

HTTP, сети

Среды где многопоточность появляется сама это WEB server. На каждый запрос пользователя выделяется поток.

Стек знания сетей у java developer:

1. Model OSI
2. TCP/IP стек
3. IP
4. TCP, UDP
5. HTTP, DNS

Сети бывают с коммутацией каналов и с коммутацией пакетов:

1. С коммутацией каналов: Личный канал на данный момент вам принадлежит но дальше может принадлежать кому угодно. Пример раньше звонок по телефону переключали коммутатор и можно было разговаривать по телефону. “Мне в смольный переключите!”
2. С коммутацией пакетов: Режут страницу на куски и подают пакеты нарезаемые примерно по 10 килобайт трафика.

Передается по напряжению на всем проводе синусойдой.

В начале пакете содержится IP адрес, TCP, HTTP, и сообщение. Может занимать 1600 байт.

IP – сетевая карта.

TCP – реализованы на уровне операции

HTTP – Добавляет браузер. Get, Cookie

Нельзя написать хак ПО на Java. Как не пытайся при посылке IP пакета не поменять пакета. IP – JVM определяет.

В виде стека это выглядит model OSI.

Протокол IP отвечает за адреса IPv4 если глобально то всего 4млрд

IPv6 ----

TCP/IP это набор layer

IP – не дает точный адрес это просто идентификатор



Пример IP 200.200.200.200 и сеть настроена таким образом что бы определить куда дальше идти сигналу. После встречи с точкой в одну сторону или в другую. Но решение не может быть однозначным.

Пример коробка не отвечает и поэтому кинет на другой адрес.

TTL (Time to live) – время жизни пакета

Проблемы IP

- 1) Пакеты могут теряться, коробка может быть перегружена, странные шумы
- 2) Пакеты могут дублироваться

Латентность задержки

Офисы в разных точках направления. Была проблема

Если один офис в США другой в Сингапуре то слишком долгий запрос. Проблема: не смогли join таблицы потому что загрузка более 100 мм, а алгоритм располагает так что если запрос более 100 мм то ошибка, новый запрос. Тут скорость света решает.

Java.net socket – это TCP сокет, native method.

JNI java native interface- будет искать библиотеку она ее так подлинкует.

Мы можем кодить только нативно ниже TCP.

Java.net.ServerSocket- для java их сделали

Java.net.Socket – как input output Stream

Блокирующие методы – просто ожидает ответа.

IOException – канал, пакеты не дошли.

Input/Output блокирующие операцию

Input stream (byte) read 1 или -1 или IOEx

На сервер приходит запрос и мгновенно шлется ответ.

На сервере ограничение 100.000 сокетов, если сервер нормально держит, то на каждую 1000 новый сервер.

TCP-IP есть только тот кто первый послал пакет, давай поговорим тот является клиентом.

Процесс начала сеанса TCP (также называемый «рукопожатие», состоит из трёх шагов.

1. Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN.

- Сервер получает сегмент, запоминает номер последовательности и пытается создать сокет (буферы и управляющие структуры памяти) 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.

2. 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 10 секунд, то он повторяет процесс соединения заново.

3. 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.

Процесс называется «трёхэтапным согласованием» ([англ. three way handshake](#)), так как несмотря на то что возможен процесс установления соединения с использованием четырёх сегментов (SYN в сторону сервера, ACK в сторону клиента, SYN в сторону клиента, ACK в сторону сервера), на практике для экономии времени используется три сегмента.

Sniffer – может рассказать о всех пакетах, включая пакеты на низком уровне.


Тот кто ждет соединение называется `java.net.ServerSocket`

На уровне TCP и UDP вводится понятие порт что бы от Две конечные точки (хосты) при установке соединения по этим протоколам идентифицируются согласно номерам портов. Номера портов, используемые для конкретных специфических целей. Количество портов ограничено с учётом 16-битной адресации ($2^{16}=65536$, начало — «0»). Все порты разделены на три диапазона — *общезвестные* (или *системные*, 1023), *зарегистрированные* (или *пользовательские*, 1024—49151) и *динамические* (или *частные*, 49152—65535).

`Java.lang.ServerSocket` – это фабрика сокетов

`accept()`- блокирующая операция, потому что пассивно ждет соединения.

Блок/неблок API

Синхр. API 

- 1) Раньше был Apache -> 1.0 Блокирующие API => (IO), `java.lang`
- 2) Nginx -> 1.4.2 не блокирующие API => (NIO), `java.lang`

Асинхронное API => 3) NIO 2.0

Синхронное API это вызвал метод получил ответ, причина следствие.

Блокирующие API вызывается метод и останавливается на некоторое время, (пример `accept()` блокируется и ждет)

Неблокирующие это когда метод `accept()` может вернуть NULL/ Если вызвать Асепт а тебе не пришел возвращается null.

Асинхронное API – Я не ходил за почтой ко мне пришел почтальон. Приводит автоматически к многопоточности. Построено на `callbacks` – метод обратного вызова.

`Callbacks` механизм – при обратном вызове программист задает действия, которые должны выполняться всякий раз когда происходит некоторое событие. Создается интерфейс и класс его имплементирует. Пример `ASS.listener(Runnable)` <- шаблон `listener`

Если запрос на локальный хост 127.0.0.xx <- спускаются до сетевой карты, и обратно сразу.

DNS – система доменных имен, для получения информации о доменах. 13 главных серверов имеет.

DNS это как уровень HTTP который стоит поверх UDP, который как TCP.

Есть DNS сервер который хватает DNS

