**华中科技大学计算机学院**

**《计算机通信与网络》实验报告**

实验名称 分析TCP特性

|  |  |  |  |
| --- | --- | --- | --- |
| 姓 名 | 班 级 | 学 号 | 得 分 |
| 潘翔 | IOT1601 | U201614898 |  |

教师评语：

# Lab6 分析TCP特性

## 6.1 环境

操作系统： Manjaro-4.18.10-1 x86\_64 (Arch-Based Distribution)

网络平台： Wireshark 2.6.3

网络环境：

Link encap:Ethernet  HWaddr a0:8c:fd:24:5d:4c     
inet addr:222.20.100.153  Bcast:222.20.101.255  Mask:255.255.254.0   
inet6 addr: fe80::2476:27:cd9d:d75b/64 Scope:Link   
inet6 addr: 2001:250:4000:803c:e3c1:b69:d9f2:67b0/64 Scope:Global

## 6.2 实验目的

### 6.2.1 运输层端口观察实验

1. 理解运输层的端口与应用层的进程之间的关系；
2. 了解端口号的划分和分配。

### 6.2.2 UDP 协议与 TCP 协议的对比分析

1. 熟悉 UDP 与 TCP 协议的主要特点及支持的应用协议；
2. 理解 UDP 的无连接通信与 TCP 的面向连接通信；
3. 熟悉 TCP 报文段和 UDP 报文的数据封装格式。

### 6.2.3 TCP 的连接管理

1. 熟悉 TCP 通信的三个阶段；
2. 理解 TCP 连接建立过程和 TCP 连接释放过程。

## 6.3 实验内容及步骤

### 6.3.1 运输层端口观察实验

**任务一： 通过捕获的 DNS 事件查看并分析 UDP 的端口号**

1. 步骤 1： 捕获 DNS 事件
2. 步骤 2： 查看并分析 UDP 用户数据报中的端口号

本步骤需注意观察并分析以下几项内容：

* 1. DNS 请求包和应答包的源、 目的端口号是否发生变化；
  2. 判断 PC 和 Server 的客户端/服务器角色， 分析判断依据。

1. 步骤 3： 分析端口号的变化规律

重新回到 PC 机的浏览器窗口单击 Go（ 转到） 按钮再次请求相同的网页， 从新捕获的 DNS 事件中观察 DNS 客户端与 DNS 服务器端的端口号是否发生变化。如果没有，分析其原因；如果有，分析其变化的规律。

特别注意： 分析完成后不能单击 Reset Simulation（ 重置模拟） 按钮清空原有的事

件， 同时也不能关闭 PC 的配置窗口。

**任务二： 通过捕获的 HTTP 事件查看并分析 TCP 的端口号**

1. 步骤 1： 捕获 HTTP 事件
2. 步骤 2： 查看并分析 TCP 报文中的端口号

本步骤需注意观察并分析以下几项内容：

* 1. TCP 报文中的源端口和目的端口值；
  2. 确定 PC 和 Server 的客户端/服务器角色。

完成后单击 Reset Simulation（重置模拟）按钮，将原有的事件清空。

**任务三： 分析运输层端口号**

1. 步骤 1：分析运输层端口号与应用进程之间的关系

对比任务一中 DNS 服务器端的端口号与任务二中服务器端的端口号是否相同， 并分析其原因。

1. 步骤 2： 分析运输层动态端口号的分配规律

重新捕获 HTTP 事件以分析 TCP 协议的端口号变化情况。具体操作方法参考任务二中的步骤 2。该步骤重点观察 HTTP 客户端的端口号，并与任务二中观察到的 HTTP 客户端的端口号进行对比，分析归纳动态端口号的分配规律。

完成后单击 Reset Simulation（重置模拟）按钮，将原有的事件清空。

**思考：**

1. 运输层如何区分应用层的不同进程？
2. 若使用Reset Simulation（重置模拟）按钮后再重新进行捕获，端口号如何变化？新的值与重置前有关吗？

### 6.3.2 UDP 协议与 TCP 协议的对比分析

**任务一： 观察 UDP 无连接的工作模式**

1. 步骤 1： 捕获 UDP 事件
2. 步骤 2： 分析 UDP 无连接的工作过程

本步骤仅查看第 4 层中 UDP 报文段的内容。 注意观察并分析以下几项内容：

* 1. 运输层的 UDP 发送 DNS 的请求之前是否有先建立连接；
  2. 记录 UDP 的用户数据报首部中的 LENGTH 字段的值， 分析该报文的首部及数据部分的长度。

分析完成后单击 Reset Simulation（ 重置模拟） 按钮， 将原有的事件全部清空。

**任务二： 观察 TCP 面向连接的工作模式**

1. 步骤 1： 捕获 TCP 事件
2. 步骤 2： 分析 TCP 面向连接的工作过程

本步骤仅查看第 4 层中 TCP 报文段的内容。 注意观察并分析以下几项内容：

* 1. 在捕获到的第一个 HTTP 事件之前及最后一个 HTTP 事件之后是否有 TCP 事件；
  2. 第一个以及最后一个 HTTP 事件对应的 TCP 报文中的 sequence number（序号）、ACK number（确认号） 的值以及它们与 data length（数据长度） 的关系；
  3. 并查看 TCP 报文首部中固定部分的长度。

分析完成后单击 Reset Simulation（重置模拟） 按钮， 将原有的事件全部清空。

思考：

1. TCP 报文首部中的序号和确认号有什么作用？
2. 无连接的 UDP 和面向连接的 TCP 各有什么优缺点？

### 6.3.3 TCP 的连接管理

**任务一： 捕获 TCP 事件**

**任务二： 分析 TCP 连接建立阶段的三次握手**

1. 注意观察任务一中捕获到的 TCP 事件， 完成以下几项内容：

分析 TCP 连接建立阶段的三次握手的过程；

1. 查看 TCP 报文段首部中的各项字段的值，包括 SYN 字段、ACK 字段、 PSH 字段、FIN字段、sequence number（序号）字段、ACK number（确认号）字段、窗口大小、选项字段 MSS（ 最大报文段长度）、报文段长度等；
2. 分析三次握手过程中 TCP 连接状态的变迁。

**任务三： 分析 TCP 连接释放阶段的三次握手**

继续观察任务一中捕获到的 TCP 事件， 完成以下几项内容：

* 1. 分析 TCP 连接释放阶段的三次握手的过程；
  2. 查看 TCP 报文段首部中的各项字段的值，包括 SYN 字段、ACK 字段、PSH 字段、FIN字段、sequence number（序号） 字段、 ACK number（确认号）字段、窗口大小、选项字段 MSS（最大报文段长度）、报文段长度等；
  3. 分析三次握手过程中 TCP 连接状态的变迁。

**思考：**

1. 连接建立阶段的第一次握手是否需要消耗一个序号？ 其 SYN 报文段是否携带数据？ 为什么？ 第二次握手呢？
2. 本实验中连接释放过程的第二、三次握手是同时进行的还是分开进行的？ 这两次握手何时需要分开进行？
3. 本实验中连接释放阶段的第三次握手，PC 向Server发送最后一个TCP确认报文段后，为什么不是直接进入CLOSED（已关闭）连接状态，而是进入 CLOSING（ 正在关闭）连接状态？
4. 本实验中 TCP 连接建立后的数据通信阶段， PC 向 Server 发送的了多少数据？Server向 PC 发送的数据呢？

## 6.4 实验结果

### 6.4.1运输层端口观察实验

**任务一： 通过捕获的 DNS 事件查看并分析 UDP 的端口号**

1. 步骤 1： 捕获 DNS 事件

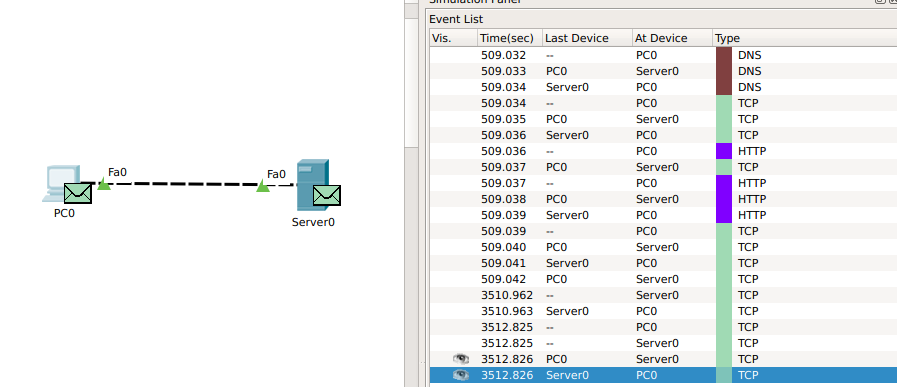


图6-1 捕获DNS事件

1. 步骤 2： 查看并分析 UDP 用户数据报中的端口号

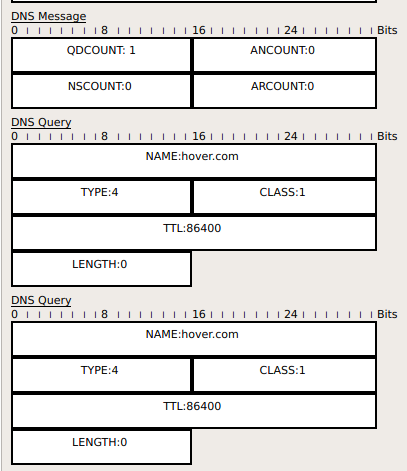


图6-2 初始DNS报文

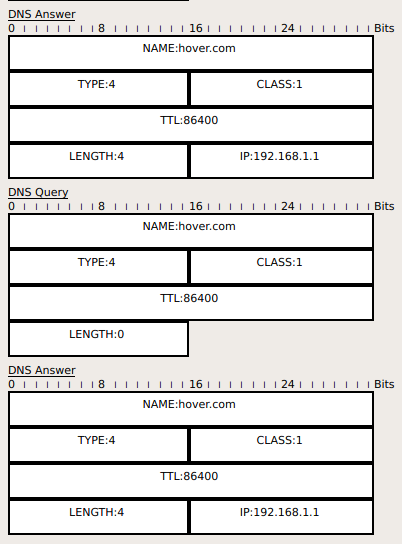


图6-3 DNS请求报文

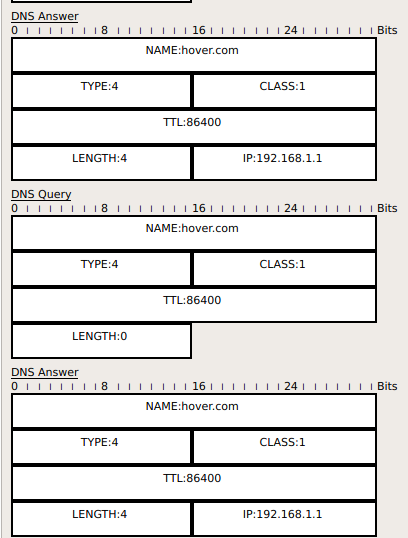


图6-4 DNS应答报文

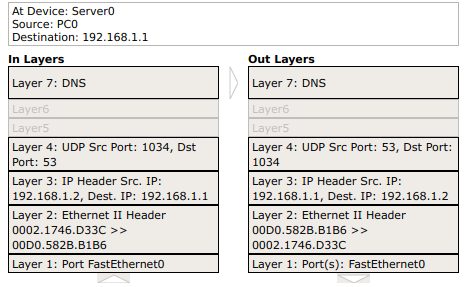


图6-5 DNS报文变换

* 1. DNS请求包和应答包的源、 目的端口号是否发生变化；
     1. 发生变化
        1. 请求包：源端口为1034，目的端口53
        2. 应答包：源端口为53，目的端口1034
  2. 判断 PC和Server的客户端/服务器角色，分析判断依据。

PC0为客户端，Server为服务器，因为PC0发出request包对服务器进行问询

1. 步骤 3： 分析端口号的变化规律

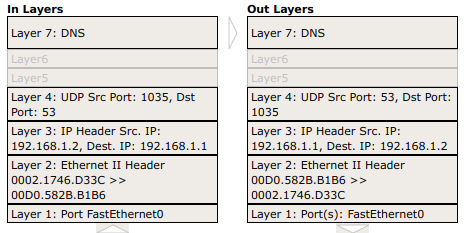


图6-6 再次DNS端口变化

在客户端端口加一，服务端端口不变，此处为模拟的相应策略，在实际环境中，为操作系统所维护的PORT资源池，为防止端口冲突，进行资源池滚动，因为DNS为应用层协议，有特定的请求端口53.

**任务二： 通过捕获的 HTTP 事件查看并分析 TCP 的端口号**

1. 步骤 1： 捕获 HTTP 事件

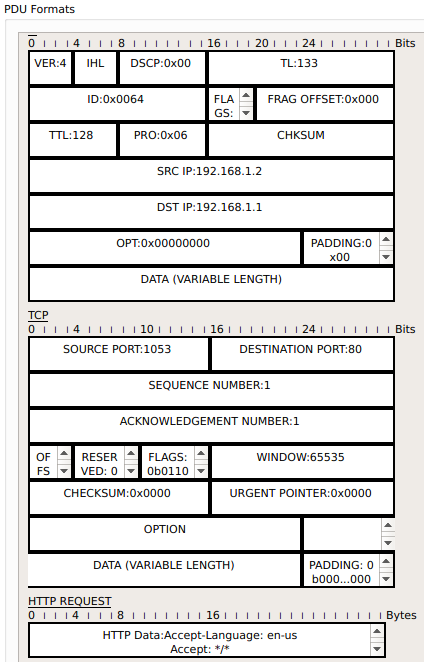


图6-7 HTTP报文

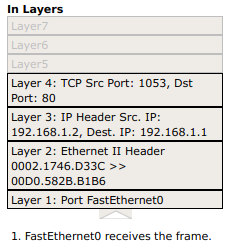


图6-7 封装而成的TCP报文结构

1. 步骤 2： 查看并分析 TCP 报文中的端口号
   1. TCP请求包和应答包的源、 目的端口号是否发生变化；
      1. 发生变化
         1. 请求包：源端口为1034，目的端口80
         2. 应答包：源端口为80，目的端口1034
   2. 判断 PC和Server的客户端/服务器角色，分析判断依据。

PC0为客户端，Server为服务器，因为Server对PC0的HTTP请求做出回应，进行服务

**任务三： 分析运输层端口号**

1. 步骤 1：分析运输层端口号与应用进程之间的关系

二者端口号不同，对于典型的应用层协议，有固定的保留端口，DNS为64，HTTP为80.

1. 步骤 2： 分析运输层动态端口号的分配规律

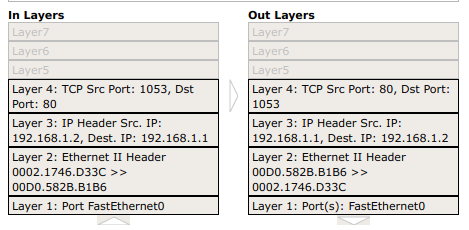


图6-8 第一次请求端口图

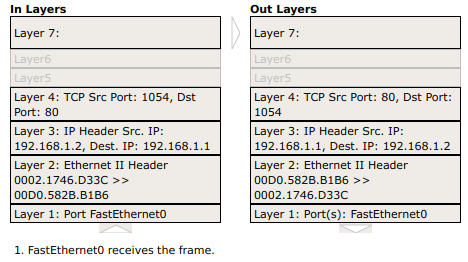


图6-9 第二次请求端口图

对于PC0发出的报文来说，作为请求报文，目的端口不变，而src端口在PacketTracer之中进行加1,实际过程中，维护端口池，每次从其中申请一个，进行循环滚动，防止端口冲突。

**思考：**

1. 运输层如何区分应用层的不同进程？

根据不同的src端口进行区分

1. 若使用Reset Simulation（重置模拟）按钮后再重新进行捕获，端口号如何变化？新的值与重置前有关吗？

有关，仍然为加一，重置模拟应该是对于网络环境的缓存进行清楚，可对于 PC0并没有掉电，其中的分配策略应该没有变，则继续进行新端口申请操作， 且维护之前的状态。

### 6.4.2 UDP协议与TCP协议的对比分析

**任务一： 观察UDP无连接的工作模式**

1. 步骤 1：捕获UDP事件

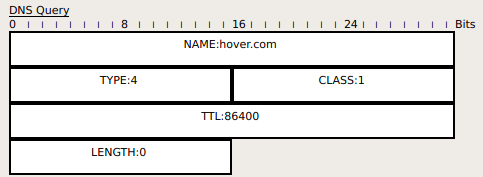


图6-10 DNS询问报文

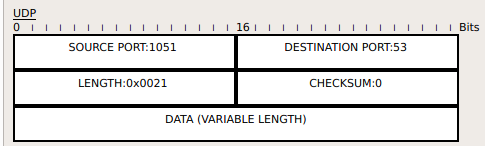


图6-11 DNS询问报文封装而成的UPD报文

1. 步骤 2： 分析UDP无连接的工作过程
   1. 运输层的UDP发送DNS的请求之前是否有先建立连接；

不进行连接

* 1. 记录UDP的用户数据报首部中的LENGTH字段的值， 分析该报文的首部及数据部分的长度。
     1. LENGTH=0x0021
     2. 首部为8字节，数据部分长度为33-8=25字节

**任务二： 观察 TCP 面向连接的工作模式**

1. 步骤 1： 捕获 TCP 事件

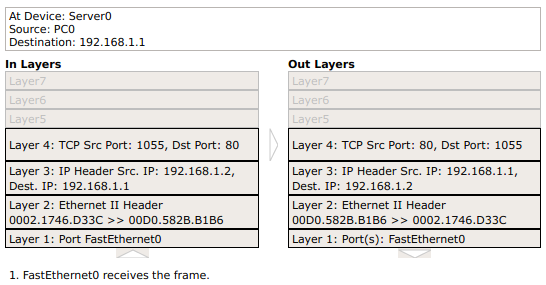


图6-12 TCP握手过程1,2

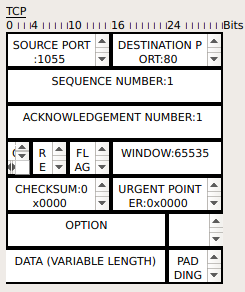


图6-13 第一个HTTP事件对应TCP报文

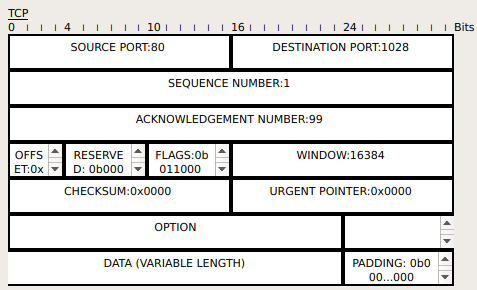


图6-14 最后一个HTTP事件对应TCP报文

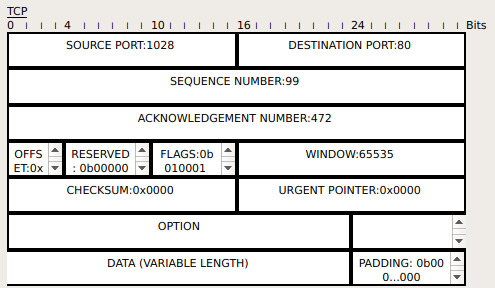


图6-15 最后一个HTTP后第一个TCP报文

1. 步骤 2： 分析 TCP 面向连接的工作过程
   1. 在捕获到的第一个 HTTP 事件之前及最后一个 HTTP 事件之后是否有 TCP 事件？

均有TCP事件,在第一个HTTP数据包之前需要建立连接故有TCP握手事件，在最后一个HTTP数据包之后需要释放连接，故有TCP挥手事件。

* 1. 第一个以及最后一个 HTTP 事件对应的 TCP 报文中的 sequence number（序号）、ACK number（确认号） 的值以及它们与 data length（数据长度） 的关系；
     1. 第一个HTTP事件

sequence number=1;

ACK number=1;

data length=下一个报文确认-1

=[最后一个HTTP报文ACK]-1

=99-1

=98

* + 1. 最后一个HTTP事件

sequence number=1;

ACK number=99;

data length=下一个报文确认-1

=[最后一个HTTP后第一个TCP报文ACK]-1

=472-1

=471

* 1. 并查看 TCP 报文首部中固定部分的长度。

TCP 报文首部中固定部分的长度:20字节。

**思考：**

1. TCP 报文首部中的序号和确认号有什么作用？

序号：本段数据第一个字节的计数码

确认号：希望对方发送的第一个字节的计数码

序列号/确认号被用来跟踪该端发送的数据量

1. 无连接的 UDP 和面向连接的 TCP 各有什么优缺点？

TCP：

提供面向连接、可靠数据传输和拥塞控制等服务。保证了数据传输的准确性，但没有时延保障。且建立和维护连接需要额外的字段和报文，造成了一定程度的相对资源占用。

UDP：

提供无连接的、不可靠的数据传输服务，没有包括拥塞控制机制，无连接的UDP具有更快的响应速度和更快的传输速度，所以UDP的发送方可以以任意速率向网络层注入数据。适用于实时应用，客人容忍少量丢包。

二者均无时延保障。

### 6.4.3 TCP 的连接管理

**任务一： 捕获 TCP 事件**

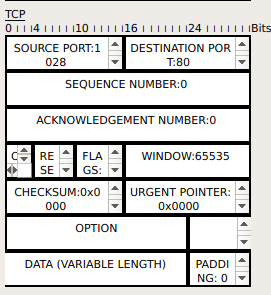


图6-16 TCP连接握手过程1

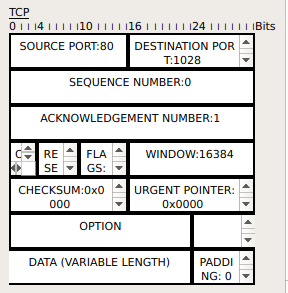


图6-17 TCP连接握手过程2

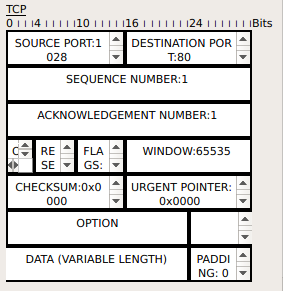


图6-18 TCP连接握手过程3

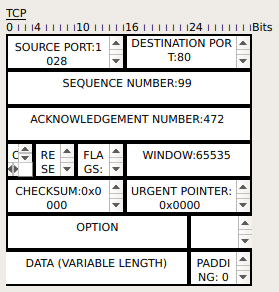


图6-19 TCP释放挥手过程1

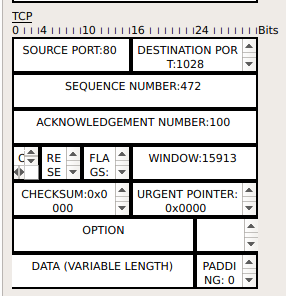


图6-20 TCP释放挥手过程2

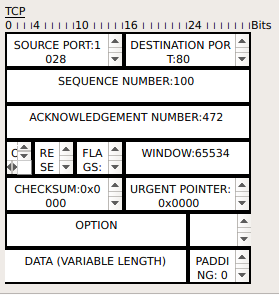


图6-21 TCP释放挥手过程4

**任务二： 分析 TCP 连接建立阶段的三次握手**

1. 注意观察任务一中捕获到的 TCP 事件， 完成以下几项内容：

分析 TCP 连接建立阶段的三次握手的过程；

1. 查看 TCP 报文段首部中的各项字段的值，包括 SYN 字段、ACK 字段、 PSH 字段、FIN字段、sequence number（序号）字段、ACK number（确认号）字段、窗口大小、选项字段 MSS（ 最大报文段长度）、报文段长

表6-1 TCP连接过程

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | SYN | ACK | PSH | FIN | sequence number | ACK number | 窗口大小 | MSS | 报文段长 |
| TCP握手过程1 | 1 | 0 | 0 | 0 | 0 | 0 | 65535 | OPTION |  |
| TCP握手过程2 | 1 | 1 | 0 | 0 | 0 | 1 | 16384 | OPTION |  |
| TCP握手过程3 | 0 | 1 | 0 | 0 | 1 | 1 | 65535 | OPTION | 第三次握手可有数据 |

1. 分析三次握手过程中 TCP 连接状态的变迁。
   1. 第一个次握手：A的TCP客户进程首先创建传输控制模块TCB，向服务器B发送连接请求报文段，即SYN报文段，此时首部中的同步位SYN=1，同时选择一个初始序号seq=x，这时，TCP客户进程进入SYN-SENT（同步已发送）状态，SYN报文段不能携带数据，但会消耗一个序号。
   2. 第二次握手：服务器B收到SYN报文段后，向A发送确认报文段，即SYN+ACK报文段，在此报文段中，SYN和ACK位置1，确认号是ack=x+1，同时也为自己选择一个初始序号seq=y，这时，TCP服务器进程进入SYN-RCVD(同步收到)状态,不能携带数据，但要消耗一个序号。
   3. 第三次握手：TCP客户进程收到服务器B的确认后，向服务器B发送确认报文段，即ACK报文段，在此报文段中，ACK置1，确认号ack=y+1，序号seq=x+1，这时，TCP客户进程进入ESTABLISHED（已建立连接）状态,可以携带数据，如果不携带数据则不消耗序号，当服务器B收到A的确认后，也进入ESTABLISHED状态，至此，TCP建立连接的三次握手结束。

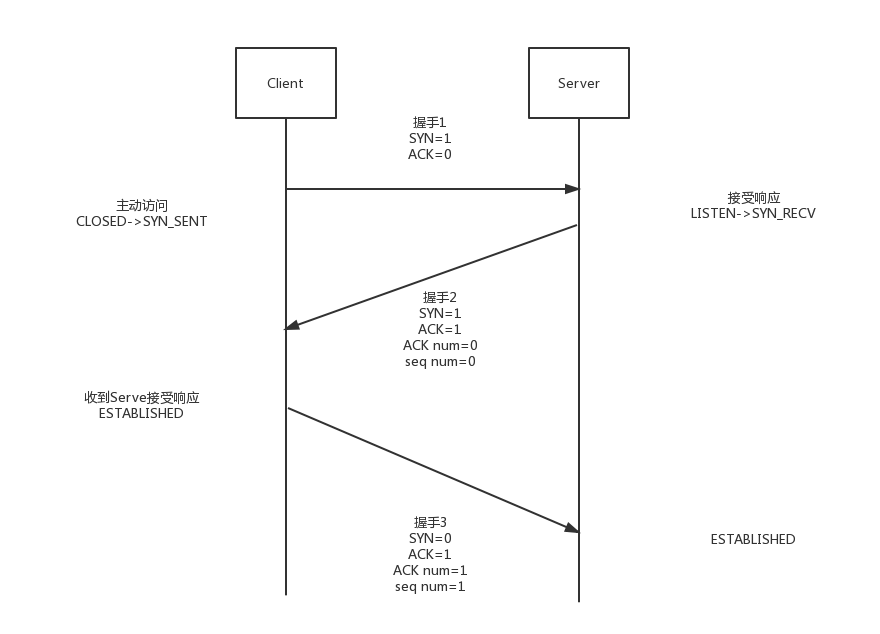


图6-22 TCP连接握手图

**任务三： 分析 TCP 连接释放阶段的四次握手（挥手）**

继续观察任务一中捕获到的 TCP 事件， 完成以下几项内容：

* 1. 分析 TCP 连接释放阶段的四次握手的过程；
     1. 客户端发送一个FIN报文给服务器，表示我将关闭客户端到服务器端这个方向的连接。
     2. 服务器收到报文4后，发送一个ACK报文给客户端，序号为报文4的序号加1。
     3. 服务器发送一个FIN报文给客户端，表示自己也将关闭服务器端到客户端这个方向的连接。
     4. 客户端收到报文6后，发回一个ACK报文给服务器，序号为报文6的序号加1。
  2. 查看 TCP 报文段首部中的各项字段的值，包括 SYN 字段、ACK 字段、PSH 字段、FIN字段、sequence number（序号） 字段、 ACK number（确认号）字段、窗口大小、选项字段 MSS（最大报文段长度）、报文段长度等；

表6-2 TCP释放过程

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | SYN | ACK | PSH | FIN | sequence number | ACK number | 窗口大小 | MSS | 数据段长 |
| TCP握手过程1 | 0 | 1 | 0 | 1 | 99 | 472 | 65535 | OPTION | 0 |
| TCP握手过程2 | 与第三次握手合并 | | | | | | | | |
| TCP握手过程3 | 1 | 1 | 0 | 0 | 472 | 100 | 15913 | OPTION | 0 |
| TCP握手过程4 | 0 | 1 | 0 | 0 | 100 | 472 | 65534 | OPTION | 0 |

* 1. 分析四次握手过程中 TCP 连接状态的变迁。



图6-23 整个TCP流程图

**思考：**

1. 连接建立阶段的第一次握手是否需要消耗一个序号？ 其 SYN 报文段是否携带数据？ 为什么？ 第二次握手呢？
   1. SYN报文段不能携带数据，但会消耗一个序号
   2. 第二次握手SYN报文段可以携带数据，如果不携带数据则不消耗序号
2. 本实验中连接释放过程的第二、三次握手是同时进行的还是分开进行的？ 这两次握手何时需要分开进行？
   1. 因为TCP连接是全双工的，即数据可在两个方向上同时传递，所以进行关闭时每个方向上都要单独进行关闭，另一个对此进行确认；
   2. 需要分开的时候是在客户端A没有要发送的数据，但是还有需要接受的数据。
3. 本实验中连接释放阶段的第三次握手，PC 向Server发送最后一个TCP确认报文段后，为什么不是直接进入CLOSED（已关闭）连接状态，而是进入 CLOSING（正在关闭）连接状态？

等待所有的数据接收完毕，同时进行第四次释放握手(挥手)

1. 本实验中 TCP 连接建立后的数据通信阶段， PC 向 Server 发送的了多少数据？Server向 PC 发送的数据呢？、

PC->Server：98个字节

Server->PC：471个字节

## 6.5 实验中的问题及心得

在实验过程中，进行了客户端对服务端的访问模拟，其中涉及DNS,HTTP等应用层协议，DNS下层为UDP协议，HTTP下层为TCP协议，而再次过程中，对协议过程进行了分析。

TCP会话的每一端都包含一个32位（bit）的序列号，该序列号被用来跟踪该端发送的数据量。每一个包中都包含序列号，在接收端则通过确认号用来通知发送端数据成功接收,当某个主机开启一个TCP会话时，他的初始序列号是随机的，可能是0和4,294,967,295之间的任意值，然而，PacketTracer和Wireshark默认显示的都是相对序列号/确认号，而不是实际序列号/确认号，相对序列号/确认号是和TCP会话的初始序列号相关联的。这是很方便的，因为比起真实序列号/确认号，跟踪更小的相对序列号/确认号会相对容易一些。

而对于DNS，服务器在其上只有一个网卡接口，故添加网卡进行级联，发现无法进行转发，可见服务器没有路由转发功能，其进行级联的操作为通过源端口进行网关转发，其源地址不变，而更改目的地址，对于DNS的操作过程为本地缓存记录、区域记录、转发域名服务器、根域名服务器。

整个实验过程中，复现了TCP和UDP过程，并对TCP其上的HTTP和UDP其上的DNS协议进行了观察，加深了理解。

## 参考文献

1. CiscoPacketTracer网络实验手册
2. DNS wiki

(https://en.wikipedia.org/wiki/Domain\_Name\_System)