THE UNIVERSITY OF SYDNEY

COMP2121 Lab 7

Peer-to-Peer

The goal of this lab is to build a simple P2P application where each peer maintains a list of other active peers.

Exercise 1: A P2P heartbeat protocol

A *heartbeat* is a simple message created by the process to indicate normal activity to other parts of a computer system. This heartbeat message is sent between machines every few seconds. If the network delivers each message within a given time bound and the endpoint does not receive a heartbeat for a long time (usually a few heartbeat intervals) the machine that should have sent the heartbeat is assumed to have failed.

Figure 1 depicts a client-server protocol where client P uses heartbeats to detect the failure of server Q. Implement the following protocol with T equals 2 seconds in **`HeartBeatClientRunnable`** and **`HeartBeatServerRunnable`** classes.

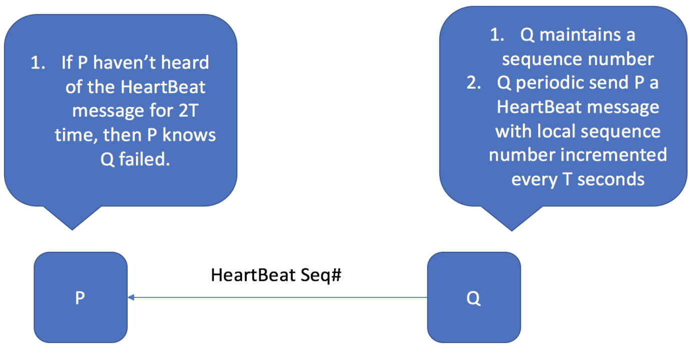

1. If P haven't heard of the HeartBeat message for 2T time, then P knows Q failed.
1. Q maintains a sequence number
2. Q periodic send P a HeartBeat message with local sequence number incremented every T seconds
P
HeartBeat Seq#
Q

Figure 1: A client-server protocol using heartbeat messages between a client P and a server Q

A *peer-to-peer* (or P2P) application is a special kind of program in which each peer acts as both a client and a server at the same time. The next step is to convert our application from the client-server model to the P2P model since we would like to let Q know P's activity as well!

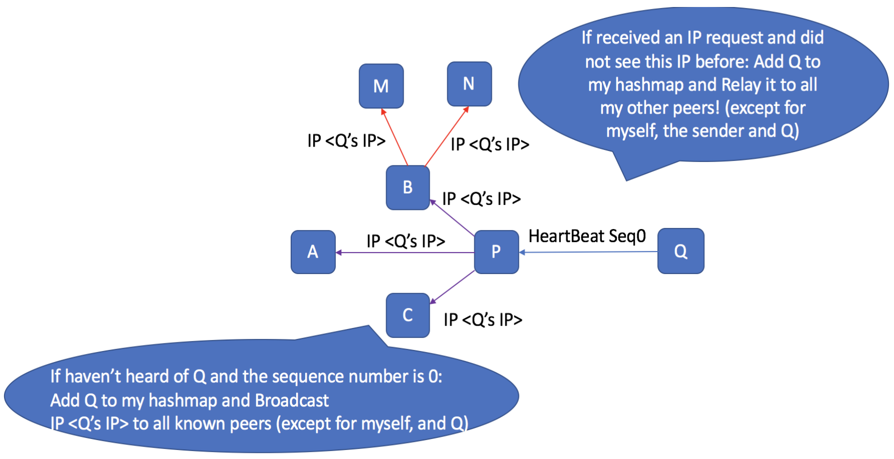
In Java, we achieve this by combining multiple client and server threads together. The client thread still has the responsibility to send messages to another process, while the server thread handles requests coming from the other side.

Fill in the **`HeartBeat`** class and test it with the person next to you, make sure you can see the heartbeat message from your neighbour and that your neighbour can observe your heart-beat message as well.

Duration: 25 min

Exercise 2: Dynamic neighbour discovery

During Assignment 2, when we implement the multicast and broadcast protocol, we let the client read the config file to acquire server information. In reality, this is not feasible if there are thousands of servers involved. Automatic peer discovery is thus required. Here, we introduce a simple dynamic neighbour discovery protocol on top of the previous heartbeat application. The protocol is defined as follows. Please implement all the requirements in **`HeartBeatServerRunnable`** class.


If received an IP request and did not see this IP before: Add Q to my hashmap and Relay it to all my other peers! (except for myself, the sender and Q)
M
N
IP <Q's IP>
IP <Q's IP>
B
IP <Q's IP>
A
IP <Q's IP>
P
HeartBeat Seq0
Q
C
IP <Q's IP>
If haven't heard of Q and the sequence number is 0: Add Q to my hashmap and Broadcast IP <Q's IP> to all known peers (except for myself, and Q)

Duration: 20 min

Exercise 3: Network message delay

Previously we assumed that the network was delivering each message within a given time period. What would happen if network messages could be arbitrarily delayed?

Duration: 5 min