## QUIC(quick UDP Internet Connections)

基于UDP的传输层协议，提供像TCP一样的可靠性。在提高web应用性能上，可以选择在应用层使用HTTP2.0实现多路传输，在物理层使用CDN解决网络拥塞和最后一公里问题。

避免前序包阻塞

零RTT建立连接

快速重启会话

**一 、 单播：**

 主机之间 **“一对一”** 的通讯模式，网络中的交换机和路由器对数据只进行转发不进行复制。如果 10 个客户机需要相同的数据，则服务器需要逐一传送，重复 10 次相同的工作。但由于其能够针对每个客户的及时响应，所以现在的网页浏览全部都是采用 IP 单播协议。网络中的路由器和交换机根据其目标地址选择传输路径，将 IP 单播数据传送到其指定的目的地。

**单播的优点：**

*1.    服务器及时响应客户机的请求*

*2.    服务器针对每个客户不通的请求发送不通的数据，容易实现个性化服务。*

**单播的缺点 ：**

1.  服务器针对每个客户机发送数据流， 服务器流量＝客户机数量×客户机流量 ；在客户数量大、每个客户机流量大的流媒体应用中服务器不堪重负。

2.   现有的网络带宽是金字塔结构，城际省际主干带宽仅仅相当于其所有用户带宽之和的 5％。如果全部使用单播协议，将造成网络主干不堪重负。现在的 P2P 应用就已经使主干经常阻塞，只要有 5 ％的客户在全速使用网络，其他人就不要玩了。而将主干扩展 20 倍几乎是不可能。

**二、  广播：**

 主机之间 **“一对所有”** 的通讯模式，网络对其中每一台主机发出的信号都进行无条件复制并转发，所有主机都可以接收到所有信息（不管你是否需要），由于其不用路径选择，所以其网络成本可以很低廉。有线电视网就是典型的广播型网络，我们的电视机实际上是接受到所有频道的信号，但只将一个频道的信号还原成画面。在数据网络中也允许广播的存在，但其被限制在二层交换机的局域网范围内，禁止广播数据穿过路由器，防止广播数据影响大面积的主机。

**广播的优点：**

1.  网络设备简单，维护简单，布网成本低廉

2.  由于服务器不用向每个客户机单独发送数据，所以服务器流量负载极低。

**广播的缺点：**

1. 无法针对每个客户的要求和时间及时提供个性化服务。

2.   网络允许服务器提供数据的带宽有限， **客户端的最大带宽＝服务总带宽** 。例如有线电视的客户端的线路支持 100 个频道（如果采用数字压缩技术，理论上可以提供 500 个频道），即使服务商有更大的财力配置更多的发送设备、改成光纤主干，也无法超过此极限。也就是说无法向众多客户提供更多样化、更加个性化的服务。

3.   广播禁止在 Internet 宽带网上传输。

**三、 组播：**

        主机之间 **“一对一组”** 的通讯模式，也就是加入了同一个组的主机可以接受到此组内的所有数据，网络中的交换机和路由器只向有需求者复制并转发其所需数据。主机可以向路由器请求加入或退出某个组，网络中的路由器和交换机有选择的复制并传输数据，即只将组内数据传输给那些加入组的主机。这样既能一次将数据传输给多个有需要（加入组）的主机，又能保证不影响其他不需要（未加入组）的主机的其他通讯。

**组播的优点：**

1.   需要相同数据流的客户端加入相同的组共享一条数据流，节省了服务器的负载。具备广播所具备的优点。

2.   由于组播协议是根据接受者的需要对数据流进行复制转发，所以服务端的服务总带宽不受客户接入端带宽的限制。 IP 协议允许有 2 亿 6 千多万个（ 268435456 ）组播，所以其提供的服务可以非常丰富。

3.   此协议和单播协议一样允许在 Internet 宽带网上传输。

**组播的缺点：**

1． 与单播协议相比没有纠错机制，发生丢包错包后难以弥补，但可以通过一定的容错机制和QOS 加以弥补。

2． 现行网络虽然都支持组播的传输，但在客户认证、 QOS 等方面还需要完善，这些缺点在理论上都有成熟的解决方案，只是需要逐步推广应用到现存网络当中。

TCP/IP协议

1.1七层协议

物理层: 为上层传输提供了物理媒介。 数据-bit比特,是通信端点的硬件接口。

数据链路层:在物理层的基础上提供可靠的数据传输，这一层利用通信信道实现无差错传输。提供物理寻址、数据成帧、流量控制以及链路控制。

网络层：负责各个子网之间数据的路由选择，提供网络互连、流量控制、拥塞控制等功能。 数据包。

传输层：数据成段，提供可靠或者不可靠的服务；

会话层：管理主机之间的会话过程

表示层：对网络传输的数据进行格式变换，使得主机之间传输的信息能够互相理解。包括数据的压缩、解压、格式转换。

应用层：给应用程序提供网络服务的接口

### 大小端

大端模式：

是指数据的高字节保存在内存的低地址中，而数据的低字节保存在内存的高地址中，这样的存储模式有点儿类似于把数据当作字符串顺序处理：地址由小向大增加，而数据从高位往低位放；这和我们的阅读习惯一致。网络数据传输使用大端

小端模式：

是指数据的高字节保存在内存的高地址中，而数据的低字节保存在内存的低地址中，这种存储模式将地址的高低和数据位权有效地结合起来，高地址部分权值高，低地址部分权值低。

### Wireshark

### tcpdump

Linux抓包是通过注册一种虚拟的底层网络协议来完成对网络报文(准确的说是网络设备)消息的处理权。当网卡接收到一个网络报文之后，它会遍历系统中所有已经注册的网络协议，例如以太网协议、x25协议处理模块来尝试进行报文的解析处理，这一点和一些文件系统的挂载相似，就是让系统中所有的已经注册的文件系统来进行尝试挂载，如果哪一个认为自己可以处理，那么就完成挂载。

当抓包模块把自己伪装成一个网络协议的时候，系统在收到报文的时候就会给这个伪协议一次机会，让它来对网卡收到的报文进行一次处理，此时该模块就会趁机对报文进行窥探，也就是把这个报文完完整整的复制一份，假装是自己接收到的报文，汇报给抓包模块。

Tcpdump ：

监视指定网络的数据包：tcpdump –i eth0

监视指定主机的数据包：

监视所有进入或者离开192.168.0.100的数据包：tcpdump host 192.168.0.100

截获主机192.168.0.100 和主机192.168.0.101或者192.168.0.102的通信：

Tcpdump host 192.168.0.1 and \(192.168.0.101 or 192.168.0.102)  
获取192.168.0.100和除了192.168.0.101之外的所有主机通信的IP包：

Tcpdump 192.168.0.100 and ! 192.168.0.101

获取主机192.168.0.101 发送的所有数据：

Tcpdump –i eth0 src host 192.168.0.101

获取所有发送到主机192.168.0.101的数据包

Tcpdump –i eth0 dst host 192.168.0.101

获取主机192.168.0.101接收的udp port 10002

Tcpdump –i eth0 udp port 10002 and

Tcpdump -n(显示ip和端口)

### 长连接短连接

之所以说HTTP分为长连接和短连接，其实本质上是说的TCP连接。TCP连接是一个双向的通道，它是可以保持一段时间不关闭的，因此TCP连接才有真正的长连接和短连接这一说。HTTP协议说到底是应用层的协议，而TCP才是真正的传输层协议，只有负责传输的这一层才需要建立连接。因此“HTTP连接”这一概念压根就不应该出现，HTTP只是一个应用层的协议，根本就没有连接这一说法，就像FTP协议一样，我们从来不会说“FTP连接”吧。归根到底，其实说的连接都是只传输层的TCP连接。相反说HTTP请求和HTTP响应反而更加准确一些，都是通过TCP连接这个数据通道来传输请求和响应的。

长连接，短连接都是指的传输层的TCP连接，而不是应用层的HTTP协议。HTTP属于应用层协议，在传输层使用TCP协议，在网络层使用IP协议。IP协议主要解决网络路由和寻址问题，TCP协议主要解决如何在IP层之上可靠的传递数据包，使得网络上接收端收到发端发出的所有包，并且顺序与发出顺序一致。TCP有可靠，面向连接的特点。

HTTP协议是无状态的面向连接的协议。

### TCP实现长连接

### Shell

变量

支持环境变量、内部变量、用户变量。变量时非类型属性的，不必规定变量的类型。环境变量和内部变量不用定义。

位置参数：

|  |  |
| --- | --- |
| $0 | 相当于C语言main函数的argv[0] |
| $1、$2... | 这些称为位置参数（Positional Parameter），相当于C语言main函数的argv[1]、argv[2]... |
| $# | 相当于C语言main函数的argc - 1，注意这里的#后面不表示注释 |
| $@ | 表示参数列表"$1" "$2" ...，例如可以用在for循环中的in后面。 |
| $? | 上一条命令的Exit Status |
| $$ | 当前Shell的进程号 |

For的用法：for i in xxx

### Elastic Stack

Elasticsearch是一个分布式、面向文档的搜索和数据分析引擎，支持结构化和非结构化的查询，通常用于web日志分析、实施应用监控和点击流分析。