Государственное образовательное учреждение высшего

профессионального образования

«Московский Государственный Технический Университет имени

Н.Э. Баумана» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

КУРСОВОЙ ПРОЕКТ

Клиентская часть для системы контроля сетевого трафика

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Мацак И. В.

(подпись, дата)

Разработчик проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Гуща Н.В

(подпись, дата)

Москва 2020

Оглавление

[Цель 3](#_Toc59723947)

[Основные определения 3](#_Toc59723948)

[Введение 3](#_Toc59723949)

[Основная идея 4](#_Toc59723950)

[Разработка клиентской части 4](#_Toc59723951)

[Выбор технологий 10](#_Toc59723952)

[Заключение 12](#_Toc59723953)

[Список используемых источников 13](#_Toc59723954)

# Цель

Написать клиентскую часть для системы контроля сетевого трафика, используя алгоритмический язык программирования C++

# Основные определения

**Алгоритмический язык программирования** - формальный язык, используемый для записи, реализации и изучения алгоритмов. В отличие от большинства языков программирования, алгоритмический язык не привязан к архитектуре компьютера, не содержит деталей, связанных с устройством машины.

**Объектно-ориентированное программирование** (ООП) - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

**Сетево́й тра́фик** или **интерне́т-тра́фик** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Traffic* — «движение», «грузооборот») — объём информации, передаваемой через компьютерную сеть за определённый период времени. Количество трафика измеряется как в [пакетах](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_(%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8)), так и в [битах](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82), [байтах](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82) и их производных: [килобайт](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BB%D0%BE%D0%B1%D0%B0%D0%B9%D1%82) (КБ), [мегабайт](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B3%D0%B0%D0%B1%D0%B0%D0%B9%D1%82) (МБ) и т. д.

**Пакет**, **пакет** данных (англ. packet) – **это** сегмент данных отправляемых с одного компьютера или устройства на другое, по **сети**.

# Введение

**Сниффер** – **это** компьютерная программа или часть компьютерной техники, которая может перехватывать и анализировать трафик проходящий через цифровую **сеть** или ее часть. Анализатор захватывает все потоки (перехватывает и протоколирует интернет трафик) и, при необходимости, производит декодирование данных, последовательно сохраняя передаваемую информацию пользователей.

Основная идея

Совокупность серверной и клиентской части системы контроля сетевого трафика позволит отслеживать сетевую активность на компьютере с клиентом. Это полезно например для организаций, для отслеживания продуктивности работы сотрудников, ведь на сервер будет передаваться информация о том, что человек делает в сети, работает или нет.

# Разработка приложения

Рассмотрим некоторые структуры и функции проекта:

* Структура sniff\_ethernet -содержит в себе адрес получателя и отправителя, а так же протокол (IP / ARP/ RARP).
* Структура sniff \_ip – содержит в себе всю информацию связанную с IP
* Структура sniff\_tcp – по аналагии с предыдущей содержит в себе информацию связанную с TCP.
* Функции print\_hex\_ascii\_line и print\_payload – отвечают за запись закодированной информации о переданном файле по сети.
* Функция got\_packet – отвечает за вывод в файл, предназначенный для отправки на сервер, всей оставшейся информации по переданному пакету: IP/URL получателя, IP/URL получателя, дата и время захвата пакета, номер пакета, закодированную информацию по переданному файлу и его размер в байтах, port получателя и port отправителя. URL программа получает из IP c помощью функции getnameinfo().

Рассмотрим функцию main()

За неимением возможности тестировать систему на двух компьютерах, пудем совершать подключение к серверу запущенному на этом же компьютере (IP”0.0.0.0”, port = 80). С помощью сокета подключаемся к серверу. В данном варианте программа может принять только две команды от сервера, каждая из которых должна начинаться с Логина пользователя компьютера с клиентом. Варианты команд: “{login} start” / “{login} close”. Если команда не читается программа отправляет информацию об этом на сервер. Команда close закрывает соединение с сервером и завершает работу программы. Команда start начинает сниффинг и по завершению построчно отправляет файл на сервер в виде набора строк длиной 1024 байта. Файл представляет из себя список считанных пакетов (всего их будет 50) с выведенной по ним информацией. Пример такого файла в формате txt можно увидеть ниже

{

Device: en0  
Number of packets: 50  
Filter expression: ip  
  
Date and time of capture: 25.12.2020 22:0  
Packet number 1:  
From: 192.168.1.102  
To: 192.168.1.68  
Protocol: TCP  
Src port: 548  
Dst port: 51364  
Payload (16 bytes):  
00000 00 05 4f 02 00 00 00 00 00 00 00 00 00 00 00 00 ..O.............  
  
Date and time of capture: 25.12.2020 22:0  
Packet number 2:  
From: 192.168.1.68  
To: 192.168.1.102  
Protocol: TCP  
Src port: 51364  
Dst port: 548  
  
Date and time of capture: 25.12.2020 22:0  
Packet number 3:  
From: 192.168.1.68  
To: 192.168.1.1  
Protocol: UDP

Capture complete.

}

На примере первого пакета можем разобраться в каждой строчке

1. Дата и время захвата

2. Номер пакета

3. IP отправителя

4. IP получателя

5. Используемый протокол

6. Порт отправителя

7. Порт получателя

8. (Объем файла) содержимое пакета в 16-чной и текстовой формах

Задача сервера – принять эти строки и записать их в файл, после чего отформатировать записаный файл.

**Алгоритм запуска и работы системы. (при условии что сервер уже включен):**

1. Через командную строку заходим в папку с проектом .
2. Затем вводом команду(все что идет после “%”) %g++ main.cpp -lpcap (подразумевается что библиотека pcap уже установлена и стабильно работает на данном устройстве).
3. Затем %sudo ./a.out
4. Программа запрашивает логин состоящий ровно из пяти символов, для отправки его на сервер. Это делается для дальнейшего разделения контроля за сотрудниками на сервере.
5. На этом этапе программа начинает общение с сервером и выполняет команды описанные выше, до тех пор пока не примет команду close.

# Выбор технологий

Основа этого проекта лежит в использовании библиотеки pcap. Она предназначена для перехвата сетевого трафика с устройства. При разработке данного приложения были использованы следующие функции.

* pcap\_t \*pcap\_open\_live(char \*device, int snaplen, int promisc, int to\_ms, char \*ebuf) – создает сессию захвата трафика
* pcap\_datalink() – анализ содержимого пакетов.
* int pcap\_compile(pcap\_t \*p, struct bpf\_program \*fp, char \*str, int optimize, bpf\_u\_int32 netmask) – компиляция фильтра (фильтр нужен для получения информации с определенных пакетов, нужных именно нам. В нашем проекте идет захват трафика с фильтром “ip”.
* int pcap\_setfilter(pcap\_t \*p, struct bpf\_program \*fp) – применение фильтра
* int pcap\_loop(pcap\_t \*p, int cnt, pcap\_handler callback, u\_char \*user) - Первый аргумент — дескриптор сессии. Дальше идет целое число, которое сообщает pcap\_loop() количество пакетов, которые нужно захватить (отрицательное значение говорит о том, что цикл должен выполняться до возникновения ошибки). Третий аргумент — имя функции обратного вызова (только идентификатор, без параметров).
* void got\_packet(u\_char \*args, const struct pcap\_pkthdr \*header, const u\_char \*packet); - функция обратного вызова для получения пакета, Первый аргумент соответствует последнему аргументу pcap\_loop(). Независимо от того, какое значение передается последним аргументом pcap\_loop(), оно передается первому аргументу нашей функции обратного вызова. Второй аргумент — это PCAP заголовок, который содержит информацию о том, когда пакет был захвачен, насколько он большой, и так далее. Последний аргумент - то другой указатель на u\_char, и он указывает на первый байт раздела данных содержащихся в пакете, который был захвачен pcap\_loop(). В packet и записывается основная информация такая как IP, TCP и payload. Получить все это из packet можно следующим способом, опирающимся на свойство указателей.

{

ethernet = (struct sniff\_ethernet\*)(packet);

ip = (struct sniff\_ip\*)(packet + SIZE\_ETHERNET);

size\_ip = IP\_HL(ip)\*4;

if (size\_ip < 20) {

printf(" \* Invalid IP header length: %u bytes\n", size\_ip);

return;

}

tcp = (struct sniff\_tcp\*)(packet + SIZE\_ETHERNET + size\_ip);

size\_tcp = TH\_OFF(tcp)\*4;

if (size\_tcp < 20) {

printf(" \* Invalid TCP header length: %u bytes\n", size\_tcp);

return;

}

payload = (u\_char \*)(packet + SIZE\_ETHERNET + size\_ip + size\_tcp);

}

Подключение к серверу происходит с помощью создания и передачи данных по сокету. Для этого используются библиотеки unistd.h, socket.h, in.h, netdb.h.

* gethostbyname() – задание адреса
* int sockfd = socket(domain, type, protocol) - sockfd: дескриптор сокета, целое число (как дескриптор файла)  
  домен: целое число, домен связи, например, AF\_INET (протокол IPv4), AF\_INET6 (протокол IPv6)  
  тип: тип связи  
  SOCK\_STREAM: TCP (надежный, ориентированный на соединение)  
  SOCK\_DGRAM: UDP (ненадежный, без установления соединения)  
  protocol: значение протокола для интернет-протокола (IP), равное 0. Это то же число, которое указывается в поле протокола в заголовке IP пакета.
* int connect(int sockfd, const struct sockaddr \*addr,  
  socklen\_t addrlen);  
  Системный вызов connect () соединяет сокет, указанный дескриптором файла sockfd, с адресом, указанным в addr. Адрес и порт сервера указаны в addr.
* read(handle, buffer, count) – Функция read позволяет читать байты count из файла, связан-  
  ного с handle, в буфер buffer. Операции чтения начинаются с теку-  
  щей позиции указателя на файл, который соответствует данному фай-  
  лу. После выполнения операции чтения этот указатель устанавлива-  
  ется на следующий непрочитанный символ.
* write(handle, buffer, count) – Функция write записывает байты count из буфера buffer в  
  файл, связанный с handle. Операции write начинаются с текущей по-  
  зиции указателя на файл (указатель ассоциирован с заданным фай-  
  лом). Если файл открыт для добавления, операции выполняются в ко-  
  нец файла. После осуществления операций записи указатель на файл  
  (если он есть) увеличивается на количество действительно записан-  
  ных байтов.
* close() – закрытие сокета для общения с сервером

В ходе разработки проекта так же появилась проблема с отправкой файла в виде строки, ведь максимально количество байт, передаваемых в функцию write – 65534, а объем файла с захваченными пакетами может значительно превышать этот объем, поэтому было принято решение построчно отправлять файл на сервер.

# Заключение

В процессе выполнения курсового проекта были изучены новые аспекты языка C++, были освоены принципы объектно-ориентированного программирования, сетевое взаимодействие с помощью сокетов, а так же технология захвата трафика с использованием библиотеки pcap. Результатом работы стала клиентская часть для системы контроля сетевого трафика.

# Список используемых источников

1. Документация: документация по C++. URL: <https://docs.microsoft.com/ru-ru/cpp/cpp/?view=msvc-160>
2. Документация: winpcap. URL: <https://www.winpcap.org/docs/docs_412/html/main.html>
3. <http://www.realcoding.net/>
4. https://habr.com/ru/