CS 380

Project 3

My repository for this class is under CS 380 – Computer Networks

<https://github.com/jarodNakamoto/College-CS-Courses.git>

https://github.com/genevieveleach/school-projects

Source Code Below:

import java.io.\*;

import java.net.Socket;

import java.math.BigInteger;

public class Ipv4Client {

public static void main(String[] args) throws Exception {

try {

Socket socket = new Socket("18.221.102.182", 38003);

System.out.println("Connected to server.");

InputStream is = socket.getInputStream();

InputStreamReader isr = new InputStreamReader(is, "UTF-8");

BufferedReader br = new BufferedReader(isr);

OutputStream os = socket.getOutputStream();

for(int i = 2; i <= Math.pow(2,12); i\*=2) {

System.out.println("Data length: " + i);

int size = 20+i;

byte[] header = new byte[size];

//version: implement

int version = 4;

//HLen: implement

int hLen = 5;

int merged = shiftAndMerge(version,hLen,4);

header[0] = (new Integer(merged)).byteValue();

//TOS: do not implement

int tos = 0;

header[1] = (new Integer(tos)).byteValue();

//length: implement

int totalLength = 20 + i;

splitAndAddToByteArr(totalLength, 2, header, 2);

//ident: do not implement

int ident = 0;

splitAndAddToByteArr(ident, 2, header, 4);

//flags: implement assuming no fragmentation

//String flag = "010";

int flag = 2;

//offset: do not implement

int offset = 0;

merged = shiftAndMerge(flag,offset,13);

splitAndAddToByteArr(merged, 2, header, 6);

//TTL: implement assuming every packet has a TTL of 50

int ttl = 50;

header[8] = (new Integer(ttl)).byteValue();

//protocol: implement assuming TCP for all packets

//TCP is six

int tcp = 6;

header[9] = (new Integer(tcp)).byteValue();

//checksum: implement

header[10] = 0;

header[11] = 0;

//sourceaddr: implement using IP address of choice

//134.71.249.45

String sourceAddr = "10000110010001111111100100101101";

int srcAddr = (new BigInteger(sourceAddr, 2)).intValue();

splitAndAddToByteArr(srcAddr, 4, header, 12);

//destaddr: implement using IP address of server

//18.221.102.182

String destAddr = "00010010110111010110011010110110";

int dstAddr = Integer.parseInt(destAddr, 2);

splitAndAddToByteArr(dstAddr, 4, header, 16);

//options/pad: ignore, dont even put in header

//add real checksum to header

int chksum = (int)(checksum(header));

splitAndAddToByteArr(chksum, 2, header, 10);

//data: implement using 0's or random data

int data = 0;

for(int j = 0; j < header.length; j++){

os.write(header[j]);

//System.out.println(String.format("0x%02X", header[j]));

}

String response = br.readLine();

System.out.println(response);

if(!response.equals("good")) {

break;

}

System.out.println();

}

} catch (IOException e) {

e.printStackTrace();

}

}

private static short checksum(byte[] b) {

//if the array length is odd

if((b.length % 2) != 0) {

byte[] bOdd = new byte[b.length+1];

System.arraycopy(b, 0, bOdd, 0, b.length);

bOdd[bOdd.length-1] = 0;

b = bOdd;

}

int sum = 0;

for (int i = 0; (i + 1) < b.length; i += 2) {

int first = b[i];

if (first < 0) {

first ^= 0xFFFFFF00;

}

int second = b[i+1];

if (second < 0) {

second^= 0xFFFFFF00;

}

first <<= 8;

sum += (first ^ second);

// overflow detection

if ((sum & 0xFFFF0000) != 0) {

/\*carry occurred, so wrap around \*/

sum &= 0xFFFF;

sum++;

}

}

return (short)(~(sum & 0xFFFF));

}

private static int shiftAndMerge(int s1, int s2, int shiftAmount){

s1 = s1 << shiftAmount;

int thingy = s1 ^ s2;

//System.out.println("thingy: " + String.format("0x%04X", thingy));

return thingy;

}

private static void splitAndAddToByteArr(int split, int numSplits, byte[] b, int index){

for(int i = 1; i <= numSplits; i++){

if(numSplits + index -i >= b.length)

return;

b[numSplits + index - i] = new Integer(split).byteValue();

split = split >> 8;

}

}

}