CS 380

Project 5

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

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

Source Code Below:

import java.io.\*;

import java.net.Socket;

import java.math.BigInteger;

import java.io.\*;

import java.net.Socket;

import java.util.Random;

import java.math.BigInteger;

public class UdpClient {

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

try {

Socket socket = new Socket("18.221.102.182", 38005);

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

InputStream is = socket.getInputStream();

OutputStream os = socket.getOutputStream();

int minsize = 20;

int numPackets = 12;

byte[] data;

//handshake

byte[] packet = new byte[minsize+4];

//use UDP (17)

setUpIpv4Header(packet, 17, 4);

splitAndAddToByteArr(0xDEADBEEF, 4, packet, 20);

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

os.write(packet[j]);

}

byte[] serverResponse = new byte[4];

for(int i = 0; i < serverResponse.length; i++){

serverResponse[i] = (byte)(is.read());

}

String response = serverByteResponseToString(serverResponse);

System.out.println("Handshake response: " + response);

if(!response.equals("0xCAFEBABE")) {

return;

}

int portNumberHalf = is.read();

int portNumberHalf2 = is.read();

int portNumber = 0;

portNumber = portNumberHalf << 8;

portNumber = portNumber ^ portNumberHalf2;

System.out.println("Port Number Received: " + portNumber);

System.out.println();

long rttAvg = 0;

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

System.out.println("Sending packet with " + i + " bytes of data");

int size = minsize+i+8;

packet = new byte[size];

data = new byte[i];

fillWithRandomData(data);

//UDP is 17

fillIPv4Packet(packet,data, portNumber);

//send packet to server

os.write(packet);

long timeSent = System.currentTimeMillis();

//process server response

is.read(serverResponse);

long timeReceived = System.currentTimeMillis();

long rtt = timeReceived - timeSent;

response = serverByteResponseToString(serverResponse);

System.out.println(response);

System.out.println("RTT: " + rtt +"ms");

rttAvg += rtt;

if(!response.equals("0xCAFEBABE")) {

break;

}

System.out.println();

}

System.out.println("Average RTT: " + (rttAvg \*1.0/numPackets) + "ms\n");

} catch (IOException e) {

e.printStackTrace();

}

}

private static void fillWithRandomData(byte[] data){

Random dataGen = new Random();

dataGen.nextBytes(data);

}

private static void fillIPv4Packet(byte[] packet, byte[] data, int destPort){

int udpLength = data.length+8;

//UDP is protocol 17

int protocol = 17;

setUpIpv4Header(packet, protocol, udpLength);

//IPv4 data: implement using 8 byte UDP header

int startOfData = 28;

//using udp protocol

//pretend source port is 0

//takes 2 bytes in array

//destination port is placed after source port

splitAndAddToByteArr(destPort, 2, packet, 22);

//length of UDP header and data

splitAndAddToByteArr(udpLength, 2, packet, 24);

//checksum of psuedoheader //happens later

//copy data into packet

System.arraycopy(data, 0, packet, startOfData, data.length);

//udp: do checksum on header, data, psuedoheader "includes Ipv4 header"

byte[] psuedoHeader = new byte[20 + data.length];

//copy source and destination addresses into psuedoHeader

System.arraycopy(packet, 12, psuedoHeader, 0, 8);

//let psuedoHeader know its using UDP

psuedoHeader[9] = (byte)protocol;

//add udp length into psuedoHeader

splitAndAddToByteArr(udpLength, 2, psuedoHeader, 10);

//copy port numbers and length into psuedoHeader

System.arraycopy(packet, 20, psuedoHeader, 12, 6);

//copy data into psuedoHeader

System.arraycopy(data, 0, psuedoHeader, 20, data.length);

//add UDP checksum to packet

int chksum = (int)(checksum(psuedoHeader, psuedoHeader.length));

splitAndAddToByteArr(chksum, 2, packet, 26);

}

private static void setUpIpv4Header(byte[] packet, int protocol, int dataLength){

//version: implement

int version = 4;

//HLen: implement

int hLen = 5;

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

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

//TOS: do not implement

int tos = 0;

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

//length: implement

int totalLength = 20 + dataLength;

splitAndAddToByteArr(totalLength, 2, packet, 2);

//ident: do not implement

int ident = 0;

splitAndAddToByteArr(ident, 2, packet, 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, packet, 6);

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

int ttl = 50;

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

//protocol: implement

packet[9] = (new Integer(protocol)).byteValue();

//checksum: implement

packet[10] = 0;

packet[11] = 0;

//sourceaddr: implement using IP address of choice

//192.168.56.1

String sourceAddr = "11000000101010000011100000000001";

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

splitAndAddToByteArr(srcAddr, 4, packet, 12);

//destaddr: implement using IP address of server

//18.221.102.182

String destAddr = "00010010110111010110011010110110";

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

splitAndAddToByteArr(dstAddr, 4, packet, 16);

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

//add real checksum on header to packet

int chksum = (int)(checksum(packet, 20));

splitAndAddToByteArr(chksum, 2, packet, 10);

}

private static String serverByteResponseToString(byte[] serverResponse){

String response = "0x";

for(int k = 0; k < serverResponse.length; k++){

response += String.format("%02X", serverResponse[k]);

}

return response;

}

private static short checksum(byte[] b, int length) {

//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) < 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;

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;

}

}

}