

[< 趣谈网络协议](#)[首页](#) | [🔍](#)

第38讲 | 知识串讲：用双十一的故事串起碎片的网络协议（中）

2018-08-13 刘超

**朗读人：刘超**

时长09:50 大小4.51M

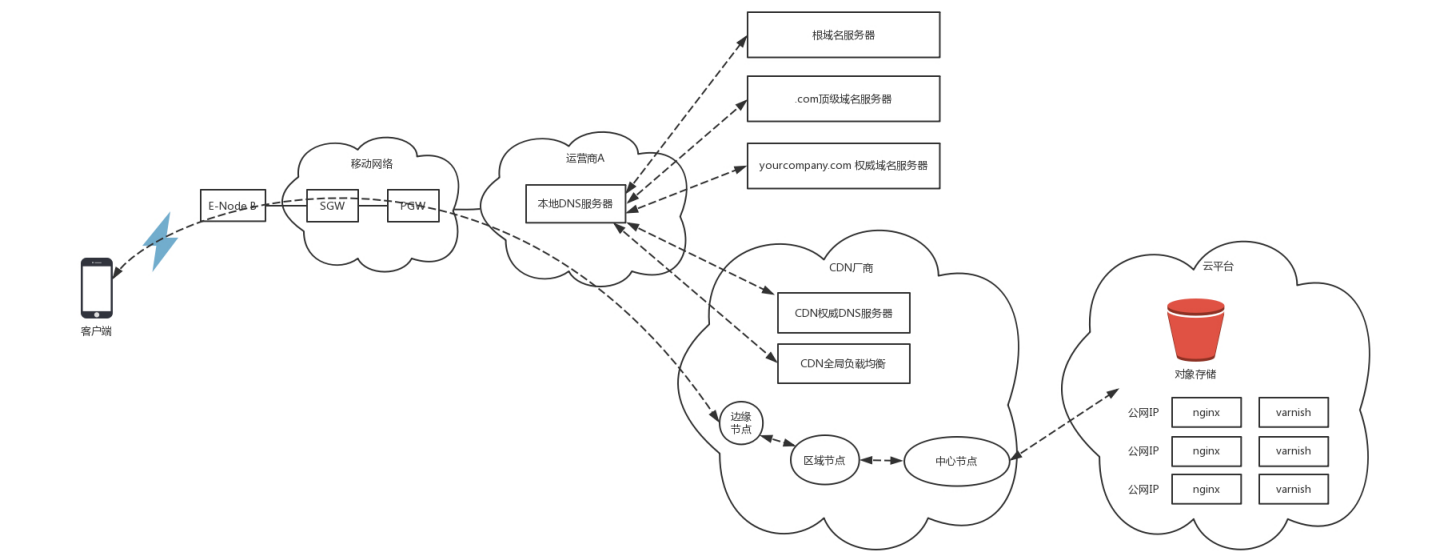


上一节我们讲到，手机 App 经过了一个复杂的过程，终于拿到了电商网站的 SLB 的 IP 地址，是不是该下单了？

别忙，俗话说的好，买东西要货比三家。大部分客户在购物之前要看很多商品图片，比来比去，最后好不容易才下决心，点了下单按钮。下单按钮一按，就要开始建立连接。建立连接这个过程也挺复杂的，最终还要经过层层封装，才构建出一个完整的网络包。今天我们就来看这个过程。

4. 购物之前看图片，静态资源 CDN

客户想要在购物网站买一件东西的时候，一般是先去详情页看看图片，是不是想买的那一款。



我们部署电商应用的时候，一般会把静态资源保存在两个地方，一个是接入层 nginx 后面的 varnish 缓存里面，一般是静态页面；对于比较大的、不经常更新的静态图片，会保存在对象存储里面。这两个地方的静态资源都会配置 CDN，将资源下发到边缘节点。

配置了 CDN 之后，权威 DNS 服务器上，会为静态资源设置一个 CNAME 别名，指向另外一个域名 [cdn.com](#)，返回给本地 DNS 服务器。

当本地 DNS 服务器拿到这个新的域名时，需要继续解析这个新的域名。这个时候，再访问的时候就不是原来的权威 DNS 服务器了，而是 [cdn.com](#) 的权威 DNS 服务器。这是 CDN 自己的权威 DNS 服务器。

在这个服务器上，还是会设置一个 CNAME，指向另外一个域名，也即 CDN 网络的全局负载均衡器。

本地 DNS 服务器去请求 CDN 的全局负载均衡器解析域名，全局负载均衡器会为用户选择一台合适的缓存服务器提供服务，将 IP 返回给客户端，客户端去访问这个边缘节点，下载资源。缓存服务器响应用户请求，将用户所需内容传送到用户终端。

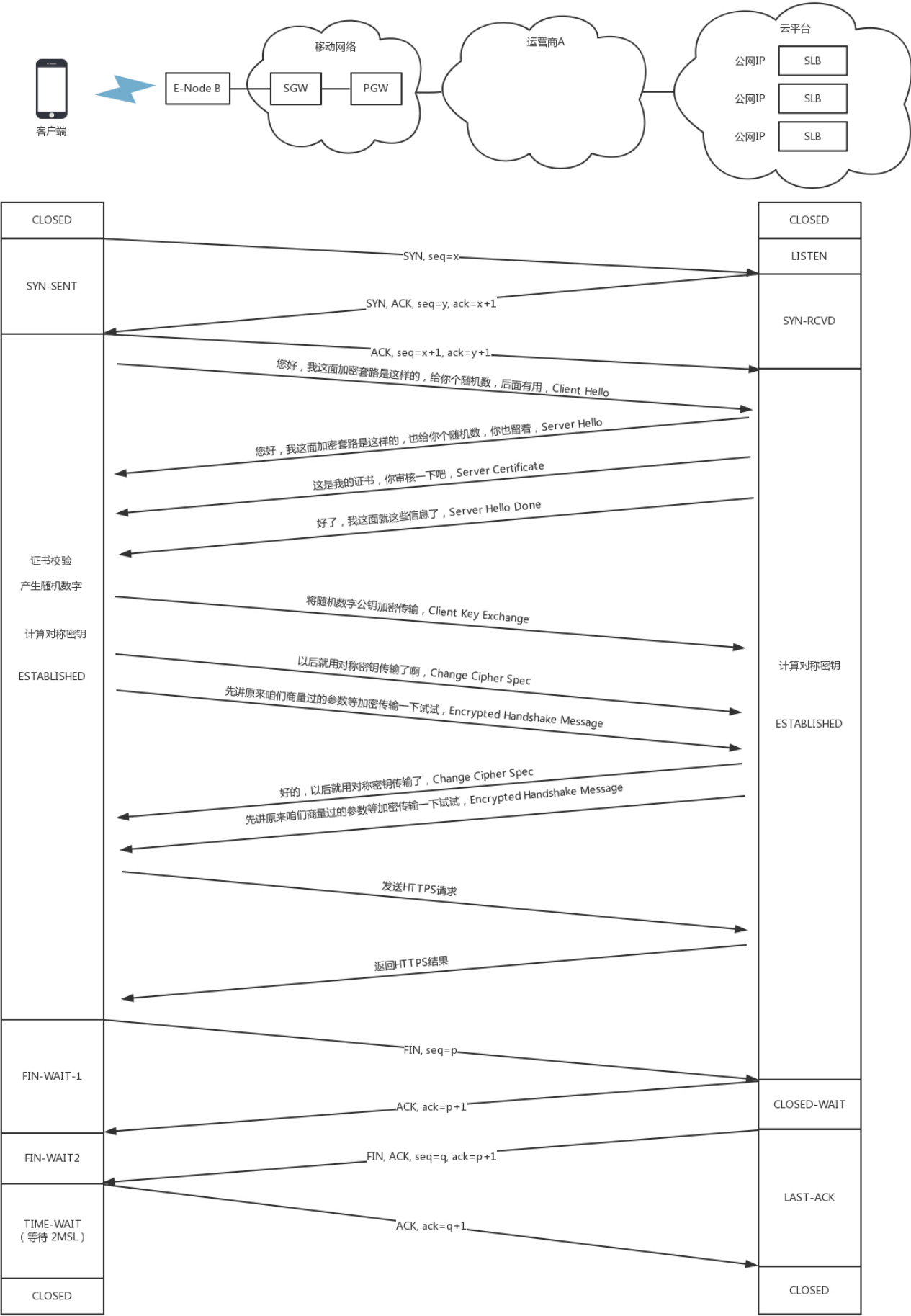
如果这台缓存服务器上并没有用户想要的内容，那么这台服务器就要向它的上一级缓存服务器请求内容，直至追溯到网站的源服务器，将内容拉到本地。

5. 看上宝贝点下单，双方开始建连接

当你浏览了很多图片，发现实在喜欢某个商品，于是决定下单购买。

电商网站会对下单的情况提供 RESTful 的下单接口，而对于下单这种需要保密的操作，需要通过 HTTPS 协议进行请求。

在所有这些操作之前，首先要做的事情是**建立连接**。



HTTPS 协议是基于 TCP 协议的，因而要先**建立 TCP 的连接**。在这个例子中，TCP 的连接是从手机上的 App 和负载均衡器 SLB 之间的。

尽管中间要经过很多的路由器和交换机，但是 TCP 的连接是端到端的。TCP 这一层和更上层的 HTTPS 无法看到中间的包的过程。尽管建立连接的时候，所有的包都逃不过在这些路由器和交换机之间的转发，转发的细节我们放到那个下单请求的发送过程中详细解读，这里只看端到端的行为。

对于 TCP 连接来讲，需要通过三次握手建立连接，为了维护这个连接，双方都需要在 TCP 层维护一个连接的状态机。

一开始，客户端和服务端都处于 CLOSED 状态。服务端先是主动监听某个端口，处于 LISTEN 状态。然后客户端主动发起连接 SYN，之后处于 SYN-SENT 状态。服务端收到发起的连接，返回 SYN，并且 ACK 客户端的 SYN，之后处于 SYN-RCVD 状态。

客户端收到服务端发送的 SYN 和 ACK 之后，发送 ACK 的 ACK，之后处于 ESTABLISHED 状态。这是因为，它一发一收成功了。服务端收到 ACK 的 ACK 之后，处于 ESTABLISHED 状态，因为它的一发一收也成功了。

当 TCP 层的连接建立完毕之后，接下来轮到**HTTPS 层建立连接**了，在 HTTPS 的交换过程中，TCP 层始终处于 ESTABLISHED。

对于 HTTPS，客户端会发送 Client Hello 消息到服务器，用明文传输 TLS 版本信息、加密套件候选列表、压缩算法候选列表等信息。另外，还会有一个随机数，在协商对称密钥的时候使用。

然后，服务器会返回 Server Hello 消息，告诉客户端，服务器选择使用的协议版本、加密套件、压缩算法等。这也有一个随机数，用于后续的密钥协商。

然后，服务器会给你一个服务器端的证书，然后说：“Server Hello Done，我这里就这些信息了。”

客户端当然不相信这个证书，于是从自己信任的 CA 仓库中，拿 CA 的证书里面的公钥去解密电商网站的证书。如果能够成功，则说明电商网站是可信的。这个过程中，你可能会不断往上追溯 CA、CA 的 CA、CA 的 CA 的 CA，反正直到一个授信的 CA，就可以了。

证书验证完毕之后，觉得这个服务端是可信的，于是客户端计算产生随机数字 Pre-master，发送 Client Key Exchange，用证书中的公钥加密，再发送给服务器，服务器可以通过私钥解密出来。

接下来，无论是客户端还是服务器，都有了三个随机数，分别是：自己的、对端的，以及刚生成的 Pre-Master 随机数。通过这三个随机数，可以在客户端和服务端产生相同的对称密钥。

有了对称密钥，客户端就可以说：“Change Cipher Spec，咱们以后都采用协商的通信密钥和加密算法进行加密通信了。”

然后客户端发送一个 Encrypted Handshake Message，将已经商定好的参数等，采用协商密钥进行加密，发送给服务器用于数据与握手验证。

同样，服务器也可以发送 Change Cipher Spec，说：“没问题，咱们以后都采用协商的通信密钥和加密算法进行加密通信了”，并且也发送 Encrypted Handshake Message 的消息试试。


当双方握手结束之后，就可以通过对称密钥进行加密传输了。

真正的下单请求封装成网络包的发送过程，我们先放一放，我们来接着讲这个网络包的故事。

6. 发送下单请求网络包，西行需要出网关

当客户端和服务端之间建立了连接后，接下来就要发送下单请求的网络包了。

在用户层发送的是 HTTP 的网络包，因为服务端提供的是 RESTful API，因而 HTTP 层发送的就是一个请求。

 复制代码

```
1 POST /purchaseOrder HTTP/1.1
2 Host: www.geektime.com
3 Content-Type: application/json; charset=utf-8
4 Content-Length: nnn
5
6 {
7   "order": {
```

```
8   "date": "2018-07-01",  
9   "className": " 趣谈网络协议 ",  
10  "Author": " 刘超 ",  
11  "price": "68"  
12 }  
13 }
```

HTTP 的报文大概分为三大部分。第一部分是**请求行**，第二部分是**请求的首部**，第三部分才是**请求的正文实体**。

在请求行中，URL 就是 www.geektime.com/purchaseOrder，版本为 HTTP 1.1。

请求的类型叫作 POST，它需要主动告诉服务端一些信息，而非获取。需要告诉服务端什么呢？一般会放在正文里面。正文可以有各种各样的格式，常见的格式是 JSON。

请求行下面就是我们的首部字段。首部是 key value，通过冒号分隔。

Content-Type 是指正文的格式。例如，我们进行 POST 的请求，如果正文是 JSON，那么我们就应该将这个值设置为 JSON。

接下来是正文，这里是一个 JSON 字符串，里面通过文本的形式描述了，要买一个课程，作者是谁，多少钱。

这样，HTTP 请求的报文格式就拼凑好了。接下来浏览器或者移动 App 会把它交给下一层传输层。

怎么交给传输层呢？也是用 Socket 进行程序设计。如果用的是浏览器，这些程序不需要你自己写，有人已经帮你写好了；如果在移动 APP 里面，一般会用一个 HTTP 的客户端工具来发送，并且帮你封装好。

HTTP 协议是基于 TCP 协议的，所以它使用面向连接的方式发送请求，通过 Stream 二进制流的方式传给对方。当然，到了 TCP 层，它会把二进制流变成一个的报文段发送给服务器。

在 TCP 头里面，会有源端口号和目标端口号，目标端口号一般是服务端监听的端口号，源端口号在手机端，往往是随机分配一个端口号。这个端口号在客户端和服务端用于区分请

求和返回，发给那个应用。

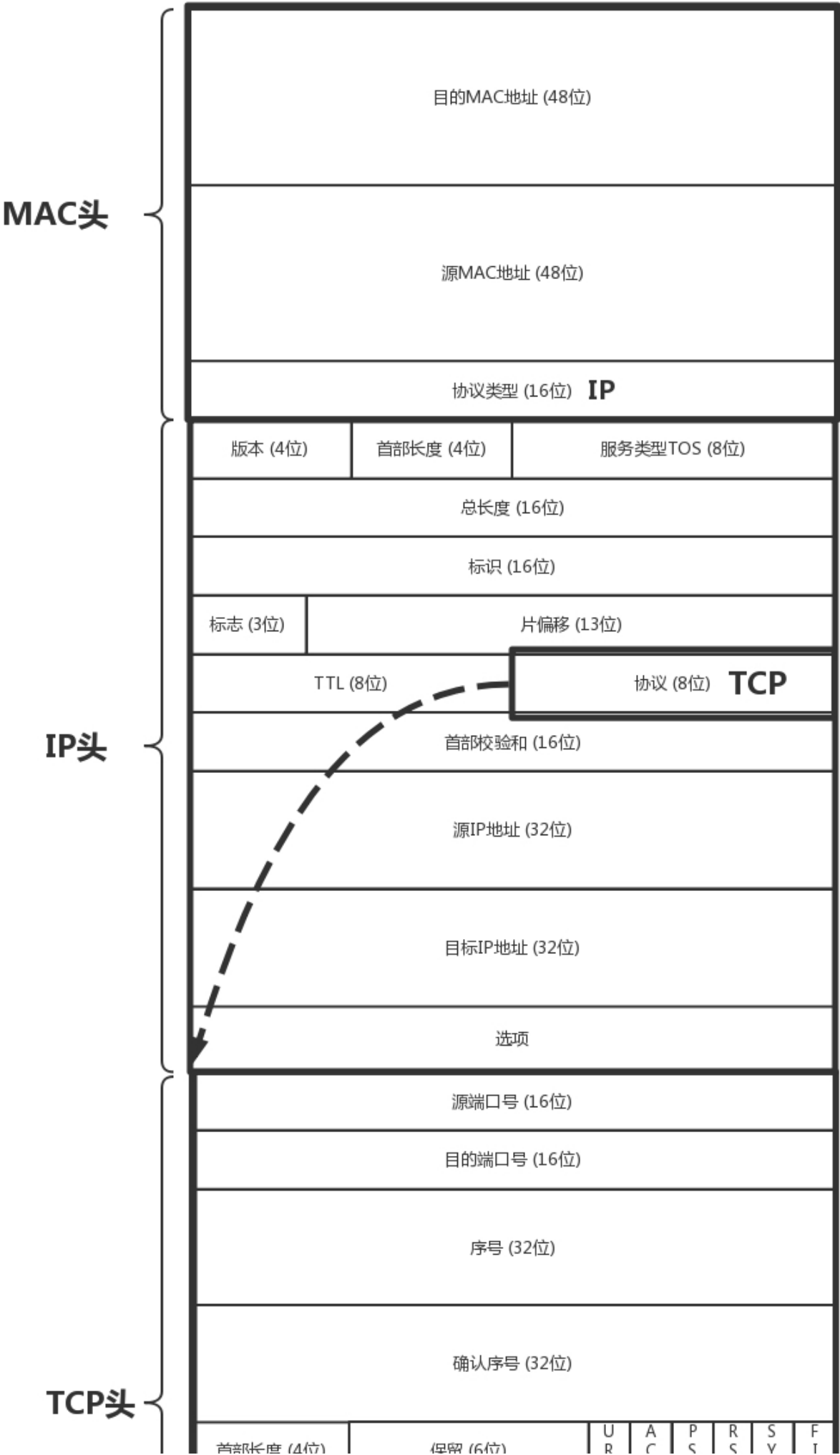
在 IP 头里面，都需要加上自己的地址（即源地址）和它想要去的地方（即目标地址）。当一个手机上线的时候，PGW 会给这个手机分配一个 IP 地址，这就是源地址，而目标地址则是云平台的负载均衡器的外网 IP 地址。

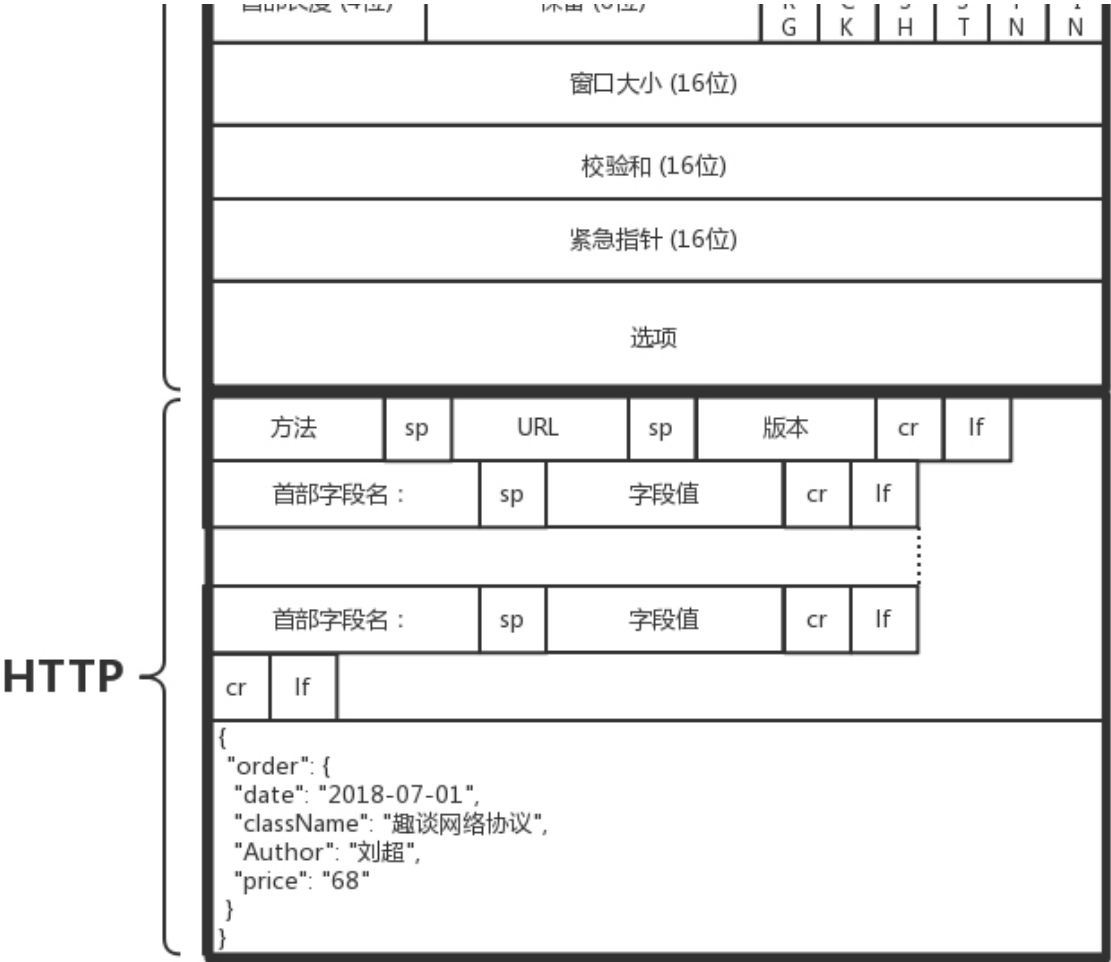
在 IP 层，客户端需要查看目标地址和自己是否是在同一个局域网，计算是否是同一个网段，往往需要通过 CIDR 子网掩码来计算。

对于这个下单场景，目标 IP 和源 IP 不会在同一个网段，因而需要发送到默认的网关。一般通过 DHCP 分配 IP 地址的时候，同时配置默认网关的 IP 地址。

但是客户端不会直接使用默认网关的 IP 地址，而是发送 ARP 协议，来获取网关的 MAC 地址，然后将网关 MAC 作为目标 MAC，自己的 MAC 作为源 MAC，放入 MAC 头，发送出去。

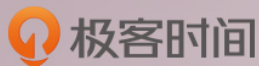
一个完整的网络包的格式是这样的。





真不容易啊，本来以为上篇就发送下单包了，结果到中篇这个包还没发送出去，只是封装了一个如此长的网络包。别着急，你可以自己先预想一下，接下来该做什么了？

欢迎你留言和我讨论。趣谈网络协议，我们下期见！



趣谈网络协议

像小说一样的网络协议入门课

刘超 网易研究院
云计算技术部首席架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得转载

上一篇 第37讲 | 知识串讲：用双十一的故事串起碎片的网络协议（上）

下一篇 第39讲 | 知识串讲：用双十一的故事串起碎片的网络协议（下）

精选留言

写留言



空档滑行

2018-08-13

4

下一步是将这个封装好的包发送给网关，网关根据路由表寻找下一跳的地址，然后把原mac和目的mac替换掉发送

展开



程启

2018-08-13

2

因为Tcp/ip协议栈是内核态，接下来客户端内核程序会发送网络包到网关，网关会再查看路由规则，这里一般是'玄奘西游型'，然后最终到达数据中心的slb。

...

展开



小宇宙

2018-08-14

👍 1

下一步就是网关进行路由寻址了，根据路由协议找到目的网关地址