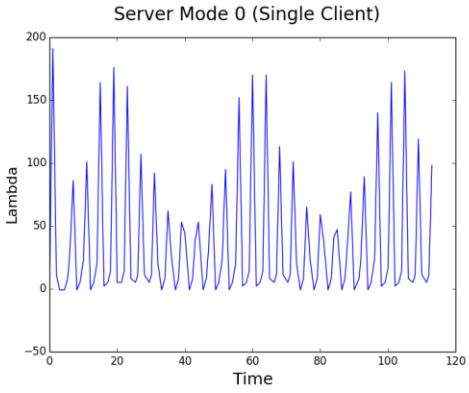
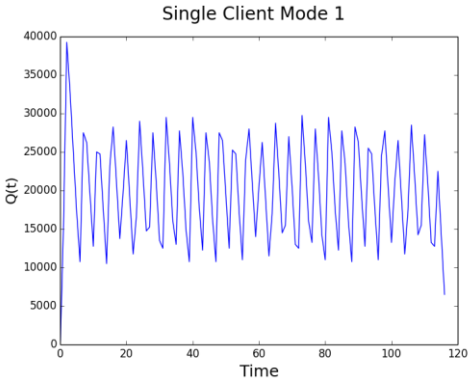
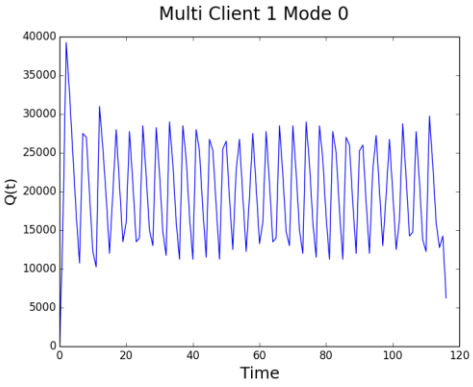
Lab 5

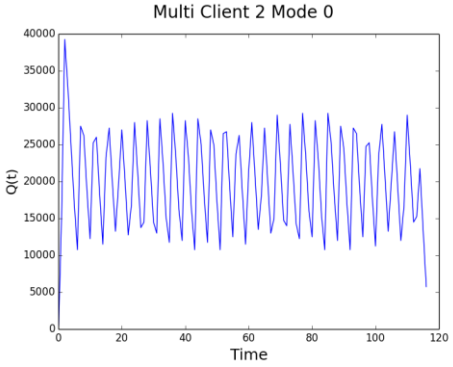
Question 1

Mode 0:
a=3.0





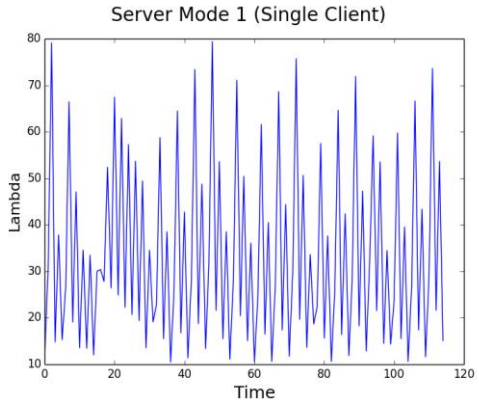


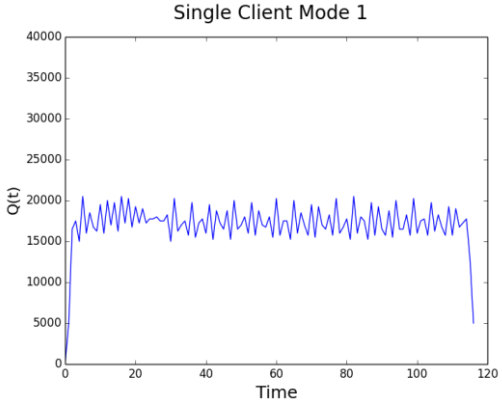


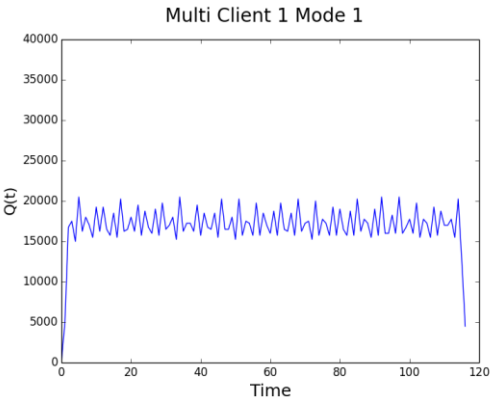
The audio quality was flaky. There were times when the audio broke in between and then continued after wards. However, change ‘a’ from 1 to 3 made the quality better. Two clients did not hamper the quality a lot but there were slightly bigger breaks in the audio at alternative

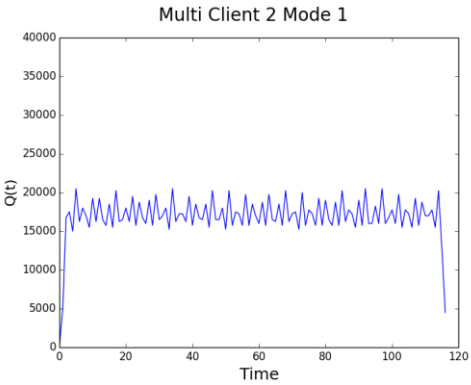
places. The figures are coherent with the fact that lambda becomes 0 at times. With a=1, there were patches where buffer reached its full capacity or became completely empty.

Mode 1:
delta=0.75
a=1.0



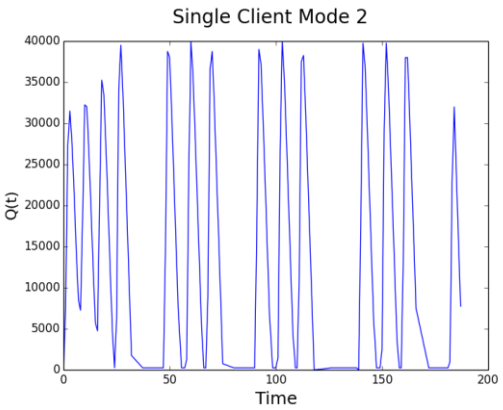


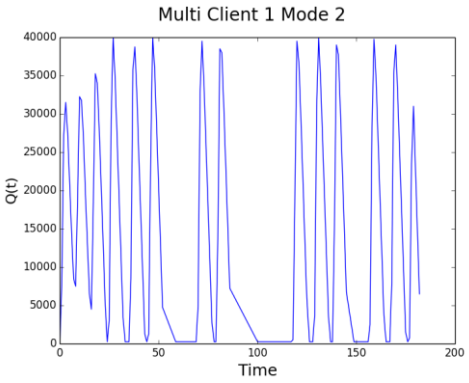


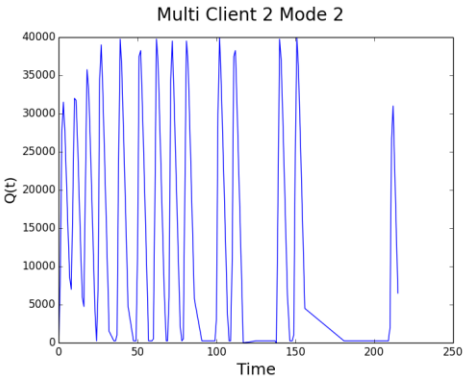


The audio quality was better than mode 0. The audio did not break in between. Two clients did not hamper the quality much. The pattern seen is in coherence with the fact that lambda does not go below 0 and the buffer oscillates around the mean.

Mode 2:
Epsilon=0.0001



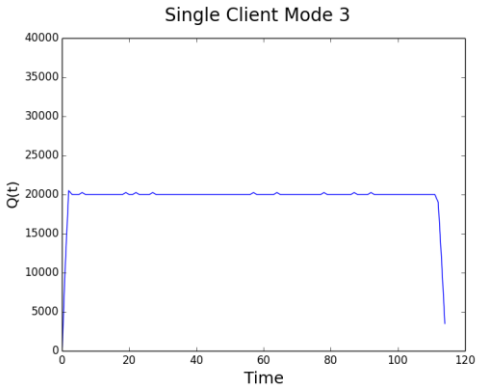


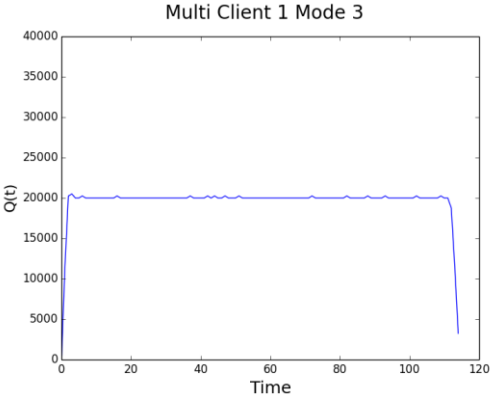


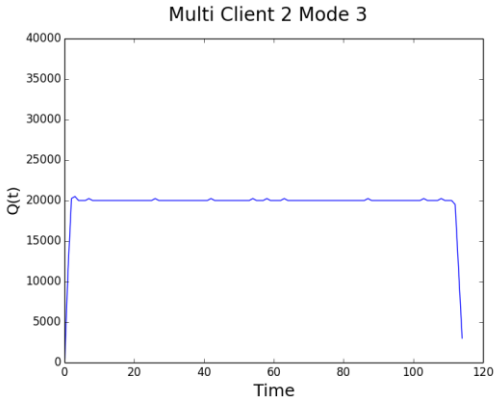
This mode was the worst in terms of performance. There were breaks in between and the buffer capacity oscillated between empty and full. Lambda varied by large amplitude and

became –ve or 0 a lot of times. Transferring audio to two clients worsened the position where one of the clients finished much later than the other client.

Mode 3:
Epsilon=.001
Beta=1.0







This mode, as expected, gave the best results by stabilizing very quickly. The audio was perfectly transferred and two clients also did not get hampered.

Question 2:

Server side data structure

Hashtable implemented as an array of size audiobuf where each packet is stored in memory at location seq_num%audiobuf.

This ensures that I have exactly audiobuf number of packets in memory for retransmission. The current packet overwrites the stale packet in the hashtable.

Client side data structure

I have used a combination of hashtable and circular buffer to maintain a data structure that can accommodate holes.

Each unit of my circular buffer is of size payload_size. Every new packet (not discarded) is written to next available position. Every packet that is to be discarded is stored as a hole at next available position for writing in the circular buffer.

The hashtable is of size of total number of sequence numbers possible. Each discarded packet's write position in circular buffer is stored at seq_num position in this hashtable.

Whenever a retransmitted packet comes in, it looks up the position of hole in circular buffer from the hash table and writes the valid value there.

If the current read position is ahead of the retransmitted packet, it is discarded. While reading, if the reader encounters a hole (garbage value), it is not read.