|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное автономное образовательное учреждение высшего образования  «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» |

Кафедра №43

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ОТЧЁТ ПО ПРАКТИКЕ  ЗАЩИЩЁН С ОЦЕНКОЙ  Руководитель |  | | | |
| ст. преподаватель |  |  |  | М.Д.Поляк |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |  |
| --- | --- | --- | --- |
| вид практики | производственная | |  |
| тип практики | научно-исследовательская работа | |  |
| на тему индивидуального задания | | Разработка программы захвата и передачи видео по |  |
| сети | | | |
|  | | | |

|  |  |
| --- | --- |
| выполнен | Андреевым Дмитрием Ивановичем |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.04 |  | Программная инженерия |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности |  |  |  |
|  | код |  | наименование направленности |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | z9431 |  |  |  | Д.И.Андреев |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2020

Оглавление

[1. Цель работы 3](#_Toc104934030)

[2. Исходные данные. 4](#_Toc104934031)

[3. Теоретическая часть 5](#_Toc104934032)

[3.1 Общее описание стека TCP/IP 5](#_Toc104934033)

[3.2 Прикладной уровень 6](#_Toc104934034)

[3.3 Транспортный уровень 6](#_Toc104934035)

[3.4 Сетевой (межсетевой) уровень 7](#_Toc104934036)

[3.5 Канальный уровень 8](#_Toc104934037)

[3.6 Описание фреймворка Qt 5.11 9](#_Toc104934038)

[4. Разработка программного обеспечения. 11](#_Toc104934039)

[4.1 Общее описание архитектуры 11](#_Toc104934040)

[4.2 Диаграмма классов разрабатываемого приложения. 11](#_Toc104934041)

[Выводы 14](#_Toc104934042)

# 1. Цель работы

Целью работы является изучение работы TCP/IP стека технологий, а именно реализация передачи данных с использованием протоколов TCP и UDP с помощью фреймворка Qt 5.11.

Для достижения заданной цели, необходимо изучить реализацию протоколов TCP и UDP фреймворком Qt 5.11, а также разработать приложение по захвату изображения экрана на одной машине и передаче его на другую машину, объединенную с первой в одну локальную сеть.

# 2. Исходные данные.

Для реализации задачи будем использовать версию фреймворка Qt 5.11. Данный фреймворк написан на языке C++ 11. Для сборки проекта будем применять систему сборки qmake, которая является «родной» для данной версией фреймворка. Разработка будет проводится под Unix подобную ОС — Linux.

Для тестирования приложения, оно будет развернуто на двух машинах под управлением ОС Ubuntu 20 и Ubuntu 22, объединенных в локальную сеть.

# 3. Теоретическая часть

## 3.1 Общее описание стека TCP/IP

TCP/IP — сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется сеть Интернет.

Набор интернет-протоколов обеспечивает сквозную передачу данных, определяющую, как данные должны пакетироваться, обрабатываться, передаваться, маршрутизироваться и приниматься. Эта функциональность организована в четыре слоя абстракции, которые классифицируют все связанные протоколы в соответствии с объёмом задействованных сетей. От самого низкого до самого высокого уровня:

- Канальный уровень (Network Access Layer) - содержит методы связи для данных, которые остаются в пределах одного сегмента сети;

- Межсетевой уровень (Сетевой уровень) (Internet Layer) - обеспечивает межсетевое взаимодействие между независимыми сетями (напр., IP, ICMP и IGMP);

- Транспортный уровень (Transport Layer) - обрабатывает связь между хостами (напр, TCP, UDP, SCTP, DCCP);

- Прикладной уровень (Application Layer) - обеспечивает обмен данными между процессами для приложений (напр, HTTP, RTSP, FTP, DNS).

Протоколы этих уровней полностью реализуют функциональные возможности модели OSI. На стеке протоколов TCP/IP построено всё взаимодействие пользователей в IP-сетях. Стек является независимым от физической среды передачи данных, благодаря чему, в частности, обеспечивается полностью прозрачное взаимодействие между проводными и беспроводными сетями.

## 3.2 Прикладной уровень

На прикладном уровне (Application layer) работает большинство сетевых приложений.

Эти программы имеют свои собственные протоколы обмена информацией, например, интернет браузер для протокола HTTP, ftp-клиент для протокола FTP (передача файлов), почтовая программа для протокола SMTP (электронная почта), SSH (безопасное соединение с удалённой машиной), DNS (преобразование символьных имён в IP-адреса) и многие другие.

В массе своей эти протоколы работают поверх TCP или UDP и привязаны к определённому порту, например:

- HTTP на TCP-порт 80 или 8080,

- FTP на TCP-порт 20 (для передачи данных) и 21 (для управляющих команд),

- SSH на TCP-порт 22,

- запросы DNS на порт UDP (реже TCP) 53,

- обновление маршрутов по протоколу RIP на UDP-порт 520.

К этому уровню относятся: Echo, Finger, Gopher, HTTP, HTTPS, IMAP, IMAPS, IRC, NNTP, NTP, POP3, POPS, QOTD, RTSP, SNMP, SSH, Telnet, XDMCP.

## 3.3 Транспортный уровень

Протоколы транспортного уровня (Transport layer) могут решать проблему негарантированной доставки сообщений («дошло ли сообщение до адресата?»), а также гарантировать правильную последовательность прихода данных. В стеке TCP/IP транспортные протоколы определяют, для какого именно приложения предназначены эти данные.

Протоколы автоматической маршрутизации, логически представленные на этом уровне (поскольку работают поверх IP), на самом деле являются частью протоколов сетевого уровня; например OSPF (IP идентификатор 89).

TCP (IP идентификатор 6) — «гарантированный» транспортный механизм с предварительным установлением соединения, предоставляющий приложению надёжный поток данных, дающий уверенность в безошибочности получаемых данных, перезапрашивающий данные в случае потери и устраняющий дублирование данных. TCP позволяет регулировать нагрузку на сеть, а также уменьшать время ожидания данных при передаче на большие расстояния. Более того, TCP гарантирует, что полученные данные были отправлены точно в такой же последовательности. В этом его главное отличие от UDP.

UDP (IP идентификатор 17) протокол передачи датаграмм без установления соединения. Также его называют протоколом «ненадёжной» передачи, в смысле невозможности удостовериться в доставке сообщения адресату, а также возможного перемешивания пакетов. В приложениях, требующих гарантированной передачи данных, используется протокол TCP.

UDP обычно используется в таких приложениях, как потоковое видео и компьютерные игры, где допускается потеря пакетов, а повторный запрос затруднён или не оправдан, либо в приложениях вида запрос-ответ (например, запросы к DNS), где создание соединения занимает больше ресурсов, чем повторная отправка.

И TCP, и UDP используют для определения протокола верхнего уровня число, называемое портом.

## 3.4 Сетевой (межсетевой) уровень

Межсетевой уровень (Network layer) изначально разработан для передачи данных из одной сети в другую. На этом уровне работают маршрутизаторы, которые перенаправляют пакеты в нужную сеть путём расчёта адреса сети по маске сети. Примерами такого протокола является X.25 и IPC в сети ARPANET.

С развитием концепции глобальной сети в уровень были внесены дополнительные возможности по передаче из любой сети в любую сеть, независимо от протоколов нижнего уровня, а также возможность запрашивать данные от удалённой стороны, например в протоколе ICMP (используется для передачи диагностической информации IP-соединения) и IGMP (используется для управления multicast-потоками).

ICMP и IGMP расположены над IP и должны попасть на следующий — транспортный — уровень, но функционально являются протоколами сетевого уровня, и поэтому их невозможно вписать в модель OSI.

Пакеты сетевого протокола IP могут содержать код, указывающий, какой именно протокол следующего уровня нужно использовать, чтобы извлечь данные из пакета. Это число — уникальный IP-номер протокола. ICMP и IGMP имеют номера, соответственно, 1 и 2.

К этому уровню относятся: DVMRP, ICMP, IGMP, MARS, PIM, RIP, RIP2, RSVP

## 3.5 Канальный уровень

Канальный уровень (англ. Link layer) описывает способ кодирования данных для передачи пакета данных на физическом уровне (то есть специальные последовательности бит, определяющих начало и конец пакета данных, а также обеспечивающие помехоустойчивость). Ethernet, например, в полях заголовка пакета содержит указание того, какой машине или машинам в сети предназначен этот пакет.

Примеры протоколов канального уровня — Ethernet, IEEE 802.11 WLAN, SLIP, Token Ring, ATM и MPLS.

PPP не совсем вписывается в такое определение, поэтому обычно описывается в виде пары протоколов HDLC/SDLC.

MPLS занимает промежуточное положение между канальным и сетевым уровнем и, строго говоря, его нельзя отнести ни к одному из них.

Канальный уровень иногда разделяют на 2 подуровня — LLC и MAC.

Кроме того, канальный уровень описывает среду передачи данных (будь то коаксиальный кабель, витая пара, оптическое волокно или радиоканал), физические характеристики такой среды и принцип передачи данных (разделение каналов, модуляцию, амплитуду сигналов, частоту сигналов, способ синхронизации передачи, время ожидания ответа и максимальное расстояние).

При проектировании стека протоколов на канальном уровне рассматривают помехоустойчивое кодирование — позволяющие обнаруживать и исправлять ошибки в данных вследствие воздействия шумов и помех на канал связи.

## 3.6 Описание фреймворка Qt 5.11

Qt — фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++. Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Является полностью объектно-ориентированным, расширяемым и поддерживающим технику компонентного программирования.

Начиная с версии 4.5 Qt распространяется по трём лицензиям:

- Qt Commercial — для разработки программного обеспечения с собственнической лицензией, допускающая модификацию самой Qt без раскрытия изменений;

- GNU GPL — для разработки с открытыми исходниками, распространяемыми на условиях GNU GPL, а также для модификации Qt;

- GNU LGPL — для разработки программного обеспечения с собственнической лицензией.

Исходный код, единый для всех вариантов лицензий, свободно доступен в Git-хранилище, расположенном на Github.

Библиотека разделена на ряд модулей. В данной работе будут применяться следующие из них:

- QtCore — классы ядра библиотеки, используемые другими модулями;

- QtNetwork — набор классов для сетевого программирования. Поддержка различных высокоуровневых протоколов может меняться от версии к версии. В версии 4.2.x присутствуют классы для работы с протоколами FTP и HTTP. Для работы с протоколами TCP/IP предназначены такие классы, как QTcpServer, QTcpSocket для TCP и QUdpSocket для UDP;

- Qt QML — модуль для поддержки QML;

# 4. Разработка программного обеспечения.

## 4.1 Общее описание архитектуры

Так как Qt использует Model-View-Delegate для отображения данных, при разработке будет придерживаться данной модели. Часть пользовательского интерфейса будет написана на языке Qml для упрощения разработки, логика приложения, в частности, захват экрана и передача данных на удаленных хост, будет реализовано на стороне C++.

## 4.2 Диаграмма классов разрабатываемого приложения.

Для упрощения на диаграмме классов указаны основные поля и методы.

Основным классом, который агрегирует всю логику, является класс VideoStreamer, представленный на рис. 1. Данный класс контролирует начало и окончание передачи, обмен управляющими сообщениями между хостами на верхнем уровне. Методы данного класса вызываются со стороны Qml посредством зарегистрированного и создаваемого в файле main.qml класса MainWindow.

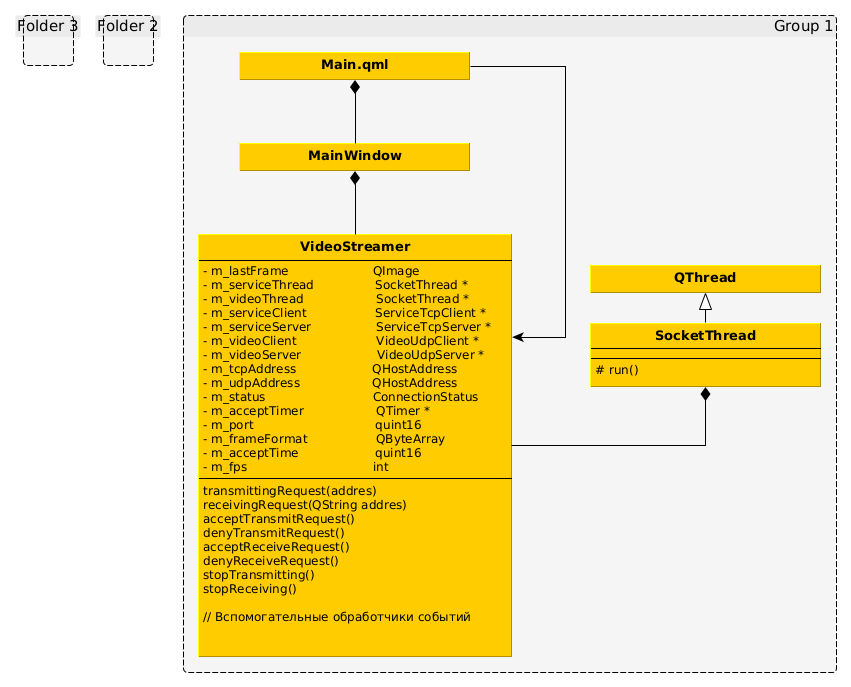
На рис. 2 представлена диаграмма классов части приложения, отвечающей, за передачу управляющих сообщений. Данные классы используют TCP соединение, так как для данного класса сообщений необходима гарантированная доставка пакетов. Перед началом работы, класс VideoStreamer инициализирует классы данной группы портом, по которому хосты будут передавать сообщения.

Рисунок 1 Диаграмма классов верхнего уровня приложениях

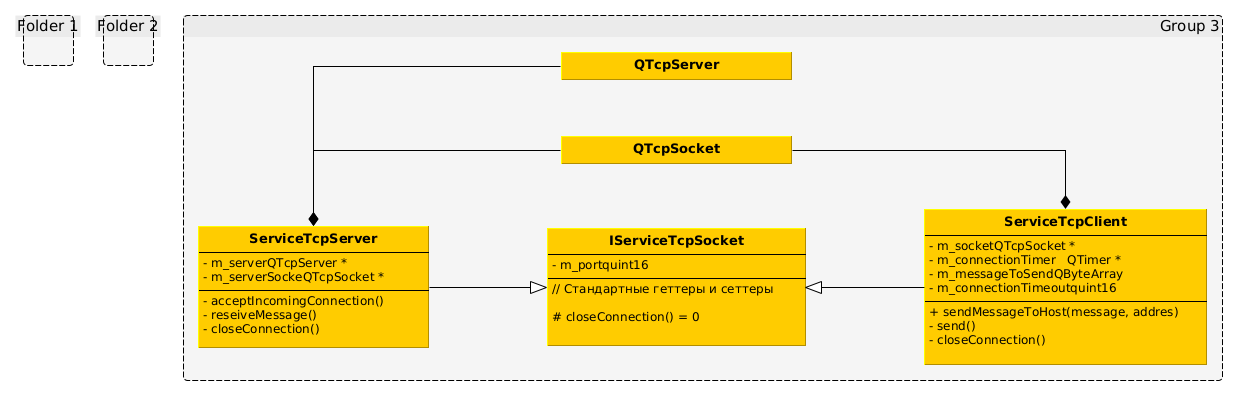
На рис. 3 представлена диаграмма классов части приложения, отвечающей за захват кадров и передачу их. Данные классы реализуют логику с использованием протокола UDP, так как передача кадров не нуждается в гарантированной доставке, а также, так как при передаче большого числа больших по размеру пакетов или при передаче пакетов с нестабильным соединением, есть вероятность переполнения буфера и полной остановки передачи.

Рисунок 2 Диаграмма классов части приложения, отвечающей за отправку управляющих сообщений

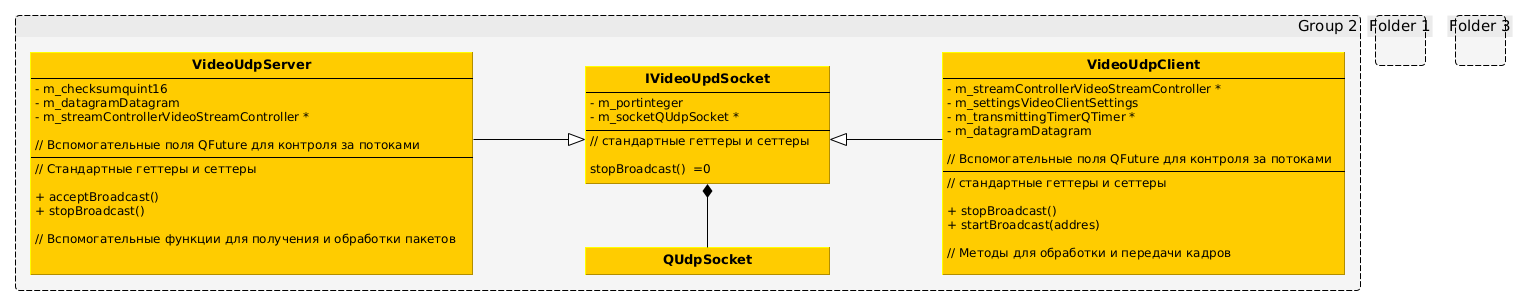


Рисунок 3 Диаграммы классов части приложения, отвечающей за отправку полезной нагрузки (кадров видеопотока)

# Выводы

В ходе практики были изучены протоколы передачи сообщений TCP и UDP из стека TCP/IP, а также реализована программа, демонстрирующая работу данных протоколов.

Из основных недостатков данной программы, можно отметить невозможность использования протоколов кодирования видеопотока (например, h264/h265/mpeg4). Исправление данного недостатка влечет за собой необходимость перехода с протоколов TCP/UDP на протоколы RTCP/RTP, так как для потоковой передачи видео в кодированных форматах, необходима передача дополнительной метаинформации (например, временные метки или флаги передачи новых I кадров в случае кодирования протоколами h264/h265).

Из очевидных преимуществ, можно выделить использование фреймворка Qt, который позволяет гибко расширять приложение и реализовывать более сложный пользовательский интерфейс за относительно малое время и с относительно малыми издержками.

Ссылка на репозиторий с приложением:

https://github.com/MrAnderson1110/VSC.git

**Использованная литература**

- Терри Оглтри. Модернизация и ремонт сетей = Upgrading and Repairing Networks. — 4-е изд. — М.: «Вильямс», 2005. — С. 1328. — ISBN 0-7897-2817-6.

- Дуглас Камер. Сети TCP/IP, том 1. Принципы, протоколы и структура = Internetworking with TCP/IP, Vol. 1: Principles, Protocols and Architecture. — М.: «Вильямс», 2003. — С. 880. — ISBN 0-13-018380-6.

- Семенов Ю. А. Протоколы Internet. — 2-е изд., стереотип.. — М.: Горячая линия - Телеком, 2005. — 1100 с. — ISBN 5-93517-044-2.

- Паркер Т., Сиян К. TCP/IP. Для профессионалов. — 3-е изд.. — СПб.: Питер, 2004. — 859 с. — ISBN 5-8046-0041-9.

- Таненбаум Эндрю, Уэзеролл Дэвид Компьютерные сети — 5-е изд. - СПб.: Питер, 2016 — 960 с. — 978-5-496-00831-0, 978-0132126953