

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 运输层实验

姓 名	班 级	学 号	得 分

教师评语：

## 一 实验环境

1. 实验环境：运行 Arch Linux x86\_64 操作系统的 PC 机一台
2. PacketTracer 版本：7.0.0.0202

## 二 实验目的

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。
3. 熟悉 UDP 与 TCP 协议的主要特点及支持的应用协议。
4. 理解 UDP 的无连接通信与 TCP 的面向连接通信。
5. 熟悉 TCP 报文段和 UDP 报文的数据封装格式。
6. 熟悉 TCP 通信的三个阶段。
7. 理解 TCP 连接建立过程和 TCP 连接释放过程。

## 三 实验内容及步骤

### 3.1 网络拓扑配置

打开 PacketTracer，配置网络拓扑如下图。

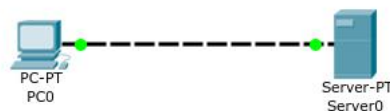


图 1 网络拓扑配置

其中，各台主机配置如下：

PC0: IP:192.168.1.2; Gateway:192.168.1.254; DNS Server:192.168.1.1

Server0: IP:192.168.1.2; Gateway:192.168.1.254

同时，检查 Server 端的服务配置，HTTP 和 DNS 的服务都需要打开。

### 3.2 运输层端口观察实验--事件捕获

点击 PC0/Desktop，点击浏览器 Web Browser，输入域名如下：

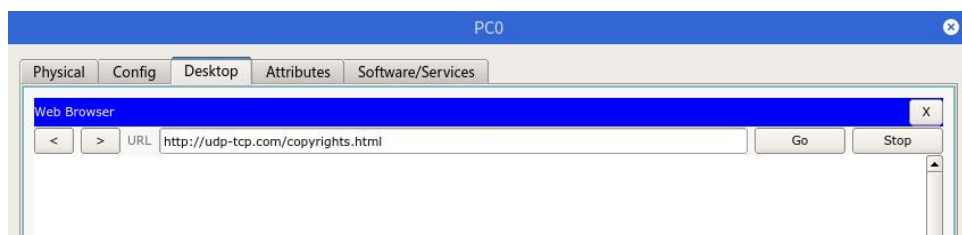
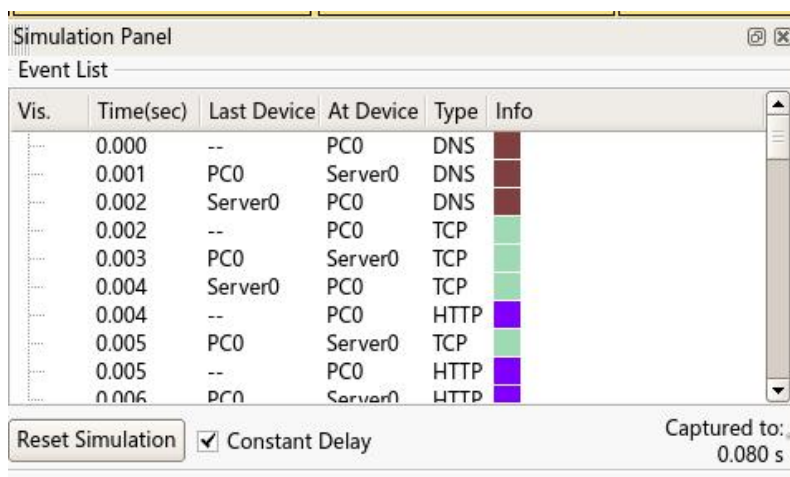


图 2 浏览器输入域名

不断点击 Capture/Forward，可以观察到，PC0 和 Server 之间经历了 ARP 协议获取 MAC 地址，DNS 协议解析域名，TCP 协议建立连接，HTTP 协议传输文本的整个过程，最后 Event List 中的事件如下：



Vis.	Time(sec)	Last Device	At Device	Type	Info
	0.000	--	PC0	DNS	
	0.001	PC0	Server0	DNS	
	0.002	Server0	PC0	DNS	
	0.002	--	PC0	TCP	
	0.003	PC0	Server0	TCP	
	0.004	Server0	PC0	TCP	
	0.004	--	PC0	HTTP	
	0.005	PC0	Server0	TCP	
	0.005	--	PC0	HTTP	
	0.006	PC0	Server0	HTTP	

Reset Simulation ☒ Constant Delay Captured to: 0.080 s

图 3 事件列表

可以观察到，在经历很多次 HTTP 报文传输之后，浏览器上最终出现了我们请求的结果。

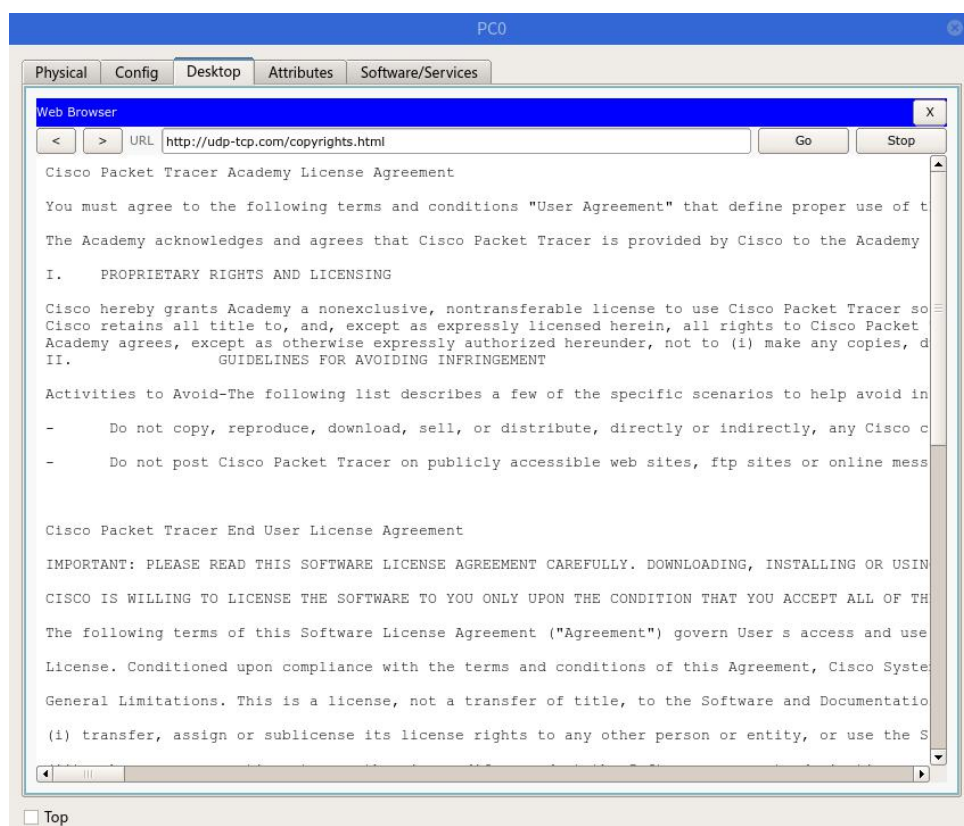


图 4 浏览器请求结果

有了事件列表信息，我们就可以分析 UDP/TCP 中的端口号了。

### 3.3 UDP 和 TCP 协议的对比分析

单击 Reset Simulation，重新进行一次浏览器请求，并捕获所有的报文。

### 3.4 TCP 连接管理分析

点击 Server，在 HTTP 服务中新建一个 test.html，里面内容只写上 Hello world! 作为实验文件。在 PC0 的浏览器中请求这个文件，在 Event List 中查看捕获到的事件，如下图：

Vis.	Time(sec)	Last Device	At Device	Type	Info
...	0.000	--	PC0	DNS	
...	0.001	PC0	Server0	DNS	
...	0.002	Server0	PC0	DNS	
...	0.002	--	PC0	TCP	
...	0.003	PC0	Server0	TCP	
...	0.004	Server0	PC0	TCP	
...	0.004	--	PC0	HTTP	
...	0.005	PC0	Server0	TCP	
...	0.005	--	PC0	HTTP	
...	0.006	PC0	Server0	HTTP	

Reset Simulation ☒ Constant Delay Captured to: 140.896 s

图 5 Event List 结果（一）

Vis.	Time(sec)	Last Device	At Device	Type	Info
...	0.004	--	PC0	HTTP	
...	0.005	PC0	Server0	TCP	
...	0.005	--	PC0	HTTP	
...	0.006	PC0	Server0	HTTP	
...	0.007	Server0	PC0	HTTP	
...	0.007	--	PC0	TCP	
...	0.008	PC0	Server0	TCP	
...	0.009	Server0	PC0	TCP	
...	0.010	PC0	Server0	TCP	

Reset Simulation ☒ Constant Delay Captured to: 140.896 s

图 6 Event List 结果（二）

从途中我们可以清晰的看到 TCP 连接时的三次握手过程，接下来在实验结果中对捕获到的数据包进行具体分析。

## 四 实验结果

### 4.1 通过捕获的 DNS 事件查看并分析 UDP 的端口号

在 Event List 中点击数据包，可以查看其详细信息，其中 DNS 请求和应答的详细信息如下：

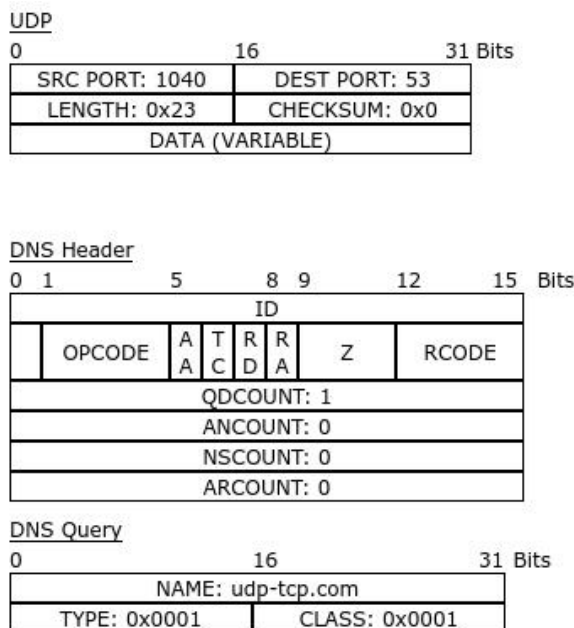


图 7 DNS 请求报文

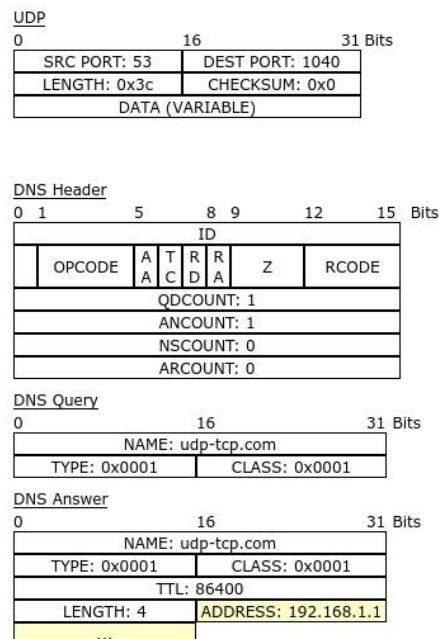


图 8 DNS 应答报文

观察发现 PC0 上的端口是 1040，Server 上的端口是 53，在一次 DNS 请求中是成对出现，没有变化的。此时 PC0 相当于客户端，Server 相当于服务器，因为 Server 使用的是 DNS 的周知端口 53。

重新回到 PC 机的浏览器窗口单击 Go 按钮再次请求相同的网页，再次捕获得到的结果如下：

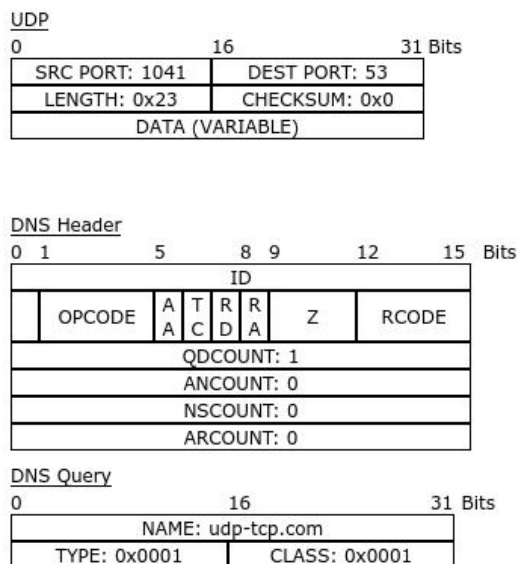


图 9 DNS 请求报文

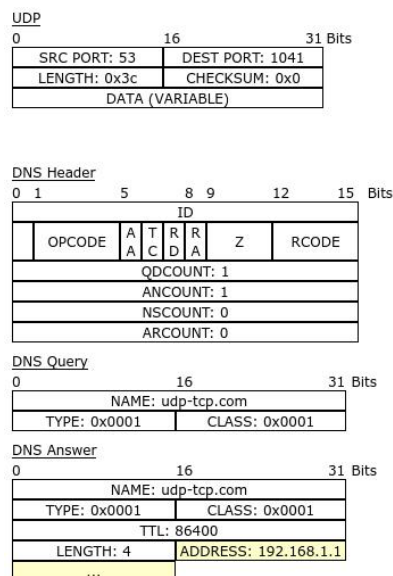


图 10 DNS 应答报文

观察发现，此时 PC0 上使用的端口变成了 1041，于是猜测 DNS 客户端使用端口的规律是依次递增 1。

于是进行多次浏览器请求/捕获的操作，最后观察端口从 1042,1043 开始逐渐递增，验证了这个猜测。

而服务端上使用的端口始终是周知端口 53。

## 4.2 通过捕获的 HTTP 事件查看并分析 TCP 的端口号

在 Event List 中点击数据包，可以查看其详细信息，其中选取一对 HTTP 请求和应答报文，如下图：

TCP					
0		16		31 Bits	
SRC PORT: 1033			DEST PORT: 80		
SEQUENCE NUM: 1					
ACK NUM: 1					
OFF.	RES.	PSH + ACK		WINDOW	
CHECKSUM: 0x0				URGENT POINTER	
OPTION				PADDING	
DATA (VARIABLE)					

HTTP	
Get /copyrights.html HTTP/1.1	
Accept-Language: en-us	
Accept: */*	
Connection: close	
Host: udp-tcp.com	

图 11 HTTP 请求报文

TCP					
0		16		31 Bits	
SRC PORT: 80			DEST PORT: 1033		
SEQUENCE NUM: 1					
ACK NUM: 116					
OFF.		RES.		ACK	
CHECKSUM: 0x0				WINDOW	
URGENT POINTER					
OPTION				PADDING	
DATA (VARIABLE)					

图 12 HTTP 应答报文

观察发现 PC0 上的端口是 1033，Server 上的端口是 80。此时 PC0 相当于客户端，Server 相当于服务器，因为 Server 使用的是 TCP 的周知端口 80。

## 4.3 分析运输层端口号

4.1 中 DNS 服务器端的端口号和 4.2 中服务器端的端口号并不相同，这是因为端口号在网络中唯一标识进程，DNS 服务器和 HTTP 服务器是两个不同的进程，因此端口号不会相同。

重新捕获 HTTP 事件，发现此时 PC0 上的端口由 1033 变成了 1034，反复操作发现端口号依次递增。于是归纳得到，运输层动态端口号的分配是从第一个端口号开始依次递增 1 的。

## 4.4 观察 UDP 与 TCP 协议的工作模式

首先观察 UDP 的无连接的工作模式。

运输层的 UDP 发送 DNS 请求之前，没有建立连接，一开始就发送了 DNS 报文，如下图：



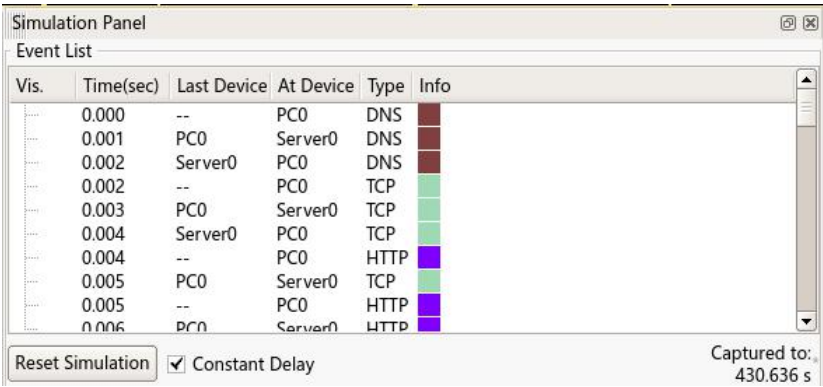


图 13 Event List（一）

UDP 首部中 LENGTH 字段的值为 0x23，即 35，而 UDP 首部有 8 个字节，因此其数据部分的长度为 27 个字节。

对于 TCP，最后一个 HTTP 事件之后的 Event List 如下图：

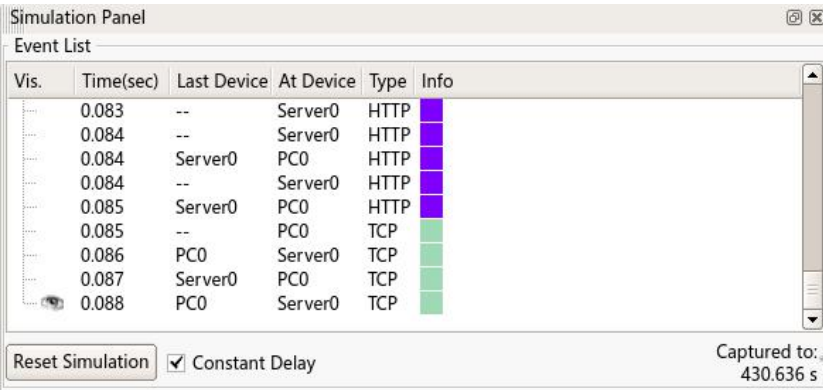


图 14 Event List（二）

从上面两张图可以看到，在第一个 HTTP 事件之前和最后一个 HTTP 事件之后，都存在 TCP 事件，记录最后两个 HTTP 报文如下图：

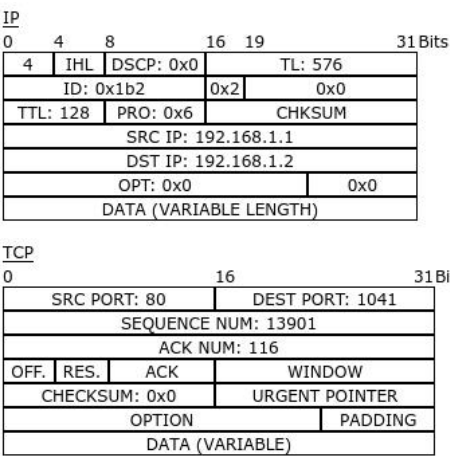


图 15 第一个 HTTP 报文

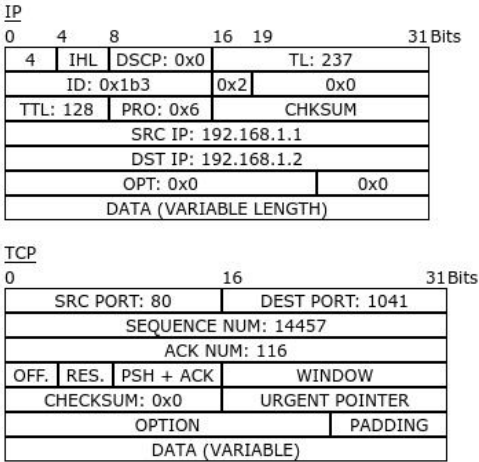


图 16 最后一个 HTTP 报文

两个报文之间的 sequence number 之差为 556，刚好是发过去的报文的 data

length 的大小（利用 IP 长度 576 减去首部 20 字节得到）

## 4.5 TCP 连接建立阶段的三次握手分析

点击 Event List 中的数据包，得到三次握手中的 TCP 数据包的详细信息如下图：

IP											
0		4		8		16		19		31 Bits	
4		IHL		DSCP: 0x0				TL: 44			
ID: 0xf2				0x2		0x0					
TTL: 128				PRO: 0x6				CHKSUM			
SRC IP: 192.168.1.2											
DST IP: 192.168.1.1											
OPT: 0x0								0x0			
DATA (VARIABLE LENGTH)											

TCP											
0						16				31 Bits	
SRC PORT: 1044						DEST PORT: 80					
SEQUENCE NUM: 0											
ACK NUM: 0											
OFF.		RES.		SYN		WINDOW					
CHECKSUM: 0x0						URGENT POINTER					
OPTION								PADDING			
DATA (VARIABLE)											

图 17 TCP 连接三次握手（一）

IP					
0	4	8	16	19	31 Bits
4	IHL	DSCP: 0x0	TL: 44		
ID: 0x1c0		0x2	0x0		
TTL: 128	PRO: 0x6		CHKSUM		
SRC IP: 192.168.1.1					
DST IP: 192.168.1.2					
OPT: 0x0				0x0	
DATA (VARIABLE LENGTH)					

TCP					
0	16	31 Bits			
SRC PORT: 80		DEST PORT: 1044			
SEQUENCE NUM: 0					
ACK NUM: 1					
OFF.	RES.	SYN + ACK	WINDOW		
CHECKSUM: 0x0			URGENT POINTER		
OPTION				PADDING	
DATA (VARIABLE)					

图 18 TCP 连接三次握手（二）

TCP									
0			16				31 Bits		
SRC PORT: 1044					DEST PORT: 80				
SEQUENCE NUM: 0									
ACK NUM: 0									
OFF.		RES.		SYN		WINDOW			
CHECKSUM: 0x0					URGENT POINTER				
OPTION							PADDING		
DATA (VARIABLE)									

TCP														
0					16					31 Bits				
SRC PORT: 80					DEST PORT: 1044									
SEQUENCE NUM: 0														
ACK NUM: 1														
OFF.		RES.		SYN + ACK			WINDOW							
CHECKSUM: 0x0					URGENT POINTER									
OPTION										PADDING				
DATA (VARIABLE)														

IP					
0	4	8	16	19	31 Bits
4	IHL	DSCP: 0x0	TL: 40		
ID: 0xf3			0x2	0x0	
TTL: 128	PRO: 0x6		CHKSUM		
SRC IP: 192.168.1.2					
DST IP: 192.168.1.1					
OPT: 0x0				0x0	
DATA (VARIABLE LENGTH)					

TCP					
0	16			31 Bits	
SRC PORT: 1044			DEST PORT: 80		
SEQUENCE NUM: 1					
ACK NUM: 1					
OFF.	RES.	ACK	WINDOW		
CHECKSUM: 0x0			URGENT POINTER		
OPTION				PADDING	
DATA (VARIABLE)					

图 19 TCP 连接三次握手（三）

分析 TCP 连接的建立如下：

1. PC0 发给 Server 一个 TCP SYN 报文，其中 SEQ 和 ACK 均为 0。
2. Server 令 ACK 等于收到的上一个报文的 SEQ 值+1，然后令自己的 SEQ 为 0，发送一个 TCP SYN+ACK 报文给 PC0。
3. PC0 收到报文之后，令 SEQ 为自己上一次发送的报文的 SEQ+1，ACK 等于收到的上一个报文的 SEQ+1，发送一个 TCP ACK 报文给 Server。

完成上述步骤之后，一个 TCP 连接就建立了。



分析 TCP 连接的状态变迁如下：

1. 开始时，PC0 和 Server 都处于 CLOSED 状态。
2. Server 被动打开，处于 LISTEN 状态。
3. PC0 发送一个 SYN 报文之后，进入 SYS\_SENT 状态。
4. Server 收到 SYN 报文之后，进入 SYN\_RCVD 状态，并向 PC0 发送 SYN+ACK 报文。
5. PC0 收到 SYN+ACK 报文之后，进入 ESTABLISHED 状态，并向 Server 发送 ACK 报文。
6. Server 收到 ACK 报文之后，进入 ESTABLISHED 状态，连接建立。

#### 4.6 TCP 连接释放阶段的四次握手分析

点击 Event List 中的数据包包，得到三次握手中的 TCP 数据包的信息如下图：

IP				31 Bits			
0	4	8	16	19			
4	4	8	16	19	31	Bits	
IHL: 0x0				DSCP: 0x0			
ID: 0x101				TL: 40			
TTL: 128				PRO: 0x6			
SRC IP: 192.168.1.2				CHKSUM			
DST IP: 192.168.1.1				OPT: 0x0			
DATA (VARIABLE LENGTH)							

图 20 TCP 连接释放四次握手（一）

IP				31 Bits			
0	4	8	16	19			
4	4	8	16	19	31	Bits	
IHL: 0x0				DSCP: 0x0			
ID: 0x1cc				TL: 40			
TTL: 128				PRO: 0x6			
SRC IP: 192.168.1.1				CHKSUM			
DST IP: 192.168.1.2				OPT: 0x0			
DATA (VARIABLE LENGTH)							

图 21 TCP 连接释放四次握手（二）

IP				31 Bits			
0	4	8	16	19			
4	4	8	16	19	31	Bits	
IHL: 0x0				DSCP: 0x0			
ID: 0x102				TL: 40			
TTL: 128				PRO: 0x6			
SRC IP: 192.168.1.2				CHKSUM			
DST IP: 192.168.1.1				OPT: 0x0			
DATA (VARIABLE LENGTH)							

TCP				31 Bits			
0	16						
0	16	31	Bits				
SRC PORT: 1045				DEST PORT: 80			
SEQUENCE NUM: 111				ACK NUM: 138			
OFF. RES. FIN + ACK				WINDOW			
CHECKSUM: 0x0				URGENT POINTER			
OPTION				PADDING			
DATA (VARIABLE)							

图 22 TCP 连接释放四次握手（三）

分析 TCP 连接的建立如下：

1. PC0 发给 Server 一个 TCP FIN 报文，其中 SEQ 和 ACK 分别为 110 和 138。
2. Server 令 ACK 等于收到的上一个报文的 SEQ 值+1，然后令自己的 SEQ 为 138，发送一个 TCP ACK 报文给 PC0。
3. Server 发给 PC0 一个 TCP FIN+ACK 报文，SEQ 和 ACK 和上一个发送的 ACK 报文一致。
4. PC0 收到报文之后，令 SEQ 为自己上一次发送的报文的 SEQ+1，ACK 等于收到的上一个报文的 SEQ+1，发送一个 TCP ACK 报文给 Server。

完成上述步骤之后，一个 TCP 连接就释放了。

在这里，步骤 2 和步骤 3 被合并为一个数据包一起发送了。

分析 TCP 连接的状态变迁如下：

1. 开始时，PC0 和 Server 都处于 ESTABLISHED 状态。
2. PC0 向 Server 发送 FIN 报文之后，进入 FIN\_WAIT\_1 状态。
3. Server 收到 PC0 发送的 FIN 报文之后，进入 CLOSE\_WAIT 状态。
4. Server 向 PC0 发送 FIN+ACK 报文之后，进入 LAST\_ACK 状态。
5. PC0 收到 Server 发送的 FIN+ACK 报文之后，进入 CLOSING 状态，并向 Server 发送一个 ACK 报文，进入 TIME\_WAIT 状态。
6. Server 收到 PC0 发来的 ACK 报文之后，进入 CLOSED 状态。
7. PC0 过一段时间（一般是两倍的报文寿命）之后，进入 CLOSED 状态，连接完全释放。

## 五 实验中的问题及心得

### 5.1 实验中的问题

1. 运输层如何区分应用层的不同进程？

通过端口来唯一标识进程，以此来区分不同的进程。

2. 若使用 Reset Simulation 按钮后再重新进程捕获，端口号如何变化？新的值与充值前有关吗？

有关，新的值仍然是之前的端口值加一，该值由系统维护，并不会因为重置模拟而重置。

3. TCP 报文首部中的序号和确认号有什么作用？

可以为字节流提供可靠的传输服务。

4. 无连接的 UDP 和面向连接的 TCP 各有什么优缺点？

无连接的 UDP 可以以任何速率向网络层注入数据，在能够容忍少量的数据

丢失，并不需要可靠的运输服务的实时应用中可以避免 TCP 拥塞控制机制和通信开销的不利影响。而 TCP 则为网络传输提供了可靠的传输服务，能够为进程数据传输提供无差错，按适当顺序交付的服务。TCP 还提供了拥塞控制服务，有利于提高因特网的整体性能，在电子邮件，文档下载等应用中使用效果很好。

5. 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？

从图 17 中可以看到，TCP 连接建立阶段的第一次握手需要消耗一个序号，但 SYN 报文段不需要携带数据，第二个报文段也同样要消耗一个序号，都不能携带数据。

6. 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？

本实验中第二次和第三次握手是同时进行的。当 PC0 释放连接之后，Server 仍要发送数据时，第二次和第三次握手就要分开进行。

7. 本实验中连接释放阶段的第四次握手，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 状态，而是进入 CLOSING 连接状态？

这是因为，此时 TCP 出现了一种同时关闭的情况。即 PC0 在接受到 Server 发送的 FIN，并发送了 ACK 之后，还没有接收到 Server 对自己的 FIN 的 ACK，此时就进入了 CLOSING 状态。这也与实际情况相符，一次 HTTP 请求结束，客户端和服务端同时关闭 TCP 连接。

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

HTTP 请求和应答的报文如下图：

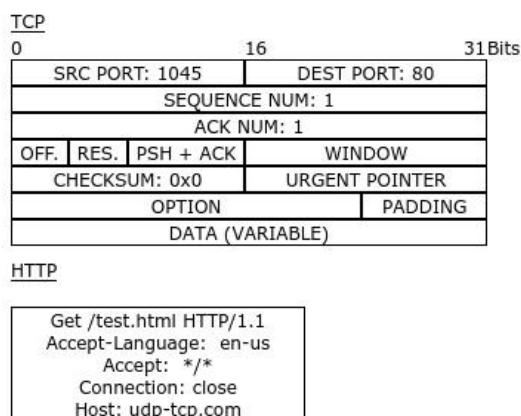


图 23 HTTP 请求

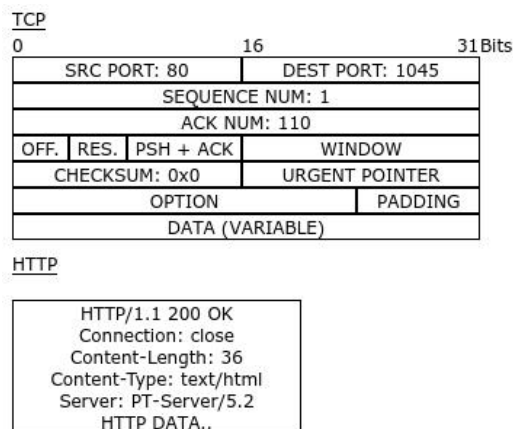


图 24 HTTP 应答

将上述两个报文的 Sequence Num 和 TCP 连接释放是报文的 Sequence Num 相比，即可计算 PC 向 Server 发送的数据为长度  $110 - 1 = 109$  字节，Server 向 PC 发送的数据长度为  $138 - 1 = 137$  字节。

## 5.2 实验总结

本次实验相比之前的实验复杂了一些，主要是涉及到 TCP 相关的各种细节，处理要比之前麻烦一些，而且也涉及到一些值的计算。在实验中一开始怎么也找不到 TCP 连接释放时的四次握手，查阅资料后才发现原来是存在一种同时关闭的状况，并查阅资料了解了具体的情况。

通过本次实验，我实际见证了 TCP 的连接建立的具体的一步步的情况，熟悉了 TCP 和 UDP 的工作模式，对上课讲过的东西有了更深一步的认识。