z5145006 Chuguan Tian

# COMP9331 Assignment Report

## 1.Learning objectives

Gaining sufficient expertise in the following skills:

1. Understanding of routing mechanism and connectivity maintenance in peer-to-peer systems, and Circular DHT in particular.
2. Socket programming for both UDP and TCP transport protocols.
3. Protocol and message design for applications.

## 2.Background

The following is extracted from the Peer Churn section of the text:
In P2P systems, a peer can come or go without warning. Thus, when designing a DHT, we also must be concerned about maintaining the DHT overlay in the presence of such peer churn. To get a big-picture understanding of how this could be accomplished, let's once again consider the DHT in Figure 2.27(a) [Reproduced here as Figure 1]. To handle peer churn, we will now require each peer to track (that is, know the IP address of) its first and second successor; for example, peer 4 now tracks both peer 5 and peer 8. We also require each peer to periodically verify that its two successors are alive (for example, by periodically sending ping messages to them and asking for responses). Let's now consider how DHT is maintained when a peer abruptly leaves. For example, suppose peer 5 in Figure 2.27(a)[Figure 1 in this assignment spec] abruptly leaves. In this case, the two peers preceding the departed peer (4 and 3) learn that 5 has departed, since it no longer responds to ping messages. Peers 4 and 3 thus need to update their successor state information. Let's consider how peer 4 updates its state:

1. Peer 4 replaces its first successor (peer 5) with its second successor (peer 8).
2. Peer 4 then asks its new first successor (peer 8) for the identifier and IP addresses of its immediate successor (peer 10). Peer 4 then makes peer 10 its second successor.

Having briefly addressed what has to be done when a peer leaves, let's now consider what happens when a peer wants to join the DHT. Let's say a peer with identifier 13 wants to join the DHT, and at the time of joining, it only knows about peer 1's existence in the DHT. Peer 13 would first send peer 1 a message, saying "what will be peer 13's predecessor and successor?" This message gets forwarded through the DHT until it reaches peer 12, who realises it will be peer 13's predecessor and its current successor, peer 15, will become its successor. Next, peer 12 sends this predecessor and successor information to peer 13. Peer 13 can now join the DHT by making peer 15 its successor and by notifying peer 12 that it should be its immediate successor to peer 13.

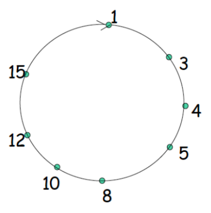

Figure 1. DHT configuration.

## 3.Assignment implementation

In this Assignment, I implement DHT for P2P with Java. In Java source  file cdht.java I use multi-thread to implement the PingServer, PingClient, TCP-message client, TCP-message server and main thread to start other threads and listen to the command from console.

**Step 1: Ping**

Firstly, assigning identifier and successors by input arguments, then creating DatagramSocket which bind IP address as “localhost” and port number as identifier+50000 to receive and send ping messages through UDP.

The ping message contains word “request” and sequence number, and creating static variables named seq_num_1 and seq_num_2 which indicate the sequence number of messages between current peer and its first and second successors. And seq_num_1 and seq_num_2 decides what sequence number should be send along the request message as well as it received from response. PingServer receive a ping request, it will response a UDP message contains word “response” and the same sequence number with the request. After receiving the response from successors, the corresponding seq_num_1 or seq_num_2 add one and continue sending the ping, which means the sequence number would be increased only when server receive the response message with the same sequence number. The messages that indicate the success of pinging e.g. “A ping response message was received from Peer” and “A ping request message was received from Peer” only occurs when the sequence number is 0, so they only be printed at the beginning of pinging successfully.

With the request messages of ping, peer recognise the identifier of its predecessor peers. If peer realise identifier of its first predecessor is greater than identifier of itself, then it marks itself as the first peer of the DHT circule.

**Step 2: File Requesting**

Main thread is listening to the commands from console, when we get “request xxxx”, then starting TCP server and send message containing “requestFile, identifier of requesting peer, hash code of filename” to peer’s first successor .

If a peer receives a file request the peer it compare the hash code with its identifier and its predecessor’s to decide if the file is stored in itself. If peer find that the hash code of requested file is greater that it’s first predecessor and smaller than its or the hash code equals to its identifier or if the first peer receive the request whose hash code of file is greater than its identifier, it realises the file is stored in itself . Then it sends file response containing “response ,its identifier, filename” to the original requesting peer. ( because TCP messages in this program is sending from auto-arranged port so it mush send its identifier to the original requesting peer ) If peer find it does not satisfy above conditions it pass the original request to its first successor.

**Step 3: Quit**

When main thread listens to the console and gets the command “quit”. Then it send message to its first and second predecessors with identifiers of its first and second successors to advises them the identifier of their further first and second successors. When peers receive the quit message from other peers, they change their new successors according to the information from message.

For the ping client, every time after it send the ping request it will update the current identifier to an array which stores the previous successors’ identifiers, And every time before it send the ping request it will check if the current successor is the same with previous one, if not, the ping client will reset the sequence number to 0 and start ping the new successors. As the mechanism of ping server and client discussed in step one, when receiving the ping request whose sequence number is 0, console will print the request and response message of new connections to console.

**Step 4: Kill peer**

Every 1 second, ping client sends a ping message to the peers' successors. Corresponding to each ping request, it will receive a response with same sequence number. If it does not receive the response, the client will send the request with the previous sequence number again. Now I creating two int variables to calculate the times that the ping client sends the ping messages with the same sequence number continuously. After sending the same sequence number to the first successor continuously over 10 times, which means do not receive the response from the first successor for 10 ping requests, it realises that the peer is killed, and it sends asking message to its second successor to find the new second successor. Similar to first successor, if peer finds that it has sent the same sequence number to the second successor continuously over 15 times (the time is greater than that of first successor killed condition because it must give time to its first successor to reset the first successor's new successors ) it will sends asking message to its first successor and try to reset the successors.

Pinging mechanism is similar to that in step 3.

---

## 4.Video

https://youtu.be/m6sx-UsCPCI