

**计算机网络**

**课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 利用Wireshark进行协议分析 | | | | | |
| 姓名 | 朱明彦 | | 院系 | 计算机科学与技术学院 | | |
| 班级 | 1603109 | | 学号 | 1160300314 | | |
| 任课教师 | 李全龙 | | 指导教师 | 李全龙 | | |
| 实验地点 | 格物213 | | 实验时间 | 2018/11/17 | | |
| 实验课表现 | 出勤、表现得分(10) |  | 实验报告  得分(40) |  | 实验总分 |  |
| 操作结果得分(50) |  |
| 教师评语 | | | | | | |
|  | | | | | | |

****

|  |
| --- |
| 实验目的： |
| 熟悉并掌握Wireshark的基本操作，了解网络协议实体间进行交互以及报文交换的情况。 |
| 实验内容： |
| * 1. 学习Wireshark的使用   2. 利用Wireshark分析HTTP协议   3. 利用Wireshark分析TCP协议   4. 利用Wireshark分析IP协议   5. 利用Wireshark分析Ethernet数据帧   **选做内容：**   * 1. 利用Wireshark分析DNS协议   2. 利用Wireshark分析UDP协议   3. 利用Wireshark分析ARP协议 |
| 实验过程： |
| 以文字描述、实验结果截图等形式阐述实验过程，必要时可附相应的代码截图或以附件形式提交。  在本次实验中，共使用了两台不同的电脑在不同的地点进行实验，其IP地址分别是：   1. Windows 10 主机：172.20.85.207      1. Ubuntu 16.04 主机：172.20.145.140/172.20.45.201   **（一）Wireshark的使用**  在Wireshark的使用环节，选择Ubuntu的无线网卡进行分组捕获，并访问<http://www.hit.edu.cn>。在完整的页面加载完成后，结束分组捕获。在这一段时间Wireshark捕获了本机所有利用该无线网卡与其他网络实体进行交换的报文，具体文件见1.pcapng.gz，实验结果的分析见**实验结果**部分。  **（二）HTTP分析**  1) HTTP GET/response交互  在Wireshark显示过滤部分输入“HTTP”即仅显示捕获的HTTP报文，开始捕获后访问<http://hitgs.hit.edu.cn/>，在加载完全部页面后停止分组捕获。在这一段时间，Wireshark捕获的全部分组见2\_http\_with\_buffer.pcapng.gz，实验结果的分析见实验结果部分。  2)HTTP 条件 GET/response交互  首先将浏览器内的所有缓存清空。  启动Wireshark分组捕获，访问<http://hitgs.hit.edu.cn/>，在加载完全部页面后，重新刷新页面；在刷新页面后，停止Wireshark分组捕获。在这一段时间Wireshark捕获的全部报文，见文件2\_http\_without\_buffer.pcapng.gz，实验结果的分析见**实验结果**部分。  **（三）TCP分析**  1. 访问<http://gaia.cs.umass.edu/wireshark-labs/alice.txt>，获得alice.txt文件。再打开<http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>，选择好本地alice.txt文件的位置，开始Wireshark分组捕获后，点击”Upload alice.txt file”按钮；在文件上传完毕后，停止Wireshark分组捕获。  2. 在这一段时间内Wireshark捕获的分组见3\_tcp\_final.pcapng.gz，在筛选规则中选择“tcp”部分，进行分析，具体分析见**实验结果**部分。  **（四）IP分析**  1. 使用pingplotter进行实验，启动Wireshark开始分组捕获，首先发送一系列56字节的包；再发送一系列2000字节的包；再发送一系列3500字节的包，然后停止Wireshark捕获。  2. 在这段时间捕获的数据包见文件4\_ip.gz.pcapng.gz，具体结果分析见**实验结果**部分。  **（五）抓取ARP数据包**  1. 利用arp查看本机的ARP缓存表  2. 开始Wireshark分组捕获，在命令行中输入：ping 192.168.45.95  3. ping通之后利用停止Wireshark捕获，这段时间捕获的分组见5\_arp.pcapng.gz，具体结果分析见**实验结果**部分。  **（六）抓取UDP数据包**  启动Wireshark分组捕获，利用QQ给好友发送消息，消息发送结束后，停止分组捕获；这段时间捕获的报文分组见6\_udp.pcapng.gz，具体结果分析见**实验结果**部分。  **（七）利用Wireshark进行DNS协议分析**  首先清空dns缓存，在浏览器中访问<http://www.google.com.hk/>，进行Wireshark抓包，这段时间抓取的分组见7\_dns.pcapng.gz，具体的结果分析见**实验结果**部分。 |
| 实验结果： |
| 采用演示截图、文字说明等方式，给出本次实验的实验结果。  **（一）Wireshark的使用**  图见下页  可以看到，经过过滤后的第一条HTTP报文，即为向[www.hit.edu.cn](http://www.hit.edu.cn/)发送HTTP GET报文。可以看到分装在tcp段中的HTTP请求报文，相关信息见图片。    **（二）HTTP分析**  1) HTTP GET/response交互  利用Wireshark打开2\_http\_with\_buffer.pcapng.gz，捕获分组的截图如下    思考问题：  1. 浏览器运行的协议为HTTP/1.1；访问的服务器运行的HTTP协议版本号同为HTTP/1.1。  2. 浏览器向服务器指明其可以接收的对象为  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  3. 本机ip地址为 **172.20.45.140**（Ubuntu 主机ip地址）  服务器<http://hitgs.hit.edu.cn的ip>地址为**219.217.226.25**  4. 服务器向本机浏览器返回的状态码为**200。**  2) HTTP条件GET/response交互  思考问题：  清除浏览器缓存之后，直接访问<http://hitgs.hit.edu.cn>，发出的第一个HTTP GET请求如下：    1. 可以看到是没有if-modified-since头部信息的。  接下来的几个HTTP GET请求获得主页HTML中的其他元素，可以看到返回的状态代码均为200，如下：    2. 在前几个HTTP GET请求**是明确回复了文件内容**，如图：      可以看到，其回复的消息中有12个不同的数据帧发送，一共有40K多字节。紧接的GET请求也可以看到，如下图，请求一个css文件：    在刷新之后，可以看到请求同一个css文件的请求如下:    在这次请求中，可以看到使用了条件GET方法，在头部增加了if-modified-since一行。  3. **在条件GET中增加了if-modified-since；并且其后边所跟信息为缓存文件上次修改的时间.**  4. 服务器对于较晚的HTTP GET请求会的的状态代码为304。根据同一文件前后请求的差别，可以看到，如上图，最近请求服务器回复消息的长度仅有272字节，远远小于之前的长度，从而服务器并未明确返回文件的内容。  **（三）TCP分析**  在3\_tcp\_final.pcapng.gz中筛选tcp相关的报文，如下    思考问题：  1. 向gaia.cs.umass.edu服务器传送文件的客户端（源主机）主机的ip地址与tcp端口号为172.20.85.207以及9738.  2. gaia.cs.umass.edu服务器ip地址为128.119.245.12，接收端口号为80.    3. 初始化tcp连接的tcp syn报文段的序号为0（随机值）；该报文段将SYN标志位置为1，表示该报文段为SYN段用于tcp建立连接。  4. SYNACK报文段的序号为0（随机值）；acknowledgement字段为1，服务器通过SYN请求报文段的seq序号加1确定acknowledgement字段；在该报文段中，使用flags部分的ack和SYN标志位置为1表示，该报文段为SYNACK报文段。    5. 以上两张图，恰好反映了tcp连接的三次握手中的前两次，分别是客户机向服务器端发送SYN请求报文，以及服务器向客户机回复SYNACK报文；下图为客户机向服务器回复ack报文段：    可以根据回复的ack报文段中，ack端的内容为1（恰好为SYNACK报文段序号0加1所得），可知这是对SYN ACK报文段的回复，即**第三次握手**。  6.包含HTTP POST命令的tcp报文段信息如下：    **可以看到带有POST的tcp报文段的序号是1.**  7.第6个tcp报文段如下：    有关其具体信息如下：    即在第一帧发送0.852721s后发送该报文段。  对应的ack报文接收如下：    可以看到接收时间为第一帧发送后的1.096824s  7. 前6个报文段长度如下：    第1个报文段长度为681，其余5个均为1460.  8. 在整个的跟踪过程中，接收端**公示最小可用的缓存空间为65536，并且始终为65536.所以始终没有出现限制发送端发送的情况。**  9. 没有重传的片段，依据为**发送端的报文段序号始终在增加，没有出现重复发送某一个序号的报文段的情况，故没有重传的。**  **seq变化如下图：**    10. **吞吐率计算**      第一张图为建立连接后发送的第一个tcp段，第二张图为回复最后一个。  吞吐率计算  **（四）IP分析**  分析4\_ip.gz.pcapng.gz，如下：    思考问题：  1. 本机地址可以看到172.20.85.207  2. ip数据包的头具体如下：    可以看到，上层协议为ICMP（1）。  3.根据上图，可以看到ip数据包的头部大小为20字节；而ip数据包的整个大小为56字节，因此ip数据包的净载为36字节。  4. 关于ip数据包是否分片，观察flag区，可以发现    所以，**没有其余的帧并且帧的偏移为0，可以推断出没有进行分片。**  5. 观察ip数据包的详细信息可以发现    TTL、checksum和sequence number总是发生改变。  6. 必须保持常量的是版本号、首部长度、区分服务（Differentiated Services Field）以及协议（始终为ICMP）；必须改变的是TTL、checksum和sequence number，TTL为生存时间，每次转发必然改变，由于TTL的改变，checksum也会改变，sequence number是变化的区分不同的ICMP报文。  7. **Identification字段值，为2个字节呈递增形式。**  8. 由最近的路由器返回的ICMP time-to-live exceeded消息，形式如下：    identification段为0x0ad2，TTL为64  9. identification段变化，为了区分不同的ICMP time-to-live exceeded消息；但TTL保持不变，均为一次转发。  10. 当包的大小变为2000字节后，第一个ICMP echo request消息形式如下:    可以看到确实被分解成了不止一个数据包。  11. 第一个分片的详细信息如下：    ip头部可以看到有flag域中，是否有更多分片位被置为1，表示该分片不为最后一片。该分片的长度为1500字节。  对比第二个分片的信息如下：    可以看到这两个分配那的identification段相同，并且第二片为最后一组且有偏移。  12.改为3500字节后的信息如下：    可以看到原始数据被分成了3片。  **标志位部分和checksum部分发生了变化。**  **（五）抓取ARP数据包**  首先删除对应172.20.45.95的ARP表项，删除后的ARP如下：    思考问题：  1. ARP表中，每一列均由其表头决定，如地址等；每一行为某地址对应的硬件地址。如上图所示。  在命令行中ping 172.20.45.95，利用Wireshark抓包，其中的ARP表项如下：    可以看到第一条为ARP请求报文，广播“谁是172.20.45.95告诉172.20.45.140（本机地址）”；第二条为回复报文，将172.20.45.95对应的Mac地址返回。  2.  1)ARP数据包的格式如下：    对应在实际数据包中，表现为：    总共有10部分组成，其中每一部分的所占字节数如上图。  2) 判断一个ARP数据包为请求包还是应答包，仅仅需要判断op部分，当其值为0x0001时为请求，0x0002时为应答，如下：      3. ARP查询要在广播帧中传送，是因为不知道具体的ip地址对应的mac地址，需要通过ARP查询得到ip地址对应的mac地址；而由于响应报文能从ARP查询的ip数据包中，拆分出发送查询的主机的ip地址和mac地址，所以不需要再利用广播帧发送信息。  另外在ping 172.20.45.95之后，ARP表更新为，可以看到其中增加了172.20.45.95表项。    **（六）抓取UDP数据包**  捕捉的UDP数据包如下：    思考问题：  1. 可以看到消息是基于UDP的  2. 本机ip地址为172.20.85.207，目的主机ip地址为182.254.110.91  3. 发送消息的端口号为8000，接收端口号为4001  udp数据包形式    4. 由上图可以看到UDP数据包的格式，头部包含源端口号、目的端口号、长度和checksum组成。各占1个字节。  5. 因为UDP是不可靠的数据传输，需要上层协议实验可靠数据传输，因此每次发送ICQ报文后又回复一个ICQ数据包。  UDP是无连接的，可以看到发送数据之前没有连接的建立过程，与TCP不同，因此为无连接数据传输。  **（七）利用Wireshark进行DNS分析**  dns抓包结果如下：    可以看到查询的目的地址均为相同的202.118.224.100，可以知道这是哈工大的dns服务器，经过ip地址查询确实这是一个来自哈尔滨市南岗区的教务网ip地址。 |
| 问题讨论： |
| （一）Wireshark的使用部分 [思考问题](#Wireshark的使用)  （二）HTTP的分析 [思考问题](#HTTP)  （三）TCP的分析 [思考问题](#TCP)  （四）IP的分析 [思考问题](#IP)  （五）ARP的分析 [思考问题](#ARP)  （六）UDP的分析 [思考问题](#UDP) |
| 心得体会： |
| 通过Wireshark分析各种网络协议的执行，可以加深对于各种协议交互过程的理解。并且对于协议中各个字段的作用有了更深的了解。  Wireshark是个十分强大的工具，不仅仅在本门实验中会利用到他的一些知识，在以后的学习工作生活中还有很大的作用。 |