西北工業大學

计算机网络

Computer Networking

软件学院 **钱锋** 编

2024 年 4 月 14 日

计算机网络

钱锋[1,2]

2024 年 4 月 14 日

[1]Email: strik0r.qf@gmail.com
[2]西北工业大学软件学院, School of Software, Northwestern Polytechnical University, 西安 710072

目录

第 1 章　计算机网络和因特网

1.1　计算机网络概述

1.1.1　计算机网络的概念和功能

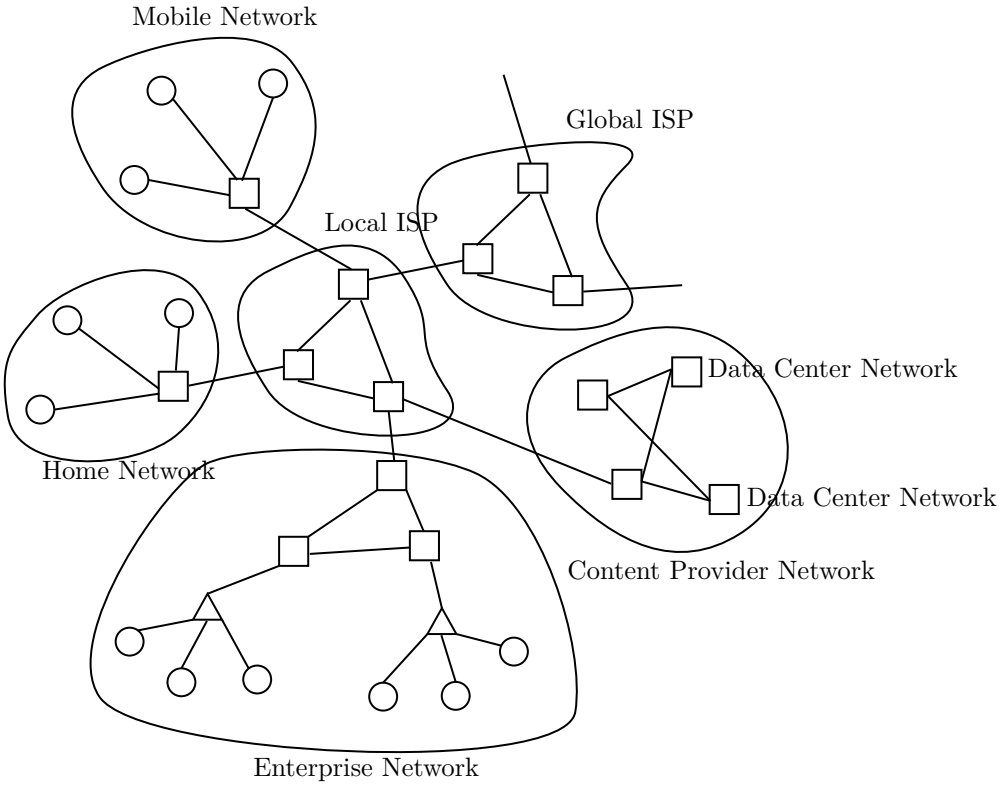


图 1.1: 计算机网络与因特网

计算机网络是一个将一系列分散的、具有独立功能的计算机系统通过通信设备与线路连接起来, 由功能完善的软件实现资源共享和信息传递的系统. 简而言之, 计算机网络就是一系列互连的 (即计算机网络中的各台计算机是通过通信链路相互连接起来的)、自治的 (即计算机网络中的各台计算机是没有主从关系的) 计算机系统的集合. 计算机网络的功能主要有以下五个方面:

- 数据通信: 计算机网络最基本和最重要的功能, 用来实现联网计算机之间各种信息的传输, 并将分散在不同地理位置的计算机联系起来, 进行统一的调配、控制和管理.

- 资源共享: 同一个计算机网络中的计算机可以互相访问彼此的计算机资源, 这里共享的计算机资源可以是软件资源、数据资源, 也可以是计算机硬件资源.
- 分布式处理: 多台计算机各自承担统一工作的不同部分.
- 提高可靠性: 计算机网络中的各台计算机互为替代机.
- 负载均衡: 将工作任务均衡的分配给计算机网络中的各台计算机.

例 1.1.1. 在神经网络与深度学习高度发达的今天, 昂贵的 GPU 价格让许多用户选择了将自己的深度学习代码放到云端的计算机上来运行, 使用云端的专用高性能计算设备的 GPU 来完成深度学习模型的训练, 这就是典型的计算机网络中的硬件资源共享.

1.1.2 计算机网络的组成

1. 从组成部分上看

一个完整的计算机网络主要由硬件、软件、协议三大部分组成.
硬件主要由主机 (也称为端系统, end system)、通信链路、交换设备和通信处理机组成.
软件主要包括实现资源共享的软件和方便用户使用的各种工具软件.
协议是计算机网络的核心, 规定了网络传输数据时所遵循的规范.

2. 从功能组成上看

计算机网络由通信子网和资源子网组成.

通信子网由各种传输介质、通信设备和相应的网络协议组成, 它使得计算机网络具有数据传输、交换、控制和存储的能力, 实现联网计算机之间的数据通信.

资源子网是实现资源共享功能的设备及其软件的集合, 向网络用户提供共享其他计算机上的硬件资源、软件资源和数据资源的服务.

1.1.3 因特网的服务描述

1.1.4 网络协议

1.1.5 计算机网络的拓扑结构

网络拓扑结构主要是指通信子网的拓扑结构.

1.2 网络边缘

例 1.2.1. 现在大量的计算机使用过诸如以太网这样的局域网接入广域网的, 而局域网于广域网的互连主要是通过 _____ 实现的.

A. 路由器　　B. 资源子网　　C. 桥接器　　D. 中继器

评注. 中继器和桥接器通常是指用于局域网的物理层和链路层的联网设备. 目前局域网接入广域网主要是通过称为路由器的互连设备来实现的, 因此本题选 A.

1.3 网络核心

1.3.1 复习参考题

例 1.3.1. 假定用户共享一条 2Mbps 链路, 同时假定当每个用户传输时连续以 1Mbps 传输, 但每个用户仅传输 20% 的时间. 当使用电路交换时, 能够支持的最大用户数为

$$\frac{2\text{Mbps}}{1\text{Mbps}} = 2,$$

这是因为电路交换为所有的呼叫保留资源 (即使它是静默的). 当使用分组交换时, 如果同时传输的用户数量不超过 2, 那么链路的传输速率将大于分组到达的速率, 这时链路的输出缓存队列不会增长, 但当用户数量超过 2 时, 假设用户数量为 n, 那么 n 个用户同时传输时链路的分组到达速率

$$R_{\text{rcv}} = n \times 1\text{Mbps} = n\text{Mbps},$$

这将会导致队列以

$$R_{\text{rcv}} - R_{\text{trans}} = (n-2)\text{Mbps}$$

的速率增长. 根据二项分布可知, k 个用户同时传输的概率为

$$P(\text{activeUserNum} = k) = \text{C}_n^k p^k (1-p)^{n-k},$$

其中 $p = 0.2$ 为某个用户正在传输的概率. 具体而言, 如果有 3 个用户的话, 3 个用户同时传输的概率仅为 0.008, 这就是说, 如果采用分组交换, 那么这个线路有 99.2% 的时间都能很好的服务于 3 个用户.

1.4 分组交换网络中的时延、丢包和吞吐量

1.4.1 分组交换网络中单个节点的时延

我们把分组从源端系统出发, 经过一系列路由器传输, 到达目的端系统的过程中经过的每一站称为一个节点 (node), 当分组在沿途的各个节点传输的时候, 经历了不同类型的时延. 其中比较重要的有:

- 节点处理时延 (nodal processing delay);
- 排队时延 (queuing delay);
- 传输时延 (transmission delay);
- 传播时延 (propagation delay);
- ……

这些时延全部累加起来就是节点总时延 (nodal total delay), 互联网应用受到网络时延的影响很大, 接下来我们就具体介绍这些时延类型.

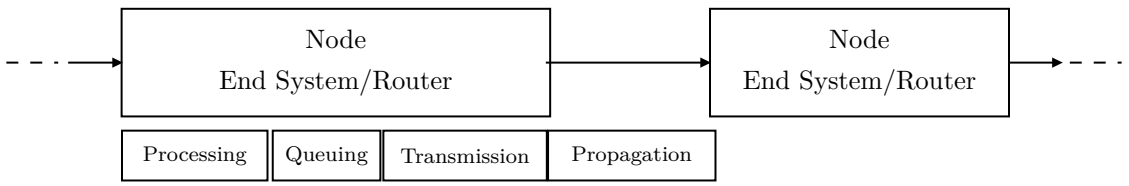


图 1.2: 节点时延

1. 处理时延

服务器需要先对分组进行一些检查和处理, 然后决定将该分组推入哪一条链路的传输队列中. 这一系列操作所占用的时间就是**处理时延** (processing delay), 记作 d_{proc}.

2. 排队时延

在当前节点通往后继节点的链路中, 有一个传输分组的队列 (或者说缓存). 任何需要通过该链路传输的分组都必须先排队, 才能传输. 一般地, 某分组开始传输需要同时满足以下两个条件:

1° 当前链路中没有其他分组正在传输;

2° 当前队列中没有分组排在该分组前面,

它的逆否命题是, 如果该链路当前正忙 (即正在传输其他的分组) 或者队列前面有其他的分组等待传输, 那么该分组只能老老实实地排队.

分组在传输队列上等待传输时所需要的时间就是**排队时延** (queuing delay), 记作 d_{queue}, 某个分组的排队时延取决于队列前面正在等待传输的节点数量和链路的传输速度. 一般地说, 队列前方等待传输的分组数量是到达该队列的流量的强度和性质的函数.

排队时延是节点时延中最复杂, 也最有趣的成分. 与处理时延、传输时延和传播时延不同的是, 排队时延对于不同的分组可能是不同的. 例如, 若干个分组同时到达空队列, 那么第一个分组没有排队时延, 可以直接传输, 而最后一个分组将要经受较大的排队时延. 因此人们通常使用统计量——例如平均排队时延、排队时延的方差, 和排队时延超过某些特定值的概率——来度量、表征排队时延.

3. 传输时延

将分组中的所有比特全部推入节点间的链路所需要的时间是**传输时延** (transmission delay), 记作 d_{trans}. 一般地, 速率为 R Mbps 传输长度为 L Mb 的分组所需要的时间

$$d_{\text{trans}} = \frac{L}{R}.$$

在这里请一定要带着单位去计算, 注意 b 和 B 的区别, 此外, 在考虑数据传输的效率时, 我们一般是以 10^3 作为基准来划定计数单位的[1], 而不是 2^{10}. 这意味着, $1\text{K} = 10^3$, $1\text{M} = 10^6$, $1\text{T} = 10^9$, $1\text{P} = 10^{12}$. 此外, 我们一般会把分组的长度 (即分组的数据量) 说成是分组的大小, 在本书中我们它们指的是一个意思, 都是在说这个分组里含有多少个 bit, 即含有多少个二进制位.

4. 传播时延

分组从链路的一端传播到另一端所需要的时间称为**传播时延** (propagation delay), 记作 d_{prop}. 设链路的传播速率是 s, 链路两端之间的距离为 d, 那么传播时延

$$d_{\text{prop}} = \frac{d}{s}.$$

链路上的传播速率通常与链路本身有关, 也就是说这是一个由链路的物理性质所决定的量, 数据传播采用的物理媒介不同, 传播速率也不同.

[1] 我们通常用 K, M, G, T 等符号来代表 10^3, 10^6, 10^9, $10^{12}, \cdots$, 用 Ki, Mi, Gi, Ti 等符号来表示 $2^{10}, 2^{20}, 2^{30}, 2^{40}, \cdots$. 这就是说, $1\text{KB} = 1000\text{B}$ 而 $1\text{KiB} = 1024\text{B}$.

在这里需要强调的是要注意传输时延和传播时延的区别. 传输时延是将分组从节点推入链路所需要的时间, 它与分组长度和链路的传输速率有关, 与两节点之间的距离和链路的传播速率无关; 而传播时延则是分组从链路的一端传播到另一端, 进入目的节点所需要的时间, 它与链路的传播速率和节点间的距离有关, 与节点到链路的传输速度和分组长度无关.

表 1.1: 对比: 传输时延与传播时延

时延类型	**起始于……**	**终点是……**	**有关的影响因素**
传输时延 d_{trans}	源节点	节点间的链路	节点-链路传输速率 R、分组长度 L
传播时延 d_{prop}	节点间的链路	源节点	链路传播速率 s、节点间的距离 d

如上文所述, 节点的总时延 (total nodal delay, 记作 d_{nodal}) 就是处理时延、排队时延、传输时延和传播时延的累加, 因此我们有计算公式

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}.$$

在不同的网络结构和传输场景中, 不同类别的时延所占据的比例是不同的. 例如, 同一个大学内网中的两台计算机通信时, d_{prop} 可能微不足道, 但一台地面计算机和轨道上的卫星通信时的 d_{prop} 可能就会占据 d_{nodal} 的绝大部分了.

1.4.2 流量强度 丢包

1. 流量强度

估计排队时延的一个常用的物理量是**流量强度** (traffic intensity), 它是单位时间内到达的数据量与节点向链路传输数据的速度的比值. 设单位时间内到达队列的分组数量为 a, 它的单位是 pkt/s, 量纲是 T^{-1}, 它也被称为分组到达队列的平均速率; 设节点向链路推入数据的传输速率为 R, 分组的长度为 L, 那么该节点-链路交界处的流量强度就是 La/R. 其中 La 是单位时间内到达的数据量, R 是单位时间内被推入链路的数据量.

对于流量强度, 一般的规律是, 当流量强度接近于 0 时, 队列中要么分组到达的时间间隔很长, 要么分组的数据量很小, 所以传输队列中的分组不需要太长的时间, 这时平均排队时延 $\bar{d}_{\text{queue}}$ 将接近于 0; 但当流量强度接近于 1, 平均排队时延将迅速增加, 这种情况下, 流量时延的少量扰动都将会导致平均排队时延发生巨大的变化. 总而言之, 我们有

$$\bar{d}_{\text{queue}} \to 0 \text{ for } La/R \to 0, \quad \bar{d}_{\text{queue}} \to +\infty \text{ for } La/R \to 1.$$

2. 丢包

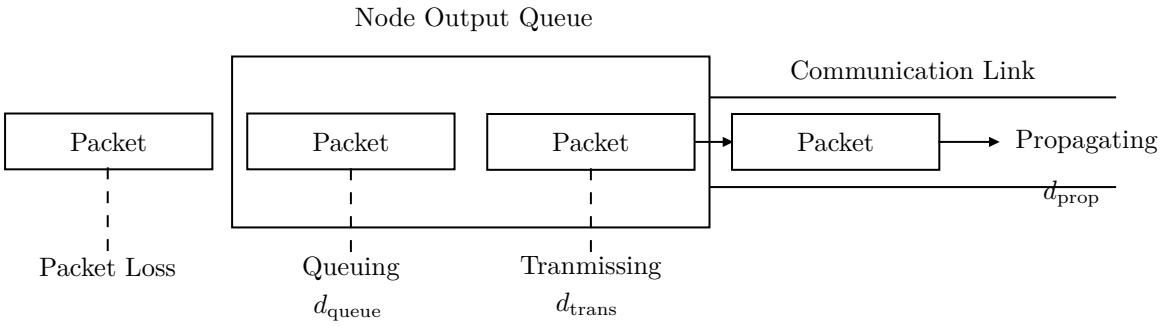


图 1.3: 丢包

节点输出队列能存储的数据量是有限的, 当节点输出队列已经装不下新到达的分组时, 由于没有地方存储这个分组, 路由器就将**丢弃** (drop) 这个分组, 分组丢包的比例将会随着流量强度的增加而增加. 丢包现象的存在, 说明我们不能仅仅依靠时延来衡量数据传输的性能, 还要考虑丢包的问题.

1.4.3　端到端时延

接下来我们考虑从源端系统到目的端系统的时延, 不妨设从源端系统到目的端系统之间有 $N-1$ 台路由器, 且此时网络是不拥塞的, 即忽略排队时延 d_{queue}, 此外, 每台主机或路由器处理分组所需要的时间为 d_{proc}, 每台主机或路由器的输出速率是 Rbps, 每条链路的传播时延是 d_{prop}. 那么把它们累加起来, 得到**端到端时延** (end-end delay):

$$d_{\text{end}-\text{end}} = N\left(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}}\right).$$

根据传输时延的定义, 式中 $d_{\text{trans}} = L/R$, 其中 L 是以 b 计的分组长度, 即每个分组中 0/1 序列的位数. 于是, 端到端时延的一般形式为

$$d_{\text{end}-\text{end}} = N\left(d_{\text{proc}} + \frac{L}{R} + d_{\text{prop}}\right). \tag{1.1}$$

如果各个分组的长度是不同的, 不同节点处的处理时延、平均排队时延和传输时延也是不同的, 不同节点之间的传播时延也不同的话, 那么 1.1 式将会变成更一般化的形式:

$$d_{\text{end}-\text{end}} = \sum_{k=1}^{N}\sum_{i=1}^{n}\left(d_{\text{proc,i}-1} + \bar{d}_{\text{queue,i}-1} + \frac{L_k}{R_{i-1}} + d_{\text{prop,(i}-1,\text{i)}}\right), \tag{1.2}$$

1.2 式中我们假设 N 个分组从源端系统 (节点索引为 0) 经历了 $n-1$ 个路由器 (这些路由器的节点索引分别为 $1,2,3,\cdots,n-1$) 最终到达了目的端系统 (节点索引为 n), 第 k 个分组的长度为 L_k, 第 i 个节点的处理时延、平均排队时延和传输时延分别为 $d_{\text{proc,i}}$, $\bar{d}_{\text{queue,i}}$ 和 $d_{\text{trans,i}}$, 第 $i-1$ 个节点到第 i 个节点的通信链路上的传播时延为 $d_{\text{prop,(i}-1,\text{i)}}$. 其中 $d_{\text{trans,i}} = L_k/R_i$, R_i 表示第 i 个节点将数据推入通往下一个节点的通信链路的传输速率.

例 1.4.1. 假定在发送主机 A 和接受主机 B 间只有一台分组交换机 S, 发送主机和交换机间以及交换机和接受主机间的传输速率分别是 R_1 和 R_2, 假设该交换机使用存储转发分组交换方式, 如果忽略排队时

延、传播时延和处理时延 (这就是说, 我们只考虑传输时延), 那么发送一个长度为 L 的分组的端到端总时延

$$d_{\text{end}-\text{end}} = d_{\text{trans}}(A,S) + d_{\text{trans}}(S,A) = \frac{L}{R_1} + \frac{L}{R_2}.$$

1.4.4 计算机网络中的吞吐量

除了时延和丢包以外, 衡量计算机网络性能的另一个重要的指标是端到端吞吐量. 考虑从主机 A 到主机 B 跨越计算机网络传送一个大文件. 在任意给定的时刻, **瞬时吞吐量** (instantaneous throughput) 是主机 B 接收到该文件的速率 (以 bps 计). 如果文件长度为 Tbit, 而主机 B 花费了时间 Ts 去接收它, 则文件传送的**平均吞吐量** (average throughput) 是 F/Tbps.

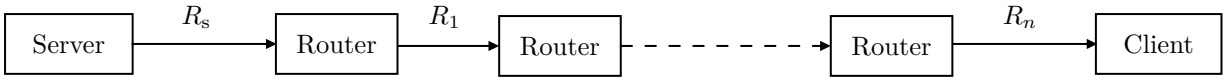
例 1.4.2. 在如图 1.4 所示的网络中, 服务器 S 和客户 C 通过一台路由器 R 相连, 要从服务器传送一个文件到客户, 其中, R_{s} 表示从服务器到路由器的链路的传输速率, R_{c} 则表示从路由器到客户的链路传输速率 (以 bps 计). 考虑这一情况下服务器到客户的网络吞吐量.

图 1.4: 两链路网络

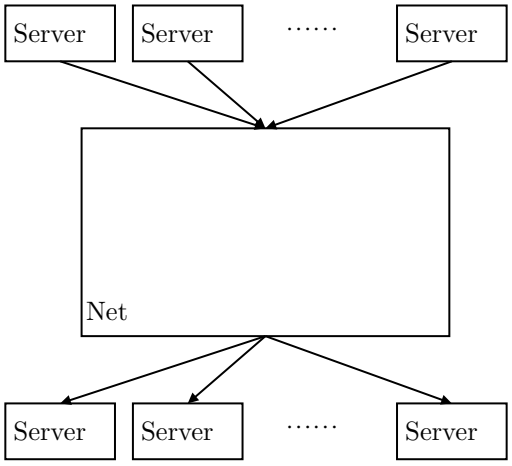
在这一个网络中, 如果 $R_{\text{s}} < R_{\text{c}}$, 那么服务器中的数据将能够顺利地进入到链路中, 然后传输到路由器, 再经由第二条链路来到客户这一边, 这时我们的吞吐量是 R_{s}. 但如果 $R_{\text{c}} < R_{\text{s}}$, 那么此时数据到达路由器的速度就会超出路由器把这些数据推入通信链路中的能力, 这些数据会在路由器当中排队, 然后产生丢包, 这是我们所不愿意看到的. 在计算机科学当中, 谁承担着数据传输的重任, 谁就有可能成为整个系统的瓶颈, 这是一条普遍成立的一般性原理, 不光是计算机网络, 在计算机组成原理或者其他学科当中, 你都有可能发现类似的规律. 总而言之, 我们把这两条链路当中传输能力较弱的那一条成为**瓶颈链路** (bottleneck link), 在这个网络当中的吞吐量就是由瓶颈链路的传输能力所决定的, 这就是说, 对于如图 1.4 所示的两链路网络而言, 起吞吐量是 $\min\{R_{\text{s}}, R_{\text{c}}\}$.

于是我们现在可以对用户接收到完整的文件所需要的时间进行一个估计, 假设文件的大小 (或者说长度) 是 Lbit, 那么所需要的时间大概就是 $F/\min\{R_{\text{s}}, R_{\text{c}}\}$s.

利用同样的原理, 我们可以把这一结论推广到含有链路更多的网络当中, 这时网络的吞吐率依然取决于其瓶颈链路的传输速率, 这就是说, 网络的吞吐率等于 $\min\{R_{\text{s}}, R_1, R_2, \cdots, R_n\}$.



例 1.4.3. 现在考虑若干个用户通过同一个网络核心向若干个服务器请求数据. 这时对于一个 Server-Client 端到端连接来说, 要传输的数据会受到网络核心中其他连接传输数据的干扰. 这时, 网络的吞吐量就不仅取决于沿着路径的传输速率的大小了, 还取决于网络中的干扰流量. 事实上, 如果许多其他的数据流也通过这个网络核心在流动, 那么这个网络核心也有可能会成为整个连接当中的瓶颈 (尽管在一般情况下, 网络核心就像一根粗大的管子, 影响吞吐率的往往是网络边缘).



1.5　协议层次及其服务模型

1.5.1　分层的体系结构

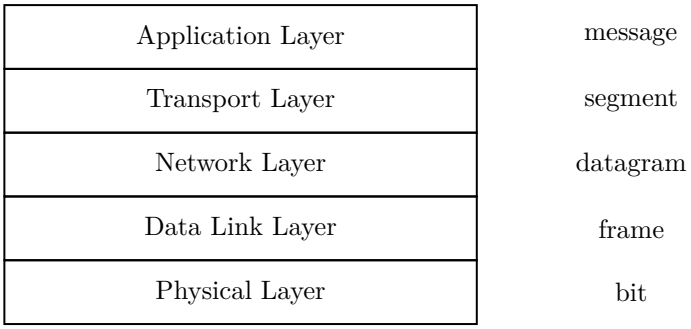


图 1.5: 因特网的协议栈

各个层次的所有协议被称为**协议栈** (protocol stack), 如图 1.5 所示, 因特网的协议栈由 5 个层次组成: 应用层、传输层、网络层、链路层、物理层.

a) **应用层** (Application Layer): 应用层是网络应用程序及其应用层协议所在的层次, 它分布于多个端系统上, 一个端系统中的网络应用程序使用协议与另一个端系统中的网络应用程序交换信息分组, 这种信息分组成为**报文** (message), 用 M 表示. 应用层提供了用户与网络服务之间的接口, 使用户能够访问各种网络应用和网络服务.

b) **运输层** (Transport Layer): 运输层收取应用层报文分组并附上运输层首部信息 H_t, 一同构成**运输层报文段** (transport-layer segment), 注意一个大报文分组可能会被划分为多个应用层报文段分组 (下列各层同理, 这意味着在接收端必须从连续的报文段中重新把大报文给“拼装”出来.), 然后在网络应用程序端点之间传送, 到达目的端系统后, 需要将报文上交给应用层.

- 允许接收端运输层向上向适当的应用程序交付报文的信息;
- 差错检测位信息: 该信息让接收端能够判断报文中的比特是否在途中已被改变;

c) **网络层** (Network Layer): 网络层负责传输路线的选取, 将源端系统和目的端系统地址等附加信息 H_n 封装到运输层报文段分组中, 得到**网络层数据报** (network-layer datagram), 然后传输给链路层.

d) **链路层** (Data Link Layer): 链路层负责在临近的网络节点之间传输数据, 链路层在接收到网络层下派的数据报分组后, 在其分组上附加链路层首部信息 H_l 并生成**链路层帧** (data-link-layer frame), 然后交付到物理层实现在链路上的传输.

e) **物理层** (Physical Layer): 将帧中的**比特** (bit, 即一位 0/1 序列) 以比特流的形式从链路的一个端点移动到另一个端点, 物理层决定并保证了网络物理连接的特性、拓扑结构, 定义了数据传输的物理规范.

在因特网的协议栈中, 下层是上层的实现, 上层使用下层提供的服务来实现本层次的功能. 下层**封装** (encapsulation) 上层分组并附加自己的首部信息, 然后传递给在再下一层处理.

在每一层, 一个分组具有两种类型的字段, 分别是**首部字段** (head field) 和 **有效载荷字段** (payload field), 有效载荷通常来自上一层分组.

数据从发送端系统 (Source) 的协议栈向下, 沿着中间的链路层交换机 (Data Link Layer Switch) 和路由器 (Router) 的协议栈上上下下, 然后向上到达接收端系统 (Destination) 的协议栈.

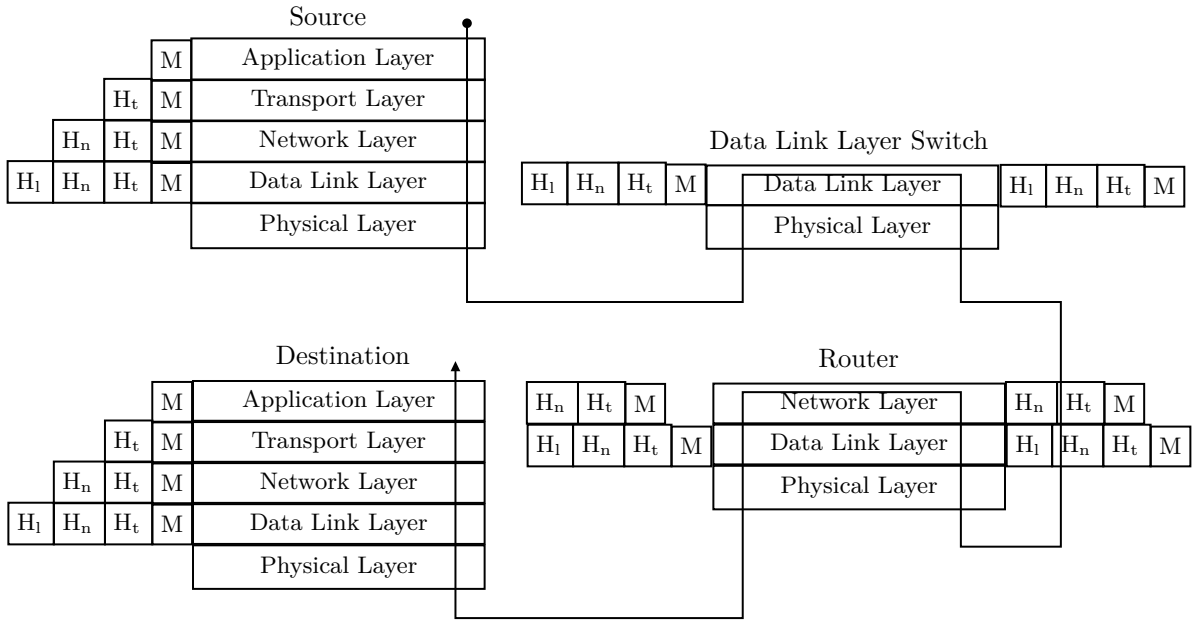
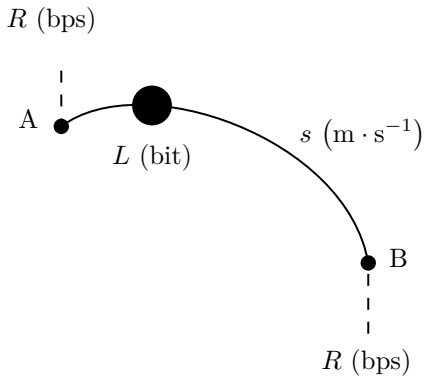


图 1.6: 数据传送的路径

我们注意到, 端系统、路由器和链路层交换机, 都以多层次的方式组织它们的网络硬件和软件. 链路层交换机实现了第一层和第二层, 而路由器实现了第一层到第三层.

1.6 习题

例 1.6.1. 考虑两台主机 A 和 B 有一条速率为 R (bps) 的链路相连. 假定这两台主机相隔 m (m), 沿该链路的传播速率为 s (m/s). 主机 A 向主机 B 发送长度为 L (bit) 的分组.



a) 用 m 和 s 来表示传播时延 d_{prop}.

解. 根据传播时延的定义可知, 传播时延是节点间距离与传播速度的比值, 即 $d_{\text{prop}} = m/s$.

b) 用 L 和 R 来表示该分组的传输时间 t_{trans}.

解. 传输时延是指数据从节点被推入通信链路中所花的时间, 它是数据长度与链路传输速率的比值, 因此 $d_{\text{trans}} = L/R$.

c) 忽略处理和排队时延, 得出端到端时延的表达式.

解. 在该问题中, 端到端时延由传输时延和传播时延两部分组成, 而这两个部分在前两问中都已经计算过了, 因此

$$d_{\text{end}-\text{end}} = d_{\text{prop}} + d_{\text{trans}} = \frac{m}{s} + \frac{L}{R}.$$

d) 假定主机 A 在时刻 $t = 0$ 开始传输该分组, 在时刻 $t = t_{\text{trans}}$, 该分组的最后一个比特在什么地方?

解. 该时刻已经经历了一个传输时延, 即第一个分组已经完全传输到链路当中了. 该分组的最后一个比特在链路中已经传播了 $d_{\text{trans}} - d_{\text{trans}} = 0$ 的时间. 在这段时间内其走过的距离为

$$l = s \times 0 = 0,$$

即此时该分组的最后一个比特位于链路的 A 端点处.

e) 假定 d_{prop} 大于 d_{trans}. 在时刻 $t = d_{\text{trans}}$, 该分组的第一个比特在何处?

解. 根据已知条件, 该时刻该分组的第一个比特依然在链路中传播, 由于第一个比特是直接进入链路开始传播的, 因此它已经传播了 d_{trans} 个单位时间, 于是它所处的位置是 $l = sd_{\text{trans}} = sL/R$.

f) 假定 d_{prop} 小于 d_{trans}. 在时刻 $t = d_{\text{trans}}$, 该分组的第一个比特在何处?

解. 根据已知条件, 该时刻第一个比特已经到达 B 主机了.

例 1.6.2. 考虑从主机 A 向主机 B 通过分组交换网发送语音 (VoIP). 主机 A 将模拟语音转换为传输中的 64kbps 数字比特流, 然后主机 A 将这些比特分为 56 字节的分组. A 和 B 之间有一条链路: 它的传输速率是 2Mbps, 传播时延是 10ms. 一旦 A 收集了一个分组, 就将它向主机 B 发送. 一旦主机 B 接收到一个完整的分组, 它将该分组的比特转换成模拟信号. 从比特产生 (从位于主机 A 的初始模拟信号起) 的时刻起, 到该比特被解码 (在主机 B 上作为模拟信号的一部分), 花了多少时间?

解. 我们先统一单位, 统一用比特作为单位来进行计算. 于是, 主机 A 发送的每个分组的大小为 $56 \times 8 = 448\text{b}$. 这就是说, 主机 A 只需要 448/64000 秒就能产生一个完整的分组, 这就是传输过程中的处理时延. 这就是说,

$$d_{\text{proc}} = \frac{56\text{B}}{64\text{kbps}} = \frac{56 \times 8\text{b}}{64 \times 10^3\text{b} \cdot \text{s}^{-1}} = 7 \times 10^{-4}\text{s} = 0.7\text{ms}.$$

接下来考虑传输时延, 事实上, 根据传输时延的定义,

$$d_{\text{trans}} = \frac{56\text{B}}{2\text{Mbps}} = \frac{56 \times 8\text{b}}{2 \times 10^6\text{b} \cdot \text{s}^{-1}} = 2.24 \times 10^{-4}\text{s} = 0.224\text{ms}.$$

于是最终的端到端时延为

$$d_{\text{end-end}} = d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}} = 0.7\text{ms} + 0.224\text{ms} + 10\text{ms} = 10.924\text{ms}.$$

例 1.6.3. 假设用户共享一条 3Mbps 的链路. 又设每个用户传输时要求 150kbps, 但是每个用户仅有 10% 的时间传输.

a) 当使用电路交换时, 能够支持多少用户?

解. 使用电路交换时, 每个用户都会占用一定的带宽, 因此我们可以直接用链路带宽和用户需求做商, 则

$$\text{userNum} = \frac{3\text{Mbps}}{150\text{kbps}} = \frac{3 \times 10^6\text{bps}}{150 \times 10^3\text{bps}} = 20,$$

因此在使用电路交换时, 能够支持最多 20 名用户.

b) 当使用分组交换时, 用户正在传输的概率为?

解. 由于每个用户只有 10% 的时间传输, 所以其正在传输的概率为 0.1.

c) 假设使用分组交换, 且有 120 名用户. 在任何给定时刻, 实际有 n 个用户正在传输的概率是?

解. 显然, 我们可以将这个问题视为连续进行 120 次随机事件发生概率为 0.1 的 Bernoulli 试验, 正在传输数据的用户数量 $X \sim B(0.1, 120)$. 那么

$$P(X = n) = \text{C}_{120}^{n} \times 0.1^n \times 0.9120 - n.$$

其中 $\text{C}_m^k = \binom{m}{k}$ 为组合数, 即有限集合 A (A 中含有 m 个元素, 记作 $|A| = m$) 的满足 $|S| = k$ 的子集 $S \subset A$ 的数量.

例 1.6.4. 在一个电路交换网络中, 每台端系统与一台交换机直接相连, 当两台端系统要通信时, 该网络在两台端系统之间建立一条专用的端到端连接. 用户在忙时以 1000kbps 的速率产生数据, 但忙时仅以 $p = 0.1$ 的概率产生数据, 假定电路交换链路的带宽为 1Gbps.

a) 在采用电路交换技术时, 能被支持的最大用户数量 N 是多少?

解. 可以用链路带宽与用户需求带宽做商直接得到能够支持的最大用户数量

$$N = \frac{1\text{Gbps}}{100\text{kbps}} = \frac{1 \times 10^9\text{bps}}{100 \times 10^3\text{bps}} = 10^4.$$

b) 在使用分组交换时, 假设有 M 个用户, 多于 N 个用户发送数据的概率为?

解. 设正在发生数据的用户数量为 X, 显然 $X \sim B(0.1, M)$. 同时, 由于两个用户发送数据与否是独立的, 因此随机事件 $\{X = N+1\}, \{X = N+2\}, \cdots, \{X = M\}$ 是随机事件 $\{X > N\}$ 的一个划分, 由全概率公式得

$$P(X > N) = \sum_{k=N+1}^{M} P(X = k) = \sum_{k=N+1}^{M} \mathrm{C}_M^k \times 0.1^k \times 0.9^{M-k}.$$

例 1.6.5. 令 a 表示在一条链路上分组的到达率, 其单位为 pkt/s, μ 为一条链路上分组的传输率, 基于 $d = d_{\text{queue}} + d_{\text{trans}}$, 用 a 和 μ 表示 d.

解. 要用 a 和 μ 表示 d, 只需用 a 和 μ 表示 d_{queue} 和 d_{trans}. 我们只考虑 1 个分组, 那么

$$d_{\text{queue}} = \frac{1}{a}, \quad d_{\text{trans}} = \frac{1}{\mu}.$$

于是总时延

$$d = \frac{1}{a} + \frac{1}{\mu}.$$

Little 公式是排队论中的一个著名公式, 设 N 是传输队列中待传输分组的平均数量, a 为分组到达率, d 为一个分组经历的平均总时延 (即排队时延加传输时延). Little 公式表明,

$$N = ad.$$

其实呢, Little 公式的意思就是说, 每分钟有 a 个人来排队, 处理一个人平均需要 d 分钟, 那么这个队伍的长度就应该是 ad.

例 1.6.6. 考虑一台路由器缓存前面的一条出链路. 假定某队列平均包含 10 个分组, 平均分组排队时延是 10ms, 链路传输速率为 100pkt/s. 不考虑丢包情况, 使用 Little 公式计算平均分组到达率.

解. 由已知条件, $N = 10$, $d_{\text{queue}} = 10\text{ms}$, $d_{\text{trans}} = \frac{1}{100}\text{s}$, 那么

$$N = ad = a\left(d_{\text{queue}} + d_{\text{trans}}\right),$$

代入数值,

$$\begin{aligned} 10\text{pkt} &= a\text{pkt/s}\left(10\text{ms} + \frac{1}{100}\text{s}\right) = a\text{pkt/s}\left(\frac{1}{100}\text{s} + \frac{1}{100}\text{s}\right) \\ &= a \times 0.02\text{s}, \end{aligned}$$

解得平均分组到达率 $a = 10/0.02 = 500\text{pkt/s}$.

例 1.6.7. 假设你希望从波士顿向洛杉矶紧急传送 40×10^{12}B 的数据, 你有一条 1Gbps 的专用链路可用于传输数据. 你是愿意通过这条链路传输数据, 还是愿意使用 FedEx 夜间快递来交付?

解. FedEx 夜间快递是指 FedEx 提供的在晚间送达包裹的服务, 这种服务通常适用于那些需要在次日早晨或者特定时间内送达的包裹[2]. 我们不妨假设 FedEx 可以在 72 小时内将包裹送达洛杉矶. 那么使用专

[2]这段材料由 ChatGPT 生成, ChatGPT 可能会生成错误的信息, 请谨慎参考.

用链路的传输时间

$$t = \frac{40 \times 10^{12}\text{B}}{1\text{Gbps}} = \frac{8 \times 40 \times 10^{12}\text{b}}{10^9\text{bps}}$$
$$= 320,000\text{s} = \frac{320,000\text{s}}{3600\text{s/h}} \approx 89\text{h} > 72\text{h}.$$

于是我选择 FedEx 夜间快递.

例 1.6.8. 在包括因特网的现代分组交换网络中, 源端系统将长应用层报文分段为较小的分组并向网络发送, 接收方将这些分组重新装配为原始报文, 这个过程称为报文分段. 考虑一个长度为 8×10^6bit 的报文, 经过两个分组交换机到达目的端系统, 每段链路的传输速率是 2Mbps. 忽略传播、排队和处理时延.

a) 考虑从源端系统到目的端系统发送该报文且没有报文分段, 从源端系统到第一台分组交换机需要多长时间? (每台分组交换机均适用存储转发分组交换) 从源端系统移动该报文到目的端系统需要多长时间?

解. 没有报文分段时, 从源端系统到第一台分组交换机需要的时间为

$$t_1 = \frac{8 \times 10^6\text{bit}}{2\text{Mbps}} = 4\text{s},$$

该文件最终到达目的端系统的时间为

$$t = 3t_1 = 12\text{s}.$$

b) 假设报文被分段为 800 个分组, 每个分组长度为 10,000bit. 从源端系统移动第一个分组到第一个分组交换机需要多长时间? 从第一台交换机发送第一个分组到第二台交换机, 从源端系统发送第二个分组到第一台交换机各需要多长时间? 什么时候第二个分组能被第一台交换机全部收到?

解. 有报文分段时, 从源端系统移动第一个分组到第一台分组交换机需要的时间为

$$t_1' = \frac{10,000\text{bit}}{2\text{Mbps}} = 5\text{ms}$$

由于每个分组的传输速率相同, 且不考虑出了传输时延以外的各种时延, 因此从第一台交换机到第二个交换机需要 t_1' 时间, 第二个分组源端系统到第一台交换机需要 t_1' 时间. 于是在整个过程经历 $2t_1'$, 即 10ms 后, 第二个分组被第一台交换机全部收到.

c) 使用报文分段时, 该文件从源端系统到目的端系统需要多长时间?

解. 每个分组需要 $3t_1'$ 的时间才能完全送到目的端系统, 但是由于在整个过程中三条链路都在不停地传输数据, 因此事实上传输 800 个分组只需要 $(800+2)t_1$ 的时间, 这是因为我们需要 $2t_1$ 的时间才能让三条链路全部工作起来. 所以他需要的时间为

$$t' = (800+2) \times 5\text{ms} = 4010\text{ms} < 4\text{s},$$

发现使用报文分段时, 传输速率能够大幅减小时延.

d) 除了减小时延以外, 使用报文分段还有什么原因?

解. 除了减小时延以外, 使用报文分段还能够提高链路的使用率. 事实上, 在不使用报文分段的时候, 传输的第二阶段 (即从交换机 1 到交换机 2) 是源端系统到交换机 1 的链路和交换机 2 到目的端系统的链路是空闲的. 在传输的第一阶段和第三阶段同理, 而当网络中接入的用户数量增加, 链路的利用率将会变得至关重要, 因此采用报文分段的方式.

e) 讨论报文分段的缺点.

- 由于报文分段把一段报文分解为若干分组, 因此在拆分报文和重组报文的过程中会产生一定的时间开销;
- 由于分组当中包含一些头部信息, 所以报文分段会产生额外的头部信息, 增加了传输数据量;
- 分组在遇到丢包的时候需要重传 (虽然重传一个分组总比重传一整个文件更容易, 但是若干个分组中有一个分组丢包的概率是要远远大于 1 整个文件丢包的概率的);
- 报文分段可能会降低实时性.

第 2 章　应用层

本章的主要任务是:

a) 定义关键的应用层概念 (网络服务、客户、服务器、进程、运输层接口);

b) 详细考察几种网络应用程序 (Web、电子邮件、DNS、对等文件分发)

2.1　网络应用原理

研发网络应用程序的核心是写出能够运行在不同端系统和通过网络彼此通信的程序, 这个程序应该能在多个端系统上运行, 但并不需要在网络核心设备 (路由器、链路层交换机) 上运行. 于是, 网络应用程序被限制在了网络边缘, 而网络核心设备并不在应用层起作用.

2.1.1　网络应用体系结构

应用体系结构 (application architecture) 是由应用程序研发者设计的, 规定了如何在各种端系统上组织该应用程序的体系结构. 本小节介绍两种主流的网络应用体系结构: **客户—服务器体系结构** (client-server architecture, C-S) 和**对等体系结构** (P2P architecture, P2P).

在这里要强调的是, 网络应用程序的体系结构不等同于网络的体系结构, 因特网的协议栈自顶向下分别是应用层、传输层、网络层、链路层和物理层, 它为应用程序提供了特定的服务的集合. 请诸位读者注意这两个概念的区别.

1. 客户—服务器体系结构 (C-S 结构)

在 C-S 结构当中, 有一个总是打开的端系统, 我们称之为**服务器** (server), 它服务于来自许多称之为**客户** (client) 的端系统的**请求** (request). C-S 结构的主要特点为:

a) 客户之间不相互通信;

b) 服务器具有固定的、公开的地址, 我们将其称之为 IP 地址.

由于服务器的地址固定、公开, 且服务器总是打开的, 所以客户才能够向服务器的 IP 地址发送分组来取得联系.

例 2.1.1. Web 应用程序是典型的具有客户—服务器体系结构的网络应用程序. Web 服务器服务于来自运行在客户的主机上的浏览器的请求, 当 Web 服务器接收到请求时, 它向客户发送所请求的对象, 这个过程称为**响应**. 在 Web 应用中, 两个客户的浏览器并不直接通信, 这也符合 C-S 体系结构的特征.

例 2.1.2. 我们很快就会看到, FTP、Telnet、电子邮件也是具有 C-S 体系结构的典型的网络应用程序.

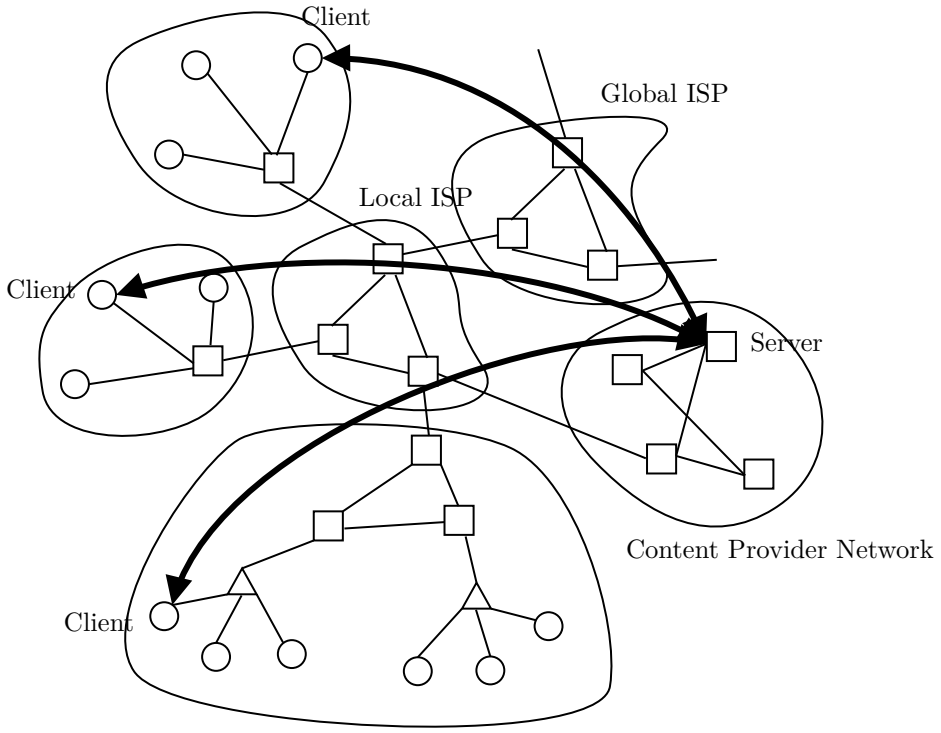


图 2.1: 客户—服务器体系结构 (Client-Server Architecture)

2. 对等体系结构 (P2P 结构)

P2P 体系结构中的用户需要通信时, 不需要通过专门的服务器, 而是可以直接与对方的主机通信, 相互通信的双方被称为**对等方**. 这种结构对专用服务器具有最小的依赖, 有时甚至没有依赖, 它的主要特征为:

a) **自扩展性** (self-scalability): 每个对等方的加入在占用网络资源的同时也在增加系统的服务能力;

b) **有成本效率**: P2P 体系结构的网络应用程序不需要庞大的服务器基础设施和服务器带宽.

c) **风险与挑战**: P2P 应用程序的非集中式结构在降低服务器成本的同时也带来了安全性、性能和可靠性等多方面的挑战.

例 2.1.3. 服务程序在 Windows 环境下工作, 并且运行该服务器的计算机也作为客户访问其他计算机提供的服务, 那么, 这种网络应用模型属于 _____.

A. 主从式 (master-slave architecture) B. 对等式

C. 客户/服务器模型 D. 集中式

评注. “运行该服务器的计算机也作为客户访问其他计算机提供的服务”说明在这个网络应用模型中并没有固定的客户与服务器的划分, 属于 P2P 模式, 因此本题选择 B 选项. P2P 模型中各用户的计算机共享资源, 从而提供比单个用户所能提供的多得多的资源, 任意一对计算机被称为对等方.

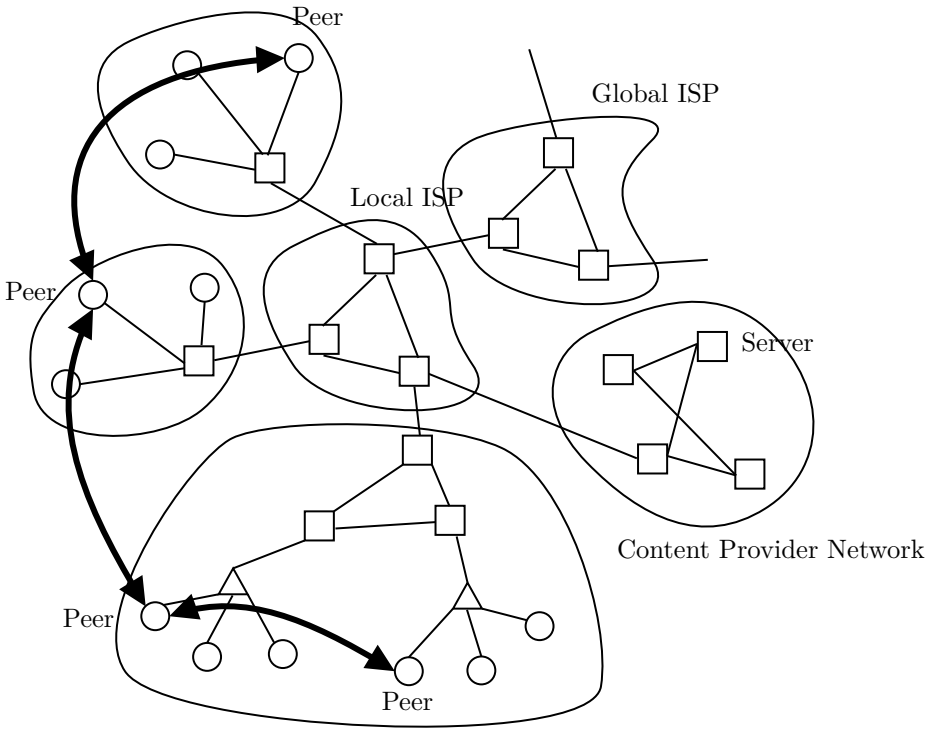


图 2.2: 对等体系结构 (P2P Architecture)

2.1.2 进程通信

1. 客户和服务器进程

在两个不同端系统上的进程, 通过跨越计算机网络交换**报文** (message) 而相互通信. 网络应用程序由成对的进程组成, 这些进程通过网络相互发送报文, 我们通常将这一对进程的一个会话场景中*发起通信的一方*, *即在该会话开始时发起与其他进程的联系的一方标识为*客户 (client), *另一个进程*, *即在会话开始时等待联系的进程标识为*服务器 (server). 在 Web 应用程序中, 浏览器就是一个客户进程, Web 服务器就是一个服务器进程; 而在一次 P2P 文件传输共享中, 下载文件的对等方就是客户, 上传文件的对等方就是服务器.

2. 进程与计算机网络之间的接口

套接字 (socket) 是一台主机内应用层与传输层之间的接口, 进程通过它来向网络发送报文和从网络接受报文, 套接字也称为应用程序和网络之间的 API. 套接字的应用层部分由应用程序开发者控制, 运输层部分则由操作系统控制, 换言之, 开发者可以控制套接字在应用层端的一切, 但对运输层端几乎没有控制权, 这就是说, 套接字的运输层端对于应用程序开发者来说是透明的.

套接字的工作方式是, 在发送端, 应用程序将报文推进套接字, 在该套接字的另一侧, 运输层协议从接收进程的套接字得到该报文.

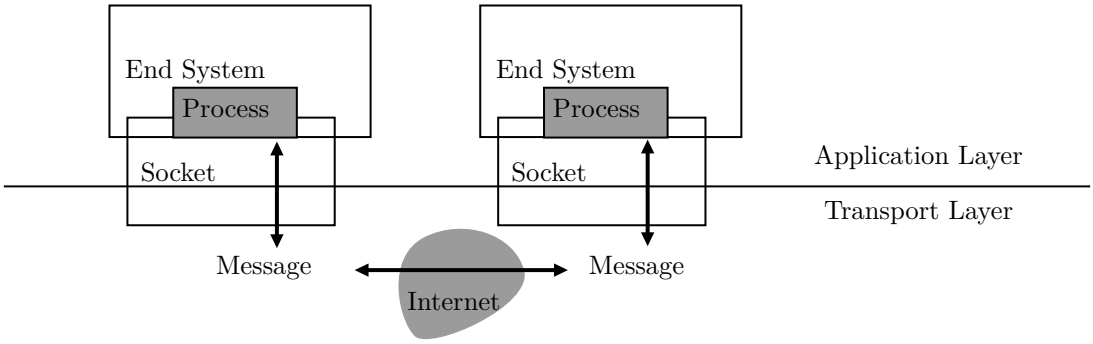


图 2.3: 套接字是应用程序和网络之间的 API

3. 进程寻址

在一台主机上运行的进程为了向另一台主机上运行的进程发送分组, 接收进程需要有一个地址. 为了标识这个接收进程, 我们首先需要标识主机的地址, 其次, 还需要在目的主机中指定接收进程的标识符.

在因特网中, 主机由其 **IP 地址** (IP Address) 标识, IP 地址是一个 32 比特的量, 它能够唯一地标识一台主机. 目的地**端口号** (port number) 用于标记目的主机上的接收进程 (接受套接字).

2.1.3　运输层协议

很多网络都提供了不止一种运输层协议, 当开发一个应用时, 必须选择一种可用的运输层协议. 而选择合适的运输层协议的思路是, 判断它提供的服务是否能满足你的需求. 我们大体上, 将应用程序的服务要求分为四类: 可靠数据传输、吞吐量、定时和安全性.

1. 可靠数据传输

一个协议如果能通过某些工作确保*由应用程序的一端发送的数据能够正确并完全的交付给该应用程序的另一端*, 那么该协议就提供了**可靠数据传输** (reliable data transfer) 的服务.

与需要可靠数据传输的应用相对的网络应用是**容忍丢失的应用** (loss-tolerant application), 这些应用能够承受一定程度的数据损失.

2. 吞吐量

某些应用程序对吞吐量是有要求的, 这些应用称为**带宽敏感应用** (bandwidth-sensitive application), 带宽敏感应用需要选择那些*能够以某种特定的速率提供确保的可用吞吐量*的运输层协议, 这就是说, 当应用程序请求 $r$bps 的确保吞吐量时, 该运输层协议能够确保可用的吞吐量总是大于等于 $r$bps.

与带宽敏感应用相对的是**带宽弹性应用** (bandwidth-elastic application), 这些应用能够根据当前可用带宽来可变地调整吞吐量.

3. 定时

我想你在打在线游戏的时候最不想看到的就是屏幕左上角 (或者右上角) 的连接指示灯变成橙色或者红色, 然后时延变成 999ms. 这些网络应用需要选择能够提供定时保证的运输层协议, 当然了, 协议也不是万能的, 有的时候尽管选择了提供定时保证的协议, 但是你打游戏还是会卡.

你可能会在想, 我打游戏经常卡成这样, 凭什么你还敢说这是一个保证定时通信的协议? 这是不是虚假宣传? 我们说, 不是的. 这是因为, 虽然作为个人来看, 在打游戏的时候确实时常会遇到卡顿的现象, 但是在整个互联网当中, 这个卡顿线性的发生频率就相当低了, 那么自然可以说这是一个“保证定时”的协议.

4. 安全性

有的运输协议能够提供一种或多种安全性服务, 最简单的一个例子是, 是网络信息传输中的信息加密. 顺带一提, QQ 和微信的信息加密等级非常之低.

2.1.4 TCP 和 UDP

因特网为应用程序提供了两个运输层协议——TCP 和 UDP.

1. TCP 服务

TCP 服务模型包括面向连接服务和可靠数据传输服务.

- **面向连接的服务**: 握手 (建立连接) ——传输数据——拆除连接. 在数据报文开始流动之前, TCP 让客户和服务器相互交换运输层控制信息, 在握手阶段后, 一个 **TCP 连接** (TCP connection) 就在两个进程的套接字之间建立了. 连接双方的进程在此链接上同时进行报文收发, 在报文结束发送后, 连接被拆除.
- **可靠数据传输服务**: 通信进程能够依靠 TCP, 无差错、按适当顺序地交付所有发送的数据, 而没有字节的丢失或冗余.

此外, TCP 还具有拥塞控制机制. 当发送方和接收方之间的网络出现拥塞时, TCP 的拥塞控制机制会抑制发送进程.

2. UDP 服务

UDP 是一种不提供不必要服务的轻量级运输协议, 它仅提供最低限度的服务. 它的主要特点为:

- 无连接: 在两个进程通信前并没有握手过程.
- 不可靠数据传输: 当进程将一个报文发送斤 UDP 套接字时, UDP 并不保证该报文将到达接收进程, 不仅如此, 到达接收进程的报文也可能是乱序到达的.
- 不包括拥塞机制: 因此 UDP 的发送端可以用它选定的任何速率向网络层注入数据.

今天的因特网能够为时间敏感应用提供满意的服务, 但它不能提供任何定时或吞吐量保证.

2.1.5 应用层协议

应用层协议 (application-layer protocol) 规定了运行在不同的端系统上的应用程序进程如何相互传递报文. 应用层协议定义了以下内容:

1° 交换的报文类型. 例如请求报文和相应报文.

2° 交换报文类型的语法. 例如报文中的各个字段以及这些字段是如何描述的.

3° 字段的语义. 即这些字段中信息的含义.

4° 一个进程何时发送报文、如何发送报文、按照什么样的规则对报文进行相应.

2.2　Web 和 HTTP

第 3 章　运输层

3.1　面向连接的运输：TCP

3.1.1　TCP 连接

1. TCP 的特征

1° 面向连接: 一个应用进程开始与另一个应用进程通信和发送数据之前, 必须要建立 TCP 连接, 因此 TCP 是 **面向连接**的 (connection-oriented). TCP 连接是一种逻辑连接, 其共同状态仅保留在两个通信端系统的 TCP 程序中.

2° 全双工: TCP 连接提供的是 **全双工服务** (full-duplex service), 即两台建立了 TCP 连接的主机可以同时向对方发送数据.

3° **点对点** (point-to-point): 一个 TCP 连接只连接一对主机.

2. TCP 连接的建立: 三次握手

假设某台主机上的一个进程 (我们将其称为客户进程) 想要与另一台主机上的一个进程 (我们将其称为服务器进程) 建立 TCP 连接. 这时, 客户首先发送一个特殊的 TCP 报文段, 服务器用另一个特殊的报文段来响应. 这两个报文段不承载"有效载荷", 即不包含应用层数据, 它们的交换意味着这两台主机已经为大量的报文段的通信做好了准备. 最后, 客户用第三个可以承载有效载荷的特殊报文段作为第二个报文段的响应, TCP 连接上的通信正式开始. 这一过程被称为**三次握手** (three-way handshake).

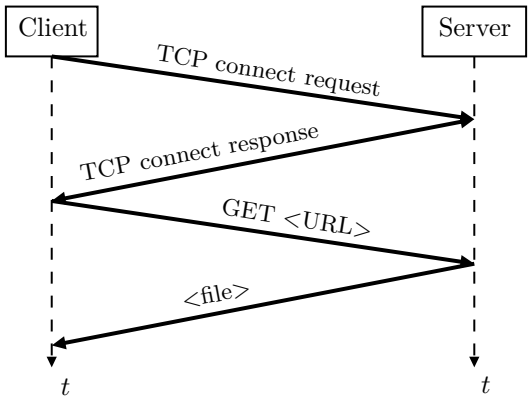


图 3.1: 三次握手

3. TCP 如何传送数据

1° 客户进程将数据通过套接字送入运输层, 交由 TCP 控制.

2° TCP 将这些数据引导到该连接的 **发送缓存** (send buffer) 中.

3° TCP 不时从发送缓存中取出一块数据, 配上一个 **TCP 首部** (TCP header) 封装成 **TCP 报文段** (TCP segment) 传递到网络层, TCP 只负责报文段的逻辑传递, 而底层的物理传递则由下层的网络层、数据链路层和物理层来实现.

4° 另一台主机上的 TCP 接收到一个报文段后, 将其放入 **接收缓存** (receive buffer) 中.

5° 应用程序从接收缓存中读取数据流.

第 4 章　网络层: 数据平面

4.1　网络层概述

网络层能够被分解为两个相互作用的部分, 数据平面和控制平面.

4.1.1　转发和路由选择: 数据平面和控制平面

网络层的数据平面的功能, 即**转发**功能, 是网络层中每台路由器的功能, 该数据平面功能决定到达路由器输入链路之一的数据报 (即网络层分组) 如何转移到该路由器的输出链路之一.

具体转移到哪一个输出链路, 则根据**转发表**的值来确定, 转发表是一台路由器的路由选择算法与其他路由器的路由选择算法进行通信和计算得到的.

控制平面功能, 即**路由**功能, 控制数据报沿着从源主机到目的主机的端到端路径中路由器之间的路由方式, 决定了端到端传输的数据路径.

每台路由器的数据平面的主要作用是从其输入链路向其输出链路转发数据报, 控制平面的主要作用是协调这些本地的路由器的转发操作, 使得数据报沿着源和目的主机之间的路由器路径最终进行端到端传送.

4.1.2　网络层分组　分组交换机　路由器

"分组"这一概念在 TCP/IP 体系结构模型的不同层次中用不同的术语来表达. 运输层分组的名字是报文段 (segment[1]), 数据链路层的分组称为帧 (frame), 而网络层的分组称为网络层**数据报** (datagram). 数据报对报文段进行了一次拆分, 使其具有更小的体积, 同时附上了网络层首部 H_n, 而链路层帧则是对网络层数据报的又一次拆封和封装, 附上了链路层首部 H_l.

分组交换机一词在计算机网络中可以指代路由器, 也可以指链路层交换机. 在进行转发时, 链路层交换机通过链路层帧中的字段值来进行决策, 而路由器则基于网络层数据报中的首部字段值决策.

4.1.3　因特网的服务模型

因特网的网络层服务模型是一种**尽力而为** (best-effort) 的服务, 这就是说, 它并不保证数据的可靠传输、并不保证数据报的顺序到达, 在带宽、延迟等方便也不做任何的保证.

[1] 读作 /segmənt/.

参考文献

[1] [美] James F.Kurose, [美] Keith W. Ross. 计算机网络: 自顶向下方法 (Computer Networking: A Top Down Approach, 8e)[M]. 陈鸣译. 北京: 机械工业出版社, 2022.

[2] 王道论坛组. 2024 年计算机网络考研复习指导 [M]. 北京: 电子工业出版社, 2022.

公诚勇毅　永矢毋忘
中华灿烂　工大无疆

2023 年　西北工业大学　中国西安