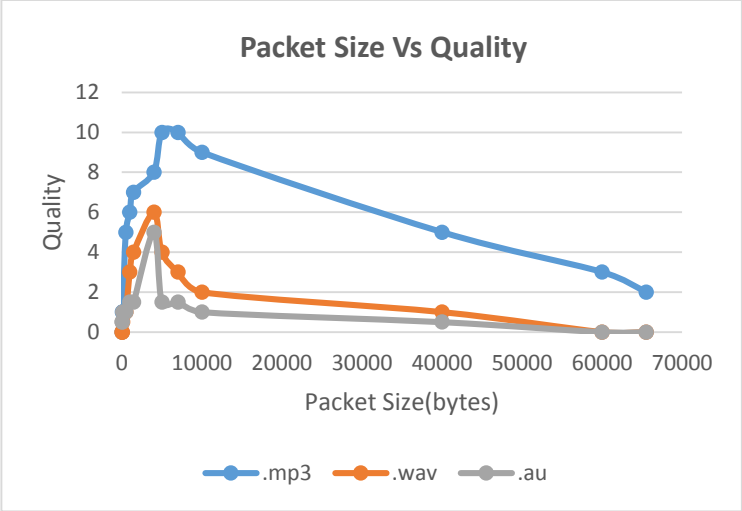


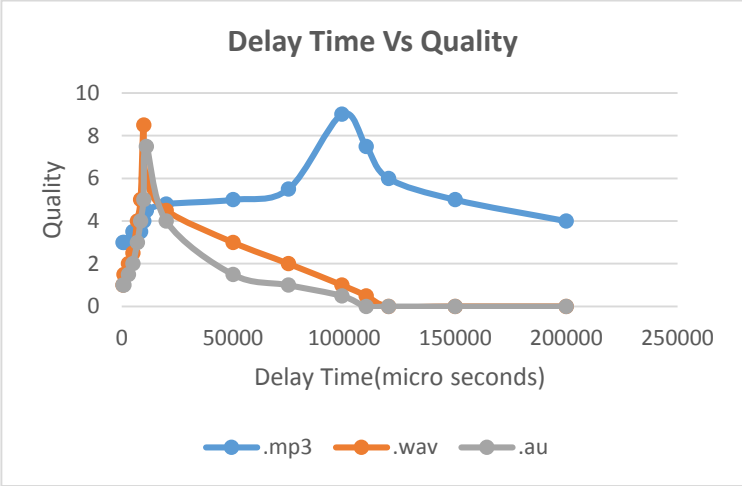
Receiver Bandwidth: 17.54 Mbps
Delay time: 0 ms
File name: quartet (in .mp3, .wav, .au formats)

Factors	Explanation
Small packet size	Freezing and skipped parts of audio. Out of order packets received.
Large packet size	Evident gap between the stream. Number of packets lost were high.

(Fig 1: Packet size Vs. Quality)

The transmitted packet size was limited to 65507 bytes after which it gave a “message too long” error.

The **bandwidth** (Internet speed) was also a factor affecting the transmission. The same packet size which provided the highest quality did not assure the same quality in a different bandwidth. A drop in the bandwidth resulted in a poor experience, with stopped or skipped parts. The quality of .wav and .au play outs improved with higher bandwidths. Despite the difference in the speed the quality slightly increased with packet size and started dropping after a point for all types of audio.



Receiver Bandwidth: 10 Mbps
Packet size: 2000 bytes
File name: quartet (in .mp3, .wav, .au formats)

Factors	Observation
Short delay time than highest quality	Mixing up of track, similar to fast forwarding.
Long delay time than highest quality	Jitter effect is large and caused jerky or disjointed audio

(Fig 2: Delay Time Vs. Quality)

A several second initial delay for the play out to begin observed at higher delays mainly for .wav and .au audios. In this case client side buffering was observed and it was possible to pause the stream and then play.

For packet size 2000 bytes, .mp3 with bit rate of 160 kbps played the best quality at delay of around 99 ms delay which is very close to the time taken for the audio consumption 100ms(2000*8/160k). The small difference could be because of the other factors like network delay. Similarly .wav with 1536 kbps bitrate worked best at around 9.8 ms delay.

Moreover, apart from the affecting factors of **packet size** and **delay time**, the **division of larger sized packets** into 1500 sized MTU packets was an affecting factor in the quality as it dropped the further divided packets of a large packet when one of them got lost.

Another affecting factor was the **jitter delay** which affected the quality in an unusual manner at higher delays causing playback pauses or skipped parts.

Now for the "violin_viola_cello" file in all three formats showed the same trends as "quartet" with a difference in delay time value for different packet size of 4000 bytes , 199 ms high quality delay for .mp3 format was observed and the .wav & .au formats gave the best quality at 20.5 ms for a 4000 bytes packet which is close to 20 ms(4000*8/1536k). Moreover, the observed trend was similar for both the parameters with the slight difference in high quality delay time with a different packet size.

Commands Used	**Description**
./ffmpeg –i quartet.mp3 quartet.wav	To convert .mp3 format to .wav format
./ffprobe quartet.wav	To check the bit rate and audio encoding
./UDP 10.200.244.45 11000 quartet.mp3 0 10000 0	To run the radio transmitter at the provided IP address and port number with delay time (1st parameter) and packet size (2nd parameter).

CODE:

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include<stdlib.h>
 int main (int argc, const char* argv[]) {
    struct sockaddr_in saddr;
    int fd, ret_val, 0,n,port,s,cnt;
    struct hostent *host;
    char *DATA_BUFFER;
    FILE *file_ptr;
    port = atoi(argv[2]);
    s = atoi(argv[4]);     //delay time between 2
packets
    cnt = atoi(argv[5]);   //packet size
    DATA_BUFFER = malloc(cnt * sizeof(char));
//allocating size of buffer according to the packet
size we input
    file_ptr = fopen(argv[3],"rb");    //rb is for read
bytes of data from a file
    if (file_ptr == NULL)
      printf("mp3 file not found");
    fd = socket(AF_INET, SOCK_DGRAM,0);
    if (fd == -1) {
       fprintf(stderr, "socket failed [%s]\n",
strerror(errno));
       return -1;
    }
    saddr.sin_family = AF_INET;
    saddr.sin_port = htons(port);
    host = gethostbyname(argv[1]);
    saddr.sin_addr = *((struct in_addr *)host-
>h_addr);
    bzero(DATA_BUFFER, cnt);
    for (i=0;!feof(file_ptr);i++){
      n = fread(DATA_BUFFER, 1,cnt, file_ptr);    //this
line will take cnt bytes from file_ptr and put into
buffer, and it will make only 1 packet
      ret_val = sendto(fd,DATA_BUFFER,n, 0, (struct
sockaddr *)&saddr, sizeof(struct sockaddr_in));
      if (ret_val < 0) {
         printf("sendto() failed [%s]\n",
strerror(errno));
      }
      bzero(DATA_BUFFER, cnt);
            usleep(s);
   }
    fclose(file_ptr);
    close(fd);
    free(DATA_BUFFER);
    return 0;
 }
```