Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Романчук Иван Сергеевич
Должность: Ректор
Дата подписания: 18.07.2024 10:49:06
Уникальный программный ключ:
6319edc2b582ffdacea443f01d5779368d0957ac34f5cd074d81181530452479

РОССИЙСКАЯ ФЕДЕРАЦИЯ МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ФГАОУ ВО ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Захаров А. А.

ОСНОВЫ ПОСТРОЕНИЯ ЗАЩИЩЕННЫХ КОМПЬЮТЕРНЫХ СЕТЕЙ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

ТЮМЕНЬ 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Целью дисциплины «Основы построения защищенных компьютерных сетей» является изложение основополагающих принципов разработки сетевого программного обеспечения в различных средах с использованием различных информационных технологий при решении разнообразных прикладных задач.

Задачей курса является:

- основных принципов разработки сетевых протоколов;
- основных принципов анализа сетевых протоколов;
- принципов разработки сетевых программ и выбора технологии и протокола передачи данных.

Изучение курса основано на следующих дисциплинах: «Организация электронно-вычислительных машин и вычислительных систем», «Структуры и алгоритмы компьютерной обработки данных», «Языки программирования» В результате изучения дисциплины студенты должны знать:

- содержание процессов самоорганизации и самообразования, их особенностей и технологий реализации, исходя из целей совершенствования профессиональной деятельности.
- принципы функционирования протоколов FTP, HTTP, SMTP и POP3, стандартные команды протоколов.
- Basic, Digest, NTLM и авторизацию с помощью форм.

уметь:

- самостоятельно строить процесс овладения информацией, отобранной и структурированной для выполнения профессиональной деятельности; планировать цели и устанавливать приоритеты при выборе способов принятия решений с учетом условий, средств, личностных возможностей и временной перспективы достижения.
- производить основные действия с протоколами FTP, HTTP, SMTP или POP3 программно.
- настраивать Basic, Digest, NTLM и авторизацию с помощью форм.

Тема 1. ОСНОВНЫЕ ПОНЯТИЯ

Протокол TCP (Transmission Control Protocol) – это сетевой протокол, который «заточен» под соединение. Иными словами, прежде, чем начать обмен данными, данному протоколу требуется установить соединение между двумя хостами. Данный протокол имеет высокую надежность, поскольку позволяет не терять данные при передаче, запрашивает подтверждения о получении от принимающей стороны и в случае необходимости отправляет данные повторно. При этом отправляемые пакеты данных сохраняют порядок отправки, то есть можно сказать, что передача данных упорядочена. Минусом данного протокола является относительно низкая скорость передачи данных, за счет того что выполнение надежной и упорядоченной передачи занимает больше времени, чем в альтернативном протоколе UDP.

Протокол UDP (User Datagram Protocol), в свою очередь, более прост. Для передачи данных ему не обязательно устанавливать соединение между отправителем и получателем. Информация передается без предварительной проверки готовности принимающей стороны. Это делает протокол менее надежным – при передаче некоторые фрагменты данных могут теряться. Кроме того, упорядоченность данных не соблюдается – возможен непоследовательный прием данных получателем. Зато скорость передачи данных по данному транспортному протоколу будет более высокой.

Сравнение по основным пунткам: основных

пунктов:

- **Надежность**: в этом случае предпочтительнее будет протокол TCP, за счет подтверждения получения данных, повторной отправки в случае необходимости, а также использованию такого инструмента как тайм-аут. Протокол UDP такого инструментария не имеет, а потому при получении отправленные данные могут приходить не полностью;

- **Упорядоченность**: опять будет предпочтительнее TCP, поскольку этот протокол гарантирует передачу пакетов данных именно в том порядке, в котором они были отправлены. В случае с UDP такой порядок не соблюдается;
- **Скорость**: здесь уже лидировать будет UDP, так как более тяжеловесному TCP-протоколу будет требоваться больше времени для установки соединения, подтверждения получения, повторной отправки данных и т.д.;
- **Метод передачи данных**: в случае с TCP данные передаются потоково, границы фрагментов данных не имеют обозначения. В случае с UDP данные передаются в виде датаграмм – проверка пакетов на целостность осуществляется принимающей стороной только в случае получения сообщения. Также пакеты данных имеют определенные обозначения границ;

Вопросы для подготовки

1. Протоколы TCP и UDP.
2. Сетевые протоколы уровня приложения.
3. Понятие стандарта на протокол.
4. Стандарты RFC и IETF.

Тема 2. ПРОТОКОЛ HTTP

HTTP — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Аббревиатура HTTP расшифровывается как *HyperText Transfer Protocol*, «протокол передачи гипертекста». В соответствии со спецификацией OSI, HTTP является протоколом прикладного (верхнего, 7-го) уровня. Актуальная

на данный момент версия протокола, HTTP 1.1, описана в спецификации RFC 2616.

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом.

Задача, которая традиционно решается с помощью протокола HTTP — обмен данными между пользовательским приложением, осуществляющим доступ к веб-ресурсам (обычно это веб-браузер) и вебсервером. На данный момент именно благодаря протоколу HTTP обеспечивается работа Всемирной паутины.

Также HTTP часто используется как протокол передачи информации для других протоколов прикладного уровня, таких как SOAP, XML-RPC и WebDAV. В таком случае говорят, что протокол HTTP используется как «транспорт».

API многих программных продуктов также подразумевает использование HTTP для передачи данных — сами данные при этом могут иметь любой формат, например, XML или JSON.

Как правило, передача данных по протоколу HTTP осуществляется через TCP/IP-соединения. Серверное программное обеспечение при этом обычно использует TCP-порт 80 (и, если порт не указан явно, то обычно клиентское программное обеспечение по умолчанию использует именно 80-й порт для открываемых HTTP-соединений), хотя может использовать и любой другой.

Сам по себе протокол HTTP не предполагает использование шифрования для передачи информации. Тем не менее, для HTTP есть распространённое

расширение, которое реализует упаковку передаваемых данных в криптографический протокол **SSL** или **TLS**.

Название этого расширения — **HTTPS** (*HyperText Transfer Protocol Secure*). Для HTTPS-соединений обычно используется TCP-порт 443. HTTPS широко используется для защиты информации от перехвата, а также, как правило, обеспечивает защиту от атак вида **man-in-the-middle** — в том случае, если сертификат проверяется на клиенте, и при этом приватный ключ сертификата не был скомпрометирован, пользователь не подтверждал использование неподписанного сертификата, и на компьютере пользователя не были внедрены сертификаты центра сертификации злоумышленника.

Протокол HTTP предполагает достаточно большое количество возможностей для расширения. В частности, спецификация HTTP 1.1 предполагает возможность использования заголовка Upgrade для переключения на обмен данными по другому протоколу. Запрос с таким заголовком отправляется клиентом. Если серверу требуется произвести переход на обмен данными по другому протоколу, то он может вернуть клиенту ответ со статусом «426 Upgrade Required», и в этом случае клиент может отправить новый запрос, уже с заголовком Upgrade.

Такая возможность используется, в частности, для организации обмена данными по протоколу WebSocket (протокол, описанный в спецификации RFC 6455, позволяющий обеим сторонам передавать данные в нужный момент, без отправки дополнительных HTTP-запросов): стандартное «рукопожатие» (handshake) сводится к отправке HTTP-запроса с заголовком Upgrade, имеющим значение «websocket», на который сервер возвращает ответ с состоянием «101 Switching Protocols», и далее любая сторона может начать передавать данные уже по протоколу WebSocket.

Вопросы для подготовки

1. История протокола.
2. Версии протокола.
3. Структура запроса.
4. Структура ответа.
5. Поля.
6. Коды ответов и их значения.
7. Аутентификация в протоколе HTTP.

Лабораторная работа №1 "Разработка чата для локальной сети"

Разработать чат для локальной сети с использованием протокола UDP. Запрещается использовать готовые сторонние библиотеки.

Тема 3. ПРОТОКОЛ FTP

FTP расшифровывается как File Transfer Protocol — протокол передачи файлов. Он отличается от других протоколов тем, что если в процессе передачи возникает какая-то ошибка, то процесс останавливается и выводится сообщение для пользователя. Если ошибок не было, значит, пользователь получил именно тот файл, который нужен, в целости и без недостающих элементов.

По FTP-протоколу можно скачивать что угодно: фильмы, музыку, документы, программы, драйверы и картинки. Сейчас многие производители железа выкладывают драйверы от устройств на FTP-серверы, чтобы их могли скачать все желающие.

В корпоративной среде FTP используется для организации локального хранилища внутренних документов и файлов для работы. Например, там могут храниться видеолекции или архивные сканы документов. Ещё FTP позволяет загружать свои файлы на сервер, чтобы их мог скачать любой желающий.

Программисты иногда используют такие серверы для обмена файлами и для бэкапов кода, хотя многие для этого предпочитают GIT. Про него ещё поговорим отдельно.

Клиент и сервер

Для работы по FTP нужны двое: FTP-сервер и FTP-клиент. Что делает сервер:

- обеспечивает доступ по логину и паролю к нужным файлам;
- показывает пользователю только те файлы и папки, которые он может просматривать или загружать в них;
- следит за качеством передачи и смотрит, чтобы не было ошибок;
  управляет параметрами соединения в пассивном режиме.

Так как FTP пришёл к нам из времён UNIX-систем, то любое соединение требует логина и пароля. Если у пользователя его нет, сервер его не пропустит. Но чтобы сделать файлы доступными для всех, используют анонимный режим. В нём логином будет слово anonymous, а паролем — любой адрес электронной почты. Современные браузеры умеют сами заходить на анонимные FTP-серверы и подставлять почту. Со стороны это выглядит так, как будто никакого логина и пароля нет, но они есть.

Когда запускается FTP-сервер, ему говорят: «Уважаемый сервер, вот список файлов и папок, которые нужно показывать на сервере. Если к тебе постучится пользователь с таким-то логином и паролем, то покажи ему всё, а если с вот таким логином — то дай ему одну только эту папку. Анонимов не пускать». Ещё один обязательный параметр — адрес сервера и порт, по которому будет идти передача файлов.

Для FTP не нужен сайт, то есть веб-интерфейс. Не нужно запускать вебсервер, настраивать шаблоны вывода списка файлов и поднимать отдельную программу, которая будет нам отдавать эти файлы (типа

Вордпресса). FTP — это как доступ к удаленной папке: ты сразу видишь файлы и можешь их качать, без посредников. А в вебе нужна какая-то программа, которая «нарисует» тебе файловую систему и поставит ссылки на файлы.

В FTP уже реализованы вопросы авторизации и прав. А в вебе их нужно создавать: например, ставить тот же Вордпресс и к нему прикручивать плагины с системой доступа. Или настраивать Apache, генерировать ключи доступа, раскладывать конфигурационные файлы по папкам — это гораздо менее элегантно, чем настройка FTP.

В FTP можно разрешить или запретить отдельным пользователям загружать файлы на FTP-сервер. В вебе загрузка файлов от пользователя на сервер — это на порядок более сложная задача.

Сам по себе FTP-протокол надёжен и гарантированно доставляет пользователю нужные файлы, если с соединением всё в порядке.

Проблема в том, что протокол изначально был незащищённый, и предполагалось, что канал передачи данных всегда надёжен. Поэтому в FTP всё передаётся в открытом виде: файлы, пароли, имена пользователей и любые данные.

Вопросы для подготовки

1. История протокола.
2. Версии протокола.
3. Команды протокола.
4. Структура ответа.
5. Коды ответов и их значения.
6. Аутентификация.

Лабораторная работа №2 "Разработка простейшего HTTP-сервера"

Разработать простейший HTTP-сервер. Запрещается использовать готовые сторонние библиотеки. Продемонстрировать работу сервера при помощи веб-браузера.

Тема 4. ПРОТОКОЛ POP3

POP3 (протокол почтового отделения версия 3) часто используется для связи с удаленным сервером электронной почты и загрузки сообщений на локальный почтовый клиент с последующим удалением его на сервере, к примеру Outlook, Thunderbird, Windows Mail, Mac Mail и т. д. Однако обычно почтовые клиенты предлагают выбор — оставлять или нет копии сообщений на сервере. Если вы используете несколько устройств для отправки сообщений, то рекомендуется оставлять эту функцию включенной, в противном случае, на другом устройстве у вас не будет доступа к отправленным сообщениям, которые не были сохранены на удаленном сервере. Также стоит отметить, что POP3 — протокол, работающий только в одном направлении, это означает, что данные берутся с удаленного сервера и отправляются на локальный клиент.

Порты POP3, по умолчанию являются такими:

Порт 110 — порт без шифрования

Порт 995 — порт SSL/TLS, также известный как **POP3S**

IMAP (протокол прикладного уровня для доступа к электронной почте), также как и POP3 используется для получения сообщений электронной почты на локальный клиент, однако, он имеет существенное отличие — загружаются только лишь заголовки электронных сообщений, сам текст письма остается на сервере. Данный протокол связи работает в две стороны, если происходят изменения на локальном клиенте, они передаются и на сервер. В последнее время IMAP стал более популярным, так как такие гиганты-провайдеры услуг

электронной почты, как Gmail, стали рекомендовать использовать его вместо POP3.

Порты IMAP, по умолчанию являются такими:

Порт 143 — порт без шифрования

Порт 993 — порт SSL/TLS, также известный как **IMAPS**

Вопросы для подготовки

1. История протокола.
2. Команды протокола.
3. Коды ответов и их значения.
4. Аутентификация.

Лабораторная работа №3 "Разработка простейшего FTP-сервера"

Разработать простейший FTP-сервер. Запрещается использовать готовые сторонние библиотеки. Для демонстрации работы сервера составить клиентское приложение с методами создания, удаления и просмотра каталога, перехода между каталогами, загрузки, выгрузки и удаления файлов.

Тема 5. ПРОТОКОЛ SMTP

SMTP (англ. Simple Mail Transfer Protocol — простой протокол передачи почты) — это сетевой протокол, предназначенный для передачи электронной почты между сервером отправителя и почтовым клиентом/сервером получателя

SMTP-сервер выполняет две функции:

- Проверяет правильность настроек и выдает разрешение компьютеру, пытающемуся отправить email-сообщение.
- Отправляет исходящее сообщение на указанный адрес и удостоверяется в успешной доставке сообщения. Если его невозможно доставить, отправителю направляется сообщение об этом.

И в случае с бесплатными SMTP серверами, и серверами, которые вы арендуете, основные параметры — это логин и пароль пользователя.

Логин и пароль — это данные, что использовались при регистрации в системе того сервиса, SMTP сервер которого будет использоваться. На основе данной информации будет проводиться аутентификация отправителя.

Например, для бесплатного SMTP сервера Gmail, Yandex, Mail.ru, Yahoo и т.д. в качестве логина и пароля выступают персональные данные для входа в почту. В случае ручной настройки SMTP, например, в программе ePochta Mailer, нужно дополнительно указывать имя сервера, порт и шифрование.

Детальнее о настройках SMTP сервера для ePochta Mailer можно прочитать в справке к программе.

25 порт — это стандартный порт, который по умолчанию используется для работы SMTP протокола. Иногда интернет-провайдеры закрывают к нему доступ с целью блокировки спам-рассылок.

В таком случае можно использовать один из дополнительных портов, на которых может работать протокол:

465 — порт требует защищенного SSL соединения,

587 — дополнительный порт, при использовании которого требуется аутентификация пользователя (проверка подлинности данных отправителя).

Как уже упоминалось, SMTP протокол отвечает за отправку почты. Эта операция сопровождается выполнением ряда команд, в виде последовательностей команд и ответов.

Основные параметры, которые передаются серверу соответствующими командами:

MAIL FROM — электронный адрес отправителя

RCPT TO — email получателя

DATA — заголовок и тело письма

Вопросы для подготовки

1. История протокола.
2. Команды протокола.
3. Коды ответов и их значения.

Лабораторная работа №4 "Разработка почтового сервера на базе POP3"

Разработать простейший почтовый сервер с применением протокола POP3. Запрещается использовать готовые сторонние библиотеки.

Тема 6. УЯЗВИМОСТИ СЕТЕВЫХ ПРОТОКОЛОВ

Протокол HTTP работает в сеансовом режиме. Это означает, что связь разрывается сразу же после получения необходимой информации. Действует это следующим образом: вы вводите адрес необходимой страницы в адресной

строке браузера, дальше ваш компьютер устанавливает соединение с сервером, на котором располагается эта страница. Браузер посредством протокола загружает html-код и отображает его на мониторе с учетом всей информации. Если вдруг в коде имеются сноски на другое содержимое (графика, звук), то браузер опять устанавливает соединение с сервером, на котором лежат данные объекты, и загружает их на компьютер, после чего соединение разрывается, и на монитор все выводится непосредственно с компьютера пользователя (рис.1). При этом каждый из документов отображается независимо от другого. Именно поэтому в браузерах имеется возможность отключения изображений и т.д. При включении этой функции браузер просто не догружает дополнительные объекты (изображения).

Активные элементы

Активными элементами являются программы небольшого размера, использующие http-протокол и работающие автоматически. Среди таковых можно выделить Java-апплеты, управляющие элементы ActiveX и команды javascript. Именно эти элементы и являются наиболее уязвимыми местами в протоколе, так как наличие того или иного элемента на загружаемой странице сразу же вызывает его запуск, и, если у вас не стоит антивирус с возможностью постоянного сканирования сетевых протоколов, этот активный элемент может загрузить на ваш компьютер вредоносное ПО, при этом не подав никаких признаков. Если же компьютер оснащен необходимыми средствами безопасности, то он сразу де среагирует на попытку загрузки вируса.

FTP - это протокол, который позволяет обмениваться файлами в режиме менеджера файлов. Как он работает? Сначала клиент (пользовательский компьютер) с любого порта устанавливает соединение с портом 21 сервера. После сервер устанавливает канал обмена данными через порт 20. Именно это и является уязвимым местом этого протокола. Вместо сервера может выступить злоумышленник и отправить на компьютер жертвы вредоносное программное обеспечение, которое будет воспринято системой как файл,

запрашиваемый пользователем. Одним из методов противостояния такого рода атакам является так называемый пассивный метод, когда компьютер пользователя сам дает команду на установку соединения (рис. 2). Пассивный FTP был добавлен к основному большей частью не из-за волнений о безопасности пользователей, а из-за того, что пользователи стали беспокоится о своей безопасности. Применение брандмауэр-систем привело к тому, что серверы не могли установить соединение с клиентом. Поэтому они перешли на пассивный режим, чтобы клиент сам мог инициировать подключение командой port

Вопросы для подготовки

1. Уязвимости протокола HTTP.
2. Уязвимости протокола FTP.
3. Уязвимости протокола POP3.
4. Уязвимости протокола SMTP.

Лабораторная работа №5 "Разработка почтового сервера на базе SMTP"

Разработать простейший почтовый сервер с применением протокола SMTP. Запрещается использовать готовые сторонние библиотеки.

Тема 7. ОБЗОР СОВРЕМЕННЫХ СЕТЕВЫХ ПРОТОКОЛОВ

На канальном уровне (Data Link) вам нужно соединить устройства между собой. Они могут находиться как недалеко, например, в локальных сетях (local networks) так и на большом расстоянии друг от друга: в городских (metropolitan area networks) и глобальных сетях (wide area networks).

В настоящее время на этом уровне в домашних и офисных сетях (LAN) используются Ethernet и Wi-Fi, а в мобильных (WAN) — 3G / 4G. Однако многие IoT-устройства маломощные, например, датчики, и питаются только от

батарей. В этих случаях Ethernet не подходит, но можно использовать low powered Wi-Fi и low powered Bluetooth.

Хотя для подключения этих устройств по-прежнему будут использоваться существующие беспроводные технологии (Wi-Fi, Bluetooth, 3G / 4G), стоит также обратить внимание на новые технологии, специально разработанные для IoT-приложений, популярность которых, по всей вероятности, будет расти.

Среди них:

- BLE – Bluetooth Low Energy
- LoRaWAN – Long Range WAN
- SigFox
- LTE-M

Сетевой уровень

На сетевом уровне (Networking) в долгосрочной перспективе будет доминировать протокол IPv6. Маловероятно, что будет использоваться IPv4, но он может играть определенную роль на начальных этапах. Большинство IoT-устройств для дома, например, умные лампочки, в настоящее время используют IPv4.

Транспортный уровень

На транспортном уровне (Transport) в Интернете и вебе доминирует TCP. Он используется как в HTTP, так и во многих других популярных протоколах Интернета (SMTP, POP3, IMAP4 и т. д.).

MQTT, который, как я ожидаю, станет одним из основных протоколов прикладного уровня для обмена сообщениями, в настоящее время использует TCP.

Однако в будущем из-за более низких накладных расходов, я ожидаю, что UDP будет более популярен для IoT. Вероятно, более широкое

распространение получит MQTT-SN, работающий поверх UDP. См. статью о сравнении TCP vs UDP.

Прикладной уровень и протоколы обмена сообщениями Важные характеристики для протоколов IoT:

- Скорость — количество передаваемых данных в секунду.
- Задержка — время, необходимое для передачи сообщения.
- Потребляемая мощность.
- Безопасность.
- Наличие программных средств.

В настоящее время на этом уровне активно используются два основных протокола: HTTP и MQTT.

HTTP, вероятно, самый известный протокол этого уровня, лежащий в основе веба (WWW). Он по-прежнему будет иметь важное значение для IoT, поскольку используется для REST API — основного механизма взаимодействия веб-приложений и сервисов. Однако, из-за высоких накладных расходов, HTTP вряд ли станет основным протоколом IoT, хотя все равно будет широко использоваться в Интернете.

Вопросы для подготовки

1. Обзор наиболее существенных современных сетевых протоколов. **Лабораторная работа №6 "Перехват сообщений"**

Разработать программу для перехвата сообщений между браузером и Web-сервером.

Изучить результаты перехвата. Проанализировать защищенные запросы к Web-серверу с использованием BASIC авторизации. Произвести запросы к Web-серверу, используя telnet. Извлечь имя пользователя и время, полученное в ответе.

Тема 8. РАЗРАБОТКА СЕТЕВЫХ ПРИЛОЖЕНИЙ НА БАЗЕ ПРОТОКОЛА TCP

Создание компьютерных сетей представляет собой сложную организационную задачу.

Во многих случаях бывает необходимо организовать взаимодействие между компьютерами различных архитектур, операционным системами и прикладным обеспечением разных версий и производителей. Среда передачи данных тоже может быть крайне разнородной –оптические каналы, сети Ethernet и другие специализированные кабельные сети, модемы для связи через телефонные сети и различные беспроводные технологии, действующие от масштаба «рабочего стола» (Bluetooth) до всей планеты (спутниковая связь). Успешная организация связи в таких разнородных условиях возможна только за счет использования общепринятых стандартов и структурного подхода к решению этой задачи. Для работы в разнородной аппаратной среде обычно создают программное обеспечение с многоуровневой структурой. Каждый уровень решает свою конкретную задачу и взаимодействует только с соседними уровнями. Самым нижним уровнем в случае сетевого взаимодействия будет являться уровень, на котором будет осуществляться физическая связь между непосредственно соединенными узлами сети (компьютерами или специальными сетевыми устройствами). На самом верхнем уровне будут находиться пользовательские процессы. Промежуточные уровни должны «спустить» данные от пользовательского процесса до физического уровня, а затем (на другом компьютере) «поднять» их опять до уровня пользовательского процесса. Вертикальное взаимодействие между уровнями обеспечивается стандартизацией способов передачи данных между уровнями, называемыми интерфейсами. Спуская данные до самого нижнего уровня, и поднимая их обратно на другом компьютере, нижние слои обеспечивают своеобразный

виртуальный канал данных между слоями, находящимися на одном уровне на разных компьютерах. Как правило, каждый уровень добавляет к передаваемым данным свою служебную информацию, которая анализируется аналогичным уровнем другой системы Формальный набор правил, определяющих последовательность и формат сообщений, которыми обмениваются сетевые компоненты различных вычислительных систем, находящихся на одном уровне, называют сетевым протоколом. Также этот термин используют для обозначения программного обеспечения, реализующего это набор правил.

Совокупность интерфейсов и протоколов, достаточную для организации взаимодействия удаленных процессов, называют семейством протоколов или стеком протоколов.

Среди существующих методов построения сетей, наиболее распространёнными являются два метода, известные как коммутация каналов и коммутация пакетов. В случае коммутации каналов перед обменом данных между клиентами сети создаётся составной канал, используемый для передачи данных только этими клиентами. Преимуществом такого способа является предоставление клиентам гарантированного качества связи: пропускная способность и задержки в канале известны заранее и не изменяются в процессе работы. Такой способ коммутации хорош для передачи непрерывного потока данных, например видео или звука

Вопросы для подготовки

1. Подходы к разработке сетевых приложений.
2. Сетевая библиотека WinSock.
3. Разработка сетевого приложения на базе оконных сообщений.
4. Разработка сетевого приложения на базе событий.
5. Разработка сетевого приложения в блокирующем режиме.

6. Разработка сетевого приложения в режиме асинхронного завершения операций.

Лабораторная работа №7

Реализовать собственный SOCKS4 прокси-сервер.

Тема 9. АНОНИМНЫЕ И ИМЕНОВАННЫЕ КАНАЛЫ СВЯЗИ

В среде операционной системы Microsoft Windows NT вам доступно такое удобное средство передачи данных между параллельно работающими процессами, как каналы типа Pipe. Это средство позволяет организовать передачу данных между локальными процессами, а также между процессами, запущенными на различных рабочих станциях в сети.

Каналы типа Pipe больше всего похожи на файлы, поэтому они достаточно просты в использовании.

Через канал можно передавать данные только между двумя процессами. Один из процессов создает канал, другой открывает его. После этого оба процесса могут передавать данные через канал в одну или обе стороны, используя для этого хорошо знакомые вам функции, предназначенные для работы с файлами, такие как ReadFile и WriteFile. Заметим, что приложения могут выполнять над каналами Pipe синхронные или асинхронные операции, аналогично тому, как это можно делать с файлами. В случае использования асинхронных операций необходимо отдельно побеспокоиться об организации синхронизации.

Существуют две разновидности каналов Pipe - именованные (Named Pipes) и анонимные (Anonymous Pipes).

Как видно из названия, именованным каналам при создании присваивается имя, которое доступно для других процессов. Зная имя какойлибо рабочей станции в сети, процесс может получить доступ к каналу, созданному на этой рабочей станции.

Анонимные каналы обычно используются для организации передачи данных между родительскими и дочерними процессами, запущенными на одной рабочей станции или на “отдельно стоящем” компьютере.

Имена каналов в общем случае имеют следующий вид:

\\ИмяСервера\pipe\ИмяКанала

Если процесс открывает канал, созданный на другой рабочей станции, он должен указать имя сервера. Если же процесс создает канал или открывает канал на своей рабочей станции, вместо имени указывается символ точки:

\\.\pipe\ИмяКанала

В любом случае процесс может создать канал только на той рабочей станции, где он запущен, поэтому при создании канала имя сервера никогда не указывается.

В простейшем случае один серверный процесс создает один канал (точнее говоря, одну реализацию канала) для работы с одним клиентским процессом.

Однако часто требуется организовать взаимодействие одного серверного процесса с несколькими клиентскими. Например, сервер базы данных может принимать от клиентов запросы и рассылать ответы на них.

В случае такой необходимости серверный процесс может создать несколько реализаций канала, по одной реализации для каждого клиентского процесса.

В этом разделе мы опишем наиболее важные функции программного интерфейса Microsoft Windows NT, предназначенные для работы с каналами Pipes. Более подробную информацию вы найдете в документации, которая поставляется в составе библиотеки разработчика Microsoft Development Library (MSDN).

Для создания именованных и анонимных каналов Pipes используются функции CreatePipe и CreateNamedPipe.

Вопросы для подготовки

1. Понятие канала.
2. Возможности и назначение канала.

3. Анонимные и именованные каналы связи.

4. Передача информации через канал.

Лабораторная работа №8

Используя любой язык программирования, разработать систему мгновенных сообщение на протоколах SMTP и POP3. Использовать почтовые серверы, разработанные в ходе выполнения предыдущих практических работ.

СПИСОК ЛИТЕРАТУРЫ

1. **Паринов А.В. Сети связи и системы коммутации:** Учебное пособие / Паринов А.В., Ролдугин С.В., Мельник В.А. - Воронеж: Научная книга, 2016. - 178 с. ISBN 978-5-4446-0906-4. - Текст: электронный. - URL: https://znanium.com/catalog/product/923309 (дата обращения: 15.05.2020). – Режим доступа: по подписке.
2. **Громов Ю. Ю. Архитектура ЭВМ и систем:** учебное пособие / Ю. Ю. Громов, О. Г. Иванова, М. Ю. Серегин [и др.]. — Тамбов: Тамбовский государственный технический университет, ЭБС АСВ, 2012. — 200 с. — ISBN 2227–8397. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/64069.html (дата обращения: 15.05.2020). — Режим доступа: для авторизированным пользователей
3. **Ахметова О. С. Компьютерные сети:** учебно-методический комплекс / составители О. С. Ахметова, А. Опабекова, А. М. Сатымбеков. — Алматы: Нур-Принт, 2012. — 295 с. — ISBN 9965-756-19-8. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/67067.html (дата обращения: 15.05.2020). — Режим доступа: для авторизированных пользователей

Интернет-ресурсы

1. Вузовские электронно-библиотечные системы учебной литературы.
2. База научно-технической информации ВИНИТИ РАН
3. Доступ к открытым базам цитирования, в т.ч. springer.com, scholar.google.com, math-net.ru
4. http://www.ietf.org/rfc.html [On-line] - документы IETF – инженерного совета Интернета.
5. http://msdn.microsoft.com