Лабораторная работа № 2
Клиент-серверное приложение для передачи файлов с использованием протоколов TCP и UDP

2.1. Цель работы

Изучение принципов работы клиент-серверных приложений на основе протоколов TCP и UDP, реализующих функцию передачи файлов.

2.2. Задание на лабораторную работу

Разработать два клиент-серверных приложения, основанных на транспортных протоколах TCP и UDP. Каждое клиент-серверное приложение должно состоять из двух самостоятельных модулей: клиентской части и серверной части.

Клиент-серверное приложение, основанное на транспортном протоколе TCP. Клиент должен инициировать соединение с сервером и отправлять ему файлы любых типов. Сервер должен принимать файлы клиента, выводить информацию о полученных файлах на экран, а также сохранять полученные файлы в локальном каталоге. Сервер также должен иметь возможность отправлять файлы подключённому клиенту.

Клиент-серверное приложение, основанное на транспортном протоколе UDP. Клиент и сервер имеют одинаковое строение, они оба должны иметь возможность отправлять файлы любых типов друг другу, выводить информацию о полученных файлах на экран, а также сохранять полученные файлы в локальном каталоге.

Все приложения должны иметь простейший графический интерфейс.

Для данной лабораторной работы необходимо использовать неблокируемые (асинхронные) сокеты.

2.3. Методические указания

Передача файлов по сети несколько отличается от передачи текстовых сообщений:

1) Каждый файл можно представить в виде последовательности байт, которые, как и текстовое сообщение можно передать по сети, однако файл обладает некоторыми неотъемлемыми атрибутами:
    a. Имя файла
    b. Расширение файла
    c. Размер файла
2) Файл является целостной структурой, поэтому он должен быть передан получателю неизменно.

Первая проблема решается путём передачи всех атрибутов получателю. Для этого рекомендуется использовать определённый алгоритм отправки данных:

1) Отправляем размер файла в некой неизменной структуре. Определим, что через нашу программу можно передавать файлы, размер которых не превышает максимального значения переменной типа unsigned int, т. е 4294967295 байт. При этом, чтобы предать размер файла достаточно отправить получателю только 4 байта (размер unsigned int). Получатель должен принять первые четыре байта в качестве размера файла.
2) Отправляем имя файла вместе с расширением в некой неизменной структуре. Определим, что через нашу программу можно передавать файлы, имена которых не превышают 64 байт. При этом, чтобы предать размер файла, необходимо передать все 64 байта, не зависимо от длины реального имени. Получатель должен принять последующие 64 байта в качестве имени файла.

3) Отправляем сам файл. Получатель, уже зная размер файла, может определить, когда закончиться передача, и после этого сохранить файл в локальном каталоге.

Решение второй проблемы зависит от используемого транспортного протокола. При передаче с помощью потоковых сокетов данная проблема не возникает, так как используемый транспортный протокол TCP гарантирует доставку. При передаче с помощью датаграмных сокетов необходимо удостовериться, что вся информация дошла до получателя. Для этого необходимо разработать некий протокол обмена данными, схема которого изображена на рисунке 2.1.

Рисунок 2.1 Диаграмма взаимодействия клиента и сервера (датаграмные сокеты)

Если сервер не получил запрошенный фрагмент в определённое время, т.е. запрос или ответ не дошли до адресатов, сервер должен снова запросить требуемый фрагмент.

2.4. Контрольные вопросы и задания

1) Объясните разницу между передачей текстового сообщения и файла по сети.
2) Нарисуйте диаграмму взаимодействия клиента и сервера для передачи файлов при использовании датаграмных сокетов.
3) Объясните разницу в скорости передачи данных по протоколу TCP и UDP.