CQ 22.

17 апреля 2023 г.

1 Какие уровни входят в модель OSI и за что они отвечают?

В модель OSI входят уровни:

- Физический: работа со средой и передача данных.
- Канальный: физическая адресация (с использованием MAC-адресов)
- Сетевой: логическая адресация.
- Транспортный: прямая передача данных и контроль этой передачи.
- Сеансовый: поддержка сеанса связи (позволяет приложениям взаимодействовать между собой длительное время).
- Представление: преобразование протоколов и кодирование/декодирование данных.
- Прикладной: обеспечение доступа к сетевым службам.

2 Перечислите основные особенности протоколов TCP и UDP?

Основные особенности протокола TCP:

- Надёжный: даёт гарантию, что сообщение будет доставлено сообщений; обрабатывает ошибки.
- Требует установки логического соединения (служебного).
- Поддерживает режим соединения один с одним (1-1).
- Потокоориентирован, данные представляют собой поток байт. (В этом плане похож на std::cin/cout: данные поступают и выдаются надлежащим – см. след. пункт – образом).
- Контролирует порядок в работе с данными и их передаче далее.

Основные особенности протокола UDP:

- Ненадёжный: нет гарантии доставки сообщений; нет возможности обрабатывать ошибки.
- Не требует установки логического соединения.

- Поддерживает как режим соединения один с одним (1-1), так и один со многими (1-N).
- Ориентирован на сообщения (вывод данных нуждается в дополнительной обработке).

Протокол UDP используется, когда есть необходимость в более быстрой работе с данными и не страшно потерять малую информацию, получив целостное представление обо всём остальном. Когда же ограничений по времени нет, то используется протокол TCP.

3 Какие данные необходимы для сетевого взаимодействия?

Для организации сетевого взаимодействия необходимо обеспечить подключение клиента к серверу (установить соединение), взаимную отправку и получение данных.

Сетевое взаимодействие организуется на основании некоторого протокола, поэтому необходимо определить какой будет использоваться протокол.

Далее, чтобы подключить клиента к какому-либо серверу, используя какой-либо протокол, необходимо знать IP-адрес хоста клиента и IP-адрес хоста сервера (т.е. получателя), например, в IPv4 или IPv6 версии, а также используемые ими номера портов. Причём не все номера портов могут быть задействованы произвольным образом (скорее всего, будут использоваться пользовательские или динамические). Кроме того, IP-адреса могут быть определны по DNS-именам хостов клиента и сервера (соответсвенно, требуется знать либо сам IP-адрес, либо его DNS-имя).

Пара из IP-адреса и номера порта формирует т.н. endpoint (конечную точку). Эти конечные точки и используются для подключения клиента к серверу, а также для принятия самим сервером любых или конкретных подключений. Это уже осуществляется с помощью т.н. sockets (сокетов) – интерфейсов для обеспечения обмена данными (отправки/получения сообщений) между процессами – при связывании сокета с конечной точкой.

Итак, необходимы данные об:

- используемом протоколе;
- используемых IP-адресах (хостов клиента и сервера);
- используемых портах (хостов клиента и сервера).

4 Какие функции выполняют пассивные и активные сокеты?

Сокеты, как уже упоминалось, являются интерфейсами для обеспечения обмена данными между процессами (в нашем случае между клиентом и сервером), поэтому они решают задачи:

- установки соединения;
- обмена данными.

Но в действительности, если сокет выполняет обе задачи, то его называют активным. Если же сокет решает лишь задачу установки соединения, то он называется пассивным. Пассивный сокет бывает удобно использовать для установки соединения именно на сервере (причём в рамках протокола TCP), когда требуется подключить произвольного клиента (просто ожидается подключение любого клиента в любой момент). Затем полученное соединение обычно перенаправляется на активный сокет. И т.о., активные сокеты клиента и сервера оказываются связанными: соединение установлено (и можно обмениваться данными, используя уже активные сокеты).

Стоит отметить, что сокеты в Boost.Asio не являются потокобезопасными.

5 Как устанавливается логическое соединение протокола TCP?

Прежде чем клиент попытается подключиться к серверу, сервер должен сначала подключиться к порту и прослушивать его, чтобы открыть его для соединений (пассивное открытие – см. пред. вопрос). Как только пассивное открытие установлено, клиент может установить соединение, инициировав активное открытие с помощью трёхэтапного рукопожатия (handshake). Приведём формальное описание (описание на основании Boost.Asio см. в конце вопроса – можно сравнить):

- Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN:
  - Сервер получает сегмент, запоминает номер последовательности и пытается создать сокет для обслуживания нового клиента.
  - В случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK, и переходит в состояние SYN-RECEIVED.
  - В случае неудачи сервер посылает клиенту сегмент с флагом RST
- Если клиент получает сегмент с флагом SYN, то он запоминает номер последовательности и посылает сегмент с флагом ACK:
  - Если клиент одновременно получает и флаг ACK (что обычно и происходит), то он переходит в состояние ESTABLISHED.
  - Если клиент получает сегмент с флагом RST, то он прекращает попытки соединиться.
  - Если клиент не получает ответа в течение некоторого времени, то он повторяет процесс соединения заново.
- Если сервер в состоянии SYN-RECEIVED получает сегмент с флагом ACK, то он переходит в состояние ESTABLISHED.
  - В противном случае после тайм-аута он закрывает сокет и переходит в состояние CLOSED

Шаги 1 и 2 устанавливают и подтверждают порядковый номер для одного направления. Шаги 2 и 3 устанавливают и подтверждают порядковый номер для другого направления. После выполнения этих шагов и клиент, и сервер получают подтверждения, и устанавливается двусторонняя связь.

Установку логического соединения протокола TCP можно описать более просто, используя средства высокоуровневой обёртки Boost.Asio. В принципе после установки клиентом пассивного открытия клиент также устанавливает двустороннее соединение с сервером с помощью метода connect() клиентского сокета:

- Устанавливается пассивная привязка клиентского сокета к серверу (использован метод bind() пассивного серверного сокета).
- Устанавливается прослушивание по заданному порту сервера (к которому обращается клиент) с ограничением возможного количества клиентских подключений (метод listen() пассивного сокета).

- Создаётся активный серверный сокет и далее с помощью метода accept() (пассивного сокета, с помощью которого и выполнялась исходная привязка) устанавливается конечное соединение активных сокетов на стороне клиента и сервера: клиент получает ответ на вызов метода connect() своего сокета. В случае успеха связь установлена (помешать может, например, размер, указанный ранее при вызове метода listen()), может происходить обмен данными. Пассивный сокет сыграл роль посредника.

Список литературы

[1] Конспект. И.С. Макаров.

[2] https://ru.wikipedia.org

[3] https://en.wikipedia.org