電気電子情報第一実験 G3 最終レポート 03-140486 下田隆貴

達成した課題一覧:

基本課題:双方向通話が可能なインターネット電話の実現
オプション課題 1:/dev/dsp のサンプリング周波数や量子化ビット数を変更による、通話品質の改善
オプション課題 2:TCP に関して、遅延の考察とその改善
オプション課題 3:音声データと同時に、テキスト入力も送信できるようなチャットシステムの実装

オプション課題 1:/dev/dsp のサンプリング周波数や量子化ビット数を変更による、通話品質の改善について

まとめ:
(1)問題点
初期設定で、/dev/dsp のサンプリング周波数は 8000 Hz、量子化ビット数は unsigned 8 bits である。サンプリング周波数については人間の可聴帯が 20000 Hz ぐらいなので、初期設定では高音領域の声が伝わらず、通話品質が低いという問題があったし、また量子化ビット数についても振幅の絶対値は(8 bits の半分の数だから)128 段階しかなく音の強弱の観点から言っても通話品質が低いという問題があった。
(2)克服方法
ioctl()関数を利用することにより、サンプリング周波数を 44100 Hz に、量子化ビット数を signed 16 bits に変更した。ともに CD の音質と同じである。これにより通話品質が改善した。
前者は
```
int freq=44100;   ioctl( fd, SOUND_PCM_WRITE_RATE, &freq );
```
として変更し、後者は
```
int fmt=AFMT_S16_LE;  ioctl( fd, SOUND_PCM_SETFMT, &fmt );
```
として変更した。(fd は/dev/dsp を open()関数の引数にとったもの)

課題達成または問題克服の過程に行われた調査や実験などの内容と結果:
/dev/dsp の仕様を調べるにあたって、文献[1]の web ページを参照した。
文献[2]にある/dev/dsp の初期設定の調べ方によると、今回使用した PC の初期設定は図 1.1 のようになった。またオプション課題 1 まとめ(2)克服方法で述べた ioctl()関数を使うと、図1. 2のように変更できた。

```
Sampling rate : 8000 Hz
Channels      : 1
Sample size   : 8 bits
Block size    : 1024 bytes

Supported formats
  mu-law
  unsigned 8 bit ( default )
  signed 8 bit
  signed 16 bit, little-endian
  signed 16 bit, big-endian
  unsigned 16 bit, little-endian
  unsigned 16 bit, big-endian
```

図1. 1:/dev/dsp の初期設定

```
Sampling rate : 44100 Hz
Channels      : 1
Sample size   : 16 bits
Block size    : 256 bytes

Supported formats
  mu-law
  unsigned 8 bit
  signed 8 bit
  signed 16 bit, little-endian ( default )
  signed 16 bit, big-endian
```

図1. 2:変更後の/dev/dsp

図1. 1と図1. 2を見ると、サンプリング周波数と量子化ビット数について、実際に、オプション課題 1 まとめ(2)克服方法で述べた通りに変更できていることが分かる。

オプション課題 2:TCP に関して、遅延の考察とその改善

まとめ:

(1)問題点

TCP 通信ではインターネット電話中の遅延があった。

(2)克服方法

問題の原因は/dev/dsp のそれぞれのバッファサイズが fragment サイズを超えるまで、そのバッファの書き込みと読み出しがブロックされてしまう仕様にあると考えた。

/dev/dsp に入出力されるバッファ(DMA バッファと呼ばれる)は図1. 3の通り fragment サイズ単位で分割されている。[3] 書き込まれるバッファのサイズが fragment サイズを超えるまでそのバッファの読み込みはできないので、それまでは送信側は音声の送信ができず、受信側は音声の再生ができない。

図1. 3:fragment サイズ単位で分割された DMA バッファ

そこで初期設定では 1024 bytes である fragment サイズを、ioctl()関数により 256 bytes に変更した。

int argx = 0x7fff0008;　ioctl(fd, SNDCTL_DSP_SETFRAGMENT, &argx);

として変更した。

なぜ 256 bytes かというと、fragment サイズは少なければ少ないほどリアルタイム性は改善されるが、fragment サイズが 256 bytes を下回ると書き込みが追いつかなくなり雑音が混ざることあるからである。[4]

計算上、

$$\frac{1}{\text{サンプリング周波数}} \times \frac{1}{\text{量子化ビット数}} \times \text{fragmentサイズ} = 1\text{fragment を埋めるのにかかる時間}$$

=読み出し、書き込みがブロックされている時間

となるので、初期設定では読み出し、書き込みがブロックされている時間は

$$\frac{1}{8000}\ [s\ /\ sound] \times 1\ [sound\ /\ bytes] \times 1024\ [bytes\ /\ fragment] = 128\ [ms\ /\ fragment]$$

となり、読み出し、書き込みがブロックされている時間は 128 ms である。つまり送信者と受信者で 128ms ずつ両方に遅延がある。/dev/dsp の設定の変更後、読み出し、書き込みがブロックされている時間は

$$\frac{1}{44100}\ [s\ /\ sound] \times \frac{1}{2}\ [sound\ /\ bytes] \times 256\ [bytes\ /\ fragment] = 2.9\ [ms\ /\ fragment]$$

となり読み出し、書き込みがブロックされている時間は 2.9 ms である。送受信者の遅延は 2.9 ms まで改善された。

これで遅延は改善された。

課題達成または問題克服の過程に行われた調査や実験などの内容と結果:
/dev/dsp の仕様と、fragment サイズの変更については文献[3]と文献[4]を参照した。図1.1と図1.2を見ると、Block size(fragment サイズのこと)が初期設定で 1024 bytes、変更後 256 bytes になっていることがわかる。

参照文献:
[1]:http://homepage3.nifty.com/rio_i/lab/oss/000prologue.htm, 2014/6/30 available.
[2]:http://homepage3.nifty.com/rio_i/lab/oss/003dsptest.htm, 2014/6/30 available.
[3]:http://homepage3.nifty.com/rio_i/lab/oss/005realtime.htm, 2014/6/30 available.
[4]:http://manuals.opensound.com/developer/SNDCTL_DSP_SETFRAGMENT.html, 2014/6/30 available.