# Министерство науки и высшего образования Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра теоретической и прикладной математики

### Лабораторная работа №3 по дисциплине « Разработка web-приложений и распределенных информационных систем»

|  |  |  |
| --- | --- | --- |
| эмблема_светлая.png |  |  |
| Факультет: | ПМИ |
| Группа: ПМИ-62 | |
| Студенты: | Мамонова Е.В., Ершов П.К., Цыденов З. Б., Блинов А. А., Смирнов А. В., Ковалёв Н. Д. |
| Преподаватель: | Цыгулин А. А. |
|  | | |

Новосибирск

2019

1. **Задание**

На основе программы из второй лабораторной разработать программное средство, способное найти файл, расположенный на другом узле сети.

1. **Анализ задачи**

**Алгоритм программы**

Объявить сокет.

Связать сокет с портом.

Запросить имя пользователя.

Запросить имя файла.

Получить хэш от имени файла.

Послать полученный хэш всем соседям.

Если пришло сообщение с запросом на поиск файла

Ищем хэш в собственном списке хэшей от файлов.

Если нашил у себя

Отсылаем обратно хэш от содержимого файла

Иначе

Ищем в массиве хэшей от соседей

Если нашли

Отсылаем узлу-клиенту IP адрес откуда пришёл искомый хэш.

Если не нашли искомый хэш

Отсылаем сообщение об отсутствии файла и ссылок на файл

Если пришло сообщение с IP адресом ссылки на файл

Отсылаем запрос на поиск файла на полученный адрес.

Если пришло сообщение с хэшем от содержимого файла

Отсылаем обратно сообщение-запрос на получение файла с полученным хэшем

Если пришло сообщение с запросом на файл

Ищем имя файла по полученному хэшу от содержимого

Отсылаем узлу, который запросил файл содержимое файла

Если пришло содержимое файла

Создаём файл с именем искомого

Записываем полученное в сообщении содержимое в созданный файл

1. **Структура данных приложения**

Вся программа хранится в файле web\_lab3.js

В папке files хранятся файлы с информационными карточками. Названия файлов состоят из имени и фамилии на английском.

Пример имени файла (программа учитывает расширение файла): Elizabet Mamonova.txt

Пример информационной карточки:

Name: Lisa

Address: Novosibirsc

Number: 8-800-555-3535

Для запуска программы необходимо ввести с консоли: **node web\_lab3.js**

Программа потребует ввести имя пользователя и имя файла, который требуется найти.

Дальнейшие действия будут происходит автоматически.

1. **Текст программы**

**web\_lab3.js**

var dgram = require('dgram'); //Модуль dgram обеспечивает реализацию сокетов UDP датаграмм

const fs = require("fs");//модуль для работы с файлом

//const vr = require('./vars');

var dgramSocket = dgram.createSocket("udp4"); //Для UDP-сокетов заставляет dgram.Socket прослушивать сообщения дейтаграммы на именованном порту

var crypto = require('crypto'); // crypto - модуль шифрования. Подключаю его для создания хеша

var readline = require('readline');// Предоставляет интерфейс для чтения данных из читаемого потока по одной строке за раз

var msg\_size = 1; // Количество байт под тип сообщения

var msg\_length\_size = 2; // Количество байт под длину сообщения

var interval = 2000; // Каждые 2 секунды запрос

var port = 1024; // Номер порта

var readline = readline.createInterface(process.stdin, process.stdout, null); // Для считывания текста с консоли

var onlineUser = []; // Массив со всеми пользователями онлайн

var allHesh = []; // Массив со всеми хэшами от файла

var alladdr = []; // Массив со всеми адресами

var allId = []; // Массив с id (хэшами) от соседей

var n\_hesh = [];//Массив с хэшами от файлов от соседей

var h\_ip = [];//Массив с IP для хэшей от файлов от соседей

var us\_h = [];//массив (да, ещё один) для имён соседей

var userName; // имя пользователя

var userId;// хэш пользователя

var flag = false; // Флаг дял онлайн\оффлайн пользователя

var ip\_addr = '255.255.255.255'; // Адрес, с которого отправляется запрос

var hesh\_file = '';//хэш от содержимого файла

var home\_ip = '';//ip пользователя

//ТИПЫ СООБЩЕНИЙ

var request = 0; // Запрос

var response = 1; // Ответ

var mess = 2; // Сообщение с хэшем

var ques = 3; //сообщение с запросом на поиск файла, если узел хранит файл, то отправить сообщение типа f\_hash по адресу отправки ques

var hash\_ip = 4; //ответ на сообщение типа ques с IP адресом отправленного в ques хэша

var no\_hash = 5;//отрицательный ответ на сообщение типа ques с IP адресами двух ближайших узлов сети, где может быть искомый хэш

var f\_hash = 6;//ответ на запрос типа ques, с хэшем от содержимого файла (В структуре типа HESH обозначен как H)

var take\_f = 7;//запрос на получение файла по его хэшу

var ff = 8;//ответ на take\_f с содержимым файла

//КУЧА ФЛАГОВ (ПРОСТИТЕ МЕНЯ ЗА ЭТО)

var fl\_q = false;//флаг, сообщающий о приходе сообщения

var fl\_IP = true;//флаг, сообщающий о приходе IP адреса

var fl\_H = false;//флаг, сообщающий о приходе хэша от содержимого файла

var fl\_TH = false;//флаг, сообщающий о приходе хэша от содержимого файла в сообщении-запросе на выдачу содержимого файла

var fl\_f = false;//флаг, сообщающий о приходе содержимого файла (сам файл, по сути)

var q\_hesh; // хэш от имени файла, который пришёл от узла, который ищет файл

var q\_IP; //IP адрес q\_hesh (нужен для обратного ответа)

var H\_IP;//IP адрес q\_hesh (ВВЕДЁН, ЧТОБЫ НЕ ЗАПУТАЛАСЬ ПРОГРАММА, ТАК КАК ПОКА МЫ ИЩЕМ ОДИН ФАЙЛ, МОЖЕТ ПРИЙТИ ЗАПРОС НА ДРУГОЙ)

var TH\_IP;//IP адрес q\_hesh (ВВЕДЁН, ЧТОБЫ НЕ ЗАПУТАЛАСЬ ПРОГРАММА, ТАК КАК ПОКА МЫ ИЩЕМ ОДИН ФАЙЛ, МОЖЕТ ПРИЙТИ ЗАПРОС НА ДРУГОЙ)

var anyIP = []; //массив IP адресов, которые пришли от улза, у которого нет никакой информации об искомом файле (ПО ХОРШЕМУ ЭТО ДОЛЖНО БЫТЬ, НО Я ЛЕНИВЫЙ ЗАСРАНЕЦ)

var IP; //IP адрес узла, с которо пришёл хэш от искомого файла (имеется любой хэш от имени файла)

var H; //хэш от содержимого искомого файла (нужет, чтобы послать запрос на получение содержимого файла по хэшу)

var TH; //хэш от содержимого искомого файла, которйы пришёл в сообщении-запросе на выдачу файла

var COM\_F; //содержимое файла (будет помещено в созданный файл)

var HESH\_M = [];//массив структур типа HESH, нужен, чтобы поставить в соответствие хэшам от имени файла (от слов в имени, от имени целиком, от имени с расширением файла) хэш от содержимого этого файла

var name\_of\_file;

var nh;

function HESH(h\_n, H, file) {

this.h\_n = h\_n;//массив хэшей от имён файла (от слов в имени, от имени целиком, от имени с расширением файла)

this.H = H;//хэш от содержимого файла

this.file = file;//полное имя файла с расширением (необходимо, чтобы вернуть содержимое файла узлу, который файл ищет)

}

// Определяем ip адрес компьютера. К Первым 3-м цифрам ip добавляем 255 (широковещательный)

require('dns').lookup(require('os').hostname(), function (err, add, fam) {

for (var i = 0; i < add.length; i++)

home\_ip += add[i];

})

// Сокет начал прослушивание

dgramSocket.on('listening', function () {

var ServerAddress = dgramSocket.address();

console.log('Listening ' + ServerAddress.address + ":" + ServerAddress.port);

});

// Сокет создан

dgramSocket.bind(port, function () {

dgramSocket.setBroadcast(true);

console.log('Broadcast on.');

readline.question('Your name: ', (name) => {

readline.question('Name of file: ', (info) => {

// Заношу информацию об имени и закрываю входной поток

userName = name;

readline.close();

readline.close();

let p = 0;

let f\_n = [];

fs.readdirSync('./files/').forEach(file => {

if (file != 'data.txt')

f\_n.push(file);

});

p = 0;

while (p < f\_n.length) {

let fr = fs.readFileSync('./files/' + f\_n[p], 'utf8');

let sb = f\_n[p].split('.')[0].split(' ');

//console.log( sb);

let p1 = 0

let h = []

while (p1 < sb.length) {

h.push(crypto.createHash('md5').update(sb[p1]).digest())

p1++;

}

if (find\_hesh(h, crypto.createHash('md5').update(f\_n[p].split('.')[0]).digest())[0] == false)

h.push(crypto.createHash('md5').update(f\_n[p].split('.')[0]).digest());

h.push(crypto.createHash('md5').update(f\_n[p]).digest());

let HH = new HESH(h, crypto.createHash('md5').update(fr).digest(), f\_n[p]);

HESH\_M.push(HH);

p++;

}

p = 0;

// console.log(HESH\_M);

//\*

while (p < HESH\_M.length) {

let u = 0;

while (u < HESH\_M[p].h\_n.length) {

allHesh.push(HESH\_M[p].h\_n[u]);

u++;

}

allHesh.push(HESH\_M[p].H);

p++;

}

//\*/

//А ТУТ МЫ ШЛЁМ ВСЕМ СОСЕДЯМ ЗАПРОС НА ПОИСК ФАЙЛА

name\_of\_file = info;

//ВНИМАНИЕ! ТЕСТОВЫЙ ПРОГОН НА ОДНОМ КОМПЬЮТЕРЕ. ДЛЯ ПЕРЕКЛЮЧЕНИЯ В СЕТЕВОЙ РЕЖИМ ЗАКОММЕНТИРУЙТЕЙ 3 СТРОКИ НИЖЕ

//name\_of\_file = 'test';

// nh = crypto.createHash('md5').update(name\_of\_file).digest();

// u\_send(nh, home\_ip, ques);

//\*

userId = crypto.createHash('md5').update(home\_ip).digest(); // id (присваиваю хеш md5)

nh = crypto.createHash('md5').update(name\_of\_file).digest();

let i = 0;

while (i < alladdr.length) {

u\_send(nh, alladdr[i], ques);

i++;

}

//\*/

loopFunction(); // Предварительно вызываем, чтобы не ждать интервального времени перед первым вызовом

setInterval(loopFunction, interval); // Задаём периодичность выполнения для функции

});

});

});

// ищет минимум в массиве и возвращает его индекс.

function arrayMin(arr) {

var len = arr.length, min = Infinity;

var cr;

while (len--) {

if (arr[len] < min) {

min = arr[len];

cr = len;

}

}

return cr;

};

//провит операцию XOR с битами хэша.

function metric(hash1, hash2) {

let ret = Buffer.from(hash1)

for (let i = 0; i < hash1.length; i++)

ret[i] ^= hash2[i]

return ret

}

function find(arr, elem) {

var len = arr.length;

var fl = false;

while (len--) {

if (arr[len] == elem) {

fl = true;

break;

}

}

return fl;

};

//Проверяет есть ли элемент в массиве возвращая 1 - да и 0 - нет.

function find\_hesh(arr, elem) {

var len = arr.length;

var fl = [];

fl[0] = false;

while (len--) {

var bf = arr[len];

if (roughScale(bf.toString("hex"), 16).toString() == roughScale(elem.toString("hex"), 16).toString()) {

fl[0] = true;

fl[1] = len;

break;

}

}

return fl;

};

//переводит хэш в 10-е значение (для поиска минимума).

function roughScale(x, base) {

var parsed = parseInt(x, base);

if (isNaN(parsed)) { return 0 }

return parsed;

}

// Сокет получил сообщение.

dgramSocket.on('message', function (msg, rinfo) {

//Если не введено имя.

if (userName === undefined)

return

var msgType = msg[0]; // Определяем тип сообщения.

// Для запроса. Формируем дейтаграмму.

if (msgType === request) {

var rmsg = Buffer.allocUnsafe(msg\_size + msg\_length\_size + Buffer.byteLength(userName) + userId.length); // Собираем сообщение, чтоб ответить.

rmsg[0] = response;

rmsg.fill(userId, msg\_size, msg\_size + userId.length)

rmsg.writeUInt16BE(Buffer.byteLength(userName), msg\_size + userId.length);

rmsg.fill(userName, msg\_size + userId.length + msg\_length\_size); // + name.

dgramSocket.send(rmsg, 0, rmsg.length, rinfo.port, rinfo.address);

}

//Для расшифровки.

if (msgType === response) {

var msgLength; //длина сообщения.

msgLength = msg.readUInt16BE(msg\_size + userId.length);

var msgText = msg.toString('utf-8', msg\_size + userId.length + msg\_length\_size); // Извлекаем текст сообщения.

var id = Buffer.from(msg.slice(msg\_size, msg\_size + userId.length));

var msg\_addr = rinfo.address;

var msg\_port = rinfo.port;

var flag = 0;

if (msg\_addr != home\_ip) {

// заносим ip в массив (кроме домашнего, для проверки работоспособности убрать второре условие).

if (find(alladdr, msg\_addr) == false) {

alladdr.push(msg\_addr);

//console.log(msg\_addr);

}

// заносим id в массив (кроме домашнего, для проверки работоспособности убрать второре условие).

if (find\_hesh(allId, id)[0] == false)

allId.push(id);

}

onlineUser.push(msgText + " " + id.toString('hex') + ": " + msg\_addr); // Добавляем онлайн пользователя в массив.

}

//Если в сообщении есть хэш от файла.

if (msgType === mess) {

var msgLength; //длина сообщения.

msgLength = msg.readUInt16BE(msg\_size + userId.length);

var msgText = msg.toString('utf-8', msg\_size + userId.length + msg\_length\_size); // Извлекаем текст сообщения

var id = Buffer.from(msg.slice(msg\_size, msg\_size + userId.length));

//console.log(msg);

var msg\_addr = rinfo.address;

var flag = 0;

var bf = id;

//console.log(bf);

if (msg\_addr != '127.0.0.1' && find(us\_h, msgText) == false)

us\_h.push(msgText);// добавляетм имя соседа, пославшего хэш в массив имён

if (msg\_addr != '127.0.0.1' && find(h\_ip, msg\_addr.toString()) == false)

h\_ip.push(msg\_addr.toString());// добавляем адрес соседа, пославшего хэш, в массив

if (msg\_addr != '127.0.0.1' && find(n\_hesh, bf) == false)

n\_hesh.push(bf); // Добавляем хэш от соседа в массив.

}

//получен запрос на поиск файла, полученный хэш будет занесён в ques\_hesh, для поска по всех хэшам (как своим, из массива HESH\_M, так и по чужим, из массива n\_hesh

if (msgType === ques) {

var msgLength; //длина сообщения.

msgLength = msg.readUInt16BE(msg\_size + userId.length);

var msgText = msg.toString('utf-8', msg\_size + userId.length + msg\_length\_size); // Извлекаем текст сообщения

q\_hesh = Buffer.from(msg.slice(msg\_size, msg\_size + userId.length));

q\_IP = rinfo.address;

// console.log(q\_hesh);

fl\_q = true;

}

//пришёл ответ с IP адресом местонахожения файла

if (msgType === hash\_ip) {

var msgLength; //длина сообщения.

msgLength = msg.readUInt16BE(msg\_size + userId.length);

var msgText = msg.toString('utf-8', msg\_size + userId.length + msg\_length\_size); // Извлекаем текст сообщения

IP = Buffer.from(msg.slice(msg\_size, msg\_size + msg.length - 1 - msg\_length\_size));

//console.log(IP.toString());

IP = IP.toString().split(' ')[0];

//console.log(IP);

fl\_IP == true;

}

//если файла на узле нет и нет хэша от его имени, то такой узел пошлёт в ответ IP адреса ближайших двух узлов, где могут быть хэши от имени файла

if (msgType == no\_hash) {

var msgLength; //длина сообщения.

msgLength = msg.readUInt16BE(msg\_size + userId.length);

var msgText = msg.toString('utf-8', msg\_size + userId.length + msg\_length\_size); // Извлекаем текст сообщения

var id = Buffer.from(msg.slice(msg\_size, msg\_size + msg.length - 1 - msg\_length\_size));

anyIP = id.toString();

//console.log(anyIP);

}

//если файл на узле есть, то он отправит узлу, который послал запрос хэш от содержимого файла (в стурктуре HESH хранится в переменной H)

if (msgType === f\_hash) {

var msgLength; //длина сообщения.

msgLength = msg.readUInt16BE(msg\_size + userId.length);

var msgText = msg.toString('utf-8', msg\_size + userId.length + msg\_length\_size); // Извлекаем текст сообщения

H = Buffer.from(msg.slice(msg\_size, msg\_size + userId.length));

fl\_H = true;

H\_IP = rinfo.address;

//console.log(H);

}

//сообщение с хэшем от содержимого файла с запросом на отправку содержимого файла

if (msgType === take\_f) {

var msgLength; //длина сообщения.

msgLength = msg.readUInt16BE(msg\_size + userId.length);

var msgText = msg.toString('utf-8', msg\_size + userId.length + msg\_length\_size); // Извлекаем текст сообщения

TH = Buffer.from(msg.slice(msg\_size, msg\_size + userId.length));

TH\_IP = rinfo.address;

fl\_TH = true;

}

//сообщение с содержимым файла

if (msgType === ff) {

var msgLength; //длина сообщения.

msgLength = msg.readUInt16BE(msg\_size + userId.length);

var msgText = msg.toString('utf-8', msg\_size + userId.length + msg\_length\_size); // Извлекаем текст сообщения

var id = Buffer.from(msg.slice(msg\_size, msg\_size + msg.length - 1 - msg\_length\_size));

COM\_F = id;

fl\_f = true;

}

});

function loopFunction() {

// Если не подключились, пропускаем этап вывода на экран информации об online пользователях.

// Подключение в данном случае - первый вызов этой функции.

//ЧТО ТУТ ПРОИСХОДИТ, ЕСЛИ В КРАТЦЕ

//ПРОВЕРЯЕМ, ПРИШЁЛ ЛИ НАМ АДРЕС ГДЕ ЛЕЖИТ САМ ФАЙЛ

//ВНИМАНИЕ! ТЕСТОВЫЙ ПРОГОН НА ОДНОМ КОМПЬЮТЕРЕ. ДЛЯ ПЕРЕКЛЮЧЕНИЯ В СЕТЕВОЙ РЕЖИМ ЗАМЕНИТЕ home\_ip НА q\_IP

if (fl\_IP == false) {//НЕТ НЕ ПРИШЁЛ, ТОГДА ПРОВЕРИМ, ПРИСЛАЛИ НАМ ЗВПРОС НАПОИСК ФАЙЛА

if (fl\_q == true) {//ДА, ПРИСЛАЛИ

let i = 0;

let f = [];

let t = false;

while (i < HESH\_M.length) {//СНАЧАЛА ИЩЕМ В СВОИХ ХЭШАХ ИСКОМЫЙ, ВДРУГ У НАС ЕСТЬ НУЖНЫЙ ФАЙЛ

let bh = HESH\_M[i].h\_n;

if (find\_hesh(bh, q\_hesh)[0] == true) {

u\_send(HESH\_M[i].H, q\_IP, f\_hash);//q\_IP - стандартный адрес, home\_ip - тестовый

t = true;

break;

}

i++;

}

if (t == false) {//НЕ НАШЛИ У СЕБЯ ФАЙЛ, ИЩЕМ У СОСЕДЕЙ

f = find\_hesh(n\_hesh, q\_hesh)

if (f[0] == true) {//НАШЛИ СОСЕДСКИЙ ХЭШ ОТ ИМЕНИ ФАЙЛА, СОПОСТАВИМ ЕГО С АДРЕСОМ С МАССИВЕ И ПОШЛЁМ НАЙДЕННЫЙ АДРЕС УЗЛУ, КОТОРЫЙ ИЩЕТ ФАЙЛ

u\_send(h\_ip[f[1]] + ' IP', q\_IP, hash\_ip);////q\_IP - стандартный адрес, home\_ip - тестовый

}

else {//НЕТ У НАС ИСКОМОГО ХЭША, ПОШЛЁМ В ОТВЕТ СОБЩЕНИЕ, ЧТО У НАС НЕТ НИ ФАЙЛА, НИ ССЫЛКИ НА НЕГО

u\_send('I Have No Mouth, and I Must Scream', q\_IP, no\_hash);////q\_IP - стандартный адрес, home\_ip - тестовый

//send\_test(home\_ip, no\_hash);

//console.log('Пришёл IP');

}

}

fl\_q = false;

}

}

else {//А ТУТ НАМ ПРИШЁЛ АДРЕСС С КОТОРОГО ПРИШЛА ССЫЛКА НА ФАЙЛ, ПЕРЕШЛЁМ НАШ ЗАПРОС ЕМУ

u\_send(nh, IP, ques);

fl\_IP = false;

// console.log('Пошёл IP');

}

//ПРИШЁЛ НУЖНЫЙ НАМ ХЭШ ОТ ОСДЕРЖИМОГО ФАЙЛА (ДЕЛО В ТОМ, ЧТО РАБОТАЕТ У НАС ПОЛУЧЕНИЕ САМОГО ФАЙЛА (ЧИТАЙ ЕГО СОДЕРЖИМОГО) ТОЛЬКО ХЭШУ ОТ СОДЕРЖИМОГО. ПО ЭТО СНАЧАЛА ШЛЁМ УЗЛУ

//КОТОРЫЙ ИЩЕТ ФАЙЛ ХЭШ ОТ СОДЕРЖИМОГО ФАЙЛА, УЗЕЛ ШЛЁТ ОБРАТНО ЭТО ХЭШ, НО В ДРУГОМ ТИПЕ СООБЩЕНИЯ, И ТОЛЬКО ПОЛУЧИВ НУЖНЫЙ ХЭШ ОТ СОДЕРЖИМОГО, ВЫСЫЛАЕТСЯ САМО СОДЕРЖИМОЕ

if (fl\_H == true) {

u\_send(H, H\_IP, take\_f);//ОТСЫЛАЕМ ЗАПРОС НА СОДЕРЖИМОЕ ФАЙЛА

console.log('Пошёл хэш');

fl\_H = false;

}

//ПРИШЁЛ ХЭШ ОТ СОДЕРЖИМОГО В СООБЩЕНИИ С ЗАПРОСОМ НА СОДЕРЖИМОЕ

if (fl\_TH == true) {

let buf = [];

let i = 0;

while (i < HESH\_M.length) {

buf.push(HESH\_M[i].H);

i++;

}

//console.log(buf);

if (find\_hesh(buf, TH)[0] == true) {//НАШЛИ НУЖНЫЙ НАМ ХЭШ, БЕРЁМ НАЗВАНИЕ ФАЙЛА И ЧИТАЕМ СОДЕРЖИМОЕ В ПЕРЕМЕННУЮ

var fl = fs.readFileSync('./files/' + HESH\_M[find\_hesh(buf, TH)[1]].file, 'utf8');

}

//console.log(fl);

u\_send(fl, TH\_IP, ff);//ОТСЫЛАЕМ СОДЕЖИМОЕ

fl\_TH = false;

}

//А ВОТ И СОЕДЕРЖИМОЕ ФАЙЛА ПРИШЛО, СДЕЛАТЬ ИЗ НЕГО ФАЙЛ НУЖНО

//\*

if (fl\_f == true) {

name\_of\_file.split('.txt');

fs.openSync(name\_of\_file + '.txt', 'w');//СОЗДАЁМ ПУСТОЙ ФАЙЛ

fs.writeFile(name\_of\_file + '.txt', COM\_F, function (error) {//ПИХАЕМ СОДЕРЖИМОЕ В ФАЙЛ

if (error) throw error;

});

fl\_f = false;

}

//\*/

//ТЕСТОВЫЙ ПРОГОН СОДЕРЖИМОЕ ЛЕЖИТ В data.txt

/\*

if (fl\_f == true) {

// name\_of\_file.split('.txt');

fs.openSync('./files/data.txt', 'w');//СОЗДАЁМ ПУСТОЙ ФАЙЛ

fs.writeFile('./files/data.txt', COM\_F, function (error) {//ПИХАЕМ СОДЕРЖИМОЕ В ФАЙЛ

if (error) throw error;

});

fl\_f = false;

}

//\*/

//let fl = fs.readFileSync('./files/' + HESH\_M[1].file, 'utf8');

//console.log(fl);

//\*

if (flag === true) {

// Выводим информацию об online пользователях, которую сформировали с предыдущей рассылки.

console.log("Online:");

// Очищаем массив с информацией от соседей.

while (onlineUser.length)

console.log(onlineUser.pop());

console.log("\n");

console.log("Hesh:");

// Очищаем массив с хэшами от файлов от соседей.

while (n\_hesh.length) {

//fs.appendFileSync("./files/data.txt", "||" + n\_hesh[n\_hesh.length - 1] + "|| \n", "UTF-8");

console.log(n\_hesh.pop());

}

}

else {

console.log("Connection...");

flag = true;

}

//\*/

// console.log("Request...");

checkOnline(); // Запрос на проверку онлайн-пользователей.

//весь бред ниже нужен для поиска адреса, на который нужно отослать хэш от файла.

//сначала берём один из хэшей от файла.

//затем, с помощю массива с хэшами от соседей (id) и хэша от файла создаём новый массив через опреацию XOR.

//полученный массив переводим в 10 систему.

//в 10-м массиве ищем индес минимального элемента - это индекс в массиве со всеми ip соседей.

//заносим найденный индекс в массив-карту, который нужен для отправки хэшей по адресам.

//в массиве-карте номер элемента массива это номер хэша от файла, а число в элементе массива это индекс в массиве ip.

//u\_send('home\_ip', home\_ip, hash\_ip);

//ТЕСТОВЫЙ ПРОГОС ОТПРАВКИ СООБЩЕНИЙ С ХЭШАМИ ОТ ИМЕНИ ФАЙЛА

for (let r = 0; r < allHesh.length; r++)

u\_send(allHesh[r], home\_ip, mess);

//console.log(home\_ip.toString());

if (alladdr.length != 0) {

var i = 0;

var send\_map = []; // Массив со всеми номерами элементов allHesh, alladdr, allport для сборки сообщений с хэшами.

send\_map.length = allHesh.length;

while (i < allHesh.length) {

var buffID = [];

var j = 0;

while (j < allId.length) {

buffID[j] = metric(allId[j], allHesh[i]);

j++;

}

var find\_buff = [];

find\_buff.length = allId.length;

j = 0;

while (j < allId.length) {

find\_buff[j] = roughScale(buffID[j].toString("hex"), 16).toString();

j++;

}

//console.log(buffID);

send\_map[i] = arrayMin(find\_buff);

i++;

}

//console.log(send\_map);

for (var k = 0; k < send\_map.length; k++) {

send\_hesh(home\_ip, k); // хэш под номером k отправляем по адресу под номеров send\_map[k]

alladdr.splice(send\_map[k], send\_map[k]);

if (allId.length > 1)

allId.splice(send\_map[k], send\_map[k]);

}

// console.log("\n");

}

}

function send\_hesh(ip, index) {

var rmsg = Buffer.allocUnsafe(msg\_size + msg\_length\_size + Buffer.byteLength(userName) + allHesh[index].length); // Собираем сообщение, чтоб ппослать хэш

rmsg[0] = mess;

rmsg.fill(allHesh[index], msg\_size, msg\_size + allHesh[index].length)

rmsg.writeUInt16BE(Buffer.byteLength(allHesh[index]), msg\_size + allHesh[index].length);

rmsg.fill(userName, msg\_size + userId.length + msg\_length\_size); // + name

dgramSocket.send(rmsg, 0, rmsg.length, port, ip);

}

function send\_test(ip, type) {

let fr = crypto.createHash('md5').update('f12').digest()// fs.readFileSync('./files/test.txt', 'utf8');

var rmsg = Buffer.allocUnsafe(msg\_size + msg\_length\_size + Buffer.byteLength(userName) + fr.length); // Собираем сообщение, чтоб ппослать хэш

rmsg[0] = type;

rmsg.fill(fr, msg\_size, msg\_size + fr.length)

rmsg.writeUInt16BE(Buffer.byteLength(fr), msg\_size + fr.length);

rmsg.fill(userName, msg\_size + fr.length + msg\_length\_size); // + name

dgramSocket.send(rmsg, 0, rmsg.length, port, ip);

}

function u\_send(name, ip, type) {

var rmsg = Buffer.allocUnsafe(msg\_size + msg\_length\_size + Buffer.byteLength(userName) + name.length); // Собираем сообщение, чтоб ппослать хэш

rmsg[0] = type;

rmsg.fill(name, msg\_size, msg\_size + name.length)

rmsg.writeUInt16BE(Buffer.byteLength(userName), msg\_size + name.length);

rmsg.fill(userName, msg\_size + name.length + msg\_length\_size); // + name

dgramSocket.send(rmsg, 0, rmsg.length, port, ip);

}

// Функция широковещательной рассылки

function checkOnline() {

var requestMsg = Buffer.allocUnsafe(msg\_size + msg\_length\_size + Buffer.byteLength(userName) + userId.length); // Формируем сообщение для проверки пользователей

requestMsg[0] = request;

requestMsg.fill(userId, msg\_size, msg\_size + userId.length)

requestMsg.writeUInt16BE(Buffer.byteLength(userName), msg\_size + userId.length);

requestMsg.fill(userName, msg\_size + userId.length + msg\_length\_size); // + name

dgramSocket.send(requestMsg, 0, requestMsg.length, port, ip\_addr);// Отсылаем запрос

}