

TCP 协议分析

一 实验目的

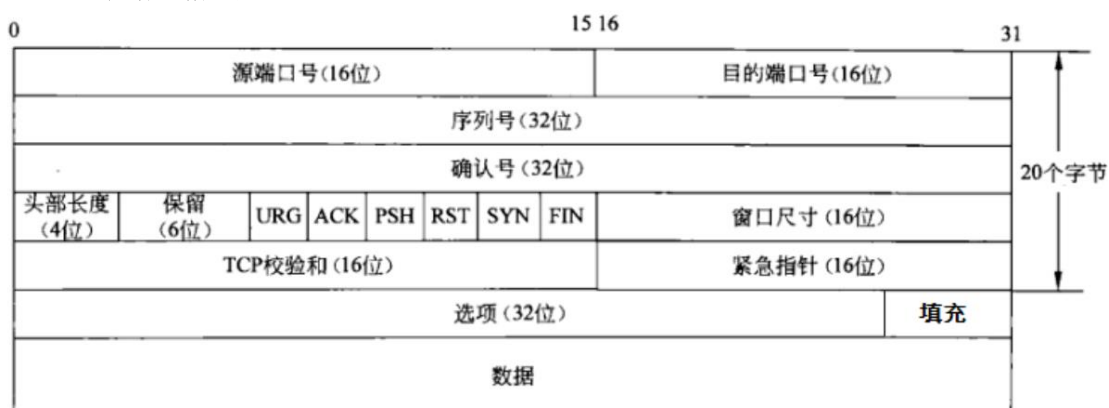
1. 学会使用 Wireshark 或其他工具进行特定包抓取，并对包进行必要的分析。
2. 分析 TCP 协议首部信息。
3. 分析验证 TCP 连接建立与释放过程。
4. 分析验证 TCP 协议的数据传输过程。

二 预备知识

1. TCP 协议

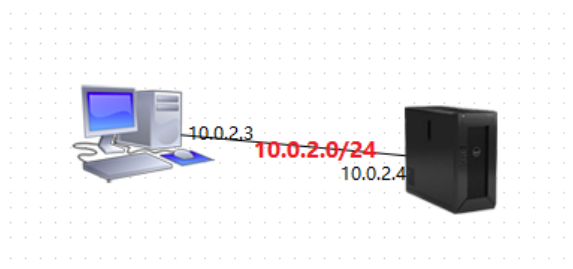
TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节的传输层通信协议，由 IETF 的 RFC 793 定义。在简化的计算机网络 OSI 模型中，它完成第四层传输层所指定的功能，用户数据报协议（UDP）是同一层内 另一个重要的传输协议。在因特网协议族（Internet protocol suite）中，TCP 层是位于 IP 层之上，应用层之下的中间层。不同主机的应用层之间经常需要可靠的、像管道一样的连接，但是 IP 层不提供这样的流机制，而是提供不可靠的包交换。

2. TCP 数据包格式



三 实验环境

在右上方的实验拓扑图菜单中选择 **TCP 协议分析实验**，点击连线设置子网网段：

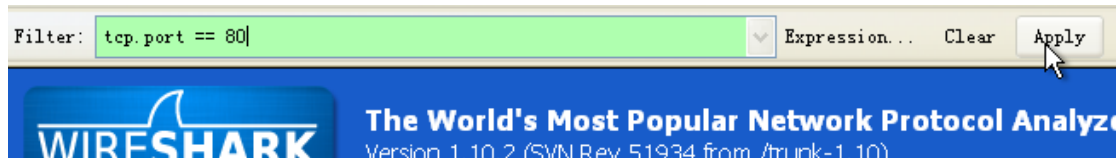


四 实验内容

1. 捕包

首先进入 ApplicationServer, (用户名: centos 密码: centos), 运行命令 `sudo /sbin/service httpd start`, 启动 http 服务。

我们进入 PC, 打开桌面的 Wireshark 软件, 在过滤规则栏上输入 `tcp.port == 80` (过滤 http 协议), 点击 Apply 按钮后开始抓包。



接下来我们打开浏览器, 输入 10.0.2.4, 然后我们可以看到所捕获的数据包:

Source	Destination	Protocol	Length	Info
10.0.2.3	10.0.2.4	TCP	62	nim > http [SYN] Seq=0 win=65535
10.0.2.4	10.0.2.3	TCP	62	http > nim [SYN, ACK] Seq=0 Ack=1
10.0.2.3	10.0.2.4	TCP	54	nim > http [ACK] Seq=1 Ack=1 win=

图 1 tcp 三次握手

2. TCP 三次握手

图 1 即为 TCP 建立连接时三次握手的过程:

- Transmission Control Protocol, Src Port: nim (1058), Dst Port: http (80), Seq: 0, Len: 0
 - source port: nim (1058)
 - destination port: http (80)
 - [Stream index: 0]
 - Sequence number: 0 (relative sequence number)
 - Header length: 28 bytes
 - Flags: 0x002 (SYN)
 - window size value: 65535
 - [Calculated window size: 65535]
 - Checksum: 0x43c2 [validation disabled]

第一次握手: 1058>http, 客户发出连接请求, 标志位为中 SYN=1, 序列号 Seq=0。发送完毕后, 客户端进入 SYN_SEND 状态。

- Internet Protocol Version 4, Src: 10.0.2.4 (10.0.2.4), Dst: 10.0.2.3 (10.0.2.3)
- Transmission Control Protocol, Src Port: http (80), Dst Port: nim (1058), Seq: 0, Ack: 1, Len: 0
 - source port: http (80)
 - destination port: nim (1058)
 - [Stream index: 0]
 - Sequence number: 0 (relative sequence number)
 - Acknowledgment number: 1 (relative ack number)
 - Header length: 28 bytes
 - Flags: 0x012 (SYN, ACK)
 - window size value: 27200
 - [Calculated window size: 27200]
 - Checksum: 0x765f [validation disabled]

第二次握手 http>1058, 服务器收到连接请求后, 向客户发出确认报文段, 标志位中 SYN 和 ACK 被置为 1, 序列号 Seq=0, 确认号 Ack = 1 (第一次握手时的 Seq+1)。发送完毕后, 服务器端进入 SYN_RCVD 状态。

- Transmission Control Protocol, Src Port: nim (1058), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
 - source port: nim (1058)
 - destination port: http (80)
 - [Stream index: 0]
 - Sequence number: 1 (relative sequence number)
 - Acknowledgment number: 1 (relative ack number)
 - Header length: 20 bytes
 - Flags: 0x010 (ACK)
 - window size value: 65535
 - [Calculated window size: 65535]
 - [window size scaling factor: -2 (no window scaling used)]
 - Checksum: 0x1821 [validation disabled]

第三次握手 1058>http, 客户收到服务器确认报文段后向服务器发送连接请求确认报文段, 标志位为 ACK, 序列号 Seq=1, 确认号 Ack=1 (第二次握手时的 Seq+1)。发送完毕后, 客户端进入 ESTABLISHED 状态, 当服务器端接收到这个包时, 也进入 ESTABLISHED 状态, TCP 握手结束。

3. TCP 四次挥手

然后我们从抓取的数据包中找出撤销连接时使用的数据包(以[Fin,Ack]开头), 分析相关信息, 检验是否与所学一致, 判断该撤销过程采用了半关闭模式。

10.0.2.4	10.0.2.3	TCP	54	http > bsquare-voip [FIN, ACK] Seq=7448 Ack=2194 win=33
10.0.2.3	10.0.2.4	TCP	54	bsquare-voip > http [ACK] Seq=2194 Ack=7449 win=64658 L
10.0.2.3	10.0.2.4	TCP	54	bsquare-voip > http [FIN, ACK] Seq=2194 Ack=7449 win=64
10.0.2.4	10.0.2.3	TCP	54	http > bsquare-voip [ACK] Seq=7449 Ack=2195 win=33232 L

以下为 TCP 连接释放四次握手的具体过程:

```
Transmission Control Protocol, Src Port: http (80), Dst Port: bsquare-voip (1071), Seq: 7448, Ack: 2194, Len: 0
  Source port: http (80)
  Destination port: bsquare-voip (1071)
  [Stream index: 2]
  Sequence number: 7448 (relative sequence number)
  Acknowledgment number: 2194 (relative ack number)
  Header length: 20 bytes
  Flags: 0x011 (FIN, ACK)
  window size value: 33232
  [Calculated window size: 33232]
  [window size scaling factor: -2 (no window scaling used)]
  Checksum: 0xbb6e [validation disabled]
```

第一次挥手: 标志位为 FIN+ACK, Seq=7448, Ack = 2194。客户端表示自己已经没有数据可以发送了, 但是仍然可以接受数据。发送完毕后, 客户端进入 FIN_WAIT_1 状态。

```
Transmission Control Protocol, Src Port: bsquare-voip (1071), Dst Port: http (80), Seq: 2194, Ack: 7449, Len: 0
  Source port: bsquare-voip (1071)
  Destination port: http (80)
  [Stream index: 2]
  Sequence number: 2194 (relative sequence number)
  Acknowledgment number: 7449 (relative ack number)
  Header length: 20 bytes
  Flags: 0x010 (ACK)
  window size value: 64658
  [Calculated window size: 64658]
  [window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x1821 [validation disabled]
```

第二次挥手: 标志位为 ACK, Seq=2194, ACK 为 7449(上一个包的 Seq+1), 此时, TCP 的连接处于半关闭状态。服务器端表明自己接受到了客户端关闭连接的请求, 但还没有准备好关闭连接。发送完毕后, 服务器端进入 CLOSE_WAIT 状态, 客户端接收到这个确认包之后, 进入 FIN_WAIT_2 状态, 等待服务器端关闭连接。

```
Transmission Control Protocol, Src Port: bsquare-voip (1071), Dst Port: http (80), Seq: 2194, Ack: 7449, Len: 0
  Source port: bsquare-voip (1071)
  Destination port: http (80)
  [Stream index: 2]
  Sequence number: 2194 (relative sequence number)
  Acknowledgment number: 7449 (relative ack number)
  Header length: 20 bytes
  Flags: 0x011 (FIN, ACK)
  window size value: 64658
  [Calculated window size: 64658]
  [window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x1821 [validation disabled]
```

第三次挥手, 标志位为 FIN+ACK, Seq=2194, ACK=7449。发送完毕后, 服务器端进入 LAST_ACK 状态, 等待来自客户端的最后一个 ACK。

```
Transmission Control Protocol, Src Port: http (80), Dst Port: bsquare-voip (1071), Seq: 7449, Ack: 2195, Len
  Source port: http (80)
  Destination port: bsquare-voip (1071)
  [Stream index: 2]
  Sequence number: 7449 (relative sequence number)
  Acknowledgment number: 2195 (relative ack number)
  Header length: 20 bytes
  Flags: 0x010 (ACK)
  window size value: 33232
  [Calculated window size: 33232]
  [window size scaling factor: -2 (no window scaling used)]
  Checksum: 0xbb6d [validation disabled]
```

第四次挥手，标志位为 ACK，Seq = 7449, Ack=2195。客户端接收到来自服务器端的关闭请求，发送一个确认包，并进入 TIME_WAIT 状态，等待可能出现的要求重传的 ACK 包。服务器端接收到这个确认包之后，关闭连接，进入 CLOSED 状态。

客户端等待了某个固定时间（两个最大段生命周期，2MSL，2 Maximum Segment Lifetime）之后，没有收到服务器端的 ACK，认为服务器端已经正常关闭连接，于是自己也关闭连接，进入 CLOSED 状态。

4. 总结

所谓三次握手(Three-way Handshake)，是指建立一个 TCP 连接时，需要客户端和服务端总共发送 3 个包。三次握手的目的是连接服务器指定端口，建立 TCP 连接，并同步连接双方的序列号和确认号，交换 TCP 窗口大小信息。TCP 的连接的拆除需要发送四个包，因此称为四次挥手(Four-way handshake)，也叫做改进的三次握手。客户端或服务端均可主动发起挥手动作。

5. 扩展

试回答以下与 TCP 相关的问题：

- 1) 为什么建立连接协议是三次握手，而关闭连接却是四次握手呢？
- 2) 为什么不能用两次握手进行连接？
- 3) 试着说明三次握手建立连接时，发送方再次发送确认的必要性？