I3 実験レポート

22 班 A 03190513 宮野広基 03190449 堀 紡希

8 月 1 日

1　達成した項目

・コマンドライン引数の違いによって、同じプログラムからサーバ、クライアントのどちらも実行できるようにした。
・データ数 n に対して線形時間のオーダー O(n) に比例する電話料金を表示できるようにした。
・Enter+scanf ではなく擬似 kbhit 関数でキーを押した瞬間にキーの値の取り込みができるようにした。
・音声データを送信する前に高速フーリエ変換を行い、人間の声の領域以外の成分を 0 にしてカットしてから送信、受信してから受信側がカットされた 0 の成分を補完することで送られるデータを軽くした。
・FFT した後の振幅が閾値より小さいものをカットすることで雑音を除去した
・FFT した後の 0 を補完する位置をずらすことで音の高さを変えた。
・着信音が鳴るようにした。
Ctrl+C で切らなくていいように通信切断ボタンを作った。
・講義で扱った FFT のプログラムを利用して, 送信側で音声を FFT し, 受信側で IFFT をするようにすることで通信データの量を削減, さらにバンドパスフィルタをかけた.
・rec の buffer の大きさを変えるオプションを調べ, 遅延と雑音が適切になるようにした.
・引数が足りない場合はヘルプを表示するようにした. また同時にプログラムから ifconfig を呼び出して自分の PC の IP アドレスを表示するようにした。
・rec の標準エラー出力を捨てるようにした. しかしデバッグの効率が下がったので結局実演では使わなかった.

2　問題点を克服する過程で行った調査や実験

・送受信する毎に 0.1 ドルを加算することで線形時間に比例する電話料金を実装した。試行錯誤した結果、これが実際の電話料金に最も近い値になった。
・音を高くする、低くするなどの信号を FFT して送るスペクトル列の最後に加えていたが、なぜか時々意図しない値に変わったので、char 型で受け付けたデータを十分離れた値の double 型の数値に変換して送ることで誤送信を防いだ。
・一般に使われている kbhit 関数が Linux で使うことができなかったので以下のように実装した。keyboard が hit されているときは 1, それ以外は 0 を返す。

```
int kbhit(void)
{
struct termios oldt, newt;
```

```
int ch;
int oldf;

tcgetattr(STDIN_FILENO, &oldt);
newt = oldt;
newt.c_lflag &= ~(ICANON | ECHO);
tcsetattr(STDIN_FILENO, TCSANOW, &newt);
oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);

ch = getchar();

tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
fcntl(STDIN_FILENO, F_SETFL, oldf);

if (ch != EOF) {
ungetc(ch, stdin);
return 1;
}

return 0;
}
```

・FFT をする前後のデータ量を調査した. 1 秒当たりのサンプル数を f_{sample} として unsigned char 型のデータとして送受信していたので,FFT 前の通信量は

$$f_{sample} * 1 * 2 = 2f_{sample}[byte] \tag{1}$$

だけのデータを送信していた. 一方でこの音声データを N 個の点で区切って FFT を行い, その結果から得た周波数スペクトルを double 型のデータとして送受信することを考える. また人間の声の成分が多く含まれる周波数スペクトルのみを取り出すこととした. このときバンドパスフィルタの下限と上限をそれぞれ $f_1, f_2[Hz]$ とすると, 周波数スペクトルの N 個の double 型のデータのうち $\frac{f_1 * N}{f_{sample}}$ から $\frac{f_2 * N}{f_{sample}}$ までのデータを送ればいいことが分かる. したがって 1 秒当たりの送信データ量は

$$\frac{f_{sample}}{N} * (\frac{f_2 * N}{f_{sample}} - \frac{f_1 * N}{f_{sample}}) * 8 * 2[byte] \tag{2}$$

$$= 16(f_2 - f_1) \tag{3}$$

ここで $f_{sample} = 44100$, N = 8192, $f_1 = 300$, $f_2 = 1500$ とすると, 式 (1) は 44100[byte/s] に対して式 (3) は 19200[byte/s] となり, 元の約 43.5% になっていることが分かる. したがってデータの圧縮ができた.

・通信が繋がった瞬間に rec と play がスタートできるように popen コマンドを用いてコマンドを実行することでサーバが通信を開始した時の音がクライアント側に聞こえるという問題点を解決したので以下にそのコードを示しておく。

```
char    *cmdrec = "rec --buffer 16384 -t raw -b 16 -c 1 -e s -r 44100 - ";
        if ( (fpr=popen(cmdrec,"r")) ==NULL) {
                perror ("can not exec commad");
                exit(EXIT_FAILURE);
        }

    FILE    *fpp;
    char    *cmdplay = "play --buffer 16384 -t raw -b 16 -c 1 -e s -r 44100 - ";
    if ( (fpp=popen(cmdplay,"w")) ==NULL) {
```

```
                perror ("can not exec commad");
                exit(EXIT_FAILURE);
        }
```

3 ソースコード

zip 中のファイルで添付しておく。着信音の wav ファイルも添付しておく。

4 参考文献

[1] 東京大学工学部：「電気電子情報第一 (前期) 実験テキスト」, 2019.