Computer Networks - Lab 05

OBJECTIVES

After these Lab students shall be able to perform

- Introduction with Wire shark
- Relation OSI and TCP/IP model
- OSI and TCP/IP Layer Analysis via Wireshark
- HTTP and HTTPS analysis using Wireshark
- Http packet sniffing on wire shark
- Http Get and Http Ok.

PRE-LAB READING ASSIGNMENT

Remember the delivered lecture carefully.

Prepared by: Engr. Khuram Shahzad

Table of Contents

Computer Networks - Lab 05 1

OBJECTIVES 1

PRE-LAB READING ASSIGNMENT 1

OSI Network Layer Analysis via Wireshark 3

OSI model and TCP/IP model: 3

Relation OSI and TCP/IP model: 4

What we see in Wireshark? 5

HTTP analysis using Wireshark 11

What is HTTP? 11

HTTP Methods: 11

HTTP is Wiresahark: 12

HTTP packets exchanges in Wireshark: 13

HTTP GET: 15

HTTP OK: 17

OSI Network Layer Analysis via Wireshark

OSI model and TCP/IP model:

We all know that OSI (Open Systems Interconnection) is a reference model for how applications communicate over a network.
Here are the 7 layers according to OSI model:

Application Layer	[Layer 7]
Presentation Layer	[Layer 6]
Session Layer	[Layer 5]
Transport Layer	[Layer 4]
Network Layer	[Layer 3]
Data Link Layer	[Layer 2]
Physical Layer	[Layer 1]

There is another network model which is TCP/IP.

Here are the 4 layers according to TCP/IP model:

Application Layer	**[Layer 4]**
Transport Layer	[Layer 3]
Internet Layer	[Layer 2]
Network Access Layer	[Layer 1]

Relation OSI and TCP/IP model:

Below is the relation between OSI model and TCP/IP model.

OSI Model TCP/IP Model

Application Layer	**Application Layer**
Presentation Layer	
Session Layer	
Transport Layer	Transport Layer
Network Layer	Internet Layer
Data Link Layer	Network access Layer
Physical Layer	

Now the question comes, in **Wireshark what model we should be expecting?**
Actually in Wireshark we observe below layers

Application Layer	[Layer 5]
Transport Layer	[Layer 4]
Network Layer	[Layer 3]

Data Link Layer	[Layer 2]
Physical Layer	[Layer 1]

Now we understand that the above layers are not exactly OSI or TCP/IP but a combination of both models.

Let’s look into Wireshark capture and understand better.

What we see in Wireshark?

We will take some protocols as example and understand the layers through Wireshark. The interesting part is all protocol does not have all the layers.

Note:
As Wireshark decodes packets at Data Link layer so we will not get physical layer information always. In some cases, capturing adapter provides some physical layer information and can be displayed through Wireshark.
So here are the sequence layers seen in Wireshark

Data Link Layer
Network Layer
Transport Layer
Application Layer

Hope you understand that Wireshark is just showing in reverse order. If physical layer information is given to Wireshark then that time we should see physical layer information on top of Data link. See below picture.

Physical Layer

Data Link Layer

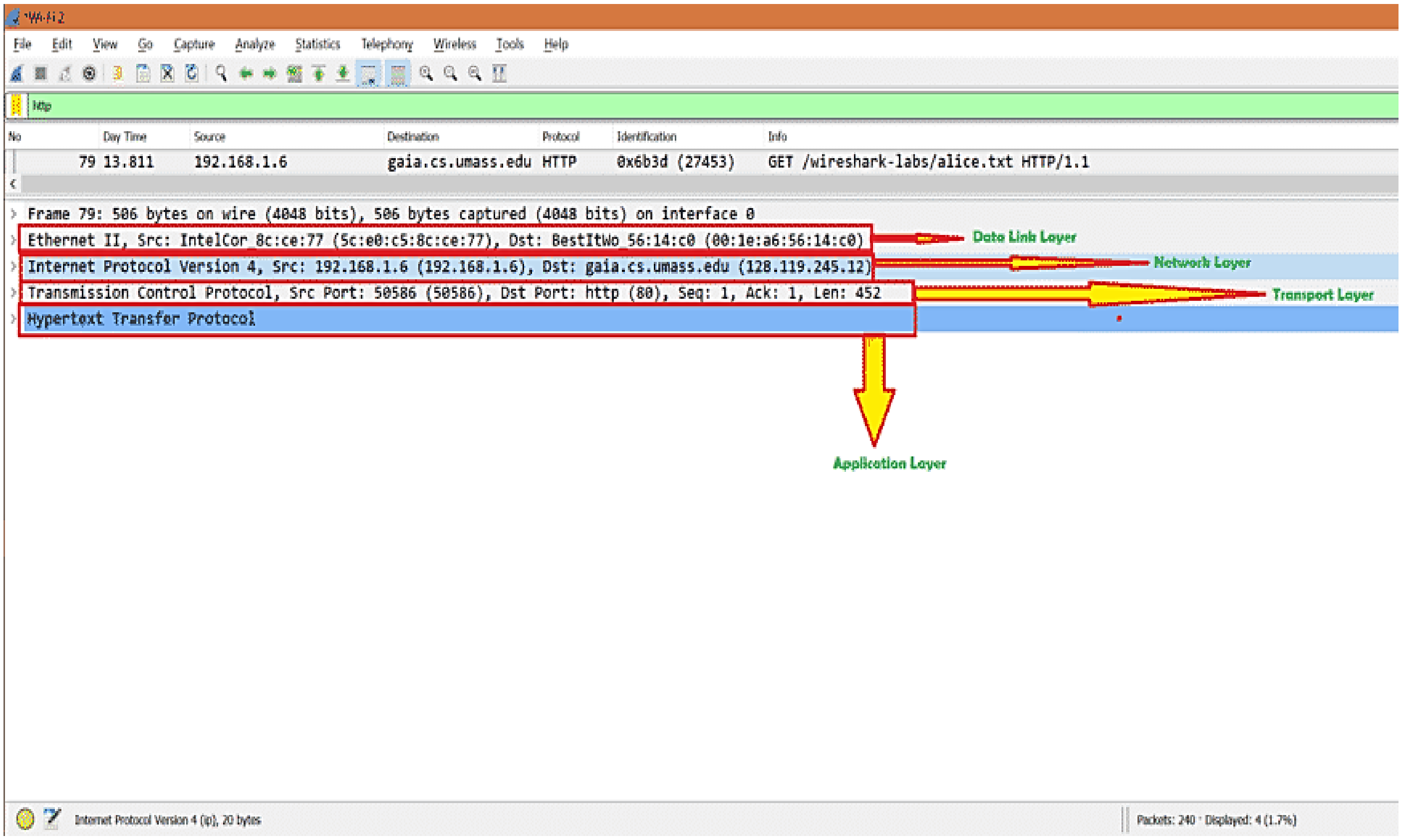
Network Layer

Transport Layer

Application Layer

1.1.1.1 HTTP [It has 4 layers]:

You can follow below link to understand HTTP through WiresharkHere is the screenshot of a HTTP packet where we can see 4 layers.


*Wi-Fi 2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Data Link Layer
Network Layer
Transport Layer
Application Layer
Internet Protocol Version 4 (ip), 20 bytes
Packets: 240 · Displayed: 4 (1.7%)

```
No    Day Time   Source        Destination         Protocol  Identification   Info
   79 13.811     192.168.1.6   gaia.cs.umass.edu   HTTP      0x6b3d (27453)   GET /wireshark-labs/alice.txt HTTP/1.1
```

```
> Frame 79: 506 bytes on wire (4048 bits), 506 bytes captured (4048 bits) on interface 0
> Ethernet II, Src: IntelCor_8c:ce:77 (5c:e0:c5:8c:ce:77), Dst: BestItWo_56:14:c0 (00:1e:a6:56:14:c0)
> Internet Protocol Version 4, Src: 192.168.1.6 (192.168.1.6), Dst: gaia.cs.umass.edu (128.119.245.12)
> Transmission Control Protocol, Src Port: 50586 (50586), Dst Port: http (80), Seq: 1, Ack: 1, Len: 452
> Hypertext Transfer Protocol
```

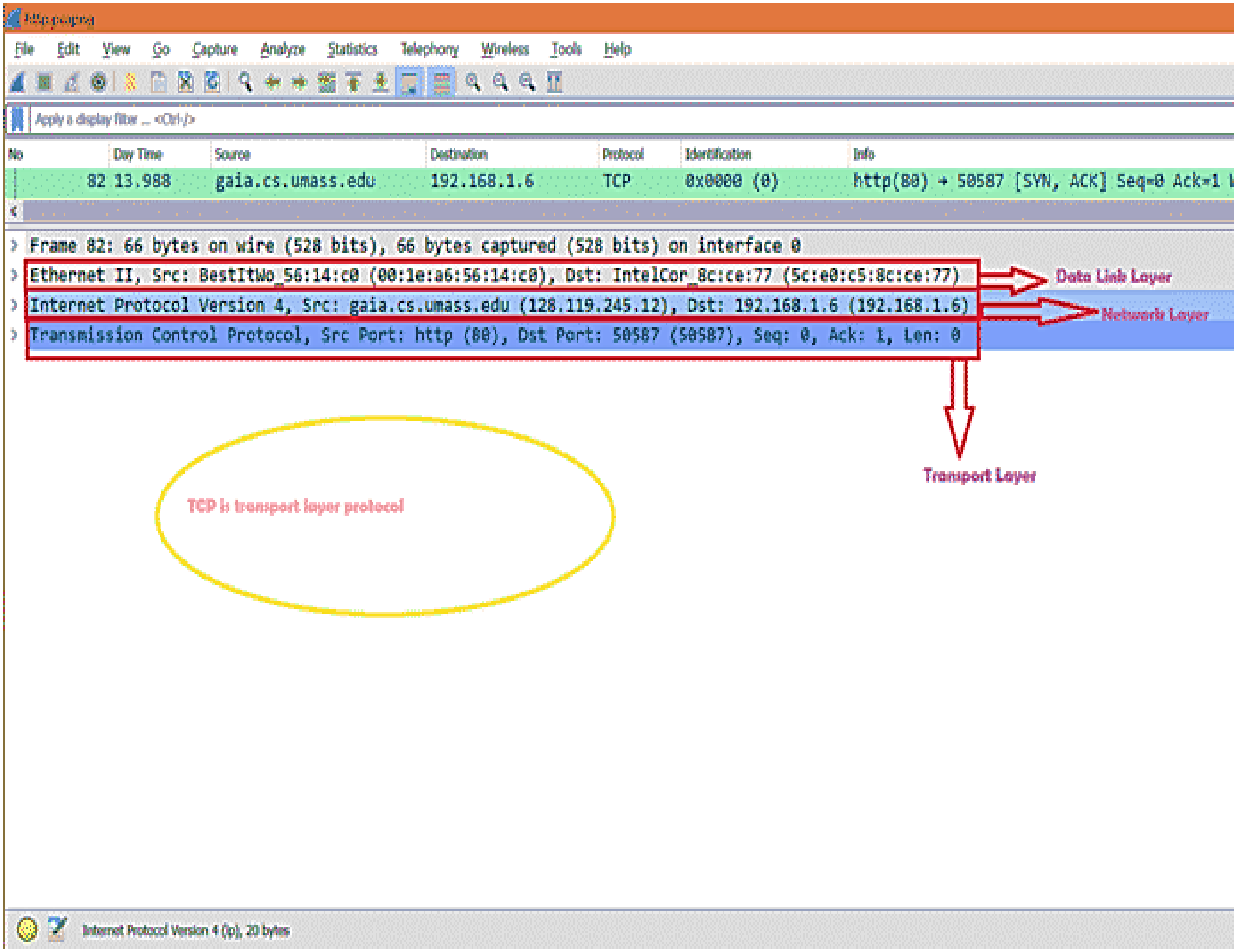
We know HTTP is an application layer so we see application layer also.

Prepared by: Engr. Khuram Shahzad

Now let’s see a transport layer protocol in Wireshark.

1.1.1.2 TCP [It has 3 layers]:

Here is the screenshot of a TCP packet where we can see 3 layers.



No	Day Time	Source	Destination	Protocol	Identification	Info
82	13.988	gaia.cs.umass.edu	192.168.1.6	TCP	0x0000 (0)	http(80) → 50587 [SYN, ACK] Seq=0 Ack=1

```
> Frame 82: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: BestItWo_56:14:c0 (00:1e:a6:56:14:c0), Dst: IntelCor_8c:ce:77 (5c:e0:c5:8c:ce:77)
> Internet Protocol Version 4, Src: gaia.cs.umass.edu (128.119.245.12), Dst: 192.168.1.6 (192.168.1.6)
> Transmission Control Protocol, Src Port: http (80), Dst Port: 50587 (50587), Seq: 0, Ack: 1, Len: 0
```

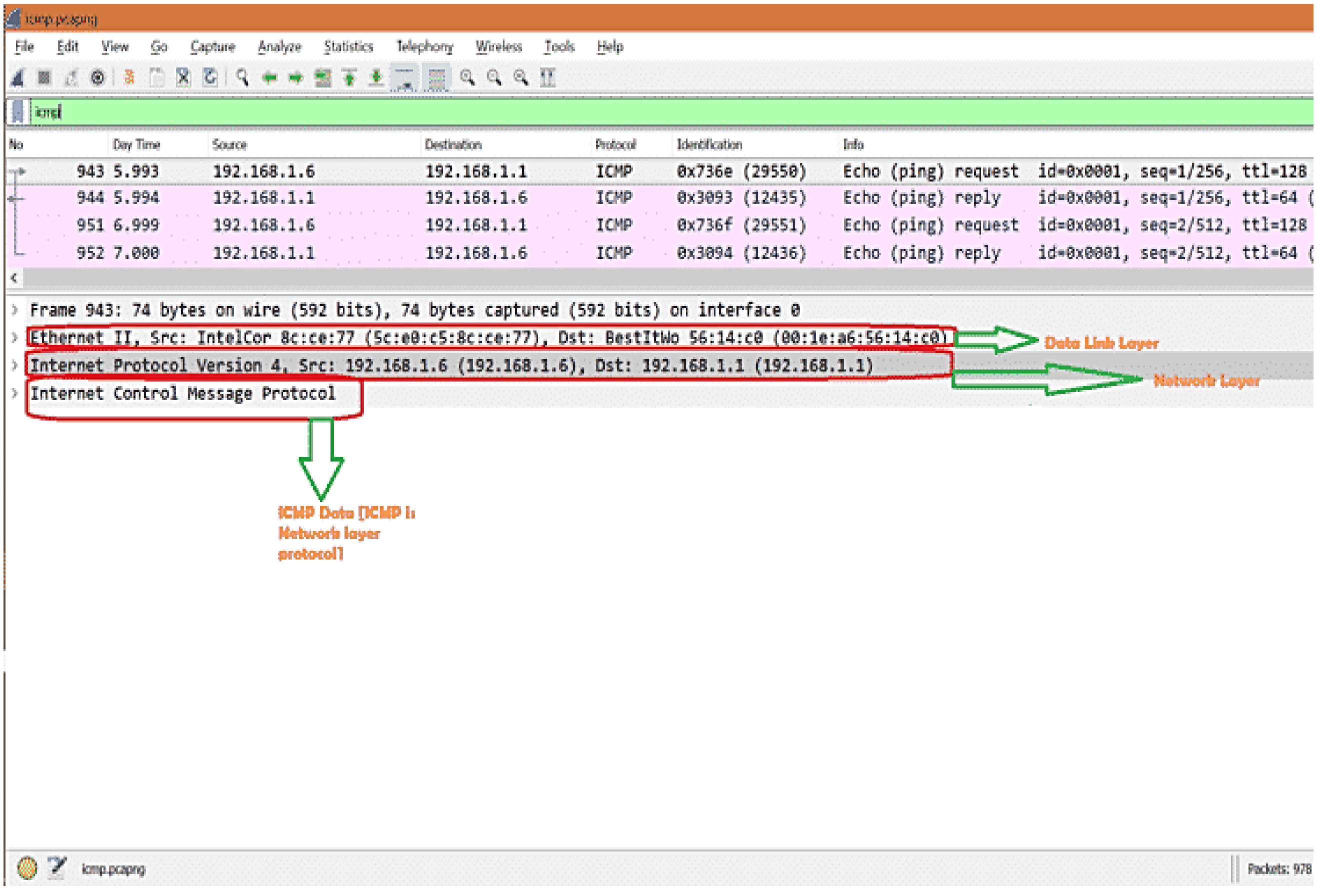
Let’s see ICMP packet.

1.1.1.3 ICMP [It has 2 layers]:

Here is the screenshot of an ICMP frame where we can see 2 layers.

-

Prepared by: Engr. Khuram Shahzad



```
No        Day Time   Source         Destination    Protocol  Identification   Info
     943 5.993       192.168.1.6    192.168.1.1    ICMP      0x736e (29550)   Echo (ping) request  id=0x0001, seq=1/256, ttl=128
     944 5.994       192.168.1.1    192.168.1.6    ICMP      0x3093 (12435)   Echo (ping) reply    id=0x0001, seq=1/256, ttl=64 (
     951 6.999       192.168.1.6    192.168.1.1    ICMP      0x736f (29551)   Echo (ping) request  id=0x0001, seq=2/512, ttl=128
     952 7.000       192.168.1.1    192.168.1.6    ICMP      0x3094 (12436)   Echo (ping) reply    id=0x0001, seq=2/512, ttl=64 (

> Frame 943: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: IntelCor 8c:ce:77 (5c:e0:c5:8c:ce:77), Dst: BestItWo 56:14:c0 (00:1e:a6:56:14:c0)
> Internet Protocol Version 4, Src: 192.168.1.6 (192.168.1.6), Dst: 192.168.1.1 (192.168.1.1)
> Internet Control Message Protocol
```

Data Link Layer

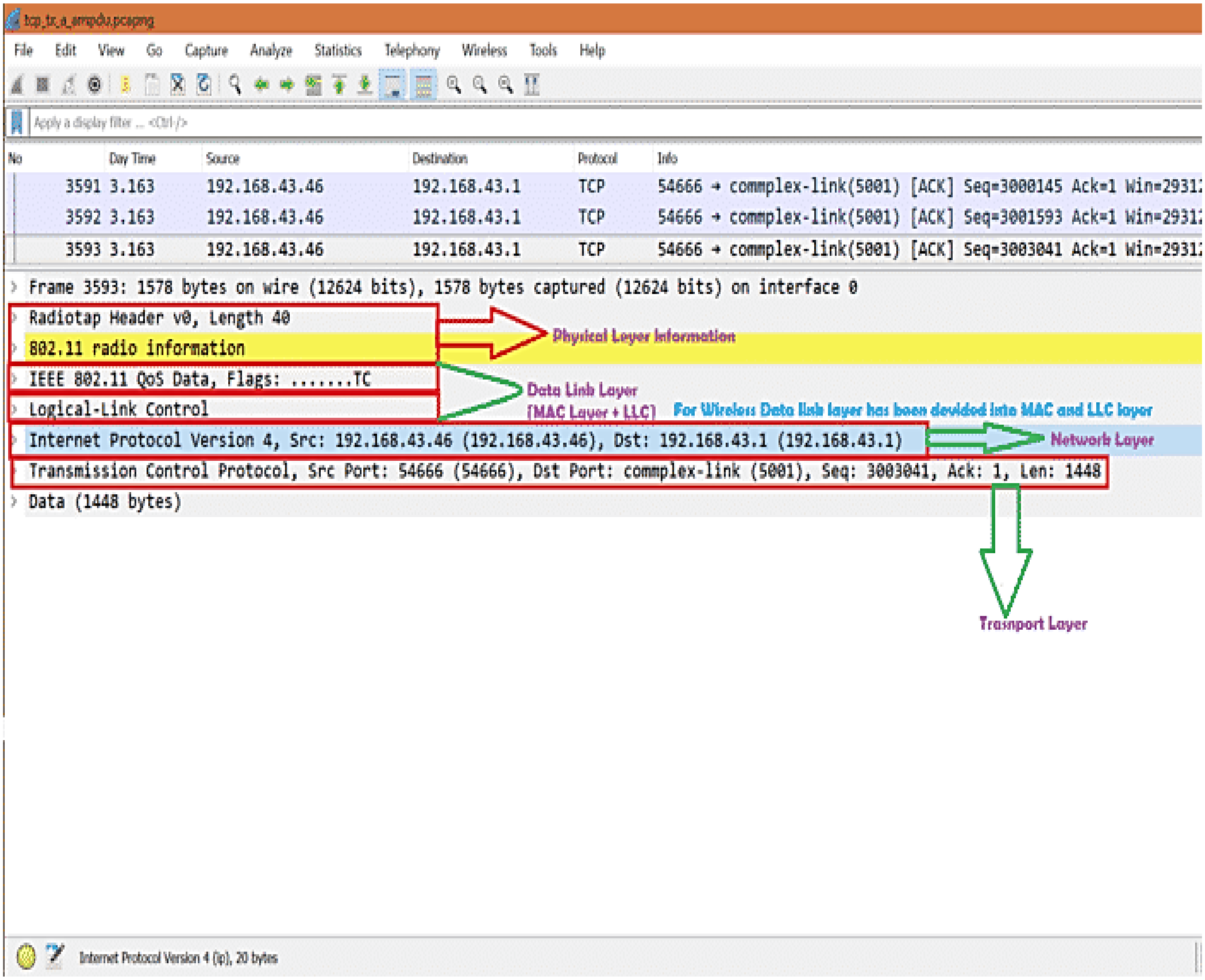
Network Layer

ICMP Data [ICMP is Network layer protocol]

Now let’s see one wireless TCP frame where we can see physical layer information.

1.1.1.4 <u>Wireless TCP [It has 4 layers]:</u>

Here is the screenshot of a TCP frame where we can see 4 layers including physical layer.



```
No        Day Time   Source           Destination     Protocol   Info
    3591  3.163      192.168.43.46    192.168.43.1    TCP        54666 → commplex-link(5001) [ACK] Seq=3000145 Ack=1 Win=2931
    3592  3.163      192.168.43.46    192.168.43.1    TCP        54666 → commplex-link(5001) [ACK] Seq=3001593 Ack=1 Win=2931
    3593  3.163      192.168.43.46    192.168.43.1    TCP        54666 → commplex-link(5001) [ACK] Seq=3003041 Ack=1 Win=2931
```

```
> Frame 3593: 1578 bytes on wire (12624 bits), 1578 bytes captured (12624 bits) on interface 0
> Radiotap Header v0, Length 40
> 802.11 radio information
> IEEE 802.11 QoS Data, Flags: .......TC
> Logical-Link Control
> Internet Protocol Version 4, Src: 192.168.43.46 (192.168.43.46), Dst: 192.168.43.1 (192.168.43.1)
> Transmission Control Protocol, Src Port: 54666 (54666), Dst Port: commplex-link (5001), Seq: 3003041, Ack: 1, Len: 1448
> Data (1448 bytes)
```

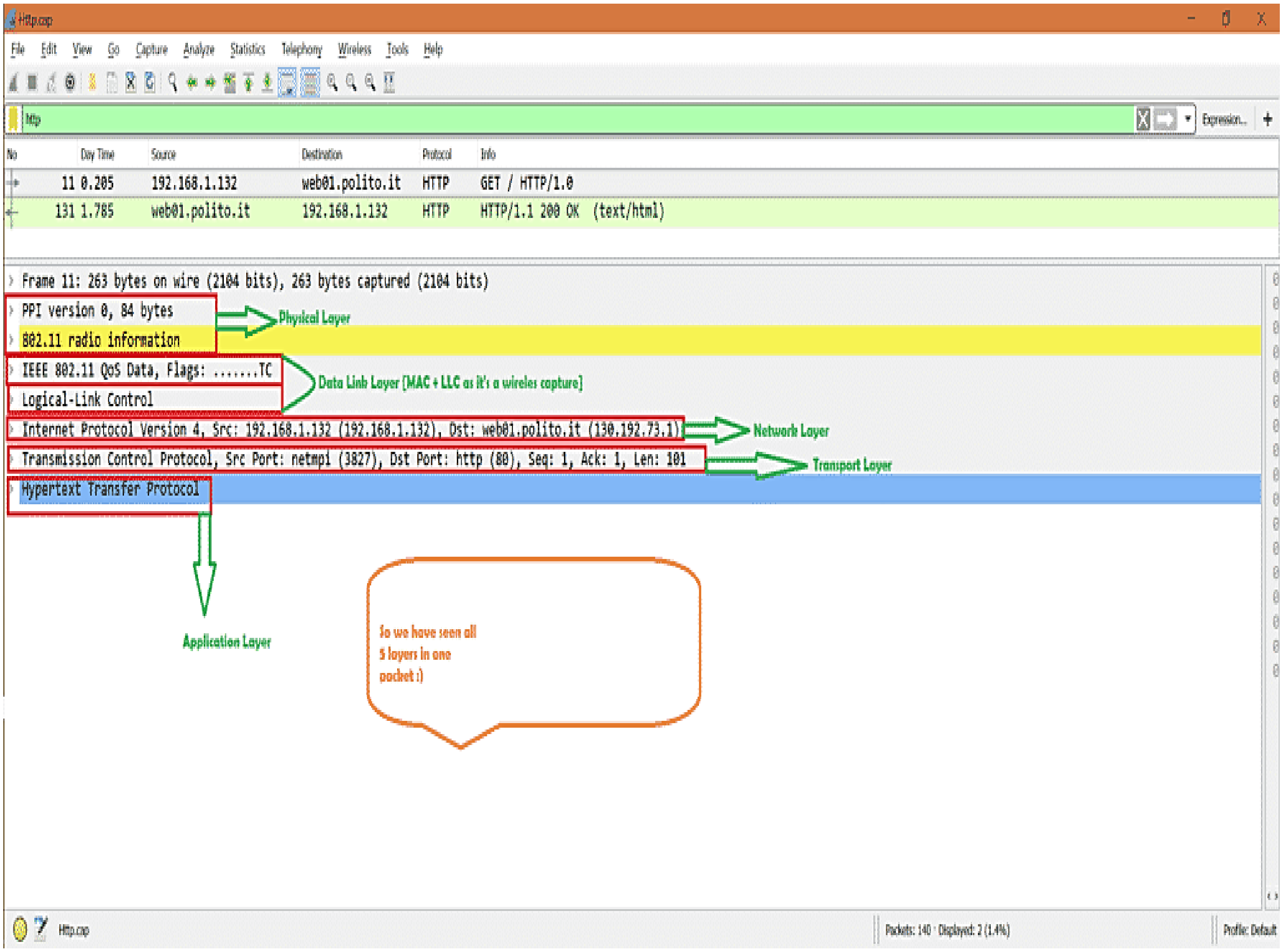
Internet Protocol Version 4 (ip), 20 bytes

As TCP is a transport layer protocol so we did not see any application layer protocol.

Now let’s see Wireless capture for HTTP and hope to see all 5 layers including Application layer and physical layer.

1.1.1.5 Wireless HTTp [It has all 5 layers]:

Here is the screenshot of a HTTP frame where we can see including Application layer and physical layer.



Http.cap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...
+

```
No    Day Time     Source           Destination       Protocol  Info
   11 0.205        192.168.1.132    web01.polito.it   HTTP      GET / HTTP/1.0
  131 1.785        web01.polito.it  192.168.1.132     HTTP      HTTP/1.1 200 OK  (text/html)

> Frame 11: 263 bytes on wire (2104 bits), 263 bytes captured (2104 bits)
> PPI version 0, 84 bytes
> 802.11 radio information
> IEEE 802.11 QoS Data, Flags: .......TC
> Logical-Link Control
> Internet Protocol Version 4, Src: 192.168.1.132 (192.168.1.132), Dst: web01.polito.it (130.192.73.1)
> Transmission Control Protocol, Src Port: netmpi (3827), Dst Port: http (80), Seq: 1, Ack: 1, Len: 101
> Hypertext Transfer Protocol
```

Physical Layer
Data Link Layer [MAC + LLC as it's a wireles capture]
Network Layer
Transport Layer
Application Layer
So we have seen all 5 layers in one packet :)
Http.cap
Packets: 140 · Displayed: 2 (1.4%)
Profile: Default

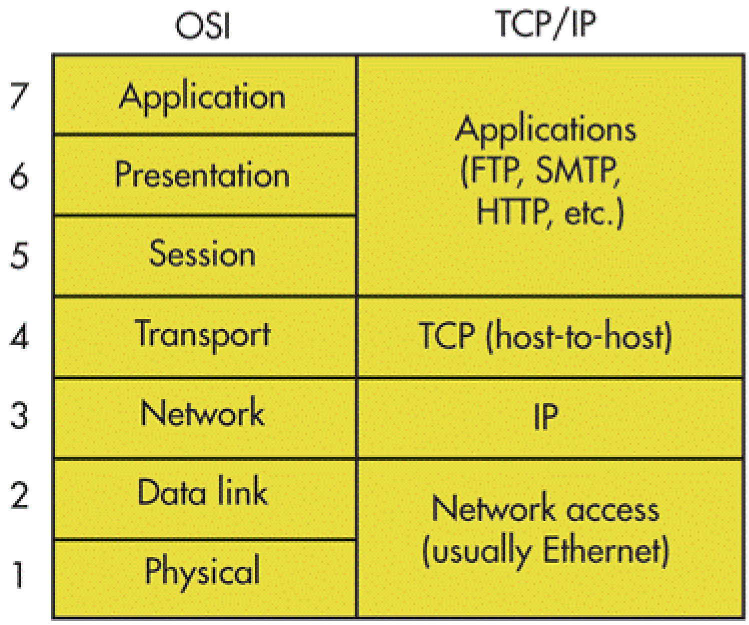
1.1.1.6 Summary:

In summary we can say that depending on protocol different layers can been seen in Wireshark.

HTTP analysis using Wireshark

What is HTTP?

First is all the full form of HTTP is HyperText Transfer Protocol. HTTP is an application layer protocol in ISO or TCP/IP model. See below picture to find out HTTP which resides under application layer.



	OSI	TCP/IP
7	Application	Applications (FTP, SMTP, HTTP, etc.)
6	Presentation	
5	Session	
4	Transport	TCP (host-to-host)
3	Network	IP
2	Data link	Network access (usually Ethernet)
1	Physical	

HTTP is used by the World Wide Web (w.w.w) and it defines how messages are formatted and transmitted by browser. So HTTP define reules what action should be taken when a browser receives HTTP command. And also HTTP defines rules for transmitting HTTP command to get data from server.
For example, when you enter a url in browser (Internet explorer, Chrome, Firefox, Safari etc) it actually sends an HTTP command to server.And server replies with appropiate command.

HTTP Methods:

There are some set of methods for HTTP/1.1 (This is HTTP version)

GET, HEAD, POST, PUT, DELETE, CONNECT, OPTION and TRACE.

We will not go in details of each method instead we will get to know about the methods which are seen quite often.Such as

GET: GET request asks data from web server. This is a main method used document retrival. We will see one practical example of this method.
POST: POST method is used when it’s required to send some data to server.

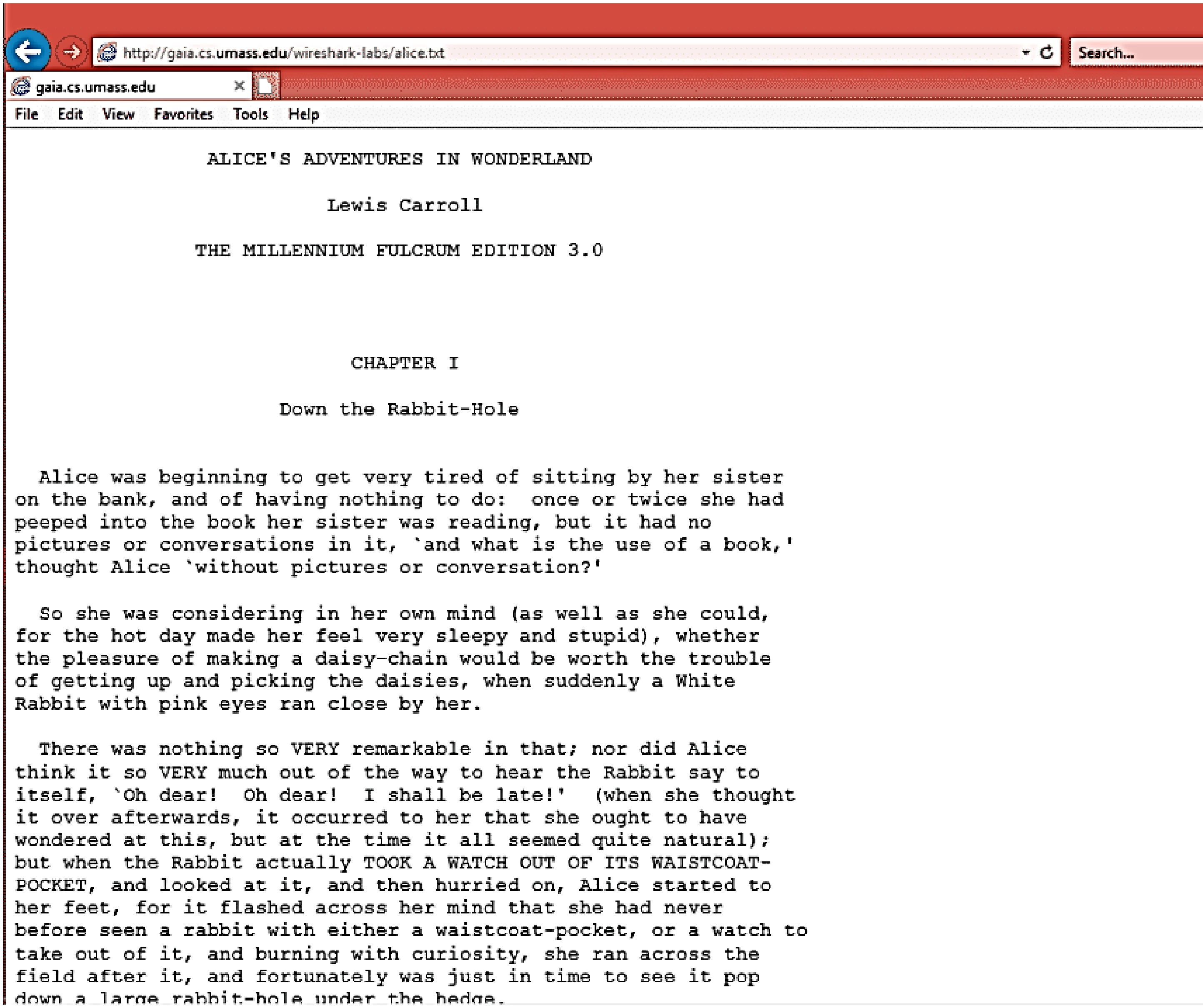
HTTP is Wiresahark:

Let’s try something practical to understand how HTTP works ?

So in this example we will download **“alice.txt” (Data file present in server)** from **“gaia.cs.umass.edu”** server.

Setps:

1. Open the URL http://gaia.cs.umass.edu/wireshark-labs/alice.txt [We know the full url for downloading alice.txt] in computer browser.
2. Now we see the downloaded file in browser. Here is the screenshot



```
                ALICE'S ADVENTURES IN WONDERLAND

                          Lewis Carroll

               THE MILLENNIUM FULCRUM EDITION 3.0




                            CHAPTER I

                      Down the Rabbit-Hole


  Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do:  once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, `and what is the use of a book,'
thought Alice `without pictures or conversation?'

  So she was considering in her own mind (as well as she could,
for the hot day made her feel very sleepy and stupid), whether
the pleasure of making a daisy-chain would be worth the trouble
of getting up and picking the daisies, when suddenly a White
Rabbit with pink eyes ran close by her.

  There was nothing so VERY remarkable in that; nor did Alice
think it so VERY much out of the way to hear the Rabbit say to
itself, `Oh dear!  Oh dear!  I shall be late!'  (when she thought
it over afterwards, it occurred to her that she ought to have
wondered at this, but at the time it all seemed quite natural);
but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-
POCKET, and looked at it, and then hurried on, Alice started to
her feet, for it flashed across her mind that she had never
before seen a rabbit with either a waistcoat-pocket, or a watch to
take out of it, and burning with curiosity, she ran across the
field after it, and fortunately was just in time to see it pop
down a large rabbit-hole under the hedge.
```

3. In parallel we have capture the packets in Wireshark.

HTTP packets exchanges in Wireshark:

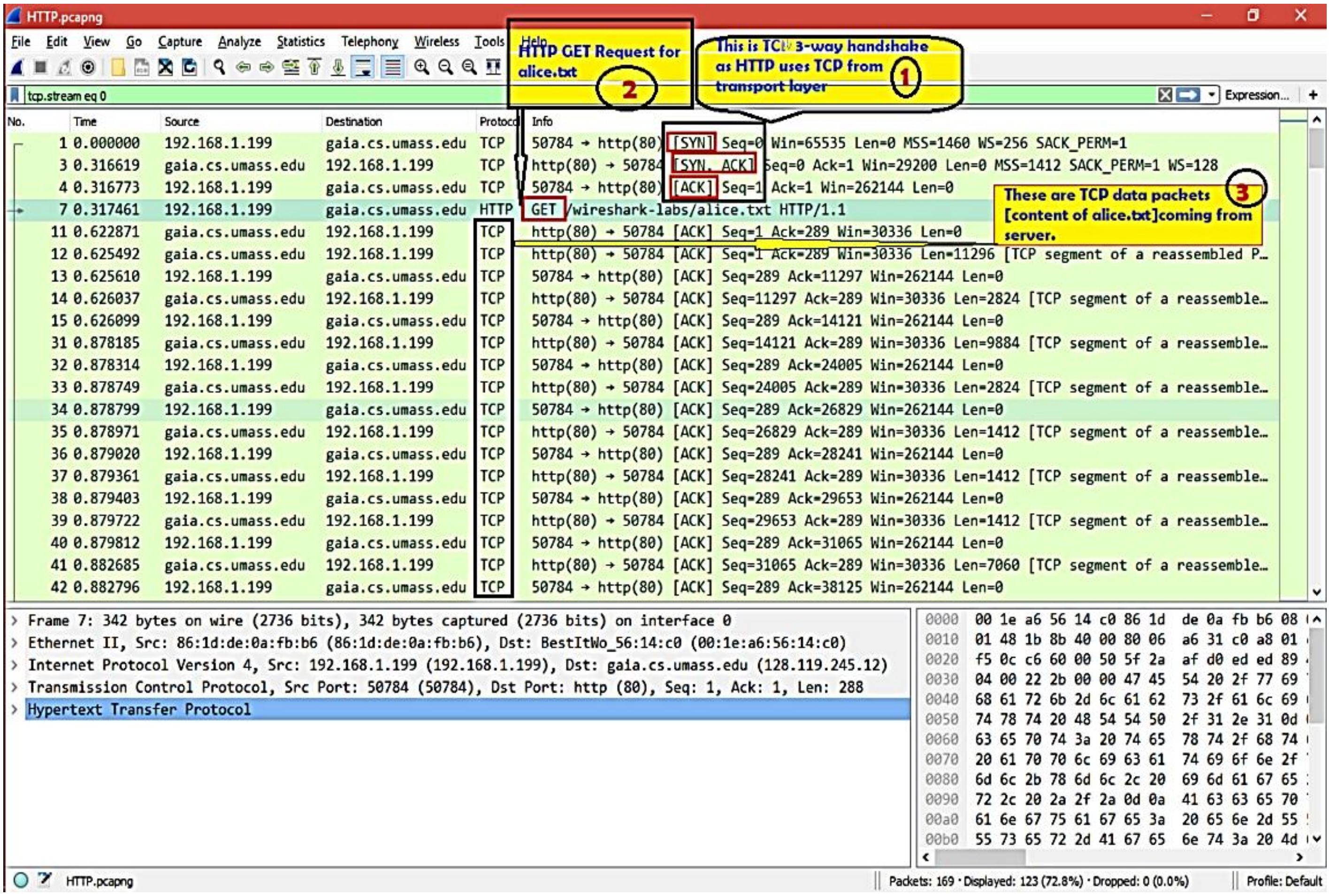
Before we go into HTTP we should know that HTTP uses port 80 and TCP as transport layer protocol [We will explain TCP in another topic discussion].

Now let’s see what happens in network when we put that URL and press enter in browser.

Here is the screenshot for

TCP 3-way handshake ——-> HTTP OK ——-> TCP Data [content of alice.txt] ——->

HTTP-OK


HTTP.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 0
Expression...
+

HTTP GET Request for alice.txt 2

This is TCP 3-way handshake as HTTP uses TCP from transport layer 1

These are TCP data packets [content of alice.txt]coming from server. 3

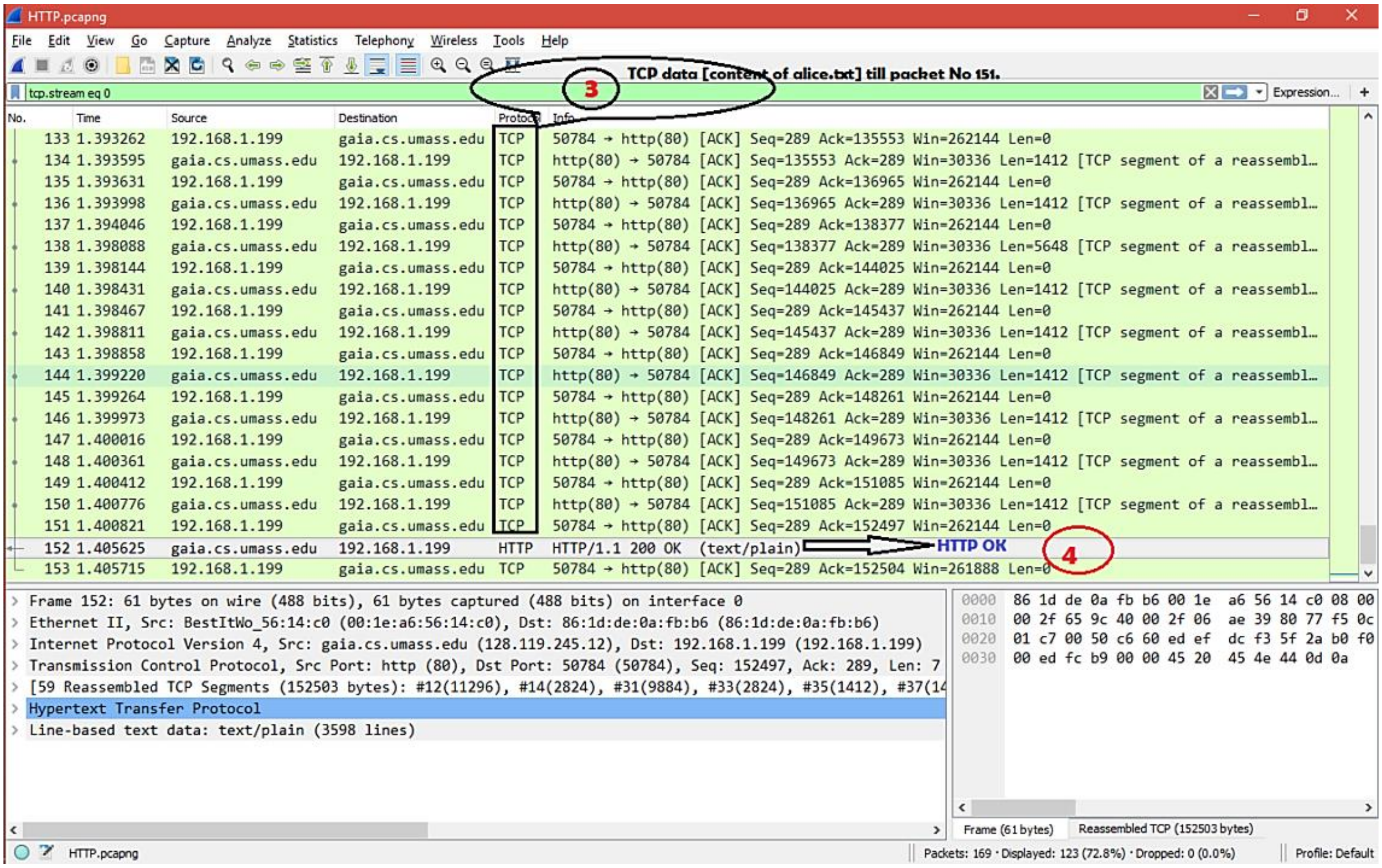
```
No.  Time      Source             Destination        Protocol Info
   1 0.000000  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
   3 0.316619  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1412 SACK_PERM=1 WS=128
   4 0.316773  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=1 Ack=1 Win=262144 Len=0
   7 0.317461  192.168.1.199      gaia.cs.umass.edu  HTTP  GET /wireshark-labs/alice.txt HTTP/1.1
  11 0.622871  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=1 Ack=289 Win=30336 Len=0
  12 0.625492  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=1 Ack=289 Win=30336 Len=11296 [TCP segment of a reassembled P…
  13 0.625610  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=289 Ack=11297 Win=262144 Len=0
  14 0.626037  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=11297 Ack=289 Win=30336 Len=2824 [TCP segment of a reassemble…
  15 0.626099  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=289 Ack=14121 Win=262144 Len=0
  31 0.878185  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=14121 Ack=289 Win=30336 Len=9884 [TCP segment of a reassemble…
  32 0.878314  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=289 Ack=24005 Win=262144 Len=0
  33 0.878749  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=24005 Ack=289 Win=30336 Len=2824 [TCP segment of a reassemble…
  34 0.878799  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=289 Ack=26829 Win=262144 Len=0
  35 0.878971  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=26829 Ack=289 Win=30336 Len=1412 [TCP segment of a reassemble…
  36 0.879020  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=289 Ack=28241 Win=262144 Len=0
  37 0.879361  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=28241 Ack=289 Win=30336 Len=1412 [TCP segment of a reassemble…
  38 0.879403  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=289 Ack=29653 Win=262144 Len=0
  39 0.879722  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=29653 Ack=289 Win=30336 Len=1412 [TCP segment of a reassemble…
  40 0.879812  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=289 Ack=31065 Win=262144 Len=0
  41 0.882685  gaia.cs.umass.edu  192.168.1.199      TCP   http(80) → 50784 [ACK] Seq=31065 Ack=289 Win=30336 Len=7060 [TCP segment of a reassemble…
  42 0.882796  192.168.1.199      gaia.cs.umass.edu  TCP   50784 → http(80) [ACK] Seq=289 Ack=38125 Win=262144 Len=0
```

```
> Frame 7: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
> Ethernet II, Src: 86:1d:de:0a:fb:b6 (86:1d:de:0a:fb:b6), Dst: BestItWo_56:14:c0 (00:1e:a6:56:14:c0)
> Internet Protocol Version 4, Src: 192.168.1.199 (192.168.1.199), Dst: gaia.cs.umass.edu (128.119.245.12)
> Transmission Control Protocol, Src Port: 50784 (50784), Dst Port: http (80), Seq: 1, Ack: 1, Len: 288
> Hypertext Transfer Protocol
```

```
0000  00 1e a6 56 14 c0 86 1d  de 0a fb b6 08
0010  01 48 1b 8b 40 00 80 06  a6 31 c0 a8 01
0020  f5 0c c6 60 00 50 5f 2a  af d0 ed ed 89
0030  04 00 22 2b 00 00 47 45  54 20 2f 77 69
0040  68 61 72 6b 2d 6c 61 62  73 2f 61 6c 69
0050  74 78 74 20 48 54 54 50  2f 31 2e 31 0d
0060  63 65 70 74 3a 20 74 65  78 74 2f 68 74
0070  20 61 70 70 6c 69 63 61  74 69 6f 6e 2f
0080  6d 6c 2b 78 6d 6c 2c 20  69 6d 61 67 65
0090  72 2c 20 2a 2f 2a 0d 0a  41 63 63 65 70
00a0  61 6e 67 75 61 67 65 3a  20 65 6e 2d 55
00b0  55 73 65 72 2d 41 67 65  6e 74 3a 20 4d
```

HTTP.pcapng | Packets: 169 · Displayed: 123 (72.8%) · Dropped: 0 (0.0%) | Profile: Default

Prepared by: Engr. Khuram Shahzad


HTTP.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 0
Expression...
3
4
Frame (61 bytes)
Reassembled TCP (152503 bytes)
HTTP.pcapng
Packets: 169 · Displayed: 123 (72.8%) · Dropped: 0 (0.0%)
Profile: Default

TCP data [content of alice.txt] till packet No 151.

```
No.   Time     Source             Destination        Protocol Info
  133 1.393262 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=135553 Win=262144 Len=0
  134 1.393595 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=135553 Ack=289 Win=30336 Len=1412 [TCP segment of a reassembl…
  135 1.393631 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=136965 Win=262144 Len=0
  136 1.393998 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=136965 Ack=289 Win=30336 Len=1412 [TCP segment of a reassembl…
  137 1.394046 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=138377 Win=262144 Len=0
  138 1.398088 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=138377 Ack=289 Win=30336 Len=5648 [TCP segment of a reassembl…
  139 1.398144 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=144025 Win=262144 Len=0
  140 1.398431 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=144025 Ack=289 Win=30336 Len=1412 [TCP segment of a reassembl…
  141 1.398467 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=145437 Win=262144 Len=0
  142 1.398811 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=145437 Ack=289 Win=30336 Len=1412 [TCP segment of a reassembl…
  143 1.398858 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=146849 Win=262144 Len=0
  144 1.399220 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=146849 Ack=289 Win=30336 Len=1412 [TCP segment of a reassembl…
  145 1.399264 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=148261 Win=262144 Len=0
  146 1.399973 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=148261 Ack=289 Win=30336 Len=1412 [TCP segment of a reassembl…
  147 1.400016 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=149673 Win=262144 Len=0
  148 1.400361 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=149673 Ack=289 Win=30336 Len=1412 [TCP segment of a reassembl…
  149 1.400412 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=151085 Win=262144 Len=0
  150 1.400776 gaia.cs.umass.edu  192.168.1.199      TCP  http(80) → 50784 [ACK] Seq=151085 Ack=289 Win=30336 Len=1412 [TCP segment of a reassembl…
  151 1.400821 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=152497 Win=262144 Len=0
  152 1.405625 gaia.cs.umass.edu  192.168.1.199      HTTP HTTP/1.1 200 OK  (text/plain)
  153 1.405715 192.168.1.199      gaia.cs.umass.edu  TCP  50784 → http(80) [ACK] Seq=289 Ack=152504 Win=261888 Len=0
```

HTTP OK

```
> Frame 152: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface 0
> Ethernet II, Src: BestItWo_56:14:c0 (00:1e:a6:56:14:c0), Dst: 86:1d:de:0a:fb:b6 (86:1d:de:0a:fb:b6)
> Internet Protocol Version 4, Src: gaia.cs.umass.edu (128.119.245.12), Dst: 192.168.1.199 (192.168.1.199)
> Transmission Control Protocol, Src Port: http (80), Dst Port: 50784 (50784), Seq: 152497, Ack: 289, Len: 7
> [59 Reassembled TCP Segments (152503 bytes): #12(11296), #14(2824), #31(9884), #33(2824), #35(1412), #37(14
> Hypertext Transfer Protocol
> Line-based text data: text/plain (3598 lines)
```

```
0000  86 1d de 0a fb b6 00 1e  a6 56 14 c0 08 00
0010  00 2f 65 9c 40 00 2f 06  ae 39 80 77 f5 0c
0020  01 c7 00 50 c6 60 ed ef  dc f3 5f 2a b0 f0
0030  00 ed fc b9 00 00 45 20  45 4e 44 0d 0a
```

Now let’s see what’s there inside HTTP GET and HTTP OK packets.

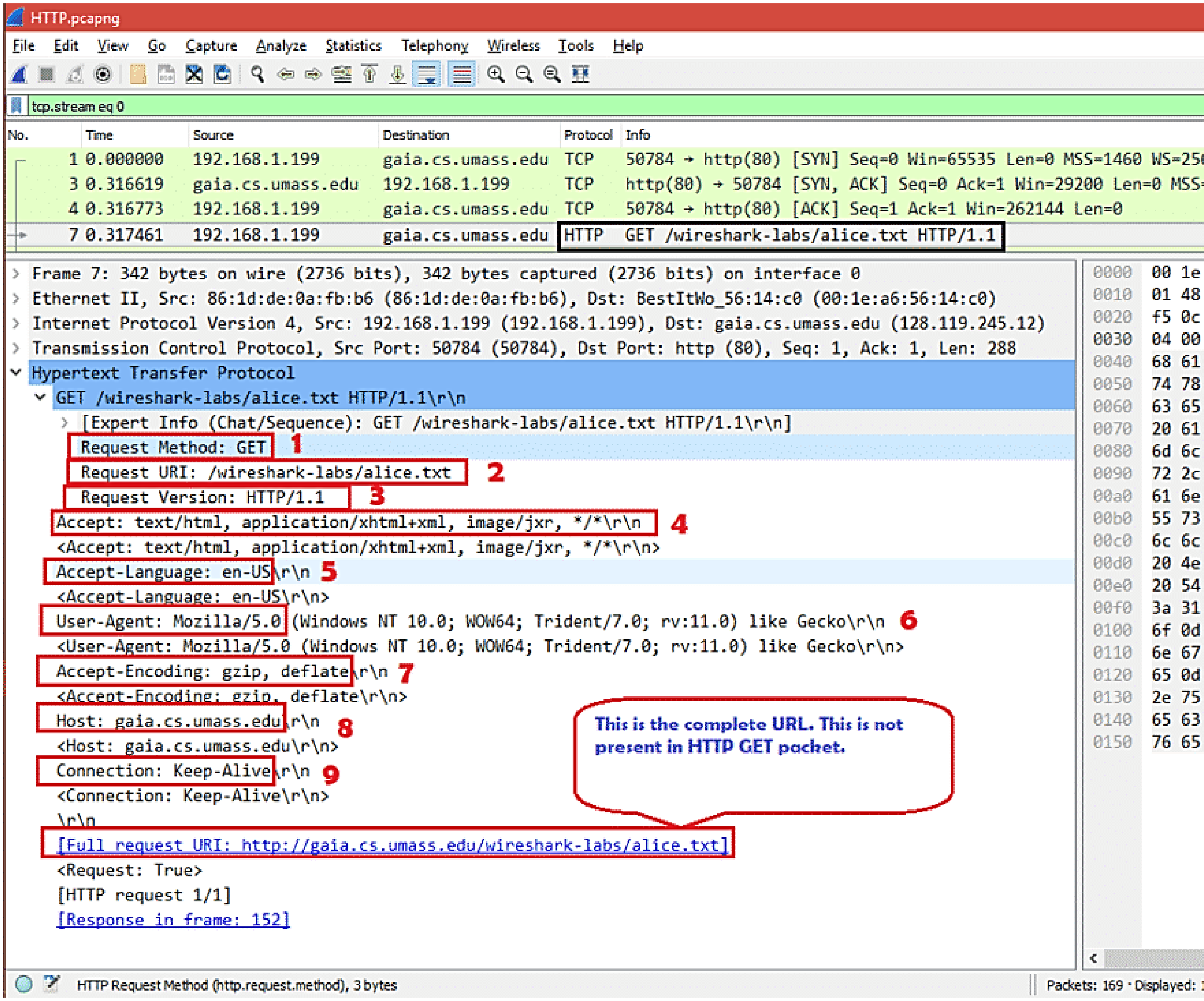
Note: We will explain TCP exchanges in another topic discussion.

HTTP GET:

After TCP 3-way handshake [SYN, SYN+ACK and ACK packets] is done HTTP GET request is sent to the server and here are the important fields in the packet.

1.Request Method: GET ==> The packet is a HTTP GET .
2.Request URI: /wireshark-labs/alice.txt ==> The client is asking for file alice.txt present under /Wireshark-labs
3.Request version: HTTP/1.1 ==> It’s HTTP version 1.1
4.Accept: text/html, application/xhtml+xml, image/jxr, */* ==> Tells server about the type of file it [client side browser] can accept. Here the client is expecting alice.txt which is text type.
5.Accept-Language: en-US ==> Accepted language standard.

6.User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko ==> Client side browser type. Even if we used internet explorer but we see it always/maximum time says Mozilla
7.Accept-Encoding: gzip, deflate ==> Accepted encoding in client side.
8.Host: gaia.cs.umass.edu ==> This is the web server name where client is sending HTTP GET request.
9.Connection: Keep-Alive ==> Connection controls whether the network connection stays open after the current transaction finishes. Connection type is keep alive.
Here is the screenshot for HTTP-GET packet fields


HTTP.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 0
1
2
3
4
5
6
7
8
9

```
No.    Time      Source             Destination        Protocol Info
     1 0.000000  192.168.1.199      gaia.cs.umass.edu  TCP      50784 → http(80) [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=25
     3 0.316619  gaia.cs.umass.edu  192.168.1.199      TCP      http(80) → 50784 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=
     4 0.316773  192.168.1.199      gaia.cs.umass.edu  TCP      50784 → http(80) [ACK] Seq=1 Ack=1 Win=262144 Len=0
     7 0.317461  192.168.1.199      gaia.cs.umass.edu  HTTP     GET /wireshark-labs/alice.txt HTTP/1.1
```

```
> Frame 7: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
> Ethernet II, Src: 86:1d:de:0a:fb:b6 (86:1d:de:0a:fb:b6), Dst: BestItWo_56:14:c0 (00:1e:a6:56:14:c0)
> Internet Protocol Version 4, Src: 192.168.1.199 (192.168.1.199), Dst: gaia.cs.umass.edu (128.119.245.12)
> Transmission Control Protocol, Src Port: 50784 (50784), Dst Port: http (80), Seq: 1, Ack: 1, Len: 288
v Hypertext Transfer Protocol
  v GET /wireshark-labs/alice.txt HTTP/1.1\r\n
    > [Expert Info (Chat/Sequence): GET /wireshark-labs/alice.txt HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /wireshark-labs/alice.txt
      Request Version: HTTP/1.1
    Accept: text/html, application/xhtml+xml, image/jxr, */*\r\n
    <Accept: text/html, application/xhtml+xml, image/jxr, */*\r\n>
    Accept-Language: en-US\r\n
    <Accept-Language: en-US\r\n>
    User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko\r\n
    <User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko\r\n>
    Accept-Encoding: gzip, deflate\r\n
    <Accept-Encoding: gzip, deflate\r\n>
    Host: gaia.cs.umass.edu\r\n
    <Host: gaia.cs.umass.edu\r\n>
    Connection: Keep-Alive\r\n
    <Connection: Keep-Alive\r\n>
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/alice.txt]
    <Request: True>
    [HTTP request 1/1]
    [Response in frame: 152]
```

```
0000  00 1e
0010  01 48
0020  f5 0c
0030  04 00
0040  68 61
0050  74 78
0060  63 65
0070  20 61
0080  6d 6c
0090  72 2c
00a0  61 6e
00b0  55 73
00c0  6c 6c
00d0  20 4e
00e0  20 54
00f0  3a 31
0100  6f 0d
0110  6e 67
0120  65 0d
0130  2e 75
0140  65 63
0150  76 65
```

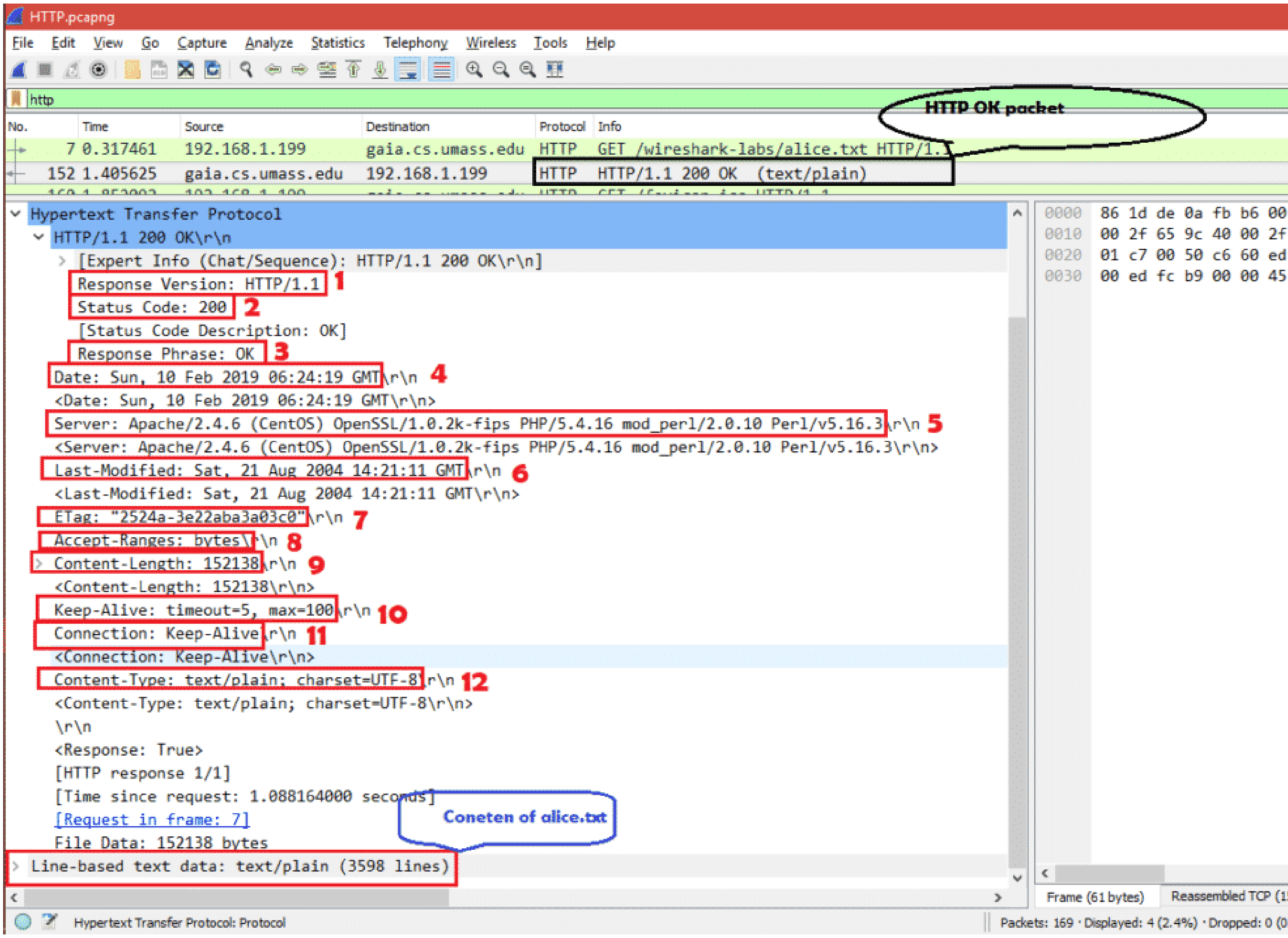
This is the complete URL. This is not present in HTTP GET packet.

```
HTTP Request Method (http.request.method), 3 bytes        Packets: 169 · Displayed:
```

HTTP OK:

After TCP data [content of alice.txt] is sent successfully HTTP OK is sent to the client and here are the important fields in the packet.
1. Response Version: HTTP/1.1 ==> Here server also in HTTP version 1.1
2.Status Code: 200 ==> Status code sent by server.
3.Response Phrase: OK ==> Response phrase sent by server.
So the from 2 and 3 we get 200 OK which means the request [HTTP GET] has succeeded.

4.Date: Sun, 10 Feb 2019 06:24:19 GMT ==> Current date , time in GMT when HTTP GET was received by server.
5.Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3 ==> Server details and configurations versions.
6.Last-Modified: **Sat, 21 Aug 2004 14:21:11 GMT** ==> Last modified date and time for the file "alice.txt".
7.ETag: "2524a-3e22aba3a03c0" ==> The ETag indicates the content is not changed to assist caching and improve performance. Or if the content has changed, etags are useful to help prevent simultaneous updates of a resource from overwriting each other.
8. Accept-Ranges: bytes ==> Byte is the unit used in server for content.
9.Content-Length: 152138 ==> This is the total length of the alice.txt in bytes.
10. Keep-Alive: timeout=5, max=100 ==> Keep alive parameters.
11.Connection: Keep-Alive ==> Connection controls whether the network connection stays open after the current transaction finishes. Connection type is keep alive.
12.Content-Type: text/plain; charset=UTF-8 ==> The content [alice.txt] type is text and charset standard is UTF-8.
Here is the screenshot for different fields of HTTP OK packet.


HTTP.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
HTTP OK packet
1
2
3
4
5
6
7
8
9
10
11
12
Coneten of alice.txt

```
No.  Time      Source             Destination        Protocol Info
   7 0.317461  192.168.1.199      gaia.cs.umass.edu  HTTP     GET /wireshark-labs/alice.txt HTTP/1.1
 152 1.405625  gaia.cs.umass.edu  192.168.1.199      HTTP     HTTP/1.1 200 OK  (text/plain)
```

```
v Hypertext Transfer Protocol
  v HTTP/1.1 200 OK\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Date: Sun, 10 Feb 2019 06:24:19 GMT\r\n
    <Date: Sun, 10 Feb 2019 06:24:19 GMT\r\n>
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3\r\n
    <Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3\r\n>
    Last-Modified: Sat, 21 Aug 2004 14:21:11 GMT\r\n
    <Last-Modified: Sat, 21 Aug 2004 14:21:11 GMT\r\n>
    ETag: "2524a-3e22aba3a03c0"\r\n
    Accept-Ranges: bytes\r\n
  > Content-Length: 152138\r\n
    <Content-Length: 152138\r\n>
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    <Connection: Keep-Alive\r\n>
    Content-Type: text/plain; charset=UTF-8\r\n
    <Content-Type: text/plain; charset=UTF-8\r\n>
    \r\n
    <Response: True>
    [HTTP response 1/1]
    [Time since request: 1.088164000 seconds]
    [Request in frame: 7]
    File Data: 152138 bytes
> Line-based text data: text/plain (3598 lines)
```

```
0000  86 1d de 0a fb b6 00
0010  00 2f 65 9c 40 00 2f
0020  01 c7 00 50 c6 60 ed
0030  00 ed fc b9 00 00 45
```

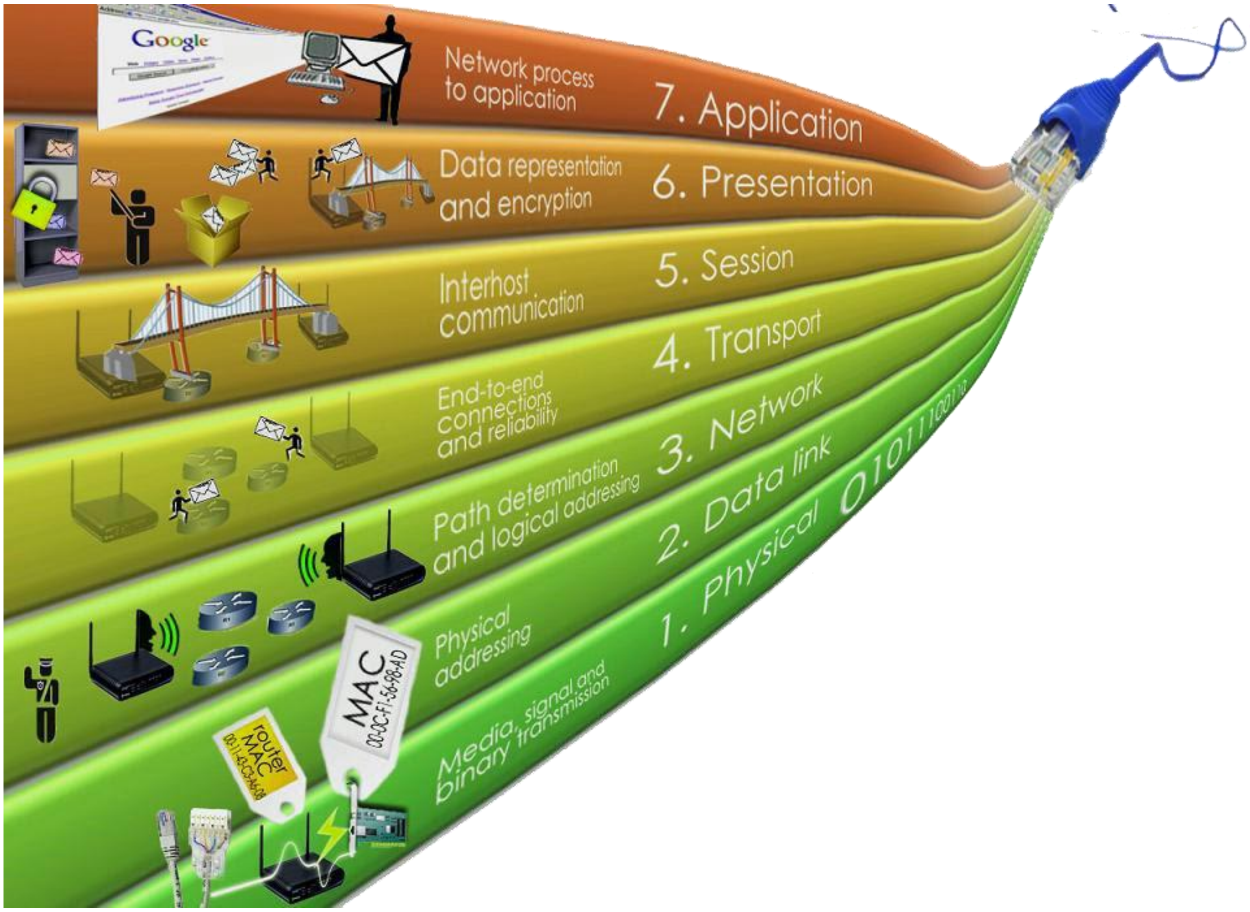
Frame (61 bytes)   Reassembled TCP (1

Hypertext Transfer Protocol: Protocol   Packets: 169 · Displayed: 4 (2.4%) · Dropped: 0 (0

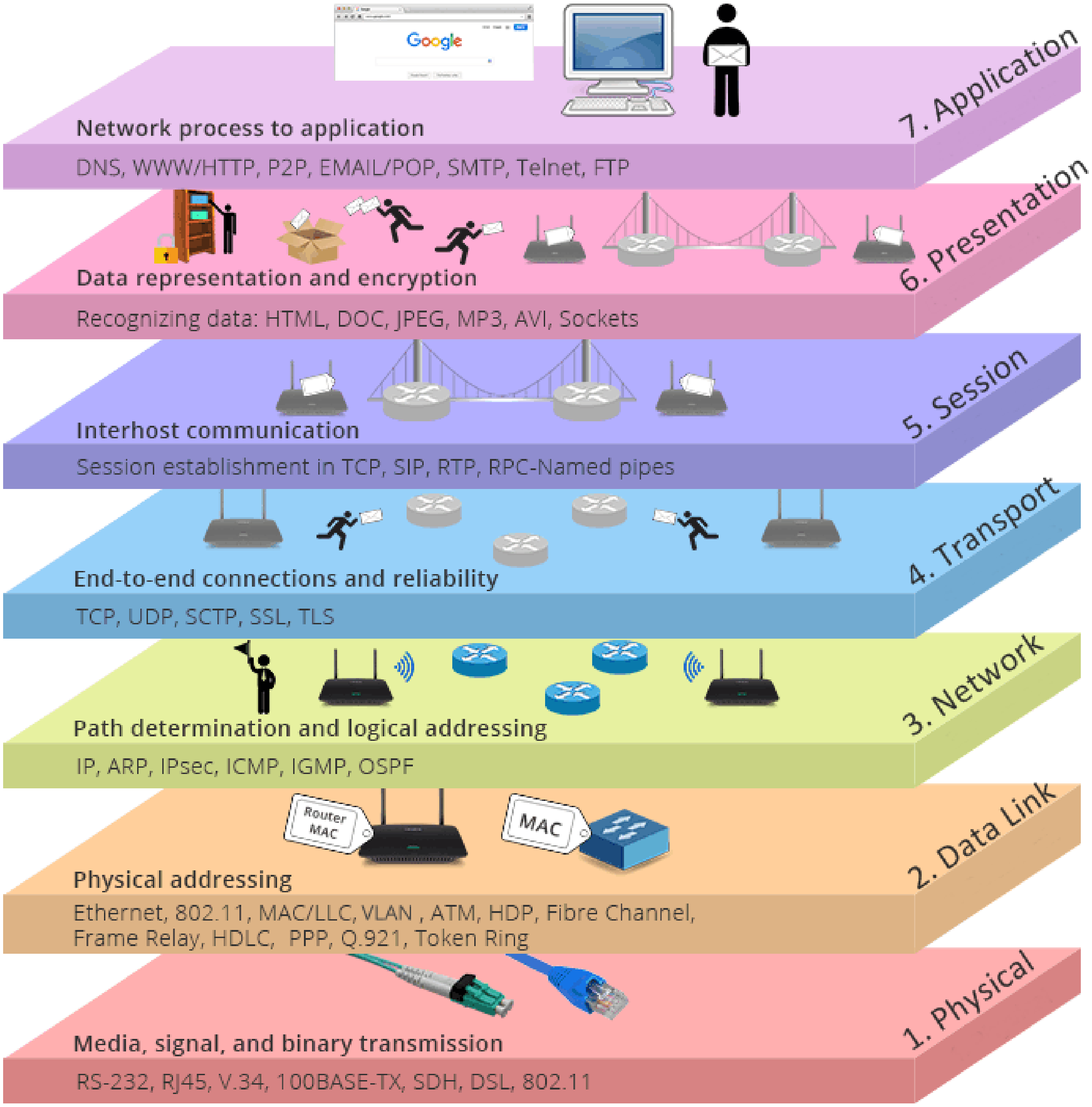
So now we know what happens when we request for any file that is present in web server.

1.1.1.7 Conclusion:

HTTP is simple application protocol that we use every day in our life. But it’s not secure so HTTPS has been implemented. That “S” stands for secure. That’s why you so maximum web server name start with http***s://[websitename]***. This means all communication between you and server are encrypted.


Google
Network process to application
7. Application
Data representation and encryption
6. Presentation
Interhost communication
5. Session
End-to-end connections and reliability
4. Transport
Path determination and logical addressing
3. Network
Physical addressing
2. Data link
Media, signal and binary transmission
1. Physical
0101110011
router MAC
MAC
00-0C-F1-56-98-AD

Prepared by: Engr. Khuram Shahzad


7. Application
Network process to application
DNS, WWW/HTTP, P2P, EMAIL/POP, SMTP, Telnet, FTP
Google
6. Presentation
Data representation and encryption
Recognizing data: HTML, DOC, JPEG, MP3, AVI, Sockets
5. Session
Interhost communication
Session establishment in TCP, SIP, RTP, RPC-Named pipes
4. Transport
End-to-end connections and reliability
TCP, UDP, SCTP, SSL, TLS
3. Network
Path determination and logical addressing
IP, ARP, IPsec, ICMP, IGMP, OSPF
2. Data Link
Router MAC
MAC
Physical addressing
Ethernet, 802.11, MAC/LLC, VLAN , ATM, HDP, Fibre Channel,
Frame Relay, HDLC, PPP, Q.921, Token Ring
1. Physical
Media, signal, and binary transmission
RS-232, RJ45, V.34, 100BASE-TX, SDH, DSL, 802.11

Prepared by: Engr. Khuram Shahzad