Лабораторные работы к курсу
«Сетевое программное обеспечение»

Лабораторная работа № 1

На выполнение Лабораторной работы 1 дается 3 недели.
Срок –Первая неделя октября.

Тема: Сервер на основе UDP протокола. Поддержка подтверждения получения информации.

Задание:

1. Написать клиент-серверную программу на основе транспортного протокола UDP [1]. Реализовать: подтверждение приема для каждой датаграммы, с целью сохранения целостности всей информации (см. особенности протокола TCP). Клиент передает файл или сообщение (несколькими датаграммами), сервер принимает.
2. Продемонстрировать реализованные возможности программ согласно заданию, при одновременной передачи файлов от нескольких клиентов к серверу. Например, при запуске сервера, указать - какие пакеты и сколько раз будут потеряны. Результат правильности приема выводить на экран.
3. Реализация - на языке C/C++, консольные приложения.

Серверная программа должна находить номер свободного порта и выводить его на экран (см., например, [1], стр. 338-342: обратите внимание на функции bind, getsockname). При запуске клиентской программы задавать со строки IP адрес сервера и порт.

Полезные ссылки:

1. Фейт С. TCP/IP: Архитектура, протоколы, реализация (включая IP версии 6 и IP Security). – М.: Лори, 2000. – 424 с.
2. http://rsdn.ru/article/unix/sockets.xml .

Лабораторная работа № 2

На выполнение Лабораторной работы 2 дается 4 недели.
Срок – Последняя неделя октября.

Тема: Параллельный (мультипроцессный) сервер.

Задание:

1. Написать программу обеспечивающую параллельную работу севера, принимающего информацию от клиента по сети. Условие: мультипроцессная организация на основе функции **fork**, транспортный протокол – TCP [1-3].

   Обеспечить в сервере завершение «зомби-процессов» !!!
2. Написать клиентскую программу, передающую заданное число ***i*** в цикле (определенное число раз с задержкой в ***i*** сек) на сервер. Соответствующий процесс сервера выводит полученную информацию на экран.
3. Реализация - на языке C/C++, консольные приложения.

Серверная программа должна находить номер свободного порта и выводить его на экран (см. [1-3], стр. 338-342, функции bind, getsockname). При запуске клиентской программы задавать со строки IP адрес сервера и порт.

Полезные ссылки:

1. Фейт С. TCP/IP: Архитектура, протоколы, реализация (включая IP версии 6 и IP Security). – М.: Лори, 2000. – 424 с.
2. Павский К. В., Ефимов А. В. Разработка сетевых приложений (протоколы TCP/IP, клиент-сервер, PCAP, Boost.ASIO): Учебное пособие / Сибирский государственный университет телекоммуникаций и информатики. – Новосибирск, 2018. – 80 с.
3. Протоколы TCP/IP и разработка сетевых приложений: учеб. пособие / К.В. Павский ; Сиб. гос. ун-т телекоммуникаций и информатики. - Новосибирск: СибГУТИ, 2013. – 130с.

Лабораторная работа № 3

На выполнение Лабораторной работы 3 дается 4 недели.
Срок – Последняя неделя ноября.

Вариант 1

Тема: Мультипротокольный сервер.

Задание:

1. Написать программу (на языке C/C++/Java, консольное приложение), реализующую работу сервера по двум протоколам TCP и UDP с помощью системного вызова select.
2. Для TCP протокола сервер создает поток (Thread) для обработки каждого клиента. Клиент передает текстовые сообщения из нескольких слов или предложений. Информацию, получаемую от клиентов сохранять в одном общем файле и при этом сохранять целостность сообщений (использовать mutex).
3. Для UDP протокола реализовать: подтверждение приема для каждой датаграммы, т.е. сохранение целостности всей информации клиентов. Клиент передает файл или сообщение (несколькими датаграммами), сервер принимает. Продемонстрировать реализованные возможности при одновременной передачи файлов от нескольких UDP клиентов к серверу. Например, при запуске сервера указать - какие пакеты и сколько раз будут потеряны. Результат правильности приема выводить на экран.
4. Реализация - на языке C/C++, консольные приложения.

Серверная программа должна находить номер свободного порта и выводить его на экран (см., например, [1], стр. 338-342: функции bind, getsockname). При запуске клиентской программы задавать со строки IP адрес сервера и порт.

Полезные ссылки:

1. Фейт С. TCP/IP: Архитектура, протоколы, реализация (включая IP версии 6 и IP Security). – М.: Лори, 2000. – 424 с.

Вариант 2

Тема: Создание клиент-серверного приложения на базе транспортного протокола SCTP.

Задание:

Требуется разработать клиент-серверные программы (на языке C/C++, консольные приложения) передачи данных на базе транспортного протокола SCTP [1, 2]. Обеспечить передачу информации по нескольким потокам в одной ассоциации (особенность протокола SCTP).

1. Клиентская программа посылает в одном соединении или ассоциации: текстовое сообщение в первом потоке; файл с картинкой во втором потоке.

2. Серверная программа принимает в одной ассоциации от каждого из клиентов: текстовое сообщение с первого потока и выводит на экран; со второго потока принимает данные и сохраняет в файл.

3. В серверной программе реализовать (на выбор) параллельную или псевдопараллельную обработку клиентов.

Серверная программа должна находить номер свободного порта и выводить его на экран (см., например, [3], стр. 338-342: функции bind, getsockname). При запуске клиентской программы задавать со строки IP адрес сервера и порт.

Полезные ссылки:

1. Протокол SCTP http://rfc.com.ru/rfc2960.htm
2. Стивенс У.Р., Феннер Б., Рудофф Э. M.UNIX: разработка сетевых приложений. - 3-е изд. - СПб. : ПИТЕР, 2007. - 1038с.
3. Фейт С. TCP/IP: Архитектура, протоколы, реализация (включая IP версии 6 и IP Security). – М.: Лори, 2000. – 424 с.

Вариант 3

Тема: Разработка клиент-серверной программы на основе сетевой библиотеки Boost.Asio.

Задание:

Используя сетевую библиотеку Boost.Asio разработать программы асинхронного сервера, клиента на протоколе TCP [1]. Реализация - на языке C/C++, консольные приложения.

Замечание. Серверная программа должна находить номер свободного порта и выводить его на экран. При запуске клиентской программы задавать со строки IP адрес сервера и порт.

Реализация клиент-серверной программы на основе сетевой библиотеки Boost.Asio выполняет нижеследующие действия.

Вариант первый.

Клиент отправляет текстовое сообщение. В ответе сервер возвращает преобразованное сообщение, в котором каждое слово «перевернуто» относительно исходного, но сохранен порядок их расположения.

Вариант второй.

При запуске клиента пользователь задает число *i* от 1 до 10. Клиент передает серверу в цикле это число с задержкой в *i* секунд между передачей. Сервер возвращает клиенту квадрат числа *i*.

Например:
1-й клиент посылает число 1 в цикле с задержкой в 1 сек.
2-ой клиент посылает число 2 с задержкой в 2 сек.
3-й клиент посылает число 3 в цикле с задержкой в 3 сек.

Сервер отображает информацию полученную от клиентов. Если у Вас правильно организован асинхронный ввод/вывод, то на экран со стороны сервера будут выводиться, с чередованием, числа 1, 2 и 3. Причем частота появления определенного числа будет зависеть от задержки по времени его передачи.

Полезная ссылка:

1. Boost.Asio http://habrahabr.ru/post/192284/