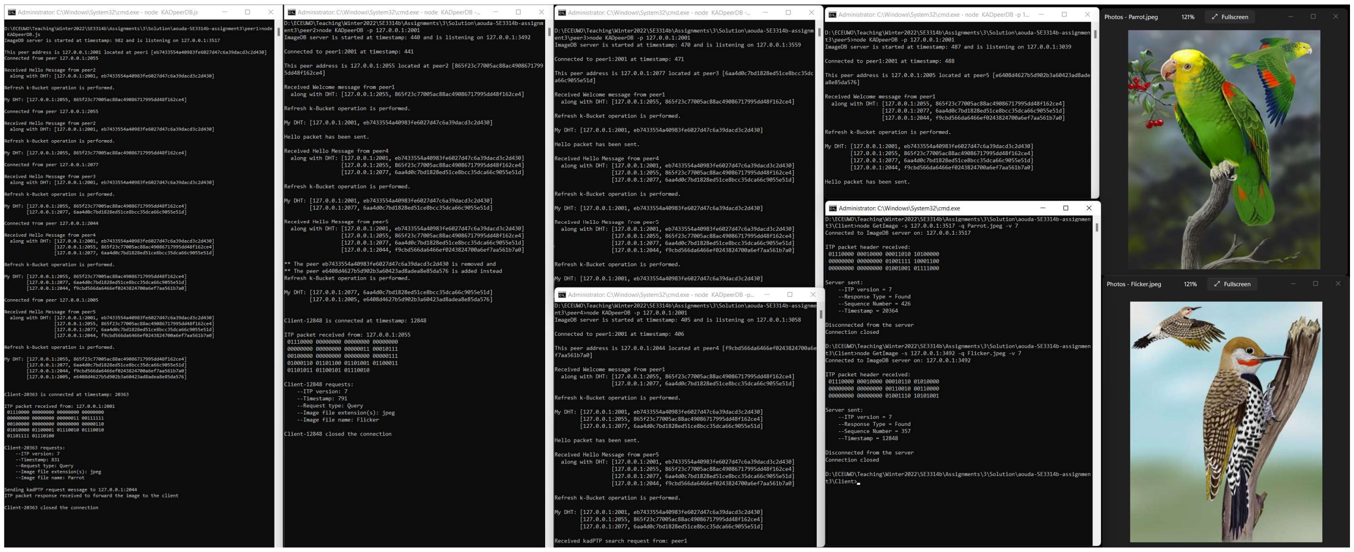
Figure 2: The ITP response packet format (the header and payload)

Here are the descriptions of the ITP response packet fields:

- (V) is a 4-bit ITP version field. You must set this to 7.
- The response type is 8-bit value so that the value 0 means "Query", 1 means "Found to Client", 2 and 3 are not used in this assignment, and 4 means "Found to Originator".
- Set the sequence number. The sequence number of the first packet is chosen randomly; it is incremented by 1 for each subsequent packet sent out of the server. This field is 20 bits in length and the increment will be performed mod 2^{20}.
- The timestamp is a 32-bit field has the current value of the server's timer.
- The image size is a 32-bit field which we will need to extract the image data from the packet and display it.
- The actual image data (part of the payload) will then follow the image file name field.

7. Testing your code

Below is a running scenario as an example to show the execution of the program and to summarize the requirements of this assignment. Create five different folders in your computer and name them: peer1, peer2, peer3, peer4, and peer5. After creating these folders, copy and paste the provided images as per the map we described in Section 3 above. Also copy your program in all these 5 folders. Open 6 command line windows, and perform the command as shown in the figure below.


Administrator: C:\Windows\System32\cmd.exe - node KADpeerDB.js
Administrator: C:\Windows\System32\cmd.exe - node KADpeerDB -...
Administrator: C:\Windows\System32\cmd.exe - node KADpeerDB -p 1...
Administrator: C:\Windows\System32\cmd.exe - node KADpeerDB -p...
Administrator: C:\Windows\System32\cmd.exe
Photos - Parrot.jpeg
121%
Fullscreen
Photos - Flicker.jpeg
121%
Fullscreen

```
D:\ECEUWO\Teaching\Winter2022\SE3314b\Assignments\3\Solution\aouda-SE3314b-assignment3\peer1>node
 KADpeerDB.js
ImageDB server is started at timestamp: 982 and is listening on 127.0.0.1:3517

This peer address is 127.0.0.1:2001 located at peer1 [eb7433554a40983fe6027d47c6a39dacd3c2d430]
Connected from peer 127.0.0.1:2055

Received Hello Message from peer2
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]

Connected from peer 127.0.0.1:2055

Received Hello Message from peer2
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]

Connected from peer 127.0.0.1:2077

Received Hello Message from peer3
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
        [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]

Connected from peer 127.0.0.1:2044

Received Hello Message from peer4
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
                  [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
        [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
        [127.0.0.1:2044, f9cbd566da6466ef0243824700a6ef7aa561b7a0]

Connected from peer 127.0.0.1:2005

Received Hello Message from peer5
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
                  [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
                  [127.0.0.1:2044, f9cbd566da6466ef0243824700a6ef7aa561b7a0]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
        [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
        [127.0.0.1:2044, f9cbd566da6466ef0243824700a6ef7aa561b7a0]
        [127.0.0.1:2005, e6408d4627b5d902b3a60423ad8adea8e85da576]


Client-20363 is connected at timestamp: 20363

ITP packet received from: 127.0.0.1:2001
 01110000 00000000 00000000 00000000
 00000000 00000000 00000011 00111111
 00100000 00000000 00000000 00000110
 01010000 01100001 01110010 01110010
 01101111 01110100

Client-20363 requests:
    --ITP version: 7
    --Timestamp: 831
    --Request type: Query
    --Image file extension(s): jpeg
    --Image file name: Parrot

Sending kadPTP request message to 127.0.0.1:2044
ITP packet response received to forward the image to the client

Client-20363 closed the connection
```

```
D:\ECEUWO\Teaching\Winter2022\SE3314b\Assignments\3\Solution\aouda-SE3314b-assignm
ent3\peer2>node KADpeerDB -p 127.0.0.1:2001
ImageDB server is started at timestamp: 440 and is listening on 127.0.0.1:3492

Connected to peer1:2001 at timestamp: 441

This peer address is 127.0.0.1:2055 located at peer2 [865f23c77005ac88ac4908671799
5dd48f162ce4]

Received Welcome message from peer1
  along with DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]

Hello packet has been sent.

Received Hello Message from peer4
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
                  [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
        [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]

Received Hello Message from peer5
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
                  [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
                  [127.0.0.1:2044, f9cbd566da6466ef0243824700a6ef7aa561b7a0]

** The peer eb7433554a40983fe6027d47c6a39dacd3c2d430 is removed and
** The peer e6408d4627b5d902b3a60423ad8adea8e85da576 is added instead
Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
        [127.0.0.1:2005, e6408d4627b5d902b3a60423ad8adea8e85da576]


Client-12848 is connected at timestamp: 12848

ITP packet received from: 127.0.0.1:2055
 01110000 00000000 00000000 00000000
 00000000 00000000 00000011 00010111
 00100000 00000000 00000000 00000111
 01000110 01101100 01101001 01100011
 01101011 01100101 01110010

Client-12848 requests:
    --ITP version: 7
    --Timestamp: 791
    --Request type: Query
    --Image file extension(s): jpeg
    --Image file name: Flicker

Client-12848 closed the connection
```

```
D:\ECEUWO\Teaching\Winter2022\SE3314b\Assignments\3\Solution\aouda-SE3314b-assignm
ent3\peer3>node KADpeerDB -p 127.0.0.1:2001
ImageDB server is started at timestamp: 470 and is listening on 127.0.0.1:3559

Connected to peer1:2001 at timestamp: 471

This peer address is 127.0.0.1:2077 located at peer3 [6aa4d0c7bd1828ed51ce8bcc35dc
a66c9055e51d]

Received Welcome message from peer1
  along with DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]

Hello packet has been sent.

Received Hello Message from peer4
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
                  [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]

Received Hello Message from peer5
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
                  [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
                  [127.0.0.1:2044, f9cbd566da6466ef0243824700a6ef7aa561b7a0]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
```

```
D:\ECEUWO\Teaching\Winter2022\SE3314b\Assignments\3\Solution\aouda-SE3314b-assignme
nt3\peer4>node KADpeerDB -p 127.0.0.1:2001
ImageDB server is started at timestamp: 405 and is listening on 127.0.0.1:3058

Connected to peer1:2001 at timestamp: 406

This peer address is 127.0.0.1:2044 located at peer4 [f9cbd566da6466ef0243824700a6e
f7aa561b7a0]

Received Welcome message from peer1
  along with DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
        [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
        [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]

Hello packet has been sent.

Received Hello Message from peer5
  along with DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
                  [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
                  [127.0.0.1:2044, f9cbd566da6466ef0243824700a6ef7aa561b7a0]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
        [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
        [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]

Received kadPTP search request from: peer1
```

```
D:\ECEUWO\Teaching\Winter2022\SE3314b\Assignments\3\Solution\aouda-SE3314b-assignmen
t3\peer5>node KADpeerDB -p 127.0.0.1:2001
ImageDB server is started at timestamp: 487 and is listening on 127.0.0.1:3039

Connected to peer1:2001 at timestamp: 488

This peer address is 127.0.0.1:2005 located at peer5 [e6408d4627b5d902b3a60423ad8ade
a8e85da576]

Received Welcome message from peer1
  along with DHT: [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
                  [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
                  [127.0.0.1:2044, f9cbd566da6466ef0243824700a6ef7aa561b7a0]

Refresh k-Bucket operation is performed.

My DHT: [127.0.0.1:2001, eb7433554a40983fe6027d47c6a39dacd3c2d430]
        [127.0.0.1:2055, 865f23c77005ac88ac49086717995dd48f162ce4]
        [127.0.0.1:2077, 6aa4d0c7bd1828ed51ce8bcc35dca66c9055e51d]
        [127.0.0.1:2044, f9cbd566da6466ef0243824700a6ef7aa561b7a0]

Hello packet has been sent.
```

```
D:\ECEUWO\Teaching\Winter2022\SE3314b\Assignments\3\Solution\aouda-SE3314b-assignmen
t3\Client>node GetImage -s 127.0.0.1:3517 -q Parrot.jpeg -v 7
Connected to ImageDB server on: 127.0.0.1:3517

ITP packet header received:
 01110000 00010000 00011010 10100000
 00000000 00000000 01001111 10001100
 00000000 00000000 01001001 01111000

Server sent:
    --ITP version = 7
    --Response Type = Found
    --Sequence Number = 426
    --Timestamp = 20364

Disconnected from the server
Connection closed

D:\ECEUWO\Teaching\Winter2022\SE3314b\Assignments\3\Solution\aouda-SE3314b-assignmen
t3\Client>node GetImage -s 127.0.0.1:3492 -q Flicker.jpeg -v 7
Connected to ImageDB server on: 127.0.0.1:3492

ITP packet header received:
 01110000 00010000 00010110 01010000
 00000000 00000000 00110010 00110000
 00000000 00000000 01001110 10101001

Server sent:
    --ITP version = 7
    --Response Type = Found
    --Sequence Number = 357
    --Timestamp = 12848

Disconnected from the server
Connection closed

D:\ECEUWO\Teaching\Winter2022\SE3314b\Assignments\3\Solution\aouda-SE3314b-assignmen
t3\Client>
```

Hand In

- Your source code should be fully commented and formatted.
- Your project folder structure should be created such that all code for your program and package.json file should be under one folder only named '**peer1**'. Don't add the image files.
- No need to include the client program, we will use the sample code we developed for assignment1.
- Compress this folder in an achieve file (zip file) and name it ***yourUWOID*-SE3314b-assignment3.zip**.
- Submit this zip file on OWL by the due date mentioned above.