infotecs
INTERN

ТЕСТОВОЕ ЗАДАНИЕ

для стажера по направлению

Программист-исследователь

Часть 1

Разработать клиент-серверное приложение TCP/IPv4 (в виде одного исполняемого файла, работающего в одном из режимов: клиент или сервер).

1. Использовать epoll и неблокирующие сокеты (при этом приложение может быть однопоточным).

2. Сервер должен ожидать подключения, читать данные, дожидаться закрытия соединения и продолжать свою работу.

3. Клиент должен поддерживать заданное пользователем количество параллельных соединений к серверу без ограничения времени. Каждое соединение должно выполнить подключение к серверу, передачу случайного количества произвольных данных и закрытие. После закрытия должно создаваться новое соединение, и весь алгоритм выполняться для него заново – чтобы общее число параллельных соединений было постоянным.

4. Штатным средством завершения работы приложения должен быть Ctrl-C.

5. Пример запуска клиента для генерации 512 параллельных соединений с зерном 1337 для инициализации генератора случайных чисел:

./gen-app --addr localhost:8000 --mode client --connections 512 --seed 1337

6. Пример запуска сервера:

./gen-app --addr localhost:8000 --mode server

Часть 2

Разработать приложение, которое на основе библиотеки libpcap будет выполнять прослушивание указанного пользователем интерфейса (например, lo) и собирать в реальном времени статистику по TCP/IPv4-соединениям.

1. Из каждого прочитанного с помощью libpcap пакета выделить 4-tuple: IP-адреса источника и назначения, TCP-порты источника и назначения. 4-tuple – уникальный идентификатор потока данных. Будем считать, что потоки (IP1, IP2, Port1, Port2) и (IP2, IP1, Port2, Port1) разные (т.е. одно соединение – это два разных потока).

2. На основе 4-tuple отнести пакет к потоку.

3. Для каждого потока рассчитать: средний размер пакета на уровне Ethernet (включая заголовок Ethernet), количество переданных байтов в полезной нагрузке TCP, среднюю скорость передачи данных.

4. В режиме реального времени раз в одну секунду выводить на экран ТОП-10 потоков по скорости передачи данных.

5. Штатным средством завершения работы приложения должен быть Ctrl-C.

6. Протестировать работу приложения как минимум на основе приложения из Части 1.

Необязательно, но будет большим плюсом:

1. Юнит-тесты с помощью Google Test Framework.

2. Сделать приложение многопоточным.

3. Использовать lockfree структуры данных.

4. Учесть на уровне TCP возможность дропов, пересылок и дублирований сегментов (может быть сложно).

ТРЕБОВАНИЯ К ПРИСЫЛАЕМЫМ РЕШЕНИЯМ

- Использовать C++ стандарта 20 или выше.
- Использовать CMake в качестве системы сборки.
- Приложения должны быть рассчитаны для работы в среде GNU/Linux x64 (например, Debian 11 или новее).
- Не использовать стороннего кода, кроме стандартной библиотеки C++ и libpcap (а еще возможно gtest).
- Код должен быть выполнен в ООП стиле. Обратите особое внимание на архитектуру, стиль кода и оптимизацию кода.
- Готовое решение, содержащее только исходные коды (включая необходимые файлы CMake), должно быть передано в zip-архиве.

Максимальное время на выполнение задания

1 неделя