全媒体人才培养丛书 · 数据科学系列

计算机网络协议分析与程序设计实践

COMPUTER NETWORKS
PROTOCOL ANALYSIS LAB GUIDE AND
PROGRAMMING PRACTICE

林卫国 著

CUCP 中国传媒大学出版社

全媒体人才培养丛书 · 数据科学系列

计算机网络协议分析与程序设计实践

COMPUTER NETWORKS PROTOCOL ANALYSIS LAB GUIDE AND PROGRAMMING PRACTICE

林卫国 著

中国传媒大学出版社
· 北京 ·

图书在版编目(CIP)数据

计算机网络协议分析与程序设计实践 / 林卫国著 .
-- 北京 : 中国传媒大学出版社 , 2021.12
(全媒体人才培养丛书 . 数据科学系列)
ISBN 978-7-5657-3124-2

Ⅰ . ①计… Ⅱ . ①林… Ⅲ . ①计算机网络—通信协议
②计算机网络—程序设计 Ⅳ . ① TN915.04 ② TP393.09

中国版本图书馆 CIP 数据核字 (2021) 第 274927 号

计算机网络协议分析与程序设计实践

JISUANJI WANGLUO XIEYI FENXI YU CHENGXU SHEJI SHIJIAN

著　　者	林卫国
策划编辑	阳金洲
责任编辑	黄松毅
封面设计	风得信设计 · 阿东
责任印制	李志鹏

出版发行	中国传媒大学出版社		
社　　址	北京市朝阳区定福庄东街 1 号	邮　　编	100024
电　　话	86-10-65450528　65450532	传　　真	65779405
网　　址	http://cucp.cuc.edu.cn		
经　　销	全国新华书店		

印　　刷	唐山玺诚印务有限公司
开　　本	787mm × 1092mm　1 / 16
印　　张	6.75
字　　数	123 千字
版　　次	2021 年 12 月第 1 版
印　　次	2021 年 12 月第 1 次印刷

书　　号	ISBN 978-7-5657-3124-2/TP · 3124	定　　价	36.00元

本社法律顾问:北京李伟斌律师事务所　郭建平

Preface

"Tell me and I forget. Show me and I remember. Involve me and I understand." — Chinese proverb

"不闻不若闻之，闻之不若见之，见之不若知之，知之不若行之。"

——中国谚语

The author strongly believes in "learning by doing", so all the lab and project guides in this book are hands-on approaches when conducting courses of Computer Networks and Advance Network Programming to students majoring in computer science in the school of Computer and Cyber Sciences of Communication University of China. We use Professor Andrew Tanenbaum's Computer Networks (5th edition) as the text, and Professor James Kurose, Keith Ross' Computer Networking: A Top-Down Approach (7th edition) as a reference for the course.

All the labs in part 1 are done by Wireshark, in those Wireshark labs, students will be running various network applications in different scenarios using their own computer or using a computer in the school's lab. Students will observe the network protocols "in action", interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, students observe, and learn, by doing.

Another approach for students to understand network protocols deeply is to manipulate the protocols by writing program codes, so all the projects in part

2 are done by programming applications with C/C++ using WinPcap and Windows Socket library, – assembling the sequence of data unit exchanged between two protocol entities, delving down into the details of protocol operation, designing and implementing application level protocols, Thus, students code, and learn, by doing.

This book intends for undergraduate or graduate students to practice in the lab as a supplement to the text, it can be used as training text for computer engineers, programmers, software developers, network and system administrators, and others who want to learn the principles of computer networks and network programming as well.

Complete with step-by-step deigned labs and projects from this book, readers will be able to overcome challenges in understanding the complex network protocols, design protocols and implement network applications, and more importantly, apply the learned problem-solving principles to tack real-world problems.

I have to thank many open source projects and open course materials. Many of Wireshark guides are adapted from Professor James Kurose, Keith Ross' teaching materials, and many project guides in part 2 are adapted from all kinds open resources.

Author

2020.12.18

目 录

CHAPTER

Part 1
Lab Guides to Computer Network Protocol Analysis

Lab 1.1 Introduction to Wireshark

One's understanding of network protocols can often be greatly deepened by "seeing protocols in action" and by "playing around with protocols" – observing the sequence of messages exchanged between two protocol entities, delving down into the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences. This can be done in simulated scenarios or in a "real" network environment such as the Internet. In the Wireshark labs you'll be doing in this course, you'll be running various network applications in different scenarios using your own computer or using a computer in the school's lab. You'll observe the network protocols in your computer "in action," interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, you and the computer you are using will be an integral part of these "live" labs. You'll observe, and you'll learn, by doing.

In this first network protocol analysis lab, you'll get acquainted with Wireshark, and make some simple packet captures and observations.

The basic tool for observing the messages exchanged between executing protocol entities is called a packet sniffer. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer; it will also typically store and/or display the

contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent/received from/by application and protocols executing on your machine.

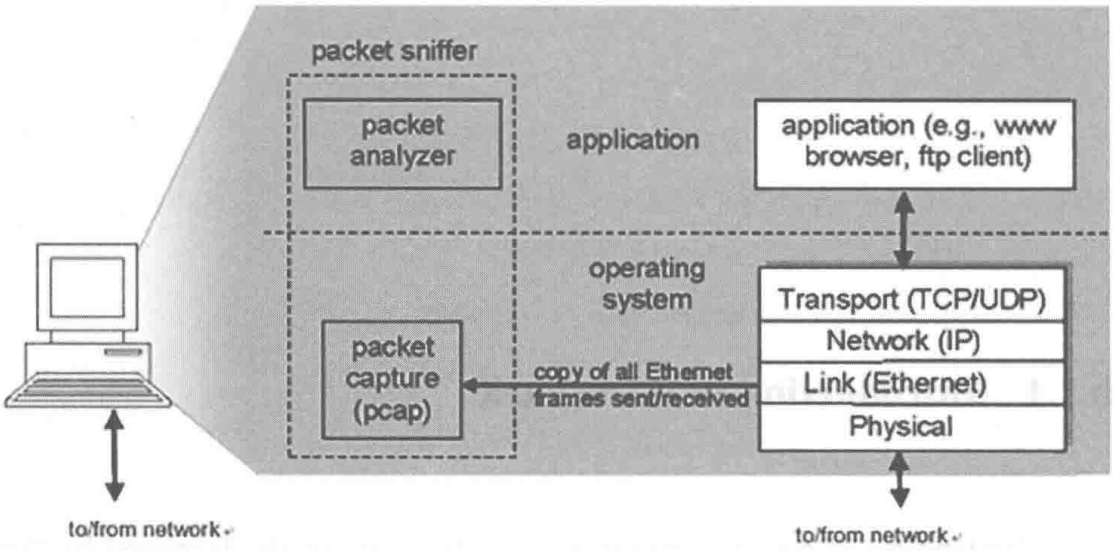

Figure 1.1 Packet sniffer structure

Figure 1.1 shows the structure of a packet sniffer. At the right of Figure 1.1 are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in Figure 1.1 is an addition to the usual software in your computer, and consists of two parts. The **packet capture library** receives a copy of every link-layer frame that is sent from or received by your computer. Messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 1.1, the assumed physical media is an Ethernet, and so all upper-layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/ received from/by all protocols and applications executing in your computer.

The second component of a packet sniffer is the **packet analyzer** (or protocol analyzer), which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must "understand" the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages

exchanged by the HTTP protocol in Figure 1.1 The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string "GET," "POST," or "HEAD".

We will be using the Wireshark packet sniffer (the Wireshark defines itself as the world's foremost and widely-used network protocol analyzer) [http://www.wireshark.org/] for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers. It's an ideal packet analyzer for our labs – it is stable, has a large user base and well-documented support that includes a user-guide, man pages, and a detailed FAQ, rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, serial (PPP and SLIP), 802.11 wireless LANs, and many other link-layer technologies (if the OS on which it's running allows Wireshark to do so).

Getting Wireshark

In order to run Wireshark, you will need to have access to a computer that supports both Wireshark and the *libpcap* or *WinPCap* packet capture library. The *libpcap* software will be installed for you, if it is not installed within your operating system, when you install Wireshark. See http://www.wireshark.org/download.html for a list of supported operating systems and download sites.

Download and install the Wireshark software:

- Go to http://www.wireshark.org/download.html and download and install the Wireshark binary for your computer.

The Wireshark FAQ has a number of helpful hints and interesting tidbits of information,

particularly if you have trouble installing or running Wireshark.

Running Wireshark

When you run the Wireshark program, you'll get a startup screen, as shown below:

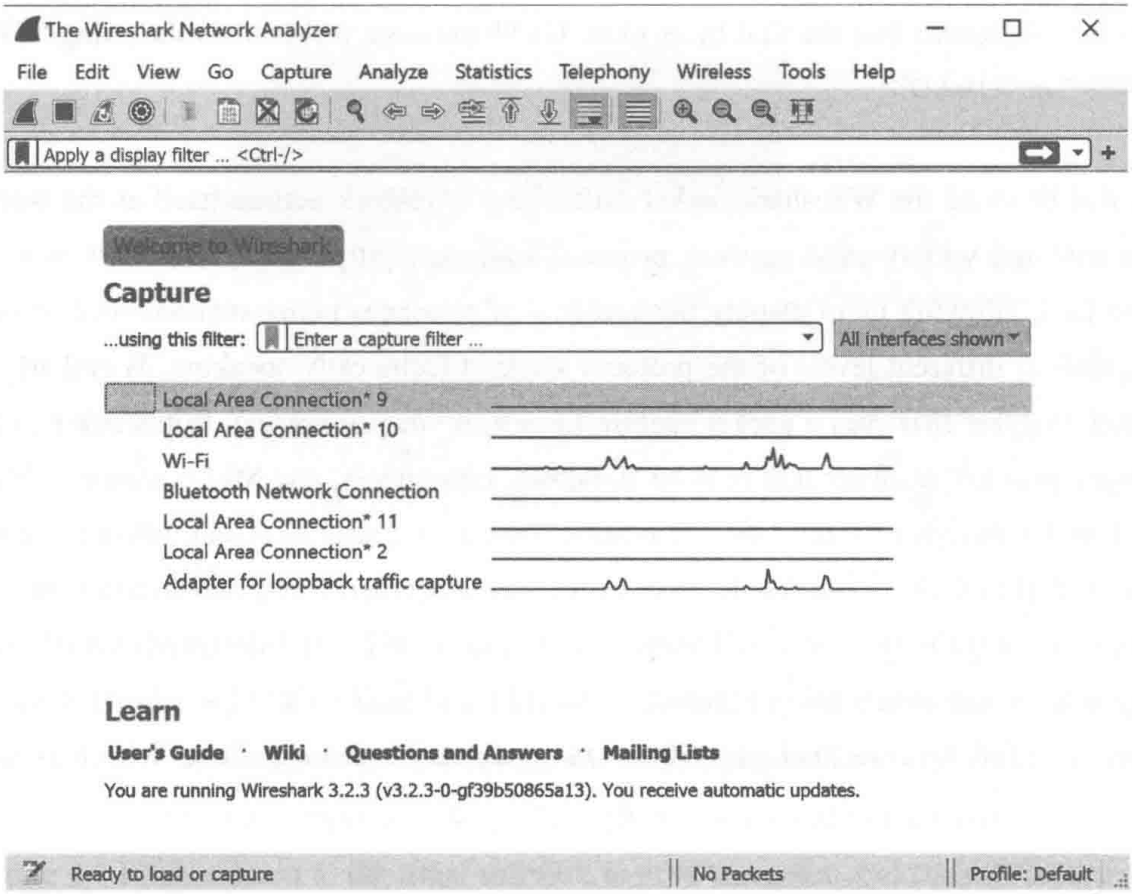

Figure 1.2 Initial Wireshark Screen

Take a look at the center of the screen – you'll see an "Interface list". This is the list of network interfaces on your computer. The waving line indicates the corresponding network interface has active traffic flow.

If you double-click on one of these interfaces to start packet capture (i.e., for Wireshark to begin capturing all packets being sent to/from that interface), a screen like the one below (Figure 1.3) will be displayed, showing information about the packets being captured. Once you start packet capture, you can stop it by using the Capture pull down menu and selecting Stop or just click the Stop button on the toolbar.

The Wireshark interface has five major components:

- The command **menus** are standard pulldown menus located at the top of the window. Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.
- The **packet-listing window** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is *not* a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.
- The **packet-header details window** provides details about the packet selected (highlighted) in the packet-listing window. (To select a packet in the packet-listing window, place the cursor over the packet's one-line summary in the packet-listing window and click with the left mouse button.). These details include information about the Ethernet frame (assuming the packet was sent/received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the plus minus boxes to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.
- The **packet-contents window** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.
- Towards the top of the Wireshark graphical user interface, is the **packet display filter field**, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond

to HTTP messages.

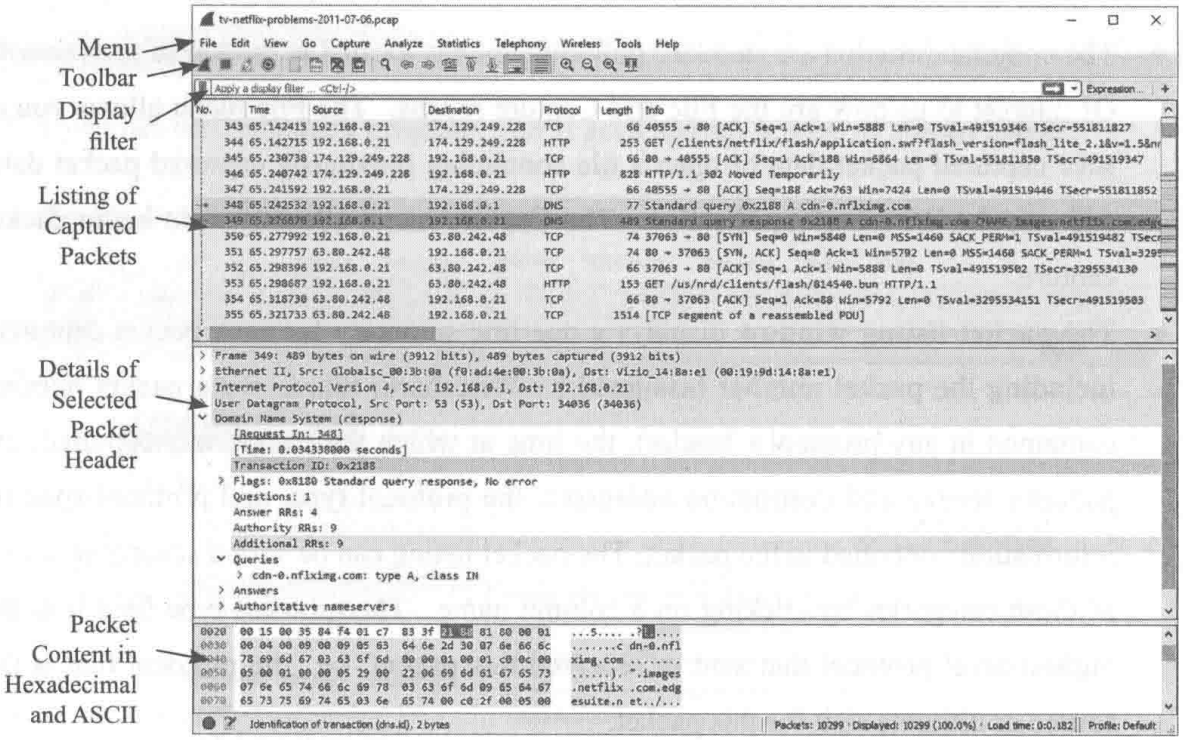

Figure 1.3 Wireshark Graphical User Interface, during packet capture and analysis

Taking Wireshark for a Test Run

The best way to learn about any new piece of software is to try it out! We'll assume that your computer is connected to the Internet via a wired Ethernet interface. Indeed, I recommend that you do this first lab on a computer that has a wired Ethernet connection, rather than just a wireless connection. Do the following:

1. Start up your favorite web browser, which will display your selected homepage.
2. Start up the Wireshark software. You will initially see a window similar to that shown in Figure 1.2. Wireshark has not yet begun capturing packets.
3. To begin packet capture, double-click the network interface which you want to begin packet capture or choose a network interface then click the *start capture packets* button in the toolbar. Packet capture will now begin -Wireshark is now capturing all packets being sent/received from/by your computer!
4. Once you begin packet capture, a window similar to that shown in Figure 1.3 will appear. This window shows the packets being captured. By clicking the *stop capture*

packets button in the toolbar or selecting *Capture* pulldown menu and selecting *Stop*, you can stop packet capture. But don't stop packet capture yet. Let's capture some interesting packets first. To do so, we'll need to generate some network traffic. Let's do so using a web browser, which will use the HTTP protocol to download content from a website.

5. While Wireshark is running, enter the URL: http://cloud.cuc.edu.cn/computernetworks/Labs/helloworld.html , and have that page displayed in your browser. In order to display this page, your browser will contact the HTTP server at cloud.cuc.edu.cn and exchange HTTP messages with the server in order to download this page, as discussed in the text. The Ethernet frames containing these HTTP messages (as well as all other frames passing through your Ethernet adapter) will be captured by Wireshark.
6. After your browser has displayed the helloworld.html page (it is a simple one line of helloworld), stop Wireshark packet capture by clicking *stop capture packets* button in the Wireshark toolbar. The main Wireshark window should now look similar to Figure 1.3. You now have live packet data that contains all protocol messages exchanged between your computer and other network entities! The HTTP message exchanges with the cloud.cuc.edu.cn web server should appear somewhere in the listing of packets captured. But there will be many other types of packets displayed as well (see, e.g., the many different protocol types shown in the *Protocol* column in Figure 1.3). Even though the only action you took was to download a web page, there were evidently many other protocols running on your computer that are unseen by the user. We'll learn much more about these protocols as we progress through the text! For now, you should just be aware that there is often much more going on than "meet's the eye"!
7. Type in "http" (without the quotes, and in lower case – all protocol names are in lower case in Wireshark) into the display filter specification window at the top of the main Wireshark window. Then select *Apply* (to the right of where you entered "http"). This will cause only HTTP message to be displayed in the packet-listing window.
8. Find the HTTP GET message that was sent from your computer to the cloud.cuc.edu.cn HTTP server. (Look for an HTTP GET message in the "listing of captured packets" portion of the Wireshark window (see Figure 1.3) that shows "GET" followed by the cloud.cuc.edu.cn URL that you entered. When you select the HTTP

GET message, the Ethernet frame, IP datagram, TCP segment, and HTTP message header information will be displayed in the packet-header window (the HTTP GET message that is sent to the web server is contained within a TCP segment, which is encapsulated in an IP datagram, which is encapsulated in an Ethernet frame.). By clicking on '+' and '- 'right-pointing and down-pointing arrowheads to the left side of the packet details window, *minimize* the amount of Frame, Ethernet, Internet Protocol, and Transmission Control Protocol information displayed. *Maximize* the amount information displayed about the HTTP protocol.

Your Wireshark display should now look roughly as shown in Figure 1.4. (Note, in particular, the minimized amount of protocol information for all protocols except HTTP, and the maximized amount of protocol information for HTTP in the packet-header window).

9. Exit Wireshark.

Congratulations! You've now completed the first lab.

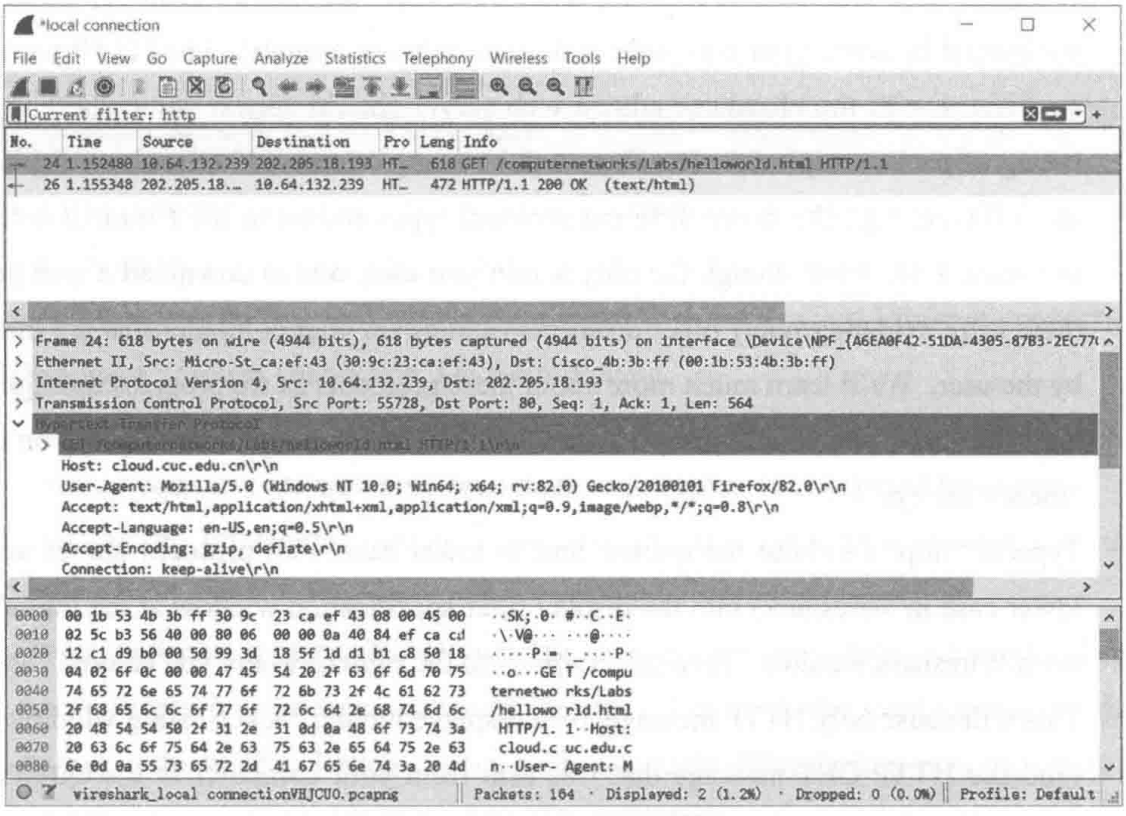

Figure 1.4 Wireshark window after step 9

What to hand in

The goal of this first lab was primarily to introduce you to Wireshark. The following questions will demonstrate that you've been able to get Wireshark up and running, and have explored some of its capabilities. Answer the following questions in your lab report, based on your Wireshark experimentation:

1. List 3 different protocols that appear in the protocol column in the unfiltered packet-listing window in step 7 above.
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.)
3. What is the Internet address of the cloud.cuc.edu.cn? What is the Internet address of your computer?
4. What are the source Ethernet address and the destination Ethernet address of the frame which contains the HTTP GET message? Guess who has the source Ethernet address and who has the destination Ethernet address correspondingly.
5. 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. [or you can just make a screen capture and insert it to your lab report.]

Lab 1.2 Ethernet and ARP

In this lab, we'll investigate the Ethernet protocol and the ARP protocol. Before beginning this lab, you'll probably want to review relevant sections in the text of computer networks. RFC 826 (https://tools.ietf.org/html/rfc826) contains the details of the ARP protocol, which is used by an IP device to determine the IP address of a remote interface whose Ethernet address is known.

1. Capturing and analyzing Ethernet frames

Let's begin by capturing a set of Ethernet frames to study. Do the following:

- First, make sure your browser's cache is empty. To do this under Firefox, select *Options,* click on the *clear data* button in the *Cookies and Site Data* section.
- Start up the Wireshark packet sniffer, Enter the following URL into your browser http://cloud.cuc.edu.cn/computernetworks/Labs/HTTP-Wireshark-file3.html ,Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture. First, find the packet numbers (the leftmost column in the upper Wireshark window) of the HTTP GET message that was sent from your computer to cloud.cuc.edu.cn, as well as the beginning of the HTTP response message sent to your computer by cloud.cuc.edu.cn. You should see a screen that looks something like Figure 1.5 (where packet 16 in the screen shot below contains the HTTP GET message).
- Since this lab is about Ethernet and ARP, we're not interested in IP or higher-layer protocols. So let's change Wireshark's "listing of captured packets" window so that it shows information only about protocols below IP. To have Wireshark do this, select Analyze->Enabled Protocols. Then uncheck the IPv4 box and select OK. You should now see the Wireshark window that looks like Figure 1.6.

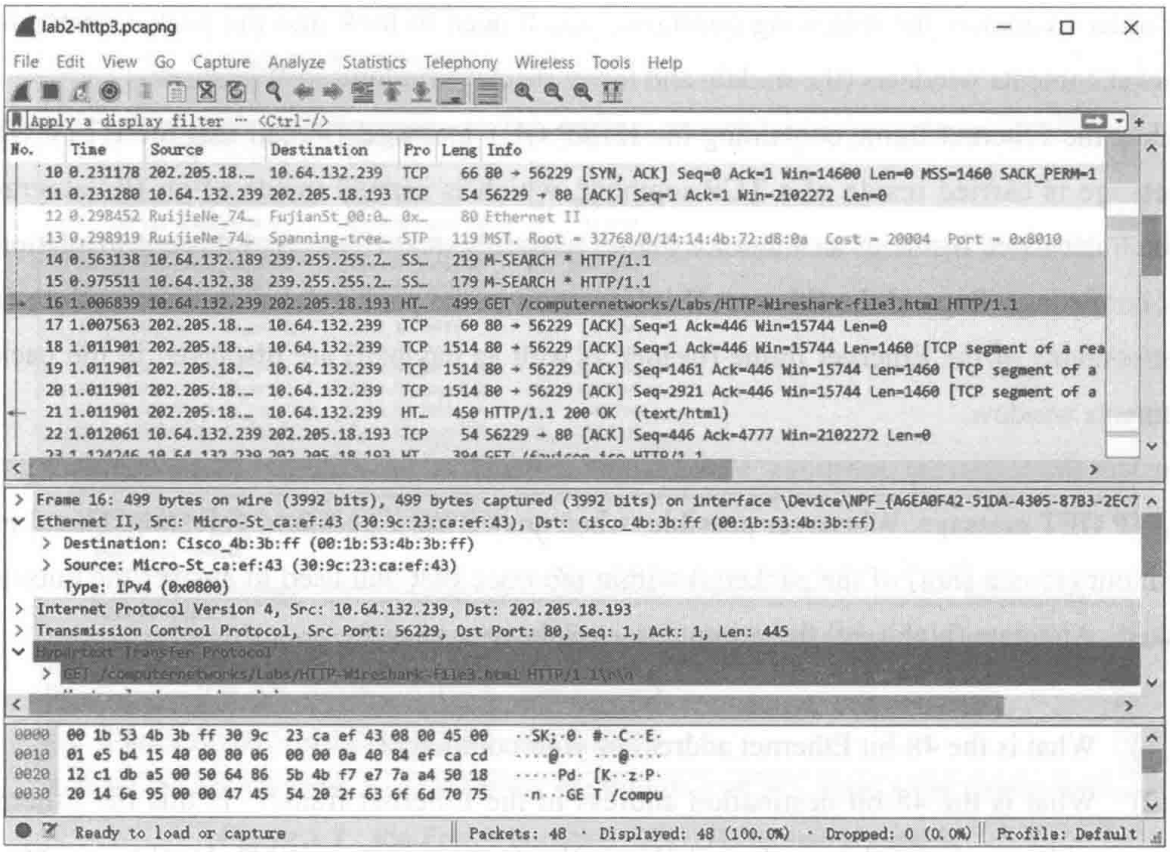

Figure 1.5 the HTTP GET Message

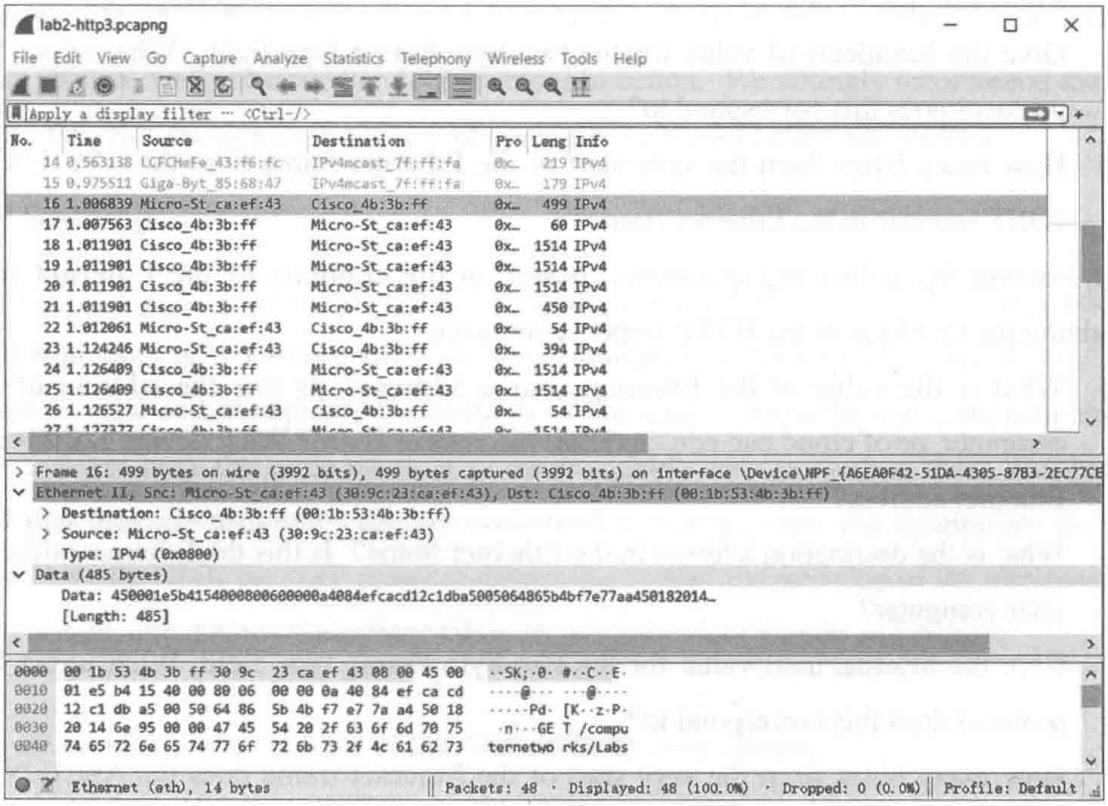

Figure 1.6 IPv4 Protocol disabled