计算机网络

(第四版)

习题答案

1-3 The performance of a client-server system is influenced by two network factors: the bandwidth of the network (how many bits/sec it can transport) and the latency (how many seconds it takes for the first bit to get from the client to the server). Give an example of a network that exhibits high bandwidth and high latency. Then give an example of one with low bandwidth and low latency.

客户-服务器系统的性能会受到两个网络因素的影响：网络的带宽（每秒可以传输多少位数据）和延迟（将第一个数据位从客户端传送到服务器端需要多少秒时间）。请给出一个网络的例子，

它具有高带宽和高延迟。然后再给出另一个网络的例子，它具有低带宽和低延迟。

答：横贯大陆的光纤连接可以有很多千兆位 /秒带宽， 但是由于光速度传送要越过数千公里，时

延将也高。相反，使用 56 kbps调制解调器呼叫在同一大楼内的计算机则有低带宽和较低的时延。

1-4 Besides bandwidth and latency, what other parameter is needed to give a good characterization of the quality of service offered by a network used for digitized voice traffic?

除了带宽和延迟以外， 针对数字化的语音流量， 想要让网络提供很好的服务质量， 还需要哪个参

数？

声音的传输需要相应的固定时间， 因此网络时隙数量是很重要的。 传输时间可以用标准偏差方式

表示。 实际上，短延迟但是大变化性比更长的延迟和低变化性更糟。

1-6 A client-server system uses a satellite network, with the satellite at a height of 40,000 km. What is the best-case delay in response to a request?

一个客户－服务器系统使用了卫星网络， 卫星的高度为40000km。 在对一个请求进行响应的时候，

最佳情形下的延迟是什么？

答：由于请求和应答都必须通过卫星，因此传输总路径长度为 160,000千米。在空气和真空中的

光速为 300，000 公里/秒， 因此最佳的传播延迟为 160,000/300，000秒，约 533 msec。

1-9 A group of 2n - 1 routers are interconnected in a centralized binary tree, with a router at each tree node. Router i communicates with router j by sending a message to the root of the tree. The root then sends the message back down to j. Derive an approximate expression for the mean number of hops per message for large n, assuming that all router pairs are equally likely.

在一个集中式的二叉树上，有 2n-1个路由器相互连接起来；每个树节点上都有一个路由器。路由

器i为了与路由器j进行通信，它要给树的根发送一条信息。然后树根将消息送下来给 j。假设所有的路由器都是等概率出现的，请推导出当 n很大时每条消息的平均跳数的一个近似表达式。

答：这意味着，从路由器到路由器的路径长度相当于路由器到根的两倍。 若在树中，根深度为1，深度为 n，从根到第 n 层需要 n-1 跳，在该层的路由器为 0.50(50%)，从根到 n-1 层的路径的路由器为 0.25(25%)和 n --2 跳步。 因此，路径长度 l 为：

$$l = 0.5 \times (n-1) + 0.25 \times (n-2) + 0.125 \times (n-3) + \cdots$$

或

$$l = \sum_{i=1}^{\infty} n\,(0.5)^i - \sum_{i=1}^{\infty} i(0.5)^i$$

表达式归约为 l＝n－2，平均的路由器到路由器路径为 2n-4。

1-11 What are two reasons for using layered protocols?

请说出使用分层协议的两个理由？

答：通过协议分层可以把设计问题划分成较小的易于处理的片段。 分层意味着某一层协议的改变不会影响高层或低层的协议。

1-13 What is the principal difference between connectionless communication and connection-oriented communication?

在无连接通信和面向连接的通信二者之间，最主要的区别是什么？

答：主要的区别有两条。

其一：面向连接通信分为三个阶段，第一是建立连接，在此阶段，发出一个建立连接的请求。只有在连接成功建立之后，才能开始数据传输，这是第二阶段。接着，当数据传输完毕，必须释放连接。而无连接通信没有这么多阶段，它直接进行数据传输。

其二：面向连接的通信具有数据的保序性， 而无连接的通信不能保证接收数据的顺序与发送数据的顺序一致。

1-14 Two networks each provide reliable connection-oriented service. One of them offers a reliable byte stream and the other offers a reliable message stream. Are these identical? If so, why is the distinction made? If not, give an example of how they differ.

两个网络都可以提供可靠的面向连接的服务。 其中一个提供可靠的字节流， 另一个提供可靠的报

文流。这二者是否相同？如果你认为相同的话， 为什么要有这样的区别？如果不相同， 请给出一个例子说明它们如何不同。

答：不相同。在报文流中，网络保持对报文边界的跟踪；而在字节流中，网络不做这样的跟踪。例如，一个进程向一条连接写了 1024 字节，稍后又写了另外 1024 字节。那么接收方共读了 2048 字节。对于报文流，接受方将得到两个报文。每个报文 1024 字节。 而对于字节流，报文边界不被识别。接收方把全部的 2048 个字节当作一个整体，在此已经体现不出原先有两个报文的事实。

1-17 In some networks, the data link layer handles transmission errors by requesting damaged frames to be retransmitted. If the probability of a frame's being damaged is p, what is the mean number of transmissions required to send a frame? Assume that acknowledgements are never lost.

在有些网络中，数据链路层处理传输错误的做法是， 请求重传被损坏的帧。如果一帧被损坏的概率为p，那么发送一帧所需要的平均传输次数是多少？假设确认帧永远不会丢失。

帧请求正好是 k 次的概率 Pk，就是起初的 k-1 次尝试都失败的概率。 pk-1，乘以第 k 次传输成功的概率。平均传输次数就是

$$\sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} k(1-p)p^{k-1} = \frac{1}{1-p}$$

1-22 What is the main difference between TCP and UDP?

TCP和UDP之间最主要的区别是什么？

TCP 是面向连接的，而 UDP 是一种数据报服务。

1-25 When a file is transferred between two computers, two acknowledgement strategies are possible. In the first one, the file is chopped up into packets, which are individually acknowledged by the receiver, but the file transfer as a whole is not acknowledged. In the second one, the packets are not acknowledged individually, but the entire file is acknowledged when it arrives. Discuss these two approaches.

当一个文件在两台计算机之间传输的时候， 可能会有两种不同的确认策略。 在第一种策略中，该文件被分解成许多个分组，接收方会独立地确认每一个分组， 但是文件传输过程作为整体并没有被确认。在第二种策略中，这些分组并没有被单独地确认，但是当整个文件到达的时候，它会被确认。请讨论这两种方案。

如果网络容易丢失分组，那么对每一个分组逐一进行确认较好， 此时仅重传丢失的分组。而在另一方面，如果网络高度可靠，那么在不发差错的情况下，仅在整个文件传送的结尾发送一次确认，从而减少了确认的次数，节省了带宽；不过，即使有单个分组丢失，也需要重传整个文件。

1-27 How long was a bit on the original 802.3 standard in meters? Use a transmission speed of 10 Mbps and assume the propagation speed in coax is 2/3 the speed of light in vacuum.

在原始的802.3标准中，一位是多长（按米来计算）？请使用 10Mbps的传输速率，并且假设同轴电缆的传播速度是真空中光速的 2/3.

波在同轴电缆中的速度是大约 200,000 km/sec，即 200 m/μsec。在 10 Mbps, 传输一位需要 0.1 μsec 因此，这个位在时间上持续 0.1 μsec，在此期间传播 20 meters。因此，这里的一位是 20 米。

1-28 An image is 1024 x 768 pixels with 3 bytes/pixel. Assume the image is uncompressed. How long does it take to transmit it over a 56-kbps modem channel? Over a 1-Mbps cable modem? Over a 10-Mbps Ethernet? Over 100-Mbps Ethernet?

一幅图像的分辨率为 1024×768像素，每个像素用 3字节来标识。假设该图像没有被压缩。请问，通过56kbps的调制解调器信道来传输这幅图像需要多长时间？通过 1Mbps的电缆调制解调器（cable modem）呢？通过10Mbps的因特网呢？通过100Mbps的因特网呢？

图像是 1024×768× 3 bytes，或 2,359,296 bytes.就是 18,874,368 bits.在 56,000 bits/sec速度下，传输需要大约 337.042 sec.在 1,000,000 bits/sec需要大约 18.874 sec.在 10,000,000 bits/sec需要大约 1.887 sec.在 100,000,000 bits/sec需要大约 0.189 sec.

1-30 Wireless networks are easy to install, which makes them inexpensive since installation costs usually far overshadow equipment costs. Nevertheless, they also have some disadvantages. Name two of them.

无线网络很容易安装，这使得它们并不非常昂贵。因为安装费用通常会占去整个设备费用的很大比例。然而，它们也有一些缺点。请说出两个缺点。

一个缺点是安全性。每个碰巧在此房屋内的随机发送者都能在网络上监听。另一个缺点是可靠性。无线网络造成大量错误。第三个潜在的问题是电池寿命，因为多数无线设备倾向于可移动性。

1-31 List two advantages and two disadvantages of having international standards for network protocols.

请列举出网络协议国际标准化的两个优点和缺点。

优点 1：如果每个人都使用标准，那么每个人都可以与其他任何人交流；优点 2：广泛使用标准将导致规模经济，比如生产大规模集成电路芯片。缺点 1：为了取得标准化所需要的政治妥协经常会导致差的标准；缺点 2：一旦标准被广泛采用了，要对它再做改变就会非常困难，即使发现了新的更好的技术或方法，也难以替换。

第 2 章 物理层

2-2 A noiseless 4-kHz channel is sampled every 1 msec. What is the maximum data rate?

一条无噪声 4kHz信道按照每 1ms一次进行采样，请问最大数据传输率是多少？

答：无噪声信道最大数据传输率公式：最大数据传输率 =2Hlog2V b/s。因此最大数据传输率决定于每次采样所产生的比特数，如果每次采样产生 16bits，那么数据传输率可达 128kbps；如果每次采样产生 1024bits，那么可达 8.2Mbps。注意这是对无噪声信道而言的，实际信道总是有噪声的，其最大数据传输率由香农定律给出。

2-3 Television channels are 6 MHz wide. How many bits/sec can be sent if four-level digital signals are used? Assume a noiseless channel.

电视频道的带宽是 6MHz。如果使用4级数字信号，则每秒钟可以发送多少位？假设电视频道为无噪声信道。

答：采样频率 12MHz，每次采样 2bit，总的数据率为 24Mbps。

2-4 If a binary signal is sent over a 3-kHz channel whose signal-to-noise ratio is 20 dB, what is the maximum achievable data rate?

如果在一条3kHz信道上发送一个二进制信号，该信道的信噪比为 20dB，则最大可达到的数据传输率为多少？

答：信噪比为 20 dB 即 S/N = 100.由于 log2101≈6.658，由香农定理，该信道的信道容量为

3log2(1 +100) =19.98kbps。

又根据乃奎斯特定理，发送二进制信号的 3kHz 信道的最大数据传输速率为

2*3 log22=6 kbps。

所以可以取得的最大数据传输速率为 6kbps。

2-5 What signal-to-noise ratio is needed to put a T1 carrier on a 50-kHz line?

在50kHz的线路上使用 T1线路需要多大的信噪比？

答：为发送 T1 信号，我们需要

$$H\log_2(1+\frac{S}{N})=1.544\times10H^6$$

$$H=50000$$

$$\frac{S}{N}=2^{31}-1$$

$$10\log_{10}(2^{31}-1)=93\,\mathrm{dB}$$

所以，在 50kHz 线路上使用 T1 载波需要 93dB 的信噪比。

2-7　How much bandwidth is there in 0.1 micron of spectrum at a wavelength of 1 micron?

在1μm波长上，在0.1μm的频段中有多少带宽？

答：

$$f=\frac{c}{\lambda}\qquad \frac{df}{d\lambda}=\frac{c}{\lambda^2}$$

$$df=\frac{c}{\lambda^2}d\lambda\quad \Delta f=\frac{c}{\lambda^2}\Delta\lambda$$

$$c=3\times10^8\qquad \lambda=10^{-6}m$$

$$\Delta\lambda=0.1\times10^{-6}=10^{-7}m$$

$$\Delta f=\frac{3\times10^8}{(10^{-6})^2}\times10^{-7}=30\times10^{12}Hz=30THz$$

因此，在 0.1μm的频段中可以有 30THz。

2-8　It is desired to send a sequence of computer screen images over an optical fiber. The screen is 480 x 640 pixels, each pixel being 24 bits. There are 60 screen images per second. How much bandwidth is needed, and how many microns of wavelength are needed for this band at 1.30 microns?

现需要在一条光纤发送一系列计算机屏幕图像，屏幕的分辨率为 480× 640像素，每个像素为 24

位。每秒钟有60幅屏幕图像。请问，需要多少带宽？在 1.30μm波长上，这段带宽需要多少 μm

的波长?

答：数据速率为 480×　640×24× 60bps 即 442Mbps。

$$\Delta f=4.42\times10^8$$

$$f=\frac{c}{\lambda}\qquad \frac{df}{d\lambda}=-\frac{c}{\lambda^2}$$

$$|\Delta f|=\frac{\lambda^2\Delta f}{c}=\frac{(1.3\times10^{-6})^2\times4.42\times10^8}{3\times10^8}=2.5\times10^{-12}m=2.5\times10^{-6}\,\mu\mathrm{m}$$

需要 442Mbps 的带宽，对应的波长范围是 $2.5\times10^{-6}\,\mu\mathrm{m}$。

2-18 A simple telephone system consists of two end offices and a single toll office to which each end office is connected by a 1-MHz full-duplex trunk. The average telephone is used to make four calls per 8-hour workday. The mean call duration is 6 min. Ten percent of the calls are long-distance (i.e., pass

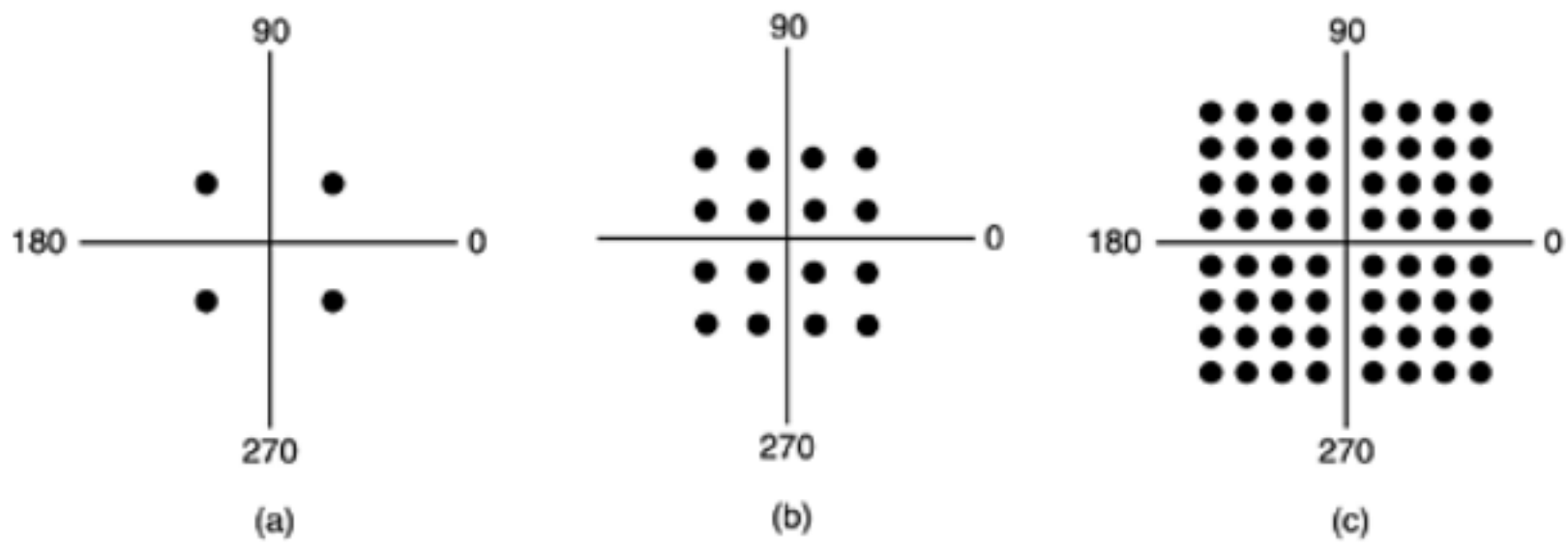
through the toll office). What is the maximum number of telephones an end office can support? (Assume 4 kHz per circuit.)

一个简单的电话系统包括两个端局和一个长途局，每个端局通过一条 1MHz全双工干线连接到长途局。在每8小时的工作日中，平均每部电话有 4次呼叫，每次呼叫平均6分钟，10%的呼叫是长途（即通过长途局）。请问一个端局能够支持最多多少部电话？（假设每条线路为 4kHz）

答：每部电话每小时做 0.5 次通话，每次通话 6 分钟。因此一部电话每小时占用一条电路 3 分钟，60/3=20，即 20 部电话可共享一条线路。由于只有 10%的呼叫是长途，所以 200 部电话占用一条完全时间的长途线路。局间干线复用了 1000000/4000=250条线路，每条线路支持 200 部电话，因此，一个端局可以支持的电话部数为 200*250=50000。

2-22 A modem constellation diagram similar to Fig. 2-25 has data points at the following coordinates: (1, 1), (1, -1), (-1, 1), and (-1, -1). How many bps can a modem with these parameters achieve at 1200 baud?

Figure 2-25. (a) QPSK. (b) QAM-16. (c) QAM-64.



一个类似于图2.25的调制解调器星座图有以下几个坐标点（ 1，1）、（1，-1）、（-1，1）、（-1，-1）。请问一个具备这些参数的调制解调器在 1200波特上可以达到多少 bps？

每个波特有 4 个合法值，因此比特率是波特率的两倍。对应于 1200 波特，数据速率是 2400bps。

2-28 Ten signals, each requiring 4000 Hz, are multiplexed on to a single channel using FDM. How much minimum bandwidth is required for the multiplexed channel? Assume that the guard bands are 400 Hz wide.

有10个信号，每个都要求4000Hz，现在用FDM将它们复用在一条信道上。对于被复用的信道，

最小要求多少带宽？假设防护频段为 400Hz宽。

有 10 个 4000Hz信号。我们需要 9 个防护频段来避免干扰。最小带宽需求是 4000×10+400×9 =43,600 Hz.

2-29 Why has the PCM sampling time been set at 125 μsec?

29. 为什么 PCM 采样时间被设置为 125μs?

答：125μs的采样时间对应于每秒 8000 次采样。一个典型的电话通道为 4kHz。根据奈奎斯特定理，为获取一个 4kHz 的通道中的全部信息需要每秒 8000 次的采样频率。

(实际上额定带宽稍有些少，截止点并不清晰)

2-30 What is the percent overhead on a T1 carrier; that is, what percent of the 1.544 Mbps are not delivered to the end user?

30. T1 线路上额外开销的百分比为多少？也就是说，1.541Mbps 中有百分之多少没有被递交给最终用户？

每一帧中，端点用户使用 193 位中的 168（7*24）位，开销占 25（=193-168）位，因此开销比例

等于 25/193=13%。

2-33 What is the difference, if any, between the demodulator part of a modem and the coder part of a codec? (After all, both convert analog signals to digital ones.)

33. 调制解调器的解调部分与编解码器的编码部分有没有区别？如果有的话，区别是什么？（之所以有此问，是因为两者都将模拟信号转换成数字信号。）

答：有。编码器接受任意的模拟信号，并从它产生数字信号。而解调器仅仅接受调制了的正弦（或余弦）波，产生数字信号。

2-34 A signal is transmitted digitally over a 4-kHz noiseless channel with one sample every 125 μs. How many bits per second are actually sent for each of these encoding methods?

(a) CCITT 2.048 Mbps standard.(b) DPCM with a 4-bit relative signal value.(c) Delta modulation.

34. 一个信号在4kHz的无噪声信道上以数字方式进行传输，每125μs采样一次。请问，按照以下的编码方法，每秒钟实际发送多少位？

(c) CCITT 2.048Mbps标准。

(d) 有4位相对信号值的DPCM。

(e) 增量调制。

答：a. CCITT 2.048Mbps 标准用 32 个 8 位数据样本组成一个 125μs 的基本帧，30 个信道用于传信息，2 个信道用于传控制信号。在每一个 4kHz 信道上发送的数据率就是

8*8000=64kbps。

b. 差分脉码调制（DPCM）是一种压缩传输信息量的方法，它发送的不是每一次抽样的二进制编码值，而是两次抽样的差值的二进制编码。现在相对差值是 4 位，所以对应每个 4kHz 信道实际发送的比特速率为 4*8000=32bps。

c. 增量调制的基本思想是：当抽样时间间隔 s t 很短时，模拟数据在两次抽样之间的变化很小，可以选择一个合适的量化值？作为阶距。把两次抽样的差别近似为不是增加一个 ?就是减少一个? 。这样只需用 1bit 二进制信息就可以表示一次抽样结果，而不会引入很大误差。因此，此时对应每个 4kHz 信道实际发送的数据速率为 1*8000=8kHz。

2-39What is the essential difference between message switching and packet switching?

39. 消息交换和分组交换之间的本质区别是什么?

信息交换发送到数据单元可以是任意长度。分组交换有最大报文大小限制，任何大于限制的信息将被拆分成多个报文。

2-41 Three packet-switching networks each contain n nodes. The first network has a star topology with

a central switch, the second is a (bidirectional) ring, and the third is fully interconnected, with a wire from every node to every other node. What are the best-, average-, and-worst case transmission paths in hops?

三个分组交换网络每个包含 n个节点。第一个网络是一个星型拓扑结构，有一个中心交换机；第

二个网络是一个双向环；第三个网络是一个全连接结构， 从任何一个节点到其他的节点都有一条

线路。请问从传输路径的跳数来看，哪个最好？其次？最差？

答：The three networks have the following properties:

星型：最好为 2，最差为 2，平均为 2；

环型：最好为 1，最差为 n/2，平均为 n/4

如果考虑 n 为奇偶数，

则 n 为奇数时，最坏为（n-1）/2，平均为（n+1）/4

n 为偶数时，最坏为 n/2，平均为 n2/4(n 1)

全连接：最好为 1，最差为 1，平均为 1。

2-42 Compare the delay in sending an x-bit message over a k-hop path in a circuit-switched network and in a (lightly loaded) packet-switched network. The circuit setup time is s sec, the propagation delay is d sec per hop, the packet size is p bits, and the data rate is b bps. Under what conditions does the packet network have a lower delay?

请比较一下在一个电路交换网络中和在一个负载较轻的分组交换网络中，沿着 k跳到路径发送x

位消息的延迟情况。电路建立的时间为 s秒，每一跳的传播延迟为 d秒，分组的大小为 p位，数据

传输率为b bps。 在什么条件下分组网络的延迟比较短？

对于电路交换， t= s时电路建立起来；t ＝s+ 时报文的最后一位发送完毕；t =

s+ x/b+kd 时报文到达目的地。而对于分组交换，最后一位在 t=x/b 时发送完毕。

为到达最终目的地，最后一个分组必须被中间的路由器重发 k 1次，每次重发花时间 p/ b，所

以总的延迟为

$$\frac{x}{b}+(k-1)\frac{p}{b}+kd$$

为了使分组交换比电路交换快，必须：

$$\frac{x}{b}+(k-1)\frac{p}{b}+kd<s+\frac{x}{b}+kd$$

所以：

$$s>(k-1)\frac{p}{b}$$

2-43 Suppose that x bits of user data are to be transmitted over a k-hop path in a packet-switched network as a series of packets, each containing p data bits and h header bits, with x p + h. The bit rate of the lines is b bps and the propagation delay is negligible. What value of p minimizes the total delay?

假定x位用户数据将以一系列分组的形式，在一个分组交换网络中沿着一条共有 k跳到路径向前传输，每个分组包含 p位数据和h位的头，这里x>>p+h。线路的传输率为 b bps，传播延迟忽略不计。请问什么样的p值使总延迟最小？

答：所需要的分组总数是 x /p ，因此总的数据加上头信息交通量为 (p+h)x/p 位。

源端发送这些位需要时间为 (p+h)x /pb ;中间的路由器重传最后一个分组所花的总时间为 (k-1)(p +h)/ b

因此我们得到的总的延迟为

$$(p+h)\frac{x}{pb}+(p+h)(k-1)\frac{1}{b}$$

对该函数求 p 的导数，得到

$$\frac{p-(p+h)}{p^2}\frac{x}{b}+\frac{k-1}{b}$$

令

$$\frac{p-(p+h)}{p^2}\frac{x}{b}+\frac{k-1}{b}=0$$

得到

$$\frac{hx}{p^2}=k-1$$

因为 p>0，所以

$$p=\sqrt{\frac{hx}{k-1}}$$

故

$$p=\sqrt{\frac{hx}{k-1}}$$

时能使总的延迟最小。

2-44 In a typical mobile phone system with hexagonal cells, it is forbidden to reuse a frequency band in an adjacent cell. If 840 frequencies are available, how many can be used in a given cell?

在一个典型的移动电话系统中， 蜂窝单元为六角形，在相邻的单元内禁止重新使用频段。 如果总

共有840个频率可以使用的话，则任何一个给定的单元内可以使用多少个频率？

每个单元有 6 个邻居。如果中间的单元使用频段组合 A，它的六个邻居可以分别使用的频段组合 B, C, B, C, B, C。换句话说，只需要 3 个单一的单元。因此，每个单元可以使用 280 个频率。

2-50 Suppose that A, B, and C are simultaneously transmitting 0 bits, using a CDMA system with the chip sequences of Fig. 2-45(b). What is the resulting chip sequence?

A: (–1 –1 –1 +1 +1 –1 +1 +1)
B: (–1 –1 +1 –1 +1 +1 +1 –1)
C: (–1 +1 –1 +1 +1 +1 –1 –1)
D: (–1 +1 –1 –1 –1 –1 +1 –1)

(b)

FIG 2-45(b)

50. 假设 A、B 和 C 通过一个 CDMA 系统同时传输位 0，他们的时间片序列如图 2.45(b)所示。请问结果得到的时间片序列是什么？

结果是通过对 A、B、C 求反再将这三个码片序列相加得到的。

结果是(+3 +1 +1 1 3 1 1 +1).

2-53 A CDMA receiver gets the following chips: (-1 +1 -3 +1 -1 -3 +1 +1). Assuming the chip sequences defined in Fig. 2-45(b), which stations transmitted, and which bits did each one send?

一个CDMA 接收器得到了下面的时间片（ -1+1-3+1-1-3+1+1）。假设时间片序列如图 2.45b 中所定

义，请问那些移动站传输了数据？每个站发送了什么位？

Just compute the four normalized inner products.

此处答案中的~疑为-号之误？

(1 +1 3 +1 1 1 1 +1 +1 1 +1 +1)/8
(1 +1 3 +1 1 1 +1 1 +1 +1 +1 1)/8 = 1
(1 +1 3 +1 1 1 +1 1 +1 +1 +1 1 1)/8 = 0
(1 +1 3 +1 1 1 +1 1 1 1 1 +1 1)/8 = 1

结果是 A 和 D 发送了 1 位, B 发送了 0 位, C 没有发送。

第 3 章 数据链路层

3-1 An upper-layer packet is split into 10 frames, each of which has an 80 percent chance of arriving undamaged. If no error control is done by the data link protocol, how many times must the message be sent on average to get the entire thing through?

一个上层的分组被切分成 10帧，每一帧有80%的机会可以无损地到达。如果数据链路协议没有提供错误控制的话，请问，该报文平均需要发送多少次才能完整地到达接收方？

答：由于每一帧有 0.8 的概率正确到达，整个信息正确到达的概率为 p=0.810=0.107

为使信息完整的到达接收方，发送一次成功的概率是 p ，二次成功的概率是(1-p)p，三次成功的概率为(1-p)2 p，i 次成功的概率为(1-p)i-1 p，因此平均的发送次数等于：

$$E = \sum_{i=1}^{\infty} ip(1-p)^{i-1} = \frac{1}{p} = \frac{1}{0.107} \approx 9.3$$

3-2The following character encoding is used in a data link protocol: A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000 Show the bit sequence transmitted (in binary) for the four-character frame: A B ESC FLAG when each of the following framing methods are used:

(a) Character count. (b) Flag bytes with byte stuffing. (c) Starting and ending flag bytes, with bit stuffing.

2. 数据链路协议中使用了下面的字符编码：

A：01000111；B：11100011；FLAG：01111110；ESC：11100000

为了传输一个包含 4 个字符的帧：A B ESC FLAG，请给出当使用下面的成帧方法时所对应的位序列（用二进制表达）：

(a) 字符计数。

(b) 包含字节填充的标志字节。

(c) 包含位填充的起始和结束标志。

结果是

(a) 00000100 01000111 11100011 11100000 01111110

(b) 01111110 01000111 11100011 11100000 11100000 11100000 01111110 01111110

(c) 01111110 01000111 110100011 111000000 011111010 01111110

3-5 A bit string, 0111101111101111110, needs to be transmitted at the data link layer. What is the string actually transmitted after bit stuffing?

位串0111101111101111110需要在数据链路层上被发送，请问，经过位填充之后实际被发送出去的是什么？

输出是 1110111110011111010.

3-6　When bit stuffing is used, is it possible for the loss, insertion, or modification of a single bit to cause an error not detected by the checksum? If not, why not? If so, how? Does the checksum length play a role here?

假设使用了位填充成帧方法，请问，因为丢失一位，插入一位，或者篡改一位而引起的错误是否

有可能通过校验和检测出来？如果不能的话， 请问为什么？如果能的话， 请问校验和长度在这里

是如何起作用的？

答：可能。假定原来的正文包含位序列 01111110 作为数据。位填充之后，这个序列将变成 01111010。如果由于传输错误第二个 0 丢失了，收到的位串又变成 01111110，被接收方看成是帧尾。然后接收方在该串的前面寻找检验和，并对它进行验证。如果检验和是 16 位，那么被错误的看成是检验和的 16 位的内容碰巧经验证后仍然正确的概率是 1/216。如果这种概率的条件成立了，就会导致不正确的帧被接收。显然，检验和段越长，传输错误不被发现的概率会越低，但该概率永远不等于零。

3-16 Data link protocols almost always put the CRC in a trailer rather than in a header. Why?

数据链路协议几乎总是将 CRC放在尾部，而不是头部，为什么？

答：CRC 是在发送期间进行计算的。一旦把最后一位数据送上外出线路， 就立即把 CRC 编码附加在输出流的后面发出。如果把 CRC 放在帧的头部，那么就要在发送之前把整个帧先检查一遍来计算 CRC。这样每个字节都要处理两遍，第一遍是为了计算检验码，第二遍是为了发送。把 CRC 放在尾部就可以把处理时间减半。

3-17 A channel has a bit rate of 4 kbps and a propagation delay of 20 msec. For what range of frame

sizes does stop-and-wait give an efficiency of at least 50 percent?

一个信道的位速率为 4kbps，传输延迟为20ms。请问帧的大小在什么范围内，停 -等协议才可以获得至少50%的效率？

答：当发送一帧的时间等于信道的传播延迟的 2 倍时，信道的利用率为 50%。或者说，当发送一帧的时间等于来回路程的传播延迟时， 效率将是 50%。而在帧长满足发送时间大于延迟的两倍时，效率将会高于 50%。

现在发送速率为 4Mb/s，发送一位需要 0.25μs 。

$$(20\times10^{-3}\times2)\div(0.25\times10^{-6})=160000 \quad \text{bit}$$

只有在帧长不小于 160kb 时，停等协议的效率才会至少达到 50%。

3-18 A 3000-km-long T1 trunk is used to transmit 64-byte frames using protocol 5. If the propagation speed is 6衽ec/km, how many bits should the sequence numbers be?

一条3000公里长的T1骨干线路被用来传输 64字节的帧，两端使用了协议 5.如果传输速度为6μs/公里，则序列号应该有多少位？

答；为了有效运行，序列空间（实际上就是发送窗口大小）必须足够的大，以允许发送方在收到第一个确认应答之前可以不断发送。信号在线路上的传播时间为

6×3000= 18000 μs，即 18ms。

在 T1 速率，发送 64 字节的数据帧需花的时间： 64×8÷(1.536 106) = 0.33μs 。

所以，发送的第一帧从开始发送起， 18.33ms 后完全到达接收方。确认应答又花了很少的发送时间（忽略不计）和回程的 18ms。这样，加在一起的时间是 36.33ms，发送方应该有足够大的窗口，从而能够连续发送 36.33ms。

36. 33/0.33=110

也就是说，为充满线路管道，需要至少 110 帧，因此序列号为 7 位。

3-19In protocol 3, is it possible that the sender starts the timer when it is already running? If so, how might this occur? If not, why is it impossible?

19. 在协议 3 中，当发送方的定时器正在运行的时候，它还有可能启动定时器吗？如果可能的话，请问这种情况是如何发生的？如果不可能的话，请问为什么这是不可能的。

有可能发生。假设发送方传输率一个帧、很快返回了一个引起误解的确认。主循环将再次被执行,

一个帧将在定时器仍在运行的情况下被发送。

3-20 Imagine a sliding window protocol using so many bits for sequence numbers that wraparound never occurs. What relations must hold among the four window edges and the window size, which is constant and the same for both the sender and the receiver.

想象这样一个滑动窗口协议，它的序列号有非常多的位，所以序列号几乎永远不会回转。请问 4

个窗口边界和窗口大小之间必须满足什么样的关系？这里的窗口大小是固定不变的， 并且发送方

和接收方的窗口大小相同。

令发送方窗口为 (Sl , Su)，接收方窗口为 (Rl , Ru)，令窗口大小为 W。二者必须保持的关系是：

0 ≤ Su - Sl + 1 ≤ W1

Ru - Rl + 1 = W

Sl ≤ Rl ≤ Su + 1

3-21 If the procedure between in protocol 5 checked for the condition a b c instead of the condition a b < c, would that have any effect on the protocol's correctness or efficiency? Explain your answer.

如果协议5中的between过程检查的条件是a<=b<=c，而不是a<=b<c，则对于协议的正确性和效率

有影响吗？解释你的答案。

答：改变检查条件后，协议将变得不正确。假定使用 3 位序列号，考虑下列协议运行过程：

A 站刚发出 7 号帧；B 站接收到这个帧，并发出捎带应答 ack。A 站收到 ack，并发送 0~6 号帧。假定所有这些帧都在传输过程中丢失了。 B 站超时，重发它的当前帧，此时捎带的确认号是 7。考察 A 站在 r.rack=7 到达时的情况，关键变量是 ack_expected=0 r.rack=7，next_frame_to_send_=7 修改后的检查条件将被置成“ 真 ”，不会报告已发现的丢失帧错误，而误认为丢失了的帧已被确认。另一方面，如果采用原先的检查条件，就能够报告丢失帧的错误。

所以结论是：为保证协议的正确性，已接收的确认应答号应该小于下一个要发送的序列号。

3-22 In protocol 6, when a data frame arrives, a check is made to see if the sequence number differs from the one expected and no_nak is true. If both conditions hold, a NAK is sent. Otherwise, the auxiliary timer is started. Suppose that the else clause were omitted. Would this change affect the protocol's correctness?

在协议6中，当一个数据帧到达的时候，需要执行一个检查，看它的序列号是否与期望的序列号不同，而且no_nak为真。如果这两个条件都成立，则发送一个 NAK，否则的话，启动辅助定时器。假定else字句被省略掉，这种改变会影响协议的正确性吗？

答：可能导致死锁。假定有一组帧正确到达，并被接收。然后，接收方会向前移动窗口。

现在假定所有的确认帧都丢失了， 发送方最终会产生超时事件， 并且再次发送第一帧， 接收方将发送一个 NAK 。然后 NONAK 被置成伪。假定 NAK 也丢失了。那么从这个时候开始，发送方会不断发送已经被接收方接受了的帧。接收方只是忽略这些帧，但由于 NONAK 为伪，所以不会再发送 NAK ，从而产生死锁。如果设置辅助计数器（实现“ else”子句），超时后重发 NAK ，终究会使双方重新获得同步。

3-23 Suppose that the three-statement while loop near the end of protocol 6 were removed from the code. Would this affect the correctness of the protocol or just the performance? Explain your answer.

假设在协议6中接近尾部的内含三条语句的 while循环被去掉的话，这样会影响协议的正确性吗？

还是仅仅影响协议的性能？请解释答案。

答：删除这一段程序会影响协议的正确性， 导致死锁。因为这一段程序负责处理接收到的确认帧，

没有这一段程序，发送方会一直保持超时条件，从而使得协议的运行不能向前进展。

3-24 Suppose that the case for checksum errors were removed from the switch statement of protocol 6. How would this change affect the operation of the protocol?

24. 假设从协议 6 的 switch 语句中去掉检查校验和错误的那个 case 子句。请问这种变化将如何影响协议的操作？

这样将使得 NAK 的作用失效，于是我们将退回到超时。尽管效率会降低，正确性却不会受到影响。NAK 不是必不可少的。

3-25 In protocol 6 the code for frame_arrival has a section used for NAKs. This section is invoked if the

incoming frame is a NAK and another condition is met. Give a scenario where the presence of this other condition is essential.

在协议6中，针对frame_arrival的代码中有一部分被用于 NAK 。如果收到的帧是一个 NAK，并且

另一个条件也满足的话，则这部分代码会被调用到。 请给出一个场景，在此场景下这另一个条件

是非常关键的。

答：这里要求 r.rack+1<next_frame_to_send考虑下列操作细节：

A 站发送 0 号帧给 B 站。B 站收到此帧，并发送 ACK 帧，但 ACK 丢失了。A 站发生超时，

重发 0 号帧。但 B 站现在期待接收 1 号帧，应此发送 NAK，否定收到的 0 号帧。显然，现在

A 站最好不重发 0 号帧。由于条件 r.rack+1<next_frame_to_send不成立，所以用不着选择性重

传 0 号帧，可以继续向前推进传送 1 号帧。这个例子就说明了这段程序中的另一个条件，即

r.rack+1<next_frame_to_send也是重要的。

3-26 Imagine that you are writing the data link layer software for a line used to send data to you, but not from you. The other end uses HDLC, with a 3-bit sequence number and a window size of seven frames. You would like to buffer as many out-of-sequence frames as possible to enhance efficiency, but you are not allowed to modify the software on the sending side. Is it possible to have a receiver window greater than 1, and still guarantee that the protocol will never fail? If so, what is the largest window that can be safely used?

想象你正在编写一个数据链路层软件， 它被用在一条专门给你发送数据的线路上， 而不是让你往

外发送数据。另一端使用了 HDLC，3位序列号和一个可容纳 7帧的窗口。你希望将乱序的帧尽可

能多地缓存起来， 以提高效率，但是你又不允许修改发送方的软件。 是否有可能让接收方的窗口

大于1，并且仍然保证该协议不会失败呢？如果可能的话，能够安全使用的最大窗口是多少？

答：不可以。最大接收窗口的大小就是 1。现在假定该接收窗口值变为 2。开始时发送方发送 0 至

6 号帧，所有 7 个帧都被收到，并作了确认，但确认被丢失。现在接收方准备接收 7 号和 0 号

帧，当重发的 0 号帧到达接收方时，它将会被缓存保留，接收方确认 6 号帧。当 7 号帧到来的

时候，接收方将把 7 号帧和缓存的 0 号帧传递给主机，导致协议错误。因此，能够安全使用的

最大窗口值为 1。

3-28 In protocol 6, MAX_SEQ = 2n - 1. While this condition is obviously desirable to make efficient

use of header bits, we have not demonstrated that it is essential. Does the protocol work correctly for MAX_SEQ = 4, for example?

在协议6中，MAX_SEQ＝2n－1.这个条件显然是希望尽可能地利用头部的位， 但是我们无法证明

这个条件确实很关键。例如，协议在 MAX_SEQ＝4的时候也能够正确地工作吗？

答：不能，协议的运行将会失败。当 MaxSeq=4，序列号的模数 =4+1=5，窗口大小将等于：

NrBufs<=5/2=2.5，即得到，NrBufs=2。因此在该协议中，偶数序号使用缓冲区 1。这种映射意味着帧 4 和 0 将使用同一缓冲区。假定 0 至 3 号帧都正确收到了，并且都确认应答了，并且都确认应答了。如果随后的 4 号帧丢失，且下一个 0 号帧收到了，新的 0 号帧将被放到缓冲区 0 中，变量 arrived[0]被置成 “ 真 ”。这样，一个失序帧将被投递给主机。 事实上，采用选择性重传的滑动窗口协议需要 MaxSeq 是奇数才能正确的工作。然而其他的滑动窗口协议的实现并不具有这一性质。

3-29 Frames of 1000 bits are sent over a 1-Mbps channel using a geostationary satellite whose propagation time from the earth is 270 msec. Acknowledgements are always piggybacked onto data frames. The headers are very short. Three-bit sequence numbers are used. What is the maximum achievable channel utilization for (a) Stop-and-wait. (b) Protocol 5. (c) Protocol 6.

利用地球同步卫星在一个 1Mbps的信道上发送1000位的帧，该信道离开地球的传输延迟为 270ms。

确认信息总是被捎带在数据帧傻姑娘。头部非常短，并且使用 3位序列号。在下面的协议中，最大可获得的信道利用率是多少？ (a)停-等协议(b)协议5（c)协议6

答：对应三种协议的窗口大小值分别是 1、7 和 4。

使用卫星信道端到端的典型传输延迟是 270ms，以 1Mb/s 发送， 1000bit 长的帧的发送时间为 1ms。我们用 t=0 表示传输开始的时间，那么在 t=1ms 时，第一帧发送完毕；t=271ms时，第一帧完全到达接收方；t=272ms，对第一帧的确认帧发送完毕；t=542ms，带有确认的帧完全到达发送方。因此一个发送周期为 542ms。如果在 542ms 内可以发送 k 个帧，由于每一个帧的发送时间为 1ms，则信道利用率为 k/542，因此：

（a） k=1，最大信道利用率 =1/542=0.18%

（b） k=7，最大信道利用率=7/542=1.29%

（c） k=4，最大信道利用率=4/542=0.74%

3-30 Compute the fraction of the bandwidth that is wasted on overhead (headers and retransmissions) for protocol 6 on a heavily-loaded 50-kbps satellite channel with data frames consisting of 40 header and 3960 data bits. Assume that the signal propagation time from the earth to the satellite is 270 msec. ACK frames never occur. NAK frames are 40 bits. The error rate for data frames is 1 percent, and the error rate for NAK frames is negligible. The sequence numbers are 8 bits.

30. 在一个负载很重的 50kbps 的卫星信道上使用协议 6，数据帧包含 40 位的头和 3960 位的数据，请计算一下浪费在头部和重传的开销占多少比例。假设从地球到卫星的信号传输时间为 270ms。ACK 帧永远不会发生。NAK 帧为 40 位。数据帧的错误率为 1%，NAK 帧的错误率忽略不计。序列号为 8 位。

答：使用选择性重传滑动窗口协议，序列号长度是 8 位。窗口大小为 128。卫星信道端到端的传输延迟是 270ms。以 50kb/s 发送，4000bit（3960+40）长的数据帧的发送时间是 0.02*4000=80ms。我们用 t=0 表示传输开始时间，那么， t=80ms，第一帧发送完毕；t=270+80=350ms，第一帧完全到达接收方； t=350+80=430ms，对第一帧作捎带确认的反向数据帧可能发送完毕； t=430+270=700ms，带有确认的反向数据帧完全到达发送方。因此，周期为 700ms，发送 128 帧时间 80*128=10240ms，这意味着传输管道总是充满的。每个帧重传的概率为 0.01，对于 3960 个数据位，头开销为 40 位，平均重传的位数为 4000*0.01=40位，传送 NAK 的平均位数为 40*1/100=0.40 位，所以每 3960 个数据位的总开销为 80.4 位。

因此，开销所占的带宽比例等于 80.4/(3960+80.4)=1.99%

3-32 A 100-km-long cable runs at the T1 data rate. The propagation speed in the cable is 2/3 the speed of light in vacuum. How many bits fit in the cable?

32. 一条 100 公里长的电缆运行在 T1 数据速率上。电缆的传输速度是真空中光速的 2/3。请问电缆中可以容纳多少位？

答：在该电缆中的传播速度是每秒钟 200 000km，即每毫秒 200km，因此 100km 的电缆将会在 0.5ms 内填满。T1 速率 125μs 传送一个 193 位的帧，0.5ms 可以传送 4 个 T1 帧，即 193*4=772bit。

第 4 章 介质访问子层

4-1 For this problem, use a formula from this chapter, but first state the formula. Frames arrive randomly at a 100-Mbps channel for transmission. If the channel is busy when a frame arrives, it waits its turn in a queue. Frame length is exponentially distributed with a mean of 10,000 bits/frame. For each of the following frame arrival rates, give the delay experienced by the average frame, including both queueing time and transmission time.
(a) 90 frames/sec. (b) 900 frames/sec. (c) 9000 frames/sec.

(time delay, T // a channel of capacity C bps// with an arrival rate of frames/sec)

1. 在这个练习中，请使用本章中的一种规则（方案），但是在计算之前请先声明这种规则。在一个 100Mbps 的信道上，待传输的帧随机地到达。如果当一帧到达的时候该信道正忙，那么它必须排队等待。帧的长度呈指数分布，均值为每帧 10 000 位。对于下列每一种帧到达率，请给出平均一帧的延迟，包括排队时间和传输时间。

(a) 90 帧/秒

(b) 900 帧/秒

(c) 9000 帧/秒

这个公式是 4.1.1 段落给出的 Markov 排队问题的标准公式。也就是，$T = 1/(\mu C - \lambda)$。这里 C =108 、

$\mu = 10^{-4}$，$T = 1/(10000 - lambda)$ sec。对于这三种到达速率，我们得出的是(a) 0.1 msec,(b) 0.11 msec,

(c) 1 msec.对于 c 的情况，我们操作一个带来 10 倍延迟的排队系统 $\rho = \lambda/\mu C = 0.9$

4-2 A group of N stations share a 56-kbps pure ALOHA channel. Each station outputs a 1000-bit frame on an average of once every 100 sec, even if the previous one has not yet been sent (e.g., the stations can buffer outgoing frames). What is the maximum value of N?

N个站共享一个56kbps的纯ALOHA 信道。每个站平均每 100秒输出一个1000位的帧，及时前面的

帧还没有被送出，它也这样进行（比如这些站可以将送出的帧缓存起来）。请问 N的最大值是多

少？

答：对于纯的 ALOHA，可用的带宽是 0.184 *56 Kb/s =10.304 Kb/ s 每个站需要的带宽为

1000/100=10b/s而 N=10304/10≈1030 所以, 最多可以有 1030 个站, 即 N 的最大值为 1030。

4-3 Consider the delay of pure ALOHA versus slotted ALOHA at low load. Which one is less? Explain your answer.

3. 考虑在低载荷情况下纯的 ALOHA 和分槽的 ALOHA 的延迟。哪一个延迟更小？请说明你的理由。

答：对于纯的 ALOHA，发送可以立即开始。对于分隙的 ALOHA，它必须等待下一个时隙。这样，平均会引入半个时隙的延迟。因此，纯 ALOHA 的延迟比较小。

4-4 Ten thousand airline reservation stations are competing for the use of a single slotted ALOHA channel. The average station makes 18 requests/hour. A slot is 125 μsec. What is the approximate total channel load?

4. 10 000 个航线预定站正在竞争使用一个分槽的 ALOHA 信道。这些站平均每小时发出 18 次请求。时槽为 125μs。总的信道载荷大约是多少？

每个终端每 200（=3600/18）秒做一次请求，总共有 10 000 个终端，因此，总的负载是 200 秒做 10000 次请求。平均每秒钟 50 次请求。每秒钟 8000 个时隙，所以平均每个时隙的发送次数为 50/8000=1/160。

4-5 A large population of ALOHA users manages to generate 50 requests/sec, including both originals and retransmissions. Time is slotted in units of 40 msec. (a) What is the chance of success on the first attempt? (b) What is the probability of exactly k collisions and then a success? (c) What is the expected number of transmission attempts needed?

一大群ALOHA 用户每秒钟产生 50个请求，包括原始的请求和重传的请求。时槽单位为 40ms。(a) 首次发送成功的几率是多少？ (b)恰好k次冲突之后成功的概率是多少？(c)所需传送次数的期望值是多少？

答：（a）在任一帧时间内生成 k 帧的概率服从泊松分布

$$\Pr[k] = \frac{G^k e^{-G}}{k!}$$

生成 0 帧的概率为 e-G

对于纯的 ALOHA，发送一帧的冲突危险区为两个帧时，在两帧内无其他帧发送的概率是 e-G×e-G=e-2G

对于分隙的 ALOHA，由于冲突危险区减少为原来的一半，任一帧时内无其他帧发送的概率是 e-G 。

现在时隙长度为 40ms，即每秒 25 个时隙，产生 50 次请求，所以每个时隙产生两个请求， G=2。

因此，首次尝试的成功率是： e-2 = 1/ e2

（b） $(1-e^{-G})^k e^{-G} = (1-e^{2})^k e^{-2} = 0.135\times(1-0.135)^k = 0.135\times 0.865^k$

（c）尝试 k 次才能发送成功的概率（即前 k-1 次冲突，第 k 次才成功）为：

$$p_k = e^{-G}(1-e^{-G})^{k-1}$$

那么每帧传送次数的数学期望为

$$E = \sum_{k=1}^{\infty} kp_k = \sum_{k=1}^{\infty} ke^{-G}(1-e^{-G})^{k-1} = e^{G} = e^{2} = 7.4$$

4-6　Measurements of a slotted ALOHA channel with an infinite number of users show that 10 percent of the slots are idle. (a) What is the channel load, G? (b) What is the throughput? (c) Is the channel underloaded or overloaded?

对一个无限用户的分槽 ALOHA 信道的测试表明，10%的时槽是空闲的。(a)信道载荷G是多少?(b)

吞吐量是多少？(c)该信道是载荷不足，还是过载了？

答：（a）从泊松定律得到 p0＝e -G ，因此 G＝ -lnp0= -ln0.1＝2.3

（b） S＝G e -G ，　G　　＝2.3，e -G=0.1

S=2.3 ⊗0.1=0.23

（c）因为每当 G>1 时，信道总是过载的，因此在这里信道是过载的。

4-8　How long does a station, s, have to wait in the worst case before it can start transmitting its frame over a LAN that uses (a) the basic bit-map protocol? (b) Mok and Ward's protocol with permuting virtual station numbers?

如果一个LAN 使用了下列协议，请问在最差情况下，一个站 s在开始传送帧之前必须要等到多长

时间？（a）基本的位图协议（b）改变虚拟站序列号的 Mok-Ward协议

(a) 最坏的情况是：所有站都想发送，其中 s 是编号最小的站点。等待时间是 N 位争用周期+(N-1)

×d 位帧传输，一共是 N+(N-1) d 位时间；

(b) (b) 最坏的情况是：所有站都有帧要传输，s 具有其中最小的虚拟站编号。因此，s 在其它 N-1

个站各发送了一个帧之后将获得传输机会，以及每个大小为 log2 N 的 N 个争用周期。

等待时间是(N+1)×d+N×log2 Nbits.？？？

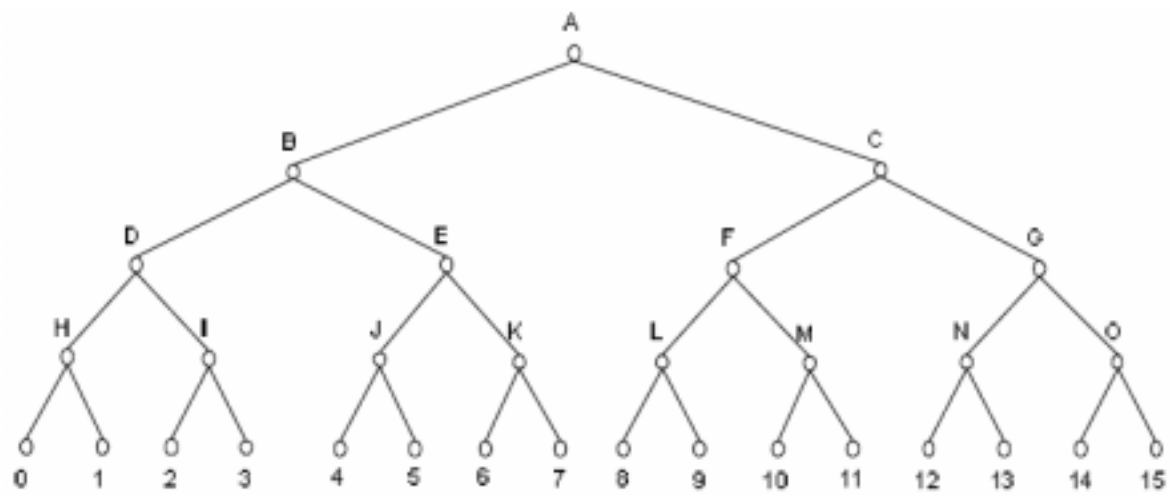
4-10 Sixteen stations, numbered 1 through 16, are contending for the use of a shared channel by using the adaptive tree walk protocol. If all the stations whose addresses are prime numbers suddenly become ready at once, how many bit slots are needed to resolve the contention?

16个站的编号从1到16，它们正在竞争使用一个使用了可适应树径协议的共享信道。如果地址编

号为素数的所有站突然间全部要发送帧，请问需要多少位时槽才能解决竞争？

答：在自适应树遍历协议中，可以把站点组织成二叉树（见图）的形式。在一次成功的传输之后，

在第一个竞争时隙中， 全部站都可以试图获得信道，如果仅其中之一需用信道， 则发送冲突，则第二时隙内只有那些位于节点 B 以下的站（0 到 7）可以参加竞争。如其中之一获得信道，本帧后的时隙留给站点 C 以下的站；如果 B 点下面有两个或更多的站希望发送，在第二时隙内会发生冲突，于是第三时隙内由 D 节点以下各站来竞争信道。



本题中，站 2、3、5、7、11 和 13 要发送，需要 13 个时隙，每个时隙内参加竞争的站的列表如下：

第一时隙：2、3、5、7、11、13

第二时隙：2、3、5、7

第三时隙：2、3

第四时隙：空闲

第五时隙：2、3

第六时隙：2

第七时隙：3

第八时隙：5、7

第九时隙：5

第十时隙：7

第十一时隙：11、13

第十二时隙：11

第十三时隙：13

4-14 Six stations, A through F, communicate using the MACA protocol. Is it possible that two transmissions take place simultaneously? Explain your answer.

14. 6 个站的编号从 A 到 F，它们使用 MACA 协议进行通信。请问有可能同时发生两个传输操作吗？说明你的理由。

是的，想像它们排成一条直线，每个站只能到达其最近的邻居。当 E 向 F 发送时 A 也能向 B 发送。

4-21 Consider building a CSMA/CD network running at 1 Gbps over a 1-km cable with no repeaters. The signal speed in the cable is 200,000 km/sec. What is the minimum frame size?

考虑在一条1km长的电缆（无中继器）上建立一个1Gbps速率的CSMA/CD网络。信号在电缆中的速度为200000km/s。请问最小的帧长度为多少？

答：对于 1km 电缆，单程传播时间为 1/200000 =5×10-6 s，即 5μs，来回路程传播时间为 2t =10μs。为了能够按照 CSMA/CD 工作，最小帧的发射时间不能小于 10μs。以 1Gb/s 速率工作，10μs 可以发送的比特数等于：

$$\frac{10\times10^{-6}}{1\times10^{-9}}=10000$$

因此，最小帧是 10 000 bit 或 1250 字节长。

4-22 An IP packet to be transmitted by Ethernet is 60 bytes long, including all its headers. If LLC is not in use, is padding needed in the Ethernet frame, and if so, how many bytes?

一个通过以太网传送到 IP分组有60字节长，其中包括所有的头部。如果没有使用 LLC的话，则以太网帧中需要填补字节码？如果需要的话，请问需要填补多少字节？

最小的以太帧是 64bytes，包括了以太帧头部的二者地址、类型 /长度域、校验和。因为头部域占用 18 bytes 报文是 60 bytes，总的帧长度是 78 bytes，已经超过了 64-byte 的最小限制。因此，不需要填补。

4-23 Ethernet frames must be at least 64 bytes long to ensure that the transmitter is still going in the event of a collision at the far end of the cable. Fast Ethernet has the same 64-byte minimum frame size but can get the bits out ten times faster. How is it possible to maintain the same minimum frame size?

23. 以太网帧必须至少 64 字节长，这样做的理由是，当电缆的另一端发生冲突的时候，传送方仍然还在发送过程中。快速以太网也有同样的 64 字节最小帧长度限制，但是，它可以以快 10 倍的速度发送数据。请问它如何有可能维持同样的最小帧长度限制？

快速以太网的最大线路长度是以太网的 1/10 。

4-24 Some books quote the maximum size of an Ethernet frame as 1518 bytes instead of 1500 bytes. Are they wrong? Explain your answer.

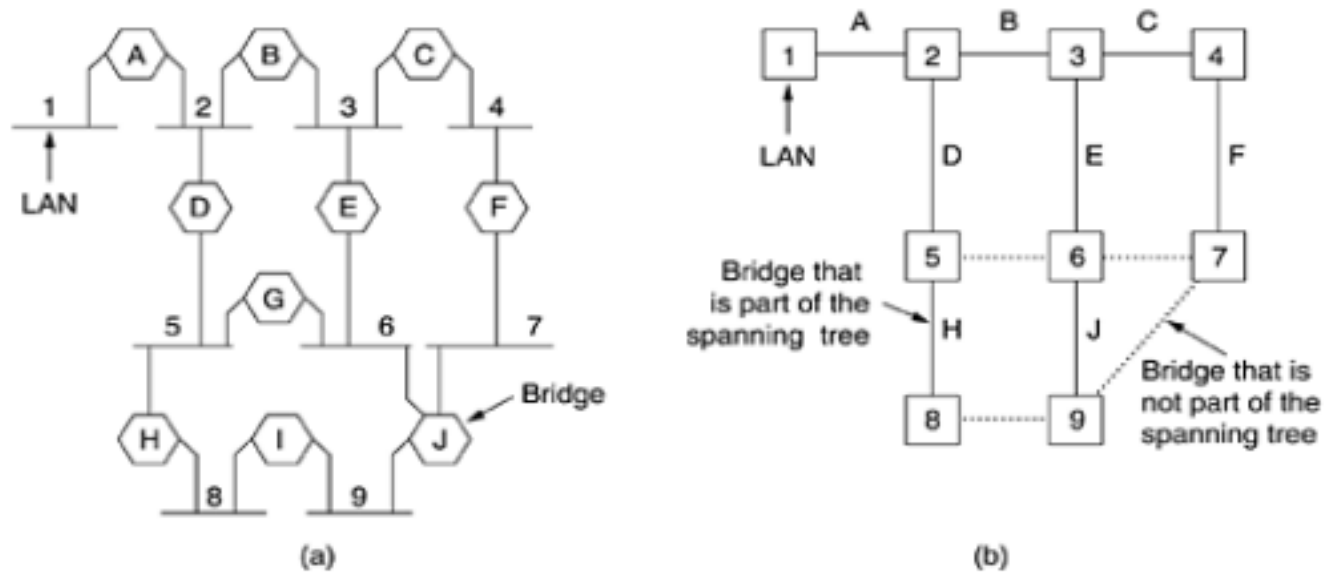
24. 有些书将以太网帧的最大长度说成是 1518 字节，而不是 1500 字节。这些书错了吗？请说明你的理由。

有效载荷是 1500 bytes,但将目的地址、源地址、类型 /长度和校验和域都计算进去的话，总和就

是 1518.

4-37 Consider the interconnected LANs showns in Fig. 4-44. Assume that hosts a and b are on LAN 1, c is on LAN 2, and d is on LAN 8. Initially, hash tables in all bridges are empty and the spanning tree shown in Fig 4-44(b) is used. Show how the hash tables of different bridges change after each of the following events happen in sequence, first (a) then (b) and so on. (a) a sends to d. (b) c sends to a. (c) d sends to c. (d) d moves to LAN 6. (e) d sends to a.

Figure 4-44. (a) Interconnected LANs. (b) A spanning tree covering the LANs. The dotted lines are not part of the spanning tree.



考虑图4.44中相互连接的LAN 。假设主机 a和b在LAN1 上，主机c在LAN2 上，主机 d在LAN8 上。

刚开始的时候，所有的网桥内部的散列表都是空的，并且使用了图 4.44b所示的生成树。在下面给出的每个事件依次发生以后，不同网桥的散列表将如何变化。（a)a向d发送帧(b)c向a发送帧(c)d向c发送帧(d)d移动到LAN6 上(e)d向a发送帧

第一个帧会被每个网桥转发。这次传输后，每个网桥的散列表会得到一个带有适当端口的目的地为 a 的项目。例如 D 的散列表会有一个向在 LAN2 上的目的地为 a 转发帧的项目。第二个信息会

被网桥 B, D 和 A 看到。这些网桥会在它们的散列表中添加一个目的地为 c 的新项目。例如，网桥 D 的散列表现在会有另一个向在 LAN2 上的目的地为 c 转发帧的项目。第三个信息会被网桥 H, D, A 和 B 看到。这些网桥会在它们的散列表中添加一个目的地为 d 的新项目。第五条信息会被网桥 E, C, B, D和 A 看到。网桥 E 和 C 会在他们的散列表添加一个目的地为 d 的新项目，与此同时，网桥 D, B 和 A 将会更新它们对应目的地 d 的散列表项目。

4-38 One consequence of using a spanning tree to forward frames in an extended LAN is that some bridges may not participate at all in forwarding frames. Identify three such bridges in Fig. 4-44. Is there any reason for keeping these bridges, even though they are not used for forwarding?

在一个扩展的LAN 中使用生成树来转发帧的一个结果是，有的网桥可能根本不参与帧的转发过程。请在图4.44中标出三个这样的网桥。既然这些网桥没有被用于转发帧，那么是否有理由要保留这些网桥呢？

网桥 G, I 和 J 没有被用来转发任何帧。在一个扩展的 LAN 中具有回路的主要原因是增加可靠性。如果当前生成树中的任何网桥出了故障， （动态）生成树算法重构一个新的生成树，其中可能包括一个或更多不属于先前生成树部分的网桥。

4-42 Briefly describe the difference between store-and-forward and cut-through switches.

42. 简略地描述一下存储-转发型交换机和直通型交换机之间的区别。

存储-转发型交换机完整存储输入的每个帧，然后检查并转发。直通型交换机在输入帧没有全部到达之前就开始转发。一得到目的地址，转发就开始了。

4-43 Store-and-forward switches have an advantage over cut-through switches with respect to damaged frames. Explain what it is.

43. 从损坏帧的角度而言，存储-转发型交换机比起直通型交换机更有优势。请说明这种优势是什么。

Store-and-forward switches store entire frames before forwarding them. After a frame comes in, the checksum can be verified. If the frame is damaged, it is discarded immediately. With cut=through, damaged frames cannot be discarded by the switch because by the time the error is detected, the frame is already gone. Trying to deal with the problem is like locking the barn door after the horse has escaped.

存储-转发型交换机在转发帧之前存储整个帧。当一个帧到达时，校验和将被验证。如果帧已被损坏，它将被立即丢弃。

在直通型交换机，损坏的帧不能被交换机丢弃。 因为当错误被检测到时，帧已经过去了。想要处

理这个问题就像是在马已经逃逸之后再锁上牲口棚。

第 5 章 网络层

5-1 Give two example computer applications for which connection-oriented service is appropriate. Now give two examples for which connectionless service is best.

1. 请列举出两个适合于使用面向连接服务的计算机应用，再列举出两个最好使用无连接服务的计算机应用例子。

答：文件传送、远程登录和视频点播需要面向连接的服务。另一方面，信用卡验证和其他销售点终端、电子资金转移，及许多形式远程数据库访问生来具有无连接性质， 在一个方向上传送查询，在另一个方向上返回应答。

5-2 Are there any circumstances when connection-oriented service will (or at least should) deliver packets out of order? Explain.

2. 请问有没有可能发生这样的情形：面向连接的服务也会(或者至少应该)以乱序的方式递交分组？请解释原因。

答：有。中断信号应该跳过在它前面的数据， 进行不遵从顺序的投递。典型的例子是当一个终端用户键入退出（或 kill）健时。由退出信号产生的分组应该立即发送，并且应该跳过当前队列中排在前面等待程序处理的任何数据（即已经键入但尚未被程序读取的数据）。

5-3 Datagram subnets route each packet as a separate unit, independent of all others. Virtual-circuit subnets do not have to do this, since each data packet follows a predetermined route. Does this observation mean that virtual-circuit subnets do not need the capability to route isolated packets from an arbitrary source to an arbitrary destination? Explain your answer.

3. 数据报子网将每个分组当作独立的单位进行路由，所以每个分组的路由过程独立于其他所有的分组。虚电路子网不必采用这种方式，因为每个数据分组都沿着一条预先确定的路径向前传送。这是否意味着虚电路子网并不需要具备将独立的分组从任意的源端路由到任意目标端的能力呢？请解释你的答案。

答：不对。为了从任意源到任意目的地，为连接建立的分组选择路由，虚电路网络肯定需要这一能力。

5-5 Consider the following design problem concerning implementation of virtual-circuit service. If virtual circuits are used internal to the subnet, each data packet must have a 3-byte header and each router must tie up 8 bytes of storage for circuit identification. If datagrams are used internally, 15-byte headers are needed but no router table space is required. Transmission capacity costs 1 cent per 106 bytes, per hop. Very fast router memory can be purchased for 1 cent per byte and is depreciated over two years, assuming a 40-hour business week. The statistically average session runs for 1000 sec, in which time 200 packets are transmitted. The mean packet requires four hops. Which implementation is cheaper, and by how much?

5. 请考虑以下涉及到实现虚电路服务的设计问题。如果在子网内部使用虚电路，那么，每个数据分组必须有一个 3 字节的头，每台路由器必须提供 8 字节的存储空间用于电路标识。如果子网内部使用数据报，那么，每个数据分组需要一个 15 字节的头，但是不要求路由器的表空间。假设每一跳每 10^6 字节的传输开销为 1 美分。快速路由器内存的价格是每字节 1 美分，2 年以后就贬值了，这里假设每周的工作时间为 40 小时。平均每个会话的持续时间为 1000 秒，在这段时间中平均传输 200 个分组。平均每个分组要求 4 跳。请问哪种实现方法更加便宜，便宜多少？

答：虚电路实现需要在 1000 秒内固定分配 5*8=40 字节的存储器。数据报实现需要比虚电路实现多传送的头信息的容量等于 (15-3) ×4×200=9600 字节-跳段。现在的问题就变成了 40000 字节-秒的存储器对比 9600 字节-跳段的电路容量。如果存储器的使用期为两年，即 3600×8×5×52×2= 1.7 ×107 秒，一个字节-秒的代价为 1/(1.5 107) = 6.7 10-8 分，那么 40000 字节-秒的代价为 2.7 毫分。另一方面，1 个字节-跳段代价是 10-6 分，9600 个字节-跳段的代价为 10-6 × 9600=9.6 ×10-3 分，即 9.6 毫分，即在这 1000 秒内的时间内便宜大约 6.9 毫分。

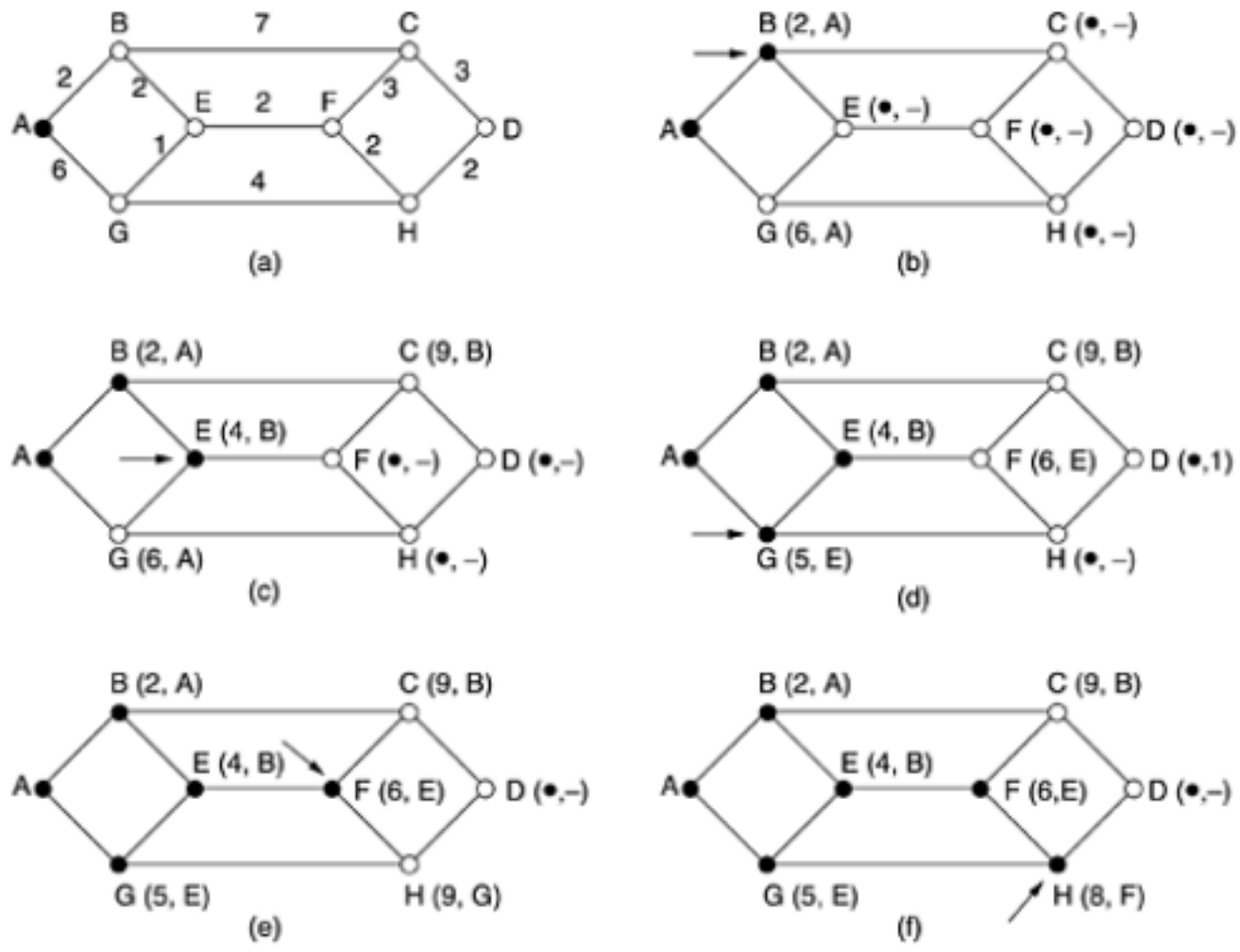
5-6 Assuming that all routers and hosts are working properly and that all software in both is free of all errors, is there any chance, however small, that a packet will be delivered to the wrong destination?

6. 假设所有的路由器和主机都正常工作，并且它们的软件也都没有错误，请问一个分组被递交到错误目的地的可能性有没有（无论可能性有多小）？

答：有可能。大的突发噪声可能破坏分组。使用 k 位的检验和，差错仍然有 2 k的概率被漏检。如果分组的目的地段或虚电路号码被改变，分组将会被投递到错误的目的地，并可能被接收为正确的分组。换句话说，偶然的突发噪声可能把送往一个目的地的完全合法的分组改变成送往另一个目的地的也是完全合法的分组。

5-7 Consider the network of Fig. 5-7, but ignore the weights on the lines. Suppose that it uses flooding as the routing algorithm. If a packet sent by A to D has a maximum hop count of 3, list all the routes it will take. Also tell how many hops worth of bandwidth it consumes.

Figure 5-7. The first five steps used in computing the shortest path from *A* to *D*. The arrows indicate the working node.

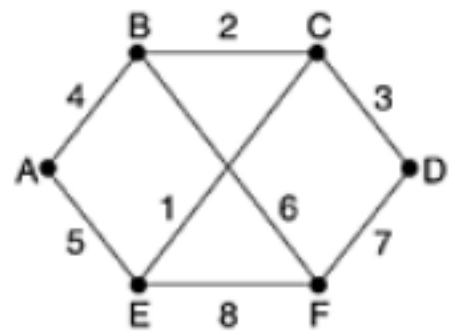


7. 请考虑图 5.7 中的网络，但是忽略线路上的权值。假设它使用扩散法作为路由算法。如果一个从 A 发向 D 的分组的最大跳计数值为 3，请列出它将要走的所有路径。同时也说明它需要消耗多少跳带宽。

路径将依次为下面的路由 : ABCD, ABCF, ABEF, ABEG, AGHD, AGHF, AGEB, 和 AGEF. 用到的跳数是 24。

5-9 Consider the subnet of Fig. 5-13(a). Distance vector routing is used, and the following vectors have just come in to router C: from B: (5, 0, 8, 12, 6, 2); from D: (16, 12, 6, 0, 9, 10); and from E: (7, 6, 3, 9, 0, 4). The measured delays to B, D, and E, are 6, 3, and 5, respectively. What is C's new routing table? Give both the outgoing line to use and the expected delay.

Figure 5-13. (a) A subnet.



9. 考虑图 5.13(a)中的子网。该子网使用了距离矢量路由算法，下面的矢量刚刚到达路由器 C：来自 B 的矢量为(5,0,8,12,6,2)；来自 D 的矢量为(16,12,6,0,9,10)；来自 E 的矢量为(7,6,3,9,0,4)。经测量，到 B、D 和 E 的延迟分别为 6、3 和 5。请问 C 的新路由表将会怎么样？请给出将使用的输出线路以及期望的延迟。

答：通过 B 给出（11, 6, 14, 18, 12, 8）

通过 D 给出（19, 15, 9, 3, 12, 13）

通过 E 给出（12, 11, 8, 14, 5, 9）

取到达每一目的地的最小值（C 除外）得到：（11, 6, 0, 3, 5, 8）

输出线路是：（B, B, -, D, E, B）

5-10 If delays are recorded as 8-bit numbers in a 50-router network, and delay vectors are exchanged twice a second, how much bandwidth per (full-duplex) line is chewed up by the distributed routing algorithm? Assume that each router has three lines to other routers.

10. 假设在一个 50 台路由器的网络中，用 8 位数值记录延迟信息，并且每秒钟交换延迟矢量两次，请问，分布式路由算法需要在每条(全双工)线路上消耗多少带宽？假设每台路由器有三条线路连接到其他的路由器。

答：路由表的长度等于 8*50=400bit。该表每秒钟在每条线路上发送 2 次，因此 400*2=800b/s

即在每条线路的每个方向上消耗的带宽都是 800 bps。

5-12 For hierarchical routing with 4800 routers, what region and cluster sizes should be chosen to minimize the size of the routing table for a three-layer hierarchy? A good starting place is the hypothesis that a solution with k clusters of k regions of k routers is close to optimal, which means that k is about the cube root of 4800 (around 16). Use trial and error to check out combinations where all three parameters are in the general vicinity of 16.

12. 对于 4800 台路由器的三层次分级路由，请问应该选择多大的区域和群才可以将路由表的尺寸降低到最小？一个好的起点是，假设在方案中 k 台路由器构成一个区域，k 个区域构成一个群，并且总共有 k 个群，这样的方案接近于最优的方案。这意味着 k 大约是 4800 的立方根(约等于 16)。请试验所有这三个参数在 16 附近的各种组合。

所谓分级路由，就是将路由器按区（ REGION）进行划分，每个路由器只须知道在自己的区内如何为分组选择路由到达目的地的细节， 而不用知道其他区的内部结构。 对于大的网络，也许两级

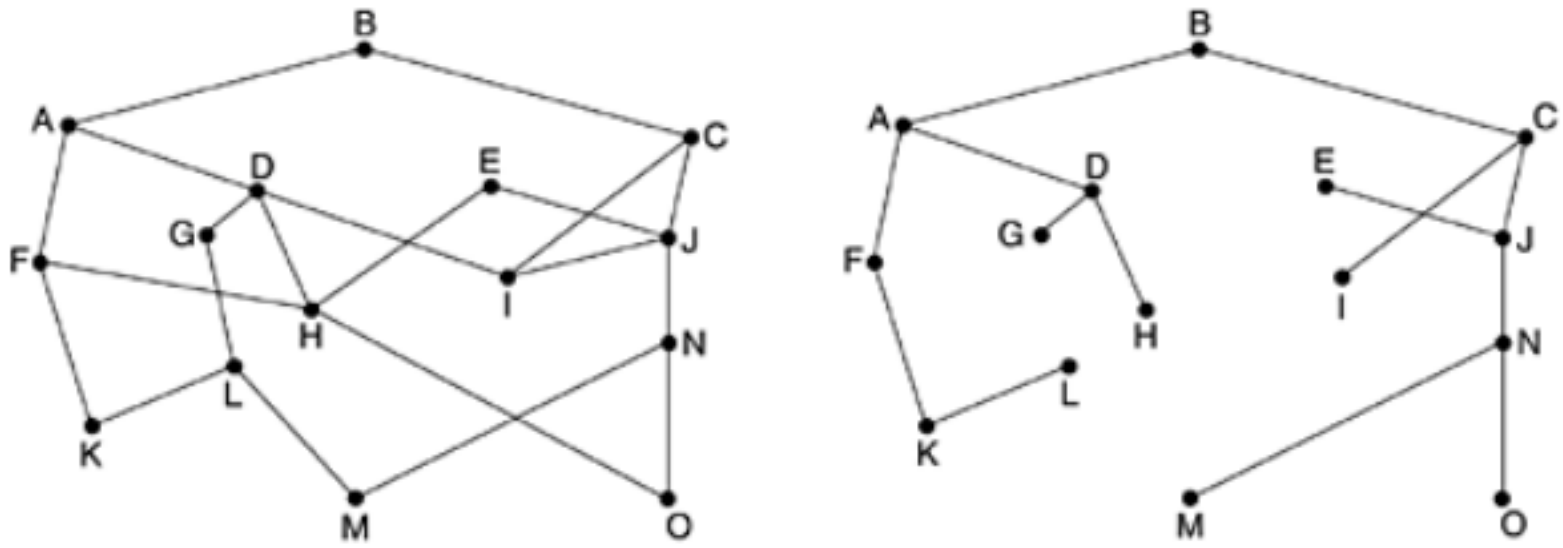
结构是不够的，还可以把区组合成簇（CLUSTER），把簇再组合成域（ZONE），??对于等级式路由，在路由表中对应所有的本地路由器都有一个登录项，所有其他的区（本簇内）、簇（本域内）和域都缩减为单个路由器，因此减少了路由表的尺寸。

在本题中，4800=15*16*20。当选择 15 个簇、16 个区，每个区 20 个路由器时（或等效形式，例如 20 个簇、16 个区，每个区 15 个路由器），路由表尺寸最小，此时的路由表尺寸为

15+16+20=51。

5-14 Looking at the subnet of Fig. 5-6, how many packets are generated by a broadcast from B, using (a) reverse path forwarding? (b) the sink tree?

Figure 5-6. (a) A subnet. (b) A sink tree for router *B*.



14. 参照图 5.6 中的子网，请问：若使用以下方法，从 B 发出的广播将生成多少个分组？

（a）逆向路径转发

（b）汇集树

答：在一个子网中，从所有的源到一个指定的目的地的最佳路由的集合形成一棵以该目的地为根的树。这样的树就称作汇集树。汇集树不必是唯一的，其他具有相同通路长度的树可能存在。所有路由选择算法的目标都是要为所有的路由器寻找和使用汇集树。 在广播形式的应用中，源主机需要向所有其他的主机发送报文。 在称为反向通路转发的广播路由选择中， 当广播分组到达路由器时，路由器对此分组进行检查， 查看该分组是否来自于通常用于发送分组到广播源的线路， 如果是，则此广播分组本身非常有可能是从源路由器来的第一个拷贝。

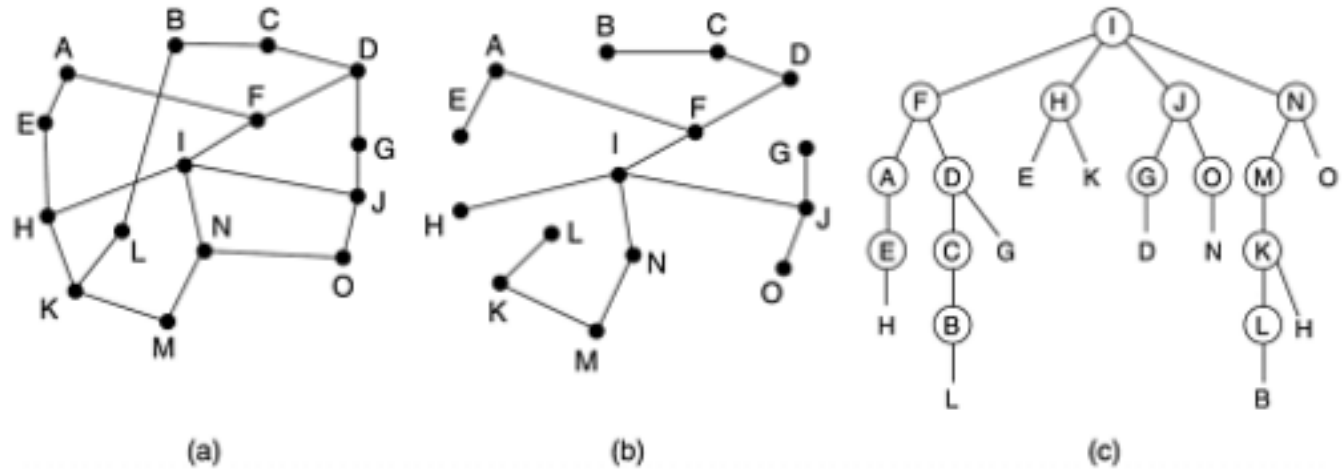
在这种情况下，路由器将此分组复制转发到进入线路以外的所有线路。 然而，如果广播分组到来的线路不是到达源端的线路，那么分组就被当作副本而扔掉。

（1）反向通路转发算法，算法进行到 5 个跳段后结束，总共产生 28 个分组。

（2）使用汇集树算法，需要 4 个跳段，总共产生 14 个分组。

5-15 Consider the network of Fig. 5-16(a). Imagine that one new line is added, between F and G, but the sink tree of Fig. 5-16(b) remains unchanged. What changes occur to Fig. 5-16(c)?

Figure 5-16. Reverse path forwarding. (a) A subnet. (b) A sink tree. (c) The tree built by reverse path forwarding.



15. 考虑图 5.16(a)中的网络。想象在 F 和 G 之间加入一条新的线路，但是，图 5.16(b)中的汇集树仍然不变。请问对于图 5.16(c)有什么变化？

节点 F 目前有两个子节点 A 和 D.它现在获得了第三个——G，但并未构成环路，因为沿着 IFG 的报文不在汇集树上。节点 G 除 D 之外获得第二个子节点，标记为 F，它也因没有加入汇集树而没有构成环路。

5-21 As a possible congestion control mechanism in a subnet using virtual circuits internally, a router could refrain from acknowledging a received packet until (1) it knows its last transmission along the virtual circuit was received successfully and (2) it has a free buffer. For simplicity, assume that the routers use a stop-and-wait protocol and that each virtual circuit has one buffer dedicated to it for each direction of traffic. If it takes T sec to transmit a packet (data or acknowledgement) and there are n routers on the path, what is the rate at which packets are delivered to the destination host? Assume that transmission errors are rare and that the host-router connection is infinitely fast.

21. 在内部采用虚电路的子网中，可能采用这样一种拥塞控制机制：路由器直到“(1)知道沿着虚电路的最后一次传输已经成功地到达了，并且(2)它有一个空闲缓冲区”的时候，才对一个接收到的分组进行确认。为了简单起见，假定路由器使用了停-等协议，并且每条虚电路的每个方向都有一个专用的缓冲区。如果传输一个分组(数据或者确认)需要 T 秒，在路径上有 n 台路由器，那么分组被递交给目标主机的速率是多少？假设几乎没有传输错误，并且从主机到路由器之间连接的速度为无限快。

答：对时间以 T 秒为单位分时隙。在时隙中，源路由器发送第一个分组。在时隙 2 的开始，第 2 个路由器收到了分组，但不能应答。在时隙 3 的开始，第 3 个路由器收到了分组，但也不能

应答。这样，此后所有的路由器都不会应答。仅当目的地主机从目的地路由器取得分组时才会发送第 1 个应答。现在确认应答开始往回传播。在源路由器可以发送第 2 个分组之前，需要两次穿行该子网，需要花费的时间等于 2(n-1)T 秒/分组，显然，这种协议的效率是很低的。

5-22 A datagram subnet allows routers to drop packets whenever they need to. The probability of a router discarding a packet is p. Consider the case of a source host connected to the source router, which is connected to the destination router, and then to the destination host. If either of the routers discards a packet, the source host eventually times out and tries again. If both host-router and router-router lines are counted as hops, what is the mean number of

(a) hops a packet makes per transmission? (b) transmissions a packet makes? (c) hops required per received packet?

22. 一个数据报子网允许路由器在必要的时候丢弃分组。一台路由器丢弃一个分组的概率为 p。请考虑这样的情形：源主机连接到源路由器，源路由器连接到目标路由器，然后目标路由器连接到目标主机。如果任一台路由器丢掉了一个分组，则源主机最终会超时，然后再重试发送。如果主机至路由器以及路由器至路由器之间的线路都计为一跳，那么：

(a) 一个分组每次传输中的平均跳数是多少？

(b) 一个分组的平均传输次数是多少？

(c) 每个接收到的分组平均要求多少跳？

答：（1）由源主机发送的每个分组可能行走 1 个跳段、2 个跳段或 3 个跳段。走 1 个跳段的概率为 p，走 2 个跳段的概率为（1- p）p，走 3 个跳段的概率为（1- p）2 p。那么，一个分组平均通路长度的期望值为：

$$L = 1 \cdot p + 2 \cdot (1-p)p + 3 \cdot (1-p)^2 = p^2 - 3p + 3$$

即每次发送一个分组的平均跳段数是 p2-3 p +3。

（2）一次发送成功（走完整个通路）的概率为(1- p)2，令 a = (1- p)2，两次发射成功的概率等于 (1- a) a，三次发射成功的概率等于 (1- a)2 a ，…，因此一个分组平均发送次数为：

$$T = \sum_{n=1}^{\infty} n\alpha(1-\alpha)^{n-1} = \frac{1}{\alpha} = \frac{1}{(1-p)^2}$$

即一个分组平均要发送 1/(1- p)2次。

（3）最后，每一个接收到的分组行走的平均跳段数等于

$$H = L \times T = (p^2 - 3p + 3)/(1-p)^2$$

5-23 Describe two major differences between the warning bit method and the RED method.

23. 描述一下警告位法和 RED 方法的两个主要区别。

第一，警告位方法通过设置一个位明确地发送一个拥塞通知给来源，而 RED 方法通过简单地丢弃它的一个报文来隐含地通知源；

第二，警告位方法只在没有缓冲空间时才丢弃报文，而 RED 方法在所有缓冲区耗尽之前就丢弃报文。

5-24 Give an argument why the leaky bucket algorithm should allow just one packet per tick, independent of how large the packet is.

24. 请说明为什么漏桶算法应该每个时钟滴答只允许一个分组，而不管分组的大小？

答：通常计算机能够以很高的速率产生数据，网络也可以用同样的速率运行。然而，路由器却只能在短时间内以同样高的速率处理数据。对于排在队列中的一个分组，不管它有多大，路由器必须做大约相同分量的工作。显然，处理 10 个 100 字节长的分组所作的工作比处理 1 个 1000 字节长的分组要做的工作多得多。

5-25 The byte-counting variant of the leaky bucket algorithm is used in a particular system. The rule is that one 1024-byte packet, or two 512-byte packets, etc., may be sent on each tick. Give a serious restriction of this system that was not mentioned in the text.

25. 在一个特殊的系统中用到了漏桶算法的字节计数变种算法。其规则是，每个时钟滴答可以发送一个 1024 字节的分组，或者两个 512 字节的分组，以此类推。请就这个系统说出一个本章正文中没有提到的严重限制。

答：不可以发送任何大于 1024 字节的分组。

5-27 A computer on a 6-Mbps network is regulated by a token bucket. The token bucket is filled at a rate of 1 Mbps. It is initially filled to capacity with 8 megabits. How long can the computer transmit at the full 6 Mbps?

27. 在一个 6Mbps 的网络上，有一台主机通过一个令牌桶进行流量调整。令牌桶的填充速率为 1Mbps。初始时候它被填充到 8Mb 的容量。请问该计算机以 6Mbps 的全速率可以传输多长时间？

答：本题乍看起来，似乎以 6Mb/s 速率发送用 4/3 秒的时间可以发送完桶内 8Mb 的数据，使漏桶变空。然而，这样回答是错误的，因为在这期间，已有更多的令牌到达。正确的答案应该使用公式 S= C /(M-P)，这里的 S 表示以秒计量的突发时间长度， M 表示以每秒字节计量的最大输出速率， C 表示以字节计的桶的容量， P 表示以每秒字节计量的令牌到达速率。则：

$$S = \frac{(8\times10^6)/8}{(6\times10^6)/8-(1\times10^6)/8} = 1.6\,\mathrm{s}$$

因此，计算机可以用完全速率 6Mb/s 发送 1.6 s 的时间。

5-28 Imagine a flow specification that has a maximum packet size of 1000 bytes, a token bucket rate of 10 million bytes/sec, a token bucket size of 1 million bytes, and a maximum transmission rate of 50 million bytes/sec. How long can a burst at maximum speed last?

28. 想象这样一个流规范：最大分组长度为 1000 字节，令牌桶速率为每秒 10MB，令牌桶的大小为 1M 字节，最大传输速率为每秒 50MB。请问以最大速度传输的突发数据会持续多长时间？

答：令最大突发时间长度为？t 秒，在极端情况下，漏桶在突发期间的开始是充满的（ 1MB），

在突发期间另有 10? t MB 进入桶内。在传输突发期间的输出包含 50? t MB。由等式 1+10? t=50?

t，得到? t=1/40s，即 25ms。因此，以最大速率突发传送可维持 25ms 的时间。

5-31 Consider the user of differentiated services with expedited forwarding. Is there a guarantee that expedited packets experience a shorter delay than regular packets? Why or why not?

31. 请考虑使用快速型转发方法的区分服务用户。是否可以保证快速型分组比常规的分组会经历更短的延迟？为什么是，或者为什么不是？

不能保证。如果过多报文被加快，它们的通道性能可能比常规通道更差。

5-32 Is fragmentation needed in concatenated virtual-circuit internets or only in datagram systems?

32. 在串联虚电路的互联网络中，需要分段机制吗？还是说只有在数据报系统中才需要分段机制？

答：在这两种情况下都需要分割功能。即使在一个串接的虚电路网络中，沿通路的某些网络可能

接受 1024 字节分组，而另一些网络可能仅接受 48 字节分组，分割功能仍然是需要的。

5-34 Suppose that host A is connected to a router R 1, R 1 is connected to another router, R 2, and R 2 is connected to host B. Suppose that a TCP message that contains 900 bytes of data and 20 bytes of TCP header is passed to the IP code at host A for delivery to B. Show the Total length, Identification, DF, MF, and Fragment offset fields of the IP header in each packet transmitted over the three links. Assume that link A-R1 can support a maximum frame size of 1024 bytes including a 14-byte frame header, link R1-R2 can support a maximum frame size of 512 bytes, including an 8-byte frame header, and link R2-B can support a maximum frame size of 512 bytes including a 12-byte frame header.

34. 假设主机 A 被连接到一台路由器 R1 上，R1 又连接到另一台路由器 R2 上，R2 被连接到主机 B。假定一条 TCP 消息包含 900 字节的数据和 20 字节的 TCP 头，现在该消息被传递给主机 A 的 IP 代码，请它递交给主机 B。请写出在三条链路上传输的每个分组中 IP 头部的 Total length、Identification、DF、MF 和 Fragment offset 域。假定链路 A-R1 可以支持的最大帧长度为 1024 字节，其中包括 14 字节的帧头；链路 R1-R2 可以支持的最大帧长度为 512 字节，其中包括 8 字节的帧头；链路 R2-B 可以支持的最大帧长度为 512 字节，其中包括 12 字节的帧头。

开头的 IP 数据报会在 I1 被拆分成两个 IP 数据包，不会出现其他的拆分。？？？

链路 A-R1:

Length = 940; ID = x; DF = 0; MF = 0; Offset = 0

链路 Link R1-R2:

(1) Length = 500; ID = x; DF = 0; MF = 1; Offset = 0
(2) Length = 460; ID = x; DF = 0; MF = 0; Offset = 60

链路 R2-B:

(1) Length = 500; ID = x; DF = 0; MF = 1; Offset = 0
(2) Length = 460; ID = x; DF = 0; MF = 0; Offset = 60

6-1 In our example transport primitives of Fig. 6-2, LISTEN is a blocking call. Is this strictly necessary? If not, explain how a nonblocking primitive could be used. What advantage would this have over the scheme described in the text?

Figure 6-2. The primitives for a simple transport service.

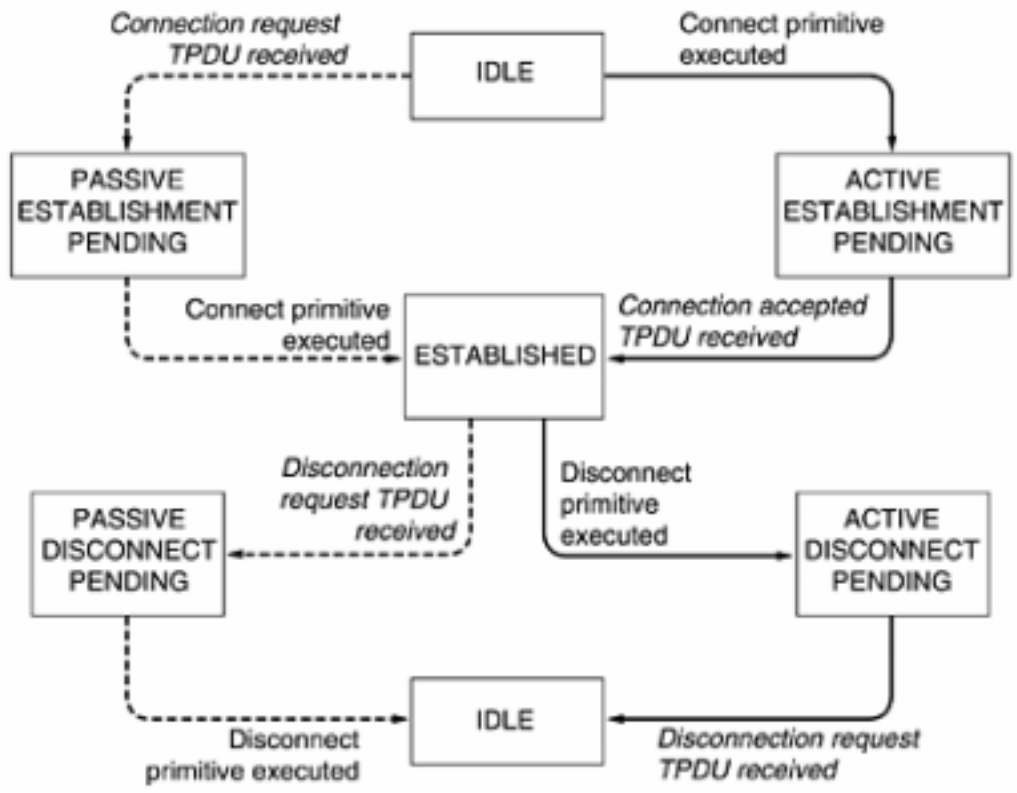
Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

1. 在图 6.2 的传输原语例子中，LISTEN 是一个阻塞调用。这是必要的吗？如果不是，请解释如何有可能使用一个非阻塞的原语。与正文中描述的方案相比，你的方案有什么优点？

答：不是。事实上，LISTEN 调用可以表明建立新连接的意愿，但不封锁。当有了建立连接的尝试时，调用程序可以被提供一个信号。然后，它执行，比如说， OK 或 REJECT 来接受或拒绝连接。然而，在原先的封锁性方案中，就缺乏这种灵活性。

6-2 In the model underlying Fig. 6-4, it is assumed that packets may be lost by the network layer and thus must be individually acknowledged. Suppose that the network layer is 100 percent reliable and never loses packets. What changes, if any, are needed to Fig. 6-4?

Figure 6-4. A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.



2. 在图 6.4 所示的模型中，它的假设条件是网络层的分组有可能会丢失，因此，分组必须被单独确认。假如网络层是百分之百可靠的，它永远不会丢失分组，那么图 6.4 需要做什么样的变化（如果需要的话）？

答：从“被动连接建立在进行中”到“已建立”的虚线不再依确认的传输情况而定。该变迁可立

即发生。实质上，“被动连接建立在进行中”状态已经消失，因为它们什么时候都不可见。

6-4　Suppose that the clock-driven scheme for generating initial sequence numbers is used with a 15-bit wide clock counter. The clock ticks once every 100 msec, and the maximum packet lifetime is 60 sec. How often need resynchronization take place (a) in the worst case? (b) when the data consumes 240 sequence numbers/min?

4. 假设采用时钟驱动方案来生成初始序列号，该方案用到了一个 15 位宽度的时钟计数器，并且，每隔 100ms 时钟滴答一次，最大分组生存期为 60s。请问，每隔多久需要重新同步一次？（a）在最差情况下？（b）当数据每分钟用掉 240 个序列号的时候？」

答：在具体解答这个问题之前，需要先熟悉一下时钟驱动方案的内容。首先我们引入参数　T，假定在发送出一个分组之后等待长度等于　T　的时间，我们就可以肯定，所有关于该分组的踪迹都已消失，不管是该分组本身，还是对于它的确认都不会再以外的出现。我们还假定，每个主机都配有一个表示一天的时间的时钟，　不同主机上的时钟不必同步。　每个时钟都采用二进制计数器的形式，并且以长度一致的间隔时间递增。　而且，计数器的比特数必须等于或超过序列号所使用的比特数。最后一点，时钟被假定是连续运行，即使主机关闭时也不间断。

时钟驱动方案的基本思想是同一时间不会有两个活动的　TPDUs 使用相同的序列号。在一条连接建立的时候，时钟的低端　k　个比特被用作初始序列号（也是　k　位）。因此，每条连接可以从不同的序列号开始为 TPDU 编号。序列号空间应该足够大，使得当编号循环一周时，具有相同号码的旧的 TPDU 已经不复存在。

当主机系统崩溃时会产生一些问题。　在重新启动后，　主机的传输层实体不知道它曾经处在序列号空间的什么位置。一种解决方法是要求传输实体在恢复后的　T　秒内处于空闲状态，让所有老的 TPDUs 都消失。然而，在一个复杂的互联网上，　T　值可能很大，　所以这不是一个好的解决方法。为了避免从崩溃恢复后的　T　秒不工作状态，需要对序列号的使用施加新的限制。在一些编号可能被用作初始序列号之前，必须在长度为　T　的时间内禁止使用这些编号。在任何连接上发送 TPDU 之前，传输层实体必须读一次时钟，检查该　TPDU 的编号是否在禁止区内。

显然，在任何连接上的最大数据率是每个时钟滴答发送一个　TPDU。在系统崩溃后重启动时，在

打开一条新的连接之前，传输实体必须等待到下一个时钟滴答，以避免同样的号码重复使用。如果数据速率低于始终速率，实际使用的序列号对于时间的曲线将最终从左边进入禁止区。如果这样的情况发生了，要么延迟 TPDU 达 T 长度时间，或者重新同步序列号。

作为例子，如果在坐标起点发 1 号 TPDU，到接近时钟大循环编码的末尾才发送第 2 个 TPDU，此时为避免在下一大循环开始重复使用序列号，就需要在大循环接近末尾处重新同步，使用大的初始序列号，以避免使用禁止区号码。

（a） 时钟大循环周期是 215，即 32768 滴答，每滴答 100ms，即 0.1 秒，所以大循环周期是 3276.8s。假定数据产生速率非常低（接近零），那么发送方在 3276.8-60=3271.8秒时进入禁止区，需要进行一次重新同步。

（b） 每分钟使用 240 个序列号，即每秒使用 4 个号码，如果时间以 t 表示（以秒为单位），那么实际的序列号是 4t。当接近大循环的末尾时以及在下一大循环的开始阶段，4t 有一定的大小，位于禁止区的上方，现在由于每秒钟 10 个滴答，禁止区的左边是 10(t-3216.8)，令 4t =10(t-3216.8)，得 t=5316.3秒。即当 t=5316.3时，开始进入禁止区，因此当 t=5316.3时需要进行一次重新同步。

6-5 Why does the maximum packet lifetime, T, have to be large enough to ensure that not only the packet but also its acknowledgements have vanished?

5. 为什么最大分组生存期 T 必须足够大以便确保不仅分组本身消失而且它的确认也消失，然后协议才有效？

答：首先看三次握手过程是如何解决延迟的重复到达的分组所引起的问题的。

正常情况下，当主机 1 发出连接请求时，主机 1 选择一个序号 x，并向主机 2 发送一个包含该序号的请求 TPDU；接着，主机 2 回应一个接受连接的 TPDU，确认 x，并声明自己所选用的初始序列号 y；最后，主机 1 在其发送的第一个数据 TPDU 中确认主机 2 所选择的初始序列号。当出现延迟的重复的控制 TPDU 时，一个 TPDU 是来自于一个已经释放的连接的延迟重复的连接请求（ CONNECTION REQUEST），该 TPDU 在主机 1 毫不知情的情况下到达主机 2。

主机 2 通过向主机 1 发送一个接受连接的 TPDU（CONNECTION ACCEPTED）来响应该 TPDU，而该接受连接的 TPDU 的真正目的是证实主机 1 确实试图建立一个新的连接。在这一点上，关键在于主机 2 建议使用 y 作为从主机 2 到主机 1 交通的初始序列号，从而说明已经不存在包含序列号为 y 的 TPDU，也不存在对 y 的应答分组。当第二个延迟的 TPDU 到达主机 2 时，z 被确认而不是 y 被确认的事实告诉主机 2 这是一个旧的重复的 TPDU，因此废止该连接过程。在这里。三次握手协议是成功的。

最坏的情况是延迟的“ 连接请求 ”和对“ 连接被接收 ”的确认应答都在网络上存活。可以设想，当第 2 个重复分组到达时，如果在网上还存在一个老的对序列号为 y 的分组的确认应答，显然会破坏三次握手协议的正常工作， 故障性的产生一条没有人真正需要的连接， 从而导致灾难性的后果。

6-6 Imagine that a two-way handshake rather than a three-way handshake were used to set up connections. In other words, the third message was not required. Are deadlocks now possible? Give an example or show that none exist.

6. 想象用两步握手过程而不是三步握手过程来建立连接。换句话说，第三个消息不再需要了。现在有可能死锁吗？请给出一个例子，或者证明死锁不存在。

答：我们知道，3 次握手完成两个重要功能， 既要双方做好发送数据的准备工作 （双方都知道彼此已准备好），也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送与确认。

现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子。考虑计算机 A 和 B 之间的通信。假定 B 给 A 发送一个连接请求分组， A 收到了这个分组，并发送了确认应答分组。按照两次握手的协定， A 认为连接已经成功的建立了，可以开始发送数据分组。

可是，B 在 A 的应答分组在传输中被丢失的情况下，将不知道 A 是否已经准备好，不知道 A 建

议什么样的序列号用于 A 到 B 的交通，也不知道 A 是否同意 A 所建议的用于 B 到 A 交通的初始序列号，B 甚至怀疑 A 是否收到自己的连接请求分组。在这种情况下，B 认为连接还未建立成功，将忽略 A 发来的任何数据分组，只等待接收连接确认应答分组。而 A 在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

6-7 Imagine a generalized n-army problem, in which the agreement of any two of the blue armies is sufficient for victory. Does a protocol exist that allows blue to win?

7. 想象一个泛化的 n 一军队问题，在这个问题中，任何两支蓝军达成一致的意见之后就足以取得胜利。是否存在一个能保证蓝军必赢的协议？

答：（a）参见教材。

（b）不存在。对于多于两支部队的情况，问题在实质上是同样的。

6-8 Consider the problem of recovering from host crashes (i.e., Fig. 6-18). If the interval between writing and sending an acknowledgement, or vice versa, can be made relatively small, what are the two best sender-receiver strategies for minimizing the chance of a protocol failure?

Figure 6-18. Different combinations of client and server strategy

Strategy used by sending host	Strategy used by receiving host: First ACK, then write			First write, then ACK		
	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

OK = Protocol functions correctly
DUP = Protocol generates a duplicate message
LOST = Protocol loses a message

8. 请考虑主机的崩溃恢复问题（即图 6.18）。如果写操作和发送确认之间的间隔可以相对非常小的话，那么，为了使协议失败的几率最小，两种最佳的发送方－接收方策略是什么？

答：在解答本题前，让我们先考察主机从崩溃恢复所带来的问题。我们总是希望，在服务器崩溃随后又很快重新引导的情况下，客户机能够继续工作。为了说明这一问题的难度，我们假定一个

客户主机发送一个长文件给另一个服务器主机，并且使用简单的停 -等协议。在服务器上的传输层只是简单的把接收到的 TPDU 一个一个的递交给传输用户。假定在文件传输的过程中，服务器崩溃了。当服务器恢复的时候，它的表被重新初始化，因此再也不知道崩溃前文件传送到什么地方了。

在试图恢复先前状态的过程中， 服务器可能发送一个广播到所有其他主机， 宣布自己刚刚发生了一次崩溃，请求客户告知所有打开的连接的状态。此时。每个客户机都可能处于二中选一的状态：有一个悬而未决的 TPDU 的 S1 状态，或者没有未确认应到的 TPDU 的 S0 状态。

可以想到的一种解决方案是基于这一状态信息，客户机决定是否要重复发最近的一个 TPDU。乍看起来，这一解决方案似乎能解决问题，可是深入仔细的分析一下，困难仍然很大。作为示例，假定服务器的传输层实体先发送 ACK，在 ACK 被发出之后，再执行把收到的 TPDU 写到应用进程的操作。把 TPDU 写到输出设备和发送 ACK 是两个不同的事件，不能同时进行。如果服务器主机的崩溃刚好发生在应答被发送之后， 并且是在写操作之前， 那么客户机将接收到确认应答，当崩溃恢复到达时会处于状态 S0。因此，客户机不会重传 TPDU，错误的认为服务器成功的接收到并存放好了它最后一次发送的 TPDU。实际的情况并非如此， 从而结果是丢失了最后一个 TPDU。

到此，你也许认为：“ 这个问题容易解决，只要你重新编写程序，让传输实体先执行写操作然后再发送 ACK 就可以了。 ”可是，写操作尽管成功了，但崩溃可能发生在发送出 ACK 之前。此时客户机将会处于状态 S1，因而重新发送，导致对服务器的应用进程的输出中产生未检测到的重复 TPDU。

如图 6-18 所示，服务器可以选择两种方式中的一种：先确认应答，或者先执行写操作。客户机可以选择 4 种方式中的一种：总是重传最后一个 TPDU，永不重传最后一个 TPDU，仅在 S0 状态时重传，或者仅在 S1 状态时重传。这样就存在 8 种可能的组合， 但可以看出，对于每一种组合，都有一些事件会使协议的运行失败。

在服务器方可能发生 3 种事件：发送一个 ACK（A），对输出进程的写操作（ W）和系统崩溃

（C）。3 种事件可能以 6 种不同的次序发生：AC（W），AWC，C（AW），C（WA），WAC 和 WC（A），这里的圆括号表示，在系统崩溃 C 后，A 和 W 事件就不可能了。图 6-18 示出了客户机和服务器的策略的所有 8 种组合，以及对于每一种组合的有效事件序列。值得注意的是，对于每一种策略都存在某些事件会引起协议失败。 例如，如果客户机选择总是重发送， AWC 事件将产生检测不出来的收到重复分组的错误。尽管对于 C（AW）和 C（WA）该协议都工作的很好。

本题的答案。如果 AW 或 WA 间隔时间很短，事件 AC（W）和 W（CA）就不太可能发生。此时最好发送方策略是，如果崩溃恢复时处于状态 S1，应该重传最后一个 TPDU，接收方采用顺序 AW 或 WA 则无关紧要。

6-9 Are deadlocks possible with the transport entity described in the text (Fig. 6-20)?

Figure 6-20. An example transport entity.

```
#define MAX_CONN 32                    /* max number of simultaneous connections */
#define MAX_MSG_SIZE 8192              /* largest message in bytes */
#define MAX_PKT_SIZE 512               /* largest packet in bytes */
#define TIMEOUT 20
#define CRED 1
#define OK 0

#define ERR_FULL -1
#define ERR_REJECT -2
#define ERR_CLOSED -3
#define LOW_ERR -3

typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT} pkt_type;
typedef enum {IDLE,WAITING,QUEUED,ESTABLISHED,SENDING,RECEIVING,DISCONN}  cstate;

/* Global variables. */
transport_address listen_address;      /* local address being listened to */
int listen_conn;                       /* connection identifier for listen */
unsigned char data[MAX_PKT_SIZE];      /* scratch area for packet data */

struct conn {
  transport_address local_address, remote_address;
  cstate state;                        /* state of this connection */
  unsigned char *user_buf_addr;        /* pointer to receive buffer */
  int byte_count;                      /* send/receive count */
  int clr_req_received;                /* set when CLEAR_REQ packet received */
  int timer;                           /* used to time out CALL_REQ packets */
  int credits;                         /* number of messages that may be sent */
} conn[MAX_CONN + 1];                  /* slot 0 is not used */
```

```
void sleep(void);                                   /* prototypes */
void wakeup(void);
void to_net(int cid, int q, int m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt, unsigned char *p, int *bytes);

int listen(transport_address t)
{ /* User wants to listen for a connection. See if CALL_REQ has already arrived. */
  int i, found = 0;

  for (i = 1; i <= MAX_CONN; i++)                   /* search the table for CALL_REQ */
       if (conn[i].state == QUEUED && conn[i].local_address == t) {
            found = i;
            break;
       }

  if (found == 0) {
       /* No CALL_REQ is waiting.  Go to sleep until arrival or timeout. */
       listen_address = t;  sleep();  i = listen_conn ;
  }
  conn[i].state = ESTABLISHED;                      /* connection is ESTABLISHED */
  conn[i].timer = 0;                                /* timer is not used */
    listen_conn = 0;                                /* 0 is assumed to be an invalid address */
    to_net(i, 0, 0, CALL_ACC, data, 0);             /* tell net to accept connection */
    return(i);                                      /* return connection identifier */
  }

int connect(transport_address l, transport_address r)
{ /* User wants to connect to a remote process;  send CALL_REQ packet. */
  int i;
  struct conn *cptr;

  data[0] = r;   data[1] = l;                       /* CALL_REQ packet needs these */
  i = MAX_CONN;                                     /* search table backward */
  while (conn[i].state != IDLE && i > 1) i = i – 1;
  if (conn[i].state == IDLE) {
       /* Make a table entry that CALL_REQ has been sent. */
       cptr = &conn[i];
       cptr->local_address = l; cptr->remote_address = r;
       cptr->state = WAITING; cptr->clr_req_received = 0;
       cptr->credits = 0; cptr->timer = 0;
       to_net(i, 0, 0, CALL_REQ, data, 2);
       sleep();                                     /* wait for CALL_ACC or CLEAR_REQ */
       if (cptr->state == ESTABLISHED) return(i);
       if (cptr->clr_req_received) {
            /* Other side refused call. */
            cptr->state = IDLE;                     /* back to IDLE state */
            to_net(i, 0, 0, CLEAR_CONF, data, 0);
            return(ERR_REJECT);
       }
  } else return(ERR_FULL);                          /* reject CONNECT: no table space */
}
```

```
int send(int cid, unsigned char bufptr[], int bytes)
{ /* User wants to send a message. */
  int i, count, m;
  struct conn *cptr = &conn[cid];

  /* Enter SENDING state. */
  cptr->state = SENDING;
  cptr->byte_count = 0;                          /* # bytes sent so far this message */
  if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
  if (cptr->clr_req_received == 0) {
       /* Credit available; split message into packets if need be. */
       do {
            if (bytes – cptr->byte_count > MAX_PKT_SIZE) {/* multipacket message */
                 count = MAX_PKT_SIZE;  m = 1;  /* more packets later */
            } else {                             /* single packet message */
                 count = bytes – cptr->byte_count;  m = 0;    /* last pkt of this message */
            }
            for (i = 0; i < count; i++) data[i] = bufptr[cptr->byte_count + i];
            to_net(cid, 0, m, DATA_PKT, data, count);   /* send 1 packet */
            cptr->byte_count = cptr->byte_count + count;     /* increment bytes sent so far */
       } while (cptr->byte_count < bytes);         /* loop until whole message sent */
       cptr->credits--;                            /* each message uses up one credit */
       cptr->state = ESTABLISHED;
       return(OK);
  } else {
       cptr->state = ESTABLISHED;
       return(ERR_CLOSED);                         /* send failed: peer wants to disconnect */
  }
}


int receive(int cid, unsigned char bufptr[], int *bytes)
{ /* User is prepared to receive a message. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received == 0) {
       /* Connection still established; try to receive. */
       cptr->state = RECEIVING;
       cptr->user_buf_addr = bufptr;
       cptr->byte_count = 0;
       data[0] = CRED;
       data[1] = 1;
       to_net(cid, 1, 0, CREDIT, data, 2);         /* send credit */
       sleep();                                    /* block awaiting data */
       *bytes = cptr->byte_count;
  }
  cptr->state = ESTABLISHED;
  return(cptr->clr_req_received ? ERR_CLOSED : OK);
}

int disconnect(int cid)
{ /* User wants to release a connection. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received) {                  /* other side initiated termination */
       cptr->state = IDLE;                       /* connection is now released */
       to_net(cid, 0, 0, CLEAR_CONF, data, 0);
  } else {                                       /* we initiated termination */
       cptr->state = DISCONN;                    /* not released until other side agrees */
       to_net(cid, 0, 0, CLEAR_REQ, data, 0);
  }
  return(OK);
}
```

```
void packet_arrival(void)
{ /* A packet has arrived, get and process it. */
  int cid;                                      /* connection on which packet arrived */
  int count, i, q, m;
  pkt_type ptype;     /* CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT */
  unsigned char data[MAX_PKT_SIZE];         /* data portion of the incoming packet */
  struct conn *cptr;

  from_net(&cid, &q, &m, &ptype, data, &count);   /* go get it */
  cptr = &conn[cid];
  switch (ptype) {
    case CALL_REQ:                              /* remote user wants to establish connection */
      cptr->local_address = data[0];  cptr->remote_address = data[1];
      if (cptr->local_address == listen_address) {
            listen_conn = cid;  cptr->state = ESTABLISHED;  wakeup();
      } else {
            cptr->state = QUEUED;  cptr->timer = TIMEOUT;
      }
      cptr->clr_req_received = 0;   cptr->credits = 0;
      break;
    case CALL_ACC:                              /* remote user has accepted our CALL_REQ */
      cptr->state = ESTABLISHED;
      wakeup();
      break;
    case CLEAR_REQ:                             /* remote user wants to disconnect or reject call */
      cptr->clr_req_received = 1;
      if (cptr->state == DISCONN) cptr->state = IDLE; /* clear collision */
      if (cptr->state ==  WAITING || cptr->state == RECEIVING || cptr->state == SENDING) wakeup();
      break;
    case CLEAR_CONF:                            /* remote user agrees to disconnect */
      cptr->state = IDLE;
      break;
    case CREDIT:                                /* remote user is waiting for data */
      cptr->credits += data[1];
      if (cptr->state == SENDING) wakeup();
      break;
    case DATA_PKT:                              /* remote user has sent data */
      for (i = 0; i < count; i++) cptr->user_buf_addr[cptr->byte_count + i] = data[i];
      cptr->byte_count += count;
      if (m == 0 ) wakeup();
  }
}

void clock(void)
{ /* The clock has ticked, check for timeouts of queued connect requests. */
  int i;
  struct conn *cptr;

  for (i = 1; i <= MAX_CONN; i++) {
      cptr = &conn[i];
      if (cptr->timer > 0) {                    /* timer was running */
            cptr->timer--;
            if (cptr->timer == 0) {             /* timer has now expired */
                  cptr->state = IDLE;
                  to_net(i, 0, 0, CLEAR_REQ, data, 0);
            }
      }
  }
}
```

IDLE — Connection not established yet.

WAITING — CONNECT has been executed and CALL REQUEST sent.

QUEUED— A CALL REQUEST has arrived; no LISTEN yet.

ESTABLISHED— The connection has been established.

SENDING— The user is waiting for permission to send a packet.

RECEIVING— A RECEIVE has been done.

DISCONNECTING— A DISCONNECT has been done locally.

9. 对于正文中描述的传输实体(见图 6.20),死锁有可能吗?

答：该传输实体有可能死锁。当双方同时执行 RECEIVE 时就会进入死锁状态。

6-10 Out of curiosity, the implementer of the transport entity of Fig. 6-20 has decided to put counters inside the sleep procedure to collect statistics about the conn array. Among these are the number of connections in each of the seven possible states, ni (i = 1, ... ,7). After writing a massive FORTRAN program to analyze the data, our implementer discovers that the relation Sni = MAX_CONN appears to always be true. Are there any other invariants involving only these seven variables?

10. 图 6.20 所示的传输实体的实现者出于好奇心,因而决定在 sleep 过程中放上一些计数器,以便收集有关 conn 数组的统计信息。其中包括这样的信息:处于 7 种可能状态之一的连接数目,$n_i(i = 1, \cdots, 7)$。他编写了一个很大的 FORTRAN 程序用于分析数据,在写完程序之后,他发现 $\sum n_i = MAX\_CONN$ 这个关系总是成立的。请问,是否还有其他仅仅涉及到这 7 个变量的恒等关系?

答：有， n2 +n3+n6+ n7=1

因为状态 listening（n2）、waiting（n 3）、sending（n6）和 receiving（n7）都意味着用户被封锁,

因此当处在其中的一个状态时，就不可能是在另一个状态。

6-11 What happens when the user of the transport entity given in Fig. 6-20 sends a zero-length message? Discuss the significance of your answer.

11. 对于图 6.20 中给出的传输实体,当它的用户发送一个 0 长度的消息时会发生什么情况?请讨论你的答案的意义所在。

答：长度为零的报文被另一边接收。这种报文的发送可以被用来表示文件结束的信号。

6-12 For each event that can potentially occur in the transport entity of Fig. 6-20, tell whether it is legal when the user is sleeping in sending state.

12. 在图 6.20 的传输实体中,针对可能发生的每一个事件,请说明当用户在发送状态(sending)睡觉的时候,该事件是否为合法的?

答：因为文件处于封锁状态，所有的传输层原语都不可能执行。 因此，仅分组到达事件是可能的，而且还不是所有的到达事件。事实上， 仅仅跟呼叫请求、清除请求、数据分组和信用量分组这几个分组到达有关的事件是合法的。

6-13 Discuss the advantages and disadvantages of credits versus sliding window protocols.

13. 请讨论一下信用协议和滑动窗口协议的优点和缺点。

答：滑动窗口协议比较简单，仅需要管理窗口边缘一组参数，而且，对于到达顺序有错的 TPDU 不会引起窗口增加和减少方面的问题。然而，信用量方案比较灵活， 允许独立于确认，动态的管理缓冲区。

6-14 Why does UDP exist? Would it not have been enough to just let user processes send raw IP packets?

14. UDP 为什么有必要存在？难道只让用户进程发送原始的 IP 分组还不够吗？

答：仅仅使用 IP 分组还不够。IP 分组包含 IP 地址，该地址指定一个目的地机器。一旦这样的分组到达了目的地机器，网络控制程序如何知道该把它交给哪个进程呢？ UDP 分组包含一个目的地端口，这一信息是必须的，因为有了它，分组才能够被投递给正确的进程。

6-15 Consider a simple application-level protocol built on top of UDP that allows a client to retrieve a file from a remote server residing at a well-known address. The client first sends a request with file name, and the server responds with a sequence of data packets containing different parts of the requested file. To ensure reliability and sequenced delivery, client and server use a stop-and-wait protocol. Ignoring the obvious performance issue, do you see a problem with this protocol? Think carefully about the possibility of processes crashing.

15. 请考虑一个建立在 UDP 基础上的简单应用层协议，它允许客户从一个远程服务器获取文件，而且该服务器位于一个知名的地址上。客户首先发送一个请求，该请求中包含了文件名，然后服务器以一个数据分组序列作为响应，这些数据分组包含了客户所请求的文件的不同部分。为了确保可靠性和顺序递交，客户和服务器使用了停一等协议。忽略显然存在的性能问题，你还看得到这个协议的另一个问题吗？请仔细想一想进程崩溃的可能性。

客户有可能得到错误的文件。假设客户 A 发送了一个对于文件 f1 的请求，然后就崩溃了。另一个客户 B 接着使用相同的协议请求另一个文件 f2。假设客户 B 运行在与 A 相同的机器上（具有相同的 IP 地址） ，把它的 UDP 套接字连接到与 A 先前用过的相同的端口上。此外，假设 B 的请求丢失了，当服务器的应答（对 A 的请求）到达时，客户 B 将接收并猜想着是给它自己的应答。

6-18 Both UDP and TCP use port numbers to identify the destination entity when delivering a message. Give two reasons for why these protocols invented a new abstract ID (port numbers), instead of using process IDs, which already existed when these protocols were designed.

18. UDP 和 TCP 在递交消息的时候，都使用端口号来标识目标实体。请给出两个理由说明为什么这两个协议要发明一个新的抽象 ID(端口号)，而不是使用进程 ID(在设计这两个协议的时候，进程 ID 早已经存在了)。

有三个原因。第一，进程 ID 是 OS 特定的，使用进程 ID 将使得协议 OS 依赖；第二，一个单一的进程可能建立多个通信通道，单一的进程 ID（每个进程）作为目的标识符不能被用来区别这些通道；第三，进程很容易监听众所周知的端口，但众所周知的进程 ID 是不可能的。

6-19 What is the total size of the minimum TCP MTU, including TCP and IP overhead but not including data link layer overhead?

19. 最小的 TCP MTU 的总长度(包括 TCP 和 IP 的开销但是不包括数据链路层的开销)是多少?

默认的分段是 536bytes，TCP 增加 20 bytes，IP 也是如此，造成默认合计 576 bytes。

6-20 Datagram fragmentation and reassembly are handled by IP and are invisible to TCP. Does this mean that TCP does not have to worry about data arriving in the wrong order?

20. 数据报的分段和重组机制是由 IP 来处理的，对于 TCP 是不可见的。这是否意味着 TCP 不用担心数据错序到达的问题呢?

答：尽管到达的每个数据报都是完整的，但可能到达的数据报的顺序是错误的，因此，TCP 必须准备适当的重组报文的各个部分。

6-23 A process on host 1 has been assigned port p, and a process on host 2 has been assigned port q. Is it possible for there to be two or more TCP connections between these two ports at the same time?

23. 主机 1 上的一个进程已经被分配了端口 p，主机 2 上的一个进程已经被分配了端口 q，请问这两个端口之间有可能同时存在两个或者多个 TCP 连接吗?

答：不可以。一条连接仅仅用它的套接口标识。因此，（1，p）-（2，q）是在这两个端口之间唯一可能的连接。

7-1 Many business computers have three distinct and worldwide unique identifiers. What are they?

1. 许多商用计算机有三个不同的全球惟一标识符。它们是什么?

它们是 DNS 名称，IP 地址，以太网地址。

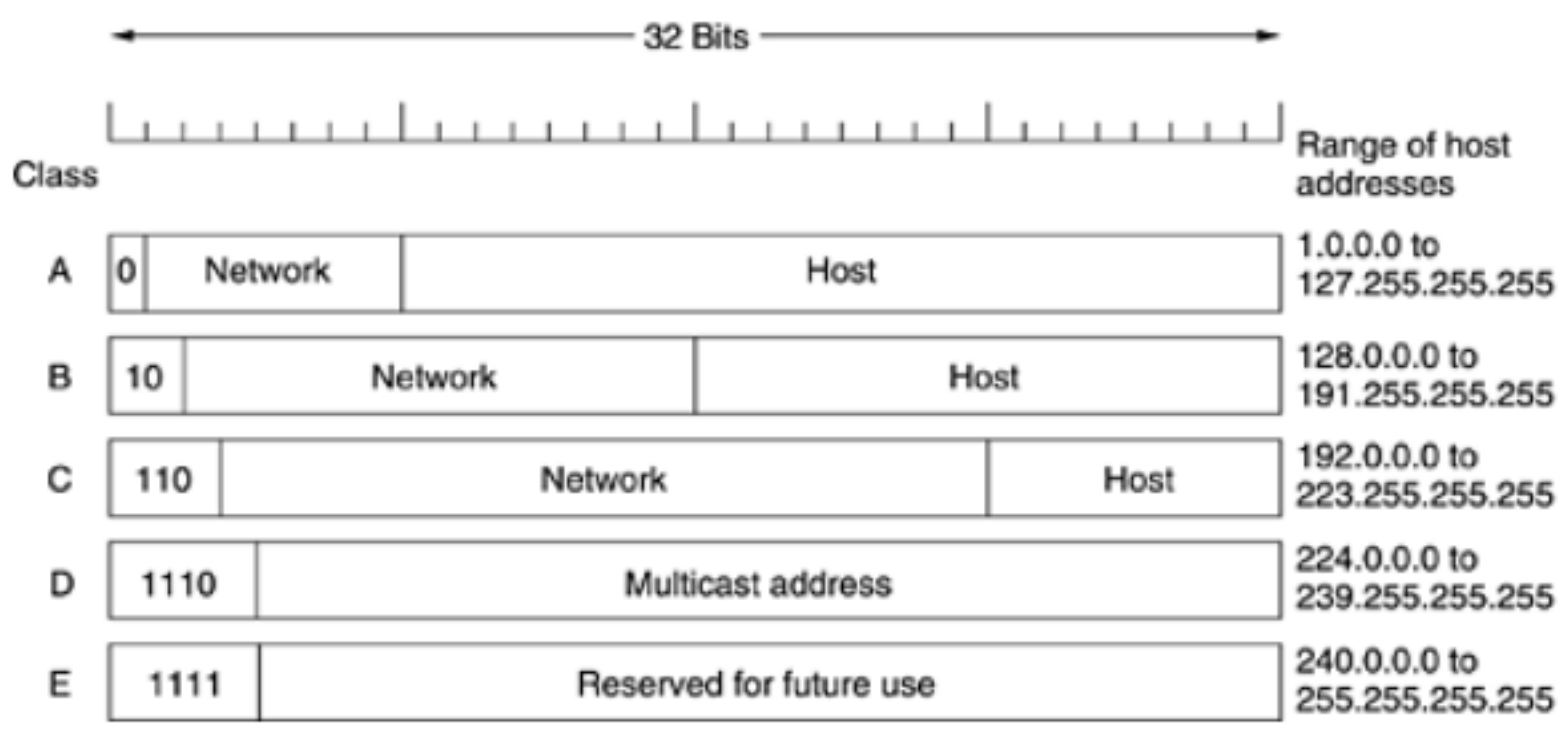
7-2 According to the information given in Fig. 7-3, is little-sister.cs.vu.nl on a class A, B, or C network?

Figure 7-3. A portion of a possible DNS database for *cs.vu.nl*

```
; Authoritative data for cs.vu.nl
cs.vu.nl.        86400   IN  SOA     star boss (9527,7200,7200,241920,86400)
cs.vu.nl.        86400   IN  TXT     "Divisie Wiskunde en Informatica."
cs.vu.nl.        86400   IN  TXT     "Vrije Universiteit Amsterdam."
cs.vu.nl.        86400   IN  MX      1 zephyr.cs.vu.nl.
cs.vu.nl.        86400   IN  MX      2 top.cs.vu.nl.

flits.cs.vu.nl.  86400   IN  HINFO   Sun Unix
flits.cs.vu.nl.  86400   IN  A       130.37.16.112
flits.cs.vu.nl.  86400   IN  A       192.31.231.165
flits.cs.vu.nl.  86400   IN  MX      1 flits.cs.vu.nl.
flits.cs.vu.nl.  86400   IN  MX      2 zephyr.cs.vu.nl.
flits.cs.vu.nl.  86400   IN  MX      3 top.cs.vu.nl.

www.cs.vu.nl.    86400   IN  CNAME   star.cs.vu.nl
ftp.cs.vu.nl.    86400   IN  CNAME   zephyr.cs.vu.nl

rowboat                  IN  A       130.37.56.201
                         IN  MX      1 rowboat
                         IN  MX      2 zephyr
                         IN  HINFO   Sun Unix

little-sister            IN  A       130.37.62.23
                         IN  HINFO   Mac MacOS

laserjet                 IN  A       192.31.231.216
                         IN  HINFO   "HP Laserjet IIISi" Proprietary
```

2. 根据图 7.3 中给出的信息，little－sister. cs. vu. nl 在一个 A、B 还是 C 类网络上?

它的 IP 地址以 130开始，所以它在一个 B 级网络上。



7-5 DNS uses UDP instead of TCP. If a DNS packet is lost, there is no automatic recovery. Does this cause a problem, and if so, how is it solved?

5. DNS 使用 UDP 协议而不是 TCP 协议。如果一个 DNS 分组丢失了，这里也没有自动恢复的机制。请问这会产生问题吗？如果会的话，它是如何被解决的？

DNS 是幂等的，操作可以被重复而不会导致损害。当一个进程发起 DNS 请求时，它启动一个定时器。当定时器终止时，它仅仅再次发起请求。没有损害会发生。

7-6 In addition to being subject to loss, UDP packets have a maximum length, potentially as low as 576 bytes. What happens when a DNS name to be looked up exceeds this length? Can it be sent in two packets?

6. 除了可能会丢失，UDP 分组还有最大长度限制，甚至可能少到只有 576 个字节。当一个待查找的 DNS 名字超过这个长度时会怎么样？它可以被放在两个分组中发送吗？

这问题不会发生。DNS 名字必须短于 256 bytes，标准要求如此，因此所有的 DNS 名称都能适合一个单一的最小长度报文。.

7-7 Can a machine with a single DNS name have multiple IP addresses? How could this occur?

7. 如果一台机器只有一个 DNS 名字，那么它可以有多个 IP 地址吗？这种情形是如何发生的？

使得，事实上，在图 7-3（7.2 题），我们能看到一个多重 IP 地址的例子。记住，IP 地址由网络号和主机号组成。如果一台机器有两个以太网卡，它可以处于两个单独的网络上，假如这样的话，它需要两个 IP 地址。

7-8 Can a computer have two DNS names that fall in different top-level domains? If so, give a plausible example. If not, explain why not.

8. 一台计算机可以有两个分别属于不同顶级域的 DNS 名字吗？如果可以，请给出一个可能的例子。如果不可以，请解释原因。

有可能，www.large-bank.com 和 www.large-bank.ny.us 可能拥有相同的 IP 地址。因此，一个 com

域下的项目和一个国家域下的项目无疑是可能的（而且常见）。

7-9 The number of companies with a Web site has grown explosively in recent years. As a result, thousands of companies are registered in the com domain, causing a heavy load on the top-level server for this domain. Suggest a way to alleviate this problem without changing the naming scheme (i.e., without introducing new top-level domain names). It is permitted that your solution requires changes to the client code.

9. 近年来，拥有 Web 站点的公司的数量爆炸式地增长。结果是，成千上万的公司被注册在 com 域中，从而导致该域的顶级服务器的负载极其繁重。请提出一种建议方案来缓解这个问题，同时又不用修改命名方案（也就是说，不引入新的顶级域名）。你的方案可以要求修改客户代码。

显然有许多方法。一种是把顶级服务器转为服务器外包；另一种是有 26 台分开的服务器，一台用于以 a 开头的名称，一台用于 b，等等。在引入新服务器后的某些时间周期（比方说 3 年），旧的服务器仍能继续运行以向人们提供一个适用他们软件的机会。

7-10 Some e-mail systems support a header field Content Return:. It specifies whether the body of a message is to be returned in the event of nondelivery. Does this field belong to the envelope or to the header?

10. 有些电子邮件系统支持 Content Return：头域。它指定了当消息未被递交时是否返回消息体。请问：这个域属于信封还是消息头？

它属于信封，因为分发系统需要知道它的值，用来处理无法递送的电子邮件。

7-11 Electronic mail systems need directories so people's e-mail addresses can be looked up. To build such directories, names should be broken up into standard components (e.g., first name, last name) to make searching possible. Discuss some problems that must be solved for a worldwide standard to be acceptable.

11. 电子邮件系统需要相应的目录，以便于查找人们的电子邮件地址。为了建立这样的目录，名字应该被分割成标准的组成部分（比如说姓和名）以便于搜索。请讨论为了建立一个可接受的全球标准而需要解决的一些问题。

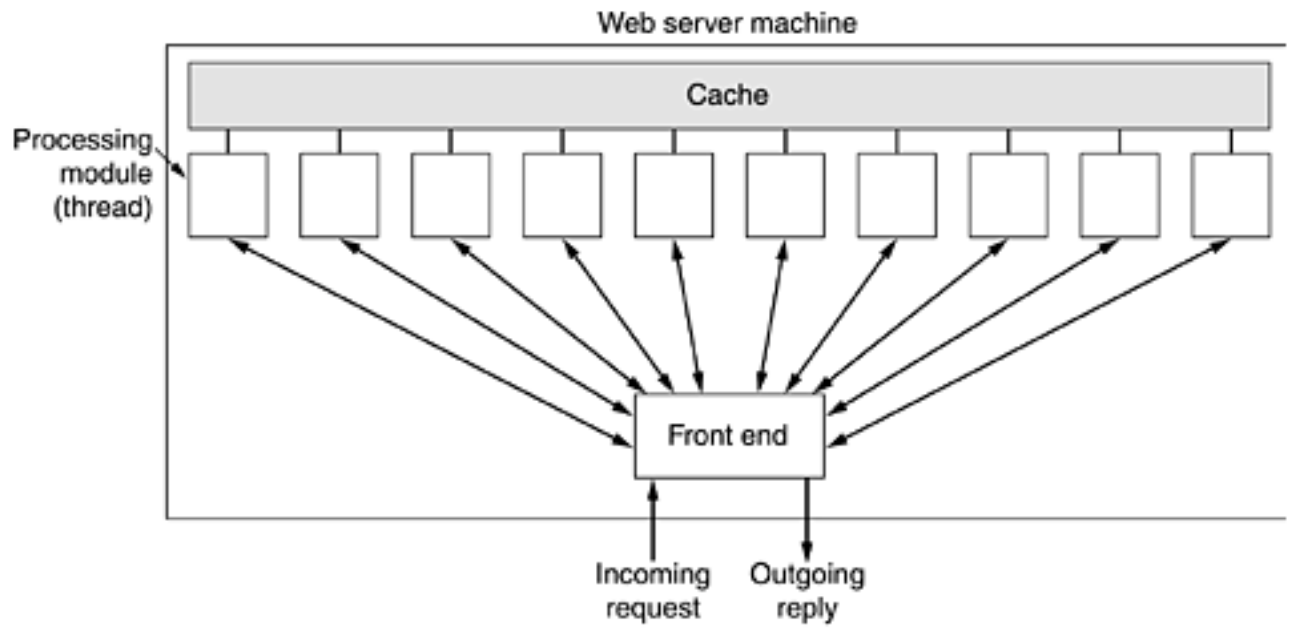
这远比你能想到的要复杂。以开始而言，全世界一半人先写名字，而后写姓氏。另外一半人（例如中国）则不同。命名系统将不得不辨别任意数量的特定名字，加上姓氏，尽管后者可能具有若干部分，例如 John von Neumann 接着有些人有一个中间首字母，但没有中间名。各种各样的称谓，例如 Mr., Miss, Mrs., Ms., Dr., Prof.,或 Lord，可以作为名字的前缀称谓。人们来自不同世

代，于是 Jr., Sr., III, IV 等等不得不也被包括进来。一些人在名字中使用他们的学术头衔，所以我们需要 B.A., B.Sc., M.A., M.Sc., Ph.D 以及其它学位。最后，有些人在名字中列入了奖励和荣誉。例如，英格兰队皇家协会的会员可能加上 FRS。现在我们能够接受学者的名字：

Prof. Dr. Abigail Barbara Cynthia Doris E. de Vries III, Ph.D., FRS

7-26 A multithreaded Web server is organized as shown in Fig. 7-21. It takes 500 μsec to accept a request and check the cache. Half the time the file is found in the cache and returned immediately. The other half of the time the module has to block for 9 msec while its disk request is queued and processed. How many modules should the server have to keep the CPU busy all the time (assuming the disk is not a bottleneck)?

Figure 7-21. A multithreaded Web server with a front end and processing modules.



26. 一个多线程的 Web 服务器被组织成如图 7.21 所示的结构。它需要 500μs 来接受一个请求并检查缓存。在一半情况下，它可以在缓存中找到文件，并立即返回。另外一半情况下，该模块必须阻塞 9ms 以等待它的磁盘请求被排队和处理。该服务器应该有多少个模块才能保持 CPU 一直处于忙的状态(假设磁盘不是一个瓶颈)?

如果一个模块得到两个请求，平均一个缓存命中一个缓存缺失。总的 CPU 时间消耗是 1 msec，总的等待时间是 9 msec，这得出一个 10%的 CPU 利用率。所以 10 个模块能使得 CPU 保持忙碌。

7-27 The standard http URL assumes that the Web server is listening on port 80. However, it is possible for a Web server to listen to some other port. Devise a reasonable syntax for a URL accessing a file on a nonstandard port.

27. 标准的 http URL 假设 Web 服务器在 80 端口上监听。然而，Web 服务器在其他的端口上进行监听也是有可能的。请设计一种合理的 URL 语法来支持在非标准端口上访问文件。

正式的 Internet标准（草案） RFC 1738方法是 http://dns-name:port/file.

7-28 Although it was not mentioned in the text, an alternative form for a URL is to use the IP address instead of its DNS name. An example of using an IP address is http://192.31.231.66/index.html. How does the browser know whether the name following the scheme is a DNS name or an IP address?

28. 尽管在正文中没有提及，URL 的一种替换形式是使用 IP 地址而不是 DNS 名字。一个使用 IP 地址的 URL 例子是 http://192.31.231.66/index.html。浏览器如何知道协议名后面的名字是 DNS 名字还是 IP 地址？

DNS 名称不会以数字终止，所以不会有歧义。