第 14 周作业答案

5-37、5-38、5-39、5-41、5-46、6-02、6-03、6-08、6-12、6-16、6-20

5-37：

【5-37】 在 TCP 的拥塞控制中，什么是慢开始、拥塞避免、快重传和快恢复算法？这里每一种算法各起什么作用？“乘法减小”和“加法增大”各用在什么情况下？

解答：慢开始算法的思路是这样的：当主机开始发送数据时，如果立即把大量数据字节注入到网络，那么就有可能引起网络拥塞，因为现在并不清楚网络的负荷情况。经验证明，较好的方法是先探测一下，即由小到大逐渐增大发送窗口，也就是说，由小到大逐渐增大拥塞窗口数值。通常在刚刚开始发送报文段时，先把拥塞窗口 cwnd 设置为一个最大报文段 MSS 的数值。而在每收到一个对新的报文段的确认后，把拥塞窗口增加至多一个 MSS 的数值。用这样的方法逐步增大发送方的拥塞窗口 cwnd，可以使分组注入到网络的速率更加合理。使用慢开始算法后，每经过一个 RTT，拥塞窗口 cwnd 就加倍。

为了防止拥塞窗口 cwnd 增长过大引起网络拥塞，还需要设置一个慢开始门限 ssthresh 状态变量。当 cwnd > ssthresh 时，停止使用慢开始算法而改用拥塞避免算法。

拥塞避免算法的思路是让拥塞窗口 cwnd 缓慢地增大，即每经过一个往返时间 RTT 就把发送方的拥塞窗口 cwnd 加 1，而不是加倍。这样，拥塞窗口 cwnd 按线性规律缓慢增长，比慢开始算法的拥塞窗口增长速率缓慢得多。

快重传算法首先要求接收方每收到一个失序的报文段后，就立即发出重复确认（为的是使发送方及早知道有报文段没有到达对方），而不要等待自己发送数据时才进行捎带确认。

快恢复算法，其过程有以下两个要点：

(1) 当发送方连续收到三个重复确认时，就执行“乘法减小”算法，把慢开始门限 ssthresh 减半。这是为了预防网络发生拥塞。请注意，接下去不执行慢开始算法。

(2) 由于发送方现在认为网络很可能没有发生拥塞，因此不执行慢开始算法，而是把 cwnd 值设置为慢开始门限 ssthresh 减半后的数值，然后开始执行拥塞避免算法（“加法增大”），使拥塞窗口缓慢地线性增大。

“乘法减小”是指不论在慢开始阶段还是拥塞避免阶段，只要出现超时（即很可能出现了网络拥塞），就把慢开始门限值 ssthresh 减半，即设置为当前拥塞窗口的一半（与此同时，执行慢开始算法）。当网络频繁出现拥塞时，ssthresh 值就下降得很快，以大大减少注入到网络中的分组数。

“加法增大”是指执行拥塞避免算法后，使拥塞窗口缓慢增大，以防止网络过早出现拥塞。

5-38：

【5-38】 设 TCP 的 ssthresh 的初始值为 8（单位为报文段）。当拥塞窗口上升到 12 时网络发生了超时，TCP 使用慢开始和拥塞避免。试分别求出 RTT = 1 到 RTT = 15 时的各拥塞窗口大小。你能说明拥塞窗口每一次变化的原因吗？

解答：拥塞窗口大小及变化原因见表 T-5-38。

表 T-5-38　15 个 RTT 与拥塞窗口大小及变化原因

RTT	拥塞窗口	拥塞窗口变化的原因
1	1	网络发生了超时，TCP 使用慢开始算法
2	2	拥塞窗口值加倍
3	4	拥塞窗口值加倍
4	8	拥塞窗口值加倍，这是 ssthresh 的初始值
5	9	TCP 使用拥塞避免算法，拥塞窗口值加 1
6	10	TCP 使用拥塞避免算法，拥塞窗口值加 1
7	11	TCP 使用拥塞避免算法，拥塞窗口值加 1
8	12	TCP 使用拥塞避免算法，拥塞窗口值加 1
9	1	网络发生了超时，TCP 使用慢开始算法
10	2	拥塞窗口值加倍
11	4	拥塞窗口值加倍
12	6	拥塞窗口值加倍，但到达 12 的一半时，改为拥塞避免算法
13	7	TCP 使用拥塞避免算法，拥塞窗口值加 1
14	8	TCP 使用拥塞避免算法，拥塞窗口值加 1
15	9	TCP 使用拥塞避免算法，拥塞窗口值加 1

5-39：

【5-39】TCP 的拥塞窗口 cwnd 大小与 RTT 的关系如表 T-5-39 所示。

表 T-5-39　拥塞窗口 cwnd 大小与 RTT 的关系

cwnd	1	2	4	8	16	32	33	34	35	36	37	38	39
RTT	1	2	3	4	5	6	7	8	9	10	11	12	13
cwnd	40	41	42	21	22	23	24	25	26	1	2	4	8
RTT	14	15	16	17	18	19	20	21	22	23	24	25	26

(1) 试画出如教材的图 5-25 所示的拥塞窗口与 RTT 的关系曲线。

(2) 指明 TCP 工作在慢开始阶段的时间间隔。

(3) 指明 TCP 工作在拥塞避免阶段的时间间隔。

(4) 在 RTT= 16 和 RTT = 22 之后发送方是通过收到三个重复的确认还是通过超时检测到丢失了报文段？

(5) 在 RTT= 1，RTT = 18 和 RTT = 24 时，门限 ssthresh 分别被设置为多大？

(6) 在 RTT 等于多少时发送出第 70 个报文段？

(7) 假定在 RTT = 26 之后收到了三个重复的确认，因而检测出了报文段的丢失，那么拥塞窗口 cwnd 和门限 ssthresh 应设置为多大？

解答：

(1) 拥塞窗口与 RTT 的关系曲线如图 T-5-39 所示。

(2) 慢开始时间间隔：[RTT = 1, RTT = 6] 和 [RTT = 23, RTT = 26]。

(3) 拥塞避免时间间隔：[RTT = 6, RTT = 16] 和 [RTT = 17, RTT = 22]。

(4) 在 RTT = 16 之后发送方通过收到三个重复的确认检测到丢失了报文段，因为题目给出，下一个 RTT 的拥塞窗口减半了。

在 RTT = 22 之后发送方通过超时检测到丢失了报文段，因为题目给出，下一个 RTT 的拥塞窗口下降到 1 了。

(5) 在 RTT = 1 时，门限 ssthresh 被设置为 32，因为从 RTT = 6 起，就进入了拥塞避免状态，每经过一个 RTT，拥塞窗口就加 1。

在 RTT = 18 时，门限 ssthresh 被设置为发生拥塞时拥塞窗口 42 的一半，即 21。

在 RTT = 24 时，门限 ssthresh 被设置为发生拥塞时拥塞窗口 26 的一半，即 13。

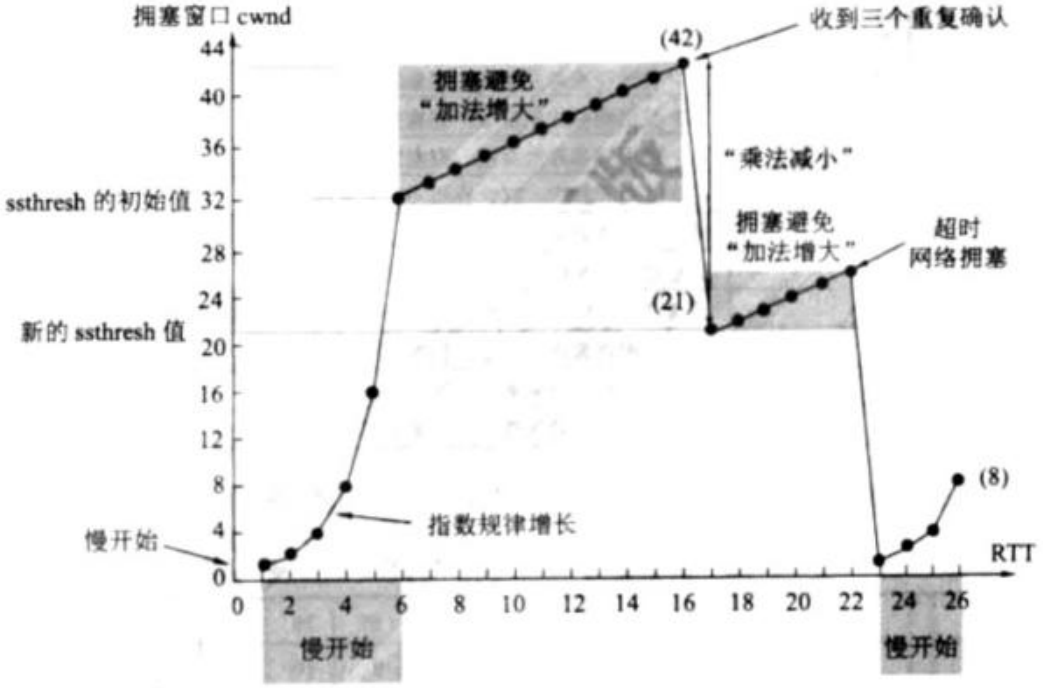


图 T-5-39 拥塞窗口与传输 RTT 的关系曲线

(6) RTT = 1 时，发送报文段 1。(cwnd = 1)

RTT = 2 时，发送报文段 2, 3。(cwnd = 2)

RTT = 3 时，发送报文段 4 ~ 7。(cwnd = 4)

RTT = 4 时，发送报文段 8 ~ 15。(cwnd = 8)

RTT = 5 时，发送报文段 16 ~ 31。(cwnd = 16)

RTT = 6 时，发送报文段 32 ~ 63。(cwnd = 32)

RTT = 7 时，发送报文段 64 ~ 94。(cwnd = 33)

因此，第 70 报文段在 RTT = 7 时发送出。

(7) 检测出报文段丢失时，拥塞窗口 cwnd 是 8，因此拥塞窗口 cwnd 的数值应当减半，等于 4，而门限 ssthresh 应设置为检测出报文段丢失时拥塞窗口 8 的一半，即 4。

请注意，只有当 RTT 为整数值时，图 T-5-39 的曲线才有意义。图中圆点之间的连线只是为了便于我们观察拥塞窗口的变化。例如，当 RTT = 3 时，cwnd = 32。当 RTT = 4 时，cwnd = 8。但是当 RTT 为 3 ~ 4 之间的任何值时，cwnd 是多少我们都无法得知。

5-41：

【5-41】用 TCP 传送 512 字节的数据。设窗口为 100 字节，而 TCP 报文段每次也是传送 100 字节的数据。再设发送方和接收方的起始序号分别选为 100 和 200，试画出类似于教材图 5-28 的工作示意图。从连接建立阶段到连接释放都要画上（可不考虑传播时延）。

解答：要传送的 512 B 数据必须划分为 6 个报文段传送，前 5 个报文段各 100 B，最后一个报文段传送 12 B。图 T-5-41 是双方交互的示意图。下面进行简单的解释。

报文段#1：A 发起主动打开，发送 SYN 报文段，处于 SYN-SENT 状态，并选择初始序号 seq = 100。B 处于 LISTEN 状态。

报文段#2：B 确认 A 的 SYN 报文段，因此 ack = 101（是 A 的初始序号加 1）。B 选择初始序号 seq = 200。B 进入到 SYN-RCVD 状态。

报文段#3：A 发送 ACK 报文段来确认报文段#2，ack = 201（是 B 的初始序号加 1）。A 没有在这个报文段中放入数据。因为 SYN 报文段#1 消耗了一个序号，因此报文段#3 的序号是 seq = 101。这样，A 和 B 都进入了 ESTABLISHED 状态。

报文段#4：A 发送 100 字节的数据。报文段#3 是确认报文段，没有数据发送，报文段#3 并不消耗序号，因此报文段#4 的序号仍然是 seq = 101。A 在发送数据的同时，还确认 B 的报文段#2，因此 ack = 201。

报文段#5：B 确认 A 的报文段#4。由于收到了从序号 101 到 200 共 100 字节的数据，因此在报文段#5 中，ack = 201（所期望收到的下一个数据字节的序号）。B 发送的 SYN 报文段#2 消耗了一个序号，因此报文段#5 的序号是 seq = 201，比报文段#2 的序号多了一个序号。在这个报文段中，B 给出了接收窗口 rwnd = 100。

从报文段#6 到报文段#13 都不需要更多的解释。到此为止，A 已经传送了 500 字节的数据。值得注意的是，B 发送的所有确认报文段都不消耗序号，其序号都是 seq = 201。

报文段#14：A 发送最后 12 字节的数据，报文段#14 的序号是 seq = 601。

报文段#15：B 发送对报文段#14 的确认。B 收到从序号 601 到 612 共 12 字节的数据。因此报文段#15 的确认号是 ack = 613（所期望收到的下一个数据字节的序号）。

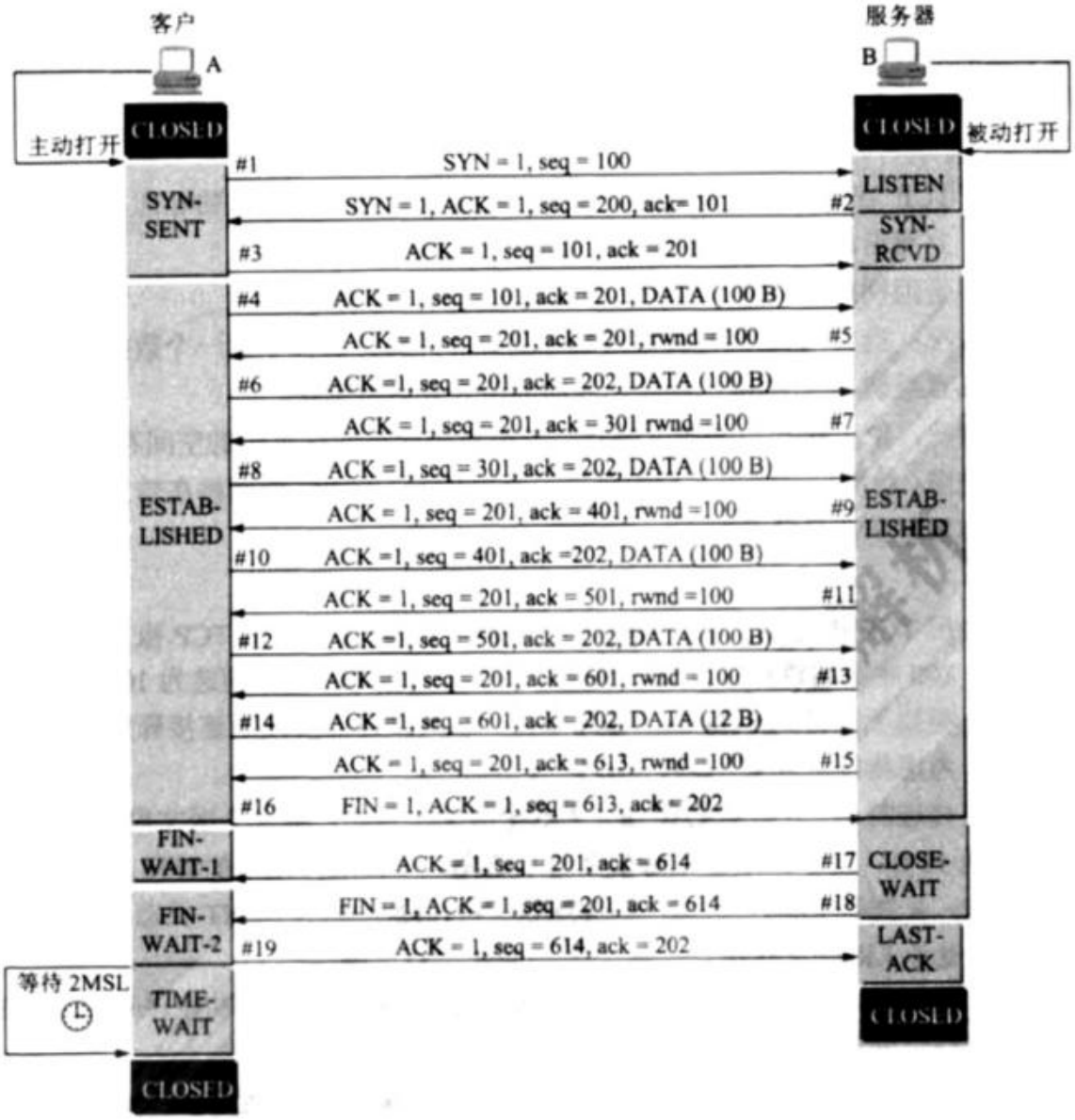


图 T-5-41　从连接建立到连接释放的全过程

需要注意的是，从报文段#5 一直到报文段#15，B 一共发送了 6 个确认，都不消耗序号，因此 B 发送的报文段#15 的序号仍然和报文段#5 的序号一样，即 seq = 201。

报文段#16：A 发送 FIN 报文段。前面所发送的数据报文段#14 已经用掉了序号 601 到 612，因此报文段#16 的序号是 seq = 613。A 进入 FIN-WAIT-1 状态。报文段#16 的确认号 ack = 202。

报文段#17：B 发送确认报文段，确认号 ack = 614，进入 CLOSE-WAIT 状态。由于确认报文段不消耗序号，因此报文段#17 的序号仍然和报文段#15 的一样，即 seq = 201。

报文段#18：B 没有数据要发送，就发送 FIN 报文段#18，其序号仍然是 seq = 201。这个 FIN 报文段会消耗一个序号。

报文段#19：A 发送最后的确认报文段。报文段#16 的序号是 613，已经消耗掉了。因此现在的序号是 seq = 614。但这个确认报文段并不消耗序号。

其他的地方不再需要更多的解释了。

5-46：

【5-46】 试用具体例子说明为什么在运输连接建立时要使用三报文握手。说明如不这样做可能会出现什么情况。

解答： 图 T-5-46 给出了一个例子。

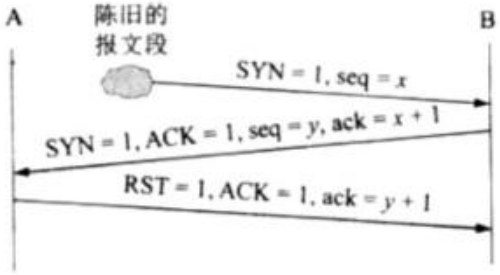


图 T-5-46　陈旧的 SYN 报文段的出现

在上一个 TCP 连接中，A 向 B 发送的连接请求 SYN 报文段滞留在网络中的某处。于是 A 超时重传，与 B 建立了 TCP 连接，交换了数据，最后也释放了 TCP 连接。

但滞留在网络中某处的陈旧的 SYN 报文段，现在突然传送到 B 了。如果不使用三报文握手，那么 B 就以为 A 现在请求建立 TCP 连接，于是就分配资源，等待 A 传送数据。但 A 并没有想要建立 TCP 连接，也不会向 B 传送数据。B 就白白等待着 A 发送数据。

如果使用三报文握手，那么 B 在收到 A 发送的陈旧的 SYN 报文段后，就向 A 发送 SYN 报文段，选择自己的序号 seq = y，并确认收到 A 的 SYN 报文段，其确认号 ack = $x + 1$。当 A 收到 B 的 SYN 报文段时，从确认号就可得知不应当理睬这个 SYN 报文段（因为 A 现在并没有发送 seq = x 的 SYN 报文段）。这时，A 发送复位报文段。在这个报文段中，RST = 1，ACK = 1，其确认号 ack = $y + 1$。我们注意到，虽然 A 拒绝了 TCP 连接的建立（发送了复位报文段），但对 B 发送的 SYN 报文段还是确认收到了。

B 收到 A 的 RST 报文段后，就知道不能建立 TCP 连接，不会等待 A 发送数据了。

6-02：

【6-02】域名系统的主要功能是什么？域名系统中的本地域名服务器、根域名服务器、顶级域名服务器以及权限域名服务器有何区别？

解答：域名系统 DNS 是互联网使用的命名系统，用来把便于人们使用的机器名字转换为 IP 地址。在域名系统中使用了层次结构的许多域名服务器。

本地域名服务器离用户较近，一般不超过几个路由器的距离。当一个主机发出 DNS 查询请求时，这个查询请求报文就发送给本地域名服务器。当所要查询的主机也属于同一个本地 ISP 时，该本地域名服务器立即就能将所查询的主机名转换为它的 IP 地址，而不需要再去询问其他的域名服务器。

根域名服务器是最高层次的域名服务器，也是最重要的域名服务器。所有的根域名服务器都知道所有的顶级域名服务器的域名和 IP 地址。根域名服务器是最重要的域名服务器，因为不管是哪一个本地域名服务器，若要对互联网上任何一个域名进行解析（即转换为 IP 地址），只要自己无法解析，就首先要求助于根域名服务器。

顶级域名服务器负责管理在该顶级域名服务器注册的所有二级域名。当收到 DNS 查询请求时，就给出相应的回答（可能是最后的结果，也可能是下一步应当找的域名服务器的 IP 地址）。

一个服务器所负责管辖的（或有权限的）范围叫作区。各单位根据具体情况来划分自己管辖范围的区。但在一个区中的所有节点必须是能够连通的。每一个区设置相应的权限域名服务器，用来保存该区中的所有主机的域名到 IP 地址的映射。因此，权限域名服务器是负责一个区的域名服务器。当一个权限域名服务器还不能给出最后的查询回答时，就会告诉发出查询请求的 DNS 客户，下一步应当找哪一个权限域名服务器。

6-03：

【6-03】举例说明域名转换的过程。域名服务器中的高速缓存的作用是什么？

解答：域名到 IP 地址的解析过程的要点如下：当某一个应用进程需要把主机名解析为 IP 地址时，该应用进程就调用解析程序，并成为 DNS 的一个客户，把待解析的域名放在 DNS 请求报文中，以 UDP 用户数据报方式发给本地域名服务器（使用 UDP 是为了减少开销）。本地域名服务器在查找域名后，把对应的 IP 地址放在回答报文中返回。应用进程获得目的主机的 IP 地址后即可进行通信。

若本地域名服务器不能回答该请求，则此域名服务器就暂时成为 DNS 中的另一个客户，并向其他域名服务器发出查询请求。这种过程直至找到能够回答该请求的域名服务器为止。

为了提高域名服务器的可靠性，DNS 域名服务器都把数据复制到几个域名服务器来保存，其中的一个是主域名服务器，其他的是辅助域名服务器。当主域名服务器出故障时，辅助域名服务器可以保证 DNS 的查询工作不会中断。主域名服务器定期把数据复制到辅助域名服务器中，而更改数据只能在主域名服务器中进行。这样就保证了数据的一致性。

主机向本地域名服务器的查询一般都采用递归查询。本地域名服务器向根域名服务器的查询通常采用迭代查询。根域名服务器通常是把自己知道的顶级域名服务器的 IP 地址告诉本地域名服务器，让本地域名服务器再向顶级域名服务器查询。顶级域名服务器在收到本地域名服务器的查询请求后，要么给出所要查询的 IP 地址，要么告诉本地域名服务器下一步应当向哪一个权限域名服务器进行查询，本地域名服务器就这样进行迭代查询。最后，知道了所要解析的域名的 IP 地址，然后把这个结果返回给发起查询的主机。当然，本地域名服务器也可以采用递归查询，这取决于最初的查询请求报文的设置是要求使用哪一种查询方式。递归查询返回的查询结果或者是所要查询的 IP 地址，或者是报错，表示无法查询到所需的 IP 地址。

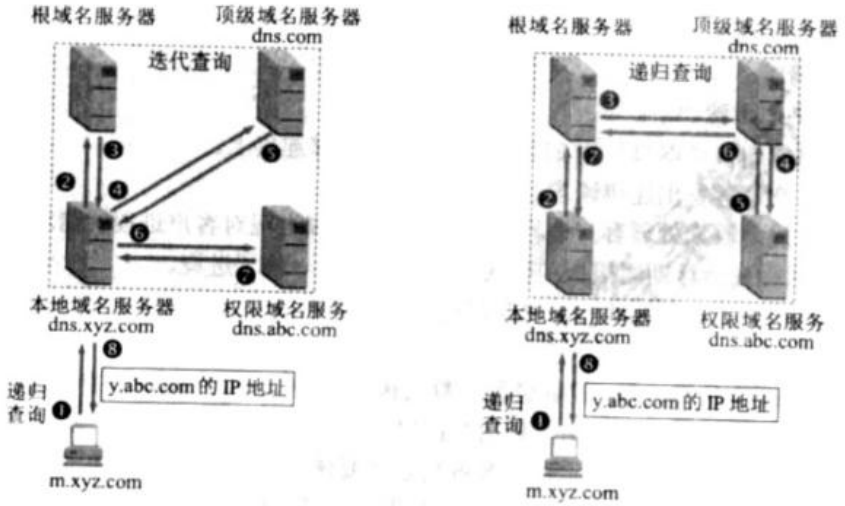
图 T-6-03 给出了两种查询的区别。

假定域名为 m.xyz.com 的主机想知道另一台主机（域名为 y.abc.com）的 IP 地址。例如，主机 m.xyz.com 打算发送邮件给主机 y.abc.com，这时就必须知道主机 y.abc.com 的 IP 地址。下面是图 T-6-03(a)的几个查询步骤：

❶ 主机 m.xyz.com 先向其本地域名服务器 dns.xyz.com 进行递归查询。

❷ 本地域名服务器采用迭代查询。它先向一台根域名服务器查询。

❸ 根域名服务器告诉本地域名服务器，下一次应查询的顶级域名服务器 dns.com 的 IP 地址。



(a) 本地域名服务器采用迭代查询　　(b) 本地域名服务器采用递归查询

图 T-6-03　DNS 查询举例

❹ 本地域名服务器向顶级域名服务器 dns.com 进行查询。

❺ 顶级域名服务器 dns.com 告诉本地域名服务器，下一次应查询的权限域名服务器 dns.abc.com 的 IP 地址。

❻ 本地域名服务器向权限域名服务器 dns.abc.com 进行查询。

❼ 权限域名服务器 dns.abc.com 告诉本地域名服务器，所查询的主机的 IP 地址。

❽ 本地域名服务器最后把查询结果告诉主机 m.xyz.com。

我们注意到，这 8 个步骤总共要使用 8 个 UDP 用户数据报的报文。本地域名服务器经过三次迭代查询后，从权限域名服务器 dns.abc.com 得到了主机 y.abc.com 的 IP 地址，最后把结果返回给发起查询的主机 m.xyz.com。

图 T-6-03(b)是本地域名服务器采用递归查询的情况。在这种情况下，本地域名服务器只需向根域名服务器查询一次，后面的几次查询都是在其他几个域名服务器之间进行的（步骤❸至❻）。只是在步骤❼，本地域名服务器从根域名服务器得到了所需的 IP 地址。最后在步骤❽，本地域名服务器把查询结果告诉主机 m.xyz.com。整个的查询也是使用 8 个 UDP 报文。

为了提高 DNS 查询效率，并减轻根域名服务器的负荷和减少互联网上的 DNS 查询报文数量，在域名服务器中广泛地使用了**高速缓存**（有时也称为高速缓存域名服务器）。高速缓存用来存放最近查询过的域名，以及从何处获得域名映射信息的记录。

6-08：

【6-08】解释以下名词。各英文缩写词的原文是什么？

WWW，URL，HTTP，HTML，CGI，浏览器，超文本，超媒体，超链，页面，活动文档，搜索引擎。

解答：

WWW (World Wide Web)是万维网的英文缩写词。万维网并非某种特殊的计算机网络，它是一个大规模的、联机式的信息储藏所，英文简称为 Web。万维网用链接的方法能非常方便地从互联网上的一个站点访问另一个站点（也就是所谓的“链接到另一个站点”），从而可以获取丰富的信息。

URL (Uniform Resource Locator)是统一资源定位符的英文缩写词。万维网使用 URL 来标志万维网上的各种文档，并使每一个文档在整个互联网的范围内具有唯一的标识符 URL。

HTTP (HyperText Transfer Protocol)是超文本传送协议的英文缩写词。HTTP 是万维网客户程序与万维网服务器程序之间进行交互时必须遵守的协议。

HTML (HyperText Markup Language)是超文本标记语言的英文缩写词。它使得万维网页面的设计者，可以很方便地使用链接，从本页面的某处链接到互联网上的任何一个万维网页面，并且能够在自己的主机屏幕上将这些页面显示出来。

CGI (Common Gateway Interface)是通用网关接口的英文缩写词。CGI 是一种标准，它定义了动态文档应如何创建，输入数据应如何提供给应用程序，以及输出结果应如何使用。

浏览器是在用户主机上的万维网客户程序。万维网文档所驻留的主机则运行服务器程序。客户程序向服务器程序发出请求，服务器程序向客户程序送回客户所要的万维网文档。

超文本是包含指向其他文档的链接的文本。也就是说，一个超文本由多个信息源链接而成，这些信息源的数目实际上是不受限制的。利用一个链接可使用户找到另一个文档，而这又可链接到其他的文档（依此类推）。这些文档可以位于世界上任何一个连接在互联网上的超文本系统中。超文本是万维网的基础。

超媒体与超文本的区别是文档内容不同。超文本文档仅包含文本信息，而超媒体文档还包含其他表示方式的信息，如图形、图像、声音、动画，甚至活动视频图像。

超链(hyperlink)就是一个超文本的链接，有时也就简称为链接。在客户程序的主窗口中，超链通常用不同颜色的文字表示，有时在超链的文字下方添加了下画线。当我们把鼠标移动到有超链的地方时，鼠标的箭头就变成了一只手的形状。

页面(page)就是在一个客户程序主窗口上显示出的万维网文档。

活动文档(active document)技术是把所有的工作都转移给浏览器端。每当浏览器请求一个活动文档时，服务器就返回一段活动文档程序副本，使该程序副本在浏览器端运行。活动文档程序可与用户直接交互，并可连续地改变屏幕的显示。只要用户运行活动文档程序，活动文档的内容就可以连续地改变。由于活动文档技术不需要服务器的连续更新传送，对网络带宽的要求也不会太高。从传送的角度看，浏览器和服务器都把活动文档看成是静态文档。在服务器上的活动文档的内容是不变的，这点和动态文档是不同的。浏览器可在本地缓存一份活动文档的副本。活动文档还可处理成压缩形式，便于存储和传送。活动文档本身并不包括其运行所需的全部软件，大部分的支持软件是事先存放在浏览器中的。

搜索引擎(search engine)是万维网中用来进行搜索信息的工具。

6-12：

【6-12】什么是动态文档？试举出万维网使用动态文档的一些例子。

解答：动态文档是指文档的内容是在浏览器访问万维网服务器时，才由应用程序动态创建的。当浏览器请求到达时，万维网服务器要运行另一个应用程序，并把控制转移到此应用程序。

接着，该应用程序对浏览器发来的数据进行处理，并输出 HTTP 格式的文档，万维网服务器把应用程序的输出作为对浏览器的响应。由于对浏览器每次请求的响应都是临时生成的，因此用户通过动态文档所看到的内容是不断变化的。动态文档的主要优点是具有报告当前最新信息的能力。例如，动态文档可用来报告股市行情、天气预报或民航售票情况等内容。但动态文档的创建难度比静态文档的高，因为动态文档的开发不是直接编写文档本身，而是编写用于生成文档的应用程序，这就要求动态文档的开发人员必须会编程，而所编写的程序还要通过大范围的测试，以保证输入的有效性。

动态文档的一个例子是我们使用携程网(www.ctrip.com)购买机票。当我们打开携程网时，便看到了携程网服务器的网页，这就是一种活动文档。例如，我们选择国内机票，当我们键入出发地点、到达地点、日期、人数、舱位等信息后，携程网的服务器就可以根据你键入的数据，生成出符合你需要的各航空公司的航班表。这种航班表不是静态的，而是根据用户的需求动态生成的。当某个航班的机票已经售完时，动态航班表也会显示出某个航班的机票已经无法购买了。

6-16：

【6-16】 在上题中，假定同一台服务器的 HTML 文件中又链接了三个非常小的对象。若忽略这些对象的发送时间，试计算客户点击读取这些对象所需的时间。

(1) 没有并行 TCP 连接的非持续 HTTP；

(2) 使用并行 TCP 连接的非持续 HTTP；

(3) 流水线方式的持续 HTTP。

解答：分别计算如下：

(1) 所需时间 $= RTT_1 + RTT_2 + \ldots + RTT_n$ （解析 IP 地址）

$+ 2RTT_W$ （建立 TCP 连接和读取 HTML 文件）

$+ 3(2RTT_W)$ （读取三个对象）

$= RTT_1 + RTT_2 + \ldots + RTT_n + 8RTT_W$

$8RTT_W$ 的图解如图 T-6-16(a)所示。

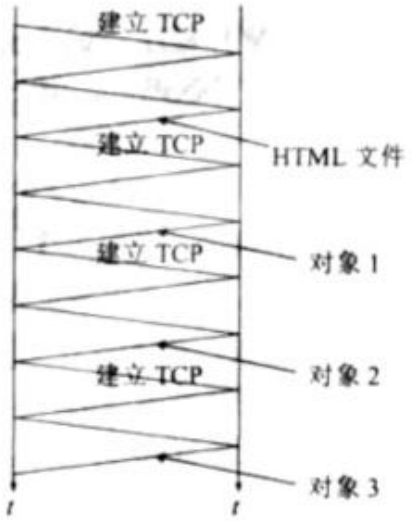


图 T-6-16(a) $8RTT_W$ 的图解

(2) 所需时间 $= RTT_1 + RTT_2 + \ldots + RTT_n$ （解析 IP 地址）

$+ 2RTT_W$ （建立 TCP 连接和读取 HTML 文件）

$+ 2RTT_W$ （并行地建立 TCP 连接和并行地读取三个对象）

$= RTT_1 + RTT_2 + \ldots + RTT_n + 4RTT_W$

$4RTT_W$ 的图解如图 T-6-16(b)所示。

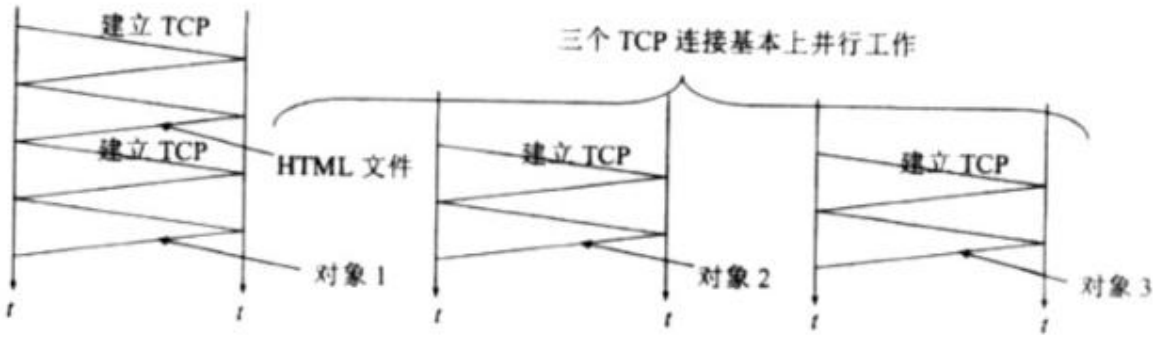


图 T-6-16(b)　$4RTT_W$ 的图解

(3) 所需时间 $= RTT_1 + RTT_2 + \ldots + RTT_n$ （解析 IP 地址）

$+ 2RTT_W$ （建立 TCP 连接和读取 HTML 文件）

$+ RTT_W$ （连续读取三个对象）

$= RTT_1 + RTT_2 + \ldots + RTT_n + 3RTT_W$

$3RTT_W$ 的图解如图 T-6-16(c)所示。

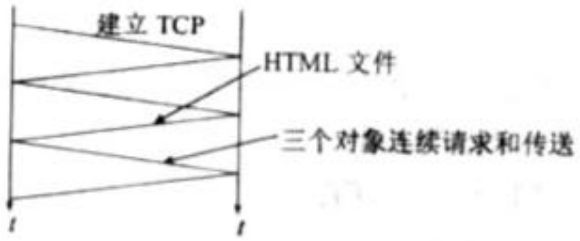


图 T-6-16(c)　$3RTT_W$ 的图解

6-20：

【6-20】 试述电子邮件的最主要的组成部件。用户代理 UA 的作用是什么？没有 UA 行不行？

解答：一个电子邮件系统应具备三个主要组成构件，这就是用户代理、邮件服务器，以及邮件发送协议（如 SMTP）和邮件读取协议（如 POP3）。

用户代理 UA (User Agent)就是用户与电子邮件系统的接口，在大多数情况下它就是运行在用户 PC 中的一个程序。因此用户代理又称为电子邮件客户端软件。用户代理向用户提供一个很友好的接口（目前主要是用窗口界面）来发送和接收邮件。具体来讲，用户代理至少应当具有以下四个功能。

(1) 撰写：给用户提供编辑信件的环境。

(2) 显示：能方便地在计算机屏幕上显示出来信和去信。

(3) 处理：包括发送邮件和接收邮件。

(4) 通信：发信人在撰写完邮件后，要利用邮件发送协议发送到用户所使用的邮件服务器。收件人在接收邮件时，要使用邮件读取协议从本地邮件服务器接收邮件。

如果没有用户代理 UA，那么对于要使用电子邮件的用户就很不方便了。因为上面所说的 UA 的功能，就要统统由用户自己来编程实现。如果用户不会计算机编程，那么他就无法使用电子邮件。即使用户会计算机编程，那么这也将耗费他很长的时间，使他不愿意使用这样的电子邮件。因此，用户代理 UA 对电子邮件用户来说是必不可少的。