UDP

用户数据报协议（英语：User Datagram Protocol，缩写：UDP；又称用户数据包协议）是一个简单的面向数据报的通信协议，位于OSI模型的传输层。

在TCP/IP模型中，UDP为网络层以上和应用层以下提供了一个简单的接口。UDP只提供数据的不可靠传递，它一旦把应用程序发给网络层的数据发送出去，就不保留数据备份（所以UDP有时候也被认为是不可靠的数据报协议）。UDP在IP数据报的头部仅仅加入了复用和数据校验字段。

UDP适用于不需要或在程序中执行错误检查和纠正的应用，它避免了协议栈中此类处理的开销。对时间有较高要求的应用程序通常使用UDP，因为丢弃数据包比等待或重传导致延迟更可取。

应用

- 域名系统（DNS），其中查询阶段必须快速，并且只包含单个请求，后跟单个回复数据包
- 动态主机配置协议（DHCP），用于动态分配IP地址
- 简单网络管理协议（SNMP）
- 路由信息协议（RIP）
- 语音和视频流量通常使用UDP传输

结构

UDP报头

偏移	字节	0								1								2								3							
字节	位	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	来源连接端口																目的连接端口														
4	32	报文长度																校验和														

- 在IPv4中，“来源连接端口”和“校验和”是可选字段。在IPv6中，只有来源连接端口是可选字段
- 报文长度：该字段指定UDP报头和数据总共占用的长度。可能的最小长度是8字节，因为UDP报头已经占用了8字节。由于这个字段的存在，UDP报文总长不可能超过65535字节（包括8字节的报头，和65527字节的数据）。实际上通过IPv4协议传输时，由于IPv4的头部信息要占用20字节，因此数据长度不可能超过65507字节（65,535 − 8字节UDP报头 − 20字节IP头部）
- 校验和：用于发现头部信息和数据中的传输错误。该字段在IPv4中是可选的，在IPv6中则是强制的。如果不使用校验和，该字段应被填充为全0。

TCP

传输控制协议（英语：Transmission Control Protocol，缩写：TCP）是一种面向连接的、可靠的、基于字节流的传输层通信协议。

在因特网协议族（Internet protocol suite）中，TCP层是位于IP层之上，应用层之下的中间层。不同主机的应用层之间经常需要可靠的、像管道一样的连接，但是IP层不提供这样的流机制，而是提供不可靠的包交换。

TCP协议的运行可划分为三个阶段：

1. 连接创建
2. 数据传送
3. 连接终止

三次握手与四次挥手

所谓三次握手(Three-way Handshake)，是指建立一个 TCP 连接时，需要客户端和服务器总共发送3个包。

三次握手的目的是连接服务器指定端口，建立 TCP 连接，并同步连接双方的序列号和确认号，交换 TCP 窗口大小信息。在 socket 编程中，客户端执行 `connect()` 时。将触发三次握手。

- 第一次握手(SYN=1, seq=x):

  客户端发送一个 TCP 的 SYN 标志位置1的包，指明客户端打算连接的服务器的端口，以及初始序号 X,保存在包头的序列号(Sequence Number)字段里。

  发送完毕后，客户端进入 `SYN_SEND` 状态。

- 第二次握手(SYN=1, ACK=1, seq=y, ACKnum=x+1):

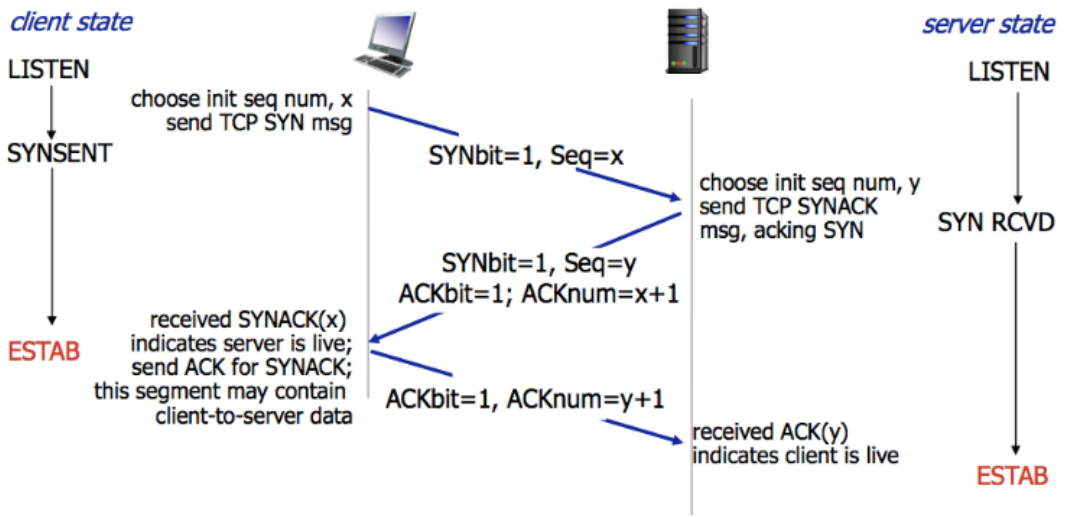
  服务器发回确认包(ACK)应答。即 SYN 标志位和 ACK 标志位均为1。服务器端选择自己 ISN 序列号，放到 Seq 域里，同时将确认序号(Acknowledgement Number)设置为客户的 ISN 加1，即 X+1。 发送完毕后，服务器端进入 `SYN_RCVD` 状态。

- 第三次握手(ACK=1，ACKnum=y+1)

  客户端再次发送确认包(ACK)，SYN 标志位为0，ACK 标志位为1，并且把服务器发来 ACK 的序号字段+1，放在确定字段中发送给对方，并且在数据段放写ISN的+1

  发送完毕后，客户端进入 `ESTABLISHED` 状态，当服务器端接收到这个包时，也进入 `ESTABLISHED` 状态，TCP 握手结束。

三次握手的过程的示意图如下：



TCP 的连接的拆除需要发送四个包，因此称为四次挥手(Four-way handshake)，也叫做改进的三次握手。客户端或服务器均可主动发起挥手动作，在 socket 编程中，任何一方执行 `close()` 操作即可产生挥手操作。

- 第一次挥手(FIN=1，seq=x)

假设客户端想要关闭连接，客户端发送一个 FIN 标志位置为1的包，表示自己已经没有数据可以发送了，但是仍然可以接受数据。

发送完毕后，客户端进入 `FIN_WAIT_1` 状态。

- 第二次挥手(ACK=1，ACKnum=x+1)

  服务器端确认客户端的 FIN 包，发送一个确认包，表明自己接受到了客户端关闭连接的请求，但还没有准备好关闭连接。

  发送完毕后，服务器端进入 `CLOSE_WAIT` 状态，客户端接收到这个确认包之后，进入 `FIN_WAIT_2` 状态，等待服务器端关闭连接。

- 第三次挥手(FIN=1，seq=y)

  服务器端准备好关闭连接时，向客户端发送结束连接请求，FIN 置为1。

  发送完毕后，服务器端进入 `LAST_ACK` 状态，等待来自客户端的最后一个ACK。

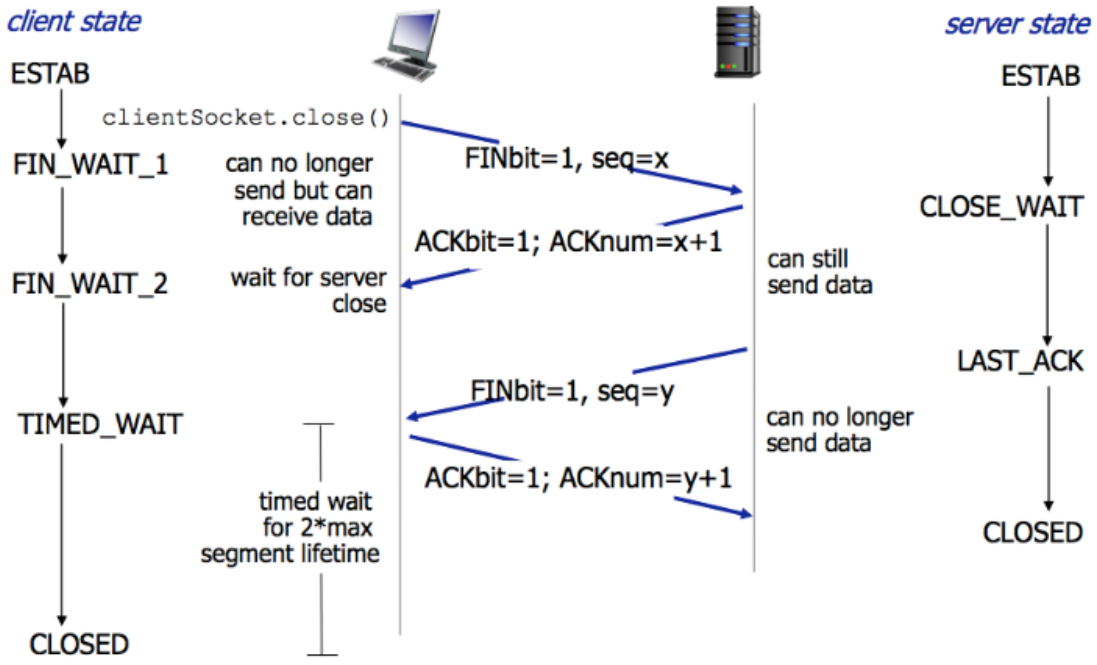
- 第四次挥手(ACK=1，ACKnum=y+1)

  客户端接收到来自服务器端的关闭请求，发送一个确认包，并进入 `TIME_WAIT` 状态，等待可能出现的要求重传的 ACK 包。

  服务器端接收到这个确认包之后，关闭连接，进入 `CLOSED` 状态。

  客户端等待了某个固定时间（两个最大段生命周期，2MSL，2 Maximum Segment Lifetime）之后，没有收到服务器端的 ACK ，认为服务器端已经正常关闭连接，于是自己也关闭连接，进入 `CLOSED` 状态。

四次挥手的示意图如下：



结构

TCP表头

偏移	字节	0								1								2								3							
字节	比特	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	来源连接端口																目的连接端口														
4	32	序列号码																															
8	64	确认号码（当ACK设置）																															
12	96	数据偏移				保留 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	窗口大小														
16	128	校验和																紧急指针（当URG设置）														
20 ...	160 ...	选项（如果数据偏移 > 5。需要在结尾添加0。 ...																															