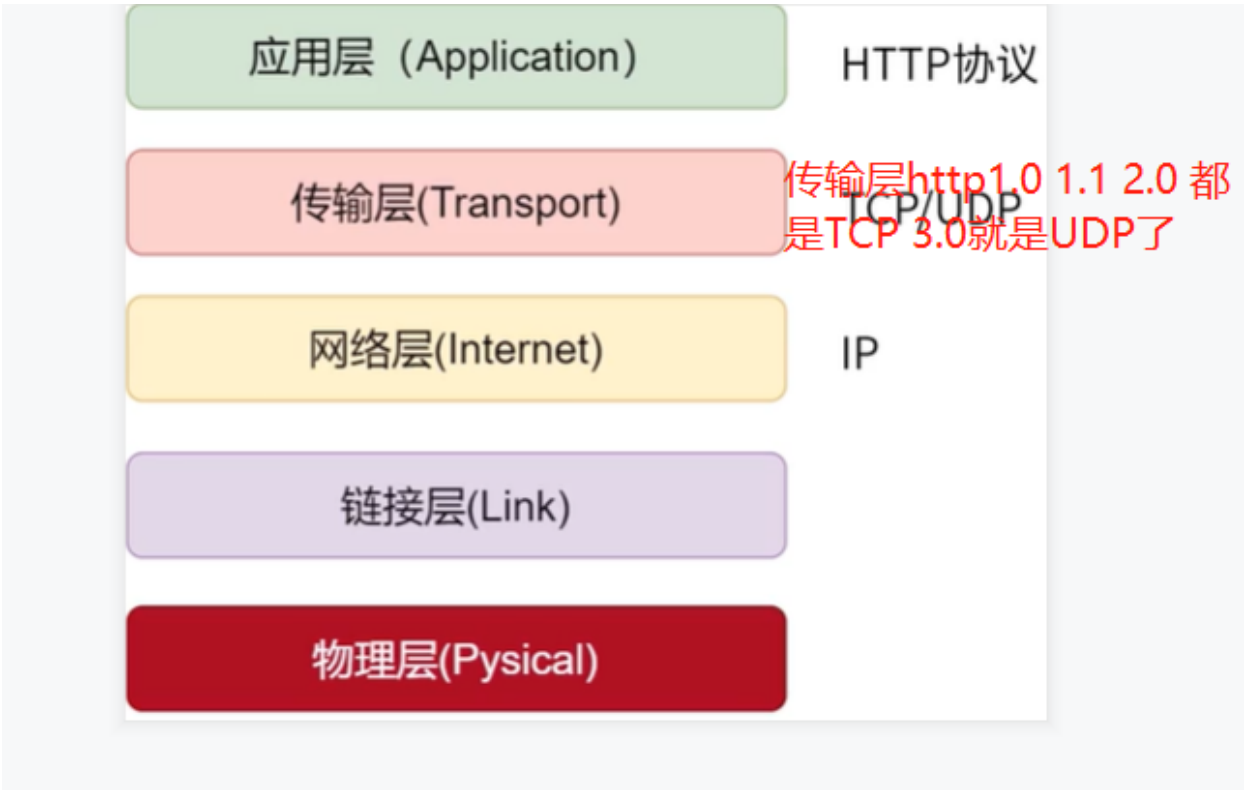
课程目标

TCP VS UDP

Internet协议群(TCP/IP协议群)



传输层和网络层

UDP(User Data Protocol) 用户数据报协议

- 比TCP节省网络资源和迅速

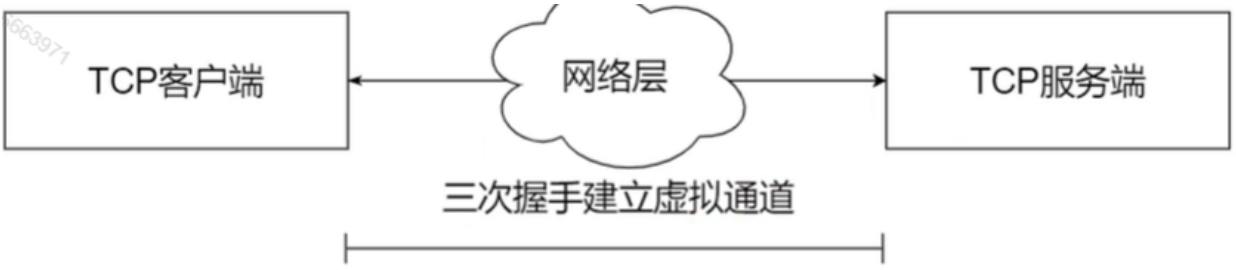
  --不需要建立连接(延迟更低)

  --封包体积更小(传输速度更快)

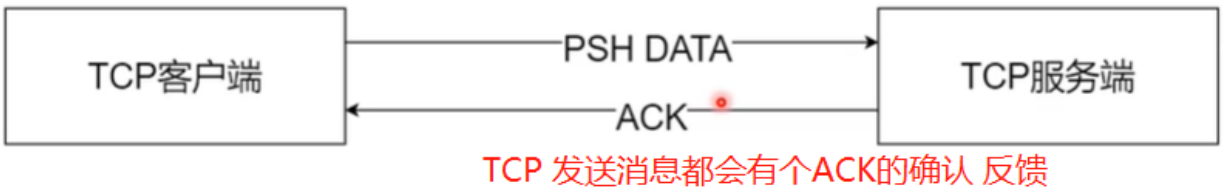
  --不关心数据顺序(不需要序号和ACK,传输快速)

  -- 不保证数据不丢失

连接



重发(校验)



UDP不保证顺序 如图所示

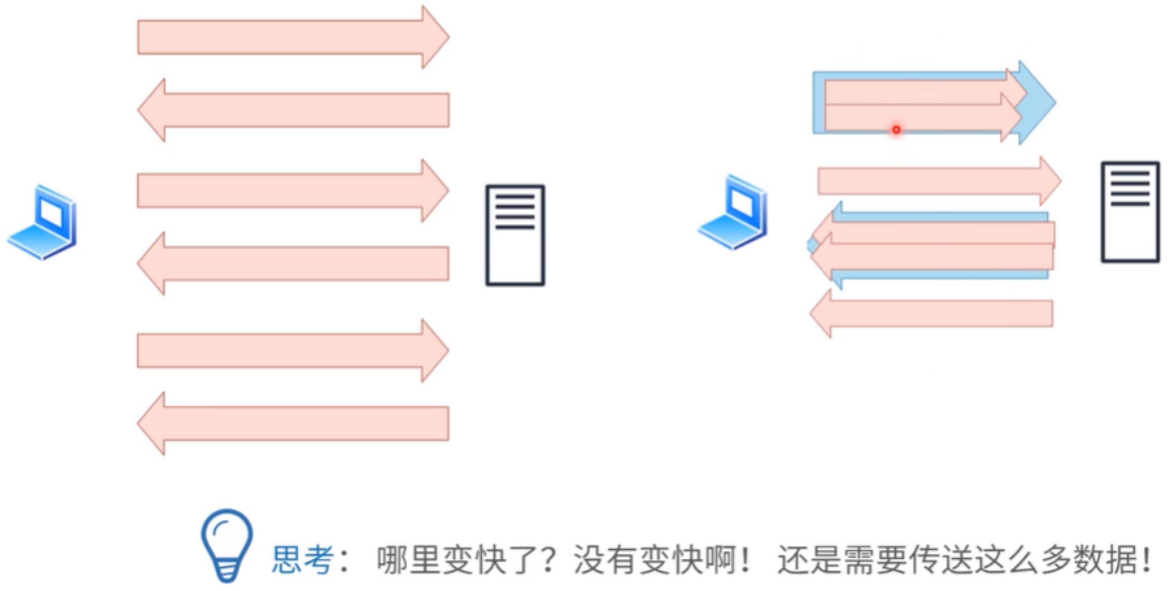
思考

UDP没有虚拟连接，不校验数据，不保证顺序，没有收到不重发 --- 是不是意味着不安全不可靠呢？并不是这样，只是UDP自由度更高,需要用户程序在应用层定义类似的机制，TCP面向流(API接收流) UDP面向对象(API接收数据包) 场景不同：TCP- 远程控制 UDP-DNS查询 -模糊:文件&文本，多媒体(TCP，现在UDP也在慢慢实现了)

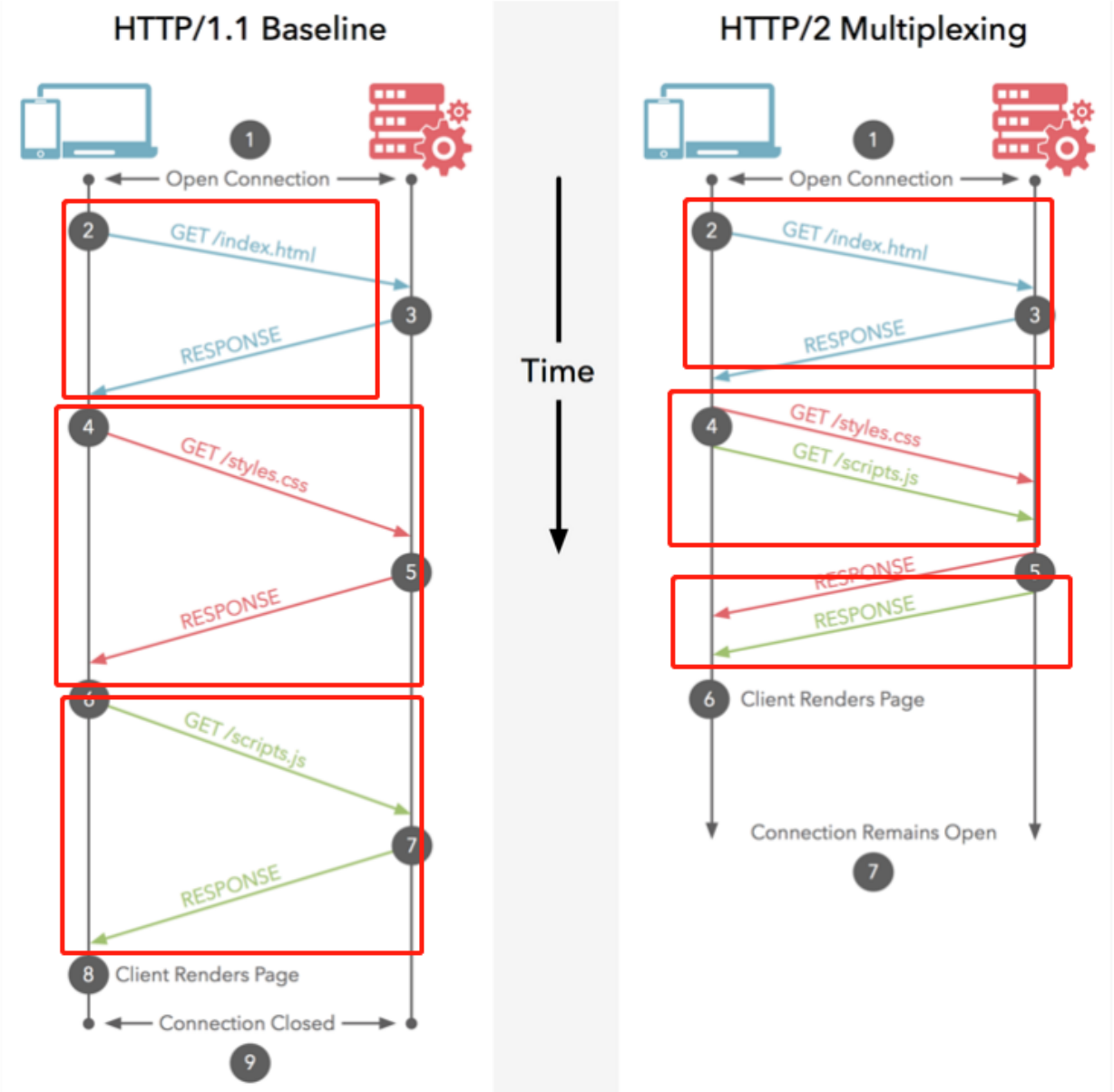
HTTP2.0目标(优化)

多个请求多路复用

http1.1排队问题，比如的上传三个文件，三次握手后，共用了一个TCP连接，这个想法很好，不用握九次手了， 但是需要排队，上传需要排队，接收需要排队。这样就出现了阻塞。文件连接限制，浏览器中对建立连接有相关限制。建立太多的连接需要占相关的资源。这里可以看到多路复用



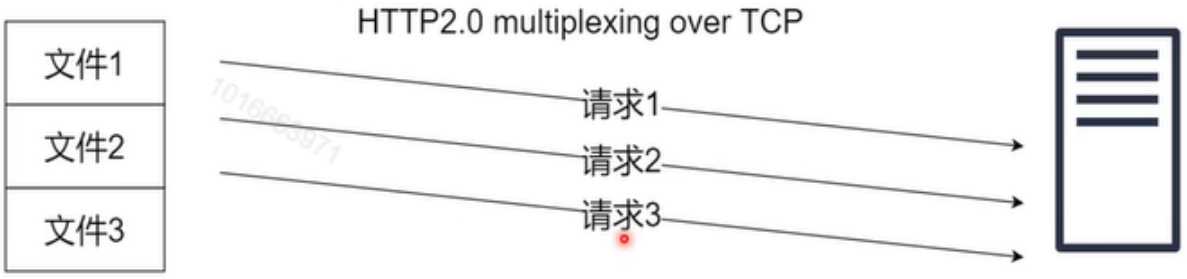
如图所示更加清晰



防止队头阻塞

文件一队头阻塞了，后面都阻塞了，看一下HTTP2.0的解决方案

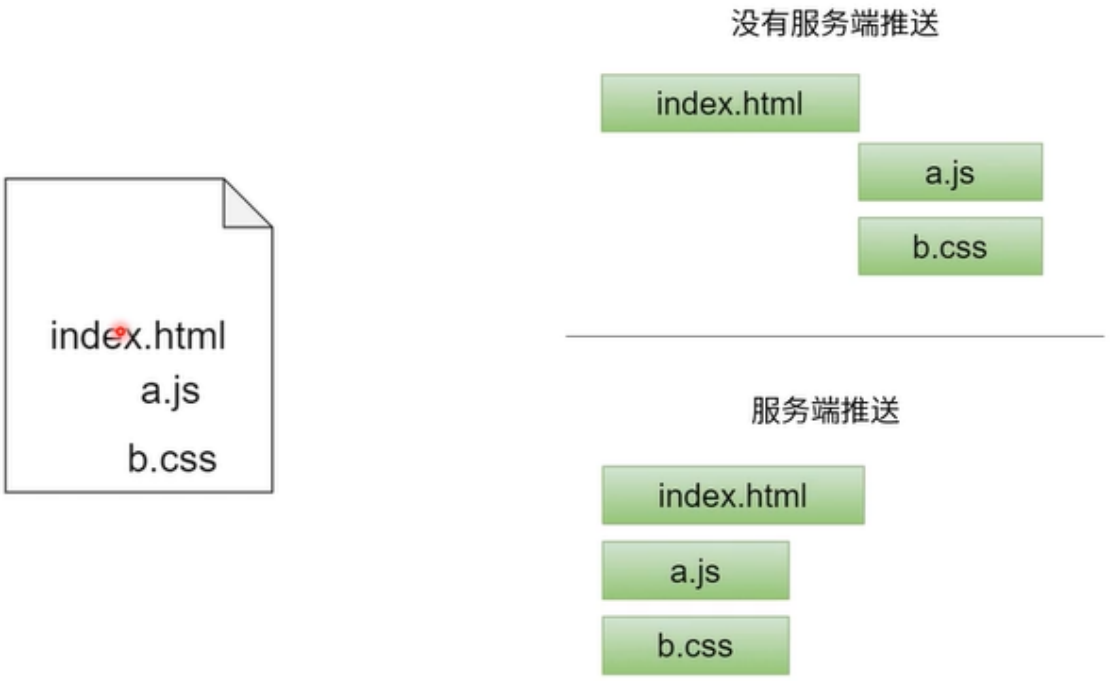
http2.0的做法，是把文件1 文件2 文件3 都打包成小块，当文件1的小块阻塞了，文件2 文件3的小块可以正常回来，也可以文件1 文件2 文件3 同时请求，就看谁先回来，谁后回来。这样对带宽的利用率是最高的



压缩HTTP头部

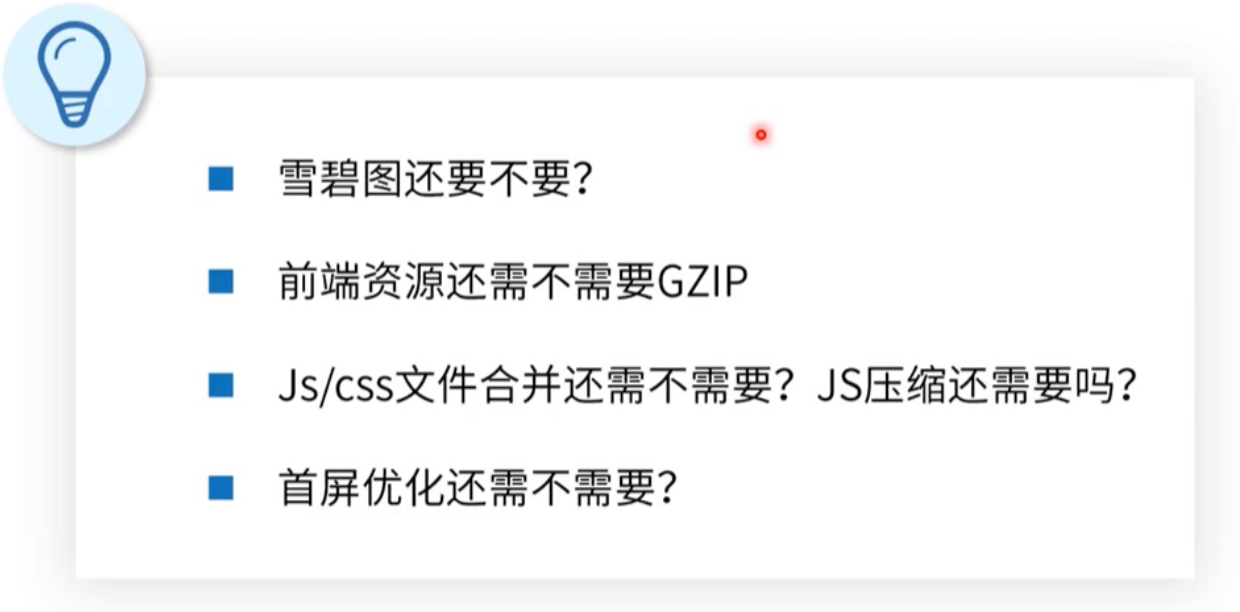
- 通过HPACK技术压缩

服务端推送



现在有一个文件，这个文件是一个HTML文件，里面有引用了js和css，没有服务端推送的时候，先请求加载index.html index.html中引入的js和css 这时候才去请求js和css 服务端才返回。而有服务端推送的话，客户端在请求index.html的时候，就把js和css一起返回给客户端了。但是有时候会出现悖论(比如，index里面有20个js文件，但是20文件加载顺序是有先后，服务端推送的话，算法不会计算这个先后顺序，造成推送顺序不确定，页面加载异常。这个就是与服务端推送初衷相悖了)

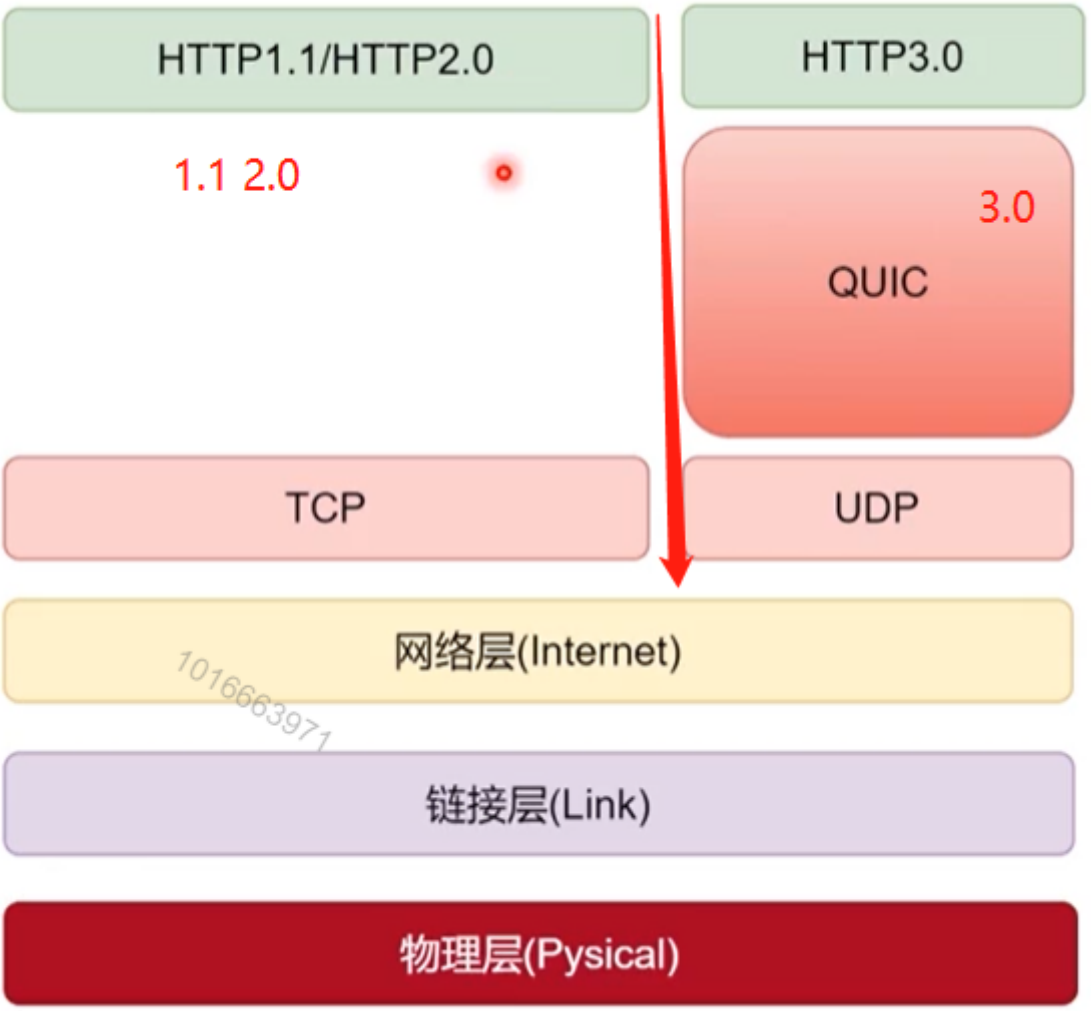
思考



1. 雪碧图(精灵图)都是在HTTP1.0 时代，现在可以忽略了。
2. gizp压缩还是需要的，http2.0只是做了头部压缩。
3. js/css 文件合并其实和雪碧图一样，没必要合并了.js压缩还是需要的
4. 首屏优化这个还是必须要的

HTTP3.0

3.0 还是一个试验阶段



课程小结

- UDP把自由度给了用户，使用的人少；TCP自由度低，用的人多。（思考）
- HTTP2.0/HTTP3.0都兼容HTTP1.1（思考）
- 问题：谁在推动协议发展？