附录A　测量网络时间

本书正文中用到了分组经过网络传输时传输时间的测量。本附录给出其细节和我们能够测量的各种时间的测量例子。我们要介绍用 Ping程序实现的RTT测量，向上和向下经过协议栈的时间测量，以及等待时间与带宽的差异。

网络程序员或系统管理员通常有两种办法可以用来测量应用事务所需的时间：

1) 采用应用程序定时器。例如，在图 1-1的UDP客户程序中，我们在调用 `sendto`之前取到了系统时钟，在`recvfrom`函数返回后又取到了系统时钟，其差额就是应用程序发送请求至收到应答的时间。

   如果内核提供了高精度的时钟(ms数量级)，则我们所测得的值(几毫秒或以上)就很精确。卷1的附录A给出了这一类测量方法的更多细节。

2) 采用软件工具来监视指定分组，并计算相应的时间差，如嵌入到数据链路层的Tcpdump。在卷1的附录A中有这些工具的更多细节。

   在这本书中，我们假定数据链路的嵌入点在 Tcpdump中用BSD分组过滤器(BPF)提供。卷2的第31章给出了BPF实现的许多细节。卷2图4-11和图4-19说明了在典型以太网驱动程序中哪里要有BPF调用，图15-27则说明了在环路测试驱动程序中的 BPF调用。

我们注意到本书中的例子用到的系统 (图1-13)，包括80386上的BSD/OS 2.0和Sparcstation ELC上的Solaris 2.4，都给应用程序计时和Tcpdump时间戳提供高精度的定时器。

最可靠的方法是在网络电缆上连接一个网络分析仪，但往往没有这样的仪器。

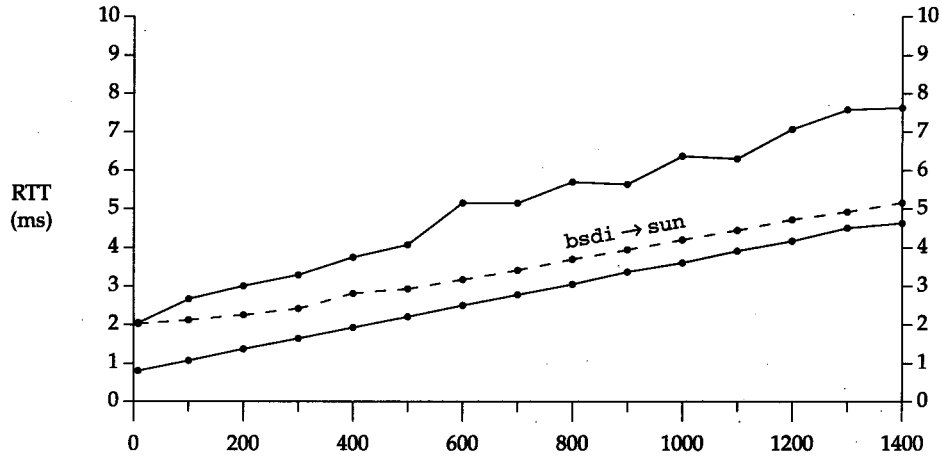
A.1　利用Ping的RTT测量

在卷1的第7章详细介绍了无所不在的 Ping程序，利用应用定时器来计算 ICMP分组的RTT(往返时间)。程序发送一个ICMP回显请求分组给服务器，服务器紧接着向客户回复一个回显应答分组。客户可以在回显请求分组中将发送时的时钟值作为用户可选数据记录在该分组中，然后服务器会在应答中返回这个时钟值。客户收到回显应答时，它就取当前时钟值计算出RTT，然后打印出来。图A-1给出了Ping分组的格式。

图A-1　Ping分组：ICMP回显请求或ICMP回显应答

Ping程序允许我们指定分组中可选用户数据的长度，使我们能够测量分组长度对 RTT的影响。如果是用Ping来测量RTT，可选数据的长度必须至少8字节(客户发出和服务器应答的时间戳要占用8个字节)。如果指定的用户数据长度少于8字节，Ping也能工作，但不能计算并打印RTT。

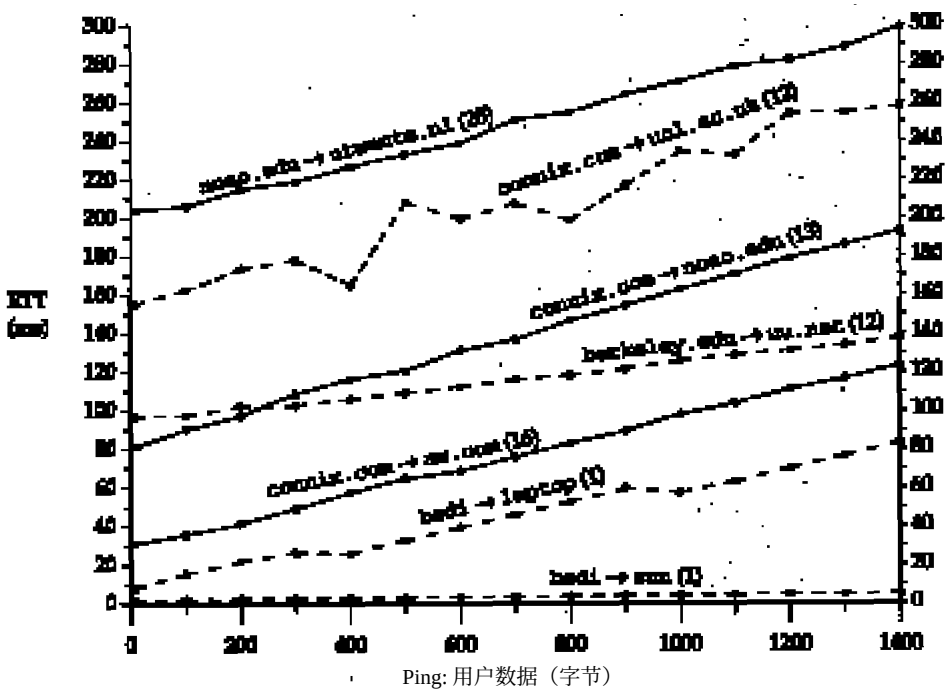
图A-2画出了在三个不同局域网上的主机间用 Ping测得的RTT典型值。图中间的一条线是图1-13中主机`bsdi`和`sun`间的RTT。



图A-2 三个以太网上的主机间Ping RTT值

用15个不同的分组长度进行了测量：8字节用户数据以及从100至1400字节的用户数据(以100字节递增)。加上20字节的IP首部和8字节的ICMP首部，IP数据报的长度就在36~1428字节之间。对每一个分组长度都进行了10次测量，图中只画出了10个值中最小的那个。与我们所期望的一致，分组长度增加后RTT也增大。三条线之间的差别是因处理器速度、接口卡和操作系统的不同而造成的。

图A-3给出了经Internet、WAN互连的各种主机之间典型的RTT值。注意*y*轴的刻度与图A-2中的有差别。



图A-3 经Internet(一个WAN)互连的主机间Ping RTT值

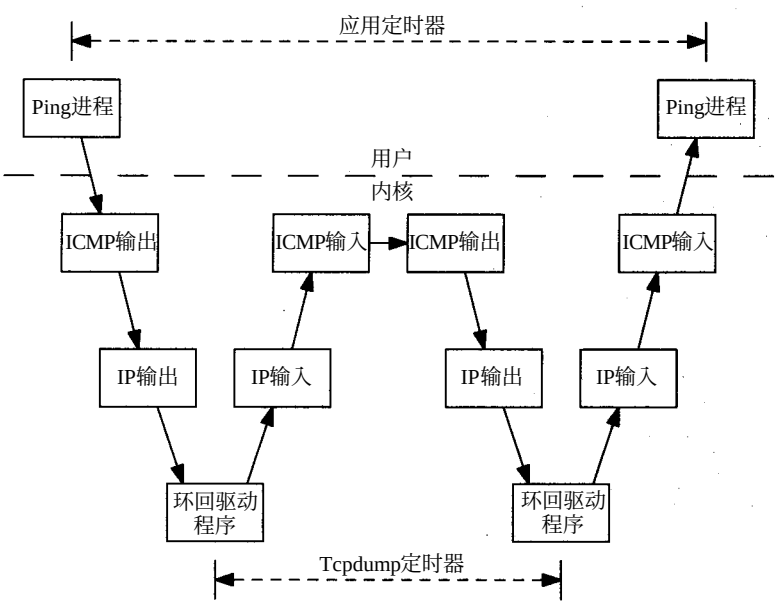
与在LAN上测量那样，对WAN进行了同样的测量：对15个不同长度的分组各进行了10次测量，每个分组长度只画出了10个值中最小的那个值。图中的括号中是每对主机之间的转发段数。

图中最上边的曲线（最长的RTT）表示Internet上分别位于Arizona(noao.edu)和Netherlands(utwente.n1)的一对主机之间需要25段转发。自上而下的第2条曲线也是跨越大西洋的，是Connecticut(connix.com)和London(ucl.ac.uk)之间的一对主机。接下来两条曲线在美国内部，分别是Connecticut和Arizona之间的一对主机(connix.com与noao.edu)，以及California和Washington D.C.之间的一对主机(berkeley.edu和uu.net)。再接下来的曲线是地理上很近的一对主机(Connecticut的connix.com和Boston的aw.com)，但从经过Internet传送的转发段数(16)来衡量，却是离得很远的。

图中底部的两条线(RTT值最小的)是作者所在局域网(图1-13)上的主机之间的。其中最底下的那条线是从图A-2复制来的，以便对典型的LAN上的RTT与典型的WAN上的RTT进行比较。在最底下的第2条线，即bsdi和laptop之间的RTT线，后者的以太网卡是插在计算机的并行口上的。尽管该系统也是接在以太网上的，但由于并行口的传输速率较慢，看上去就像是接在WAN上一样。

A.2 协议栈测量

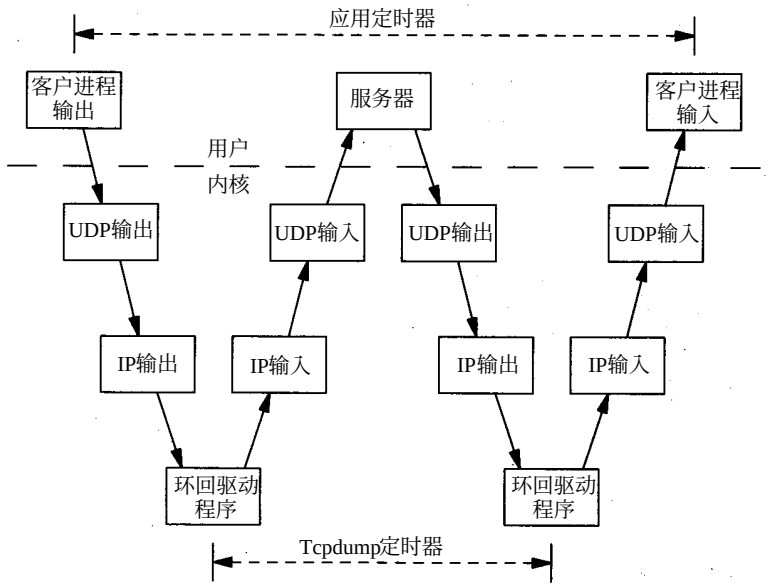
我们也可以用Ping，并加上Tcpdump来测量在协议栈上花费的时间。例如，图A-4中就给出了在一台主机上运行Ping和Tcpdump，对环回测试地址(一般是127.0.0.1)，Ping的执行步骤。



图A-4 在一台主机上运行Ping和Tcpdump

假设应用程序在就要向操作系统发出回显请求分组时启动定时器，然后在操作系统返回回显应答时停掉定时器，应用程序测得的时间差和Tcpdump测得的时间差就分别是ICMP输出、IP输出、IP输入和ICMP输入之间所需的时间。

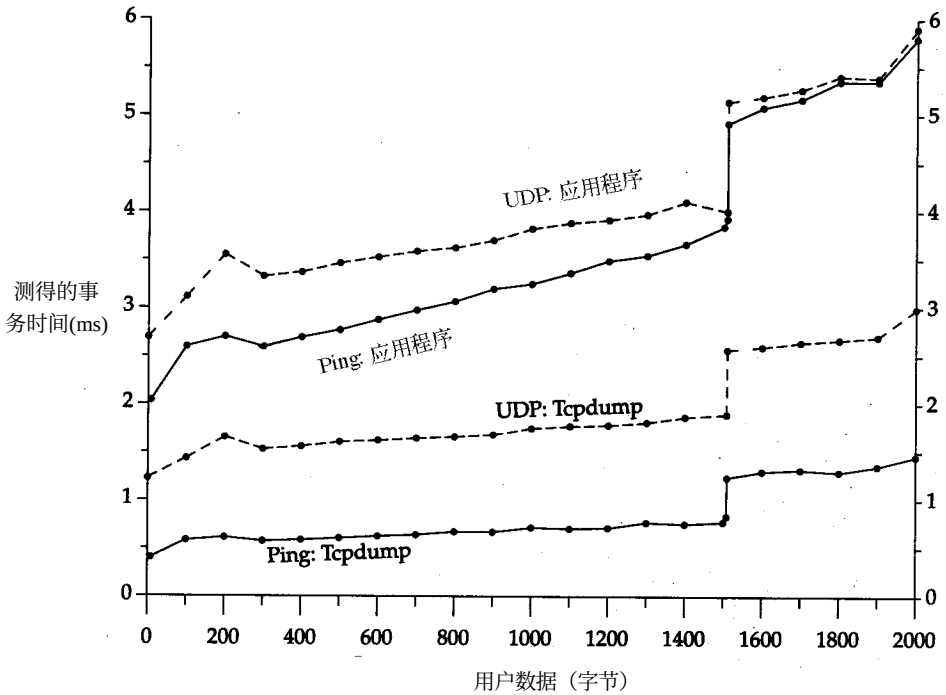
我们也可以测量任何客户—服务器应用程序之间的类似值。图A-5给出了1.2节UDP客户—服务器应用的处理步骤，其中假设客户和服务器在同一台主机上。



图A-5 UDP客户-服务器事务的处理步骤

这个UDP的客户-服务器例子与图A-4的Ping例子之间的一个不同之处是，这里的UDP服务器是一个用户进程，而Ping服务器则是ICMP内核的一部分(卷2图11-21)。因此，UDP服务器中在内核和用户进程之间要有两份客户数据：服务器输入和服务器输出。内核与用户进程之间复制数据通常都是比较费时的操作。

图A-6给出了在主机bsdi上进行的各项测试结果，可以比较Ping客户-服务器和UDP客



图A-6 单个主机上(环回接口)的Ping和Tcpdump测量结果

户-服务器这两种方式。图中y轴标的是“测得的事务时间”，因为RTT通常都是指网络的往返时间或Ping的时间输出(在图A-8中可以看到，它与网络的RTT非常接近)。在这里的UDP、TCP和T/TCP客户-服务器方式中，可以测量应用程序的事务时间。在TCP和T/TCP的例子中，这可能要包括多个分组和多次网络RTT。

在这个图的Ping测量中采用了23种不同的分组长度：用户数据从100字节到2 000字节变化，增量为100字节，再加上8、1508和1509字节。其中8字节是用Ping来测量RTT的最短用户数据长度，1508是不会在IP层分段的最大数据长度，因为BSD/OS采用了1536的MTU作为环测接口(1508+20+8)。1509字节则是会在IP层进行分段的最小数据长度。

在UDP测量中也采用了23种类似长度的分组：用户数据长度从100字节到2 000字节变化(增量100)，再加上0、1508和1509。0字节的UDP数据报也是允许的。由于UDP的首部与ICMP回显测试分组的首部一样长(8字节)，1508又是避免在环测接口上分段的最大分组，1 509是需要分段的最小分组。

我们首先注意到的是在用户数据为1509字节时的时间跳变，这时需要分段。这也是想像之中的。当出现分段时，在图A-4和图A-5中左边对“IP输出”的一次调用会产生两次对“环测驱动程序”的调用，每段一次。从1508到1509，即使用户数据只增加了一个字节，应用程序就感觉到事务时间增加了近25%，因为多出一个每分组处理时间。

所有4条线中，在200字节点的时间增加是由于BSD的mbuf实现中的非自然处理造成的(卷2的第2章)。对于最小分组(UDP测量中的0字节用户数据和Ping测量中的8字节用户数据)，数据和分组的首部可以写入一个mbuf中，在100字节点需要第二个mbuf，在200字节点则需要第三个mbuf。最后，在300字节点，内核开始采用2048字节的mbuf簇来代替较小的mbuf。看起来，用一个mbuf簇比用多个mbuf会快一些(例如，在100字节点)，可以减少处理时间。这是典型的时间—空间折衷的例子。从采用较小的mbuf到采用较大的mbuf簇的切换条件是数据量是否超过208字节，这是在许多年前当内存还很紧张时设计的。

图1-14中的定时测量是用修改后的BSD/OS内核实现的，其中的常数MINCLSIZE(卷2图2-7和图16-25)从208改为101。这样就使得一旦用户数据超过100字节就分配mbuf簇。只要注意就可以看到，图1-14中没有在200字节点出现尖角。

我们在14-11节也讨论过这个问题，在那里我们看到，许多Web客户请求都在100~200字节之间。

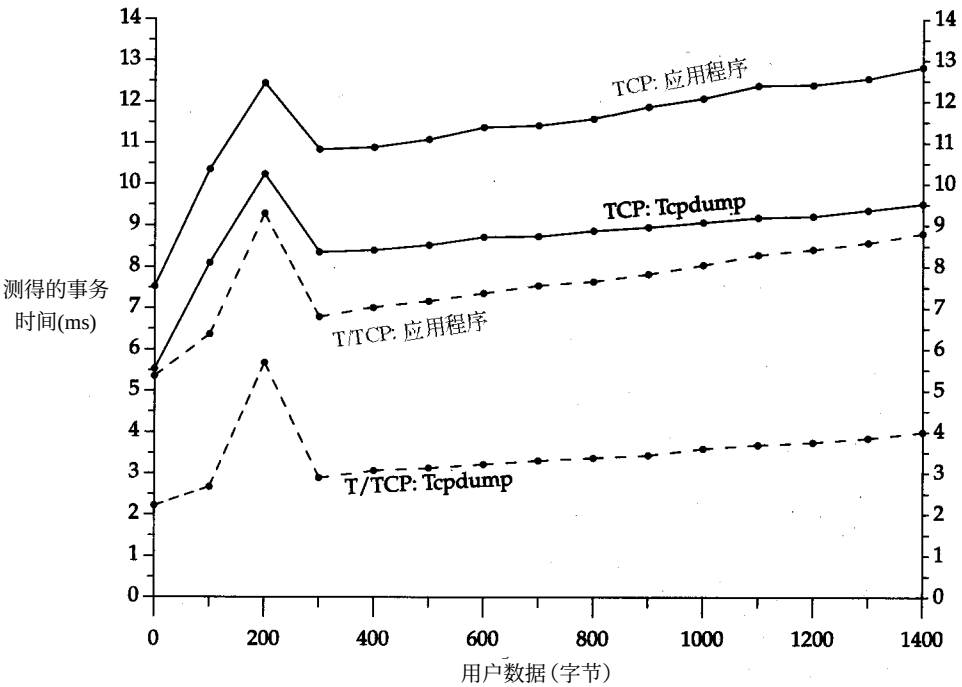
图A-6中，开始分段之前，两条UDP曲线之间的相差在1.5~2 ms之间。因为这个差额已经把UDP输出、IP输出、IP输入和UDP输入(图A-5)考虑在内，如果我们假设协议的输出逼近于协议输入，那么就相当于分组发送时向下穿过协议栈要花不到1ms的时间，接收时分组向上穿过协议栈又要花不到1 ms的时间。这些时间包括了发送时要将多份数据从进程传递给内核，以及数据返回时从内核传递到进程。

由于图A-5中Tcpdump测量要经历同样的4个步骤(IP输入、UDP输入、UDP输出和IP输出)，我们可以预计到UDP Tcpdump的两条曲线相差也在1.5~2 ms之间(只考虑发生分段前的值)。与第一个数据点不同，图A-6中其余的数据也在1.5~2 ms之间。

如果我们考虑发生了分段以后的值，图A-6中两条UDP曲线之间相差2.5~3 ms。跟预期的一样，UDP Tcpdump的值也在2.5~3 ms之间。

最后可以看到，图A-6中，Ping的Tcpdump曲线几乎是平坦的，但Ping的应用程序测量则有一个正的斜率。这很可能是因为应用程序测量了两份用户进程和内核之间的数据，但Tcpdump则一份也不需要测量(因为Ping服务器是内核的ICMP实现的一部分)。另外，Ping的Tcpdump曲线非常轻微的正斜率很可能是由于内核Ping服务器的两次操作造成的，这些操作对每一个字节都要执行：接收ICMP的检验和验证和输出ICMP的检验和计算。

我们也可以修改1.3节和1.4节的TCP和T/TCP客户-服务器应用，以测量每一次事务的时间(见1.6节的叙述)，并对不同分组长度进行测量。测量结果见图A-7(在本附录余下的事务测量中，我们测到用户数据长度为1400字节就结束了，因为TCP不分段)。



图A-7 单个主机上(环回接口)的TCP和T/TCP客户-服务器事务时间测量结果

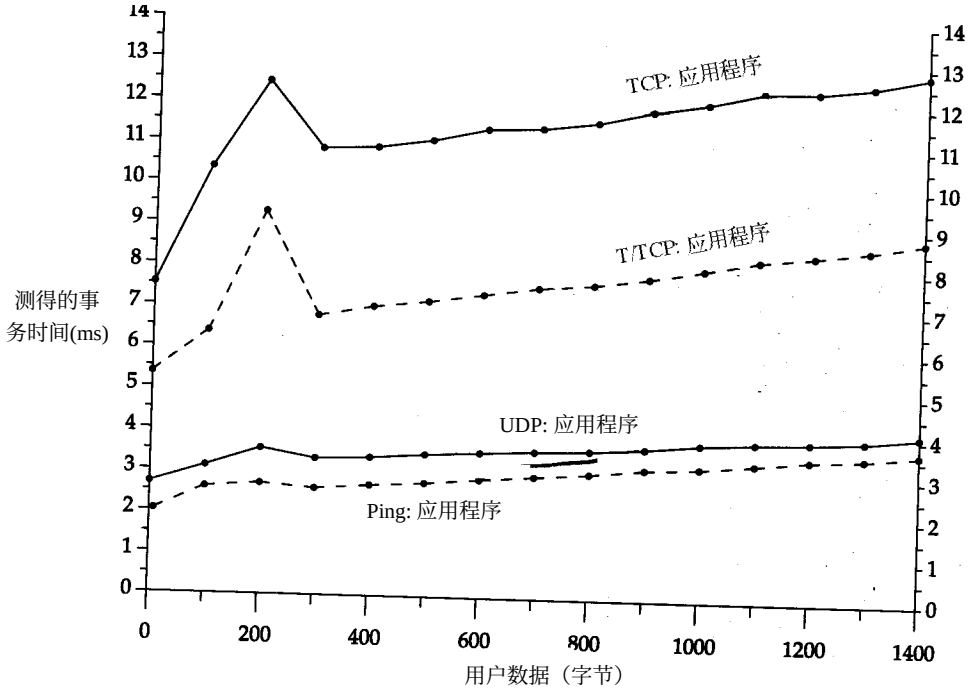
Tcpdump曲线测量的是从客户发出第一个报文段(对TCP客户是一个SYN，对T/TCP客户则是SYN、数据和FIN的组合)至收到服务器发回的最后一个报文段(对TCP客户是一个FIN，对T/TCP客户则是数据和FIN的组合)为止的时间间隔。相应地，TCP应用曲线和TCP Tcpdump曲线之间的差别也就是协议栈用于处理connect和FIN所需的时间(图1-8给出了分组交换)。T/TCP客户-服务器的应用曲线和Tcpdump曲线的差别就是协议栈用于处理客户的sendto所需的时间，其中包括客户数据和最后一个FIN(图1-12给出了分组交换)。我们可以看到，两条T/TCP曲线之间的差距(大约4 ms)大于两条TCP曲线之间的差距(大约2.5 ms)，这是合理的，因为T/TCP的协议栈处理量(在第一段报文中要发送SYN、数据和FIN)比TCP的(第一段报文中只发送SYN)要大。

4条曲线均在200字节处开始上升，再次说明内核应该尽快采用mbuf簇。注意，TCP和T/TCP在200字节处的增加比在图A-6中Ping和UDP的要大得多。对于数据报协议(ICMP和UDP)来说，尽管分配了3个mbuf来缓存首部和用户数据，但内核中插口层对协议输出例程的

调用只有一次(卷2的16.7节称，该调用是sosend函数)。而对流协议(TCP)来说，对TCP输出例程的调用有两次：一次是前 100字节用户数据，另一次是第 2个100字节用户数据。确实，Tcpdump证实了要传送两个 100字节报文段的事实。对协议输出例程多了一次调用就增加了开销。

TCP和T/TCP应用曲线之间的差别大约是 4 ms，对所有分组长度几乎都一样，因为 T/TCP处理的报文段少。图 1-8和图1-12给出了9个TCP报文段和3个TCP报文段。报文段数的减少明显减轻了两端主机的处理开销。

图A-8总结了图A-6和图A-7中的Ping、UDP以及T/TCP和TCP客户 - 服务器的应用时间测量，没有考虑Tcpdump的时间测量。



图A-8　单个主机上(环回接口)的Ping、UDP及TCP和T/TCP客户 - 服务器事务时间测量

结果是预料之中的。Ping所需的时间最少，没有比它更快的了，因为 Ping服务器是在内核中的。UDP事务所需时间略大于 Ping的时间，因为数据要在内核与服务器之间复制两次以上，但并不大，是 UDP所需的最小处理时间。 T/TCP事务所需的时间大约是 UDP的两倍，因为尽管分组数量与UDP相同，但需要更多的协议处理时间 (我们的应用程序定时器并不包括图 1-12中最后的ACK)。TCP的事务时间大约比 T/TCP多50%，因为协议需要处理的分组数较多。图A-8中UDP、T/TCP和TCP之间的相对时间差与图 1-14中的不一样，因为第 1章中的测量是在实际网络上进行的，而本附录中的测量是在环测接口上进行的。

A.3　滞后和带宽

在网络通信中，有两个因素在决定交换信息所需的时间：滞后和带宽 [Bellovin 1992]。这里忽略了服务器处理时间和网络负荷，以及其他明显影响客户事务时间的因素。

滞后(也称为传播时延)是将一个比特从客户传递到服务器再传回来所需的固定时间，受光速的限制，从而决定于两个主机之间电或光信号的传播距离。横跨美国东西两岸之间的事务RTT不会低于60 ms，除非有人可以提高光速。对滞后可做的唯一控制是将两台主机移近，或避免使用高滞后的路径(如卫星链路)。

理论上，光波穿越美国的时间应该是大约16 ms，最小的RTT是32 ms。60 ms是实际的RTT。作为试验，作者曾在分别位于美国东西海岸的主机上运行过Traceroute，只观察横跨美国的直达链路两端的路由器之间的RTT。加州与华盛顿之间的RTT是58 ms，加州与波士顿之间的RTT是80 ms。

另一方面，带宽度量每个比特进入网络的速度，发送方以这个速度顺序将数据送入网络。增加带宽只要购买更快的网络即可。例如，如果T1线路还嫌不够快(大约1 544 000 bit/s)，你可以租用T3线路(大约45 000 000 bit/s)。

可以用公园的软管作恰当的比喻(感谢Ian Lance Taylor)：滞后是水从水龙头流到喷口所需的时间，而带宽就相当于每秒从喷口流出的水量。

一个问题是，网络越来越快(即，带宽增加)，但滞后保持不变。例如，要用横跨美国的T1线路发送100万字节的数据(假设单程滞后是30 ms)，需要5.21秒：5.18秒是带宽决定的，另外0.03秒是滞后造成的。这时带宽是主要影响。但是，如果采用T3线路，则总时间是208 ms：178 ms是带宽决定的，另外30 ms是滞后造成的。这时滞后是带宽时延的1/6。而以150 000 000 b/s发送则需要82 ms：52 ms是带宽决定的，30 ms是滞后造成的。在最后这个例子中，滞后越来越接近带宽时延，而在更快的网络中，滞后就成为主要的时延因素，而不再是带宽。

在图A-3中，往返滞后基本上就是每条曲线与*y*轴的相交点。最上面两条曲线(大约在202ms和155 ms处相交)是美国和欧洲之间的，接下来的两条曲线(在98 ms和80 ms处相交)是横跨整个美国的，再下面这条(大约在30 ms处相交)是美国东海岸的两台主机之间的。

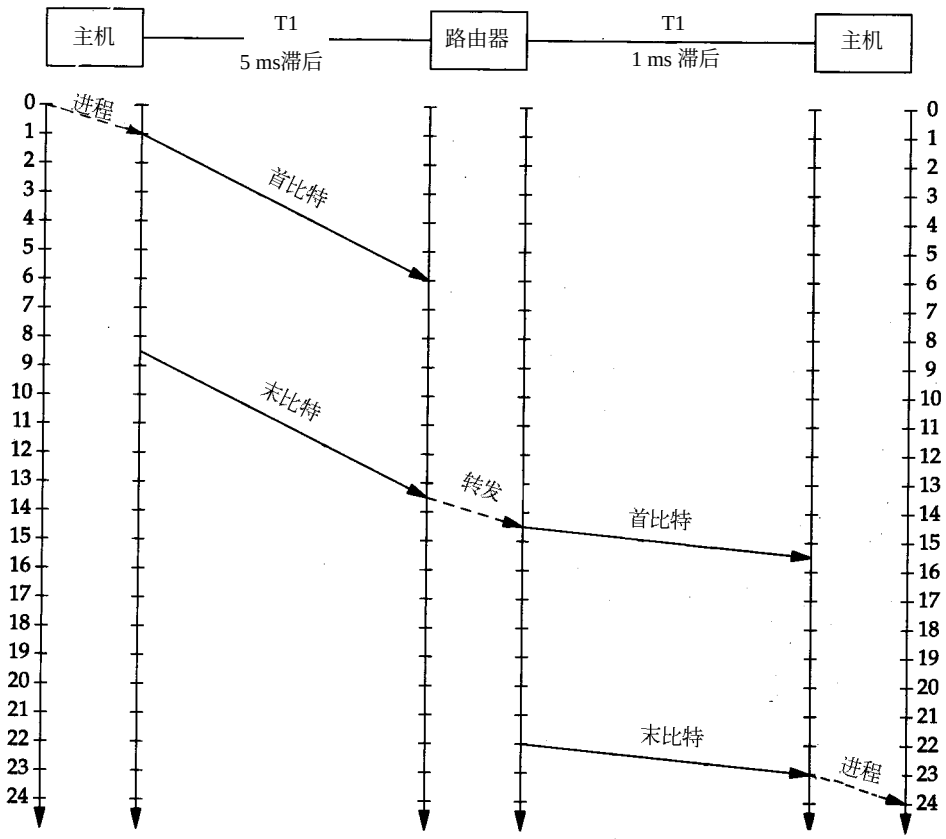
随着带宽增加，滞后变得越来越重要，这使得T/TCP更显优越。T/TCP至少使滞后减少了一个RTT。

顺序发送时延和路由器

如果我们将T1线路租给Internet服务商，用于向另一台以T1线路连接到Internet的主机发送数据，并且已知所有的中间线路都是T1或更高速率的线路，我们会对这样带来的结果感到惊奇。

例如，在图A-3中，如果我们来研究起始点为80 ms、终止点为193 ms的曲线，它是位于Connecticut的`connix.com`主机和位于Arizona的`noao.edu`主机之间的，它与*y*轴相交在80 ms处，正好反映了东西海岸之间的RTT(运行Traceroute程序，这在卷1的第8章有详细介绍，其结果说明分组的确切路由是从Arizona出发，回到California，然后到Texas、Washington DC，最后到Connecticut)。但如果我们计算在T1线路上发送1400字节所需的时间，大约只需要7.5 ms，因此可以估计1400字节分组的RTT应该是95 ms左右，远远低于实际测得的值193 ms。

出现这种情况是因为发送时延与中间路由器的数量成线性关系，因为每个路由器都必须在转发之前接收到整个数据报。考虑图A-9中的例子，要从左边的主机通过中间路由器传送一个1428字节长的分组到右边的主机。假设两条链路都是T1线路，发出1428字节大约需要7.5 ms。图中的时钟是从上向下增长的。



图A-9 数据的顺序发送

第1个箭头，从时刻0到1是主机处理输出数据报，根据本附录前面的测量，假设它需要1 ms。然后这些数据被发送到网络上，从开始发出第 1个比特到最后一个比特发完，需要 7.5 ms。另外在线路两端之间还有 5 ms的滞后，因此第1个比特到达路由器是时刻 6，最后一个比特则是在时刻13.5到达。

只有在时刻13.5最后一个比特到达以后，路由器才能转发该分组，我们假设转发又需要1ms时间。这样，路由器在时刻 14.5发出第1个比特，并且1 ms(第2段链路的滞后)以后到达目的主机。最后一个比特到达目的主机是在时刻 25。最后我们假设目的主机的处理又需要 1 ms。

确切的数据速率是在24 ms内传送了1428字节，或476 000 b/s，比T1的1/3还小。如果我们忽略主机和路由器处理分组的时间共3 ms，数据速率是544 000 b/s。

如前所述，顺序发送时延与分组所经过的路由器数量成线性关系。这项时延决定于线路速率(带宽)、分组长度和中间转发次数(路由器数)。例如，552字节分组(包含512字节数据的典型TCP报文段)在56 kb/s线路上是80 ms，在T1线路上是2.86 ms，而在T3线路上则只要0.10 ms。这样，10段T1线路就要给总时间加上 28.6 ms(几乎等于东西海岸之间的单程滞后)，而10段T3线路只增加1 ms(与滞后相比几乎可以忽略)。

最后，顺序发送时延是一种滞后效应，而不是带宽效应。例如，在图 A-9中，左边的发送主机可以在时刻8.5开始发送下一个分组的第 1比特，而不必等到时刻 24以后才开始发送下一

个分组。如果左边的主机连续发送 10个1428字节分组，假设分组之间没有间隙，则最后一个分组的最后一个比特的到达时刻是91.5(24+9×7.5)。这样的数据速率是1 248 525 b/s，非常接近T1的速率。对TCP来说，只是需要一个比较大的窗口来抵消顺序发送时延。

回到我们前面的例子，从connix.com到noao.edu，如果我们用Traceroute确定了确切的路径，知道了每条链路的速率，就可以把两台主机之间 12个路由器上的顺序发送时延考虑进去。这样，再假设滞后时间 80 ms，每个中间线路段有0.5 ms的处理时延，我们估算的总时延就是187 ms。这已经很接近实测值193 ms，比前面的估算值95 ms要接近得多。