第十八章 TCP:传输控制协议

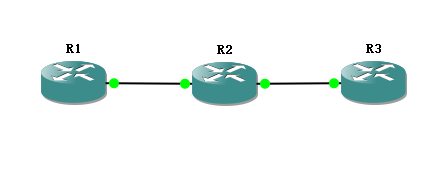
TCP是一个面向连接的协议。无论哪一方向另一方发送数据之前，都必须先在双方之间建立一条连接。

18.1 三次握手

发送第一个SYN的一端将执行主动打开(action open)。接收这个SYN并发回下一个SYN的另一端执行被动打开(passive open)

当一端为建立连接而发送它的SYN时，它为连接选择一个初始序号。ISN(随机序列号)随时间而变化，因此每个连接都将具有不同的ISN。RFC793[Postel 1981c]指出ISN可看作是一个32bit的计数器，每4ms加1。这样选择序号的目的在于防止在网络中被延迟的分组在以后被传送，而导致某个连接的一方对它错误的解释。

实验一：telnet连接



（a）拓扑图



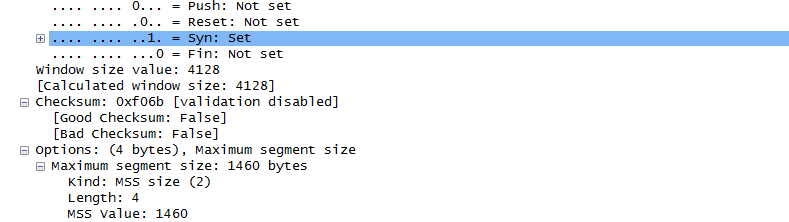
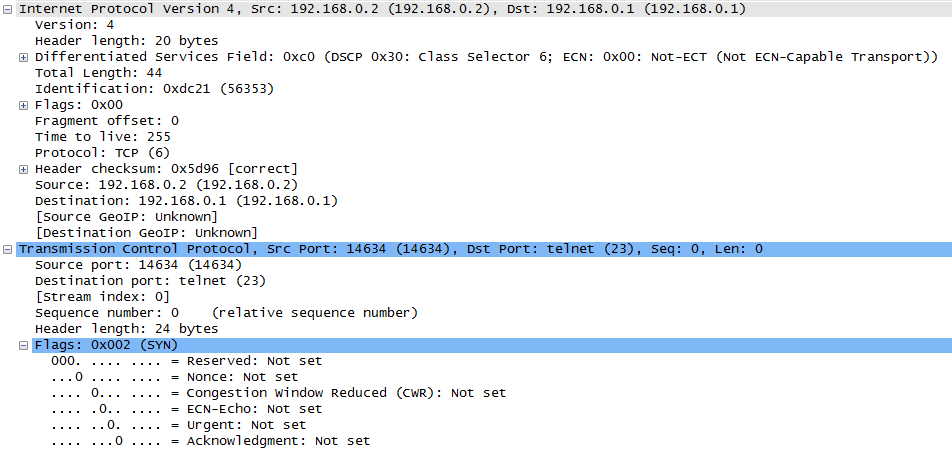
（b）三次握手流程

\*为什么TCP客户端最后还要发送一次确认呢？

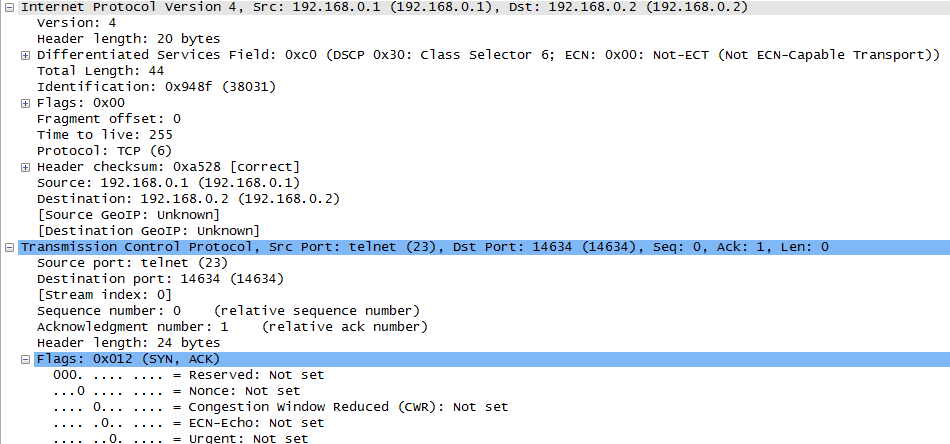
一句话，主要防止已经失效的连接请求报文突然又传送到了服务器，从而产生错误。

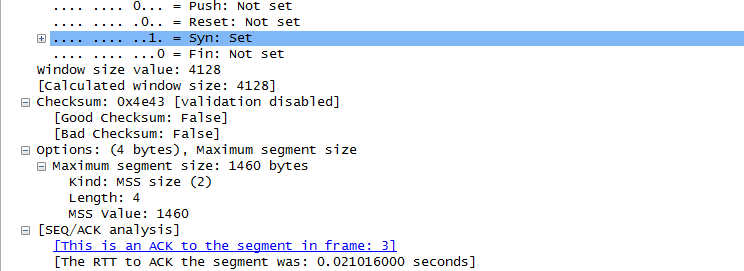
如果使用的是两次握手建立连接，假设有这样一种场景，客户端发送了第一个请求连接并且没有丢失，只是因为在网络结点中滞留的时间太长了，由于TCP的客户端迟迟没有收到确认报文，以为服务器没有收到，此时重新向服务器发送这条报文，此后客户端和服务器经过两次握手完成连接，传输数据，然后关闭连接。此时此前滞留的那一次请求连接，网络通畅了到达了服务器，这个报文本该是失效的，但是，两次握手的机制将会让客户端和服务器再次建立连接，这将导致不必要的错误和资源的浪费。

如果采用的是三次握手，就算是那一次失效的报文传送过来了，服务端接受了那条失效报文并且回复了确认报文，但是客户端不会再次发出确认。由于服务器收不到确认，就知道客户端并没有请求连接。

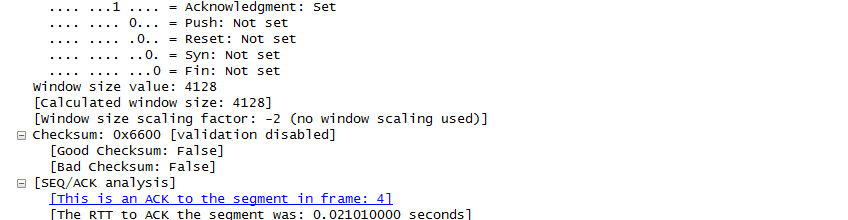
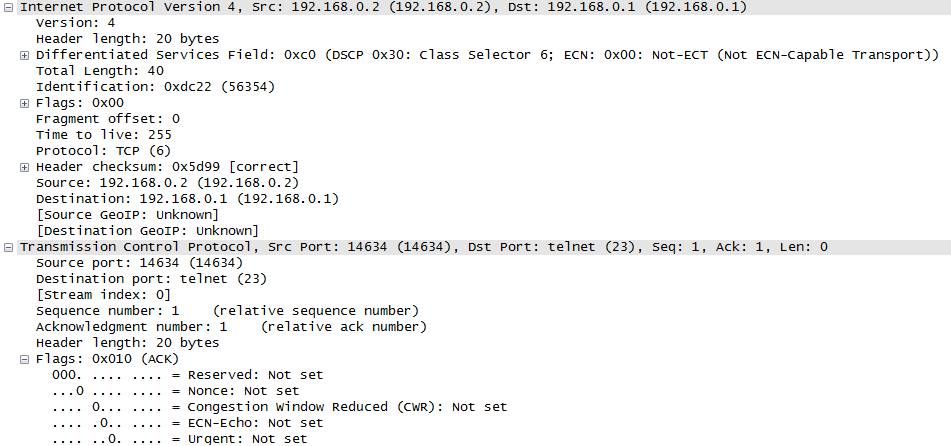


（c）wireshark抓包(ACK=0,SYN=1)





（d）wireshark抓包(ACK=1,SYN=1)



（e）wireshark抓包(ACK=1,SYN=0)

(a)R3(fa0/0 192.168.1.2)：设置默认路由(ip route 0.0.0.0 0.0.0.0 192.168.1.1)

R2(fa0/0 192.168.0.1 ，fa1/0 192.168.1.1)：设置允许广播通过(line vty 0 4 password 123456)

R1(fa0/0 192.168.0.2)：设置默认路由(ip route 0.0.0.0 0.0.0.0 192.168.0.2)

(b)R1#telnet 192.168.1.1 password 123456

18.2 四次挥手

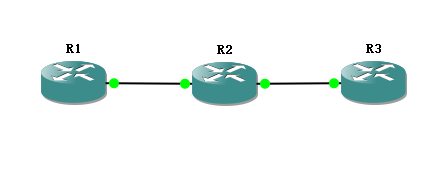
建立一个连接需要三次握手，而终止一个连接要经过4次握手。这由TCP的半关闭(half-close)造成的。既然一个TCP连接是全双工(即数据在两个方向上能同时传递)，因此每个方向必须单独进行关闭。这原则就是当一方完成它的数据发送任务后就能发送一个FIN来终止这个方向连接。当一端收到一个FIN，它必须通知应用层另一端几经终止了那个方向的数据传送。发送FIN通常是应用层进行关闭的结果。

收到一个FIN只意味着这一方向上没有数据流动。一个TCP连接在收到一个FIN后仍能发送数据。而这对利用半关闭的应用来说是可能的，尽管在实际应用中只有很少的TCP应用程序这样做。

首先进行关闭的一方(即发送第一个FIN)将执行主动关闭，而另一方(收到这个FIN)执行被动关闭。通常一方完成主动关闭而另一方完成被动关闭。

发送FIN将导致应用程序关闭他们的连接，这些FIN的ACK是由TCP软件自动产生的。

实验二：telnet断开



（a）拓扑图

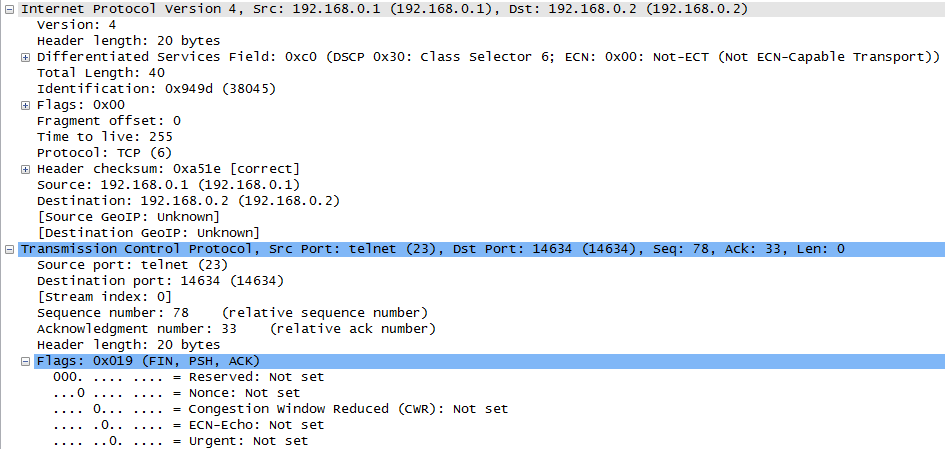


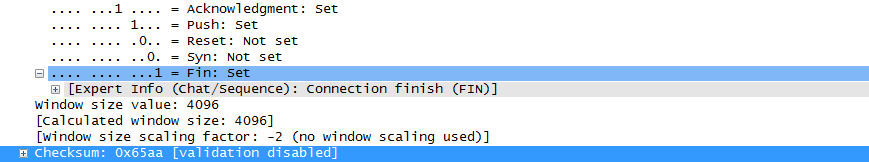
（b）四次分手流程

\*为什么客户端最后还要等待2MSL?

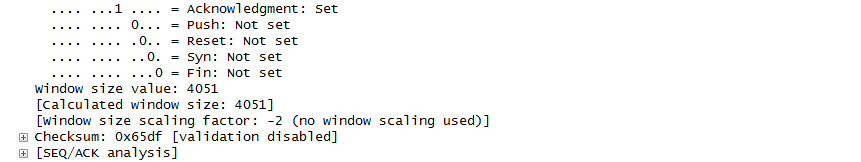
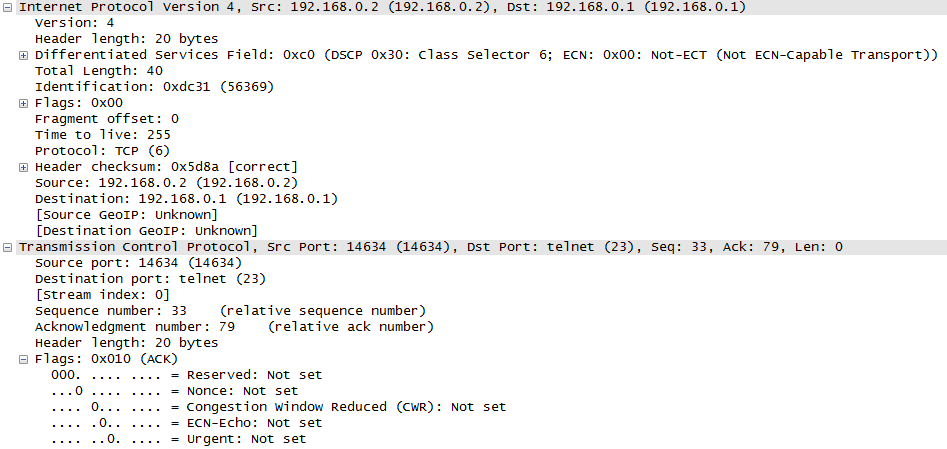
第一，保证客户端发送的最后一个ACK报文能够到达服务器，因为这个ACK报文可能丢失，站在服务器的角度看来，我已经发送了FIN+ACK报文请求断开了，客户端还没有给我回应，应该是我发送的请求断开报文它没有收到，于是服务器又重新发送一次，而客户端就能在这个2MSL时间段内收到这个重传的报文，接着给出回应报文，并且会重启2MSL计时器，所谓的2MSL是两倍的MSL(Maximum Segment Lifetime)。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果知道2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。

第二，防止类似与“三次握手”中提到了的，“已经失效的连接请求报文段”出现在本连接中，客户端发送完最后一个确认报文后，在这个2MSL时间中，就可以使本地连接持续的时间内产生的所有报文都从网络中消失。这样新的连接中不会出现旧连接的请求报文。

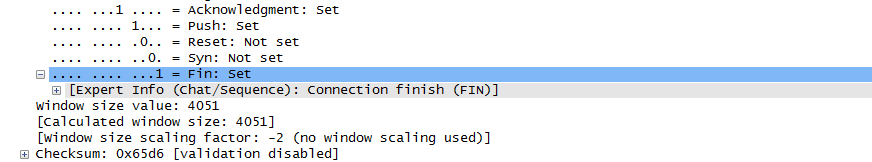
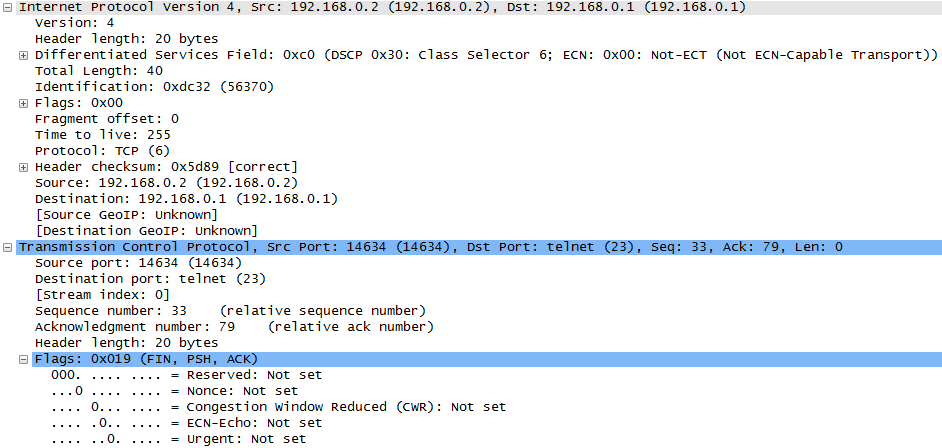




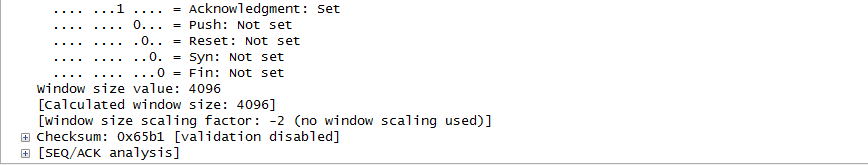
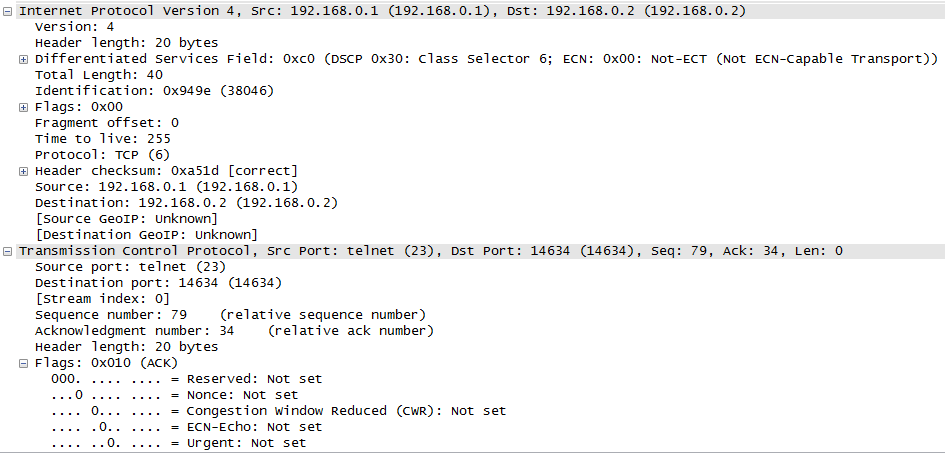
（c）wireshark抓包(FIN=1,ACK=1(完成上一次数据传输))



（d）wireshark抓包(ACK=1)



（e）wireshark抓包(FIN=1,ACK=1)



（f）wireshark抓包(FIN=1,ACK=1)

(a)R3(fa0/0 192.168.1.2)：设置默认路由(ip route 0.0.0.0 0.0.0.0 192.168.1.1)

R2(fa0/0 192.168.0.1 ，fa1/0 192.168.1.1)：设置允许广播通过(line vty 0 4 password 123456)

R1(fa0/0 192.168.0.2)：设置默认路由(ip route 0.0.0.0 0.0.0.0 192.168.0.2)

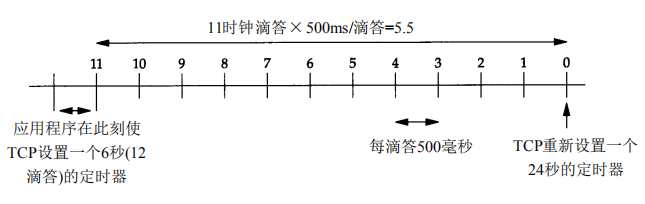
(b)R1#telnet 192.168.1.1 password 123456

18.3 建立连接超时

在思科路由器上telnet 192.168.100.5

超时重传0(0) 2(2) 6(4) 14(8) 30(16) 超时释放(指数退避)

18.4 第一次超时时间



第一次超时时，应用程序在两刻度之间，然后以下一个刻度为基础，向后数11个刻度，11时钟滴答\*500ms/滴答=5.5s，所以第一次超时时间为5.5s-6s之间，当到了整刻度是时(上图的0)，重新设置一个定时器，后面的超时间隔将会是准确的(操作系统不同设置定时器刻度存在差异)。

18.5 最大报文长度

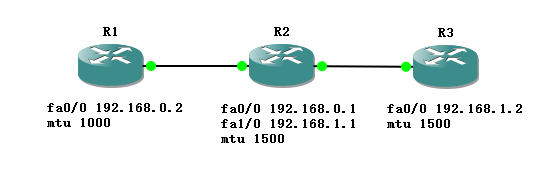
有些书中，将它看作可“协商”选项。它并不是任何条件下都可协商。当建立一个链接时，每一方都有用于通告它期望接收的MSS选项(MSS选项只能出现在SYN报文段中)。如果一方不接收来自另一方的MSS值，则MSS就定为默认值536字节。

一般来说，如果没有分段发生，MSS还是越大越好(这也并不总是正确)。报文越大允许每个报文段传送的数据就越多，相对IP和TCP首部有更高的网络利用率。当TCP发送一个SYN时，或者是因为一个本地应用进程想发起一个连接，或者是因为另一端的主机收到一个连接请求，它能将MSS值设置为外出接口上的MTU长度减去固定IP首部和TCP首部长度。对于一个以太网，MSS值可达1460字节。使用IEEE802.3的封装，它的MSS可达1452字节。

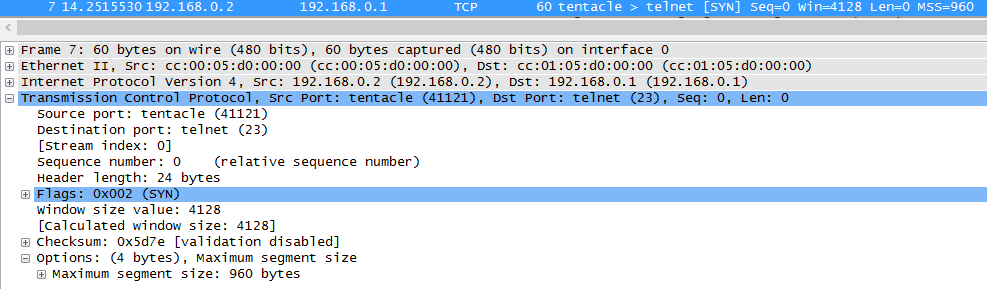
如果目的IP地址为“非本地的”，MSS通常的默认值为536。而区分地址是本地还是非本地是简单的，如果目的IP地址的网络号与子网号都和我们相同，则是本地的；如果目的IP地址的网络号与我们的完全不同，则是非本地的；如果目的IP地址的网络号与我们的相同而子网络号与我们的不同，则可能是本地的，也可能不是本地的。大多数TCP实现版都提供了一个配置选项，让系统管理员说明不同的子网是属于本地还是非本地。这个选项的设置将确定MSS可以选择尽可能大或是默认值536。

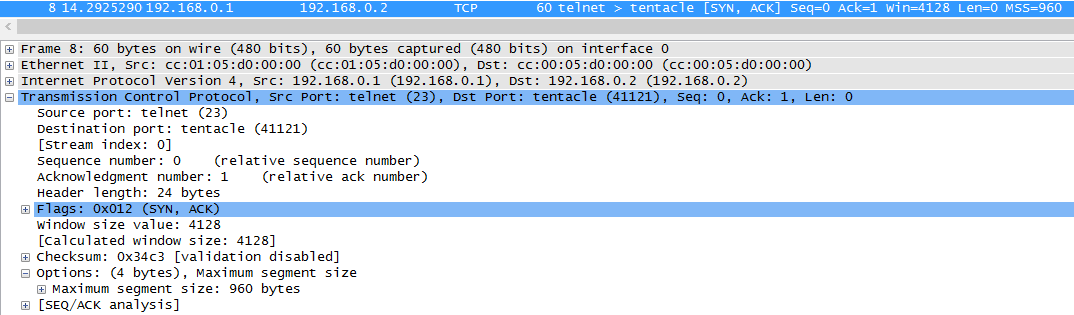
实验三：最大MSS

(1)R1 MTU 1000 telnet R2(fa0/0)



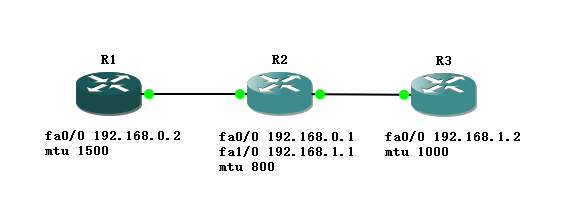
（a）拓扑图



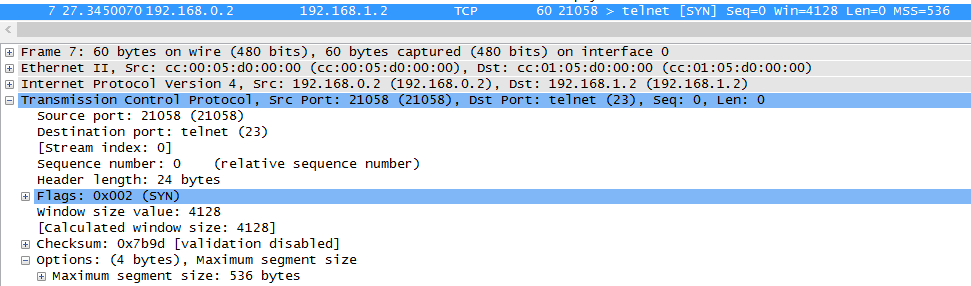
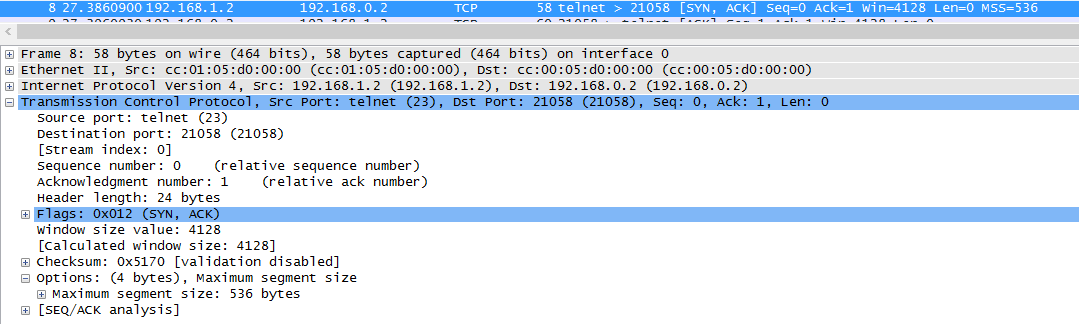


（b）三次握手前两次TCP报文(协商后最大MSS为960)

(2)R1 MTU 1500 R2 MTU 800 R3 MTU 1000 telnet R3(fa0/0)



（a）拓扑图

（b）三次握手前两次TCP报文(非本地的协商后最大MSS为536)

18.6 TCP迁移图

如实验一与实验二的三次握手和四次分手。

18.7平静时间的概念

对于来自某个连接的较早替身的迟到报文段，2MSL等待可以将它解释成使用相同插口对的新连接的一部分。但这只有在处于2MSL等待连接中的主机处于正常工作状态时才有效。

如果使用处于2MSL等待端口的主机出现故障，他会在MSL秒内重新启动，并立即使用故障前仍处于2MSL的插口对来建立一个新的连接吗?如果是这样，在故障前这个链接发出而迟到的报文段会被错误地当作属于重启后新连接的报文段。无论如何重启后新连接的初始序号，都会发生这种情况。

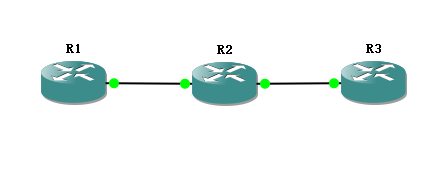
为了防止这种情况，RFC 793指出TCP在重启后的MSL秒内不能建立任何连接。这就称为平静时间。

只有极少的现实版遵守这一原则，因为大多数主机重启的时间比MSL秒要短。

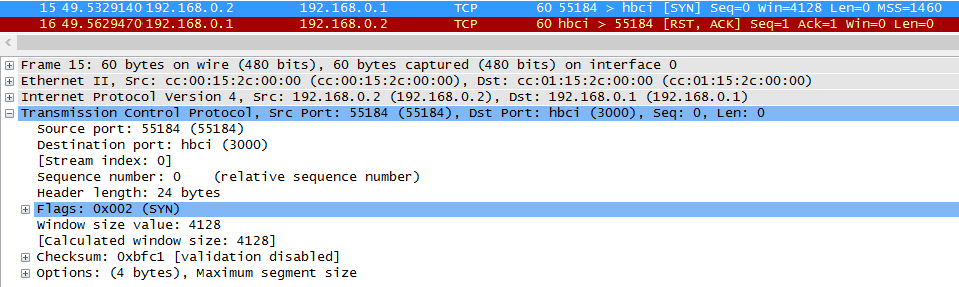
18.8复位报文段

一般来说，无论何时一个报文段发往基准的连接出现错误，TCP都会发出一个复位报文段(这里提到的“基准的连接”是指由目的的IP地址和目的的端口号以及源IP地址和源端口号指明的连接。这就是为什么RFC 793称之为插口)

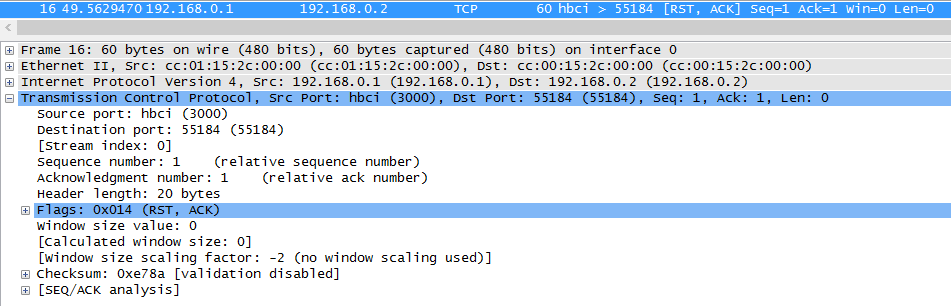
试验四：端口不可达复位(telnet 192.168.0.1 3000)



（a）拓扑图



（b）三次握手(SYN=1)



（c）端口不可达(RST=1,ACK=1)

18.9异常终止一个连接

终止一个连接的正常方式是一方发送FIN。有时这也称为有序释放。

有可能发送一个复位报文段而不是FIN来中途释放一个连接。有时称这个为异常释放。

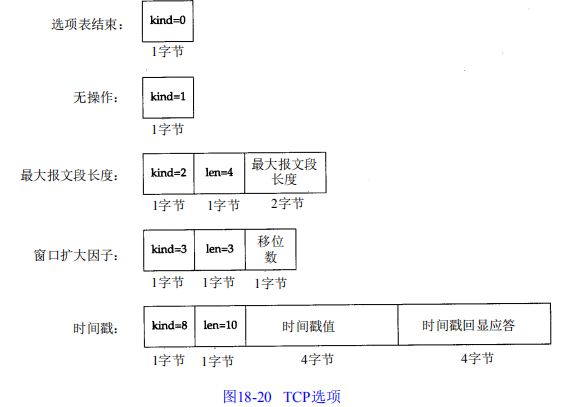
异常终止一个连接对应用程序来说有两个优点：(1)丢弃任何待发数据并立即发送复位报文段；(2)RST的接收方会区分另一端执行的是异常关闭还是正常关闭。应用程序使用的API必须提供产生异常关闭而不是正常关闭的手段。

18.10异常终止一个连接

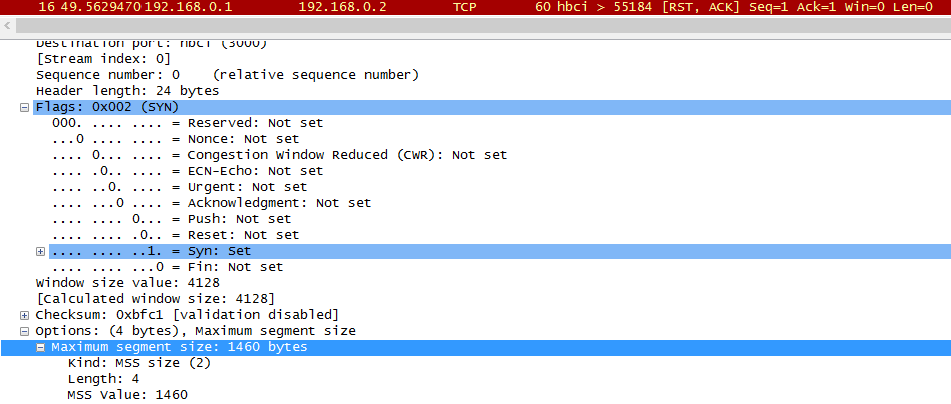
如果一方已经关闭或异常终止连接而另一方不知道，我们将这样的TCP连接称为半打开的。任何一端的主机异常都会可能导致发生这种情况。只要不打算在半打开连接上传输数据，仍处于连接状态的一方就不会检测另一方已经出现异常。比如说一方异常了，另一方发送数据，就会收到RST复位，这时候就就知道对方异常了。

18.11同时打开和关闭

18.12 TCP选项



每一个选项的开始1字节kind字段，说明选项的类型。Kind字段为0和1选项仅占1个字节。其他的选项在kind字节后还有len字节。他说明的长度是指总长度，包括kind字节和len字节。（不够四字节边界时会补齐边界）



（a）最大报文段长度