Гавриков А.П.
Научный руководитель: к.т.н. Бейлекчи Д.В.
Муромский институт (филиал) федерального государственного образовательного учреждения высшего образования «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»
602264, г. Муром, Владимирская обл., ул. Орловская, 23
E-mail: kaf-eivt@yandex.ru

Разработка протокола на основе UDP для клиент-серверного или децентрализованного приложения

UDP был создан в 1980 году Дэвидом П. Ридом. UDP расшифровывается как User Datagram Protocol, протокол пользовательских датаграмм. UDP использует простую модель передачи данных, без ответных сообщений об успешном соединении, ошибок, упорядочивания данных или целостности данных.

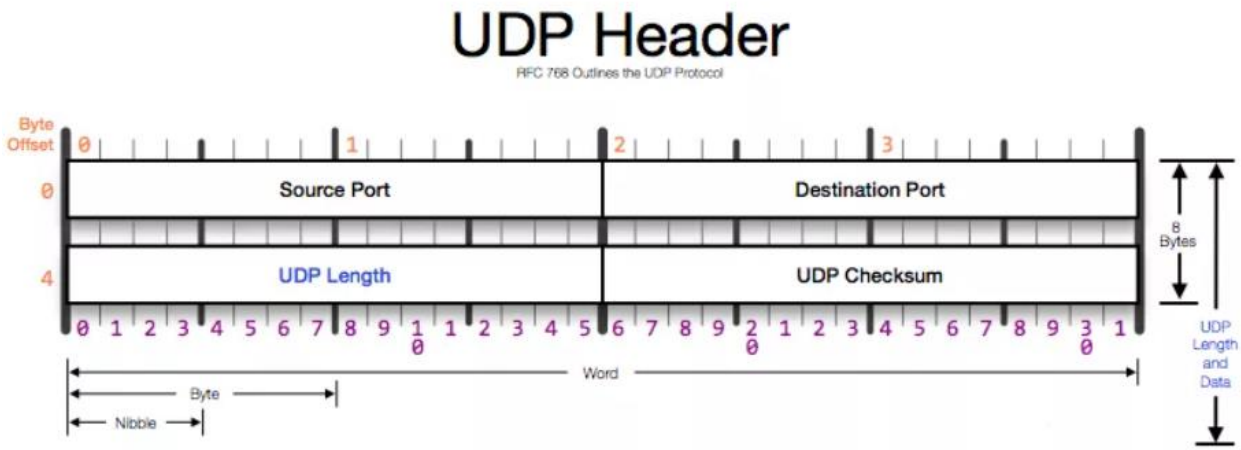

Рис. 1. Заголовок UDP

На основе UDP можно разработать собственный протокол передачи данных, в котором будут учтены специфичные особенности приложения, оборудования или данных. Так на основе UDP разработаны такие протоколы как DNS (Система доменных имён), SNMP (Простой протокол управления сетями), RIP (Протокол маршрутной информации), DHCP (Протокол динамической конфигурации узла).

Рассмотрим пример разработки собственного протокола на основе UDP для приложения обмена сообщениями. Для начала рассмотрим сколько места может занимать один UDP-пакет. При работе через сеть IPv4, фактический предел пакета составляет 65507 байт, 8 байт уходит на UDP-заголовок и ещё 20 байт на IP-заголовок. Также нужно учитывать MTU (максимальную единицу передачи данных), который составляет для Ethernet около 1500 байт. Также если учесть другие нюансы, например, в RFC 791 указывается минимальная длина IP пакета 576 байт, которую должны поддерживать все участники IPv4, то, для того чтобы быть уверенным, что пакет будет принят любым хостом, размер данных в UDP не должен превышать 508 байт.

Ещё существует вариант, что данные, которые нужно передать не умещаются в один пакет. Тогда нужно будет разбивать их на несколько частей, в таком случае можно помечать номер пакета, отправлять их все и ждать от получателя ответное сообщение, в котором будут указаны номера полученных из чего можно сделать вывод об не доставленных пакетах, а если вообще ответного сообщения нет, то пересылать снова все пакеты, либо проверять соединение каким-то тестовым пакетом.

Определившись с размером пакета, размером занимаемых в них данных и форматом передаваемого пакета, можно продумать как приложение будет работать. Есть много вариантов работы с сетью, например, клиент-серверный подход или децентрализованная сеть. У этих подходов есть свои преимущества и недостатки.

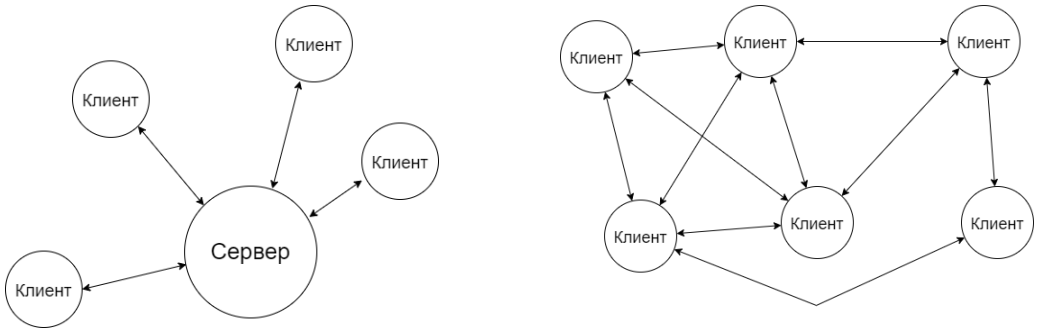

Рис. 2 — Визуальное отображение работы сети (слева клиент-серверный подход, справа децентрализованный подход)

При реализации клиент-серверного подхода можно будет разделить клиентскую и серверную программы, например, создав их на разных технологиях, тем самым ускорив и упростив разработку. Также уменьшиться нагрузка на клиентскую часть, так как она будет заниматься только получением и отправкой на сервер данных, а также сервер проще защитить от взлома и кражи данных. Но в то же время требования к мощности сервера будет возрастать, если сервер не справиться с нагрузкой и отключиться от сети, то вся сеть окажется неработоспособной.

Так для реализации программы обмена сообщениями нам необходимо написать две программы одну для клиента, другую для сервера. Так как на клиенте будет лишь принятие и отправка сообщений через сервер, то написать это приложение можно на кросс платформенных технологиях, которые не дают максимальную производительность, но ускоряют разработку и упрощают перенос на другую операционную систему. Серверную же часть нужно будет писать с учётом большой нагрузки, так как сервер будет обрабатывать каждого подключенного клиента.

Если представить работу сети, то выглядеть она будет так. Клиент связывается с сервером и получает возможность писать от своего имени, а также список находящихся в сети. Написав сообщение, он отправляет пакет с ним на сервер указывая себя как отправителя и кому нужно его доставить. Сервер, получив пакет с данными, смотрит может ли он отправить адресату, если нет, то сохраняет его и ждёт возможности отправить его, иначе отправляет его. А также если нужно, то на сервере можно хранить копии всех переписок всех клиентов. Также нужно учитывать, что во время ожидания сервер должен обрабатывать и сообщения от других участников.

При реализации децентрализованного подхода разделение на клиента и сервера не происходит, так как в такой сети каждый пользователь является равноправным её членом. В отличие от клиент-серверного подхода устройства взаимодействуют напрямую друг с другом. Главным преимуществом такой сети является практически полная независимость работоспособности от её размера. Так в сети может быть два устройства и сеть будет полностью исправно выполнять свои обязанности. К недостаткам можно отнести небезопасность сети, так как устройство в момент работы будет доступно всем участникам сети.

Если представить, что в приложении используется этот подход, то работа сети будет выглядеть следующим образом. При запуске программы на одном из устройств, он проверит сеть на возможность подключения к ней, на поступающие к нему пакеты и на наличие в сети новых и ранее работавших с ним устройств. Далее происходит стандартный цикл работы программы. Получаем от пользователя сообщение и кому его надо доставить, проверяем в сети ли он, если да, то отправляем, если нет, то запоминаем, что ему надо отправить сообщение и через какое-то время повторяем попытку. Во время работы нам может прийти сообщение, которое надо принять и отобразить пользователю.

В современном мире многим компаниям, где используются облачные технологии, приходиться реализовывать собственные протоколы ради увеличения безопасности и скорости передачи данных, а также учитывая особенности передаваемых данных. Так что опыт

разработки приложения работающего с сетью всегда будет актуальным и востребованным на рынке IT.

Литература

1. Баринов, В.В. Компьютерные сети: Учебник / В.В. Баринов, И.В. Баринов, А.В. Пролетарский. - М.: Academia, 2020. - 192 с.

2. Таненбаум, Э. Компьютерные сети / Э. Таненбаум. - СПб.: Питер, 2019. - 960 с.

3. Документ RFC 791 [Электронный ресурс] Режим доступа: https://www.ietf.org/rfc/rfc791.txt