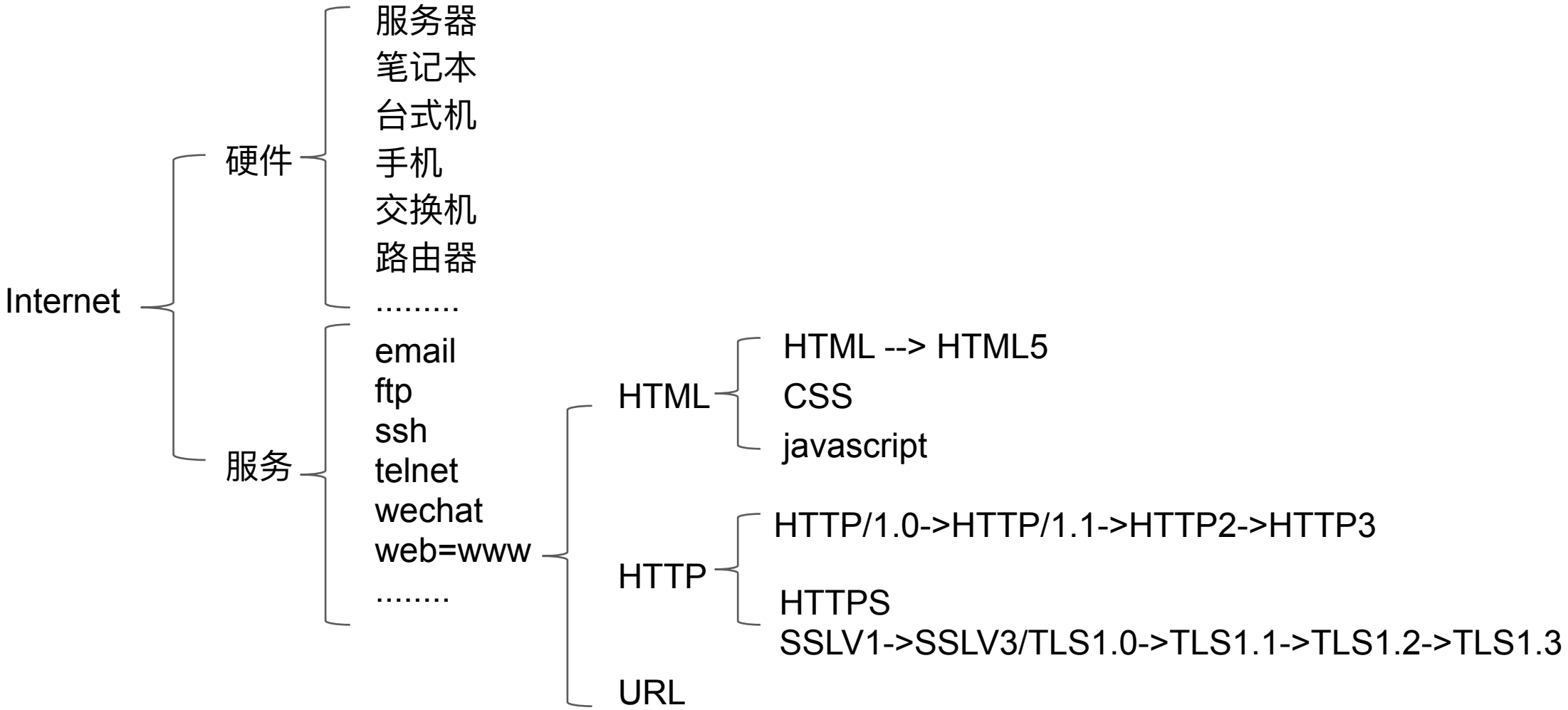
图解HTTP

--- 21张图把HTTP安排的明明白白

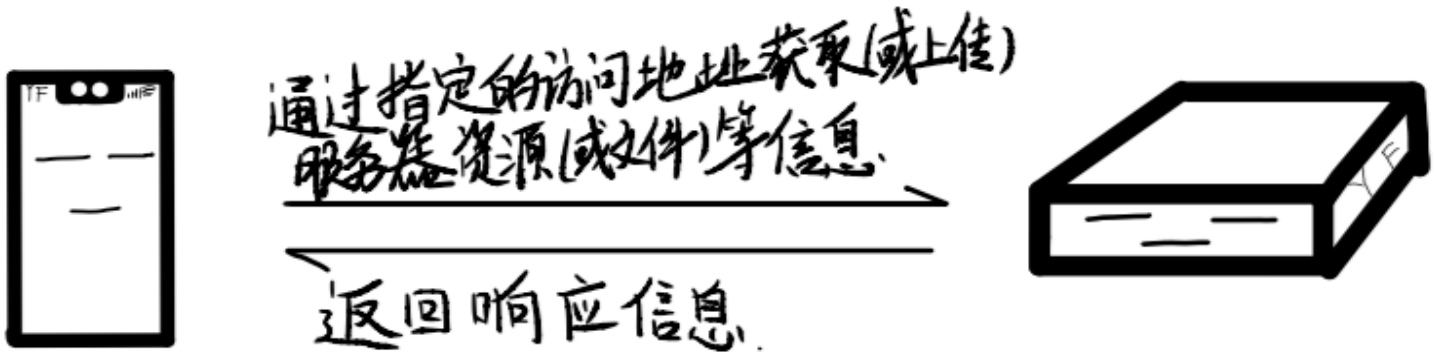
永福　2019-10

Internet和web是什么关系?



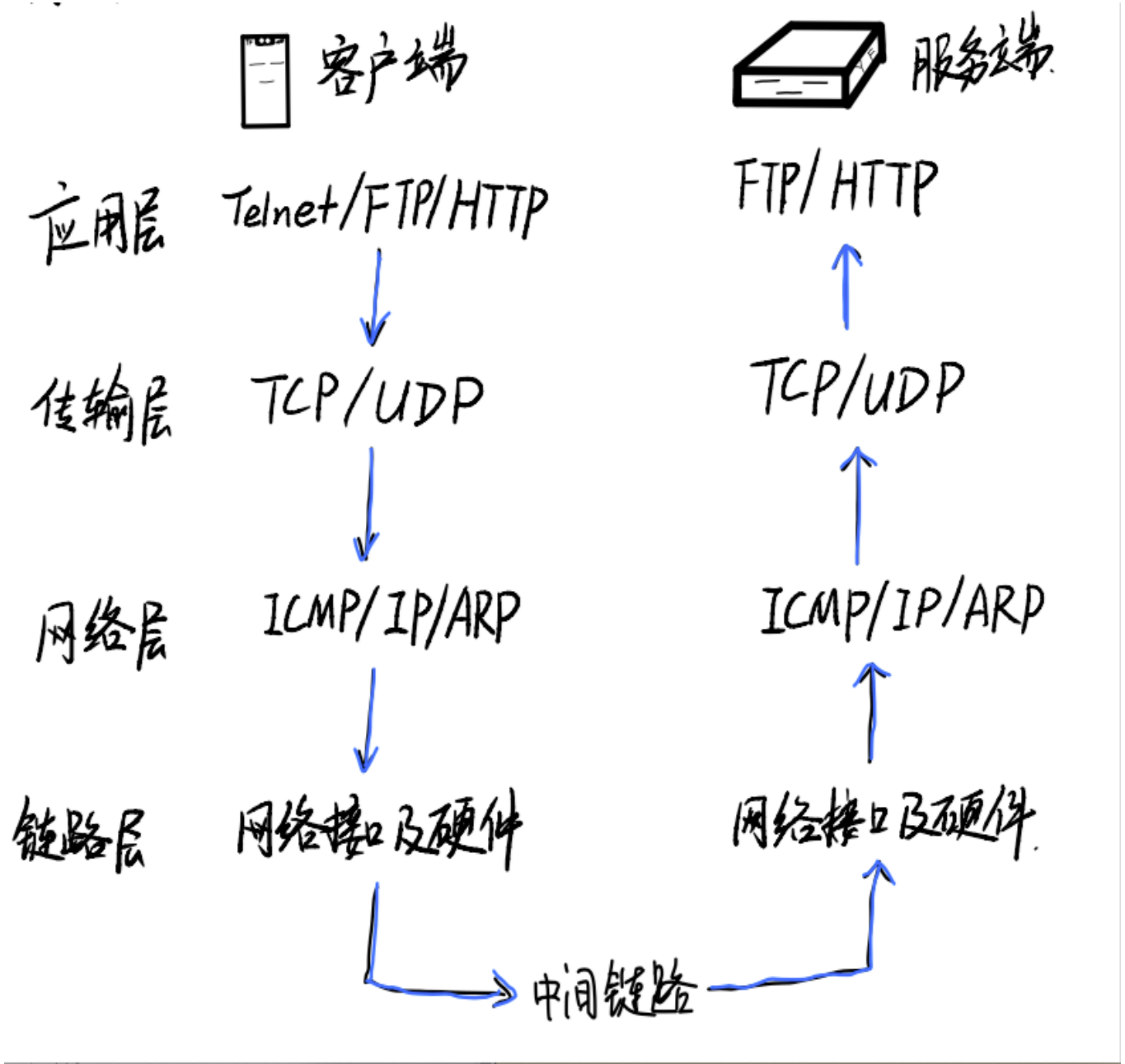
什么是HTTP?

- HTTP全称是：HyperText Transfer Protocol（超文本传输协议）
- 是一种基于TCP/IP通信协议来传递数据的网络传输协议。



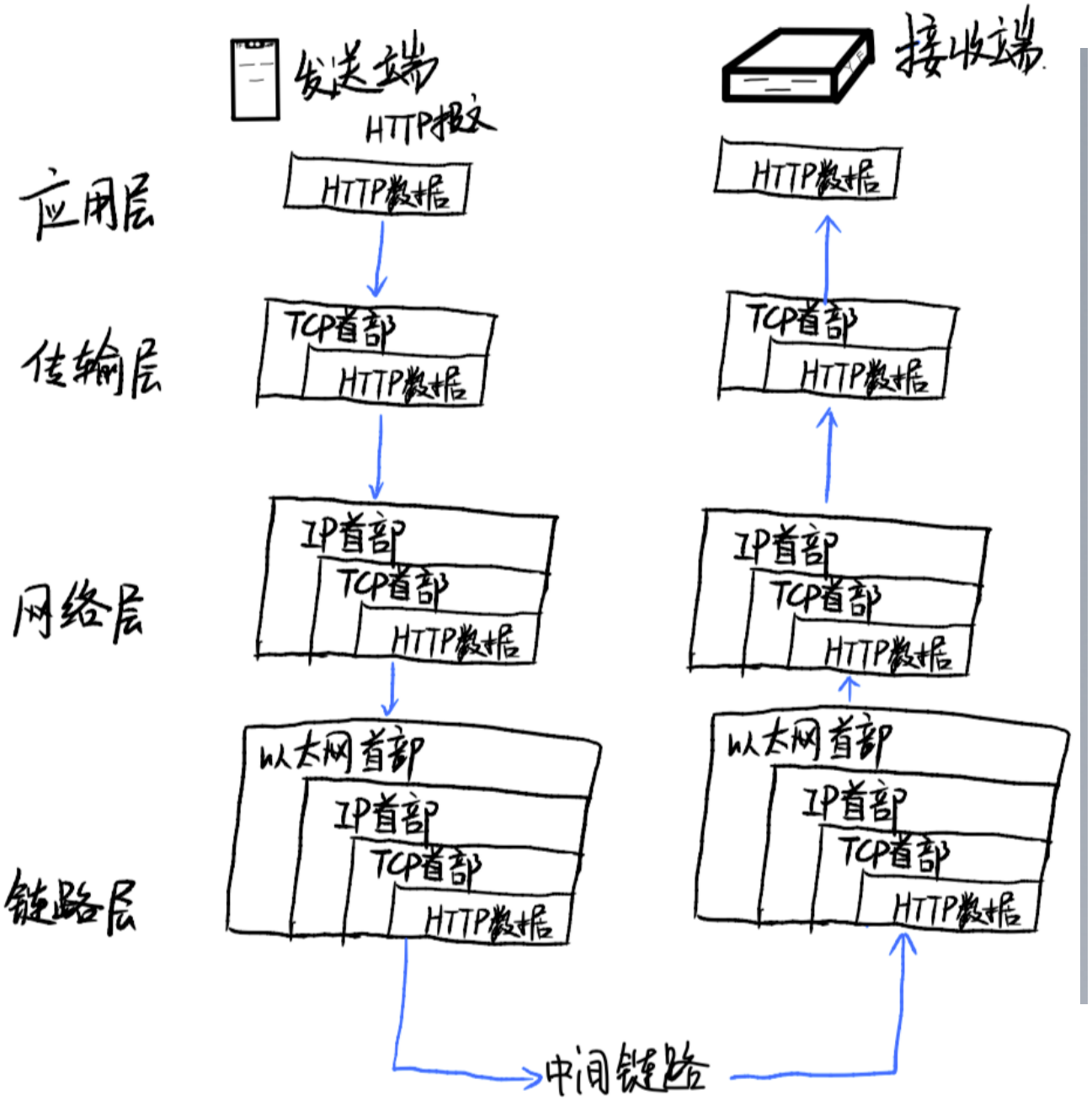
什么是TCP/IP?

- TCP/IP是指能够在多个不同网络间实现信息传输的协议簇。
- TCP/IP协议不仅仅指的是TCP和IP两个协议，而是指的一个由FTP,SMTP,TCP,UDP,IP,ARP等等协议构成的协议集合。
- 因TCP和IP最有代表性。

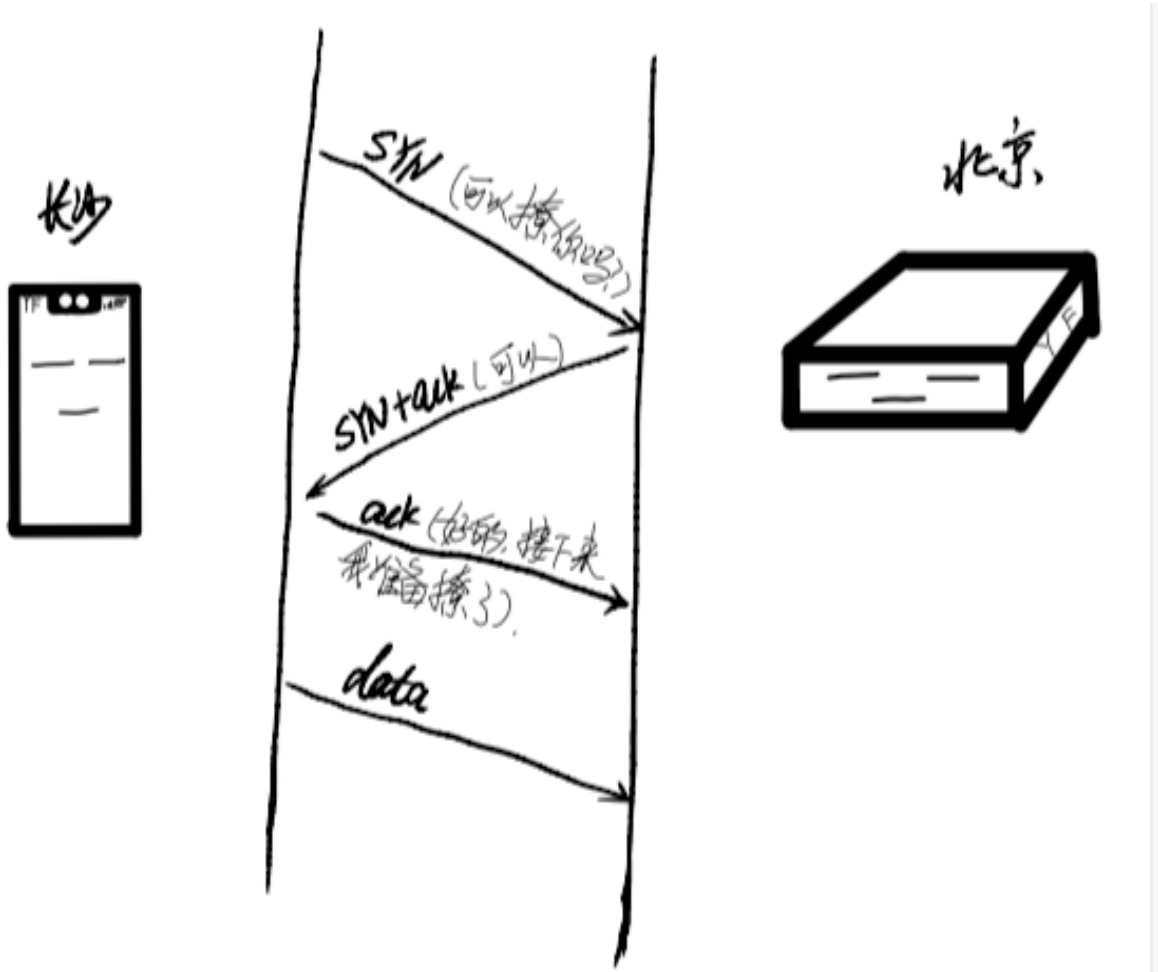


HTTP协议在TCP/IP协议中的数据传输

每一层中都会进行数据封装

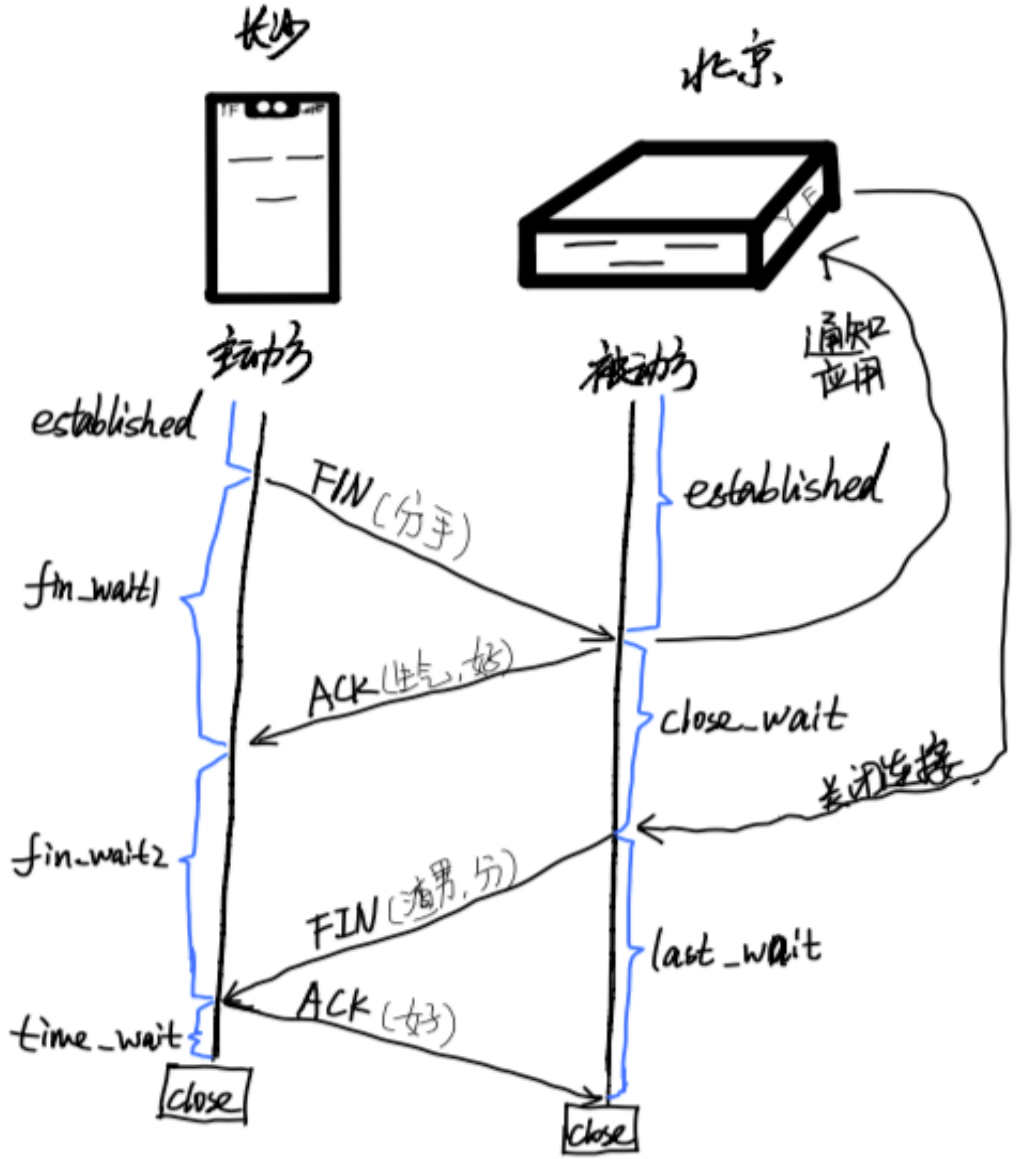


TCP的三次握手



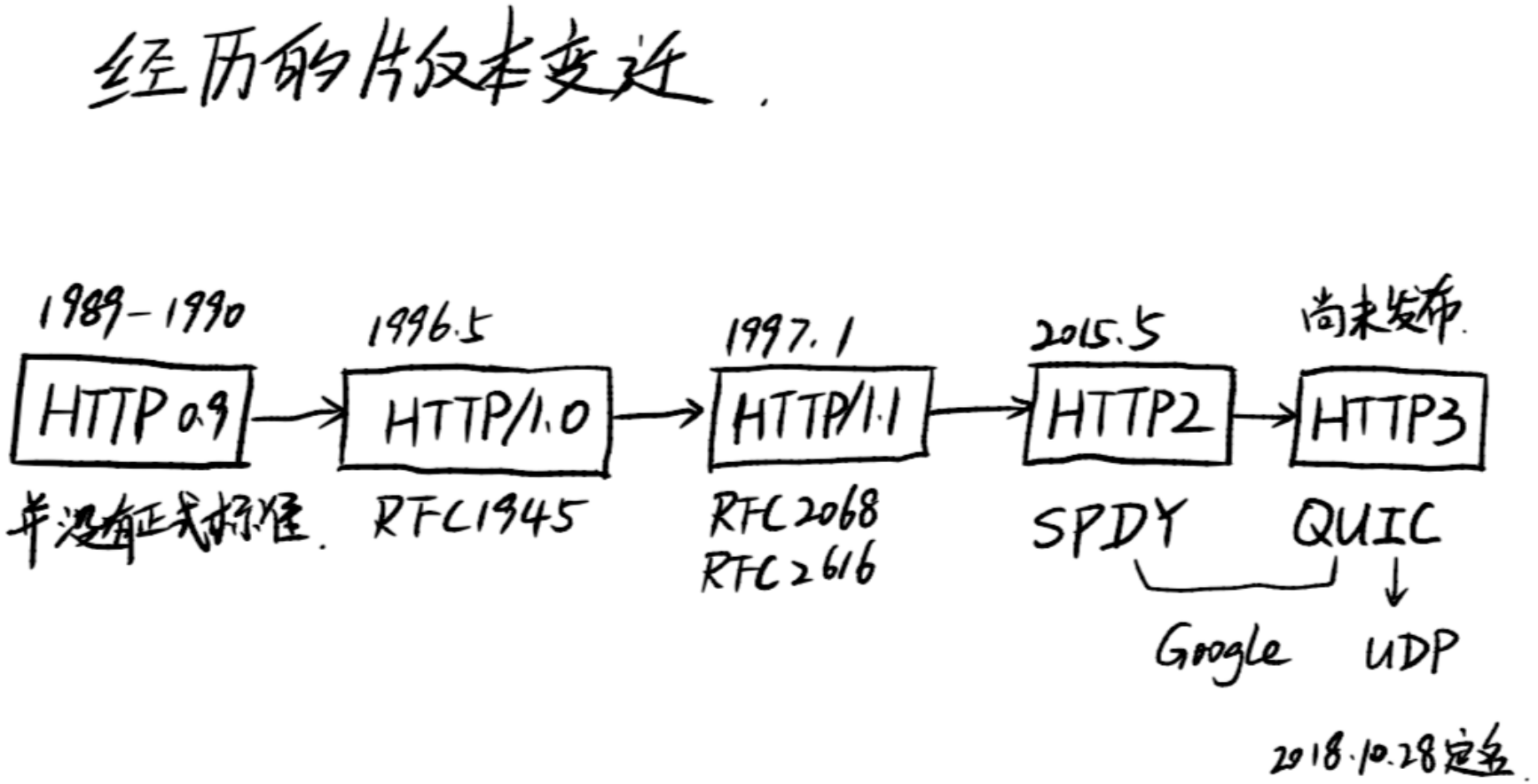
TCP的四次挥手

- tcp 断开连接时的各种状态

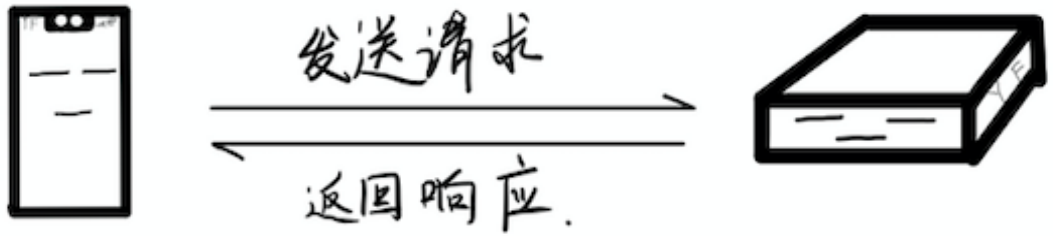


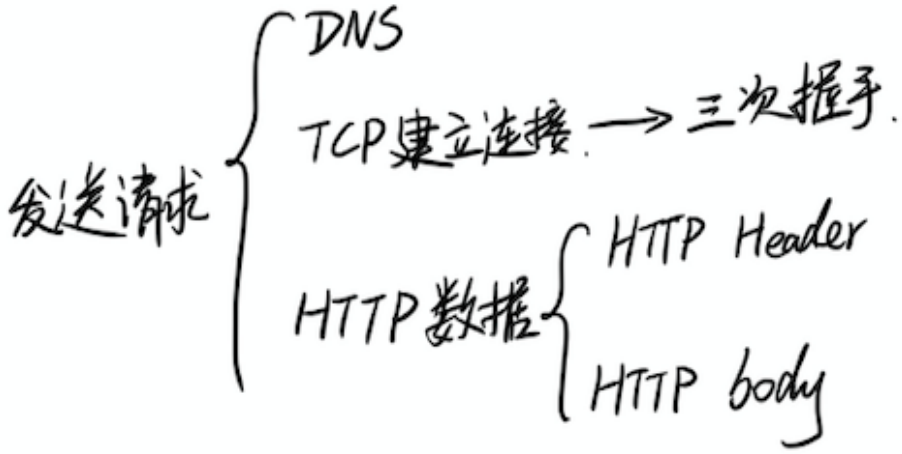
为什么我们讲HTTP而先去了解TCP/IP?

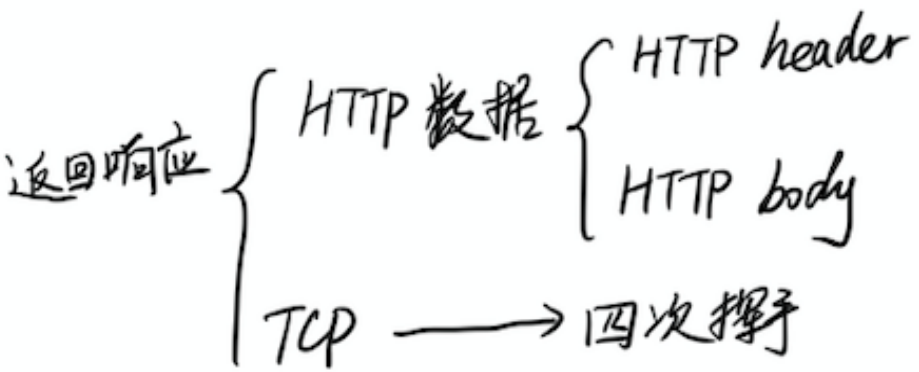
- 因HTTP版本的变迁与TCP/IP息息相关。



HTTP具体是什么?

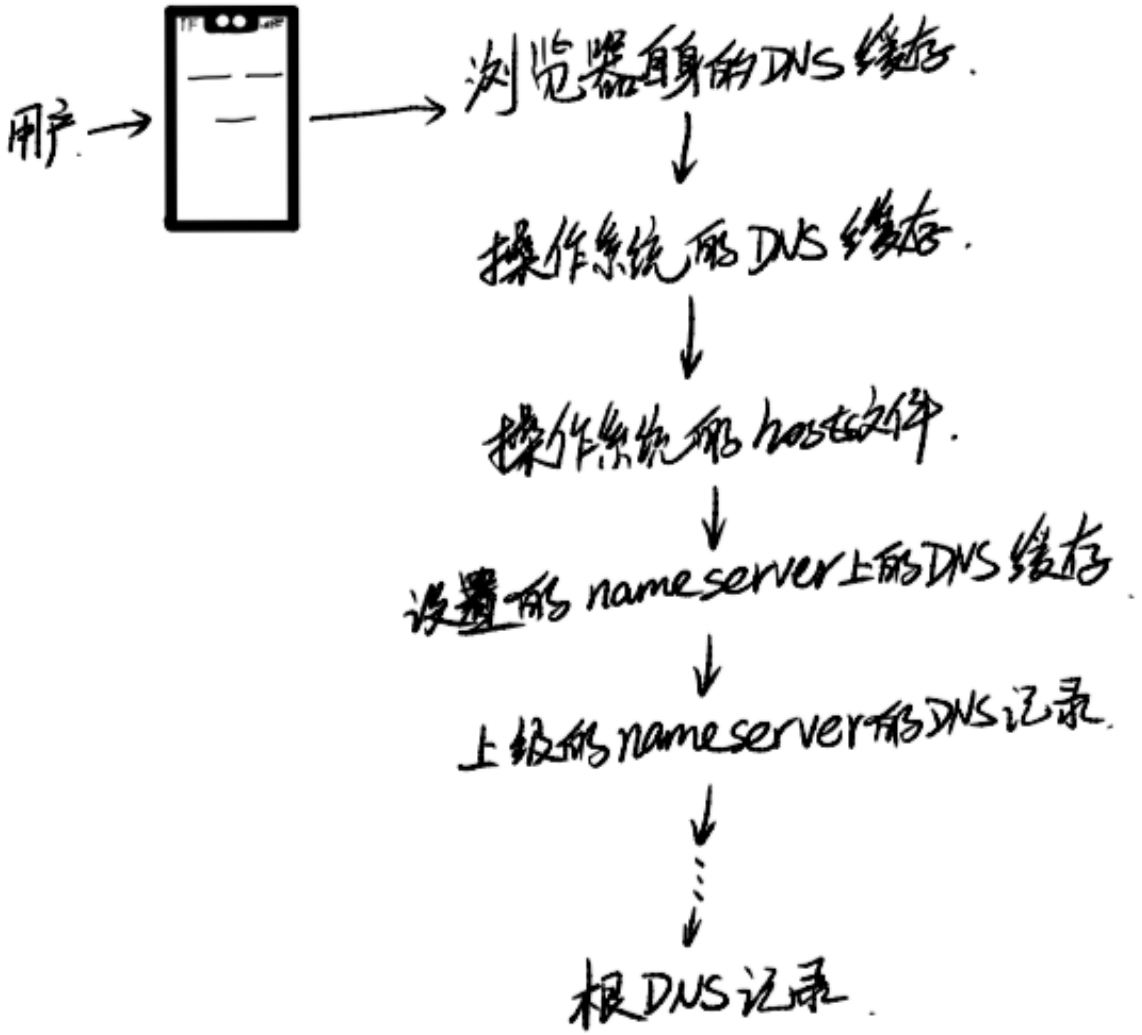

发送请求
返回响应


发送请求
DNS
TCP建立连接 → 三次握手
HTTP数据
HTTP Header
HTTP body


返回响应
HTTP数据
HTTP header
HTTP body
TCP → 四次挥手

DNS

- DNS 是计算机域名系统 (Domain Name System 或 Domain Name Service) 的缩写，它是由域名**解析器**和**域名服务器**组成的。
- ip太复杂，换成人类能读懂的方式
- 有人想换成中文，比如早期的3721中文上网，貌似没成功。
- 实际上好像记域名的人还是少数-->搜索引擎发展起来了。
- 对普通用户的用处确实不大了，但是对于网络管理来说，还是有很多实际意义的。



请求头

请求体

响应头

响应体

```
# yongfu @ yongfus-MacBook-Pro in ~ [16:54:49]
$ curl -v -H "Content-Type: application/json" http://www.liaoyongfu.com -d "love=you&time=forever"
* Rebuilt URL to: http://www.liaoyongfu.com/
*   Trying 122.114.241.250...
* TCP_NODELAY set
* Connected to www.liaoyongfu.com (122.114.241.250) port 80 (#0)
> POST / HTTP/1.1
> Host: www.liaoyongfu.com
> User-Agent: curl/7.61.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 21
>
* upload completely sent off: 21 out of 21 bytes
< HTTP/1.1 301 Moved Permanently
< Server: nginx/1.12.0
< Date: Thu, 17 Oct 2019 08:36:25 GMT
< Content-Type: text/html
< Content-Length: 185
< Connection: keep-alive
< Location: https://www.liaoyongfu.com/
<
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.12.0</center>
</body>
</html>
```

请求头

中间有个空行

请求的body

响应头信息

中间也有个空行

响应的body

HTTP请求方法

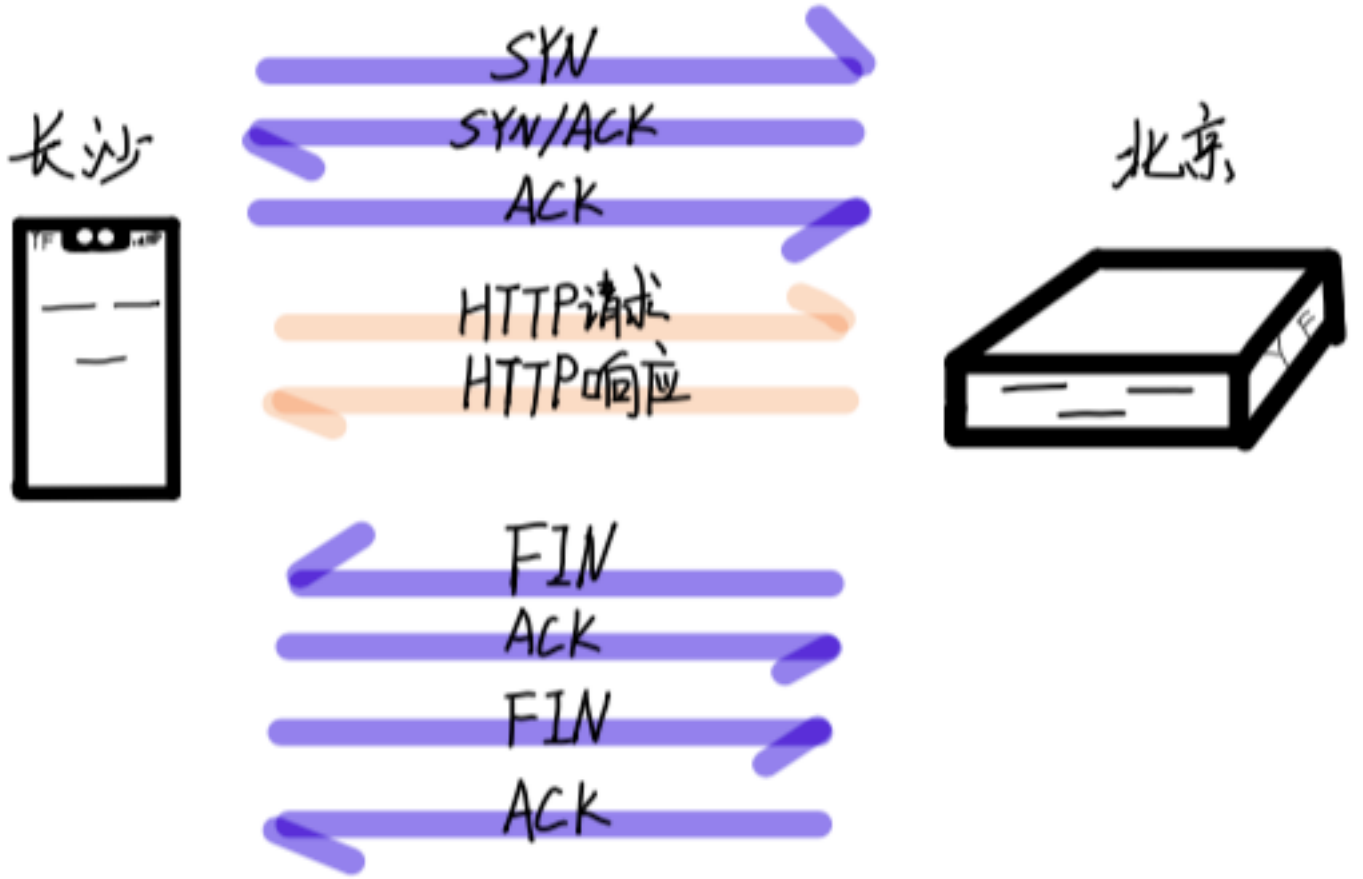
方法	描述
OPTIONS	返回服务器针对特定资源所支持的**HTTP**请求方法，也可以利用向**web**服务器发送‘*’的请求来测试服务器的性能
HEAD	向服务器索与**GET**请求相一致的响应，只不过响应体将不会被返回
GET	向特定的资源发出请求。
POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）
PUT	向指定资源位置上传其最新内容
DELETE	请求服务器删除**Request-URL**所标识的资源请求服务器删除**Request-URL**所标识的资源
CONNECT	**HTTP/1.1**协议中预留给能够将连接改为管道方式的代理服务器
TRACE	回显服务器收到的请求，主要用于测试或诊断

HTTP状态码

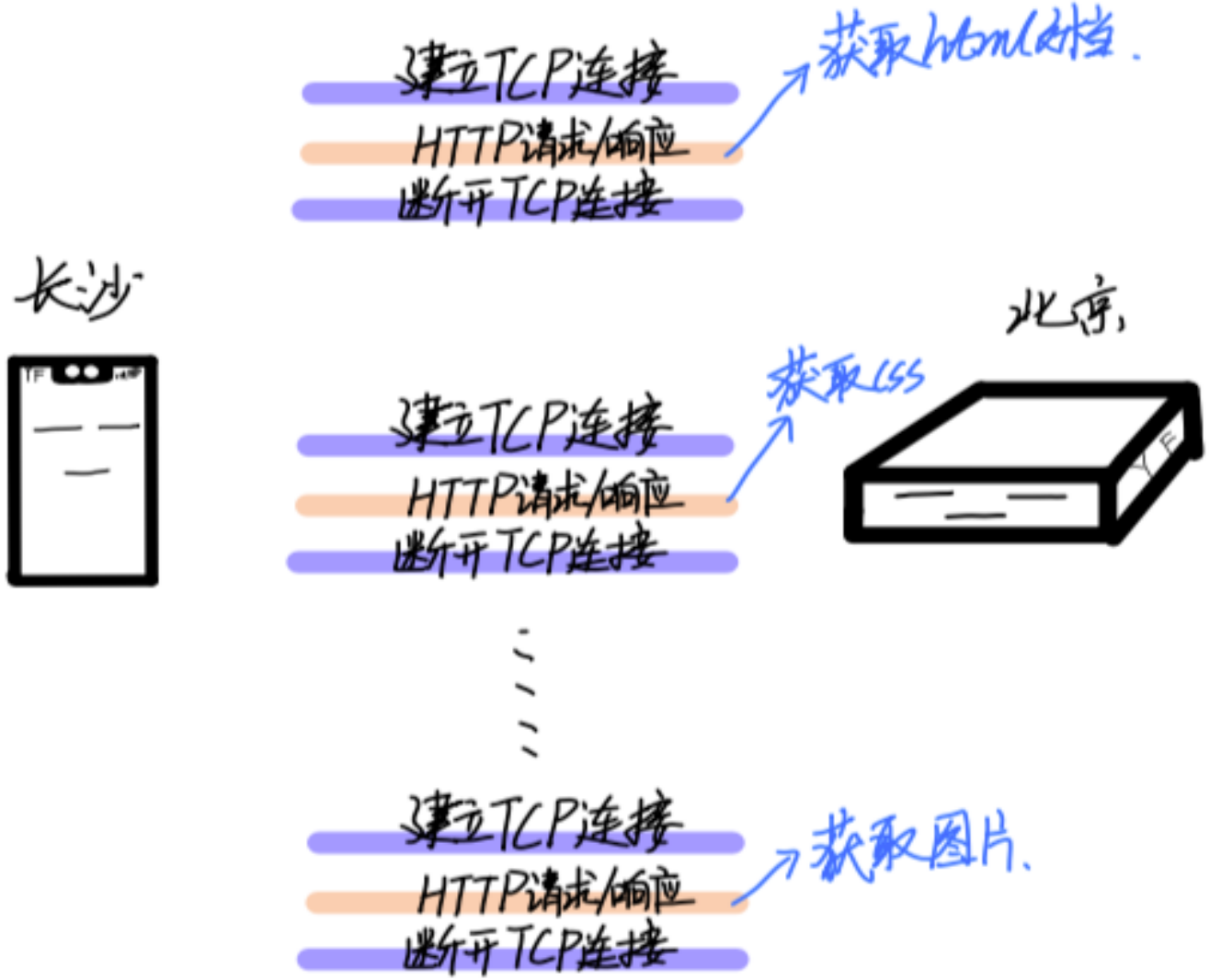
状态码	说明
1xx	信息，表示服务器收到请求，需要继续操作
2xx	成功，操作被成功接收并处理
3xx	重定向，需要进一步的操作以完成请求
4xx	客户端错误，请求包含语法错误或服务器无法完成的请求 如**400**，**401**，**403**，**404**等
5xx	服务端错误，服务器在处理请求的过程中发生的错误 如**500**，**502**，**503**等

HTTP/1.0

- 无状态（HTTP基本特性）
- 无连接（短连接）



假设使用 HTTP/1.0 访问 www.liaoyongfu.com

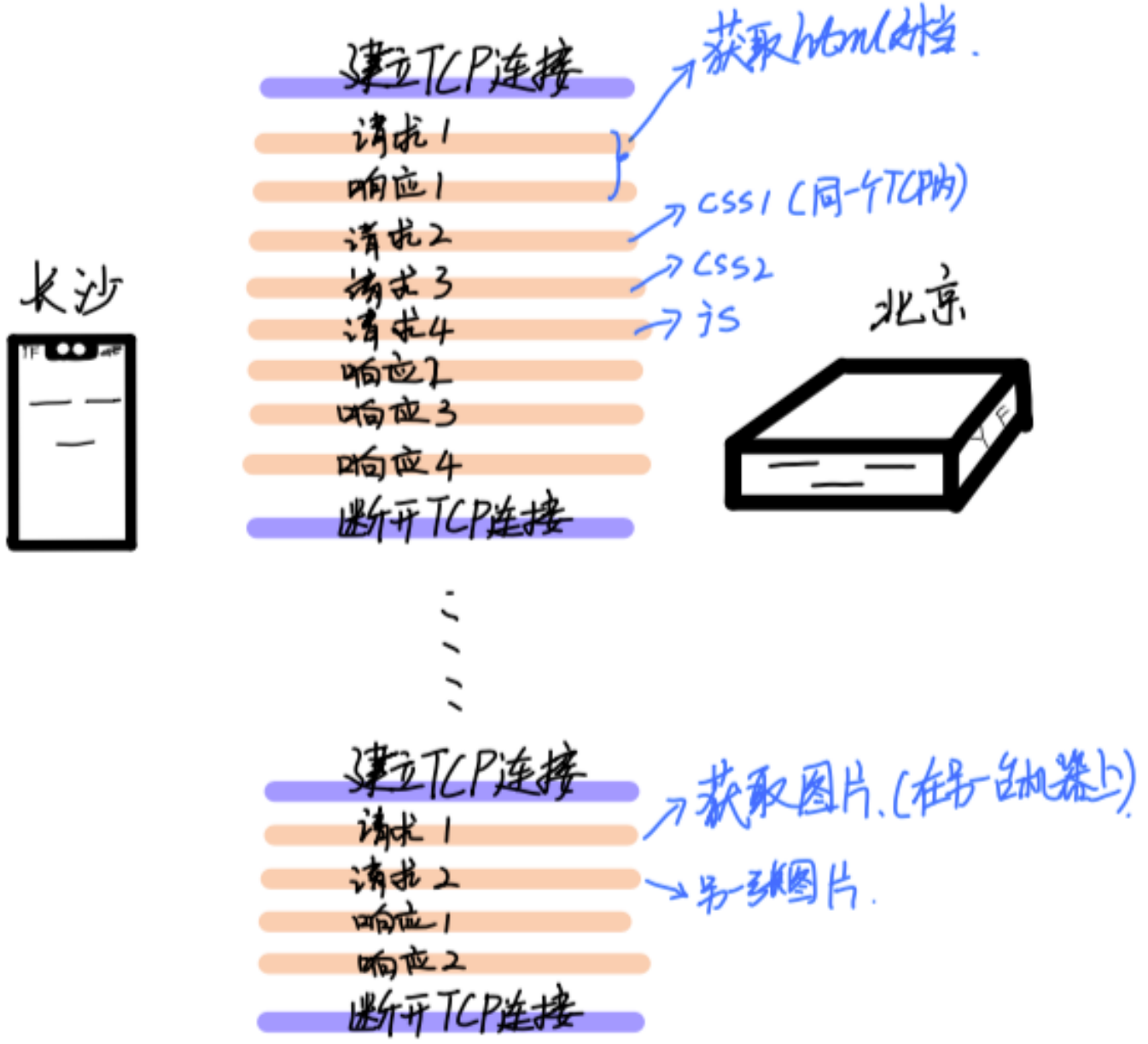


HTTP/1.0 --> HTTP/1.1

- 默认开启了长连接(keep-alive）选项
- 新增了host头信息
- 支持对body进行压缩
- 新增了cookie头信息
- 新增了内容长度控制： Content-Length
- 新增了缓存控制
- 其他（状态码、请求方法等等）

HTTP/1.1

- 长连接
- pipeline(并不是所有浏览器都支持)



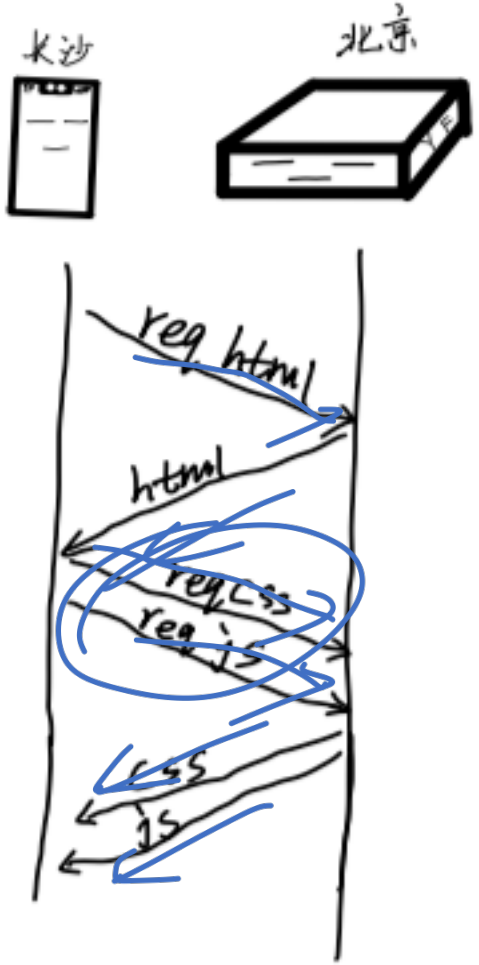
HTTP/1.1足够优秀，为啥还搞个HTTP2?

- HTTP 1.1 的弊端
- 1. 请求还是得一个一个的发送（没开pipeline的情况）
- 2. 就算开启了pipeline（开启需双方都支持），还有队头阻塞问题
- 3. header越来越大，cookie和其他的头信息越来越大，我们自己的某些网站，光头信息就有8k+
- 4. 还是只能客户端首先发起请求，服务端没法主动推送数据?
- 5. 多域名可以提高浏览器的打开网站速度（建立多条tcp长连接），同时DNS时间会更长，增加了服务器和客户端维持连接的压力。
- 6. 响应还是以请求的顺序进行的，队头阻塞。

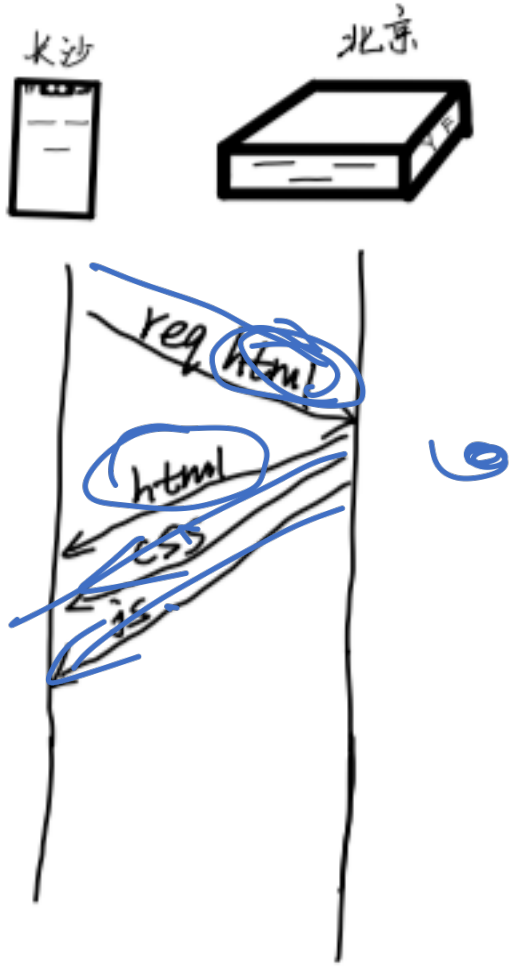
HTTP2 进行了哪些改进?

- 多路复用: 同一个连接并发处理多个请求。减少连接数，降低延时（建立新连接，是有延时的）
- 二进制分帧：在二进制分帧层上，HTTP 2.0 会将所有传输的信息分割为更小的消息和帧,并对它们采用二进制格式的编码 ，其中HTTP/1.x的首部信息会被封装到Headers帧，而我们的request body则封装到Data帧里面。客户端和服务器可以把HTTP 消息分解为互不依赖的帧，然后乱序发送，最后再在另一端把它们重新组合起来。
- header压缩：使用HPACK算法对header信息进行压缩，减少头部信息大小。
- 服务器推送: HTTP/1.x 都是只能有客户端发起请求给服务端，现在服务端可以直接往客户端推送消息了。

HTTP2的服务器推送



没有推送



有推送

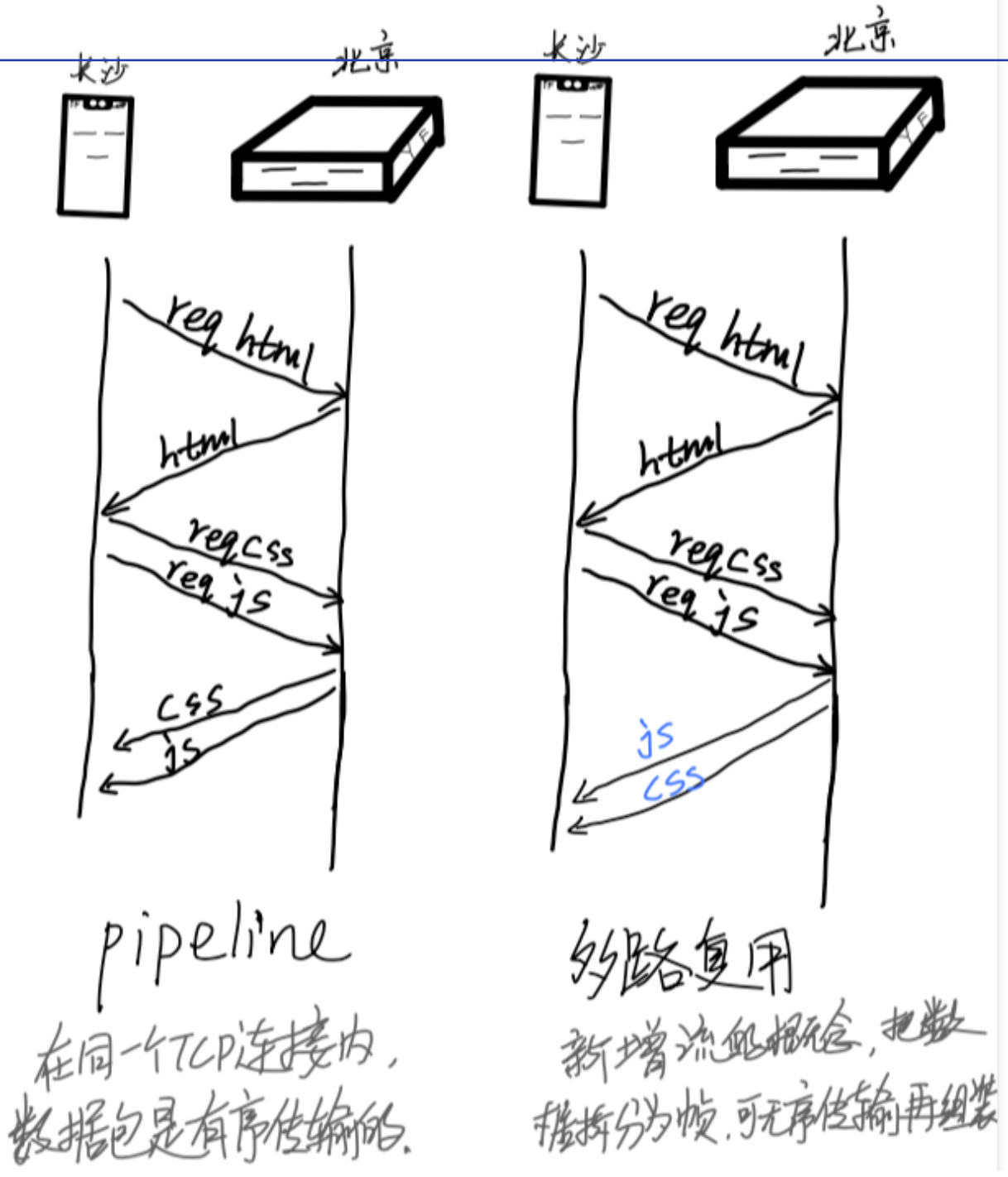
HTTP2的多路复用

帧(二进制数据)代表最小的数据单位，每个帧会有一个标识，
标识属于哪一条流，流也就是多个帧组成的数据流，
也就是对应的请求或响应信息。

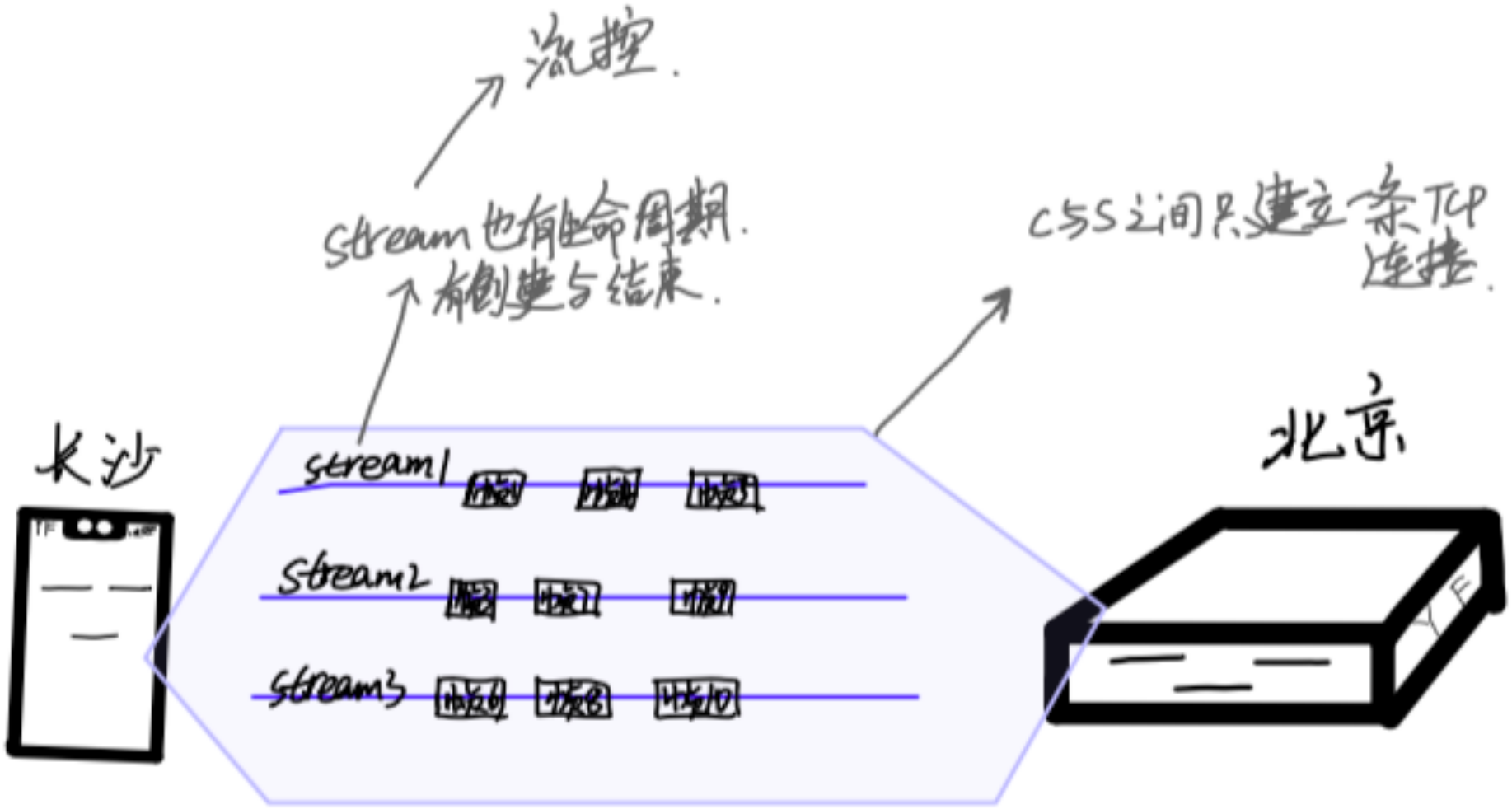
多路复用，就是在一个TCP连接中可以存在多条流。
换句话说，也就是可以发送多个请求，对端可以通过
帧中的标识，知道属于哪个请求。通过这个技术，可以
避免HTTP旧版本中的队头阻塞问题，极大的提高
传输性能。

流之间可具有不同的优先级。
流与流之间存在依赖与被依赖的关系。

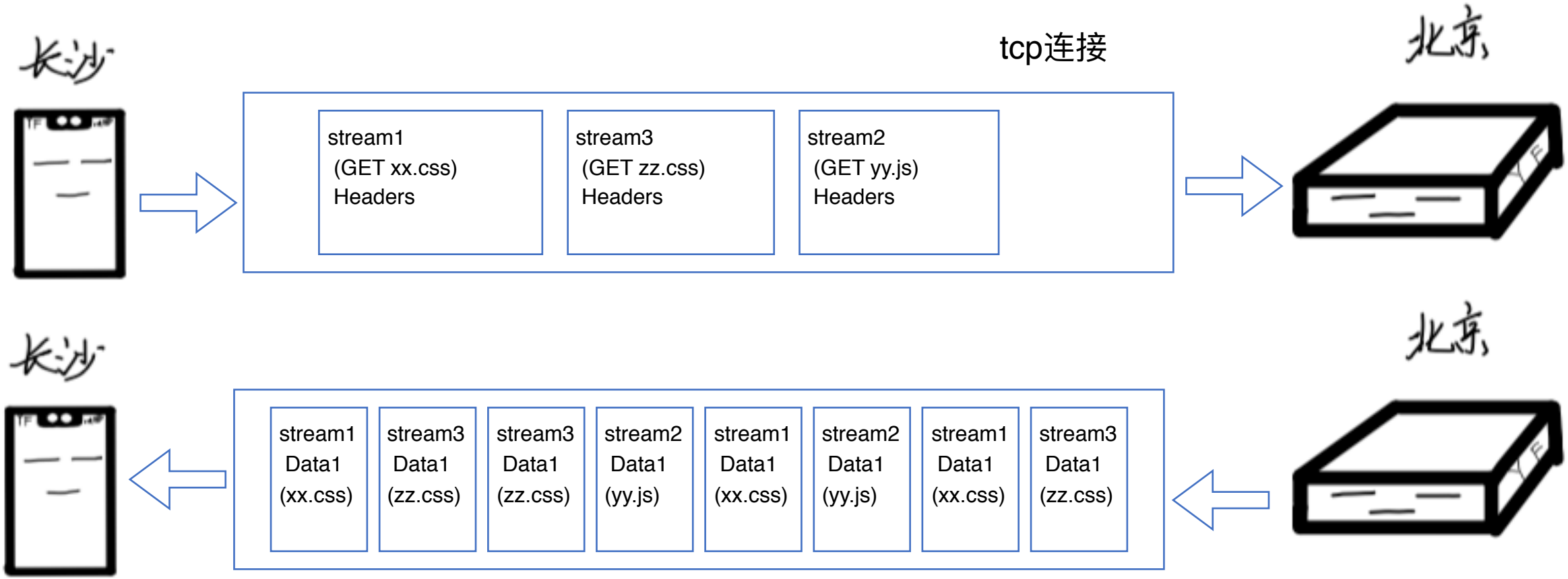
pipeline 和 多路复用 的区别



TCP、流、帧


流控.
stream也有生命周期.
有创建与结束.
CSS之间只建立一条TCP
连接.
长沙
北京
stream1
stream2
stream3

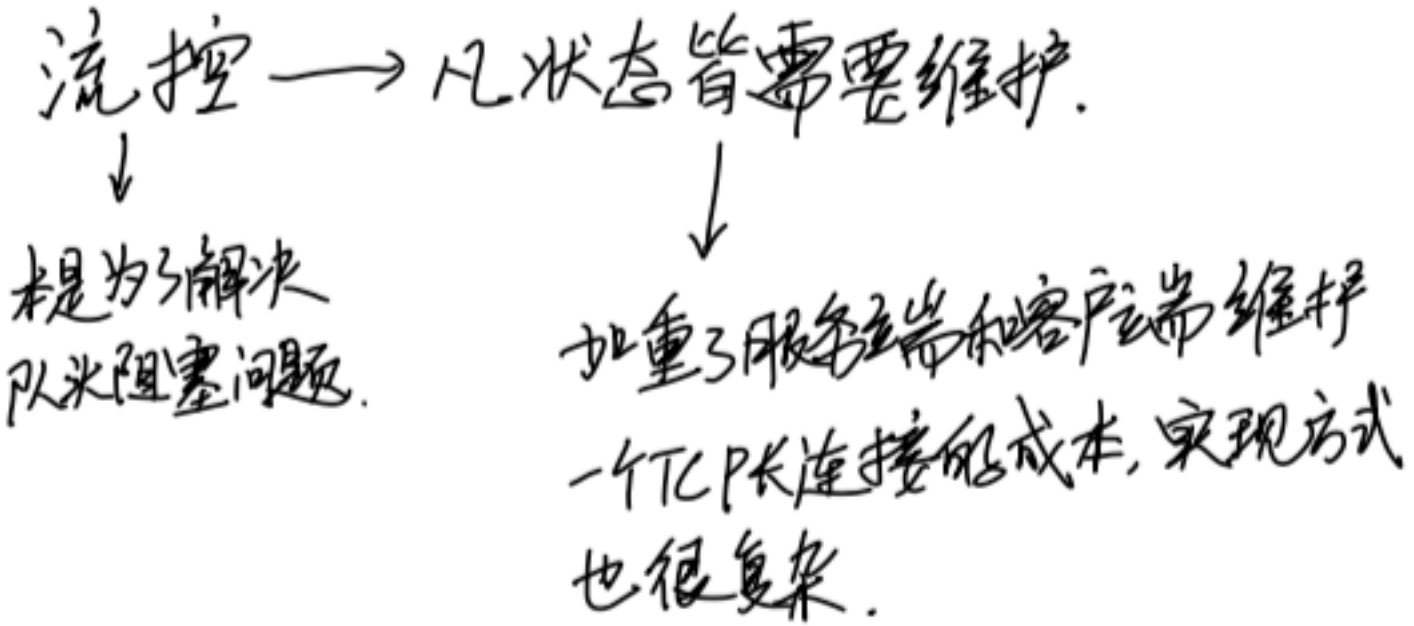
TCP、流、帧


长沙
tcp连接
北京
stream1
(GET xx.css)
Headers
stream3
(GET zz.css)
Headers
stream2
(GET yy.js)
Headers
长沙
北京
stream1
Data1
(xx.css)
stream3
Data1
(zz.css)
stream3
Data1
(zz.css)
stream2
Data1
(yy.js)
stream1
Data1
(xx.css)
stream2
Data1
(yy.js)
stream1
Data1
(xx.css)
stream3
Data1
(zz.css)

TCP、流、帧

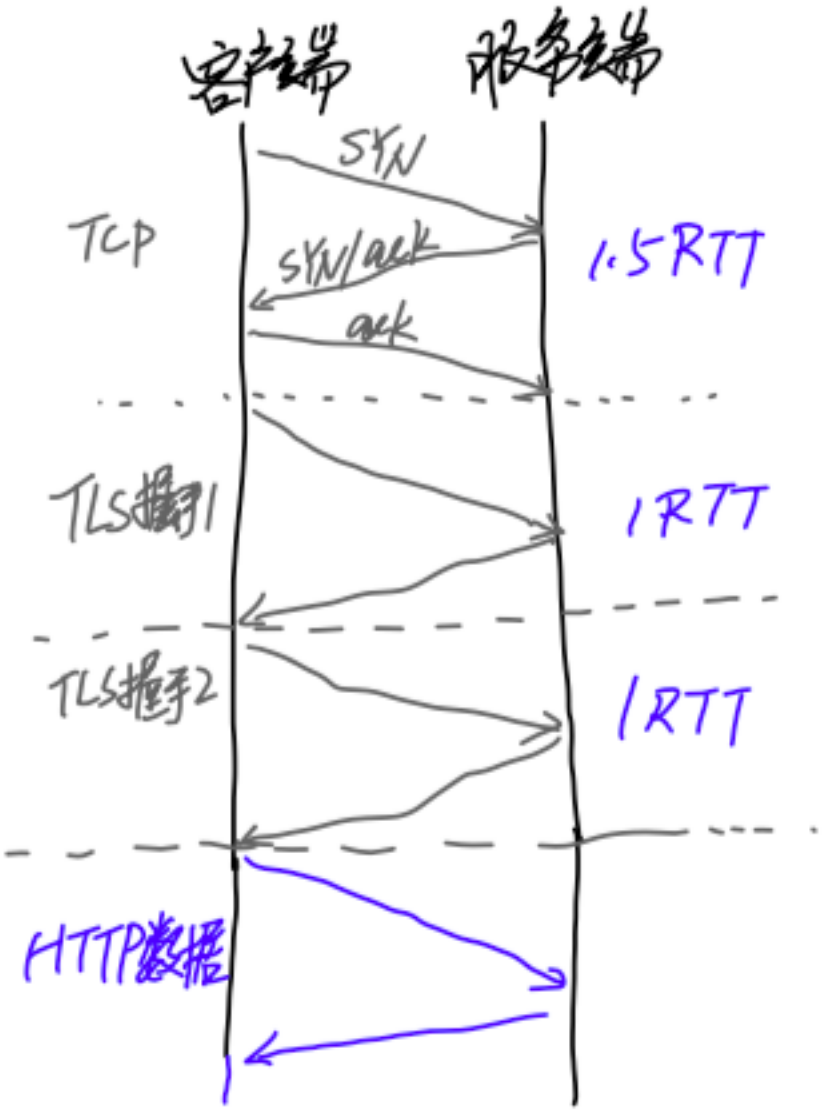
```
# yongfu @ yongfus-MacBook-Pro in ~/test [10:43:51]
$ /usr/local/Cellar/curl-openssl/7.66.0/bin/curl -vsa  https://www.liaoyongfu.com
*   Trying 122.114.241.250:443...
* TCP_NODELAY set
* Connected to www.liaoyongfu.com (122.114.241.250) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /usr/local/etc/openssl@1.1/cert.pem
  CApath: /usr/local/etc/openssl@1.1/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server accepted to use h2
* Server certificate:
*  subject: CN=www.liaoyongfu.com
*  start date: Jul 31 00:00:00 2019 GMT
*  expire date: Jul 30 12:00:00 2020 GMT
*  subjectAltName: host "www.liaoyongfu.com" matched cert's "www.liaoyongfu.com"
*  issuer: C=US; O=DigiCert Inc; OU=www.digicert.com; CN=Encryption Everywhere DV TLS CA - G1
*  SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x7fdf0b80fe00)
> GET / HTTP/2
> Host: www.liaoyongfu.com
> User-Agent: curl/7.66.0
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
< HTTP/2 200
< server: nginx/1.12.0
< date: Sat, 26 Oct 2019 02:25:35 GMT
< content-type: text/html
< content-length: 50602
< last-modified: Tue, 09 Jul 2019 01:53:29 GMT
< etag: "5d23f399-c5aa"
< accept-ranges: bytes
<
<!DOCTYPE html>
<!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]> <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
<head>
```

TCP、流、帧


流控 → 凡状态皆需要维护.
根为了解决
队头阻塞问题.
加重了服务端和客户端维护
一个TCP长连接的成本，实现方式
也很复杂.

理想很丰满，现实很骨感

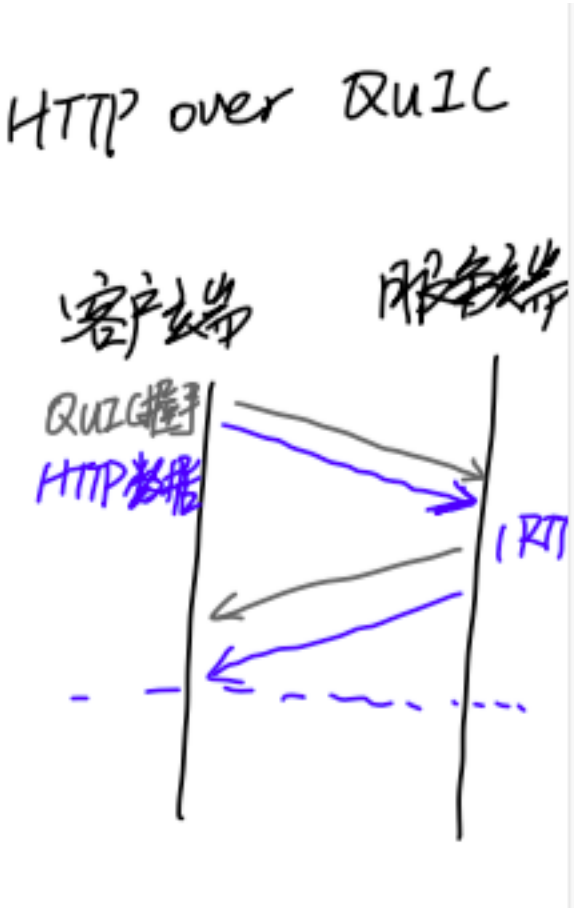
- HTTP2并没有强制推行，反而是一直在强制推行https。google和firefox甚至还有apple的小动作不断。
- TCP连接是一个可靠连接，一旦丢包，需要整个数据重传，此时HTTP2的性能表现反而不如HTTP/1.x
- 因为基于TCP连接的话，无论如何，都会有三次握手的建议过程，如果有https的话，还至少有1-2个RTT的ssl认证过程。
- 手机网络的切换（wifi和4G，高铁等快速变更基站，5G网络的覆盖范围更小）



HTTP2 --> HTTP3

- HTTP3 还没有正式发布，也是在Google的技术上(QUIC)发展起来的。

- HTTP2 和 HTTP3 的主要区别：传输层协议从tcp协议换成了udp协议
- https://daniel.haxx.se/blog/2018/11/11/http-3/



HTTP3

① 0RTT

② 多路复用（QUIC自带）

HTTP2相当于在TCP上实现了多路复用。

HTTP3 则是在UDP上实现的多路复用（QUIC）

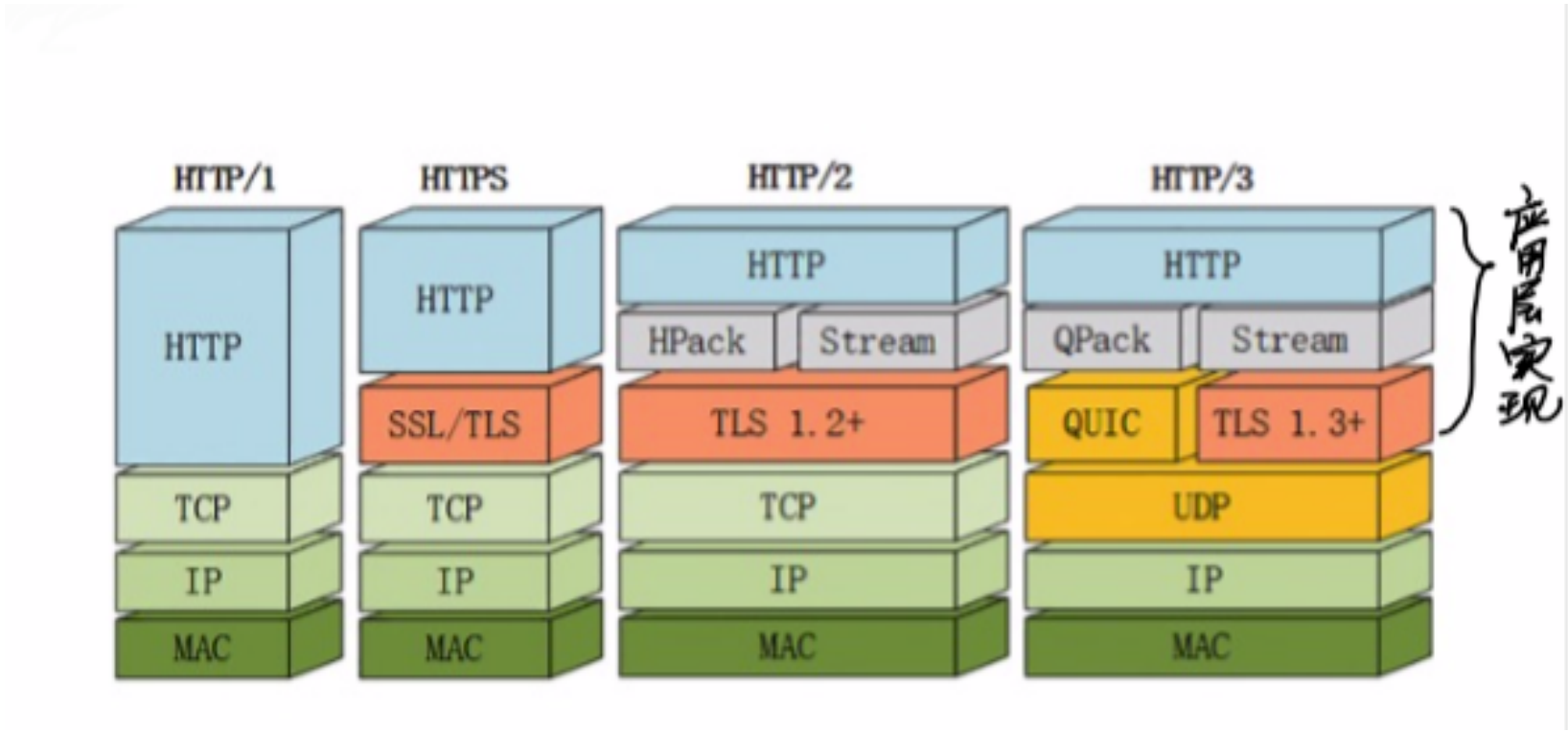
为什么没有TCP上的多路复用的问题？

③ 前向纠错机制

通过协议计算，单独发送一个校验包。如果丢失的是校验包，则通过其他的包和校验包可计算出丢失的包，而不需要重传。如果数据包丢失，那也还是要重传的。

④ 加密认证的报文（非之前那种外挂式的加密）。

回顾HTTP的发展

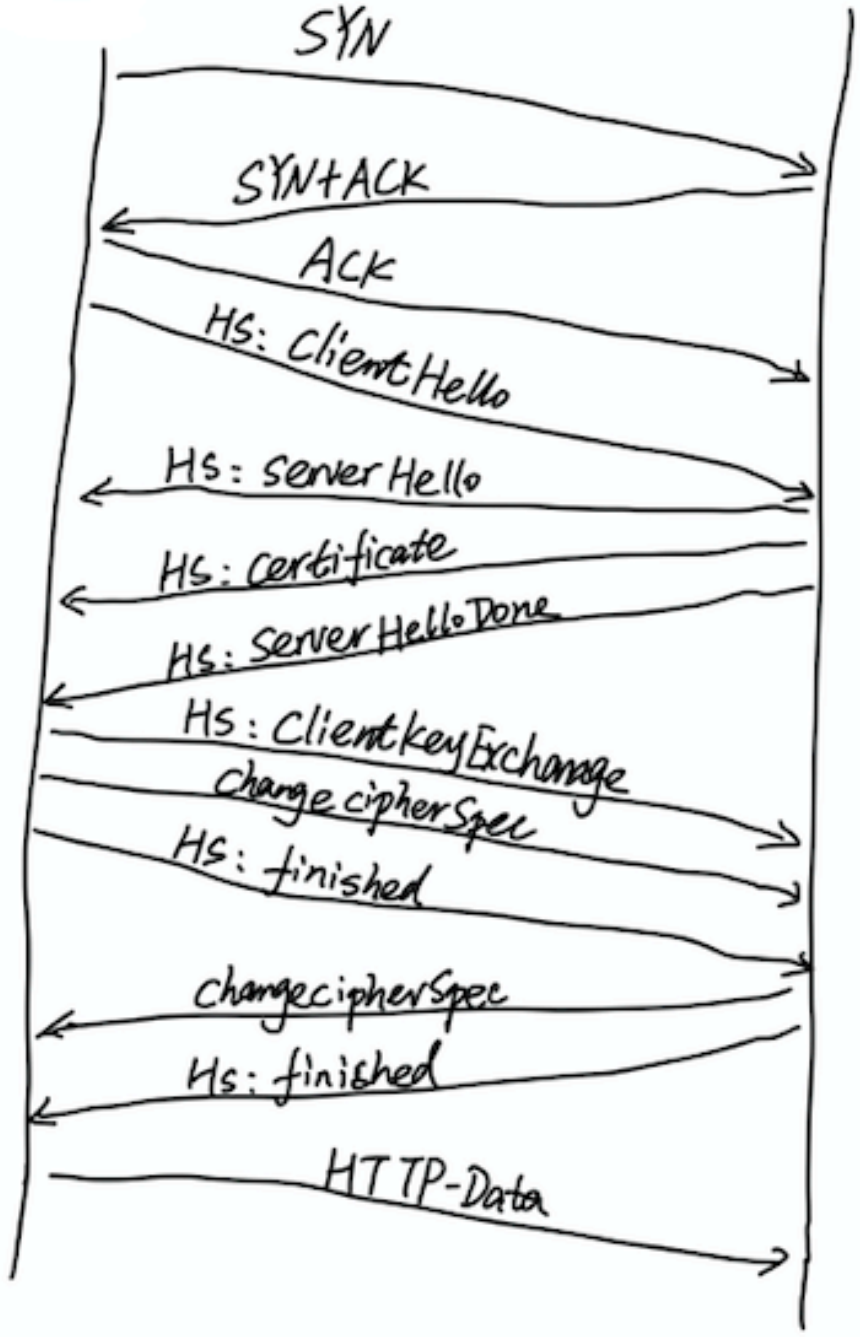

HTTP/1
HTTPS
HTTP/2
HTTP/3
HTTP
HTTP
SSL/TLS
HTTP
HPack
Stream
TLS 1.2+
HTTP
QPack
Stream
QUIC
TLS 1.3+
TCP
TCP
TCP
UDP
IP
IP
IP
IP
MAC
MAC
MAC
MAC
应用层实现

https

- 前面说了这么多的http及http协议的发展，怎么好像和https没有任何关系呢?
- 那么https又是什么呢? （基于HTTP协议，通过SSL或TLS提供加密处理数据、验证对方身份以及数据完整性保护）

https


长沙


SYN
SYN+ACK
ACK
Hs: ClientHello
Hs: ServerHello
Hs: certificate
Hs: ServerHelloDone
Hs: ClientkeyExchange
ChangecipherSpec
Hs: finished
ChangecipherSpec
Hs: finished
HTTP-Data


北京

参考

- HTTP协议超级详解
- HTTP/1.0和HTTP/1.1的区别，HTTP怎么处理长连接
- HTTP1.0、HTTP 1.1、HTTP 2.0之间的主要区别
- 使用burpsuite拦截，篡改，转发请求
- 一文读懂HTTP/2及HTTP/3特性
- HTTP和HTTPS协议，看一篇就够了
- 深入理解TCP、UDP协议及两者的区别
- Http2.0与Http1.x的区别
- 永福的博客
- 《图解HTTP》作者〔日〕上野宣 译者于均良