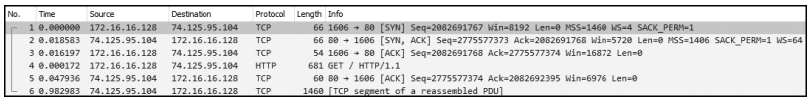
11.4.4 通信缓慢——服务器延迟

我们查看的最后一个延迟情景使用了 latency4.pcap 文件，如图 11-25 所示。这是服务器延迟的一个例子。



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.018583	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.016197	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	0.000172	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.047936	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.982983	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

图 11-25　直到最后一个数据包才表现出高延迟

在这个捕获记录中，两台主机间的 TCP 握手过程很快就顺利完成了，这是个很好的开端。下一对数据包带来了更好的消息，初始 GET 请求和响应的 ACK 数据包也很快传输完毕。直到最后一个数据包，我们才发现了高延迟的迹象。

第 6 个数据包是服务器响应客户端 GET 请求的第一个 HTTP 数据包，但它竟然在服务器为 GET 请求发送 TCP ACK 之后延迟了 0.98s。数据包 5 到 6 的切换情况与我们在上一个情景中看见的握手 ACK 与 GET 请求之间的切换很相似。然而，在这个例子中，服务器才是我们关注的焦点。

数据包 5 是服务器响应客户端 GET 请求的 ACK。一旦发送这个数据包，服务器就应该立即开始发送数据。这个数据包中的数据访问、打包和传输是由 HTTP 协议完成的，由于这是一个应用层协议，因此需要服务器作一些处理。延迟收到这个数据包表明服务器不能及时处理这个数据，最终把延迟的根源指向了它。