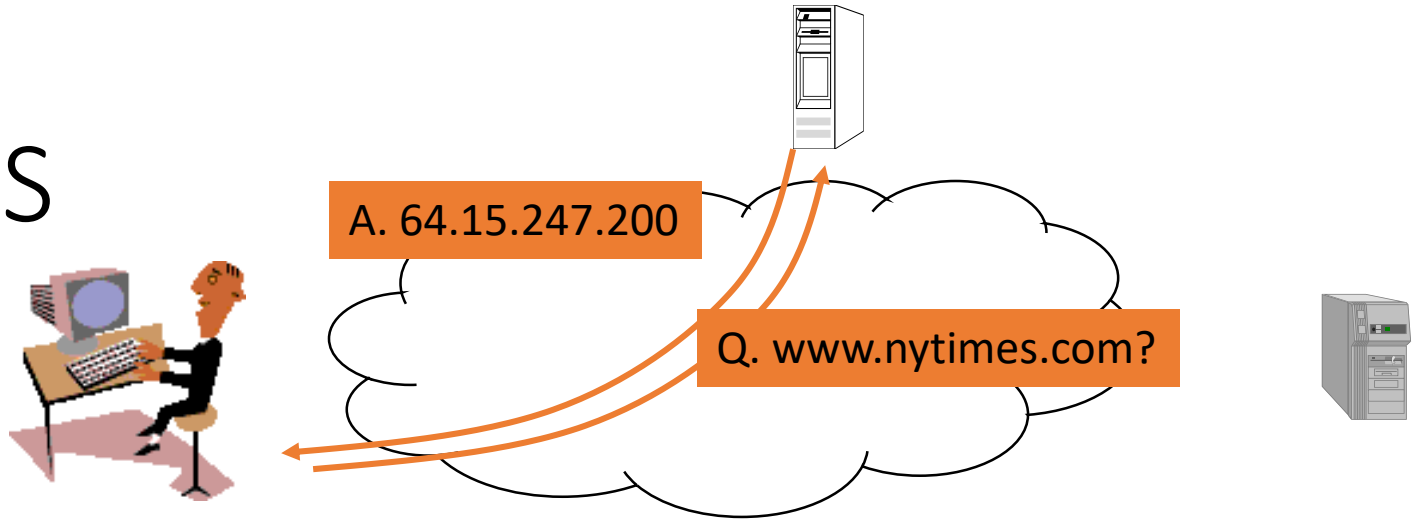
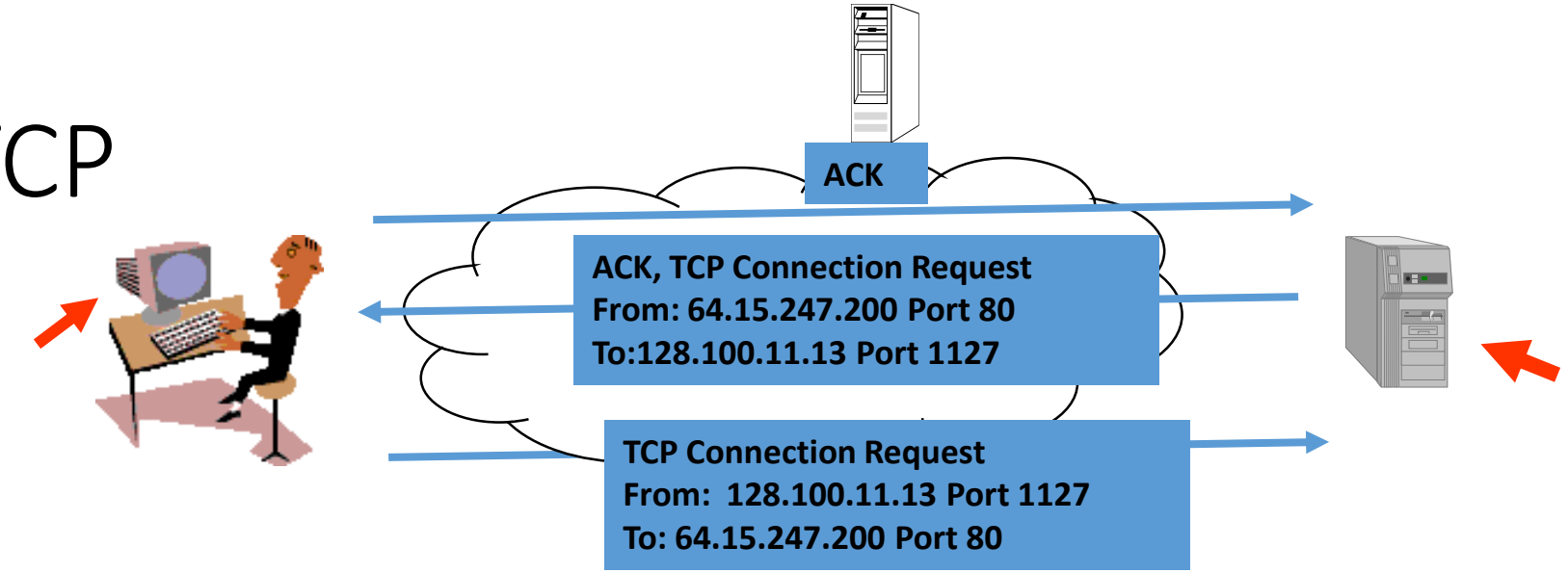
實驗二 wireshark

1. DNS



- User clicks on http://www.nytimes.com/
- URL contains Internet name of machine (www.nytimes.com), but not Internet address
- Internet needs Internet address to send information to a machine
- Browser software uses Domain Name System (DNS) protocol to send query for Internet address
- DNS system responds with Internet address

2. TCP



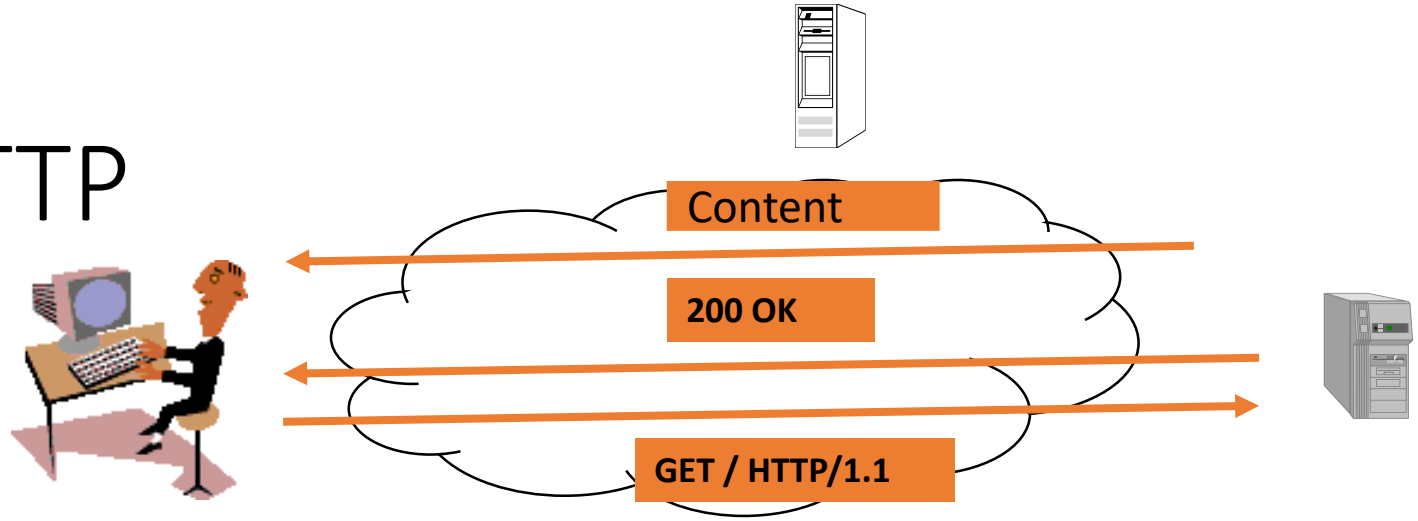
- Browser software uses HyperText Transfer Protocol (HTTP) to send request for document
- HTTP server waits for requests by listening to a well-known port number (80 for HTTP)
- HTTP client sends request messages through an “ephemeral port number,” e.g. 1127
- HTTP needs a Transmission Control Protocol (TCP) connection between the HTTP client and the HTTP server to transfer messages reliably

Example: TCP

- TCP is a transport layer protocol
- Provides *reliable byte stream service* between two processes in two computers across the Internet
- Sequence numbers keep track of the bytes that have been transmitted and received
- Error detection and retransmission used to recover from transmission errors and losses
- TCP is *connection-oriented*: the sender and receiver must first establish an association and set initial sequence numbers before data is transferred
- Connection ID is specified uniquely by

(send port #, send IP address, receive port #, receiver IP address)

3. HTTP



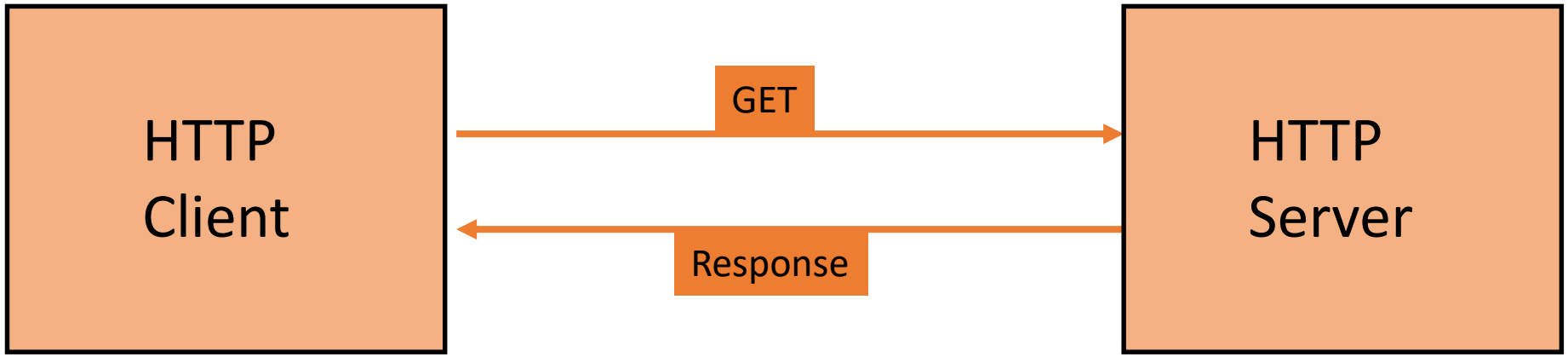
- HTTP client sends its request message: “GET …”
- HTTP server sends a status response: “200 OK”
- HTTP server sends requested file
- Browser displays document

- Clicking a link sets off a chain of events across the Internet!
- Let’s see how protocols & layers come into play…

Example: HTTP

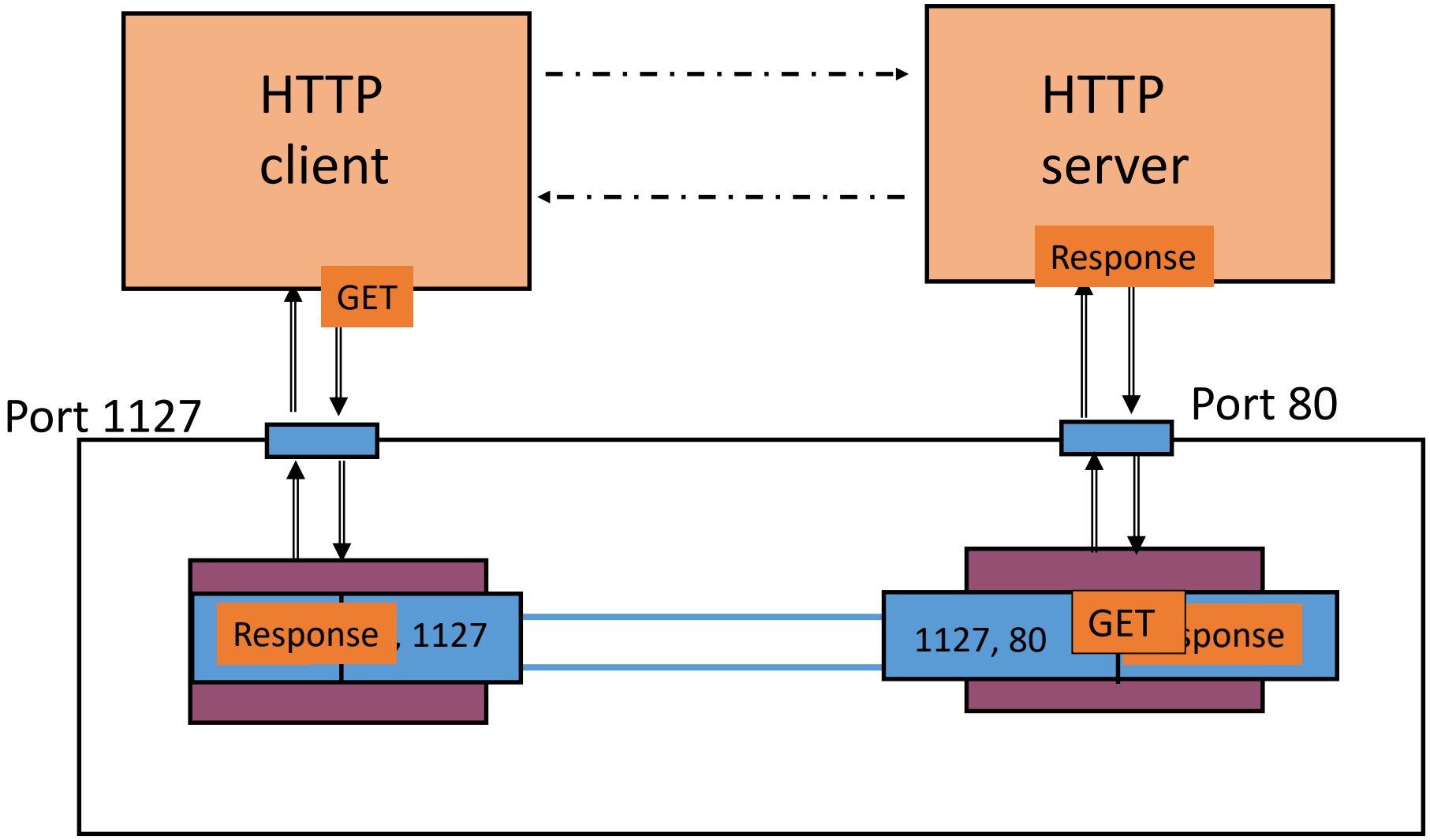
- HTTP is an application layer protocol
- Retrieves documents on behalf of a browser application program
- HTTP specifies fields in request messages and response messages
  - Request types; Response codes
  - Content type, options, cookies, …
- HTTP specifies actions to be taken upon receipt of certain messages

HTTP Protocol



- HTTP assumes messages can be exchanged directly between HTTP client and HTTP server
- In fact, HTTP client and server are processes running in two different machines across the Internet
- HTTP uses the reliable stream transfer service provided by TCP

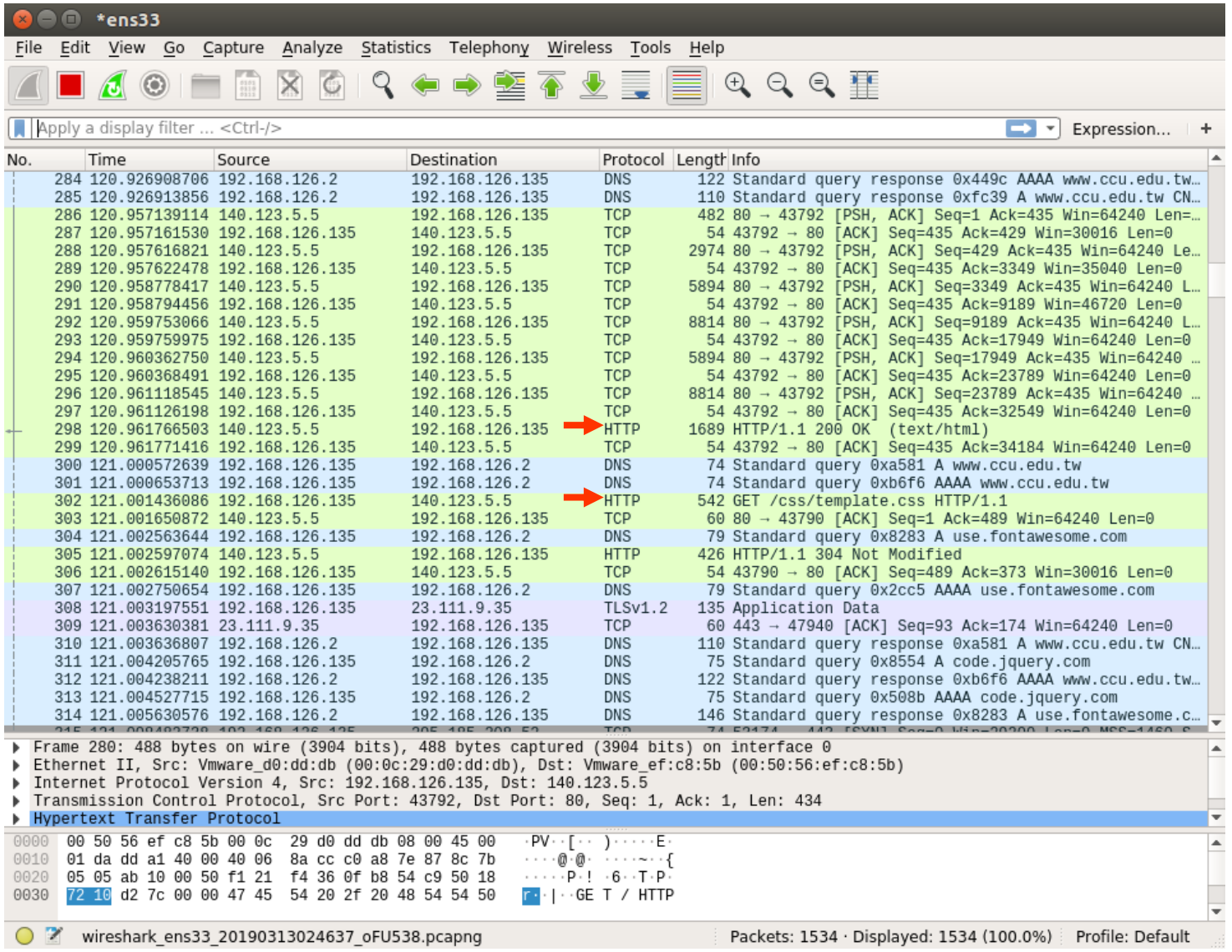
HTTP uses service of TCP



HTTP Protocol

- HTTP servers use well-known port 80
- Client request / Server reply
- Stateless: server does not keep any information about client
- HTTP 1.0 new TCP connection per request/reply (non-persistent)
- HTTP 1.1 persistent operation is default

HTTP Typical Exchange



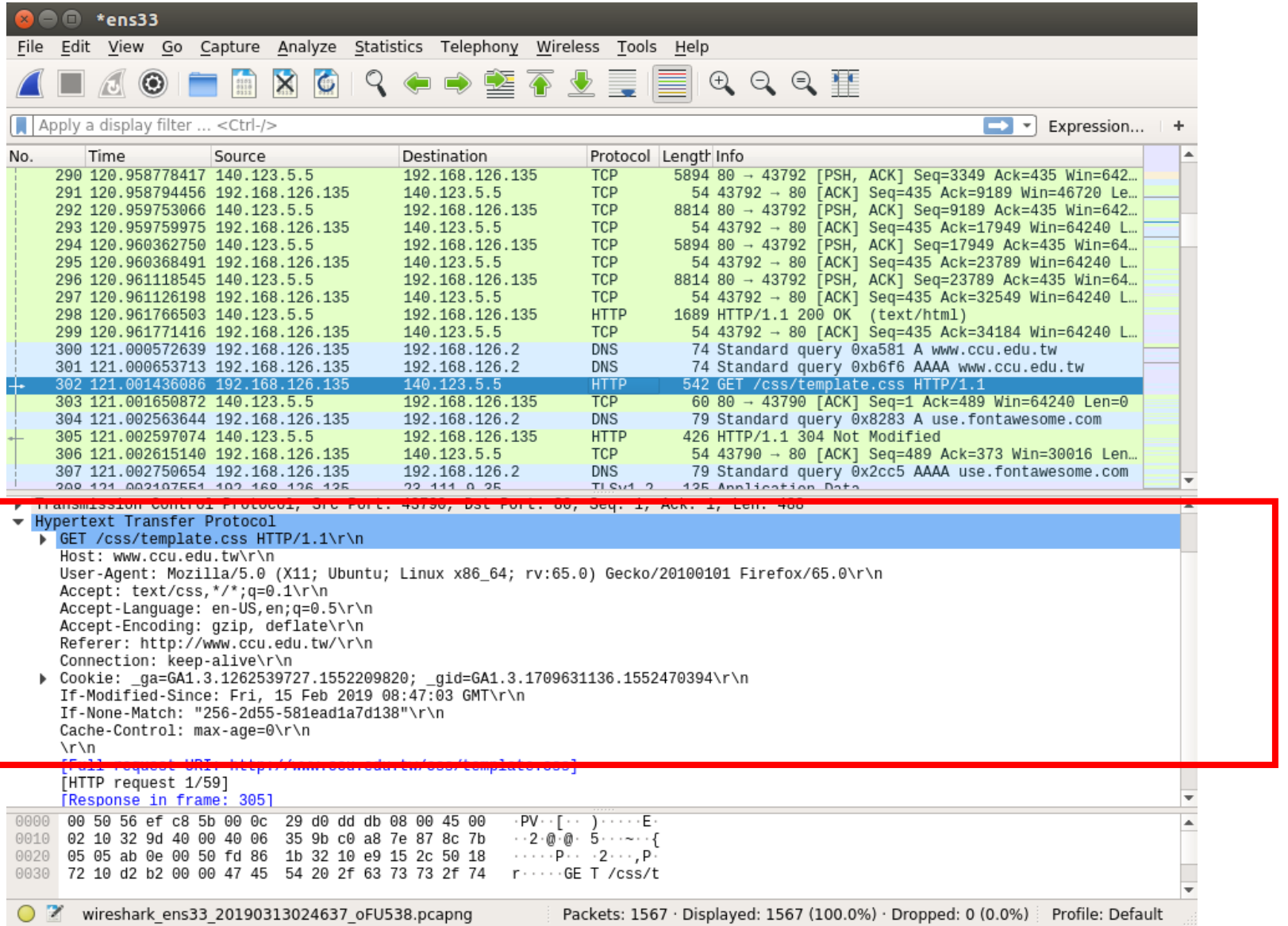
HTTP Message Formats

- HTTP messages written in ASCII text
- Request Message Format
  - *Request Line (Each line ends with carriage return)*
    - *Method URL HTTP-Version \r\n*
    - Method specifies action to apply to object
    - URL specifies object
  - *Header Lines (Ea. line ends with carriage return)*
    - *Attribute Name: Attribute Value*
    - E.g. type of client, content, identity of requester, …
    - Last header line has extra carriage return)
  - *Entity Body (Content)*
    - Additional information to server

HTTP Request Methods

Request method	Meaning
GET	Retrieve information (object) identified by the URL.
HEAD	Retrieve meta-information about the object, but do not transfer the object; Can be used to find out if a document has changed.
POST	Send information to a URL (using the entity body) and retrieve result; used when a user fills out a form in a browser.
PUT	Store information in location named by URL
DELETE	Remove object identified by URL
TRACE	Trace HTTP forwarding through proxies, tunnels, etc.
OPTIONS	Used to determine the capabilities of the server, or characteristics of a named resource.

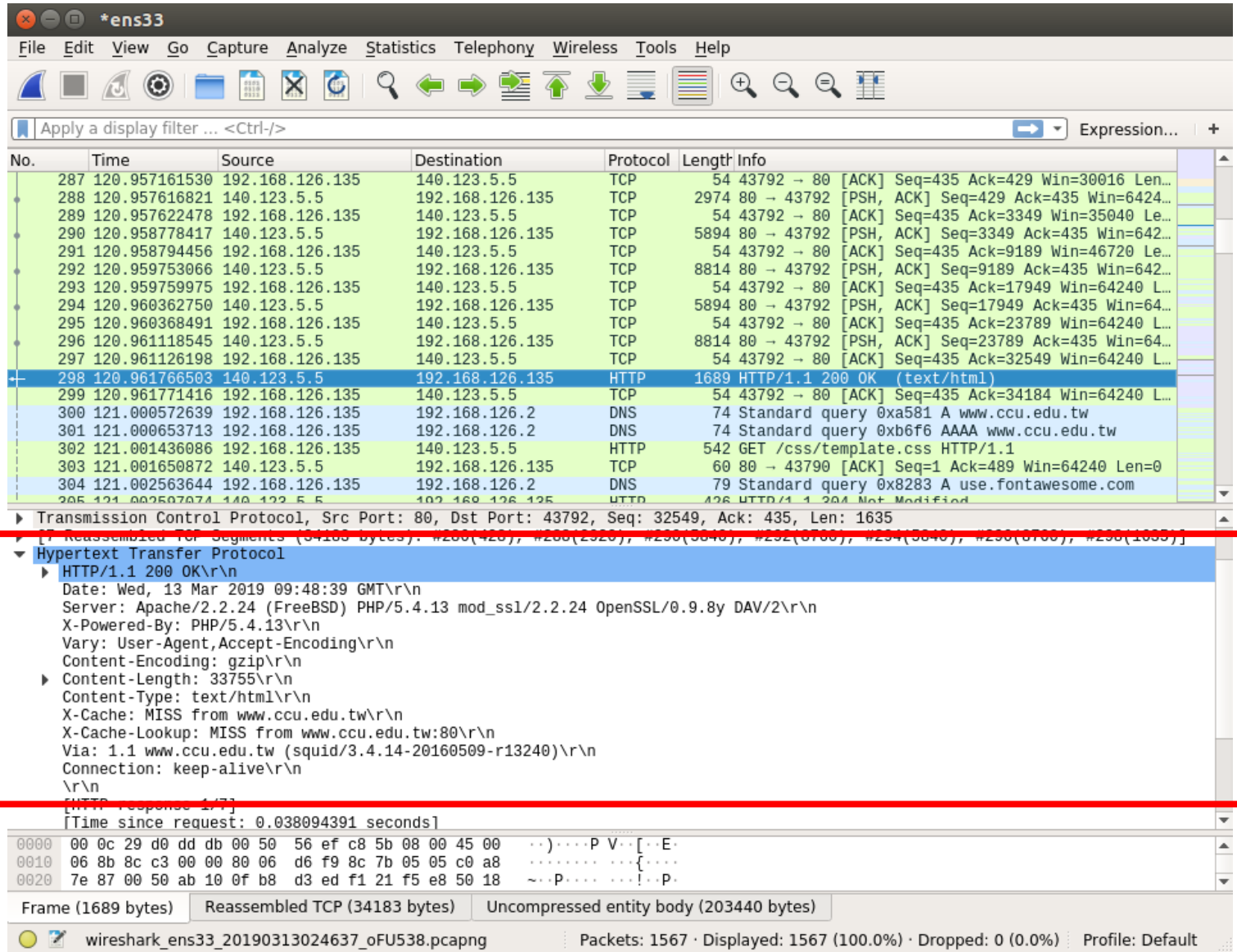
HTTP Request Message



HTTP Response Message

- Response Message Format
  - *Status Line*
    - *HTTP-Version Status-Code Message*
    - Status Code: 3-digit code indicating result
    - E.g. HTTP/1.0 200 OK
  - *Headers Section*
    - Information about object transferred to client
    - E.g. server type, content length, content type, …
  - *Content*
    - Object (document)

HTTP Response Message

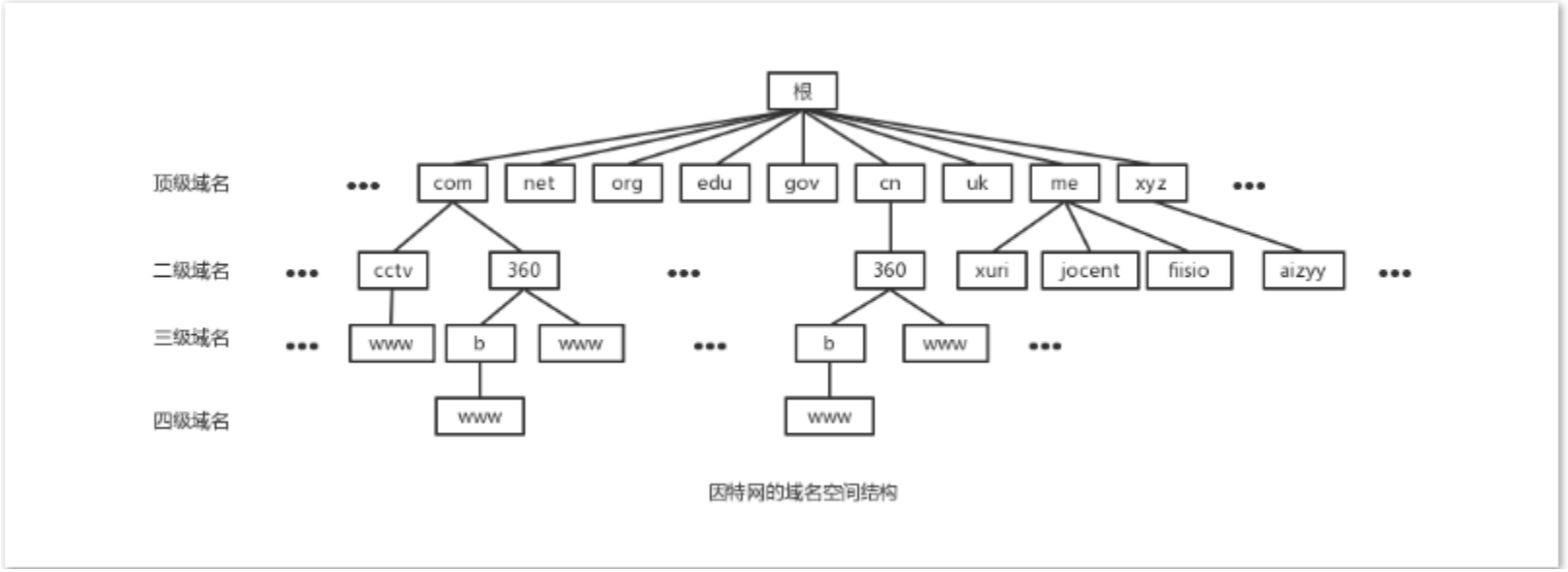


Example: DNS Protocol

- DNS protocol is an application layer protocol
- DNS is a distributed database that resides in multiple machines in the Internet
- DNS protocol different types's queries
  - iterated query
  - Recursive Queries
  - DNS usually involves short messages and so uses service provided by UDP
- Well-known port 53

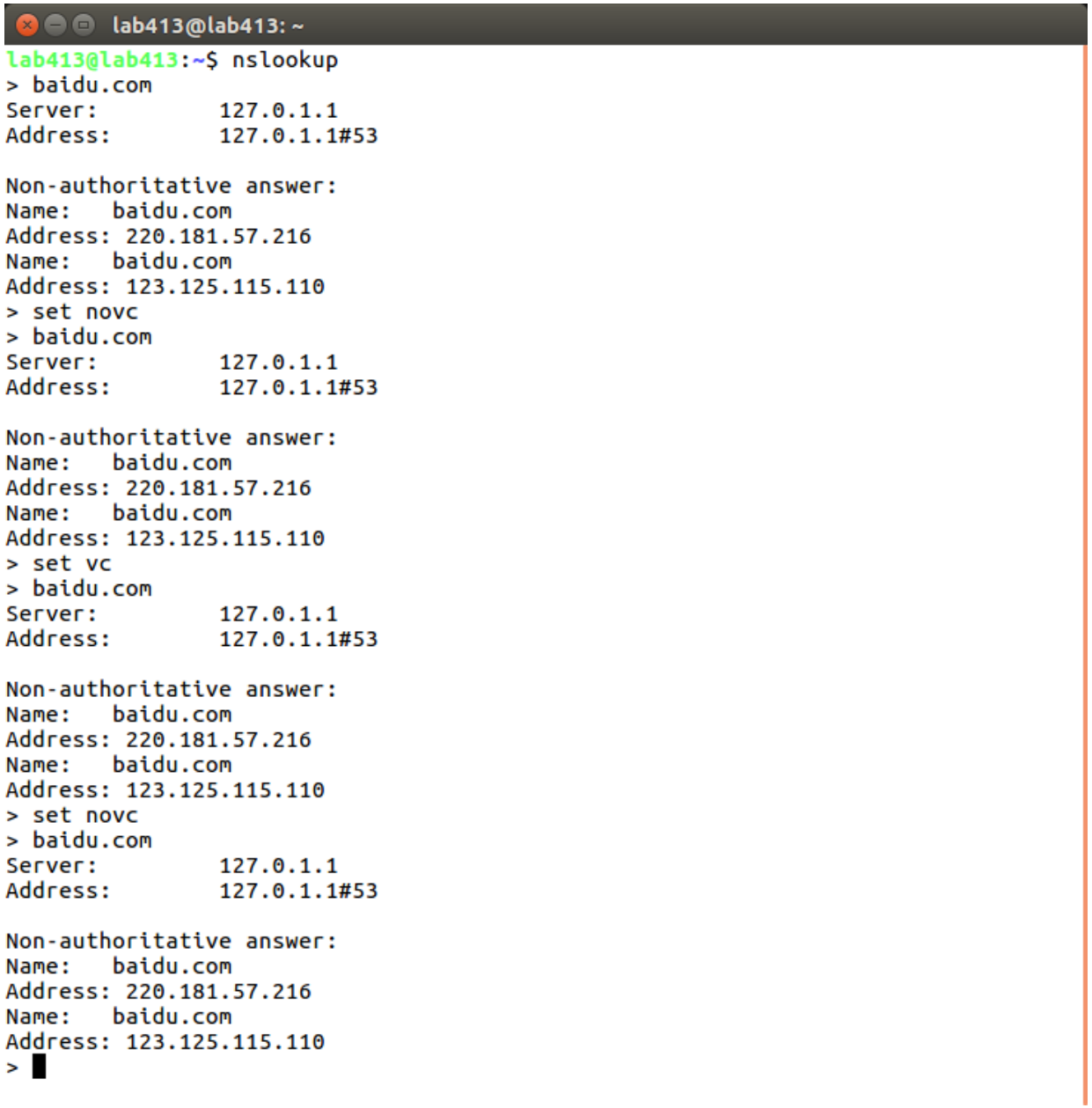
Domain域名結構

比如 www.360.com ，由點來區分成三個域名 www、360、com，其中 com 是頂級域名（TLD，Top Level Domain），360 是二級域名（SLD，Second Level Domain），www 是三級域名

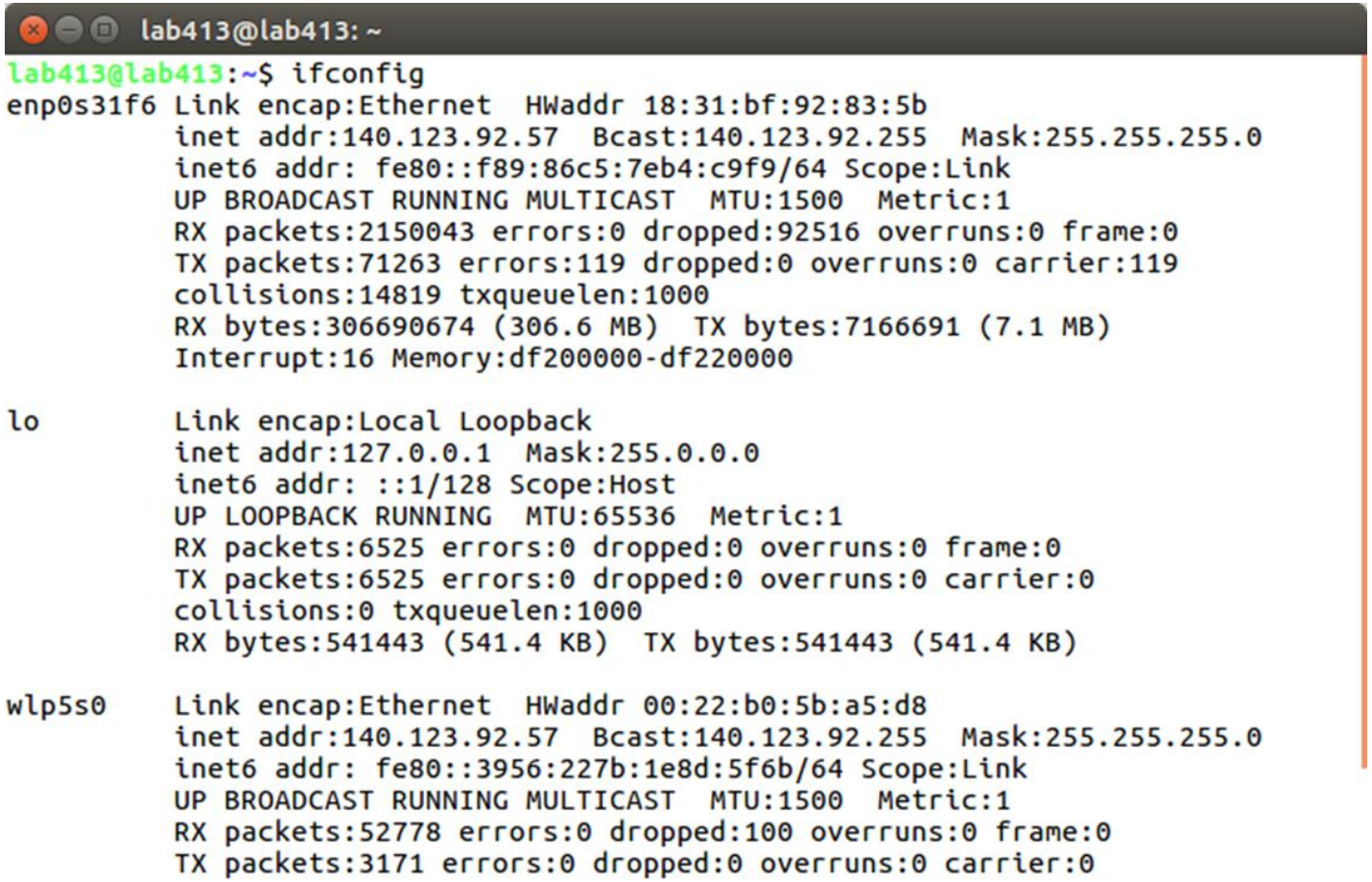


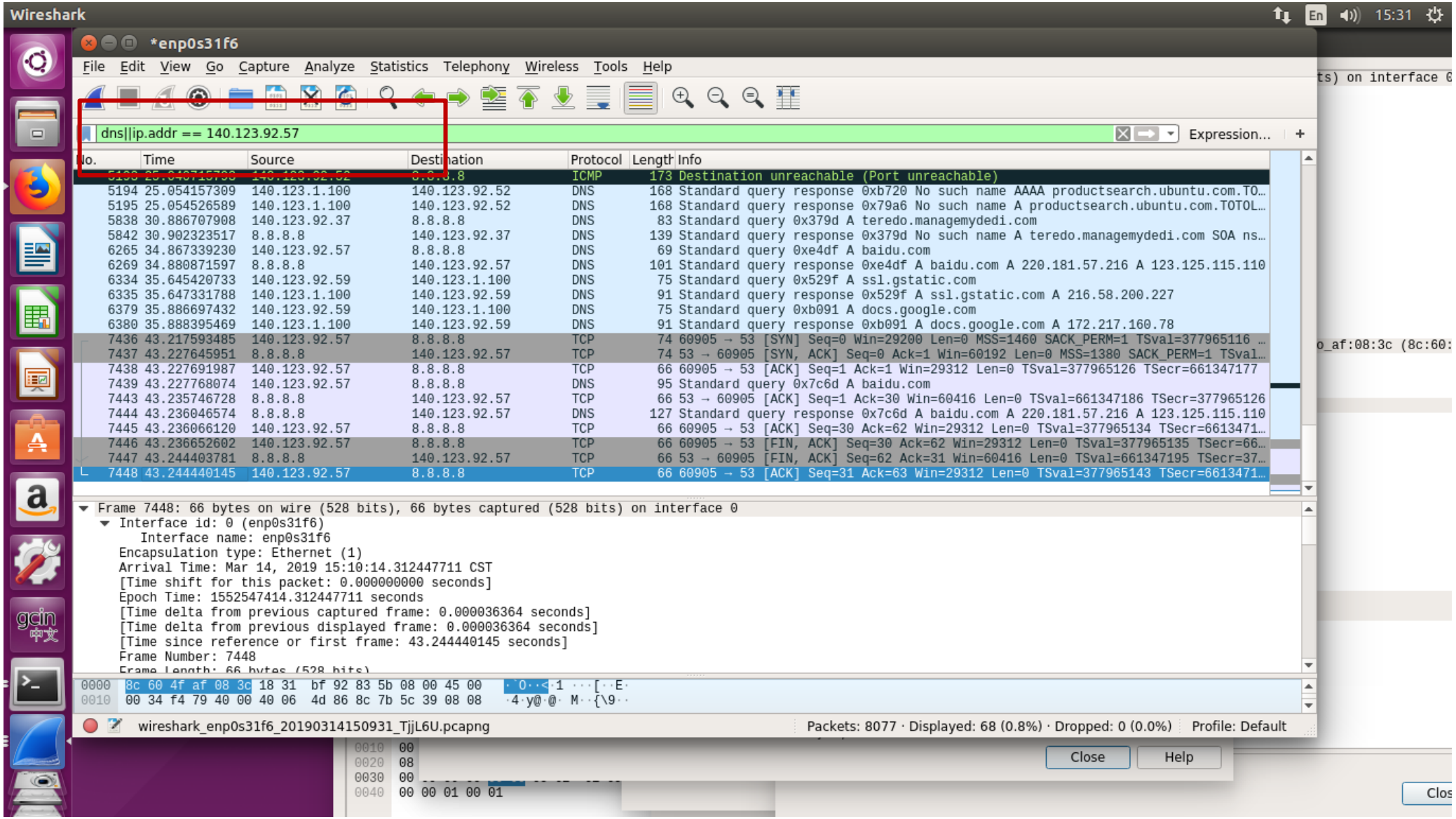
nslookup介紹

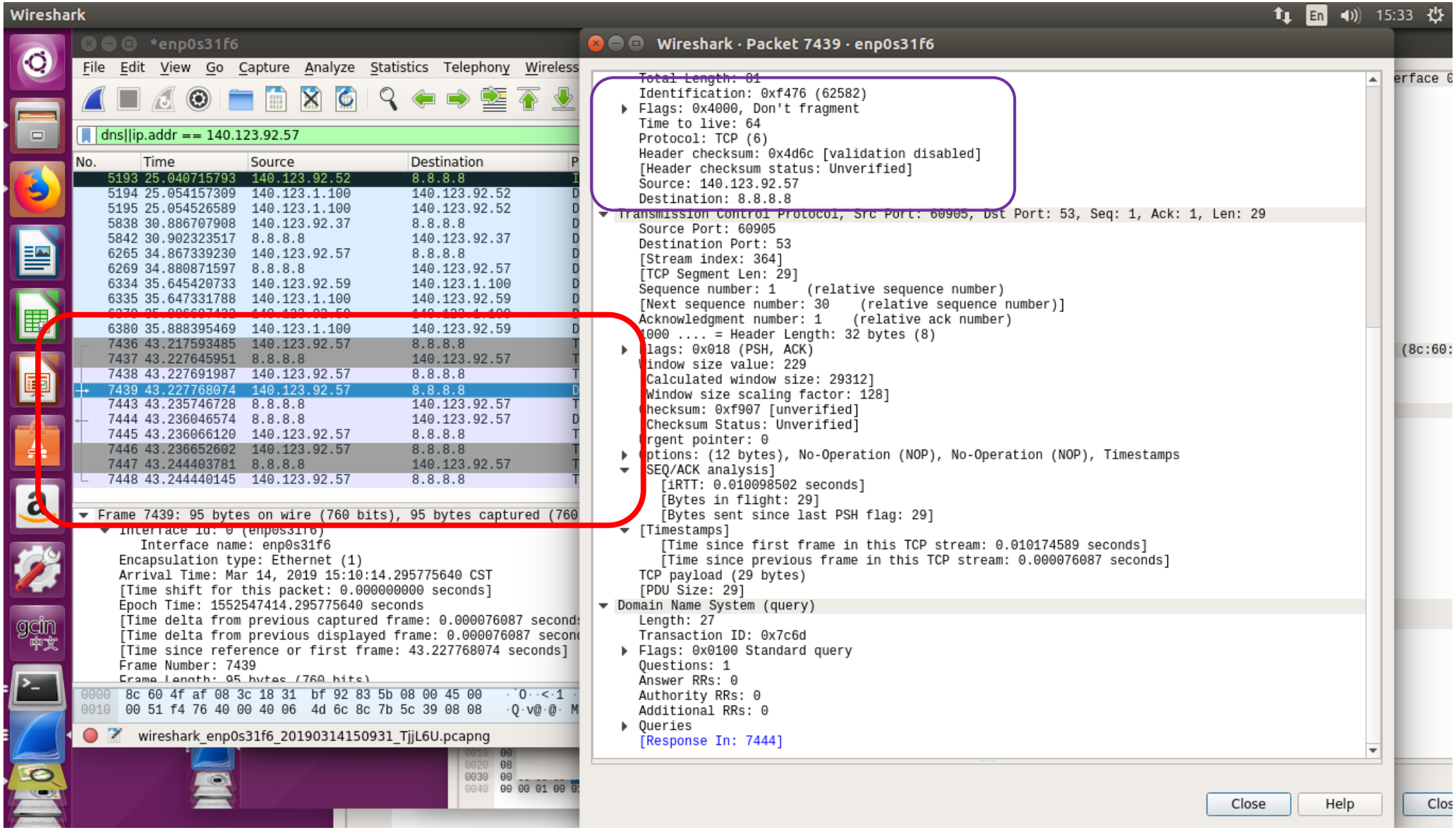
- Nslookup 是用來做DNS測試的工具
- 在terminal 輸入 $nslookup ($sudo nslookup)
- >set vc (學校電腦無法使用，參考用)
- 設定 DNS query over TCP
- >set novc
- 設定 DNS query over UDP


lab413@lab413: ~
lab413@lab413:~$ nslookup
> baidu.com
Server: 127.0.1.1
Address: 127.0.1.1#53
Non-authoritative answer:
Name: baidu.com
Address: 220.181.57.216
Name: baidu.com
Address: 123.125.115.110
> set novc
> baidu.com
Server: 127.0.1.1
Address: 127.0.1.1#53
Non-authoritative answer:
Name: baidu.com
Address: 220.181.57.216
Name: baidu.com
Address: 123.125.115.110
> set vc
> baidu.com
Server: 127.0.1.1
Address: 127.0.1.1#53
Non-authoritative answer:
Name: baidu.com
Address: 220.181.57.216
Name: baidu.com
Address: 123.125.115.110
> set novc
> baidu.com
Server: 127.0.1.1
Address: 127.0.1.1#53
Non-authoritative answer:
Name: baidu.com
Address: 220.181.57.216
Name: baidu.com
Address: 123.125.115.110
>

ifconfig--觀察與修改網路介面的相關參數




Wireshark
*enp0s31f6
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns||ip.addr == 140.123.92.57
Expression...
No. Time Source Destination Protocol Length Info
5194 25.054157309 140.123.1.100 140.123.92.52 DNS 168 Standard query response 0xb720 No such name AAAA productsearch.ubuntu.com.TO…
5195 25.054526589 140.123.1.100 140.123.92.52 DNS 168 Standard query response 0x79a6 No such name A productsearch.ubuntu.com.TOTOL…
5838 30.886707908 140.123.92.37 8.8.8.8 DNS 83 Standard query 0x379d A teredo.managemydedi.com
5842 30.902323517 8.8.8.8 140.123.92.37 DNS 139 Standard query response 0x379d No such name A teredo.managemydedi.com SOA ns…
6265 34.867339230 140.123.92.57 8.8.8.8 DNS 69 Standard query 0xe4df A baidu.com
6269 34.880871597 8.8.8.8 140.123.92.57 DNS 101 Standard query response 0xe4df A baidu.com A 220.181.57.216 A 123.125.115.110
6334 35.645420733 140.123.92.59 140.123.1.100 DNS 75 Standard query 0x529f A ssl.gstatic.com
6335 35.647331788 140.123.1.100 140.123.92.59 DNS 91 Standard query response 0x529f A ssl.gstatic.com A 216.58.200.227
6379 35.886697432 140.123.92.59 140.123.1.100 DNS 75 Standard query 0xb091 A docs.google.com
6380 35.888395469 140.123.1.100 140.123.92.59 DNS 91 Standard query response 0xb091 A docs.google.com A 172.217.160.78
7436 43.217593485 140.123.92.57 8.8.8.8 TCP 74 60905 → 53 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=377965116 …
7437 43.227645951 8.8.8.8 140.123.92.57 TCP 74 53 → 60905 [SYN, ACK] Seq=0 Ack=1 Win=60192 Len=0 MSS=1380 SACK_PERM=1 TSval…
7438 43.227691987 140.123.92.57 8.8.8.8 TCP 66 60905 → 53 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=377965126 TSecr=661347177
7439 43.227768074 140.123.92.57 8.8.8.8 DNS 95 Standard query 0x7c6d A baidu.com
7443 43.235746728 8.8.8.8 140.123.92.57 TCP 66 53 → 60905 [ACK] Seq=1 Ack=30 Win=60416 Len=0 TSval=661347186 TSecr=377965126
7444 43.236046574 8.8.8.8 140.123.92.57 DNS 127 Standard query response 0x7c6d A baidu.com A 220.181.57.216 A 123.125.115.110
7445 43.236066120 140.123.92.57 8.8.8.8 TCP 66 60905 → 53 [ACK] Seq=30 Ack=62 Win=29312 Len=0 TSval=377965134 TSecr=6613471…
7446 43.236652602 140.123.92.57 8.8.8.8 TCP 66 60905 → 53 [FIN, ACK] Seq=30 Ack=62 Win=29312 Len=0 TSval=377965135 TSecr=66…
7447 43.244403781 8.8.8.8 140.123.92.57 TCP 66 53 → 60905 [FIN, ACK] Seq=62 Ack=31 Win=60416 Len=0 TSval=661347195 TSecr=37…
7448 43.244440145 140.123.92.57 8.8.8.8 TCP 66 60905 → 53 [ACK] Seq=31 Ack=63 Win=29312 Len=0 TSval=377965143 TSecr=6613471…
Frame 7448: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Interface id: 0 (enp0s31f6)
Interface name: enp0s31f6
Encapsulation type: Ethernet (1)
Arrival Time: Mar 14, 2019 15:10:14.312447711 CST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1552547414.312447711 seconds
[Time delta from previous captured frame: 0.000036364 seconds]
[Time delta from previous displayed frame: 0.000036364 seconds]
[Time since reference or first frame: 43.244440145 seconds]
Frame Number: 7448
0000 8c 60 4f af 08 3c 18 31 bf 92 83 5b 08 00 45 00
0010 00 34 f4 79 40 00 40 06 4d 86 8c 7b 5c 39 08 08
wireshark_enp0s31f6_20190314150931_TjjL6U.pcapng
Packets: 8077 · Displayed: 68 (0.8%) · Dropped: 0 (0.0%)
Profile: Default
Close
Help


Wireshark
*enp0s31f6
File Edit View Go Capture Analyze Statistics Telephony Wireless
dns||ip.addr == 140.123.92.57
No. Time Source Destination
5193 25.040715793 140.123.92.52 8.8.8.8
5194 25.054157309 140.123.1.100 140.123.92.52
5195 25.054526589 140.123.1.100 140.123.92.52
5838 30.886707908 140.123.92.37 8.8.8.8
5842 30.902323517 8.8.8.8 140.123.92.37
6265 34.867339230 140.123.92.57 8.8.8.8
6269 34.880871597 8.8.8.8 140.123.92.57
6334 35.645420733 140.123.92.59 140.123.1.100
6335 35.647331788 140.123.1.100 140.123.92.59
6380 35.888395469 140.123.1.100 140.123.92.59
7436 43.217593485 140.123.92.57 8.8.8.8
7437 43.227645951 8.8.8.8 140.123.92.57
7438 43.227691987 140.123.92.57 8.8.8.8
7439 43.227768074 140.123.92.57 8.8.8.8
7443 43.235746728 8.8.8.8 140.123.92.57
7444 43.236046574 8.8.8.8 140.123.92.57
7445 43.236066120 140.123.92.57 8.8.8.8
7446 43.236652602 140.123.92.57 8.8.8.8
7447 43.244403781 8.8.8.8 140.123.92.57
7448 43.244440145 140.123.92.57 8.8.8.8
Frame 7439: 95 bytes on wire (760 bits), 95 bytes captured (760
Interface name: enp0s31f6
Encapsulation type: Ethernet (1)
Arrival Time: Mar 14, 2019 15:10:14.295775640 CST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1552547414.295775640 seconds
[Time delta from previous captured frame: 0.000076087 seconds]
[Time delta from previous displayed frame: 0.000076087 seconds]
[Time since reference or first frame: 43.227768074 seconds]
Frame Number: 7439
0000 8c 60 4f af 08 3c 18 31 bf 92 83 5b 08 00 45 00
0010 00 51 f4 76 40 00 40 06 4d 6c 8c 7b 5c 39 08 08
wireshark_enp0s31f6_20190314150931_TjjL6U.pcapng
Wireshark · Packet 7439 · enp0s31f6
Total Length: 81
Identification: 0xf476 (62582)
Flags: 0x4000, Don't fragment
Time to live: 64
Protocol: TCP (6)
Header checksum: 0x4d6c [validation disabled]
[Header checksum status: Unverified]
Source: 140.123.92.57
Destination: 8.8.8.8
Transmission Control Protocol, Src Port: 60905, Dst Port: 53, Seq: 1, Ack: 1, Len: 29
Source Port: 60905
Destination Port: 53
[Stream index: 364]
[TCP Segment Len: 29]
Sequence number: 1 (relative sequence number)
[Next sequence number: 30 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 229
[Calculated window size: 29312]
[Window size scaling factor: 128]
Checksum: 0xf907 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[iRTT: 0.010098502 seconds]
[Bytes in flight: 29]
[Bytes sent since last PSH flag: 29]
[Timestamps]
[Time since first frame in this TCP stream: 0.010174589 seconds]
[Time since previous frame in this TCP stream: 0.000076087 seconds]
TCP payload (29 bytes)
[PDU Size: 29]
Domain Name System (query)
Length: 27
Transaction ID: 0x7c6d
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
[Response In: 7444]
Close
Help

UDP/TCP

- UDP的速度會較TCP快一些
- 用UDP的傳輸資料的可靠度較TCP來的低
- 掉了一些封包仍然可以正常運行的，這一類的服務我們就會運用UDP服務來進行。像是DNS服務，我們就是要能夠快速的查詢到網址所對應的IP才行。

LAB1 TCP、UDP

- 課堂要求

找出TCP三向交握和UDP的封包。

- 報告

配合封包內容解釋TCP三向交握的原理。
(解釋SYN、ACK..等)
比較TCP和UDP。

LAB2 分析HTTP協議

- 課堂要求

打開使用HTTP的網站140.123.5.5

觀察HTTP協議中有哪些內容並截圖。

(找出request、response各一)

- 報告

配合封包內容解釋其中Host、accept、cookie…等。

LAB3 DNS分析

- 課堂要求

學會使用nslookup

學會用wireshark判斷連線是使用UDP還是TCP

- 報告

解釋為什麼DNS適合用UDP protocol 而不是TCP protocol。