计算机网络

Computer Networking

极夜酱

目录

Chapter 1 网络

1.1 因特网

1.1.1 因特网（Internet）

因特网是一个世界范围的计算机网络，它互联了遍及全世界数十亿的计算设备，所有这些设备都称为主机（host）或端系统（end system）。端系统通过通信链路（communication link）和分组交换机（packet switch）连接到一起，不同的链路能够以不同的速率传输数据，链路的传输速率（transmission rate）使用比特/秒（bps, bit/s）来度量。端系统通过因特网服务提供商（ISP, Internet Service Provider）接入因特网。

当一台端系统向另一台端系统发送数据时，发送端将数据分组，发送到目的端系统，在那里进行组装。一个分组所经历的一系列通信链路和分组交换机称为路径（route / path）。分组交换类似于现实中的货物运输，在出发地将货物分开并装上多辆卡车，每辆卡车独立通过公路运输，最后在目的地卸货并重新组装。

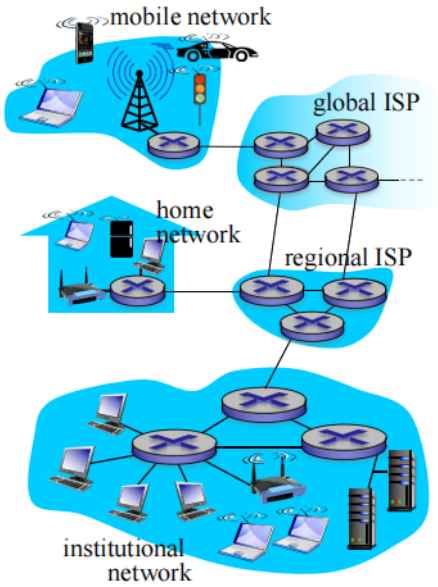

图 1.1: 计算机网络

1.1.2 分布式应用程序（Distributed Application）

分布式应用程序涉及多个相互交换数据的端系统，例如即时通信、实时道路信息、视频会议、多人游戏等。分布式应用程序的核心问题在于一个端系统上的应用程序如何能够向运行在另一个端系统上的应用程序发送数据。

套接字接口（socket interface）规定了运行在一个端系统上的程序向运行在另一个端系统上的特定程序交付数据的方式。例如 Alice 要给 Bob 寄一封信，当然 Alice 不能只是写完这封信就把它丢出窗外。Alice 需要把信放入信封，在信封上根据指定格式写上收信人的全名、地址和邮政编码，信封上贴上邮票，再将信封投入信箱中。Alice 想要寄信就必须要遵守邮政服务制定的这一套规则。因此，发送数据的程序也必须遵守 socket 接口，才能向接收数据的程序发送数据。

1.1.3 协议（Protocol）

在两个人或两台设备之间进行通信时需要遵守一些协议，协议就是用于管理通信的一组规则。传输控制协议 TCP（Transmission Control Protocol）和网际协议 IP（Internet Protocol）是因特网中两个最为重要的协议，因特网的主要协议统称为 TCP/IP。

因特网标准（Internet standard）是经过充分测试的规约，只要是与因特网打交道，就会用到它们，并要服从于它们。因特网标准由 IETF（Internet Engineering Task Force）研发，IETF 的标准文档称为 RFC (Request For Comment)，目的是解决因特网先驱者们面临的网络和协议问题。它们定义了 TCP、IP、HTTP、SMTP 等协议，目前已经有将近 7000 个 RFC。

人类无时无刻都在执行协议，人类用约定好的交互方式互相交流。但是如果两人的交谈都不在同一频道上，那就不能好好沟通了。

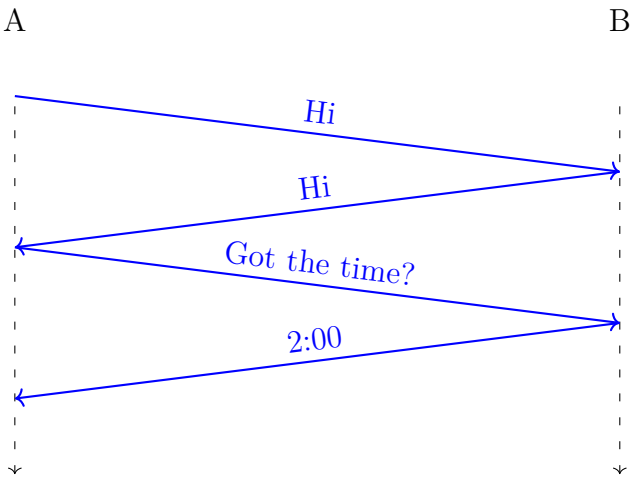

图 1.2: 人类协议

在因特网中，涉及两个或多个远程通信实体的所有活动都受协议的制约。例如在浏览器中输入 URL（Uniform Resource Locator）向一个 Web 服务器发出请求，首先你的计算机将向该 Web 服务器发送一条连接请求报文，并等待回答。Web 服务器接收到连接请求报文，并返回一条连接响应报文。在得知请求正常后，计算机会发送一条要获取的网页名字的报文，最后 Web 服务器向计算机返回该网页。

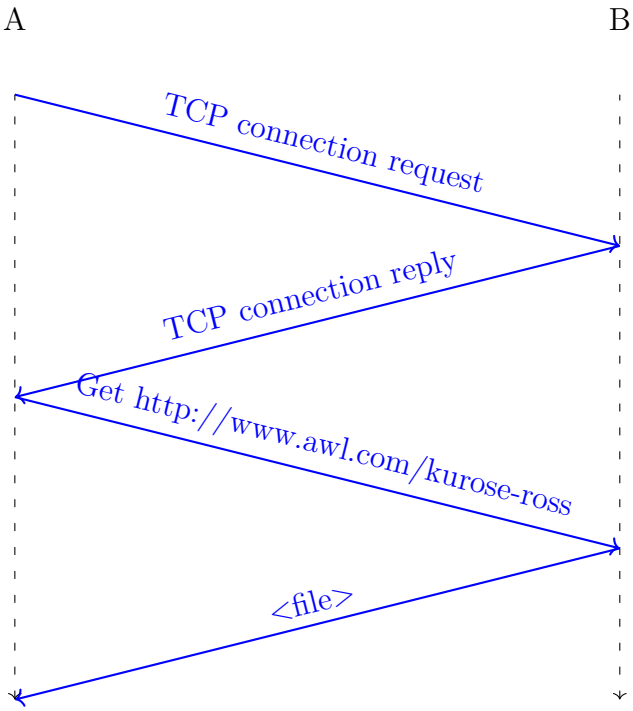

图 1.3: 网络协议

1.2 分组交换

1.2.1 存储转发传输（Store-and-Forward Transmission）

在网络应用中，端系统彼此交换报文（message），报文能够包含任何数据。报文从源端系统发送到目的端系统的过程中，长报文会被划分为较小的数据块，称为分组（packet），每个分组都通过通信链路和分组交换机传送。如果源端系统发送一个 L bits 分组，链路的传输速率为 R bits/sec，则传输该分组的时间为 $\frac{L}{R}$ 秒。

多数分组交换机在链路的输入端使用存储转发传输机制，存储转发传输是指在交换机能够开始向输出链路传输该分组的第一个 bit 之前，必须接收到整个分组。

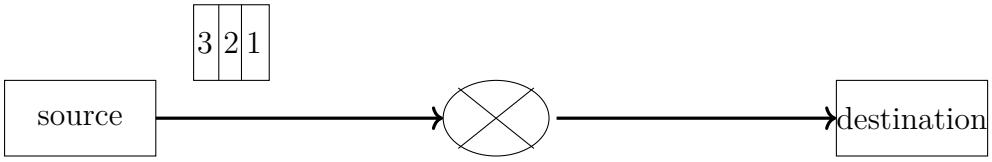

图 1.4: 存储转发

假设忽略传播时延（propagation delay），源端系统在时间 0 开始传输，路由器在时间 $\frac{L}{R}$ 刚好接收到整个分组，之后再向输出链路开始传输，在时间 $2\frac{L}{R}$ 整个分组被目的端系统接收。

因此，由 N 条速率为 R 的链路组成的路径（在源和目的地之间有 $N-1$ 台路由器）发送一个分组，端到端的时延为

$$d = N\frac{L}{R} \tag{1.1}$$

1.2.2 时延

当从一个节点到后继节点，一个分组在沿途的每个节点都经受了几种不同类型的时延，包括节点处理时延（nodal processing delay）、排队时延（queuing delay）、传输时延（transmission delay）、传播时延（propagation delay）。

处理时延包括了检查分组首部和决定分组去向所需要的时间，以及检查差错的时间。

分组交换机的每条链路都有一个输出缓存/输出队列（output buffer / output queue）。如果到达的分组需要传输到某条链路，但发现该链路正忙于传输其它分组，该分组必须在输出缓存中等待。因此，除了存储转发时延以外，分组还要承受输出缓存的排队时延。由于缓存空间的大小是有限的，到达的分组可能发现该缓存已被填满，这种情况下将出现丢包（packet loss），到达的分组或已经排队的分组将被丢弃。

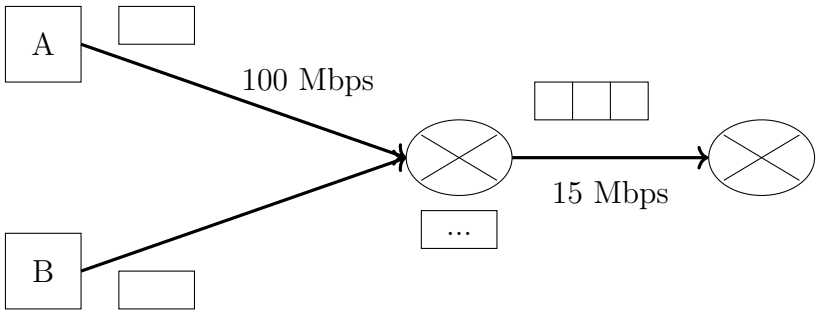

图 1.5: 排队时延

分组通常是以 FCFS（First-Come-First-Served）的方式传输，只有当所有以及达到的分组被传输之后，才能传输新到达的分组。传输时延 $\frac{L}{R}$ 就是将所有分组推向输出链路所需的时间。

传播时延是指从链路的起点到下一个路由器传播所需的时间。

假设 d_{proc}、d_{queue}、d_{trans} 和 d_{prop} 分别表示处理时延、排队时延、传输时延和传播时延，那么节点总时延为

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop} \tag{1.2}$$

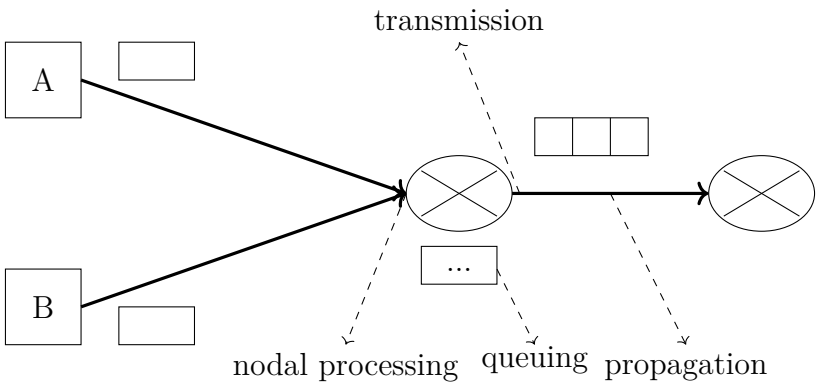

图 1.6: 时延

1.2.3 转发表（Forwarding Table）

路由器从输入链路获得分组，然后向输出链路发送分组，但是路由器怎样决定应当向哪一条链路发送呢?

因特网中每一个端系统都有一个 IP 地址，源在发送分组时在分组首部包含了目的地的 IP 地址。当分组达到每一个路由器时，路由器根据转发表，为其查询适当的输出链路。

端到端的选路过程类似于现实中的问路，每到一个地点都询问别人路线，逐步找到最终的目的地。在这个类比中，被问路的人就类似于路由器。

1.2.4 traceroute

traceroute 是诊断网络问题时常用的工具，它可以定位从源主机到目标主机之间经过了哪些路由器，以及到达各个路由器的耗时。当源主机向目标主机发送消息，发现消息无法送达。此时，可能是某个中间节点发生了问题，比如某个路由器因负载过高产生了丢包。通过 traceroute 可以定位网络包在是在哪个节点丢失的。

traceroute 将从源发送 N 个特殊的分组，当第 i 个路由器收到第 i 个分组时，该路由器会向源回送一个报文，记录了路由器的名字和地址。traceroute 会重复该

实验 3 次。

traceroute (Linux) / tracert (Windows)

```
1 tracert www.github.com
```

运行结果

```
1     2 ms     9 ms     2 ms  HS8145V [192.168.1.1]
2     5 ms    15 ms     5 ms  100.65.0.1
3     5 ms     5 ms     5 ms  124.74.22.41
4    25 ms     6 ms    17 ms  101.95.88.138
5     *        *        *     请求超时。
6     *        *        *     请求超时。
7     *        *        *     请求超时。
8    30 ms    32 ms    35 ms  106.38.244.146
9     *        *        *     请求超时。
10     *        *        *     请求超时。
11     *        *        *     请求超时。
12    30 ms    31 ms    31 ms  220.181.38.251
```

1.3 协议层

1.3.1 协议栈（protocol stack）

因特网是个极其复杂的系统，因此因特网的体系存在分层的组织结构。类似一次乘飞机的过程，首先需要购票、托运行李、登机，飞行到目的地后，需要离机、认领行李，如果对航班不满意，还可以在向票务机构投诉。

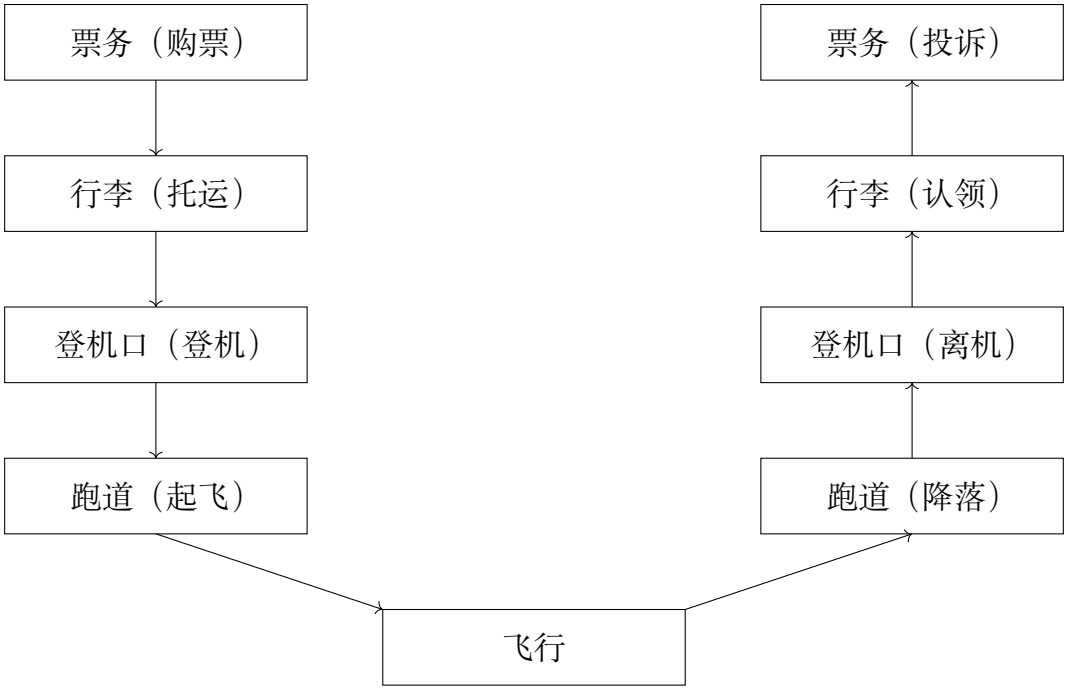

图 1.7: 航行流程

协议分层是为了使各层之间相互独立，每一层只专注于做一类事情。各层之间相互独立，各层之间不需要关心其它层是如何实现的，只需要知道自己如何调用下层提供好的功能就可以。同时协议分层提高了整体灵活性，每一层都可以使用最适合的技术来实现，只需要保证提供的功能以及暴露的接口的规则没有改变就行。

各层的所有协议被称为协议栈，TCP/IP 协议栈由 5 个层次组成，分为是物理层、链路层、网络层、运输层和应用层。

ISO（International Organization for Standard）为了更好地使网络应用更为普及，

推出了 OSI（Open System Interconnection）参考模型。但因为 OSI 七层模型出现地比 TCP/IP 五层模型晚，在 OSI 开始使用之前，TCP/IP 已经被广泛使用，最终 OSI 没有在实践中被广泛应用。

协议层	**功能**
应用层	提供网络服务操作接口
表示层	对要传输的数据进行处理
会话层	管理不同通讯节点之间的连接信息
传输层	建立不同节点之间的网络连接
网络层	将网络地址映射为 MAC 地址实现数据包转发
数据链路层	将要发送的数据包转为数据帧
物理层	利用物理设备实现数据的传输

表 1.1: OSI 七层模型

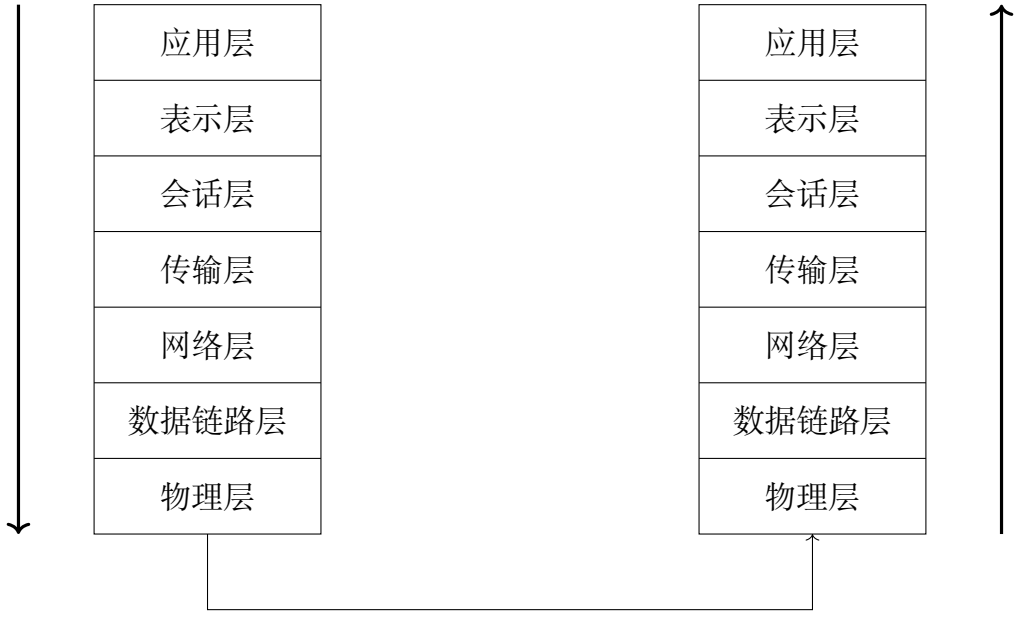

图 1.8: 数据传输

1.3.2 封装（Encapsulation）

在发送主机端，应用层报文（application-layer message）被传送给运输层，运输层收取到报文并附加首部信息，形成运输层报文段（transport-layer segment），被添加的首部将被接收端的运输层使用。首部信息包括了交付应用程序信息、差错检测位信息等。

运输层继续向网络层传递该报文段，网络层再添加首部信息，如源和目的端系统地址等，形成了网络层数据报（network-layer datagram）。

该数据报接下来被传递给链路层，链路层添加其首部信息，形成链路层帧（link-layer frame）。

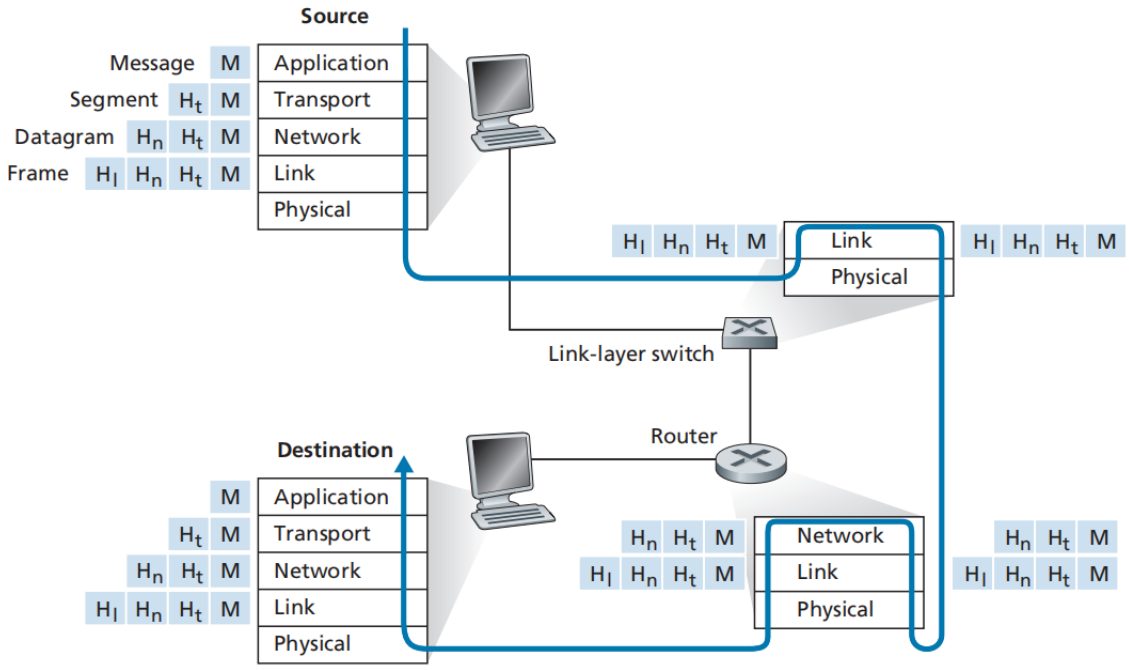

图 1.9: 封装

1.4 网络安全

1.4.1 网络攻击

2000 年 2 月 7 日，化名 MafiaBoy 的 15 岁加拿大少年攻击了 Yahoo、Amazon 和 eBay，使这些网站瘫痪达数小时，造成了超过 12 亿美元的损失，在股市中造成混乱。MafiaBoy 后来被透露为一名名叫 Michael Calce 的高中生，通过入侵几所大学的网络并利用其服务器进行 DDoS 攻击。这次袭击直接导致了今天许多网络犯罪法的制定。现在 MafiaBoy 已经金盆洗手，从事网络安全行业工作。

2017 年 1 月，University of Albert 内 20 个教室和实验室的 300 台电脑被安装恶意软件，导致近 3000 名学生的用户信息被盗。

2017 年 4 月，一名 Laurentian University 的 CS 学生为了证明学校的系统容易受到攻击，从而入侵了学校系统，导致近 2000 名学生的个人记录（包括密码、电话号码、成绩等）被泄漏。

因特网最初的设计理念是让一群相互信任的用户连接到一个透明的网络上，因此安全性并没有太过必要。然而如今的网络不再是完全透明和相互信任的，因此网络安全领域需要研究如何攻击计算机网络、如何防御免受攻击和如何设计能够更好地避免攻击的体系。

1.4.2 恶意软件

多数的恶意软件（malware）通过自我复制（self-replicating）传播，一旦设备感染了恶意软件，就有可能导致文件丢失、隐私泄漏等。恶意软件能够以病毒（virus）或蠕虫（worm）的形式传播。病毒是利用用户交互来感染用户设备的，例如包含恶意代码的电子邮件附件，如果用户无意打开附件，就会在其设备上运行恶意软件。另一种蠕虫则无需任何用户交互，例如用户在使用某些比较脆弱的网络应用程序时，该应用可能用因特网接受恶意软件并运行。

1.4.3 DoS 攻击

另一种通过攻击服务器和网络基础设施的威胁称为拒绝服务攻击（DoS, Denial-of-Service）。例如泛洪攻击，攻击者通过向目标主机发送大量的分组，堵塞目标的接入链路，使得合法分组无法达到服务器。但是如果服务器的接入速率非常大的话，单一的攻击源无法产生足够大的流量来伤害服务器。因此攻击者可以通过控制多个源向目标发送大量流量，这种方法称为分布式 DoS（DDoS, Distributed DoS）。

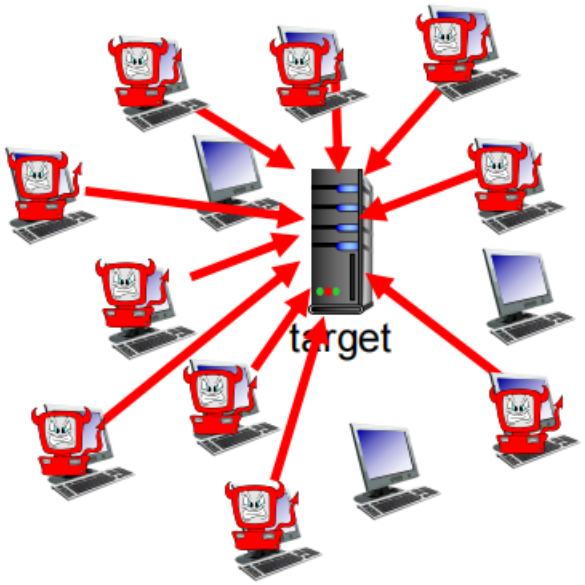

图 1.10: DDoS

1.4.4 IP 欺骗（IP Spoofing）

IP 欺骗是指伪造源 IP 地址，以便冒充其它系统或发件人的身份，从而冒充另外一台机器与服务器打交道。IP 欺骗会造成目标系统受到攻击却无法确认攻击源，或者取得目标系统的信任以便获取机密信息。IP 欺骗的防范，一方面需要目标设备采取更强有力的认证措施，不仅仅根据源 IP 就信任来访者，还需要更多的认证手段。

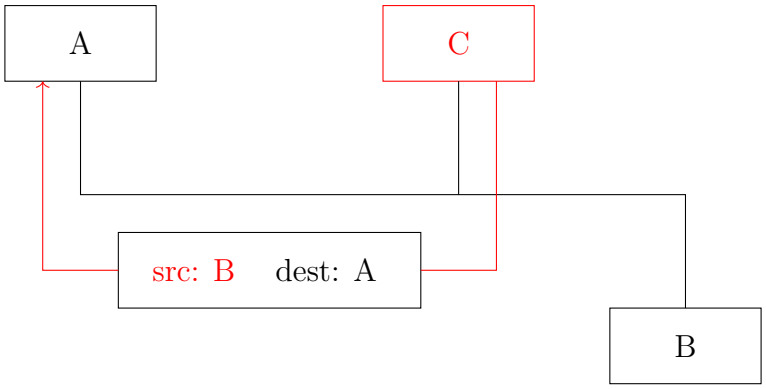

图 1.11: IP 欺骗

Chapter 2 应用层

2.1 应用层协议

2.1.1 网络应用

网络应用程序是指能够运行在不同端系统并通过网络彼此通信的程序。例如在 Web 应用程序中，有两个互相通信的程序，一个是运行在用户主机上的浏览器程序，另一个是运行在 Web 服务器主机上的服务器程序。

应用程序体系结构包括 Client-Server 和 P2P（Peer-to-Peer）两种体系结构。在 Client-Server 中，服务器主机一直运行处理客户端的请求。因为服务器需要有一个固定的的 IP 地址，因此客户端能够通过 IP 地址发送分组与其联系。著名的 Client-Server 应用程序包括 Web、FTP、Telnet 和电子邮件等。在 Client-Server 应用中，经常会出现一台单独的服务器主机不堪重负，跟不上所有客户请求的情况。例如搜索引擎（如 Google、Bing、Baidu）、电商（如 Amazon、eBay、Alibaba）、电子邮件（如 Gmail、Yahoo!）、社交媒体（如 Facebook、Instagram、Twitter、WeChat）都分步了多个数据中心，每数据中心都有数十万台服务器，必须要持续供电和维护。而在 P2P 中，主机之间的通信不必通过专门的服务器，很多流量密集型的应用都是 P2P 体系结构的，包括对等方协助下载加速器（如迅雷）、文件共享（如 BitTorrent）。

2.1.2 进程通信

在同一个端系统中，多个进程可以通过进程间通信的机制相互通信，然而在不同的端系统（可能有不同的操作系统）中，需要通过网络交换报文（message）。进程通过套接字（socket）接口向网络发送或接收报文。

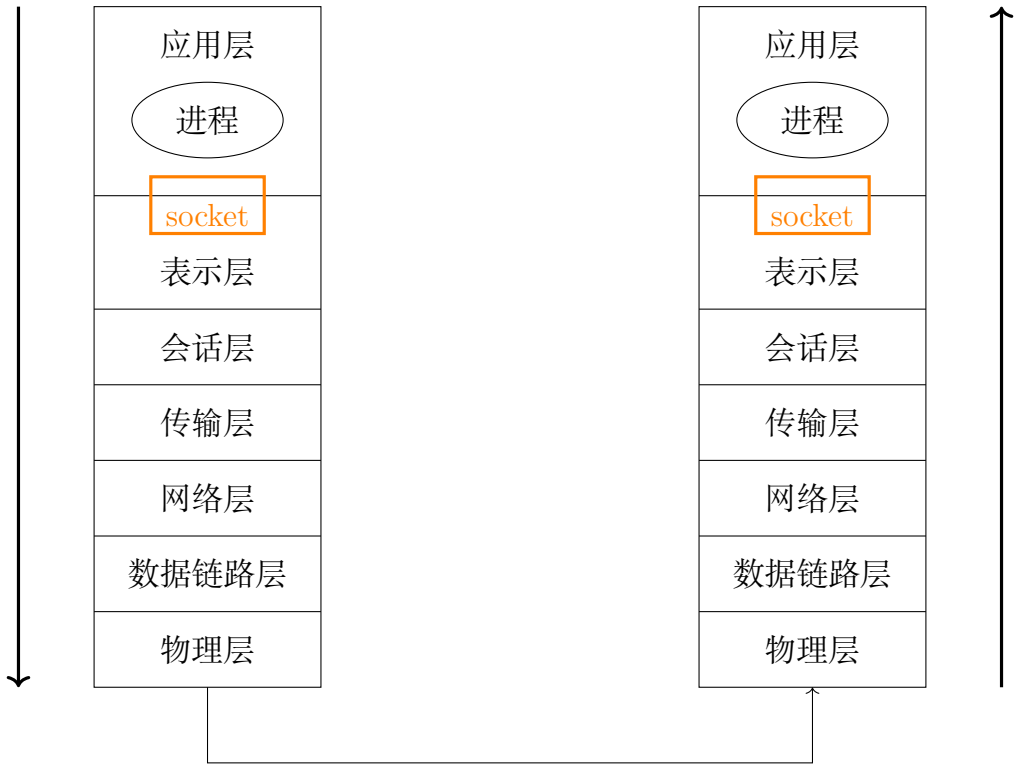

图 2.1: socket

在进程发送分组的过程中，必须要标识 IP 地址和端口号（port number）才能将分组发送给另一主机的进程。其中 IP 地址用于标识主机，端口号用于指定运行在目的主机上的接收进程。由于一台主机上会运行多个应用程序，因此端口号是不可或缺的信息。一些著名的应用已经被分配了特定的端口号，例如 Web 服务器使用端口号 80、邮件服务器（SMTP 协议）进程使用端口号 25。

发送端在使用 socket 时必须选择一种传输层协议，不同的协议会提供不同的服务。

一个传输层协议可以通过四个方面进行分类：

1. 可靠数据传输（reliable data transfer）：分组在网络传输中可能会因溢出或损坏等原因丢失，对于电子邮件、文件传输和金融相关的应用来说，数据丢失会造成灾难性的后果，这种情况下就必须采用可靠数据传输。对于一些可以容忍丢失（loss-tolerant）的应用，例如多媒体音视频，它们能够承受一定量的数据丢失，这只会造成小干扰，而非致命性的问题。

2. 吞吐量（throughput）：传输层协议能够以特定的速率提供服务。

3. 定时（timing）：传输层协议提供定时保证。例如在网络电话或多人游戏中，较长的时间延迟会出现不自然的停顿或失去真实感。

4. 安全性（security）：传输层协议保证数据安全。例如在发送端将数据加密，并在接收端解密数据，以防在传输被中途被窃听。

应用	**数据丢失**	**吞吐量**	**时间敏感**
文件传输	不允许丢失	弹性	不
电子邮件	不允许丢失	弹性	不
网络电话	容忍丢失	few kbps $\sim$ 1 Mbps	100 ms
视频会议	容忍丢失	10 kbps $\sim$ 5 Mbps	100 ms
交互式游戏	容忍丢失	few kbps $\sim$ 10 kbps	100 ms

表 2.1: 常见应用传输服务需求

2.1.3 TCP / UDP

TCP （Transmission Control Protocol）的特点包括：

- 面向连接服务（connection-oriented service）：在应用层数据包开始发送之前，TCP 让客户端和服务器之间互相交换传输层控制信息，让它们为分组的到来做好准备。在此之后，两个进程的 socket 就能建立起 TCP 连接，并可以发送报文了。在发送结束后，该 TCP 连接会被拆除。

- 可靠数据传输：TCP 确保了通信进程交付的数据无差错、不丢失、不重复、不乱序。

- 拥塞控制（congestion control）

UDP（User Datagram Protocol）的特点包括：

- 无连接（connectionless）

- 不可靠数据传输：UDP 不保证报文能够到达接收端，同时报文也有可能是乱序到达的。

- 无拥塞控制

应用	**传输协议**
文件传输	TCP
电子邮件	TCP
网络电话	UDP
视频会议	UDP
交互式游戏	UDP

表 2.2: 常见应用传输协议

2.2 HTTP

2.2.1 HTTP （HyperText Transfer Protocol）

一个 Web 页面是由对象（object）组成的，一个对象就是一个文件，例如 HTML 文件、JPEG 图片、JavaScript 文件、CSS 样式文件等。如果一个 Web 页面包含 1 个 HTML 文件和 5 个 JPEG 图片，那么这个 Web 页面就有 6 个对象。

每一个对象都可以通过 URL（Uniform Resource Locator）寻址，URL 地址由存放对象的服务器主机名（host name）和路径名（path name）组成。例如对于 http://www.someSchool.edu/someDepartment/picture.gif 而言，其中主机名就是 www.someSchool.edu，路径名是/someDepartment/picture.gif。

HTTP 定义在 RFC 1945、RFC 7230 和 RFC 7540 中。HTTP 使用 TCP 作为它的支撑传输协议，客户端首先发起 TCP 连接，连接建立后，客户进程可以向服务器进程发送 HTTP 请求报文，服务器进程可以向客户进程发送 HTTP 响应报文。

HTTP 是一个无状态协议（stateless protocol），服务器不能存储任何关于客户的状态信息。例如某个客户在短时间内两次请求同一个对象，服务器并不会因为第一次已经向客户提供了该对象而不再作出响应，而是再次重新发送对象。

2.2.2 HTTP 请求报文

HTTP 请求报文由第一行的请求行（request line）和后续的首部行（header line）组成，每行由一个回车（carriage return）和换行（line feed）结束，在首部行之后再附加一个只包含回车换行的空行。

HTTP 请求报文

```
1 GET /somedir/page.html HTTP/1.1\r\n
2 Host: www.someschool.edu\r\n
```

```
Connection: close\r\n
User-agent: Mozilla/5.0\r\n
Accept-language: fr\r\n
\r\n
[entity body]
```

请求行包含 3 个字段：

- 方法：包括 GET、POST、HEAD、PUT 和 DELETE。
- URL：带有请求对象的标识。
- HTTP 版本

首部行中 Host 指明了对象所在的主机。Connection: close 表示浏览器告诉服务器不要使用持续连接，而是要求服务器在发送完对象后就关闭此连接。User-agent 指明了用户代理，即向服务器发送请求的浏览器类型，例如 Mozilla/5.0。服务器可以通过此信息向用户代理发送不同版本的对象。Accept-language 表示用户想要获取对象的语言版本，如果服务器不存在的话，就会发送一个默认版本。

使用 GET 方法时，实体部分（entity body）为空。而使用 POST 方法时，实体部分可以用于包含用户在表单中填写的输入值。HEAD 方法与 GET 类似，服务器会响应，但并不返回请求对象。因此 HEAD 方法常用于调试跟踪。PUT 方法允许用户向服务器上传对象。DELETE 方法允许用户删除服务器上的对象。

2.2.3 HTTP 响应报文

HTTP 响应报文由状态行（status line）、首部行（header line）和实体部分组成。

HTTP 响应报文

```
HTTP/1.1 200 OK\r\n
Connection: close\r\n
Date: Tue, 18 Aug 2015 15:44:04 GMT\r\n
Server: Apache/2.2.3 (CentOS)\r\n
```

```
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT\r\n
Content-Length: 6821\r\n
Content-Type: text/html\r\n
\r\n
(data data data data data ...)
```

状态行包括了协议版本、状态码和对应状态信息。首部行中 Connection: close 用于告诉客户，发送完报文后将会关闭 TCP 连接。Date 用于表示服务器生成并发送该报文的日期时间。Last-Modified 用于表示对象创建或最后修改的日期时间。Content-Length 用于表示被发送对象的字节数。Content-Type 用于表示对象的类型。

状态码	**含义**
200 OK	请求成功
204 No Content	无内容
301 Moved Permanently	永久性重定向，资源被分配了新 URL
400 Bad Request	请求语法错误，服务器无法理解
403 Forbidden	拒绝执行请求
404 Not Found	无法找到资源
500 Internal Server Error	服务器内部错误
503 Service Unavailable	由于超载或系统维护，暂时无法处理请求
505 HTTP Version not supported	不支持请求的 HTTP 协议的版本

表 2.3: 常见 HTTP 状态码

2.3 Cookie

2.3.1 Cookie

HTTP 服务器是无状态的，然而一些 Web 站点通常希望能够识别用户，从而记住用户信息或限制用户访问。目前 Cookie 广泛用于记录用户登录信息，这样下次访问时可以不再输入用户名和密码了。当然这种方便也存在用户信息泄密的问题，尤其在多个用户公用一台电脑时很容易出现这样的情况。

例如用户首次访问 Amazon，HTTP 响应报文中会包含 Set-cookies 识别码，浏览器会将识别码添加到所管理的文件中。当用户继续浏览 Amazon 时，每一个 HTTP 请求浏览器都会从 cookie 文件中查询该网站的识别码，并添加到 HTTP 请求报文中。Amazon 也可以通过 cookie 来维护用户希望购买的商品信息，并推荐个性化产品。

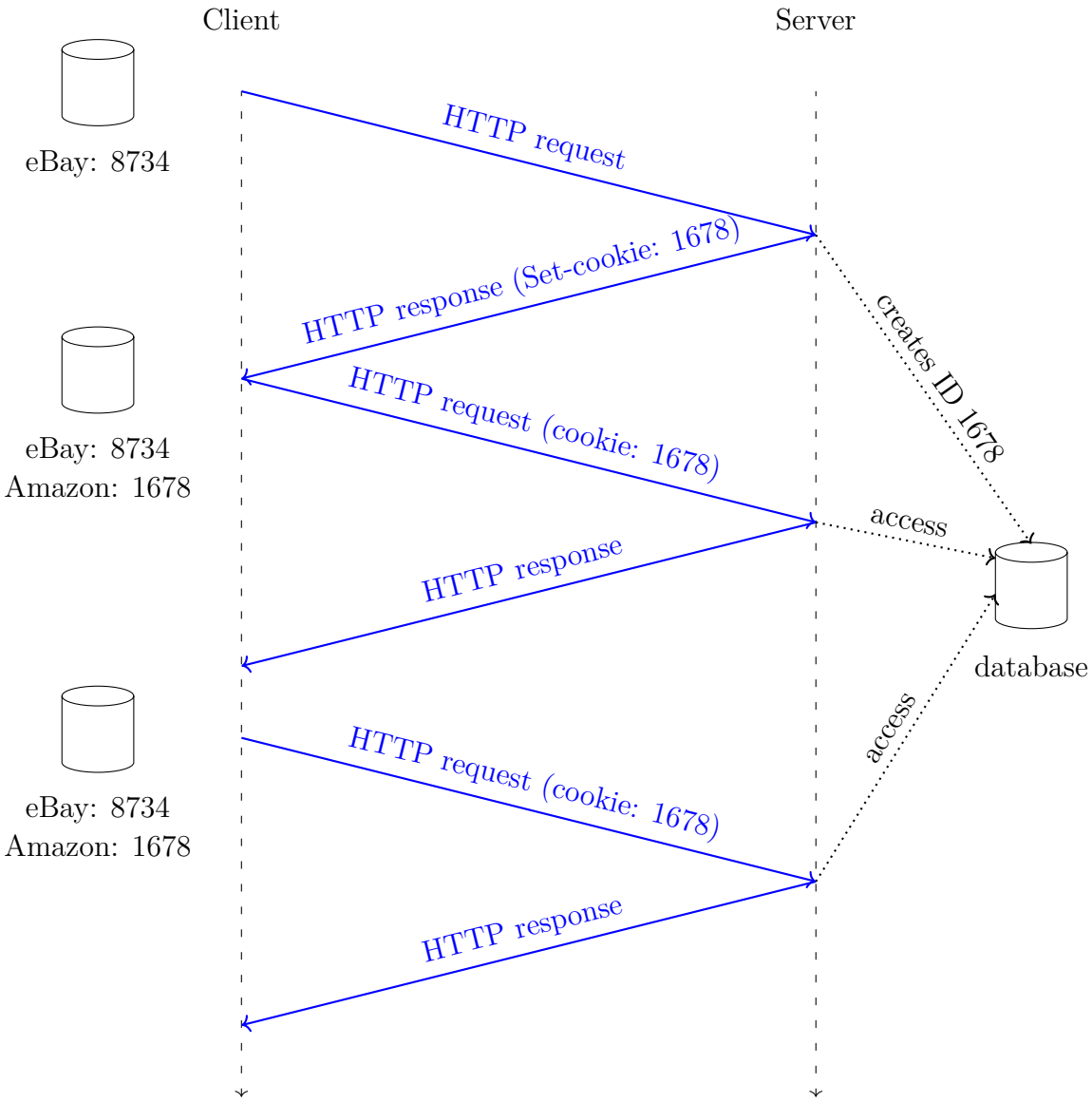
Client
Server
eBay: 8734
HTTP request
HTTP response (Set-cookie: 1678)
creates ID 1678
eBay: 8734
Amazon: 1678
HTTP request (cookie: 1678)
access
HTTP response
database
eBay: 8734
Amazon: 1678
HTTP request (cookie: 1678)
access
HTTP response

图 2.2: Cookie

2.4 Email

2.4.1 Email