

CPE 400: Homework 1

Due on February 6, 2023

Igor Remizov Section 1001

Christopher Howe

Part 1

Description

Assume a node A is sending a packet to router B who is forwarding the packet to destination C. A – B link has transmission rate of 4 Megabit/second while B – C link has transmission rate of 6 Megabit/second. There is only one packet that needs to traverse from A – B – C. The packet size is 10000 Bytes.

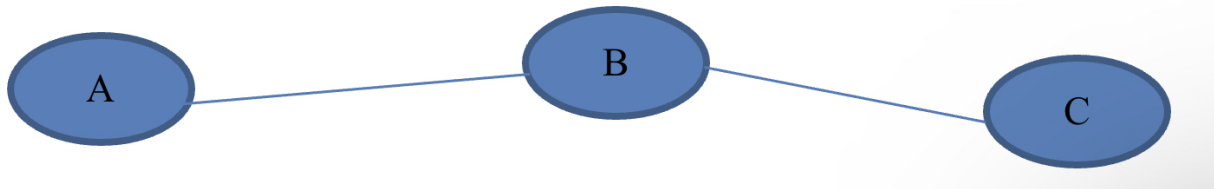


Figure 1: Part 1 Diagram

What are the transmission delays over link A – B and link B- C? Show your answer with clear explanation.

The transmission delay between two nodes is equal to L/R , where L is the size of the packet and R is the rate in bits per second. The transmission delay is how long it takes to get all the packets into the wire. The A – B link has a transmission rate of 4mbps and the packet is $10kb = 0.01mb$. The B – C link has a transmission rate of 6mbps.

$$Delay(A - B) = L/R = 0.01mb/4mbps = 0.0025s = 2.5ms$$

$$Delay(B - C) = L/R = 0.01mb/6mbps = 0.00167s = 1.67ms$$

Assume nodal processing delay at B is 5 millisecond and queuing delay at B is 2 milliseconds. Then what is the total delay from node A to node C? Assume zero propagation delay. Show your answer with clear explanation.

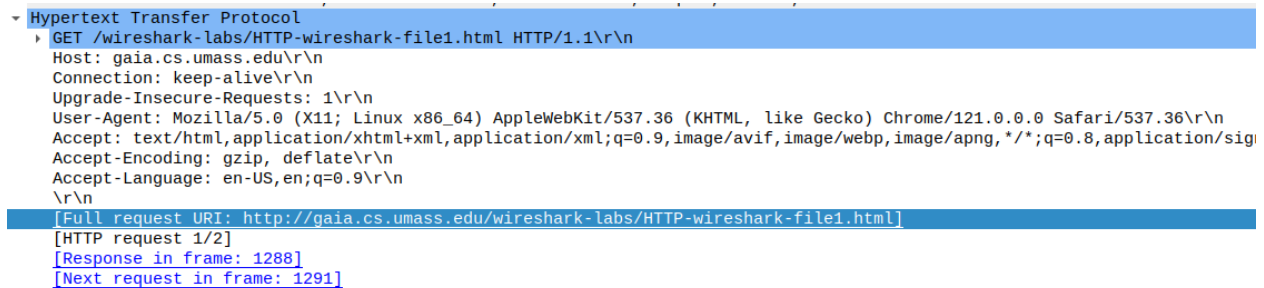
To Determine the total delay, we need to combine the Node Delay, The Noode Processing Delay, the transmission delay and the propagation delay. This includes both the transmission delay from A to B and from B to C. There is no queuing delay or processing delay at A or C so we will assume that they are the initial/final destination and that the packet is received/sent instantaneously.

$$TotalDelay = Transmission\ Delay(A-B) + Queueing\ Delay(B) + Processing\ Delay(B) + Transmission\ Delay(B-C)$$

$$TotalDelay = 2.5ms + 2ms + 5ms + 1.67ms = 11.17ms =$$

Part 2 - Wireshark

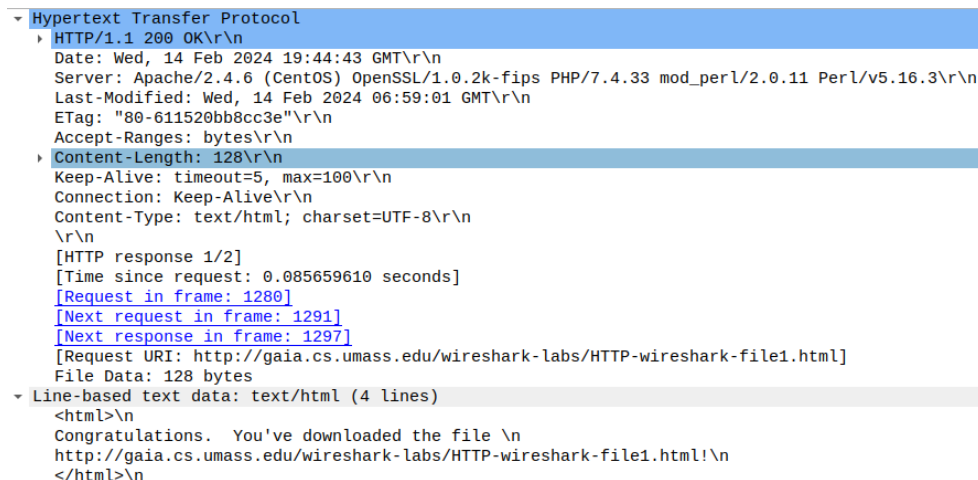
Screenshots of the packets



The screenshot shows a packet capture in Wireshark. The selected packet is a GET request from a browser to gaia.cs.umass.edu. The packet details pane shows the following information:

- Hypertext Transfer Protocol**
 - GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1\r\n
 - Host: gaia.cs.umass.edu\r\n
 - Connection: keep-alive\r\n
 - Upgrade-Insecure-Requests: 1\r\n
 - User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36\r\n
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig\r\n
 - Accept-Encoding: gzip, deflate\r\n
 - Accept-Language: en-US,en;q=0.9\r\n
 - \r\n
- [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html]
- [HTTP request 1/2]
- [Response in frame: 1288]
- [Next request in frame: 1291]

Figure 2: GET request from browser to gaia.cs.umass.edu



The screenshot shows the response packet in Wireshark. The selected packet is an HTTP 200 OK response from gaia.cs.umass.edu. The packet details pane shows the following information:

- Hypertext Transfer Protocol**
 - HTTP/1.1 200 OK\r\n
 - Date: Wed, 14 Feb 2024 19:44:43 GMT\r\n
 - Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3\r\n
 - Last-Modified: Wed, 14 Feb 2024 06:59:01 GMT\r\n
 - ETag: "80-611520bb8cc3e"\r\n
 - Accept-Ranges: bytes\r\n
 - Content-Length: 128\r\n
 - Keep-Alive: timeout=5, max=100\r\n
 - Connection: Keep-Alive\r\n
 - Content-Type: text/html; charset=UTF-8\r\n
 - \r\n
- [HTTP response 1/2]
- [Time since request: 0.085659610 seconds]
- [Request in frame: 1280]
- [Next request in frame: 1291]
- [Next response in frame: 1297]
- [Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html]
- File Data: 128 bytes

- Line-based text data: text/html (4 lines)**
 - <html>\n
 - Congratulations. You've downloaded the file \n
 - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html!\n
 - </html>\n

Figure 3: Response From gaia.cs.umass.edu

Is your browser running HTTP version 1.0 or 1.1?

My Browser uses HTTP Version 1.1. You can see this clearly in the GET/wireshark-labs/HTTP-wiresharkfile1.html HTTP/1.1.

What is the IP address of the client? Of the gaia.cs.umass.edu server?

The address of the client is 192.168.86.66. This is seen in the Internet Protocol section under the src field. The address of the server is 128.119.245.12. This can be seen in the internet protocol section under the Dst field.

What is the status code returned from the server to your browser?

When was the HTML file that you are retrieving last modified at the server?

The server returned the status code of 200 on this request. This can be seen in the HTTP Protocol section at the top. The file was last modified on 2/14/24 at 6:59 GMT which can be seen in the Last-Modified field on the response packet.

Which port is the request going to?

The request went from random port on the client/browser (port 50618) and went to port 80 on the server which is the default port for http requests.

From which port, the client is making the request?

The request went from random port on the client/browser (port 50618) and went to port 80 on the server which is the default port for http requests.

Part 3

Screenshots of the packets

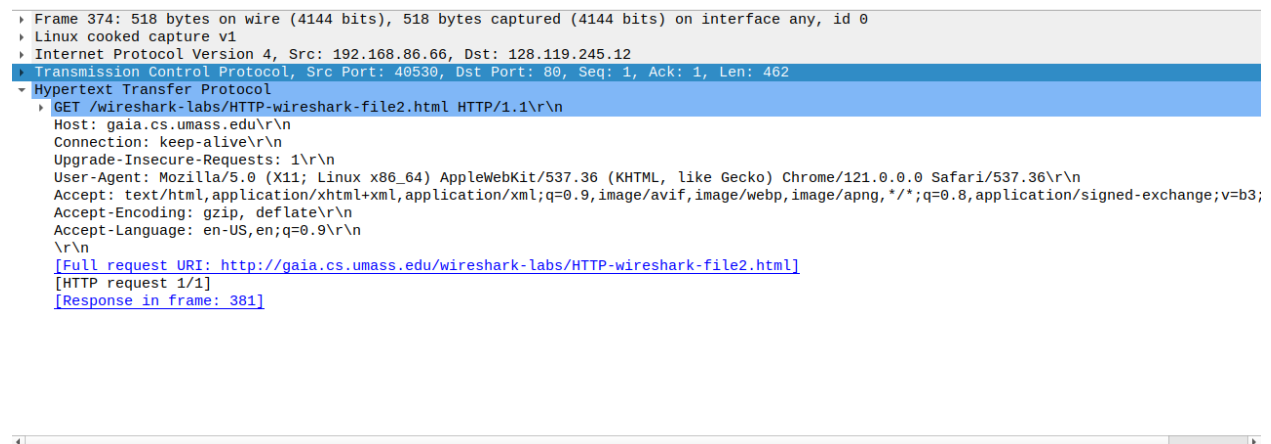


Figure 4: Initial GET request from browser to gaia.cs.umass.edu

```

> Frame 381: 786 bytes on wire (6288 bits), 786 bytes captured (6288 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.86.66
> Transmission Control Protocol, Src Port: 80, Dst Port: 49530, Seq: 1, Ack: 463, Len: 730
< Hypertext Transfer Protocol
  < HTTP/1.1 200 OK\r\n
    Date: Wed, 14 Feb 2024 20:08:50 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3\r\n
    Last-Modified: Wed, 14 Feb 2024 06:59:01 GMT\r\n
    ETag: "173-611520bb8c46e"\r\n
    Accept-Ranges: bytes\r\n
  < Content-Length: 371\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=UTF-8\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.086917755 seconds]
  [Request in frame: 374]
  [Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
  File Data: 371 bytes
< Line-based text data: text/html (10 lines)
  \n
  <html>\n
  \n
  Congratulations again! Now you've downloaded the file lab2-2.html. <br>\n
  This file's last modification date will not change. <p>\n
  Thus if you download this multiple times on your browser, a complete copy <br>\n
  will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE<br>\n
  field in your browser's HTTP GET request to the server.\n
  \n
  </html>\n

```

Figure 5: Initial response From gaia.cs.umass.edu

```

> Frame 695: 630 bytes on wire (5040 bits), 630 bytes captured (5040 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 192.168.86.66, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 49538, Dst Port: 80, Seq: 1, Ack: 1, Len: 574
< Hypertext Transfer Protocol
  < GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    Connection: keep-alive\r\n
    Cache-Control: max-age=0\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    If-None-Match: "173-611520bb8c46e"\r\n
    If-Modified-Since: Wed, 14 Feb 2024 06:59:01 GMT\r\n
  \r\n
  [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
  [HTTP request 1/17]
  [Response in frame: 700]
  [Next request in frame: 727]

```

Figure 6: Second GET request from browser to gaia.cs.umass.edu

```

> Frame 700: 296 bytes on wire (2368 bits), 296 bytes captured (2368 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.86.66
> Transmission Control Protocol, Src Port: 80, Dst Port: 49538, Seq: 1, Ack: 575, Len: 240
< Hypertext Transfer Protocol
  < HTTP/1.1 304 Not Modified\r\n
    Date: Wed, 14 Feb 2024 20:09:07 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3\r\n
    Connection: Keep-Alive\r\n
    Keep-Alive: timeout=5, max=100\r\n
    ETag: "173-611520bb8c46e"\r\n
  \r\n
  [HTTP response 1/17]
  [Time since request: 0.087570241 seconds]
  [Request in frame: 695]
  [Next request in frame: 727]
  [Next response in frame: 728]
  [Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]

```

Figure 7: Second response From gaia.cs.umass.edu

Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET? Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

On the initial request, there is no field named IF-MODIFIED-SINCE. This is because the cache is cleared and the browser has not cached the webpage. Because of this, the server returned the page.

Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header? What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Explain.

On the second request, the browser has cached the web page. When it sends the GET request, it includes a header *If - Modified - Since* : *Wed, 14 Feb 2024 06 : 59 : 01 GMT* This header denotes that the browser has a cached version of the file last modified at 6:59:01 GMT. The request asks the server if the version of this object has been updated more recently than that. The Server has not been updated since then. Instead of returning the object it returns a Response with 304 *Not Modified* This tells the browser to use the version it has cached.

Part 4

What is the IP address of the web server? Which server returned the answer? Give the IP address of this server. Is this authoritative answer or non-authoritative answer?

I used nslookup to find the address of twitter.com. After running nslookup twitter.com, I found it’s server address to be 104.244.42.129. This is the non authoritative answer. It may have been cached or from another non authoritative server. In addition to finding the ipv4 address (DNS query type A). nslookup also requested the ipv6 addresses (DNS type AAAA). The entire transaction is detailed in Figure ?? . Figure ?? shows the initial request from nslookup and Figure ?? shows the response from the router.

270	26.734515509	127.0.0.1	127.0.0.53	DNS	73 Standard query 0x36f9 A twitter.com
271	26.734656862	192.168.86.66	192.168.86.1	DNS	84 Standard query 0xe93d A twitter.com OPT
274	26.752485251	192.168.86.1	192.168.86.66	DNS	100 Standard query response 0xe93d A twitter.com A 104.244.42.129 OPT
275	26.752681470	127.0.0.53	127.0.0.1	DNS	89 Standard query response 0x36f9 A twitter.com A 104.244.42.129
276	26.753157794	127.0.0.1	127.0.0.53	DNS	73 Standard query 0xa157 AAAA twitter.com
277	26.753264280	192.168.86.66	192.168.86.1	DNS	84 Standard query 0x9e69 AAAA twitter.com OPT
279	26.772770295	192.168.86.1	192.168.86.66	DNS	141 Standard query response 0x9e69 AAAA twitter.com SOA a.u06.twtrdns.net OPT
280	26.772907342	127.0.0.53	127.0.0.1	DNS	130 Standard query response 0xa157 AAAA twitter.com SOA a.u06.twtrdns.net

Figure 8: All the DNS packets

```

> Frame 270: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
> User Datagram Protocol, Src Port: 43580, Dst Port: 53
> Domain Name System (query)
  Transaction ID: 0x36f9
  Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  Queries
    twitter.com: type A, class IN
      Name: twitter.com
      [Name Length: 11]
      [Label Count: 2]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      [Response In: 275]

```

Figure 9: Initial request from nslookup

```

> Frame 275: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.53, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 53, Dst Port: 43580
> Domain Name System (response)
  Transaction ID: 0x36f9
  Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 0
  Additional RRs: 0
  Queries
    twitter.com: type A, class IN
      Name: twitter.com
      [Name Length: 11]
      [Label Count: 2]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
  Answers
    twitter.com: type A, class IN, addr 104.244.42.129
      Name: twitter.com
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 1380 (23 minutes)
      Data length: 4
      Address: 104.244.42.129
      [Request In: 270]
      [Time: 0.018085961 seconds]

```

Figure 10: Response from router

What is/are the IP address(s) of the authoritative DNS servers for Twitter?

To get the IP of the authoritative DNS server for Twitter, I ran ‘nslookup -type=NS twitter.com’. This gave the output shown in figure ???. These list all of the authoritative DNS Servers for the twitter domain. Then, I checked that one of these were authoritative by running ‘nslookup twitter.com c.r06.twtrdns.net’ (Figure ??). Essentially, I asked c.r06.twtrdns.net for twitters IP address and since nslookup omitted the "Non-authoritative answer" line, it must be the authoritative Server. Finally, I ran ‘nslookup c.r06.twtrdns.net’ to get the Ip of that server. I found the IP of the DNS server to be 205.251.194.151 (Figure ??).

These results were confirmed by wireshark. See Figure ?? for the all the packets that resulted in this transaction. All packets sent between 127.0.0.1 and 127.0.0.53 are transactions between the loopback address and the internal DNS Server. Packets 26 and 27 represent the initial NS Query for twitters authoritative server. We will try to find the authoritative IP of the first one. Packet 28 and 29 is nslookup asking for the ip of this server (ipv4 and ipv6 versions). Packet 30 is a response from the internal dns server for that address. It determined it to be 205.251.199.195. Packets 31 and 32 show the local machine asking for the

ipv6 equivilant address. packet 33 is the response to packet 29. packets 34,38,39, and 40 are nslookup asking the authoritative server for the ip of twitter.com (ipv4 and ipv6).

26	2.418679807	127.0.0.1	127.0.0.53	DNS	73 Standard query 0x4fda NS twitter.com
27	2.418817873	127.0.0.53	127.0.0.1	DNS	220 Standard query response 0x4fda NS twitter.com NS b.r06.twtrdns.net NS
28	2.435486468	127.0.0.1	127.0.0.53	DNS	90 Standard query 0x1f8d A d.r06.twtrdns.net OPT
29	2.435495898	127.0.0.1	127.0.0.53	DNS	90 Standard query 0xe48a AAAA d.r06.twtrdns.net OPT
30	2.435572953	127.0.0.53	127.0.0.1	DNS	106 Standard query response 0x1f8d A d.r06.twtrdns.net A 205.251.199.195 C
31	2.435637214	192.168.86.66	192.168.86.1	DNS	90 Standard query 0xa1c0 AAAA d.r06.twtrdns.net OPT
32	2.454394206	192.168.86.1	192.168.86.66	DNS	158 Standard query response 0xa1c0 AAAA d.r06.twtrdns.net SOA edns101.ultr
33	2.454533157	127.0.0.53	127.0.0.1	DNS	158 Standard query response 0xe48a AAAA d.r06.twtrdns.net SOA edns101.ultr
34	2.454819922	192.168.86.66	205.251.199.195	DNS	73 Standard query 0x4de2 A twitter.com
38	2.481547656	205.251.199.195	192.168.86.66	DNS	236 Standard query response 0x4de2 A twitter.com A 104.244.42.193 NS a.r06
39	2.481780368	192.168.86.66	205.251.199.195	DNS	73 Standard query 0xac7e AAAA twitter.com
40	2.502416438	205.251.199.195	192.168.86.66	DNS	138 Standard query response 0xac7e AAAA twitter.com SOA a.u06.twtrdns.net

Figure 11: Wireshark Packets for question 2

Figure 12: Get Authoritative DNS of Twitter

Figure 13: Check Authoritative Status

Figure 14: Get Authoritative IP