Computer Networks Lab

based on

Mastering Networks - An Internet Lab Manual
by Jörg Liebeherr and Magda Al Zarki

and on

Computer Networking - A Top-Down Approach
by James Kurose and Keith Ross

Adapted for
'Computernetwerken'
by Johan Bergs and Stephen Pauwels

Tim Verhaegen
Pim Van den Bosch
Group 12

March 15, 2023

Lab 1

The Application layer

What you will learn in this lab:

- How HTTP works and how the packets look like
- How you can write your own HTTP requests
- How DNS works

Part 1. The Hypertext Transfer Protocol (HTTP)

Based on Wireshark Lab: HTTP, Supplement to Computer Networking: A Top-Down Approach, 8th ed., J.F. Kurose and K.W. Ross

When people visit webpages, these pages are requested and served using the HTTP protocol. In these first exercises we take a closer look at how the communication works and how the packets look like.

Exercise 1: Running simple HTTP requests

1. Start up your web browser. While doing these exercises, make sure that you turn off all applications that use an Internet connection and any other browsers. By doing this you will eliminate as much unwanted packets as possible.
2. Apple users: make sure that the *Private Relay* optional feature is turned off!
3. *Do not use Safari as web browser for this exercise!* Firefox, Chrome or Edge are good browsers to do this exercise with.
4. Make sure your browser cache is cleared. In Firefox, go to *Settings -> Privacy & Security -> Cookies and Site Data*, and use the *Clear data...* button to remove all cached web content. In Microsoft Edge, go to *Settings -> Privacy, search and services -> Clear browsing data*, and then click on *Choose what to clear*. Make sure Cached files and images are cleared. In Chrome, this is similarly done by navigating to *Settings -> Security and Privacy -> Clear browsing data*.
5. Start up the Wireshark packet sniffer, as described in the Introductory lab (but don't yet begin packet capture). Enter "http" (just the letters, not the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets).
6. Begin the Wireshark packet capture.
7. Enter the following to your browser:
   `http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html`.
   Your browser should display a very simple, one-line HTML file.
8. Stop Wireshark packet capture and save the results of the captures traffic as libpcap file called `L1-1-1.pcapng`.

Your Wireshark output should look similar to the window shown in Figure 1.1.

The example in Figure 1.1 shows in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the gaia.cs.umass.edu web server) and the response message from the server to your browser. The packet-contents window shows details of the selected message (in this case the HTTP OK message, which is highlighted in the packet-listing window). Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP packet information as well. We want to minimize the amount of non-HTTP data displayed (we're interested in HTTP here, and will be investigating these other protocols is later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign or a right-pointing triangle (which means there is hidden, non-displayed information), and the HTTP line has a minus sign or a down-pointing triangle (which means that all information about the HTTP message is displayed).

graphics/wireshark-http.png

Figure 1.1: Wireshark output after our HTTP request.

Use the data captured with Wireshark in L1-1-1.pcapng to answer the questions. In your answers, refer to the packets in the Wireshark capture and explain how you come to the correct answer.

L1-1-1 Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running? /1

Both my web browser and the server are running HTTP/1.1.

L1-1-2 What languages (if any) does your browser indicate is/are the natural language it is set to and prefers? /1

My browser indicates English is the preferred language, using the Accept-Language header.

L1-1-3 What is the IP address of your computer? Of the `gaia.cs.umass.edu` server? /1

The source IP address for the outgoing HTTP request is 10.0.2.15. This technically isn't the IP address of my computer but the IP address of the NIC within the virtual network created by VirtualBox.
The IP address of the server is 128.119.245.12.

L1-1-4 What is the status code returned from the server to your browser? /1

Status code 200 was returned from the server to the browser.

L1-1-5 When was the HTML file that you are retrieving last modified at the server? /1

Sun, 05 Mar 2023 06:59:01 GMT.

L1-1-6 How many bytes of content are being returned to your browser? /1

---

128 bytes of content are being returned.

L1-1-7 What is the type of web server that is running at `gaia.cs.umass.edu`? /1

The server is running an Apache web-server version 2.4.6.

Exercise 2: The HTTP Conditional GET/response interaction

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer
3. Enter the following URL into your browser:
   `http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html`.
   Your browser should display a very simple five-line HTML file.
4. Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)
5. Stop Wireshark packet capture and save the results of the captures traffic as libpcap file called `L1-2-1.pcapng`. Enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Use the data captured with Wireshark in L1-2-1.pcapng to answer the questions.

L1-2-1 Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "If-Modified-Since" field in the HTTP GET? /1

There is no If-Modified-Since header in the first request to the server. This is because nothing has been cached yet.

L1-2-2 Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell? /1

The server explicitly returned the contents of this page. You can tell from both the 200 status code, and the fact that the contents of the page are found in the packet, as html.

L1-2-3 Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "If-Modified-Since" field in the HTTP GET? If so, what information follows the "If-Modified-Since" header? What is its purpose? /1

There is a If-Modified-Since header with the value of the Last-Modified header of the previous response. This indicates the client only wants a fresh copy if the old version has changed.

L1-2-4 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. /1

The server returned a status code of 304 and phrase 'Not Modified'. It did not explicitly return the contents of the file, since the client requested the data conditionally using the If-Modified-Since header.

Exercise 3: Retrieving long documents

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer
3. Make sure that the following options are **unselected** in Wireshark: go to *Edit* -> *Preferences* -> *Protocols* -> *HTTP* and **turn off** all the options that start with *Reassemble*.
4. Enter the following URL into your browser:
   `http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html`.
   Your browser should display the rather lengthy US Bill of Rights.
5. Stop Wireshark packet capture and save the results of the captures traffic as libpcap file called `L1-3-1.pcapng`. Enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request. This multiple-packet response deserves a bit of explanation. Recall that the HTTP response message consists of a status line, followed by header lines, followed by a blank line, followed by the entity body. In the case of our HTTP GET, the entity body in the response is the entire requested HTML file. In our case here, the HTML file is rather long, and at 4500 bytes is too large to fit in one TCP packet. The single HTTP response message is thus broken into several pieces by TCP, with each piece being contained within a separate TCP segment. In recent versions of Wireshark, Wireshark indicates each TCP segment as a separate packet, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the "TCP segment of a reassembled PDU" in the Info column of the Wireshark display. Earlier versions of Wireshark used the "Continuation" phrase to indicated that the entire content of an HTTP message was broken across multiple TCP segments.. We stress here that there is no "Continuation" message in HTTP!

Use the data captured with Wireshark in L1-3-1.pcapng to answer the questions.

L1-3-1 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? /1

The browser sent 1 GET request. It is contained in packet number 7.

L1-3-2 Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request? /1

Packet 11 in the trace contains the status code and phrase.

L1-3-3 What is the status code and phrase in the response? /1

The status code is 200 and the phrase is ’OK’.

L1-3-4 How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights? /1

4 were needed in total, found in packets: 11, 12, 15 and 17.

Exercise 4: HTML documents with embedded objects

1. Start up your web browser, and make sure your browser’s cache is cleared.
2. Start up the Wireshark packet sniffer
3. Enter the following URL into your browser:
   `http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html`.
   Your browser should display a short HTML file with two images. These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. As discussed in the textbook, your browser will have to retrieve these logos from the indicated web sites. Our publisher’s logo is retrieved from the `gaia.cs.umass.edu` web site. The image of the cover for our 5th edition (one of our favourite covers) is stored at the `caite.cs.umass.edu` server. (These are two different web servers inside `cs.umass.edu`).
4. Stop the Wireshark packet capture and save the results of the captures traffic as libpcap file called `L1-4-1.pcapng`. Enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed.

Use the data captured with Wireshark in L1-4-1.pcapng to answer the questions.

L1-4-1 How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent? /1

In L1-4-1.pcapng, we can see there have been 4 GET requests sent:

1. Packet 17: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html
2. Packet 36: http://gaia.cs.umass.edu/pearson.png
3. Packet 46: http://kurose.cslash.net/8E_cover_small.jpg
4. Packet 69: http://gaia.cs.umass.edu/favicon.ico

L1-4-2 Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain. /1

We can clearly tell they have been downloaded in parallel because both requests were sent before a single response came back for either of them.

Exercise 5: HTTP Authentication

Finally, let's try visiting a web site that is password-protected and examine the sequence of HTTP messages exchanged for such a site. The URL `http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html` is password protected.
The username is *wireshark-students*, and the password is *network*. So let's access this "secure" password-protected site.

1. Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser
2. Start up the Wireshark packet sniffer
3. Enter the following URL into your browser:
   `http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html`.
   Type the requested username and password into the pop-up box.
4. Stop Wireshark packet capture and save the results of the captures traffic as libpcap file called `L1-5-1.pcapng`. Enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Use the data captured with Wireshark in L1-5-1.pcapng to answer the questions.

L1-5-1 What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser? /1

The server returned 401 as status code and 'Unauthorized' as phrase.

L1-5-2 When your browser sends the HTTP GET message for the second time, what new field is included in the HTTP GET message? /1

The request now has a ’Authorization’ header with the value of ’Basic’ and the username and password, separated using a colon.

The username (wireshark-students) and password (network) that you entered are encoded in the string of characters (`d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=`) following the “Authorization: Basic” header in the client’s HTTP GET message. While it may appear that your username and password are encrypted, they are simply encoded in a format known as Base64 format. The username and password are not encrypted! To see this, simply click on the Authorization field in Wireshark to expand it. Voila! You have translated from Base64 encoding to ASCII encoding, and thus should see your username and password! Since anyone can download a tool like Wireshark and sniff packets (not just their own) passing by their network adaptor, and anyone can translate from Base64 to ASCII (Wireshark already does this for you!), it should be clear to you that simple passwords on HTTP sites are not secure unless additional measures are taken (e.g. the use of HTTPS).

Part 2. Create your own HTTP requests

Until now, we used a browser to retrieve pages via HTTP. By using these applications we do not have to write the request to web servers ourselves. In the following exercises we are going to write our own request and send them to a web server. We will do so by using the Python3 library `socket`. More information on this library can be found on https://docs.python.org/3/howto/sockets.html#socket-howto

You can create a socket connection using the following functions:

```
    client = socket(AF_INET, SOCK_STREAM)
    client.connect((host, port))
```

To send and receive bytes you can use the following functions, for receiving you always have to indicate the amount of bytes to receive.

```
    client.send(request_in_bytes)
    response_in_bytes = client.recv(4096)
```

Note: when adding a newline to the request, use `\r\n` *in Python to ensure that the server can interpret your request correctly.*
Note: do not forget to convert your request (string) to bytes and back (for the response).
Note: if you get the "400 Bad Request" response in any of the following exercises, this means your request was malformed, so you should correct it.

Exercise 6: Creating HTTP requests

Use the `socket` library to write HTTP requests and print the response to your screen.

L1-6-1 Write a python script that sends a request for the home page of `www.google.be`. Save your script to `L1-6-1.py`. Verify that your script works and include it here. /1

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client.connect(('www.google.be', 80))
send_result = client.send(b'''GET / HTTP/1.0\r\n\r\n''')

complete_response = ''

resp = client.recv(512)

while resp != b'':
    complete_response += str(resp)
    resp = client.recv(512)

print(complete_response)
```

traces/L1-6-1.py

L1-6-2 Write a request to retrieve the following page:
`http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html`.

Save it to `L1-6-2.py`. Include your script here, as well as the response you received. You do not have to download the images via separate requests. /1

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client.connect(('gaia.cs.umass.edu', 80))
send_result = client.send(b'''GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.0\r\n
    \r\n''')

complete_response = ''

resp = client.recv(512)

while resp != b'':
    complete_response += str(resp)
    resp = client.recv(512)

print(complete_response)
```

traces/L1-6-2.py

The response (with escape characters still present) can be found here:L1-6-2.txt

L1-6-3 Write a request for the home page of `www.uantwerpen.be`, save it to `L1-6-3.py`. Include your python script here, as wel as the response. /1

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client.connect(('www.uantwerpen.be', 80))
send_result = client.send(b'''GET / HTTP/1.0\r\nHost:www.uantwerpen.be\r\n\r\n''')

complete_response = ''

resp = client.recv(512)

while resp != b'':
    complete_response += str(resp)
    resp = client.recv(512)

print(complete_response)
```

traces/L1-6-3.py

The response (with escape characters still present) can be found here:L1-6-3.txt

L1-6-4 Explain the response you got from `www.uantwerpen.be`, why did you get this response message? /1

We get a response indicating that the resource has moved. In this case, the server is configured to redirect the client to the path: /nl. This might be based on IP addresses; to serve the Dutch version for IP addresses located in Dutch-speaking regions.

L1-6-5 Now try to request the home page of `www.uantwerpen.be` by sending the request to `www.google.be`. Save your script to `L1-6-4.py` and include it here, as well as the response. /1

```python
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client.connect(('www.google.be', 80))
send_result = client.send(b'''GET www.uantwerpen.be/ HTTP/1.0\r\n\r\n''')

complete_response = ''

resp = client.recv(1024)

while resp != b'':
    complete_response += str(resp)
    resp = client.recv(1024)

print(complete_response)
```

traces/L1-6-5.py

The response can be found in: L1-6-5.txt

L1-6-6 Explain the previous response you got, why did you get this response message? /1

We received a 404 error because www.google.be does not have the homepage of www.uantwerpen.be.

Part 3. Domain Name System (DNS)

Clearing your DNS cache

To ensure that you get the correct DNS queries, you can explicitly clear the records in your DNS cache. There's no harm in doing so - it just means that your computer will need to invoke the distributed DNS service next time it needs to use the DNS name resolution service, since it will find no records in the cache.

- On a Mac computer, you can enter the following command into a terminal window to clear your DNS resolver cache:

```
% sudo killall -HUP mDNSResponder
```

- On a Windows computer you can enter the following command at the command prompt:

```
% ipconfig /flushdns
```

- On a Linux computer, enter:

```
% sudo systemd-resolve --flush-caches
```

Exercise 7: DNS queries

Let's first capture the DNS messages that are generated by ordinary Web-surfing activity.

1. Clear the DNS cache in your host, as described above.
2. Open your Web browser and clear your browser cache.
3. Open Wireshark and enter `ip.addr == <your_IP_address>` into the display filter, where `<your_IP_address>` is the IPv4 address of your computer . With this filter, Wireshark will only display packets that either originate from, or are destined to, your host.
4. Start packet capture in Wireshark.
5. With your browser, visit the Web page: `http://gaia.cs.umass.edu/kurose_ross/`
6. Stop the packet capture and save the results of the captures traffic as libpcap file called `L1-7-1.pcapng`.

Use the data captured with Wireshark in L1-7-1.pcapng to answer the questions.

L1-7-1 Locate the first DNS query message resolving the name `gaia.cs.umass.edu`. What is the packet number in the trace for the DNS query message? Is this query message sent over UDP or TCP? /1

The packet number for the first DNS query is 1. The message is sent over UDP.

L1-7-2 Now locate the corresponding DNS response to the initial DNS query. What is the packet number in the trace for the DNS response message? Is this response message received via UDP or TCP? /1

Here the packet number for the response is 4. The message is again received via UDP.

L1-7-3 What is the destination port for the DNS query message? What is the source port of the DNS response message? /1

The destination port for the DNS query message is 53. The source port for the DNS response message is also 53.

L1-7-4 To what IP address is the DNS query message sent? /1

The destination IP address is 192.168.1.1

L1-7-5 Examine the DNS query message. How many "questions" does this DNS message contain? How many "answers" does it contain? /1

The DNS query contains 1 question and it contains 0 answers.

L1-7-6 Examine the DNS response message to the initial query message. How many "questions" does this DNS message contain? How many "answers" does it contain? /1

The DNS response message contains 1 question and it contains 1 answer.

L1-7-7 The web page for the base file `http://gaia.cs.umass.edu/kurose_ross/` references the image object `http://gaia.cs.umass.edu/kurose_ross/header_graphic_book_8E3.jpg`, which, like the base webpage, is on `gaia.cs.umass.edu`.

What is the packet number in the trace for the initial HTTP GET request for the base file `http://gaia.cs.umass.edu/kurose_ross/`?

What is the packet number in the trace of the DNS query made to resolve `gaia.cs.umass.edu` so that this initial HTTP request can be sent to the `gaia.cs.umass.edu` IP address?

What is the packet number in the trace of the received DNS response?

What is the packet number in the trace for the HTTP GET request for the image object `http://gaia.cs.umass.edu/kurose_ross/header_graphic_book_8E3.jpg`?

What is the packet number in the DNS query made to resolve `gaia.cs.umass.edu` so that this second HTTP request can be sent to the `gaia.cs.umass.edu` IP address?

Discuss how DNS caching affects the answer to this last question. /1

The packet number for the initial HTTP GET request is 11.

The packet number for the initial DNS query to resolve gaia.cs.umass.edu is 1.

The packet number for the recieved DNS response is 4.

The packet number for the HTTP GET request for the image object is 211.

The packet number for the DNS query made to resolve gaia.cs.umass.edu is again 1.

If the DNS query for resolving gaia.cs.umass.edu has been previously made and the response has been cached, then a second DNS query will not be needed again to send a second HTTP request to the same IP address. Instead, the IP address will already be known and can be used directly to send the second HTTP request.

Exercise 8: `nslookup` for type A records

Now let's play with `nslookup`.

1. Start a packet capture.
2. Do an `nslookup` on www.cs.umass.edu
3. Stop the packet capture and save the results of the captured traffic as libpcap file called `L1-8-1.pcap`ng.

Use the data captured with Wireshark in L1-8-1.pcapng to answer the questions.

L1-8-1 What is the destination port for the DNS query message? What is the source port of the DNS response message? /1

Destination port for the DNS query message is 53.
Source port for the DNS response message is also 53

L1-8-2 To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server? /1

The query is sent to 192.168.1.1 . This is not the default local DNS server but rather the IP address of my modem.

L1-8-3 Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"? /1

The DNS query message is type A. A type A record is a type of DNS record that maps a domain name to an IPv4 address.

The query contains no answers.

L1-8-4 Examine the DNS response message to the query message. How many "questions" does this DNS response message contain? How many "answers"? /1

This message contains 1 question and 1 answer.

Exercise 9: `nslookup` for type NS records

Last, let's use `nslookup` to issue a command that will return a type NS DNS record.

1. Start a packet capture.
2. Enter the following command:

```
% nslookup -type=NS umass.edu
```

3. Stop the packet capture and save the results of the captured traffic as libpcap file called `L1-9-1.pcapng`.

Use the data captured with Wireshark in L1-9-1.pcapng to answer the questions.

L1-9-1 To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server? /1

Once again it's sent to 192.168.1.1, which is the IP address of my modem.

L1-9-2 Examine the DNS query message. How many questions does the query have? Does the query message contain any "answers"? /1

The query message has 1 question. The query message contains no answers.

L1-9-3 Examine the DNS response message. How many answers does the response have? What information is contained in the answers? How many additional /1

The DNS response answer contains 3 answers. Each answer corresponds to a nameserver which provides DNS services for the host. So there are 3 nameservers which provide DNS services for umass.edu

---

Acronyms

DNS	Domain Name System
HTTP	Hypertext Transfer Protocol