

websocket

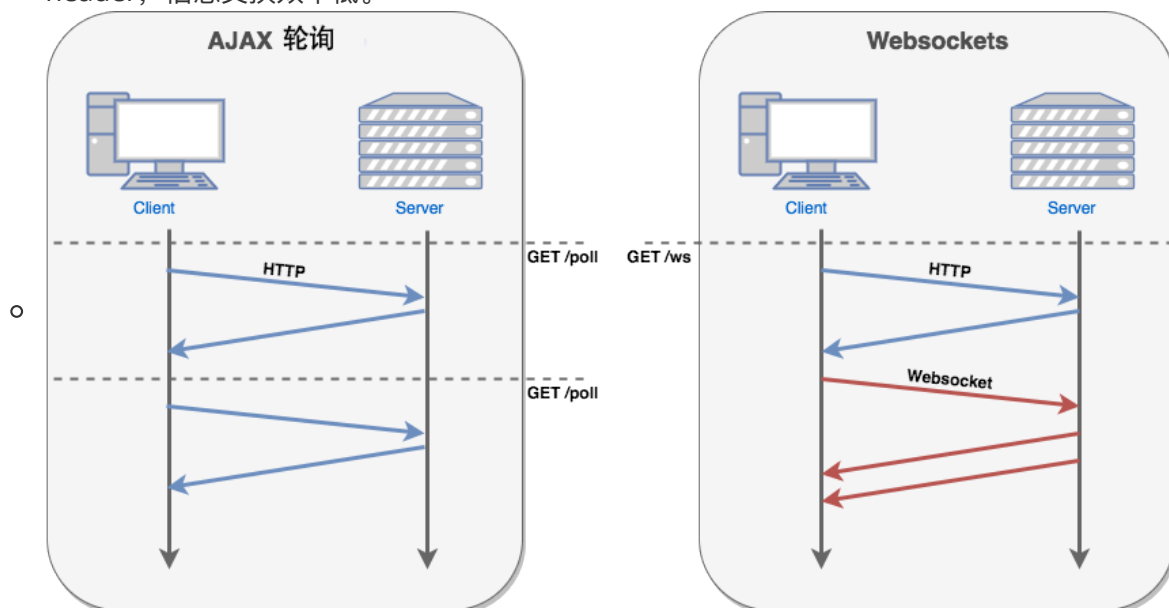
websocket借鉴了socket的思想（为web应用程序提供端到端的全双工通信）

1. 什么是websocket?

- 基于HTML5规范，是基于TCP的协议，由通信协议和编程API组成，实现浏览器与服务器全双工通信

2. 为什么需要websocket?

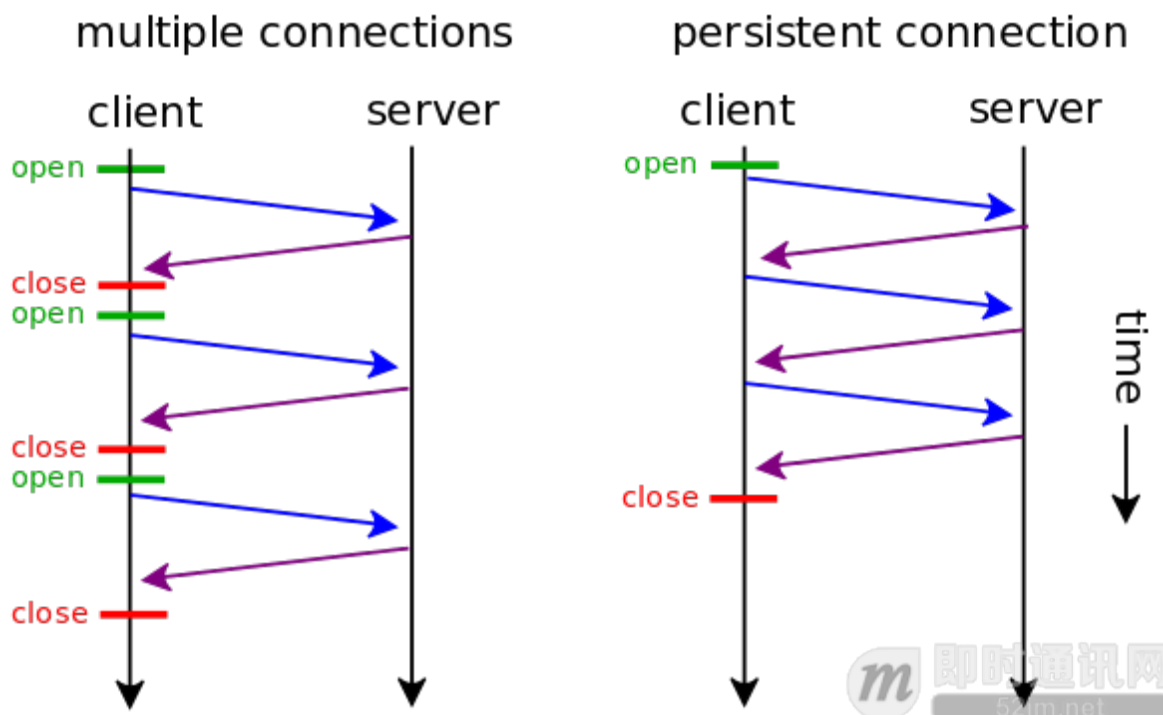
- 如何实现web端双向通信
 - 1) 轮询（polling）：轮询就会造成对网络和通信双方的资源的浪费，且非实时；
 - 2) 长轮询：客户端发送一个超时时间很长的Request，服务器hold住这个连接，在有新数据到达时返回Response，相比#1，占用的网络带宽少了，其他类似；
 - 3) 长连接：其实有些人对长连接的概念是模糊不清的，我这里讲的其实是HTTP的长连接（1）。如果你使用Socket来建立TCP的长连接（2），那么，这个长连接（2）跟我们这里要讨论的WebSocket是一样的，实际上TCP长连接就是WebSocket的基础，但是如果是HTTP的长连接，本质上还是Request/Response消息对，仍然会造成资源的浪费、实时性不强等问题。
- 由于http协议的被动性（只有client可以发起请求）
 - 轮询：ajax，long poll发送请求时，服务端可能没有更新数据，需要更多的资源，导致客户端无法接入，服务端一直正忙
 - keep-alive connection：一次TCP连接完成多个http请求，每个请求单独发header；轮询polling：客户端不断向服务器发送http请求查询是否有新数据。但是都要传递大量的http header，信息交换效率低。



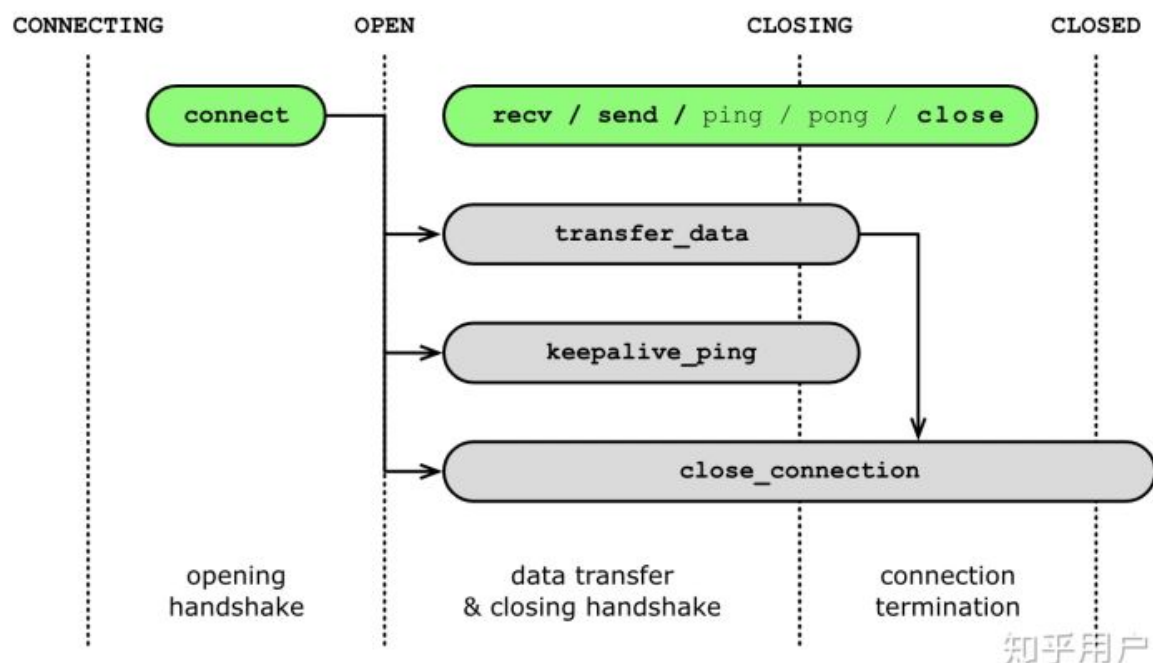
- 出现了websocket（server同样可以主动发送消息给client），补充http无法长连接的缺点。

3. websocket的通信原理和机制

- 利用http建立连接（兼容web浏览器），基于tcp实现长连接，定义了很多新的header域，所以传递信息时实际上没有用http。
- 请求响应模式：



- 流程



- 一旦连接建立以后，后续数据都以帧序列的形式传输。
- 1. 当客户端连接服务端时，会像服务端发送一个http报文，在头部告诉服务端将通信协议切换到websocket
- 2. 服务端如果支持websocket协议则切换协议成功，同时发给客户端一个响应报文头

- 以上通过http完成，称为websocket协议握手（websocket Protocol handshake）。建立以后使用的是websocket协议，实际是基于刚刚通过http连接的TCP进行数据传输（文本数据，二进制数据）。
- 3. 开始传递数据
- 4. 关闭连接

代码实例：

1. 下载websocket++

<https://github.com/zaphoyd/websocketpp>

依赖C++和boost

2. 配置依赖库

- boost: `brew install boost`
 - 路径 `/usr/local/Cellar/boost/1.72.0_2/lib/`
- scons: `brew install scons` (websocket++用scons编译，文件夹中有SConstruct)
 - 类似于cmake
 - 路径 `/usr/local/Cellar/scons/3.1.2_1/`
- openssl
 - `/usr/local/Cellar/openssl@1.1/1.1.1g/`

3. 运行websocket++中的examples/echo_client

- `g++ -std=c++11 -o test echo_client.cpp -I./ -I`
`/Users/mac/Documents/websocketpp-master/ -I`
`/usr/local/Cellar/boost/1.72.0_2/include/ -L`
`/usr/local/Cellar/boost/1.72.0_2/lib/`

- 连接成功

- websocket client

```
[MacdeMac-mini-2:echo_client mac$ ./test ws://127.0.0.1:8888  
[2020-05-06 18:00:19] [connect] Successful connection  
[2020-05-06 18:00:24] [fail] WebSocket Connection 127.0.0.1:8888 - "WebSocket++/0.  
8.2" / 0 websocketpp:22 The opening handshake timed out
```

- 本地server

```
tcpserver -- -bash -- 80x41
/usr/local/Cellar/scons/3.1.2_1/libexec/ (6 files)
/usr/local/Cellar/scons/3.1.2_1/share/man/ (3 files)
[MacdeMac-mini-2:tcpserver mac$ ls
Makefile          runServer.dSYM    server            server.h
runServer.cpp     runServer.o       server.cpp        server.o
[MacdeMac-mini-2:tcpserver mac$ make
g++ server.o runServer.o -o server -lpthread -std=c++11; ./server 8888 1
wait for guest
new client id:0new client ip:127.0.0.1new client sock:4new client msg:
Accepted
wait for guest
open client:0
recv的返回值: 128GET / HTTP/1.1
Connection: Upgrade
Host: 127.0.0.1:8888
Sec-WebSocket-Key: /Xl4OWXYXvq8e0Df3xKDlQ==
Sec-WebSocket-Version: 1
Thread done
id: 0
ip: 127.0.0.1
message: GET / HTTP/1.1
Connection: Upgrade
Host: 127.0.0.1:8888
Sec-WebSocket-Key: /Xl4OWXYXvq8e0Df3xKDlQ==
Sec-WebSocket-Version: 1
recv的返回值: 563
Upgrade: websocket
User-Agent: WebSocket++/0.8.2

Thread done
id: 0
ip: 127.0.0.1
message: 3
Upgrade: websocket
User-Agent: WebSocket++/0.8.2

18:0:1918:0:19

desc->id: 0desc->id: 0
```

bug汇总:

1. scons内python版本语言问题error
 - 修改文件python2.7修改至python3.8
2. 编译时boost环境找不到
 - 经查找os.environ发现os环境变量中没有boost
 - 手动添加BOOST_INCLUDE, BOOST_LIBS
3. fatal error:
 - 'openssl/conf.h' file not found
 - 同上解决方案
4. Implicit dependency /usr/local/Cellar/boost/1.72.0_2/lib/libboost_thread.a' not found, needed by target build/test/transport/asio/test_base_boost'.
 - 待解决