パケットキャプチャのドキュメント

##　概要 scan.pyを用いてパケットキャプチャをした後のデータ処理について記載しておく。 以下は処理のフローである。 また、ファイルパスは基本的にsrcをベースに記載している。

1. cocoaパケットの抽出
2. 抽出するアドレスの調査
3. 時間による区分もしくはCSVへの変換
4. 初回受信時刻によるアドレス抽出

COCOAパケットの抽出

grepコマンドを使ってuuidがcocoaパケットの6ffdのものを抽出する。

```
grep -a  0000fd6f-0000-1000-8000-00805f9b34fb grepしたいファイル
```

また、grep.shにdata/originalの全てのテキストに対してgrepするように実装している。この際grepされたものはdata/capture/ver3/txtに保存される。

grepVer3.sh

```
#!/bin/bash

#COCOAパケットを抽出
LS=$(ls data/capture/ver3/original/)
for inputFileName in ${LS}
  do
    grep -a  0000fd6f-0000-1000-8000-00805f9b34fb
data/capture/ver3/original/$inputFileName >
data/capture/ver3/txt/$inputFileName
  done
```

抽出するアドレスの調査

該当するアドレスを抽出するためにまずは該当アドレスを探す。その際にdataAnalyzeパッケージのdataAnalyze.javaを用いる。

dataAnalyze.sh

```
#!/bin/bash


#該当すると思われるアドレスを抽出
```

```
LS=$(ls data/capture/ver3/txt/)
for inputFileName in ${LS}
  do
    java dataAnalyze/DataAnalyze data/capture/ver3/txt/$inputFileName 9 15
8 0 >data/result/analyze/ver3/$inputFileName
  done
```

dataAnalyze/DataAnalyze.java(一部抜粋)

```
/**
     * @param args 0に読み込むファイル名,1,2に最初のデータ範囲(秒),3にデータ範囲
(T),4にカットするアドレス下限
     * @throws IOException
     *
     */
    public static void main(String[] args) throws IOException {
        // TODO 自動生成されたメソッド・スタブ
        Read read = new ReadTXT(args[0]);
        DataAnalyze analyze = new DataAnalyze(read.read());
        analyze.makeAddressList();
        analyze.removeFewAddress(Integer.parseInt(args[4]));
        analyze.identify(Integer.parseInt(args[3]));

analyze.extract(Integer.parseInt(args[1]),Integer.parseInt(args[2]));
        analyze.print();


    }
```

スクリプトを実行するとdata/result/analyze/ver3に解析結果が表示される。以下に解析結果の例を示す.

aquos_pocket_1

```
6d:86:5e:0a:19:69,ftime =11.975712,ltime
=351.049071,aveRssi=-84,numPkt=202
```

aquos_walk_2

```
43:81:91:0e:0a:67,ftime =12.662547,ltime
=161.135771,aveRssi=-83,numPkt=124

0f:7a:05:ad:a2:e2,ftime =13.684816,ltime =123.921398,aveRssi=-96,numPkt=3
```

aquos_pocket_1には該当するアドレスが一つしかないのでこれで確定と思われる。aquos_walk_2は候補が複数あるが0f:7a:05:ad:a2:e2の方はパケット数があまりに少ないので何かしらのノイズと考えられ、43:81:91:0e:0a:67が抽出すべきデータとなる。

時間による区分もしくはCSVへの変換

時間による区分

複数ルートのデータを一気に取得した場合のみ、時間による区分を行う。時間による区分を行うにはまず区分する時間をdata/capture/ver3/split/splitTimeTable以下にキャプチャファイルと同じ名前のcsvファイルを用意する。そしてcsvファイルに区切る時刻を{開始時刻、終了時刻}の形式で記録する。

aquos_pocket_1.csv

```
15:27:38,15:27:56
15:28:08,15:28:20
15:28:32,15:28:42
15:28:58,15:29:09
15:29:35,15:29:59
15:30:56,15:31:20
```

その後、processed/timeAdjustment/Split.javaを用いて時間による区分を行う、このときの結果がdata/capture/ver3/csvに保管される。

Split.java

```
package processed.timeAdjustment;

import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import processed.ReadCSV;
import processed.ReadTXT;
import processed.extract.node.Packet;

public class Split {

    public static double parseTime(String str) {
        Pattern pTime = Pattern.compile("([0-9]{2}):([0-9]{2}):([0-9]
{2})");
        Matcher mTime =pTime.matcher(str);
        mTime.find();
        double hour = Double.parseDouble(mTime.group(1));
        double minute = Double.parseDouble(mTime.group(2));
        double second = Double.parseDouble(mTime.group(3));
```

```
        return hour*3600 + minute*60 +second;
    }
    public static void main(String args[]) throws IOException {
        ArrayList<Packet> packets = ReadTXT.read(args[0]);
        ArrayList<String[]> splitTimeTable =
ReadCSV.read(args[0].replace("txt/",
"split/splitTimeTable/").replace(".txt",".csv"));
        for(int i=0;i<splitTimeTable.size();i++) {
            FileWriter fileWriter = new FileWriter(args[0].replace("txt/",
"csv/").replace(".txt","_"+(i+1)+"_.csv"));
            double fTime = parseTime(splitTimeTable.get(i)[0]);
            double lTime = parseTime(splitTimeTable.get(i)[1]);
            fileWriter.append("address,time,rssi\n");
//          System.out.print("fTime="+fTime);
//          System.out.println("Time="+lTime);
            for(Packet packet:packets) {
                //System.out.print("time="+packet.getTime());
                if(fTime<=packet.getTime()&&packet.getTime()<=lTime) {
                    //System.out.println(" *");
                    fileWriter.append(packet.getAddress()+","+
(packet.getTime()-fTime)+","+packet.getRssi()+"\n");
                }else {
                    //System.out.println("");
                }

            }
            fileWriter.close();
        }

    }



}
```

また、data/capture/ver3/txt以下の全てのファイルに対してSplit.javaを行う以下のスクリプトもあるので活用するといい。

splitVer3.sh

```
LS=$(ls data/capture/ver3/original/)

LS=$(ls data/capture/ver3/txt/)
for inputFileName in ${LS}
  do
    java processed/split/Split data/capture/ver3/txt/$inputFileName
```

```
    done
```

CSVへの変換

純粋にcsvへの変換のみ行いたい場合にはprocessed/timeAdjustment/ConvertCSV.javaを実行すると良い。

ConvertCSV

```
package processed.timeAdjustment;

import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import processed.ReadTXT;
import processed.extract.node.Packet;

public class ConvertCSV {

    public static double parseTime(String str) {
        Pattern pTime = Pattern.compile("([0-9]{2}):([0-9]{2}):([0-9]
{2})");
        Matcher mTime = pTime.matcher(str);
        mTime.find();
        double hour = Double.parseDouble(mTime.group(1));
        double minute = Double.parseDouble(mTime.group(2));
        double second = Double.parseDouble(mTime.group(3));
        return hour * 3600 + minute * 60 + second;
    }

    public static void main(String args[]) throws IOException {
        ArrayList<Packet> packets = ReadTXT.read(args[0]);
        ArrayList<Packet> toGetFTime =
ReadTXT.read(args[0].replace("txt/", "original/"));

        double fTime = toGetFTime.get(0).getTime();

        FileWriter fileWriter = new FileWriter(
                args[0].replace("txt/", "csv/").replace(".txt", ".csv"));

        fileWriter.append("address,time,rssi\n");
        for (Packet packet : packets) {

                fileWriter
                        .append(packet.getAddress() + "," +
(packet.getTime() - fTime) + "," + packet.getRssi() + "\n");

        }
```

```
            fileWriter.close();

        }
    }
```

初回受信時刻によるアドレス抽出

先程の工程で抽出したアドレスのパケットを抽出する。そのためにまず選択したアドレスをcsv形式でdata/capture/ver3/selectAddress.csvにまとめる必要がある。この際、まだアドレスを確定できていない場合は空欄でも構わないが形式が崩れると後々のプログラムでエラーが出るので形式は2行目のものに統一すること。その後processed/selectAddress/SelectAddreess.javaを実行するとアドレスをCSV形式で抽出してくれる。

selectAddress.csv

```
fileName,address
aquos_pocket_1,6d:86:5e:0a:19:69
aquos_pocket_2,
aquos_walk_1,
aquos_walk_2,
g7_pocket_1,
g7_pocket_2,
g7_walk_1,
g7_walk_2,
g8_pocket_1,
g8_pocket_2,
g8_walk_1,
g8_walk_2,
```

SelectAddress.java

```
package processed.selectAddress;

import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

import processed.ReadCSV;
import processed.extract.node.Packet;

public class SelectAddress {
    String fileName;
    String address;
    ArrayList<Packet> packets;

    public SelectAddress(String fileName, String address) {
        super();
        this.fileName = fileName;
```

```
        this.address = address;
        packets = new ArrayList<>();
    }

    public static void main(String[] args) throws IOException {
        // TODO 自動生成されたメソッド・スタブ
         ArrayList<String[]> selectAddressTable =
ReadCSV.read("data/capture/ver3/selectAddress.csv");
         //ヘッダを削除
         selectAddressTable.remove(0);
         ArrayList<SelectAddress> selectAddresses = new ArrayList<>();

         for(String[] fileData:selectAddressTable) {
             String fileName = fileData[0];
             String address = fileData[1];
             if(fileName.equals("")&&address.equals("")) {
                 SelectAddress selectAddress = new
SelectAddress(fileName,address);
                 selectAddresses.add(selectAddress);
             }
         }

         for(SelectAddress selectAddress:selectAddresses) {
             selectAddress.selectAddress();
             selectAddress.writeTXT();
         }

    }

    private void writeTXT() throws IOException {
        // TODO 自動生成されたメソッド・スタブ
        FileWriter fileWriter = new
FileWriter("data/capture/ver3/masterPiece/"+fileName+".csv");
        fileWriter.append(",address,time,rssi");
        fileWriter.append("\r\n");
        for(Packet packet:packets) {

fileWriter.append(packet.getAddress()+","+packet.getTime()+","+packet.getR
ssi());
            fileWriter.append("\r\n");
        }
        fileWriter.close();


    }

    private void selectAddress() throws IOException {
        // TODO 自動生成されたメソッド・スタブ
        ArrayList<Packet> packets = new ArrayList<>();
        ArrayList<String[]> read =
ReadCSV.read("data/capture/ver3/csv/"+fileName+".csv");
        for(String[] st:read) {
            packets.add(new
Packet(st[0],Double.parseDouble(st[1]),Integer.parseInt(st[2])));
```

```
            }
            for(Packet packet:packets) {
                if(packet.getAddress().equals(address)) {
                    this.packets.add(packet);
                }
            }
        }

    }
```

なお出力先はdata/capture/ver3/masterPieceに出力される。