**多播**（英语：**multicast**，台湾又译作**多点发送**、**多点广播**或**群播**，中国大陆又译作**组播**）是指把信息同时传递给一组目的地址。它使用策略是最高效的，因为消息在每条网络链路上只需传递一次，而且只有在链路分叉的时候，消息才会被复制。与多播相比，常规的点到单点的传递被称作[单播](https://zh.wikipedia.org/wiki/%E5%96%AE%E6%92%AD)。当以单播的形式把消息传递给多个接收方时，必须向每个接收者都发送一份数据副本。

1. **传输控制协议**（英语：Transmission Control Protocol，缩写为 TCP）是一种面向连接的、可靠的、基于[字节流](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AF%80%E6%B5%81)的[传输层](https://zh.wikipedia.org/wiki/%E4%BC%A0%E8%BE%93%E5%B1%82)通信协议，由[IETF](https://zh.wikipedia.org/wiki/IETF)的RFC 793定义。在简化的计算机网络[OSI模型](https://zh.wikipedia.org/wiki/OSI%E6%A8%A1%E5%9E%8B)中，它完成第四层传输层所指定的功能，[用户数据报协议](https://zh.wikipedia.org/wiki/%E7%94%A8%E6%88%B7%E6%95%B0%E6%8D%AE%E6%8A%A5%E5%8D%8F%E8%AE%AE)（UDP）是同一层内另一个重要的传输协议。
2. 在因特网协议族（Internet protocol suite）中，TCP层是位于[IP](https://zh.wikipedia.org/wiki/IP)层之上，[应用层](https://zh.wikipedia.org/wiki/%E5%BA%94%E7%94%A8%E5%B1%82)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](https://zh.wikipedia.org/wiki/%E7%AE%A1%E9%81%93_(Unix))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。
3. 应用层向TCP层发送用于网间传输的、用8位字节表示的数据流，然后TCP把数据流分区成适当长度的报文段（通常受该计算机连接的网络的数据链路层的[最大传输单元](https://zh.wikipedia.org/wiki/%E6%9C%80%E5%A4%A7%E4%BC%A0%E8%BE%93%E5%8D%95%E5%85%83)（MTU）的限制）。之后TCP把结果包传给IP层，由它来通过网络将包传送给接收端实体的TCP层。TCP为了保证不发生丢包，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的包发回一个相应的确认（[ACK](https://zh.wikipedia.org/wiki/ACK)）；如果发送端实体在合理的[往返时延](https://zh.wikipedia.org/wiki/%E4%BE%86%E5%9B%9E%E9%80%9A%E8%A8%8A%E5%BB%B6%E9%81%B2)（RTT）内未收到确认，那么对应的数据包就被假设为[已丢失](https://zh.wikipedia.org/wiki/%E4%B8%A2%E5%8C%85)将会被进行重传。TCP用一个[校验和](https://zh.wikipedia.org/wiki/%E6%A0%A1%E9%AA%8C%E5%92%8C)函数来检验数据是否有错误；在发送和接收时都要计算校验和

**用户数据报协议**（英语：User Datagram Protocol，缩写为UDP），又称**用户数据报文协议**，是一个简单的面向数据报的[传输层](https://zh.wikipedia.org/wiki/%E4%BC%A0%E8%BE%93%E5%B1%82)协议，正式规范为RFC 768。

在TCP/IP模型中，UDP为[网络层](https://zh.wikipedia.org/wiki/%E7%BD%91%E7%BB%9C%E5%B1%82)以上和[应用层](https://zh.wikipedia.org/wiki/%E5%BA%94%E7%94%A8%E5%B1%82)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首部字段由4个部分组成，其中两个是可选的。各16[bit](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83)的来源端口和目的端口用来标记发送和接受的应用进程。因为UDP不需要应答，所以来源端口是可选的，如果来源端口不用，那么置为零。在目的端口后面是长度固定的以字节为单位的长度域，用来指定UDP数据报包括数据部分的长度，长度最小值为8byte。首部剩下地16bit是用来对首部和数据部分一起做[校验和](https://zh.wikipedia.org/wiki/%E6%A0%A1%E9%AA%8C%E5%92%8C)（Checksum）的，这部分是可选的，但在实际应用中一般都使用这一功能。

由于缺乏可靠性且属于非连接导向协议，UDP应用一般必须允许一定量的丢包、出错和复制粘贴。但有些应用，比如[TFTP](https://zh.wikipedia.org/wiki/TFTP)，如果需要则必须在应用层增加根本的可靠机制。但是绝大多数UDP应用都不需要可靠机制，甚至可能因为引入可靠机制而降低性能。[流媒体](https://zh.wikipedia.org/wiki/%E6%B5%81%E5%AA%92%E9%AB%94)（流技术）、即时多媒体游戏和[IP电话](https://zh.wikipedia.org/wiki/IP%E7%94%B5%E8%AF%9D)（VoIP）一定就是典型的UDP应用。如果某个应用需要很高的可靠性，那么可以用[传输控制协议](https://zh.wikipedia.org/wiki/%E4%BC%A0%E8%BE%93%E6%8E%A7%E5%88%B6%E5%8D%8F%E8%AE%AE)（TCP协议）来代替UDP。

由于缺乏拥塞控制（congestion control），需要基于网络的机制来减少因失控和高速UDP流量负荷而导致的拥塞崩溃效应。换句话说，因为UDP发送者不能够检测拥塞，所以像使用包队列和丢弃技术的路由器这样的网络基本设备往往就成为降低UDP过大通信量的有效工具。[数据报拥塞控制协议](https://zh.wikipedia.org/wiki/DCCP)（DCCP）设计成通过在诸如流媒体类型的高速率UDP流中，增加主机拥塞控制，来减小这个潜在的问题。

典型网络上的众多使用UDP协议的关键应用一定程度上是相似的。这些应用包括[域名系统](https://zh.wikipedia.org/wiki/%E5%9F%9F%E5%90%8D%E7%B3%BB%E7%BB%9F)（DNS）、[简单网络管理协议](https://zh.wikipedia.org/wiki/%E7%AE%80%E5%8D%95%E7%BD%91%E7%BB%9C%E7%AE%A1%E7%90%86%E5%8D%8F%E8%AE%AE)（SNMP）、[动态主机配置协议](https://zh.wikipedia.org/wiki/%E5%8A%A8%E6%80%81%E4%B8%BB%E6%9C%BA%E9%85%8D%E7%BD%AE%E5%8D%8F%E8%AE%AE)（DHCP）、[路由信息协议](https://zh.wikipedia.org/wiki/%E8%B7%AF%E7%94%B1%E4%BF%A1%E6%81%AF%E5%8D%8F%E8%AE%AE)（RIP）和某些影音流服务等等。