Полное руководство по сетевому программированию для разработчиков игр. Часть 4. TCP

Автор: x84

Потоковая передача данных

Зачем слать письма, когда есть телефон?

TCP очень сильно отличается от UDP. Чтобы понять как все устроено, необходимо углубиться в детали реализации этого протокола, что мы сейчас и проделаем...

Как можно было заметить, UDP напоминает почту. То есть данные передаются в пакетах (конвертах) целиком и полностью. Один пакет == одно письмо. Это позволяет нам не думать об управлении соединением, потому что его попросту нет. Также нет деления на "обслуживающих" и "обслуживаемых" - то есть все равны. Передача по UDP происходит без установления канала связи, то есть это "неподключаемый" протокол (connectionless). Наша задача - выбросить пакет в сеть, дальше им займутся другие компьютеры, маршрутизаторы и передающие устройства, то есть наш пакет обладает автономностью. Получается, что он как бы сам находит дорогу до получателя. Нас даже не волнует состояние адресата на данный момент. Допустим, компьютер получателя выключен, тогда добравшись до сегмента сети, в котором находится получатель, пакет будет уничтожен. И мы не сможем повлиять на это.

В TCP все сложнее - этот протокол подключаемый. То есть для того, чтобы осуществить пересылку данных, нам необходимо установить канал связи между двумя компьютерами. Это напоминает телефон. Сначала мы дозваниваемся до абонента, потом обмениваемся данными, затем вешаем трубку. В UDP мы можем послать данные

("Вас неслышно! Говорите громче!!"). Опять же, в TCP мы должны быть вежливыми и соблюдать правила хорошего тона - отправлять запрос на закрытие соединения. Если другой стороне больше нечего нам сказать, то соединение будет закрыто. Если же еще есть какие-то данные, требующие отправки - то сначала эти данные будут отосланы и уже затем соединение будет закрыто. Эта схема взаимодействия позволяет мягко закрывать соединение.

Зачем нам это нужно? Зачем нам нужна такая надежность? Очень просто: представим себе какое-нибудь бизнес-приложение. Есть такие программы, от корректного выполнения которых напрямую зависит достаток тех людей, на чьих компьютерах они выполняются.

Возьмем самый простой пример. Есть два банка. Человеку (клиенту одного из банков) надо перечислить деньги со своего счета на счет фирмы (в другом банке), чтобы оплатить какие-то услуги. Вот ситуация. Мы отправляем пакет от имени первого банка второму, в котором говорится "Банк вычтет сумму в 100 рублей со счета клиента - ваша задача прибавить 100 рублей на счет фирмы". Пакет отослан. Банк клиента вычел из его денежных средств 100 рублей. Однако, если пакет затерялся в пути, то банк фирмы не получит нужного уведомления, соответственно клиент останется и без денег и без услуг. Конечно, этот пример утрирован, но общий смысл проблемы, думаю, ясен. Мы нуждаемся в подтверждении доставки. TCP поможет нам в этом.

Итак, раз уж TCP подключаемый протокол, то это автоматически означает, что соединение делится на две стороны: тот, кто отправляет запрос на подключение (набирает номер) и тот, кто принимает его (поднимает трубку в ответ на звонок). Это и есть зачатки той системы, которую принято называть "клиент-сервер". Есть "обслуживаемый" (клиент) и "обслуживающий"(сервер). Сервер должен подготовиться к приему запросов на подключение и начать прослушивать сокет на наличие этих запросов, в то время как клиент должен инициировать соединение, отправив запрос на подключение.

Посмотрим, как происходит сеанс связи по протоколу TCP.

1. Сначала сервер должен создать TCP-сокет и начать его прослушивание.

2. Клиент должен первым вступить в контакт - отослать запрос.

3. Сервер должен этот запрос принять (или отвергнуть - по желанию) и отослать клиенту подтверждение.

4. Если ответа от сервера не приходит, то клиент считает попытку соединения неудачной и либо завершает свою деятельность, либо пытается "прозвониться" позже. Если ответ от сервера все-таки пришел, то соединение считается установленным. После этого данные между клиентом и сервером могут передаваться в обе стороны.

5. Когда все необходимые данные доставлены, соединение мягко закрывается.

половина - для клиента, другая - для сервера.

Страницы: **1** 2 3 4 Следующая »

#OSI, #TCP, #UDP, #клиент, #сервер, #сокеты

1 ноября 2003 (Обновление: 18 ноя 2009)

Комментарии [40]