传输层

面向用户的最底层，面向通讯的最高层。只有位于网络边缘的主机的协议栈才有运输层。
运输层提供的应用进程到应用进程的通信（端到端）。

传输层的功能

连通每个计算机的通信模式。实现**流量控制、可靠传输、拥塞控制**功能。
提供了TCP或UDP报文的复用和分用，根据端口号提供复用与分用的功能。

- TCP：连接的可靠的全双工信道。不支持多播、广播，用于大多数应用
- UDP：无连接的不可靠的信道。支持单播、多播、广播

两个对等运输实体在通信时传送的数据单位是运输协议数据单元TPDU。TCP传输TCP报文段，UDP传输UDP报文或用户数据段。

端口号

对TCP/IP体系的应用程序进行标志。把要传输报文交到目的主机的某一个合适的目的端口，最后交付目的进程由TCP完成。

端口是16位，允许65535个不同端口号。端口号仅具有标记本地计算机应用层各进程的作用。

常用端口：熟知端口0~1024；登记端口1024~49151，使用时需在IANA登记防止重复；客户端端口49152~65535，短暂端口号，给用户进程暂时使用。
常用熟知端口：
UDP：RPC111, DNS53, TFTP69, SNMP161, SNMP(trap)162;
TCP：SMTP25, FTP20、21, Telnet23, HTTP80, HTTPS443

UDP协议

UDP仅在IP数据报服务上增加了 复用/分用、差错检测 功能。
其中复用使用的是源端口，分用使用的是目的端口。

UDP是无连接的，发送数据之前不建立连接，减少开销和发送数据之前的时延。提供面向报文的不可靠交付，一次性交付一个完整报文。没有拥塞控制。支持一对一、一对多、多对一、多对多的交互通信。首部开销小，只有8字节：
|源端口 |目的端口 |长度 |校验和 |;
由于一次性交付完整报文，应用程序必须选择合适的大小。

基于端口的分用：UDP数据报到达时，根据首部中的目的端口分发给一个或多个端口。

UDP校验和：把首部和数据部分仪器检验。

TCP协议

面向连接的运输层协议，在无连接、不可靠的IP网络服务基础上提供可靠交付的服务。在IP数据报服务商增加了保证可靠性的一系列措施。
面向连接、面向字节流、点对点、可靠交付、全双工通信。

面向字节流：把应用程序交付的数据看成一连串无结构的字节流。每次发送的都是一个数据段，每个段都将会编号排序，对数据顺序进行控制，当其中一个数据段丢失将会重发。

TCP是基于socket的虚连接，而非物理连接；不关心应用进程的报文发送到TCP缓冲的长度；根据对方给出的窗口值和当前网络拥塞程度决定一个报文段应包含多少字节；可把过长的数据块划分变短后传送，也可积累足够多的字节后再构成报文段发送。

TCP的连接

TCP连接的端点是套接字socket，端口号拼接到IP地址构成套接字。每条TCP连接唯一地被通信两端的两个端点/套接字所确定。即：`TCP连接::= {socket1, socket2} = {(IP1:port1),(IP2:port2)}`

TCP首部格式

TCP报文段首部的前20个字节固定，后面4n个字节根据需要而增加。因此最小长度是20字节。

```
| 源端口16                    | 目的端口16                   |
| 序号32                                                    |
| 确认号32                                                  |
|数据偏移4 |保留6 |URG1|ACK1|PSH1|RST1|SYN1|FIN1 |窗口16 |
| 校验和16                    | 紧急指针16                   |
| 选项（长度可变）                                 |填充   |
```

- 源端口和目的端口：运输层与应用层的u无接口，复用/分用都要通过端口实现
- 序号：传送数据流中的序号
- 确认号：期望收到对方下一个报文段的数据的第一个字节的序号
- 数据偏移：报文段的数据起始处距TCP报文段的起始处的距离
- 保留：今后使用，目前为0
- URG=1时有高优先级；ACK=1时确认字段有效；PSH=1时会尽快交付接收应用进程，不再等到缓存满后在上交；RST=1时表示TCP连接出现严重差错，必须释放连接；SYN=1时表示这个是一个连接请求或连接接受报文；FIN=1时表示此报文段的发送端数据已发送完毕，并要求释放运输链接
- 窗口：让对方设置发送窗口的依据
- 检验和：检验和字段检验的范围包括首部和数据这两部分
- 紧急指针：报文段中紧急数据的字节数量
- 选项字段：常用的选项是MSS，它告诉对方TCP发送方的报文段的数据字段最大长度为MSS个字节
- 其它：
    - 窗口扩大选项3：其中一个字节表示移位值S，新窗口值=TCP首部中的窗口位数增大到(16+S)，相当于窗口值向左移动S后获得的实际窗口大小
    - 时间戳选项4：字段时间戳值字段，时间戳回送回答字段
    - 选择确认选项
- 填充：让整个首部长度为4字节的整数倍

TCP可靠传输

目前唯一的*可靠的*网络协议，面相连接的传输服务。

可靠传输的机制：
编号和确认机制；超时机制；自动重传机制。其中会有两种不同的协议：停止等待协议，连续ARQ协议。

> 停止等待机制：每发送一个分组就停止发送并等待对方确认，确认后再发下一个分组。
>
> 连续ARQ协议：发送方一次发出多个分组。使用滑动窗口协议控制发送方和接收方所传播分组的数量和编号，发送方每收到一个确认就把发送窗口向前滑动，接收方一般采用累计确认的方式。重传时使用回退N `Go-Back-N` 方法进行重传。

TCP拥塞控制

拥塞是指某短时间内网络中某资源的需求超出了该资源所能提供的可用部分，使网络性能变坏，最坏结果是系统崩溃。
拥塞的原因有：缓存容量太小，链路容量不足，处理机处理速率太慢，拥塞本身进一步加剧拥塞。

拥塞控制和**流量控制**是不同的。流量控制是一直发送端发送的速率，是点对点通信量的控制；拥塞控制是一个全局性的过程，涉及与降低网络传输性能的所有因素。

有两种控制类型：

- **开环控制**：设计网络时事先考虑周全，力求工作时不拥塞，力争避免发生拥塞
- **闭环控制**：根据网络当前状态采取相应控制措施，发生拥塞后采取措施进行控制已达到消除拥塞的目的

TCP拥塞控制的基本概念：
一个传输轮次所经历的时间是往返时间RTT。传输轮次更加强调把拥塞窗口cwnd所允许发送的报文段都连续发送出去，并受到了对已发送的最后一个字节的确认。

控制拥塞窗口的原则：
只要网络没有出现拥塞，拥塞窗口就可以再增大一些以便发送更多分组，提高网络利用率。出现拥塞或可能出现拥塞时，则反之来缓解拥塞。

拥塞判断方式：重传定时器超时，收到三个重复的ACK。

拥塞控制算法：

- **慢开始slowStart**：用于确定网络的负载能力或拥塞程度，由小到大逐渐增加拥塞窗口数值，两个变量：拥塞窗口（1MSS逐渐增大），慢开始门限
  - 拥塞窗口cwnd控制方法：没收到一个对新报文段的确认后，可以把拥塞窗口最多增加一个SMSS数值。`拥塞窗口cwnd每次增加量 = min(N, SMSS)`
  - 每经过一个传输轮次，拥塞窗口加倍，指数增长，cwnd+1
- **拥塞避免算法**（已经超时时使用）：让拥塞窗口cwnd缓慢增大，避免拥塞
  - 每经过一个传输轮次，`拥塞窗口cwnd = cwnd+1`，线性增长
  - 拥塞避免并非指完全避免了拥塞，仅仅是把拥塞窗口控制为按现行规律增长来尽量不容易出现拥塞
  - 当拥塞出现时：`ssThresh = max(cwnd/2, 2), cwnd = 1, 执行慢开始算法`。当拥塞窗口cwnd增长到慢开始门限值ssThresh时，就改为拥塞避免算法
- **快重传算法fastRetransmission**：发送方一连收到三个重复确认就知道接收方没有收到报文段，因此立刻重传，避免出现超时使发送方误认为出现网络拥塞
- **快速恢复算法fastRecovery**：发送端收到连续三个重复确认，认为网络很可能没有发生拥塞，因此执行快速回复算法而不是慢开始算法：

    - 慢开始门限ssThresh = 当前拥塞窗口cwnd / 2，新拥塞窗口cwnd = 慢开始门限ssThresh，开始执行拥塞避免算法

发送窗口的上限值：发送窗口的上限值应当取位接收方窗口rwnd和拥塞窗口cwnd这两个变量中较小者，按 发送窗口上限值 = min(rwnd, cwnd) 来确定

TCP连接管理

注：下面的+1表示加一个有效载荷。

连接建立的三个阶段（三次握手）：客户端C向服务器S发出请求连接的报文段，S收到后若同意则发回确认，C收到报文段后向S给出确认。此时C的TCP向上层应用进程通知连接已建立。
整体过程描述如下：

```
C                                    S
| ----------- SYN=1, seq=x ----------> |
| <-- SYN=1, ACK=1, seq=y, ack=x+1 --- |
| ----- ACK=1, seq=x+1, ack=y+1 -----> |
```

连接释放：C的应用进程向TCP发出连接释放报文段，主动关闭TCP连接，S发出确认。此时C到S的连接已释放，TCP连接处于半关闭状态。若S发送数据，C仍要接收；若S没有需要向C发送数据，其应用进程通知TCP释放连接。C收到释放报文段后发出确认。
整体过程描述如下：

```
C                                    S
| ----------- FIN=1, seq=u ----------> |
| <----- ACK=1, seq=v, ack=u+1 ------- |
| <-- FIN=1, ACK=1, seq=w, ack=u+1 --- |
| ----- ACK=1, seq=u+1, ack=w+1 -----> |
```

TCP连接必须经过时间2MSL后才真正释放掉。MSL是报文最大生存时间。所以最后C发出确认报文段时需要等待2MSL。

数据交换格式

- **电路交换**
    - 两部电话机用一对电线相互连接，“交换”就是按照某种方式分配专用的物理线路
    - 必须是面向连接的，三个阶段包括：建立连接，通信，释放连接
    - *优点*：用户专用，随时通信，实时性强，无时序问题，适用于模拟和数字信号，控制简单
    - *缺点*：建立连接时间较长，因为专用而利用率低，不同标准的中断难以通信，仅适合语音、不适合数据通信
- **报文交换**
    - 交换单位是携带有目标地址、源地址的报文，在交换节点采用存储转发的方式

- *优点*：通讯双方不需要建立专用线路，不存在连接建立时延，随时发送报文。采用存储转发提高了传输时的可靠性，易实现代码转换和速率匹配（便于不同标准的计算机之间通信）；提供多目标服务；允许建立数据传输的优先级。通信双方在不同时段分段占用物理通路，提高了线路利用率
- *缺点*：存储转发的时延大；只适用于数字信号；报文长度没有限制，每个中间结点都要完整报文

- **分组交换**
  - 在报文交换的基础上限制每次传送数据块的大小，并加上必要的控制信息(首部)构成分组（IP分组)
  - 每个分组都携带地址等控制信息，交换网中的节点交换机根据收到分组的首部来转发给下一节点，并采用存储转发的方式让所有分组到达目的地
  - *两种实现方式*:
    - 数据报：无连接方式，类似报文交换，每个分组都携带目的信息，在发送时每个分组可能走不同路径使得具有不同时延(多数是通顺序的)，但最终都会到达目的地点
    - 虚电路：面向连接，类似线路交换，建立虚连接，传输路径相对确定地从一个节点到下一节点不断存储转发，直到到达目的地。不存在数据报方式的排序问题，但需要建立呼叫请求建立路线

补充

- UDP协议没有全双工的说法
- UDP校验出错时直接将数据丢弃
- 发送窗口始终 = min(接收窗口rwnd, 拥塞窗口cwnd)