**武汉大学计算机学院**

**实验报告**

**课程名称 Computer Networking\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**专业年级 Software Engineering\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**姓 名 Parkin Arsanamnee\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**学 号 2022326660017\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**协 作 者 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**实验学期 2024 学年 第二 学期**

**课堂时数 课外时数**

**填写时间 2024 年 5 月 16 日**

Catalogue

1. Ping and Tracert 3

2. Wireshark Introduction 8

3. Wireshark HTTP 13

4. Wireshark DNS 27

5. Wireshark UDP 46

6. Wireshark TCP 52

7. Wireshark IP 62

7. Wireshark Ethernet and ARP 75

**1) Ping and Tracert**

|  |
| --- |
| Experiment overview |
| [Experimental project name]:  Ping and Tracert |
| 【Purpose】:  The purpose of a ping and tracert lab is to provide students with practical experience in troubleshooting network connectivity issues by using utilities such as ping to test reachability and measure latency, and tracert to trace the route packets take between networked devices, aiding in understanding network topology and diagnosing routing problems.  [Experimental environment] (Software used):  Window 10 Operation System  [Reference materials]:  Lab instruments |
| **Experiment content** |
| [Experimental plan design]:  The experiment is divided into two main parts consists of Pin and Tracert part:  **Part 1: Analysis of Ping Functionality:**  1. Open the command prompt:  -On Windows: Press Win + R, type "cmd", and press Enter  2. Type the following command:  ping [IP address or hostname]  3.Press Enter.  4. The ping utility will send ICMP echo requests to the specified IP address or hostname and display the results, including round-trip times and packet loss.  By analyzing the output of the ping command, you can diagnose network connectivity problems, identify potential bottlenecks, and troubleshoot issues affecting communication between devices on a network.  **Part 2: Analysis of Tracer Functionality:**  1. Open the command prompt:  -On Windows: Press Win + R, type "cmd", and press Enter.  2. Type the following command: tracert [IP address or hostname]  Replace [IP address or hostname] with the IP address or hostname of the destination computer or website you want to trace.  3. Press Enter.  4. The tracert utility will start tracing the route to the specified IP address or hostname, displaying each hop along the path and the time it takes to reach that hop.  These steps allow you to use the ping and tracert utilities to troubleshoot network connectivity and trace network routes in a command-line interface.  【Conclusion】(Result):  **Ping another computer**  1. Turn on CMD:    2. Type the following command:  ping [IP address or hostname]    The result of the ping process that was shown in my computer means there is a connection between my computer and another computer or server. My computer sent four packets and all of them reached the destination.  **Tracert a server**  1. Turn on CMD:    2. Type the following command: tracert [IP address or hostname]  Replace [IP address or hostname] with the IP address or hostname of the destination computer or website you want to trace.  After completing the tracert process, the route to the specified destination has been successfully traced, providing insights into the network path and latency between the source and destination. |
| 【summary】:  The purpose of this experiment was to explore the functions of Ping and Tracert in computer networking.  In the first part of the experiment, the ping lab provides students with hands-on experience in testing network connectivity and diagnosing issues using the ping utility. Through practical exercises, students learn to interpret ping results, analyze round-trip times (RTT), assess packet loss, and identify potential network congestion or connectivity problems. Additionally, they gain insights into troubleshooting techniques and best practices for ensuring stable and reliable network communication.  In the second part of the experiment, students gain practical experience in tracing the route that packets take between networked devices using the tracert utility. Through this hands-on exercise, students learn to analyze network topology, identify the path packets travel through routers or hops, assess latency at each hop, and diagnose routing issues such as loops, misconfigurations, or bottlenecks. The tracert lab enhances students' understanding of network infrastructure and equips them with valuable troubleshooting skills essential for network administration and IT support roles.  Overall. This experiment gave us useful insights into how Ping and Tracert work. We discovered that Ping helps check if hosts can exchange data packets, while Tracert shows us the route these packets follow. Understanding these tools helps us with computer networking and could help solve network problems later on. |
| Instructor’s comments and results |
| [Comment]:          Score: Signature of instructor:  Approval date: |

**2) Wireshark Introduction**

|  |
| --- |
| Experiment overview |
| 【Experimental project name】:  Wireshark Introduction |
| 【Purpose】:  The aim of this experiment was to introduce participants to Wireshark, a tool for capturing and analyzing network packets. Through hands-on practice, participants learned how to use Wireshark to capture and examine messages sent between different protocol entities. This provided them with a deeper understanding of how protocols work and the procedures involved in transmitting data across networks  [Experimental environment] (Software used):  1. Wireshark  2. Window 10 Operating System  [Reference materials]:  1. Wireshark Official Website: http://www.wireshark.org/  2. Wireshark User Guide：http://www.wireshark.org/docs/wsug\_html\_chunked/  3. Wireshark Manual：http://www.wireshark.org/docs/man-pages/  4. Wireshark Frequently Asked Questions：<http://www.wireshark.org/docs/faq.html> |
| Experiment content |
| [Experimental plan design]:  1. Outline Design:  This experiment is separated into two sections consists of:  1. Downloading and installing Wireshark  - Download the Wireshark packet sniffer from the website.  - Install Wireshark on the computer (Windows 10).    2. Testing Wireshark functionality:  - Launch Wireshark and access the Capture menu.  - Select the appropriate interface and start capturing packets.  - Monitor the captured packets in the Wireshark windows.  - Open a web browser and perform a search using the HTTP protocol (e.g., enter the URL: <http://cs.whu.edu.cn/>).  - Stop the packet capture in Wireshark.  - Analyze the packet exchange during the web activity by applying a display filter for "http" in the main Wireshark window.  - Identify the HTTP GET message sent from your computer to the cs.whu.edu.cn HTTP server.  - Answer any related questions.  **2. Detailed Design**  1. Introduction:  - Provides an overview of the experiment and its goals.  - Emphasizes the importance of observing network protocols and explains the role of packet sniffers like Wireshark in analyzing network traffic.  2. Setup the Wireshark program:  - Explains the requirements for running Wireshark, including supported operating systems and necessary libraries.  - Instruct participants to visit this link <http://www.wireshark.org/download.html> in order to download and install the Wireshark binary appropriate for the computer.  - Offers additional resources such as user guides and FAQs for further reference.  3. Running Wireshark:  - Guides participants through launching Wireshark on their computers.  - Explains the initial startup screen and its components.  - Instructs participants to select the appropriate interface (e.g., Ethernet or Wi-Fi) for packet capture.  4. Packet Capture:  - Instructs participants on how to start packet capture and choose the desired interface.  - Explains the packet-listing window, its columns, and sorting capabilities.  - Describes the packet-header details window and packet-contents window.  - Demonstrates the packet display filter field for filtering packets based on protocols  5. Capturing Interesting Packets:  - Instructs participants to generate network traffic (e.g., by browsing a specific URL) to capture packets.  - Emphasizes that Wireshark captures all packets sent/received during the browsing session.  - Guides participants to stop the packet capture once desired network traffic is captured.  6. Analyze Capture Packets:  - Instructs participants to analyze captured packets to answer specific questions provided.  - Provides examples of questions participants need to answer, such as identifying protocols, calculating time intervals, and finding Internet addresses.  7.Experiment Completion:  - Concludes the experiment by stressing the importance of packet analysis for understanding network protocols.  - Encourages participants to explore Wireshark further and utilize its capabilities for future experiments and network troubleshooting.  【Conclusion】(Result):  Experiment Phenomenon:  1. Exploring Protocol Message Exchange with Wireshark: Using Wireshark, a tool that captures network data, we could see the messages passing between different parts of the network. This helped us look closely at how the network works and understand how messages are sent and received.  2. Experimenting in Different Network Settings: We tried out our experiments on our computers, pretending different situations and also using the real Internet. With Wireshark, we could watch how the network works and send and receive messages with other parts of the Internet.  3. Exploring Various Network Environments: We tested our experiments on our computers, pretending different situations and also using the real Internet. With Wireshark, we could see how the network works and talk to other parts of the Internet by sending and receiving messages.  4. Understanding How Packet Sniffers Work: We looked at how packet sniffers work, which are important for watching messages between different parts of the network. A packet sniffer grabs messages going in and out of our computer. It shows us what's inside these messages, like the different parts of the message that tell the network what to do. It just watches and doesn't send out any messages itself.  5. Understanding Packet Sniffer Components: The packet sniffer has two main parts: the packet catcher and the packet reader. The catcher grabs every message that goes in and out of our computer, letting us see messages from all apps and protocols. The reader knows how these messages are built and can show us what's inside them. It helps us understand how different parts of the messages work together.  6. Using Wireshark for Packet Sniffing: We used Wireshark, a free tool for looking at network messages, as our packet catcher in the lab. Wireshark is easy to use and can look at lots of different kinds of messages. It works on different computer systems and can understand different ways messages travel on the network, like through Ethernet or Wi-Fi.  Relative Question:  1. List 3 different protocols that appear in the protocol column in the unfiltered packet-listing window in step 7 above.  Answer: UDP, TCP, ARP  2. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received? (By default, the value of the Time column in the packet-listing window is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark *View* pull down menu, then select Time *Display Format*, then select *Time-of-day*.)  Answer:  - GET message: 21:33:44.066485  - HTTP OK: 21:33:44.070267  Time to send = 0.01 seconds  3. What is the Internet address of the cs.whu.edu.cn? What is the Internet address of your computer?  Answer:   * Address of cs.whu.ed.cn = 202.114.64.221 * Address of my computer = 10.131.166.249   4. Print the two HTTP messages (GET and OK) referred to in question 2 above. To do so, select *Print* from the Wireshark *File* command menu, and select the “*Selected Packet Only”* and *“Print as displayed”* radial buttons, and then click OK.  【summary】:  The experiment used Wireshark as a tool to watch and understand network messages. Wireshark saw the messages between different parts of the participants' computers, showing how the protocols worked. By looking at the messages Wireshark captured, participants learned more about how protocols work and the order they follow.  During the experiment, participants learned to use Wireshark to catch messages, see what's inside them, and use different parts of the Wireshark program. They focused on specific types of messages, like those used for browsing the web, and looked closely at what those messages contained. At the end of the experiment, participants answered questions based on what they saw in Wireshark, showing they understood how to use the tool and what the messages meant.  In conclusion, this lab gave participants a good start in understanding network messages. Using Wireshark helped them learn how to look at and understand these messages, which will be useful for future studies or tasks involving computer networks. |
| Instructor’s comments and results |
| [Comment]:          Score: Signature of instructor:  Approval date: |

4. Wireshark HTTP

|  |
| --- |
| Experiment overview |
| 【Experimental project name】:  Wireshark HTTP |
| 【Purpose】:  The goal of this experiment is to use the Wireshark packet analysis tool to explore the basic interactions and conditions involved in the HTTP protocol. By studying captured packets, we aim to achieve the following objectives:  1. Understand the fundamental interaction of the HTTP protocol, specifically the GET/response process.  2. Investigate the usage of conditional GET and analyze HTTP caching behavior.  3. Examine how long HTML documents are retrieved and how TCP segments are fragmented.  4. Explore the process of downloading HTML documents with embedded objects, comparing parallel and serial downloading.  5. Analyze HTTP authentication and examine the exchange of HTTP messages for websites requiring passwords.  [Experimental environment] (Software used):  1. Wireshark user-guide  2. Windows 10 Operating System  [Reference materials]:   1. Wireshark user-guide 2. Lab instruments |
| Experiment content |
| [Experimental plan design]:  1. Outline Design:  This experiment is separating into many tasks. However, each of them is focusing on a specific aspect of HTTP protocol.  The following contents are outlining design for the experiment:  Task 1: Understanding the Basic HTTP GET/Response Process  - Open a web browser and Wireshark packet sniffer.  - Capture HTTP messages by accessing a simple HTML file.  - Analyze captured packets to address questions about HTTP versions, language preferences, IP addresses, status codes, modification time, content size, and hidden headers.  Task 2: Exploring HTTP Conditional GET/Response  - Clear the browser cache and begin packet capture.  - Visit a URL and observe the HTTP GET requests and server responses.  - Analyze captured packets to address inquiries about conditional GET, server response, and HTTP status codes.  Task 3: Retrieving Lengthy Documents  - Clear the browser cache and initiate packet capture.  - Visit a URL containing a lengthy HTML file.  - Capture and examine HTTP GET and TCP response packets to ascertain the number of segments transmitted for the document  Task 4: Analyzing HTML Documents with Embedded Objects  - Clear the browser cache and begin packet capture.  - Visit a URL containing an HTML file with embedded images.  - Examine HTTP GET requests to various servers and observe whether downloading occurs in parallel or serially  Task 5: Exploring HTTP Authentication  - Open the browser and start packet capture.  - Visit a URL protected by a password and provide credentials.  - Study captured packets to investigate the server's responses, additional fields in subsequent HTTP GET requests, and how the username and password are encoded  2. Detailed Design:  1) The Basic HTTP GET/Response Interaction:  2) Open your web browser.  3) Launch the Wireshark packet sniffer and set the display filter to "http" for capturing only HTTP messages.  4) Start capturing packets with Wireshark.  5) Visit the website http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html in your browser.  6) Notice the simple, one-line HTML file appearing in your browser.  7) Stop capturing packets with Wireshark.  8) Examine the server response to confirm if the file contents were explicitly returned.  9) Check the "IF-MODIFIED-SINCE:" header content in the second HTTP GET request.  10) Identify the HTTP status code and phrase from the server's response to the second HTTP GET request and clarify if the contents were explicitly returned.  11) Retrieving Lengthy Documents:  12) Ensure your browser's cache is empty.  13) Open your web browser and launch the Wireshark packet sniffer.  14) Enter the URL http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html in your browser.  15) View the extensive US Bill of Rights displayed in your browser.  16) Stop capturing packets with Wireshark and set the display filter to "http".  17) Open your web browser and Wireshark packet sniffer.  18) Enter the URL http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html in your browser.  19) View the brief HTML file containing two images displayed.  20) Stop capturing packets with Wireshark and set the display filter to "http"  【Conclusion】(Result):  1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?  Answer: HTTP 1.1    2. What languages (if any) does your browser indicate that it can accept to the server?  Answer: en-us  3. What is the IP address of your computer? Of the gaia.cs.umass.edu server?  Answer: my computer = 10.131.205.123, gaia.cs.umass.edu server = 128.119.245.12    4. What is the status code returned from the server to your browser?  Answer: Status code = 200    5. When was the HTML file that you are retrieving last modified at the server?  Answer: Date = Thu, 16 May 2024 09:21:20 GMT  6. How many bytes of content are being returned to your browser?  Answer: 128 bytes    7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.  Answer: No, headers are not displayed in the packet window.  8. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?  Answer: Yes, I do. IF-MODIFIED-SINCE: Thu, 16 May 2024 05:59:01 GMT    9. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?  Answer: It returns 200 OK.    10. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header?  Answer: If-Modified-Since: Tue, 02 Apr 2024 05:59:01 GMT\r\n    11. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.  Answer: Status code: 200    12. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?  Answer: I saw 3 Get requests and packet number No.1993 contains the Bill of Rights.    13. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?  Answer: Packet number2157    14. What is the status code and phrase in the response?  Answer: Status code = 200    15. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?  Answer: [4 Reassembled TCP Segments (4861 bytes): #58(1380), #59(1380), #60(1380), #62(721)]    16. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?  Answer: 1GET request and it sent to destination 128.119.245.12  17. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.  Answer: I'm not sure if this concurrency is really sent at the same time, but GET packets are sent at different times, so I think they should be sent sequential not parallel.  18. What is the server’s response (status code and phrase) in response to the initial HTTP GET message from your browser?  Answer:  - [HTTP/1.1 401 Unauthorized\r\n]  - [Severity level: Chat]    19. When your browser’s sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?  Answer: Authorization: Basic cGFya2VyOjEyMw==\r\n, Credentials: parker:123 |
| 【summary】:  1. The HTTP protocol serves as a cornerstone for web communication, enabling clients to solicit resources from servers and obtain corresponding responses  2. Conditional GET requests streamline caching and minimize network congestion by enabling servers to issue a "Not Modified" response when the requested resource remains unchanged.  3. HTTP caching is vital for diminishing latency and enhancing performance by storing previously accessed resources locally. This practice enables retrieval from the cache rather than initiating full requests to the server  4. HTTP caching is essential for reducing latency and improving performance by storing previously accessed resources locally, allowing for retrieval from the cache instead of making full requests to the server.  5. When accessing HTML documents containing embedded objects, the browser has the option to download them either concurrently or sequentially. Concurrent downloading boosts performance by fetching multiple objects simultaneously.  6. HTTP authentication offers a method for secure access to password-protected websites by verifying user credentials through the exchange of HTTP messages.  In summary, this experiment has offered significant insights into the fundamental interactions and conditional retrieval processes of the HTTP protocol, underscoring its importance in web communication and resource handling |
| Instructor’s comments and results |
| [Comment]:          Score: Signature of instructor:  Approval date: |

4. Wireshark DNS

|  |
| --- |
| Experiment overview |
| [Experimental project name]:  Wireshark DNS |
| 【Purpose】:  The objective of this experiment is to acquire practical experience and comprehension of DNS (Domain Name System) functionality and its associated tools. Specifically, the experiment aims to accomplish the following goals:  1. Get Acquainted with nslookup: Learn the fundamental operation and application of the nslookup tool, which involves querying DNS servers for particular DNS records and interpreting the outcomes.  2. Investigate DNS Caching and Records: Explore the principles of DNS caching, local DNS servers, DNS records, DNS messages, and the TYPE field within DNS records.  3. Analyze DNS Packets with Wireshark: Utilize the Wireshark tool to capture and scrutinize DNS query and response messages, examining various attributes including transport protocol, port numbers, IP addresses, query types, and responses.  [Experimental environment] (Software used):  1. Wireshark  2. Windows 10 Operating System  [Reference materials]:  1. Wireshark user guide  2. Lab instruments |
| Experiment content |
| [Experimental plan design]:  1) Outline Design:  1.1) Initial Examination: Review Section 2.4 of the textbook, concentrating on DNS-related topics such as local DNS servers, DNS caching, DNS records and messages, and the TYPE field within DNS records.  1.2) nslookup Experiment: Objective are 1. Gain proficiency in nslookup's fundamental usage and syntax, 2. Execute various nslookup commands to observe outcomes, including querying a web server in China, identifying authoritative DNS servers for a European university, and querying DNS servers for Yahoo! mail servers,and 3. Document the IP addresses and responses retrieved from the nslookup commands.  1.3) ipconfig Experiment: Objectives are1. Familiarize yourself with the ipconfig utility for exhibiting TCP/IP details and overseeing DNS information saved on the host, 2. Utilize ipconfig to inspect the present TCP/IP details, comprising address, DNS server addresses, and adapter type, 3. Employ the ipconfig command to clear the DNS cache, and 4. Utilize ipconfig to exhibit DNS cache entries and observe the remaining Time to Live (TTL) for each entry.  1.4) TCP/IP with ipconfig: Objectives are 1. Gain familiarity with the ipconfig utility for displaying TCP/IP details and managing DNS information stored on the host, 2. Use ipconfig to examine current TCP/IP details, including address, DNS server addresses, and adapter type, 1.5) Clear the DNS cache using the ipconfig command, and 4. Use ipconfig to display DNS cache entries and observe the remaining Time to Live (TTL) for each entry.  1.5) nslookup Experiment with Packet Capture: The processes are 1. Commence packet capture utilizing Wireshark, 2. Conduct a nslookup on [www.stanford.edu.](http://www.stanford.edu.), 3. Cease packet capture and scrutinize the captured packets to address inquiries concerning destination/source ports, IP addresses, query types, and responses, 4.Print and furnish a screenshot of pertinent query and response messages.  1.6) Repeated nslookup Experiments with Varied Queries: Procedure are 1. Repeat the nslookup experiment using the commands "nslookup -type=NS stanford.edu" and "nslookup cs.stanford.edu argus.stanford.edu.", 2. Examine the captured packets and respond to specific inquiries regarding destination/source IP addresses, query types, answers, and Stanford nameservers, and 3. Print and present screenshots for each experiment.  2) Detailed Design  (1) nslookup  - open command prompt  - enter nslookup [www.whu.edu.cn](http://www.whu.edu.cn) to get the ip address for the host.  - enter nslookup –type=NS stanford.edu to get the host names of the authoritative DNS for stanford.edu.  - enter nslookup cs.stanford.edu argus.stanford.edu to restrict DNS server as argus.stanford.edu and get the ip address of cs.stanford.edu.    (2) ipconfig  - open command prompt  - enter ipconfig/all to get the current TCP/IP information    - enter ipconfig/displaydns to get DNS information  - enter ipconfig/flushdns to flush DNS information stored.    (3) Tracing DNS with Wireshark  1) Use ipconfig to empty the DNS cache in the host.  2) Open browser and empty the browser cache.  3) Open Wireshark and enter “ip.addr == your\_IP\_address” into the filter, where you obtain your\_IP\_address with ipconfig. This filter removes all packets that neither originate nor are destined to the host.  4) Start packet capture in Wireshark.  5) Open browser and go to Web page: <http://www.etf.org>  6) Stop packet capture and analyze the information    (4) Nslookup with Wireshark  1) Start packet capture  2) Do an nslookup on [www.stanford.edu](http://www.stanford.edu).    3) Stop packet capture.    【Conclusion】(Result):  Experiment Phenomenon:  In this lab, I learned about the usage of nslookup, ipconfig and how DNS performs through Wireshark.  1. Run nslookup to obtain the IP address of a Web server in China. What is the IP address of that server?  Answer: 183.2.172.42    2. Run nslookup to determine the authoritative DNS servers for a university in Europe.  Answer:  stanford.edu nameserver = ns7.dnsmadeeasy.com  stanford.edu nameserver = argus.stanford.edu  stanford.edu nameserver = atalante.stanford.edu  stanford.edu nameserver = ns6.dnsmadeeasy.com  stanford.edu nameserver = ns5.dnsmadeeasy.com  stanford.edu nameserver = avallone.stanford.edu    3. Run nslookup so that one of the DNS servers obtained in Question 2 is queried for the mail servers for Yahoo! mail. What is its IP address?  Answer: Address = 131.111.8.37    4. Locate the DNS query and response messages. Are they sent over UDP or TCP?  Answer: It is over TCP.    5. What is the destination port for the DNS query message? What is the source port of DNS response message?  Answer: Source Port = 49846    6. To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?  Answer: DNS query message sent Source: 10.131.191.53 and Destination: 202.114.96.1 Furthermore, they sent the same information.  Wireless LAN adapter Wi-Fi:  Connection-specific DNS Suffix . :  Description . . . . . . . . . . . : Intel(R) Wireless-AC 9560 160MHz  Physical Address. . . . . . . . . : D8-3B-BF-5B-1A-9E  DHCP Enabled. . . . . . . . . . . : Yes  Autoconfiguration Enabled . . . . : Yes  Link-local IPv6 Address . . . . . : fe80::b8e7:feb7:f398:efd5%17(Preferred)  IPv4 Address. . . . . . . . . . . : 10.131.191.53(Preferred)  Subnet Mask . . . . . . . . . . . : 255.255.128.0  Lease Obtained. . . . . . . . . . : Tuesday, April 9, 2024 9:41:02 PM  Lease Expires . . . . . . . . . . : Wednesday, April 10, 2024 1:37:25 AM  Default Gateway . . . . . . . . . : 10.131.255.254  DHCP Server . . . . . . . . . . . : 10.131.255.254  DHCPv6 IAID . . . . . . . . . . . : 282606527  DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-26-0C-2A-74-F8-75-A4-E2-D9-B8  DNS Servers . . . . . . . . . . . : 202.114.96.1  202.114.64.1  NetBIOS over Tcpip. . . . .  7) Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?  Answer: It is query type A and it did not contain any answers.  8) Examine the DNS response message. How many “answers” are provided? What do each of these answers contain?  Answer: There are three answers provided in my Wireshark and each of them contains domain name and hostname.    9) Consider the subsequent TCP SYN packet sent by your host. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?  Answer: Yes, it provides and the address is 199.16.156.11.  10) This web page contains images. Before retrieving each image, does your host issue new DNS queries?  Answer: No, it does not contain any images.  11) What is the destination port for the DNS query message? What is the source port of DNS response message?  Answer: Source Port = 56117, Destination Port = 53    12) To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?  Answer: Destination: 202.114.96.1 and it is the IP address of my local DNS server.      13) Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?  Answer: They said the query are types A but did not contain any answers.  14) Examine the DNS response message. How many “answers” are provided? What do each of these answers contain?  Answer: In my Wireshark, there are three answers and each of them contains IPV4 and IPV6 addresses of the server.    15) Provide a screenshot.  16) To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?  Answer: Destination = 202.114.96.1 and it is not the IP address of my local DNS server    17) Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?  Answer: Query type A and it does not contain any message.    18) Examine the DNS response message. What Stanford nameservers does the response message provide? Does this response message also provide the IP addresses of the Stanford name servers?  Answer: In my Wireshark, it did not show any message.  19) Provide a screenshot    20) To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server? If not, what does the IP address correspond to?  Answer: 202.114.96.1 and it is an IP address of domain name server called “argus.stanford.edu”.    21. Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?  Answer: The type of query is type A, moreover, it does not contain any message.    22. Examine the DNS response message. How many “answers” are provided? What does each of these answers contain?  Answer: In my Wireshark, it does not show any answers.  23. Provide a screenshot. |
| 【Summary】:  In summary, this experiment offered valuable practical exposure to DNS and its associated tools. Through executing nslookup commands, examining DNS packets with Wireshark, and exploring the ipconfig utility, I acquired a deeper understanding of DNS functionality, encompassing DNS queries, responses, caching, and record types. Moreover, the experiment bolstered my skills in utilizing nslookup and Wireshark for diagnosing DNS-related issues. Overall, this hands-on exercise significantly enriched my knowledge and comprehension of DNS systems and their role in computer networks. |
| Instructor’s comments and results |
| [Comment]:          Score: Signature of instructor:  Approval date: |

5. Wireshark UDP

|  |
| --- |
| Overview of the experiment |
| 【Experimental Project Name】:  Wireshark UDP |
| 【Purpose】:  The objective of this experiment is to examine UDP (User Datagram Protocol) packets utilizing Wireshark. The aim is to comprehend the structure of the UDP header, ascertain the length of header fields, recognize content and maximum payload size, investigate source port numbers, and analyze the correlation between port numbers in a set of UDP packets.  [Experimental environment] (Software used):  1. Wireshark  2. Windows 10 Operating System  [Reference materials]:  1. Wireshark user guide  2. Lab instruments |
| Experiment content |
| [Experimental plan design]:  1. Outline Design  - Start packet in order to capture in Wireshark.  - Start activities on the host that involve send and receive UDP packets.  - Stop packet capture.  - Set packet filter in Wireshark to show only UDP.  - Select one of UDP packet to analyze.  - Look for detail from the chosen UDP packet.  - Answer the questions based on information and analysis.  2. Detailed Design  1) Select *one* UDP packet from your trace. From this packet, determine how many fields there are in the UDP header and name these fields.  Answer: Select *one* UDP packet from your trace. From this packet, determine how many fields there are in the UDP header and name these fields.    2) By consulting the displayed information in Wireshark’s packet content field for this packet, determine the length (in bytes) of each of the UDP header fields.  Answer: The UDP header has a fixed length of 8 bytes. Each of them has 2 bytes long, so it is 64 bits.    3) The value in the Length field is the length of what? (You can consult the text for this answer). Verify your claim with your captured UDP packet.  Answer: The length field specifies the number of bytes in the UDP segment (header plus data). An explicit length value is needed since the size of the data field may differ from one UDP segment to the next. The length of UDP payload for selected packet is 1250 bytes. 1258 bytes - 8 bytes = 1250 bytes.    4) What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)  Answer: The maximum number of bytes that can be in the payload is 2^16- the bytes already being used by the header field (8 bytes). Maximum payload is 65535-8= 65527 bytes  5) What is the largest possible source port number? (Hint: see the hint in 4.)  Answer: The largest possible source port number is (2^16 – 1) = 65535.  6) What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation. To answer this question, you’ll need to look into the Protocol field of the IP datagram containing this UDP segment (see Figure 4.13 in the text, and the discussion of IP header fields).  Answer: The IP protocol number for UDP is 0x11 hex, which is 17 in decimal value.  7) Examine a pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). Describe the relationship between the port numbers in the two packets.  Answer:  Send Packet  Source Port: 54312  Destination Port: 443  Receiving Packet  Source Port: 443  Destination Port: 54312  The source (source) Port and Destination Port of the request and response correspond to each other. |
| 【summary】:  Overall, we utilized Wireshark to capture and analyze UDP packets. Through scrutiny of the UDP header fields, we identified their quantity and labels. The length of each field was determined by referencing the packet content in Wireshark. We verified that the Length field in the UDP header signifies the segment's length, comparing its value with the captured packet. By analyzing the header fields, we derived the maximum payload size of UDP. Utilizing this information, we calculated the largest feasible source port number. The protocol number for UDP was discerned in both hexadecimal and decimal formats by inspecting the Protocol field in the corresponding IP datagram. Finally, we examined a pair of UDP packets to grasp the correlation between port numbers, delineating the first packet as a request and the second packet as a reply. |
| Instructor’s comments and results |
| [Comment]:          Score: Signature of instructor:  Approval date: |

6) Wireshark TCP

|  |
| --- |
| Overview of the experiment |
| 【Experimental Project Name】:  Wireshark TCP |
| 【Purpose】:  1. Examine the functionality of the renowned TCP protocol meticulously by scrutinizing a trace of the TCP segments transmitted and received during the file transfer process from my computer to a remote server.  2. Explore how TCP employs sequence and acknowledgment numbers to ensure reliable data transmission.  3. Witness TCP's congestion control algorithm, including slow start and congestion avoidance, in operation.  4. Investigate TCP's receiver-advertised flow control mechanism.  5. Briefly delve into TCP connection establishment and evaluate the performance, including throughput and round-trip time, of the TCP connection between my computer and the server.  [Experimental environment] (Software used):  1. Wireshark  2. Windows 10 Operating System  [Reference materials]:  1. Wireshark user guide  2. Lab instruments |
| Experiment content  (1) Capturing a bulk TCP transfer from your computer to a remote server    (2) A first look at the captured trace  1) What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it’s probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the “details of the selected packet header window” (refer to Figure 2 in the “Getting Started with Wireshark” Lab if you’re uncertain about the Wireshark windows.  Answer: IP address = 10.131.205.123, TCP port number = 51581    2) What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?  Answer: IP address = 192.161.60.131, Port number = 80  3) What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?  Answer: IP address = 10.131.205.123, TCP port number = 51581    (3) TCP Basics  4) What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?  Answer: Sequence number of the TCP SYN segment = 0. The segment that identifies the segment as a SYN segment = “.... .... ..1. = Syn: Set”    5) What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?  Answer:  Sequence number of the TCP SYNACK segment = 0  Acknowledgement field in the SYNACK segment = Acknowledgment number (raw): 1067790315  How did gaia.cs.umass.edu determine that value? = ACK value for SYNACK is equal to sequence number of next ACK segment  What is it in the segment that identifies the segment as a SYNACK segment? = Both SYN and ACK flags are Set.    6) What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you’ll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a “POST” within its DATA field.  Answer: Sequence number = 1487649583    7) Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see Section 3.5.3 in textbook) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation in the textbook for all subsequent segments.  *Note:* Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the “listing of captured packets” window that is being sent from the client to the gaia.cs.umass.edu server. Then select: *Statistics->TCP Stream Graph->Round Trip Time Graph.*  Answer: What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? = Arrival Time: May 14, 2024 18:56:14.626742000 China Standard Time, Arrival Time: May 14, 2024 18:56:22.940641000 China Standard Time, Arrival Time: May 14, 2024 18:56:28.574848000 China Standard Time. At what time was each segment sent? = 18:56:28.574848000 China Standard Time. When was the ACK for each segment received? = 18:56:14.523021000 China Standard Time. Difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments = RTT is Receive time – Sent time = [The RTT to ACK the segment was: 0.000366000 seconds]  8) What is the length of each of the first six TCP segments?  Answer: Note that the maximum IP packet is 1500 bytes  1st = 720 bytes  2nd = 1460 bytes  3rd = 1460 bytes  4th = 1460 bytes  5th = 1460 bytes  6th = 1460 bytes    9) What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?  Answer:  Minimum amount of available buffer space advertised at the received for the entire trace = [Calculated window size: 2063]  Does the lack of receiver buffer space ever throttle the sender? = NO, because segment length is less than window size.  10) Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?  Answer: In my Wireshark there are sequence number is increasing without repeat, so it means no retransmitted occurred.  11) How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 250 in the text).  Answer:  How much data does the receiver typically acknowledge in an ACK? = 1460 bytes  Can you identify cases where the receiver is ACKing every other received segment? = No, I cannot.  12) What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.  Answer: First TCP segment is 1 byte and the last segment is 164091 bytes. So, total data is 164091 – 1 = 164090 bytes. The difference between first segment and last segment is 0.000366000 seconds. Now the Throughout is 164091 / 0.000366 seconds = 448336065.574  (4) TCP congestion control in action  13) Use the *Time-Sequence-Graph(Stevens*) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP’s slow-start phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we’ve studied in the text.  Answer: The slowstart phase begins at 0 seconds and ends at around 0.125 seconds according to the stevens graph. After this the congestion avoidance takes over and gradually increases TX rate till network limit is reached.    14) Answer each of two questions above for the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu  Answer: |
| 【summary】:  In conclusion, the TCP Wireshark lab offered practical insights into TCP behavior using Wireshark. Participants analyzed TCP segments during file transfers, observed reliability mechanisms like sequence and acknowledgment numbers, and explored congestion control algorithms and flow control mechanisms. They also investigated TCP connection setup and performance metrics like throughput and round-trip time, enhancing their understanding of TCP's role in network communication. |
| Instructor’s comments and results |
| [Comment]:          Score: Signature of instructor:  Approval date: |

7. Wireshark IP

|  |
| --- |
| Experiment overview |
| [Experimental project name]:  Wireshark IP |
| 【Purpose】:  1. Investigate the IP protocol by analyzing a trace of IP datagrams sent and received during the execution of the traceroute program.  2. Understand the functions of the IP protocol and gain a comprehensive understanding of each field within the IP datagram.  3. Examine the various fields in the IP datagram and study IP fragmentation in detail. |
| [Experimental environment] (Software used):  1. Wireshark  2. Window 10 Operating System  [Reference materials]:  1. Wireshark user guide  2. Lab instruments |
| Experiment content |
| [Experimental plan design]:  1. Outline Design:  This experiment is mainly divided into two parts:  - Use Wireshark to capture packets from a traceroute execution.  - Examine these packets to understand the fields of an IP datagram and to get an initial look at ICMP.  2. Detailed Design:  (1) Capturing packets from an execution traceroute  - Download PingPlotter 5 and do traceroutes in PingPlotter to test its correctness.    - Start Wireshark program and start packet capture then press OK on the Wireshark Packet Capture Options screen.  - Open PingPlotter and enter the target destination in the "Address to Trace" field. Navigate to Edit -> Options -> Default Settings -> Engine, set the Packet Size to 56, and click OK. Then, press the Trace button.  - To send a set of longer datagrams, go to Edit -> Options -> Default Settings -> Engine, set the Packet Size to 2000, and click OK. Then, press the Resume button.  - Send another set of longer datagrams by selecting Edit -> Options -> Default Settings -> Engine, set the Packet Size to 3500, and click OK. Then, press the Resume button.  - Stop the Wireshark capture    (2) Take a look at the captured trace  1) Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?    Answer: My IP address = 192.168.1.102    2) Within the IP packet header, what is the value in the upper layer protocol field?  Answer: The value in the upper layer protocol field is ICMP (1)    3. How many bytes are in the IP header? How many bytes are in the payload *of the IP datagram*? Explain how you determined the number of payload bytes.  Answer: The IP header is 20 bytes and the total length is 84. Therefore, the payload of IP datagram is 84 -20 = 64 bytes.    4.Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.  Answer: The datagram has not been fragmented because the More fragment bit is 0, and the offset field is 0 as well    5.Which fields in the IP datagram *always* change from one datagram to the next within this series of ICMP messages sent by your computer?  Answer: Using the first datagrams as an example, we observe that the TTL field consistently decreases, while the Header checksum and Identification fields continually change    6.Which fields stay constant? Which of the fields *must* stay constant? Which fields must change? Why?  Answer: The following fields remain constant: Source, Destination, Protocol, Header Length, and Version. The fields that must stay constant are: Source, Destination, Protocol, and Version. These fields remain unchanged because we are using the same protocol to test communication between my computer and a single target.  The fields that must change include:   * Header checksum, as the payload of each datagram is different. * TTL, because traceroute manually increments it one by one. * Identification, since each ICMP datagram must have a unique ID number   7.Describe the pattern you see in the values in the Identification field of the IP datagram  Answer: The value in the Identification field increases by 1 with each subsequent datagram to distinguish them from one another.    8. What is the value in the Identification field and the TTL field?  Answer: The Identification is 0x9d7c, and the TTL is 255.  9. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?  Answer: The Identification field will change to distinguish each datagram from the others. However, the TTL field will remain the same because for each TTL-exceeded reply, the TTL value is exactly the upper limit. Sort the packet listing by time again by clicking on the Time column.    10. Find the first ICMP Echo Request message that was sent by your computer after you changed the *Packet Size* in *pingplotter* to be 2000(No. 1027, 1026 in the example data). Has that message been fragmented across more than one IP datagram? If your computer has an Ethernet interface, a packet size of 2000 *should* cause fragmentation.  Answer: Yes, the message has been fragmented into two IP datagram    11. Print out the first fragment of the fragmented IP datagram. What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?  Answer: From the "More fragments" field, we can determine that fragmentation has occurred. Additionally, the "Offset" field, which is filled with 0, indicates that this is the initial fragment. Its total length is 1500.    12. Print out the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?  Answer: We can infer from the offset field filled with 148  0 that this is not the first segment. Furthermore, since the "More fragments" field is 0, there are no additional fragments.    13. What fields change in the IP header between the first and second fragment?  Answer: Now locate the initial ICMP Echo Request message transmitted by your computer subsequent to adjusting the Packet Size in PingPlotter to 3500, focusing on the Total Length, Fragment Offset, and Flags (More fragments).    14. How many fragments were created from the original datagram?  Answer: 3 fragments which are 216, 217, and 218  15. What fields change in the IP header among the fragments?  Answer: By comparing them, we can discern differences in the Total Length, Flags, Fragment Offset, and Header checksum. |
| 【summary】:  During this lab, we utilized Wireshark to capture packets while performing traceroute operations. Through the analysis of these packets, we obtained insights into the various fields of an IP datagram and developed an initial understanding of ICMP.  Upon finishing this lab, I gained knowledge about the basic mechanisms of the IP protocol, including the fragmentation of IP datagrams. Furthermore, I familiarized myself with ICMP's functionality by conducting traceroutes with incrementing TTL values. |
| Instructor’s comments and results |
| [Comment]:          Score: Signature of instructor:  Approval date: |

8) Wireshark Ethernet and ARP

|  |
| --- |
| Experiment overview |
| [Experimental project name]:  Wireshark Ethernet and ARP |
| 【Purpose】:  1. Capture and analyze Ethernet frames  2. Observe the ARP protocol in action |
| [Experimental environment] (Software used):  1. Wireshark  2. Window 10 Operating System  [Reference materials]:  1. Wireshark user guide  2. Lab instruments |
| Experiment content |
| [Experimental plan design]:  1. Outline Design:  - Capture Ethernet frames and analyze their contents, examining the packet details.  - Observe the ARP protocol in action, capturing and analyzing its packet details.  2. Detailed Design:  - Open the command line and type the IP address of the target host after the "ping" command. Press enter and observe the responses from the remote computer.  - Capturing and analyzing Ethernet frames.  - Ensure your browser's cache is empty, then launch the Wireshark packet sniffer.  - Enter the URL http://gaia.cs.umass.edu/wireshark-labs/HTTP-ethereal-lab-file3.html in your browser. The lengthy US Bill of Rights should display.  - Stop capturing packets in Wireshark. Identify the packet numbers corresponding to the HTTP - GET message sent to gaia.cs.umass.edu and the start of the HTTP response message received.  - Filter Wireshark to display information only about protocols below IP by going to Analyze -> - Enabled Protocols, unchecking the IPv4 box, and clicking OK.  - Using the content within the Ethernet frame containing the first byte of the HTTP response message, answer questions 5-8.  【Conclusion】(Result):  Capturing and Analyzing Ethernet Frames  1. What is the 48-bit Ethernet address of your computer?  Answer: Source: AmbitMicrosy\_a9:3d:68 (00:d0:59:a9:3d:68)    2. What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of gaia.cs.umass.edu? (Hint: the answer is *no*). What device has this as its Ethernet address? [Note: this is an important question, and one that students sometimes get wrong. Re-read pages 468-469 in the textbook and make sure you understand the answer here.]  Answer: Destination: Broadcast (ff:ff:ff:ff:ff:ff). It is not the address of the gaia.cs.umass.edu. The router of the network has that address.    3. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?  Answer: Type: IPv4 (0x0800)    4. How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame?  Answer: “G” in “Get” shows 47 bytes.    5. What is the value of the Ethernet source address? Is this the address of your computer, or of gaia.cs.umass.edu (Hint: the answer is no)? What device has this as its Ethernet address?  Answer: Source: AmbitMicrosy\_a9:3d:68 (00:d0:59:a9:3d:68)    6. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?  Answer: Destination: LinksysGroup\_da:af:73 (00:06:25:da:af:73)    7.Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?  Answer: Type: IPv4 (0x0800)    8. How many bytes from the very start of the Ethernet frame does the ASCII “O” in “OK” (i.e., the HTTP response code) appear in the Ethernet frame?  Answer: “G” in “Get” shows 4f bytes.    The Adress Resolution Protocol      9. Write down the contents of your computer’s ARP cache. What is the meaning of each column value?  Answer: The first column is IP address of every computer that connected to the network, the second column is their MAC addresses, and the third column is showing whether they are dynamic or static.    10. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message?  Answer: Destination: Broadcast (ff:ff:ff:ff:ff:ff), Source: AmbitMicrosy\_a9:3d:68 (00:d0:59:a9:3d:68)    11. Give the hexadecimal value for the two-byte Ethernet Frame type field. What upper layer protocol does this correspond to?  Answer: Type: IPv4 (0x0800)    12. Now find the ARP reply that was sent in response to the ARP request.  a) How many bytes from the very beginning of the Ethernet frame does the ARP *opcode* field begin?  Answer: The Opcode is 20 bytes from the beginning, so it means it is the 21th byte of the frame.    b) What is the value of the *opcode* field within the ARP-payload part of the Ethernet frame in which an ARP response is made?  Answer: 0x0001    c)Where in the ARP message does the “answer” to the earlier ARP request appear – the IP address of the machine having the Ethernet address whose corresponding IP address is being queried?  Answer: It contains the sender MAC address and IP address. |
| 【summary】:  In this experiment, I captured and analyzed Ethernet frame packets and observed the ARP protocol in action. Additionally, I examined the detailed format and fields of these frames using Wireshark.  Ethernet, as the most crucial protocol in the link layer, allows hosts within a subnet to send packets to each other using MAC addresses. ARP, on the other hand, translates IP addresses into MAC addresses. |
| Instructor’s comments and results |
| [Comment]:        Score: Signature of instructor:  Approval date: |

Appendix:

Experiment Report Description

1) Experimental Project Name: Use concise language to accurately reflect the experiment's content, aligning with the experimental instructions.

2) Experiment Purpose: Clearly state the objective, focusing on key points and adhering to the experimental task requirements.

3) Experimental Environment: Describe the software and hardware environment (configuration) used in the experiment.

4) Experimental Plan Design: This section is crucial and should include:

- Outline design

- Detailed design

- Core algorithm description and analysis

- System development tools

- Submit an electronic version of the program or design.

- For design-type and comprehensive experiments, include flow charts, design ideas, and methods, along with text descriptions. For innovative experiments, note their innovation points and characteristics.

5) Conclusion (Result): Summarize the conclusions based on observed phenomena and measured data during the experiment. Include screenshots of some test results if applicable.

6) Summary: Reflect on the experiment, discuss encountered problems and their solutions, and provide additional thoughts and suggestions.

7) Instructor's Comments and Results: The instructor will provide a concise evaluation and assessment of the experimental report based on the student's actual report content