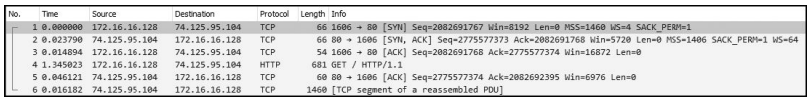
11.4.3 通信缓慢——客户端延迟

我们查看的下一个延迟情景包含在 latency3.pcap 文件中，如图 11-24 所示。



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	0.023790	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	0.014894	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	1.345023	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.046121	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.016182	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

图 11-24　这个捕获记录中的缓慢数据包是初始 HTTP GET 请求数据包

刚开始这个捕获记录很正常，迅速完成了 TCP 握手，没有任何延迟。但握手完成之后，数据包 4 的 HTTP GET 请求出现了问题。该数据包继上一个数据包有 1.34s 的延迟。

我们应该查看数据包 3 和 4 之间到底发生了什么，以找到延迟的原因。数据包 3 是 TCP 握手过程中客户端发往服务器的最后一个 ACK，数据包 4 则是客户端发往服务器的 GET 请求。它们的共同点在于都是客户端发送的，与服务器无关。发送完 ACK 之后本应该很快就发送 GET 请求，因为所有动作都是以客户端为中心的。

不幸的是，ACK 并没有快速切换到 GET 请求。创建和传输 GET 数据包确实需要应用层的处理，而这个处理过程的延迟表明客户端无法及时执行该动作。这意味着通信高延迟的根源在于客户端。