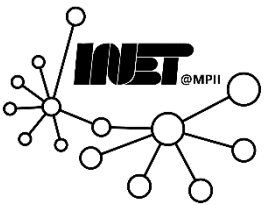
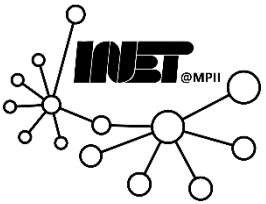
Homework 2

Application-Layer Protocols



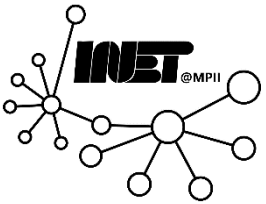
Get the Slides here

https://t.ly/Xc9N



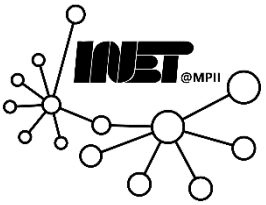
Homework Overview

- Learn about important Layer-7 protocols (HTTP & DNS)
- Learn how HTTP and DNS work in theory
- Apply the knowledge using popular tools



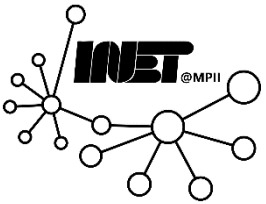
Question 1 (a)

There are two types of HTTP connections with regard to how objects are sent which are mentioned in the lecture. In one or two sentences explain each and give one advantage for each of them.



Question 1 (a)

There are two types of HTTP connections with regard to how objects are sent which are mentioned in the lecture. In one or two sentences explain each and give one advantage for each of them.



Question 1 (a)

There are two types of HTTP connections with regard to how objects are sent which are mentioned in the lecture. In one or two sentences explain each and give one advantage for each of them.

Question 1 (a)

There are two types of HTTP connections with regard to how objects are sent which are mentioned in the lecture. In one or two sentences explain each and give one advantage for each of them.

Question 1 (a)

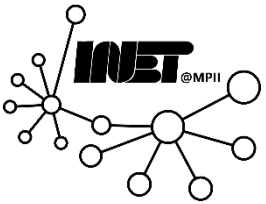
Persistent HTTP

- The connection is left open after sending a response
- When retrieving multiple objects from the same source
  - Less overhead for TCP Handshakes
  - Potentially less parallel connections in the OS

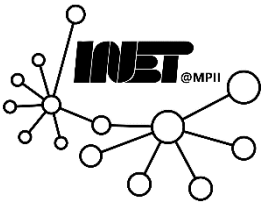
Non-persistent HTTP

- A new connection is opened for each request
- When retrieving only few objects per source
  - Resources are freed early
  - Potentially less open connections

Efficiency depends on the use case

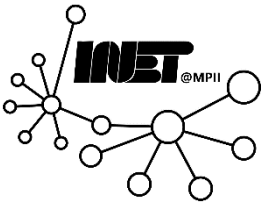


Questions?



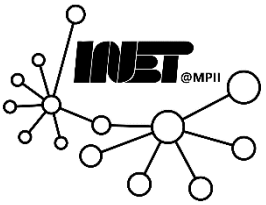
Question 1 (b)

What does it mean that HTTP is a stateless protocol? Explain a commonly used mechanism that makes it stateful.



Question 1 (b)

What does it mean that HTTP is a stateless protocol? Explain a commonly used mechanism that makes it stateful.



Question 1 (b)

What does it mean that HTTP is a stateless protocol? Explain a commonly used mechanism that makes it stateful.

- Each request is executed independently of requests that came before it
- In other words: HTTP does not remember any history of requests
- What about persistent HTTP?
- But that does not mean that there is no state in the web!

Question 1 (b)

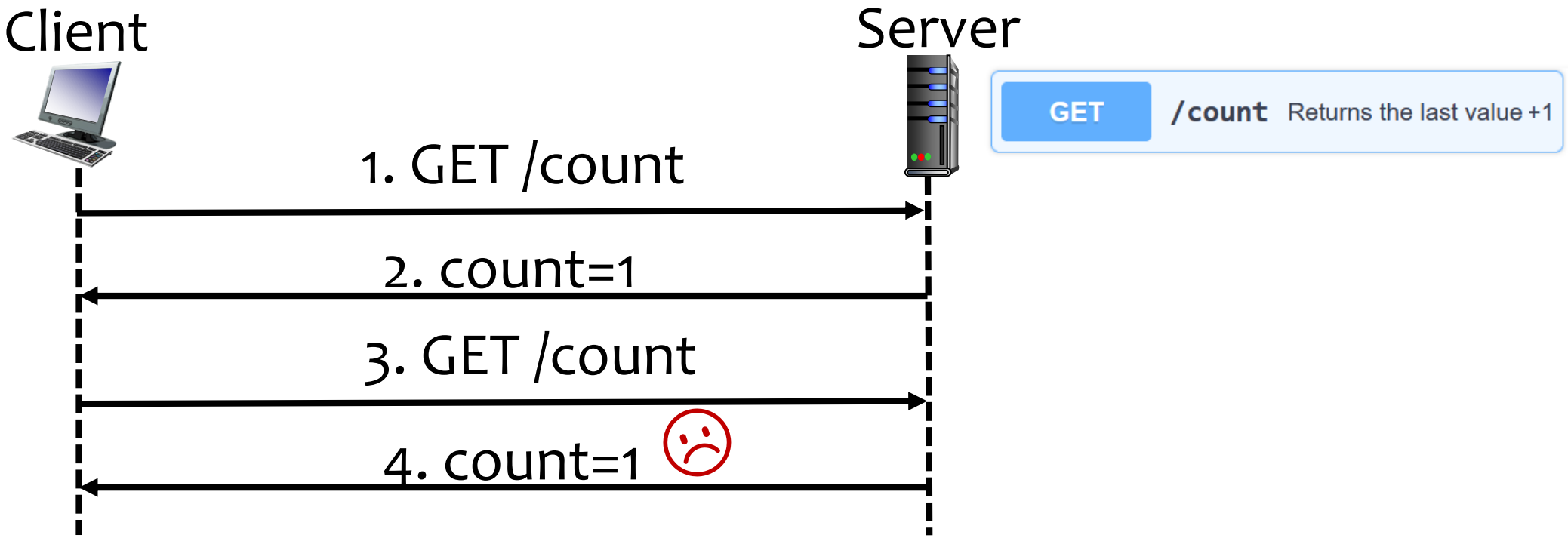
What does it mean that HTTP is a stateless protocol? Explain a commonly used mechanism that makes it stateful.

Question 1 (b)

What does it mean that HTTP is a stateless protocol? Explain a commonly used mechanism that makes it stateful.

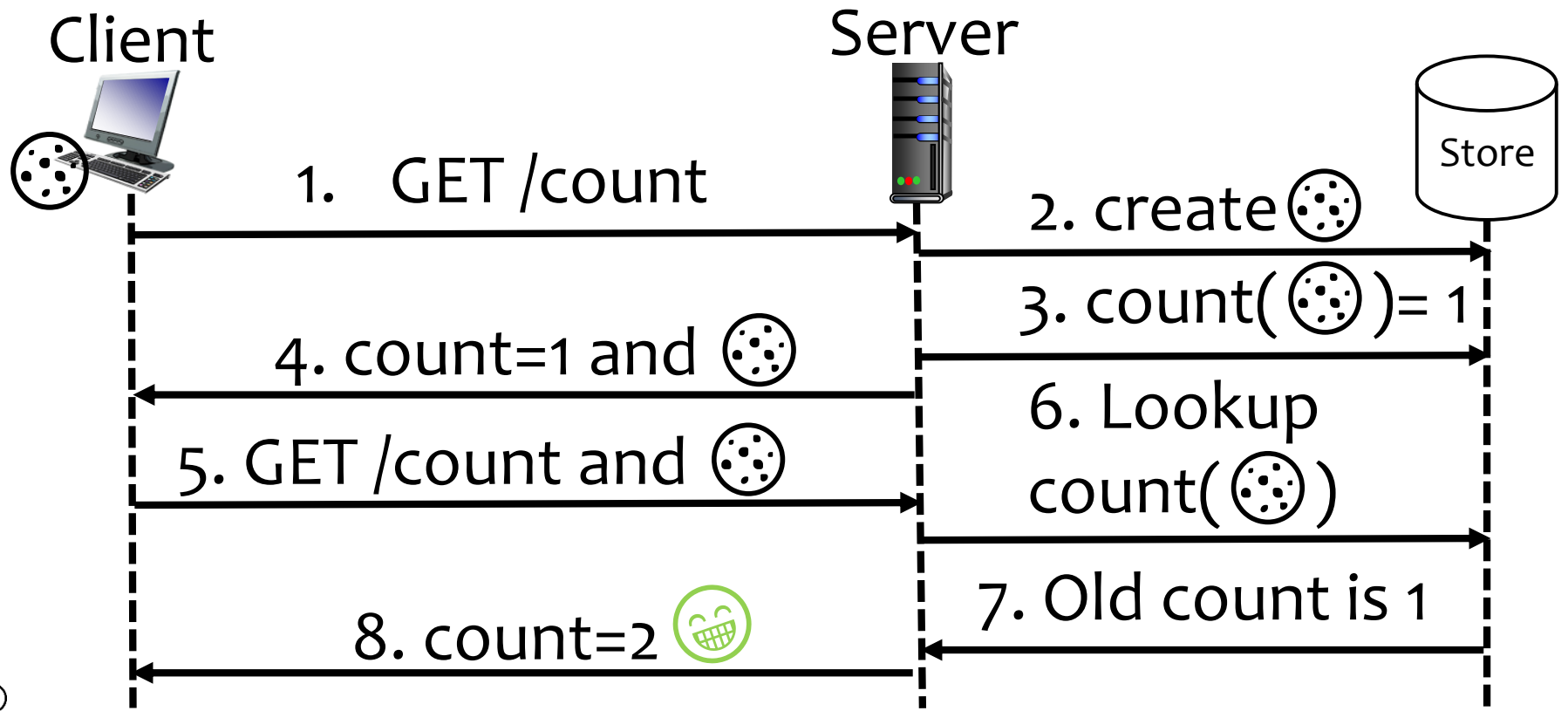
Question 1 (b)

What does it mean that HTTP is a stateless protocol? Explain a commonly used mechanism that makes it stateful.

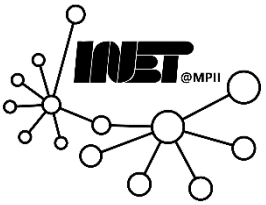


Question 1 (b)

What does it mean that HTTP is a stateless protocol? Explain a commonly used mechanism that makes it stateful.

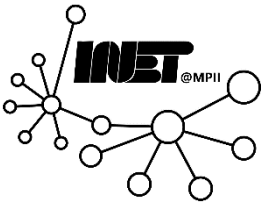


Questions?



Question 1 (c)

Describe how Web caching can reduce the delay in receiving a requested object. Mention one drawback of caching when it comes to assessing actual web servers capabilities.

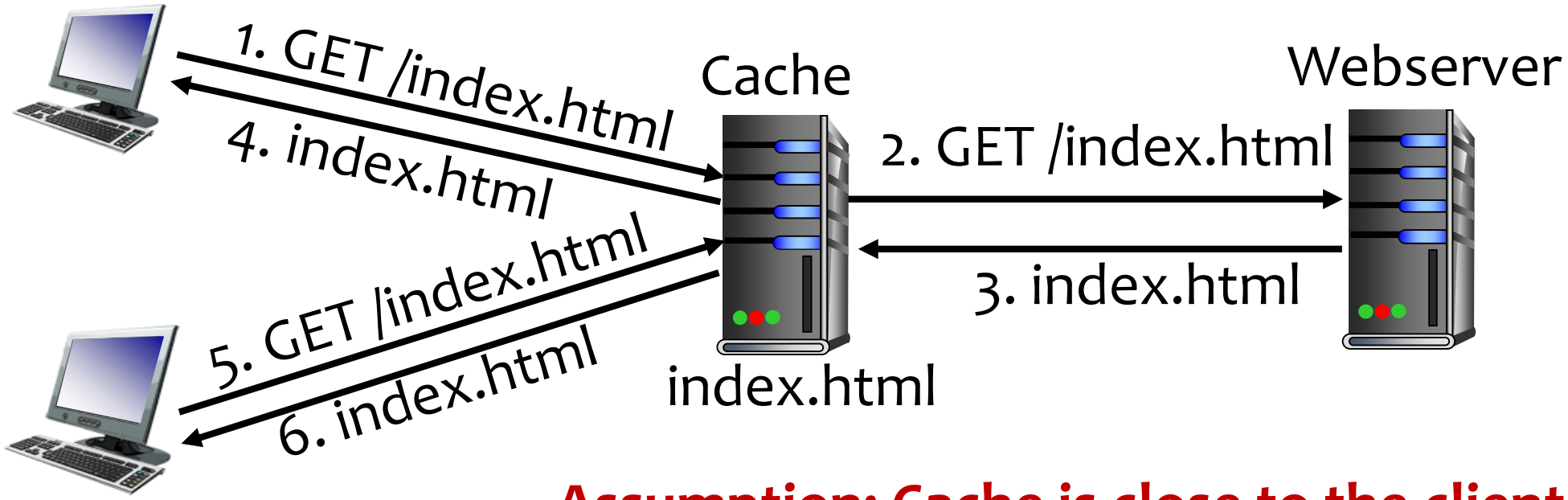


Question 1 (c)

Describe how Web caching can reduce the delay in receiving a requested object. Mention one drawback of caching when it comes to assessing actual web servers capabilities.

Question 1 (c)

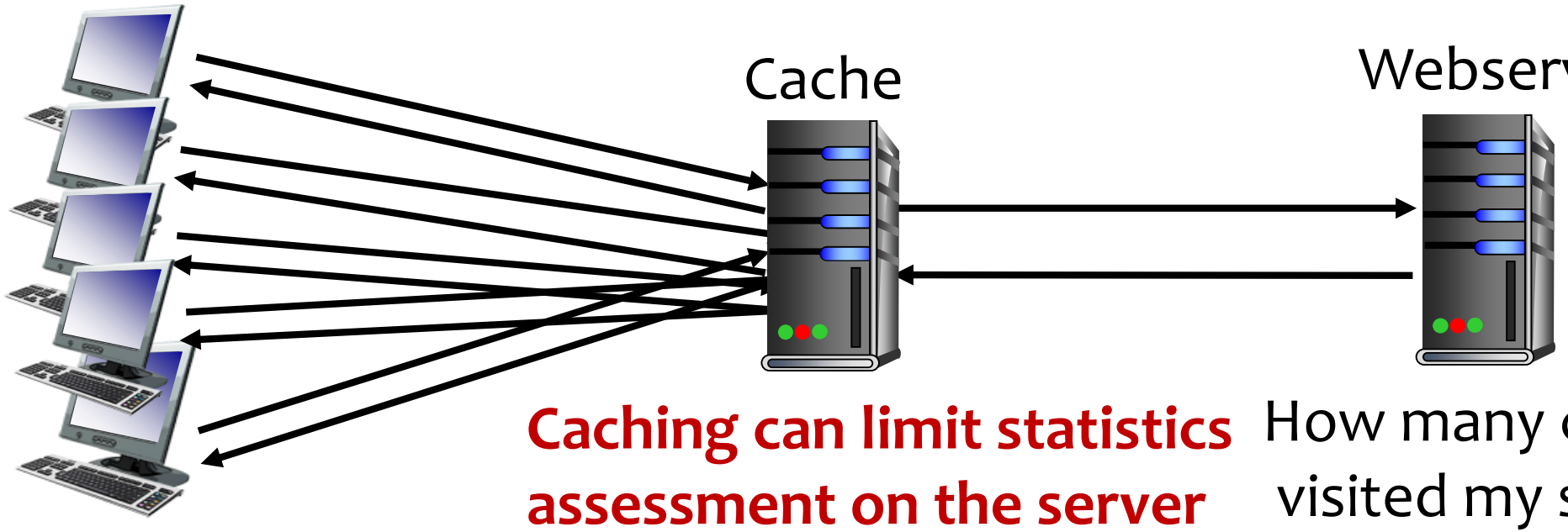
Describe how Web caching can reduce the delay in receiving a requested object. Mention one drawback of caching when it comes to assessing actual web servers capabilities.



Assumption: Cache is close to the client

Question 1 (c)

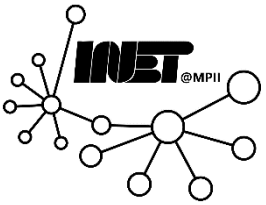
Describe how Web caching can reduce the delay in receiving a requested object. Mention one drawback of caching when it comes to assessing actual web servers capabilities.



Caching can limit statistics assessment on the server

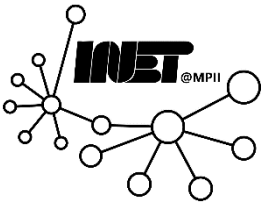
How many clients visited my site?

Questions?



Question 1 (d)

Telnet into seaborn.pydata.org and send a valid HTTP / 1.1 request message to port 80. Send the request again but this time in the request message, based on the Last-Modified: header line you got in the first attempt, include the If-modified-since: header line in a way that the response message prompts the 304 Not Modified status code. Include the messages and header responses in your solution.



Question 1 (d)

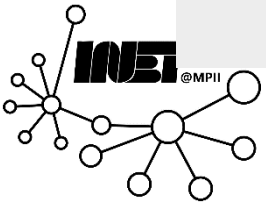
Telnet into seaborn.pydata.org and send a valid HTTP / 1.1 request message to port 80. Send the request again but this time in the request message, based on the Last-Modified: header line you got in the first attempt, include the If-modified-since: header line in a way that the response message prompts the 304 Not Modified status code. Include the messages and header responses in your solution.

Question 1 (d)

Telnet into seaborn.pydata.org and send a valid HTTP / 1.1 request message to port 80. Send the request again but this time in the request message, based on the Last-Modified: header line you got in the first attempt, include the If-modified-since: header line in a way that the response message prompts the 304 Not Modified status code. Include the messages and header responses in your solution.

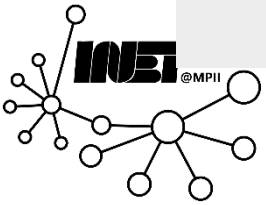
Question 1 (d)

```
1. ~$ telnet seaborn.pydata.org 80
2.  Trying 185.199.111.153...
3.  Connected to seaborn.github.io.
4.  Escape character is '^]'.
5. ~$ GET / HTTP/1.1
6. ~$ Host: seaborn.pydata.org
7.  HTTP/1.1 200 OK
8.  Connection: keep-alive
9.  Last-Modified: Fri, 30 Dec 2022 20:00:20 GMT
10. <and lots of HTML>
```



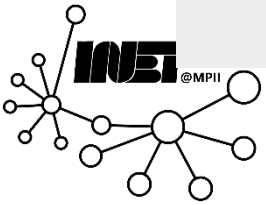
Question 1 (d)

```
1. ~$ telnet seaborn.pydata.org 80
2.  Trying 185.199.111.153...
3.  Connected to seaborn.github.io.
4.  Escape character is '^]'.
5. ~$ GET / HTTP/1.1
6. ~$ Host: seaborn.pydata.org
7.  HTTP/1.1 200 OK
8.  Connection: keep-alive
9.  Last-Modified: Fri, 30 Dec 2022 20:00:20 GMT
10. <and lots of HTML>
```



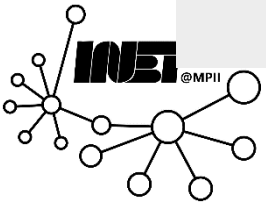
Question 1 (d)

```
1. ~$ telnet seaborn.pydata.org 80
2.  Trying 185.199.111.153...
3.  Connected to seaborn.github.io.
4.  Escape character is '^]'.
5. ~$ GET / HTTP/1.1
6. ~$ Host: seaborn.pydata.org
7.  HTTP/1.1 200 OK
8.  Connection: keep-alive
9.  Last-Modified: Fri, 30 Dec 2022 20:00:20 GMT
10. <and lots of HTML>
```



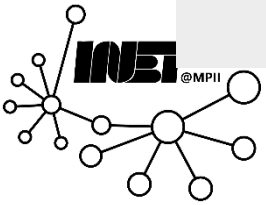
Question 1 (d)

```
1. ~$ telnet seaborn.pydata.org 80
2.  Trying 185.199.111.153...
3.  Connected to seaborn.github.io.
4.  Escape character is '^]'.
5. ~$ GET / HTTP/1.1
6. ~$ Host: seaborn.pydata.org
7.  HTTP/1.1 200 OK
8.  Connection: keep-alive
9.  Last-Modified: Fri, 30 Dec 2022 20:00:20 GMT
10. <and lots of HTML>
```



Question 1 (d)

```
1. ~$ telnet seaborn.pydata.org 80
2.  Trying 185.199.111.153...
3.  Connected to seaborn.github.io.
4.  Escape character is '^]'.
5. ~$ GET / HTTP/1.1
6. ~$ Host: seaborn.pydata.org
7.  HTTP/1.1 200 OK
8.  Connection: keep-alive
9.  Last-Modified: Fri, 30 Dec 2022 20:00:20 GMT
10. <and lots of HTML>
```

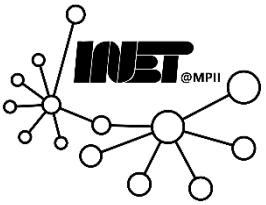


Question 1 (d)

Telnet into seaborn.pydata.org and send a valid HTTP / 1.1 request message to port 80. Send the request again but this time in the request message, based on the Last-Modified: header line you got in the first attempt, include the If-modified-since: header line in a way that the response message prompts the 304 Not Modified status code. Include the messages and header responses in your solution.

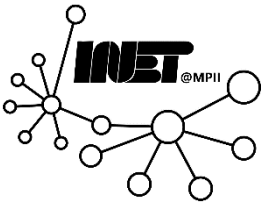
Question 1 (d)

Telnet into seaborn.pydata.org and send a valid HTTP / 1.1 request message to port 80. Send the request again but this time in the request message, based on the Last-Modified: header line you got in the first attempt, include the If-modified-since: header line in a way that the response message prompts the 304 Not Modified status code. Include the messages and header responses in your solution.



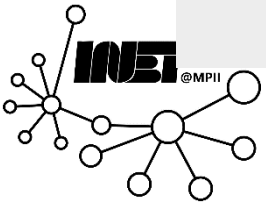
Question 1 (d)

Telnet into seaborn.pydata.org and send a valid HTTP / 1.1 request message to port 80. Send the request again but this time in the request message, based on the Last-Modified: header line you got in the first attempt, include the If-modified-since: header line in a way that the response message prompts the 304 Not Modified status code. Include the messages and header responses in your solution.



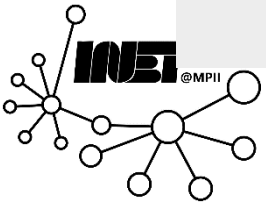
Question 1 (d)

```
1. ~$ telnet seaborn.pydata.org 80
2.  Trying 185.199.111.153...
3.  Connected to seaborn.github.io.
4.  Escape character is '^]'.
5. ~$ GET / HTTP/1.1
6. ~$ Host: seaborn.pydata.org
7. ~$ If-modified-since: Fri, 30 Dec 2022 20:00:20 GMT
8.  HTTP/1.1 304 Not Modified
9.  Connection: keep-alive
10. <and NO HTML>
```



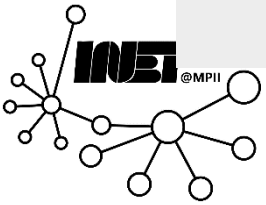
Question 1 (d)

```
1. ~$ telnet seaborn.pydata.org 80
2.  Trying 185.199.111.153...
3.  Connected to seaborn.github.io.
4.  Escape character is '^]'.
5. ~$ GET / HTTP/1.1
6. ~$ Host: seaborn.pydata.org
7. ~$ If-modified-since: Fri, 30 Dec 2022 20:00:20 GMT
8.  HTTP/1.1 304 Not Modified
9.  Connection: keep-alive
10. <and NO HTML>
```

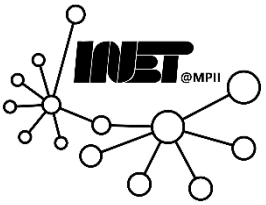


Question 1 (d)

```
1. ~$ telnet seaborn.pydata.org 80
2.  Trying 185.199.111.153...
3.  Connected to seaborn.github.io.
4.  Escape character is '^]'.
5. ~$ GET / HTTP/1.1
6. ~$ Host: seaborn.pydata.org
7. ~$ If-modified-since: Fri, 30 Dec 2022 20:00:20 GMT
8.  HTTP/1.1 304 Not Modified
9.  Connection: keep-alive
10. <and NO HTML>
```

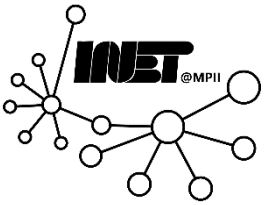


Questions?



Question 2 (a)

Compare authoritative DNS nameservers with DNS resolvers. Explain the role of the three different types of DNS resolvers (one or two sentences for each)



Question 2 (a)

Compare authoritative DNS nameservers with DNS resolvers. Explain the role of the three different types of DNS resolvers (one or two sentences for each)

Question 2 (a)

Authoritative Servers

- Answer queries authoritatively
- Responds to iterative queries issues by resolvers
- Part of the DNS hierarchy

Resolvers

- Can make use of caching
- Respond to recursive requests from clients
- Not strictly part of the DNS hierarchy

Question 2 (a)

Compare authoritative DNS nameservers with DNS resolvers. Explain the role of the three different types of DNS resolvers (one or two sentences for each)

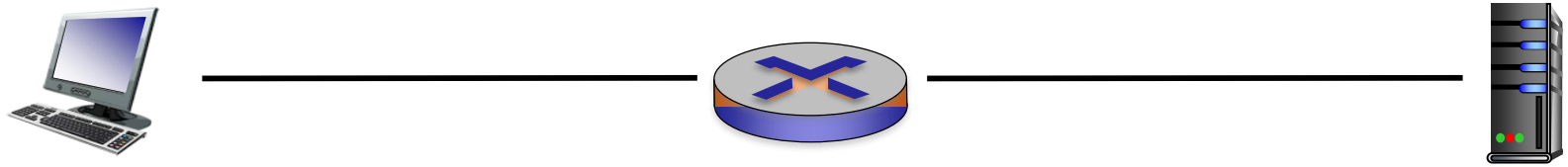
Question 2 (a)

Compare authoritative DNS nameservers with DNS resolvers. Explain the role of the three different types of DNS resolvers (one or two sentences for each)

Question 2 (a)

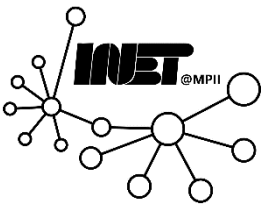
Compare authoritative DNS nameservers with DNS resolvers. Explain the role of the three different types of DNS resolvers (one or two sentences for each)

Question 2 (a)



Stub Resolver	Forwarding Resolver	Recursive Resolver
- Running on local machine - Sends query to configured resolver - Returns answers to querying software	- Acts as a proxy - Forwards queries to recursive resolver - Returns answer to querying resolvers	- Does the actual resolution - Queries Root, TLD, SLD… until it has an answer - Returns the answer to the querying resolvers

Questions?

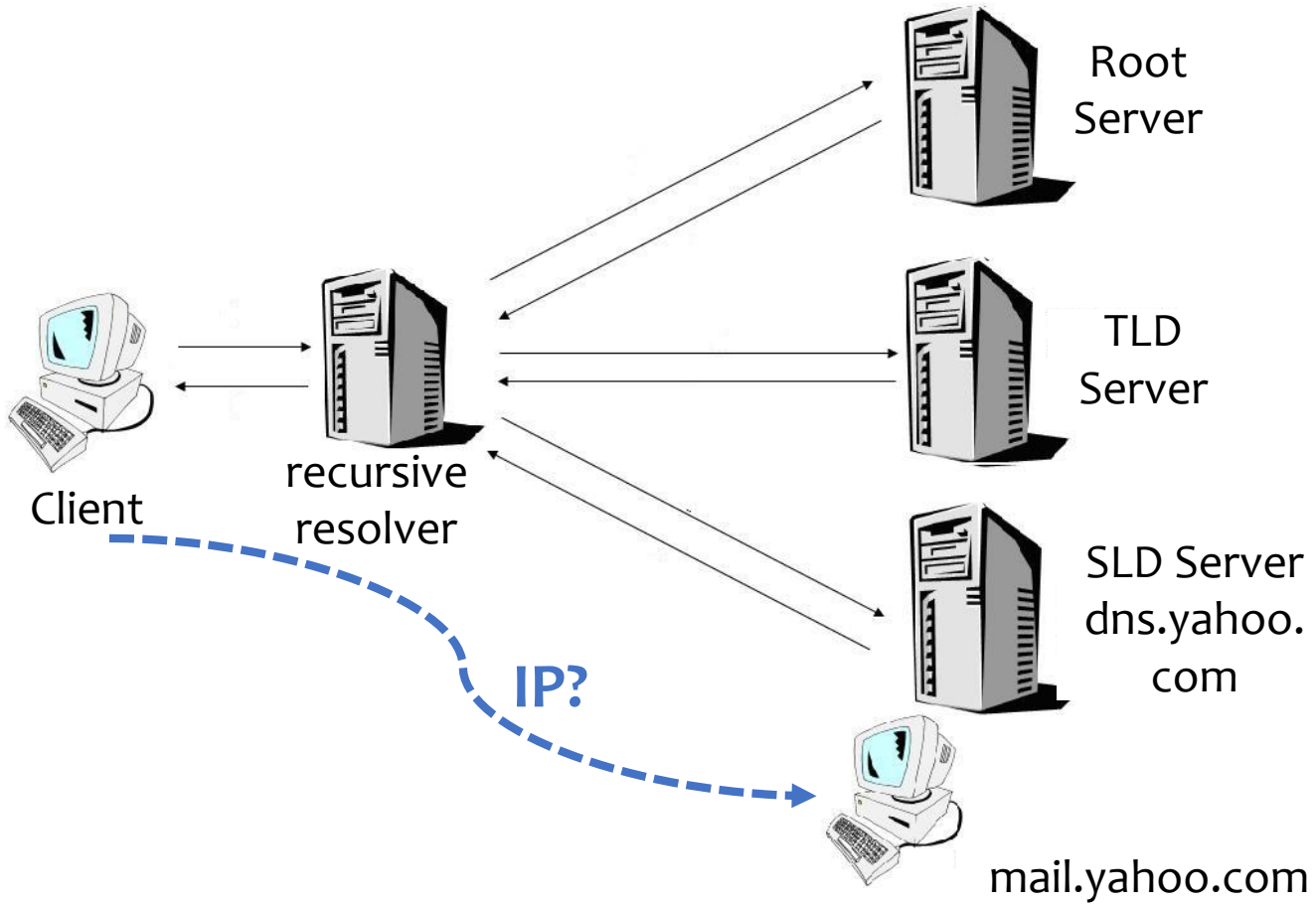


Question 2 (b)

Consider a scenario in which the client wants to get the IP address of the mail.yahoo.com. Enumerate the lines in the figure below in order of how the request would be solved. Then explain each step and mention what information is sent in each. Generally there are two types of DNS requests, namely recursive and iterative; specify type of each request in this scenario.

Question 2 (b)

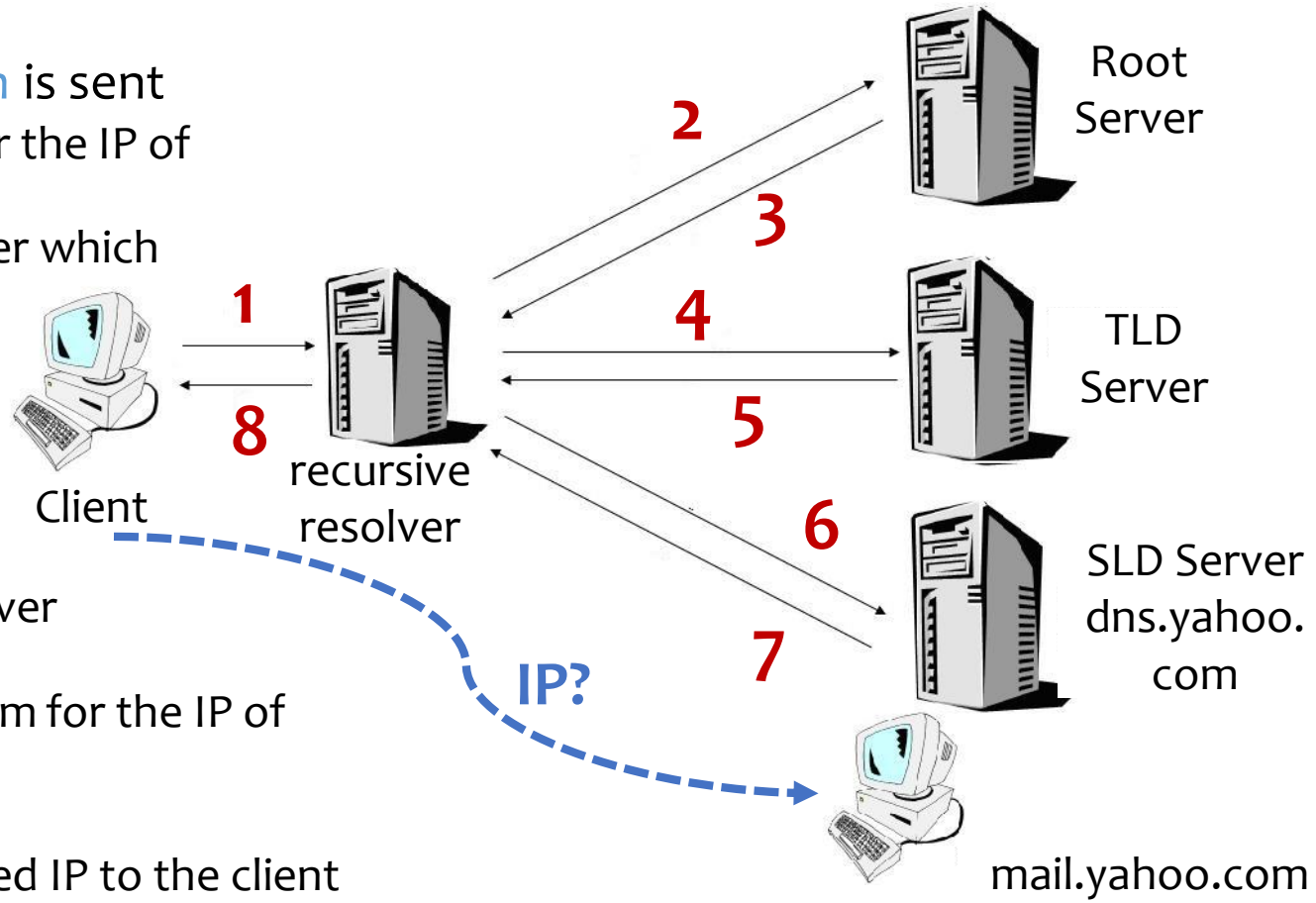
- Consider a scenario in which the client wants to get the IP address of the mail.yahoo.com
- Enumerate the lines in the figure below in order of how the request would be solved
- Explain each step and what information is sent
- Specify the type of each request



Question 2 (b)

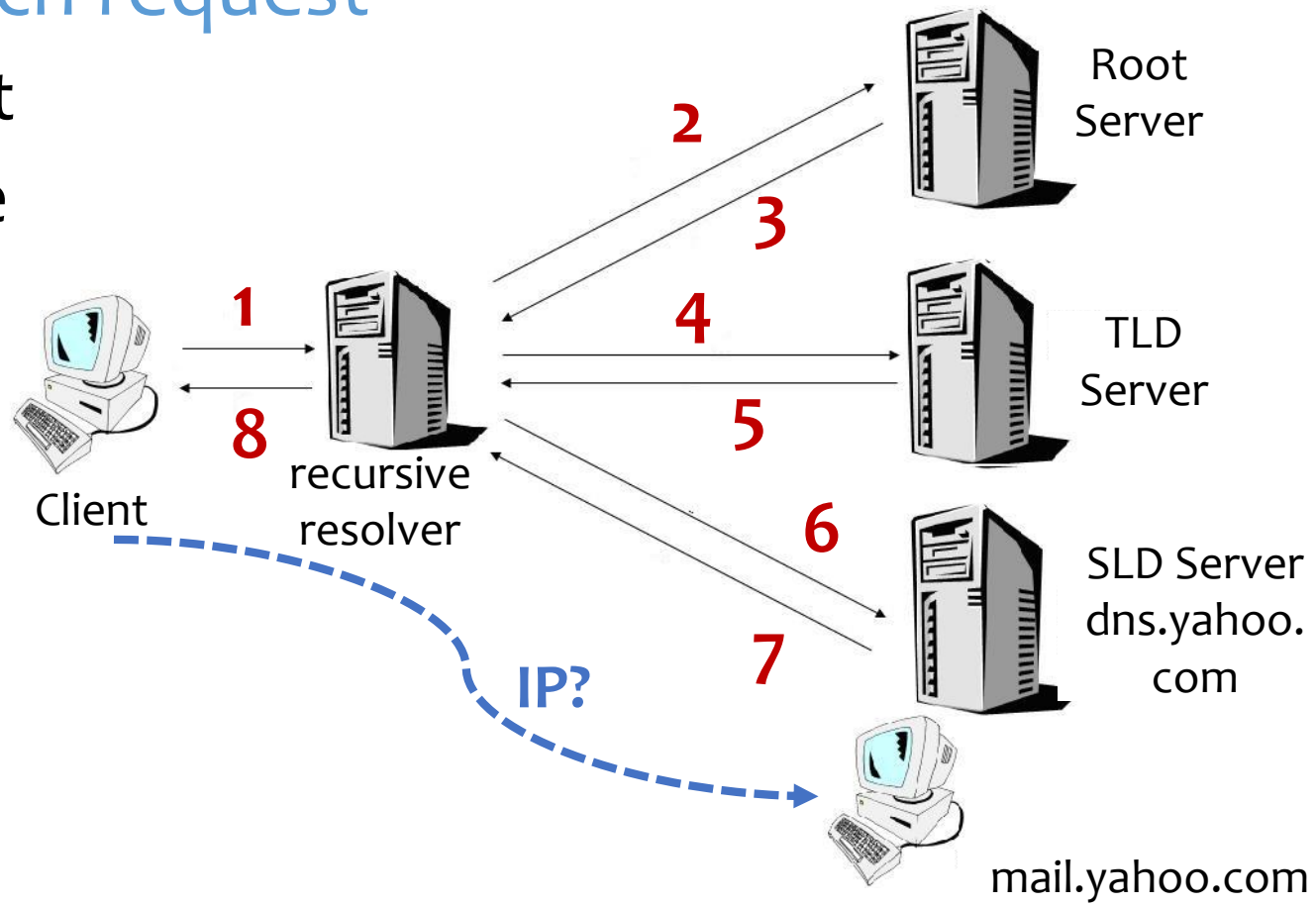
- Enumerate the lines in the figure below in order of how the request would be solved
- Explain each step what information is sent

1. Client asks the local resolver for the IP of mail.yahoo.com
2. The resolver asks the root server which NS to ask for .com
3. The root server refers to the resolver the TLD Server
4. The resolver asks the TLD Server which NS to ask for yahoo.com
5. The TLD Server refers the resolver to dns.yahoo.com
6. The resolver asks dns.yahoo.com for the IP of mail.yahoo.com
7. It responds with the IP
8. The resolver returns the resolved IP to the client



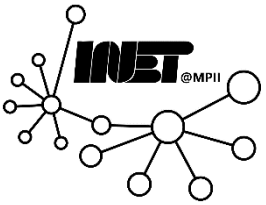
Question 2 (b)

- Specify the type of each request
  - 1 is a recursive request
  - 2,4, and 6 are iterative



Question 2 (c)

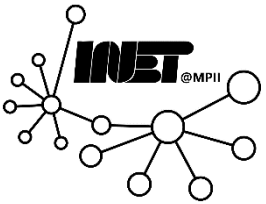
DNS is typically using an unreliable transport (UDP) instead of a reliable one (TCP). So if a DNS packet is lost in the network, there is no automatic recovery provided by UDP. How would such a packet loss be handled by a client accessing a website?



Question 2 (c)

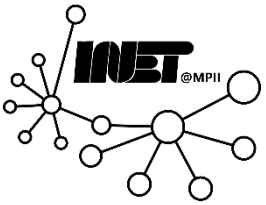
DNS is typically using an unreliable transport (UDP) instead of a reliable one (TCP). So if a DNS packet is lost in the network, there is no automatic recovery provided by UDP. How would such a packet loss be handled by a client accessing a website?

- There is a timeout on the client side, after which the client will resend the request
- Clients may try a different server for resending



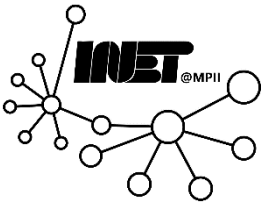
Question 3 (a)

Describe the purpose of the following DNS record types: A, AAAA, CNAME, NS and MX in one or two sentences each.



Question 3 (a)

Describe the purpose of the following DNS record types: A, AAAA, CNAME, NS and MX in one or two sentences each.



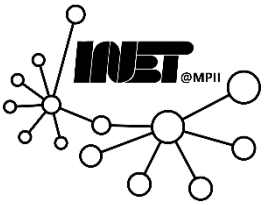
Question 3 (a)

Describe the purpose of the following DNS record types: A, AAAA, CNAME, NS and MX in one or two sentences each.

Question 3 (a)

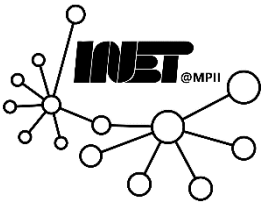
Describe the purpose of the following DNS record types: A, AAAA, CNAME, NS and MX in one or two sentences each.

- A — Map a hostname to an IPv4 Address
- AAAA — Maps a hostname to an IPv6 Address
- CNAME — Alias or nickname for a domain, allowing access via different name
- NS — Maps which Nameservers are responsible for which set of domains ('zone')
- MX — Maps a domain name to the hostname of a mail-server that is associated with it



Question 3 (b)

Send three DNS queries for domains of your choice with different record types (A, AAAA, NS, CNAME, and MX) Explore the responses and explain your findings. (You can use dig and nslookup commands to send a DNS query in Linux and Windows respectively.)



Question 3 (b)

Send three DNS queries for domains of your choice with different record types (A, AAAA, NS, CNAME, and MX) Explore the responses and explain your findings. (You can use dig and nslookup commands to send a DNS query in Linux and Windows respectively.)

Question 3 (b)

Send three DNS queries for domains of your choice with different record types (A, AAAA, NS, CNAME, and MX) Explore the responses and explain your findings. (You can use dig and nslookup commands to send a DNS query in Linux and Windows respectively.)

```
1  ~$ dig mpi-inf.mpg.de A +nocmd
2  ;; Got answer:
3  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
   49265
4  ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
   ADDITIONAL: 1

5  ;; QUESTION SECTION:
6  ;mpi-inf.mpg.de.      IN      A

7  ;; ANSWER SECTION:
8  mpi-inf.mpg.de.  355    IN      A      139.19.86.161

9  ;; Query time: 1 msec
10 ;; SERVER: 139.19.66.1#53(139.19.66.1) (UDP)
11 ;; WHEN: Wed Apr 12 12:21:39 CEST 2023
12 ;; MSG SIZE  rcvd: 87
```

The command

We got an answer!

Header of the DNS response message

Begin of the question section

The question we asked (repeated)
Domain, Class, Type

Begin of the answer section

The response to our question
(Domain, TTL, Class, Type, IP)

How long the query took

The responding server

When the query was issued

Size (in bytes) of the response

Question 3 (b)

What about this section?

```
1.   ;; OPT PSEUDOSECTION:
2.   ; EDNS: version: 0, flags:; udp: 1232
3.   ; COOKIE: 5aae44d16a795049010000006436863306eb25ee3c04a62c (good)
```

- Extension mechanism for DNS
- Not important to understand the principal workings ☺

Question 3 (b)

1	`~$ nslookup -type=MX www.uni-saarland.de`	The command
2	`Server:  nsintern2.mpi-klsb.mpg.de`	The responding server name
3	`Address:  139.19.66.1`	Responding server address
4	`Non-authoritative answer:`	The server answered non-authoritatively. Must have been a cached answer by a resolver
5	`www.uni-saarland.de canonical name = webuni.rz.uni-saarland.de`	The domain was a CNAME!
6	`webuni.rz.uni-saarland.de MX preference = 10, mail exchanger = m-relay.rz.uni-saarland.de`	But the server provided the MX records we looked for.
7	`webuni.rz.uni-saarland.de MX preference = 20, mail exchanger = m-relay2.rz.uni-saarland.de`	Multiple MX entries with different preference exist.

Question 3 (c)

Execute DNS queries for stackoverflow.com a couple of times for the A record with a few minutes intervals between them. What is the difference between the responses in the answer sections? What is the reason for the changes?

Question 3 (c)

Execute DNS queries for stackoverflow.com a couple of times for the A record with a few minutes intervals between them. What is the difference between the responses in the answer sections? What is the reason for the changes?

```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status:
NOERROR, id: 38990




;; ANSWER SECTION:

stackoverflow.com. 300 IN A
151.101.193.69

stackoverflow.com. 300 IN A
151.101.129.69

stackoverflow.com. 300 IN A
151.101.1.69

stackoverflow.com. 300 IN A
151.101.65.69



;; Query time: 31 msec
```

```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38990

;; ANSWER SECTION:
stackoverflow.com. 300 IN A 151.101.193.69
stackoverflow.com. 300 IN A 151.101.129.69
stackoverflow.com. 300 IN A 151.101.1.69
stackoverflow.com. 300 IN A 151.101.65.69

;; Query time: 31 msec
```

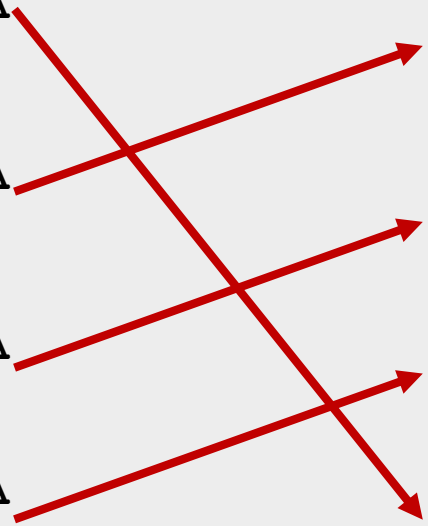
```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17617

;; ANSWER SECTION:
stackoverflow.com. 294 IN A 151.101.129.69
stackoverflow.com. 294 IN A 151.101.1.69
stackoverflow.com. 294 IN A 151.101.65.69
stackoverflow.com. 294 IN A 151.101.193.69

;; Query time: 2 msec
```

```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38990

;; ANSWER SECTION:
stackoverflow.com. 300 IN A 151.101.193.69
stackoverflow.com. 300 IN A 151.101.129.69
stackoverflow.com. 300 IN A 151.101.1.69
stackoverflow.com. 300 IN A 151.101.65.69

;; Query time: 31 msec
```

```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17617

;; ANSWER SECTION:
stackoverflow.com. 294 IN A 151.101.129.69
stackoverflow.com. 294 IN A 151.101.1.69
stackoverflow.com. 294 IN A 151.101.65.69
stackoverflow.com. 294 IN A 151.101.193.69

;; Query time: 2 msec
```

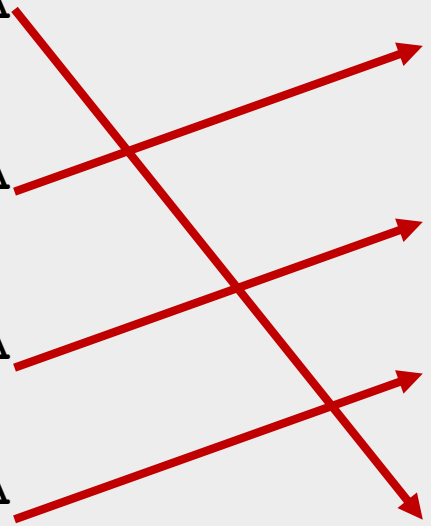
```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38990

;; ANSWER SECTION:
stackoverflow.com. 300 IN A 151.101.193.69
stackoverflow.com. 300 IN A 151.101.129.69
stackoverflow.com. 300 IN A 151.101.1.69
stackoverflow.com. 300 IN A 151.101.65.69

;; Query time: 31 msec
```

```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17617

;; ANSWER SECTION:
stackoverflow.com. 294 IN A 151.101.129.69
stackoverflow.com. 294 IN A 151.101.1.69
stackoverflow.com. 294 IN A 151.101.65.69
stackoverflow.com. 294 IN A 151.101.193.69

;; Query time: 2 msec
```



```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38990

;; ANSWER SECTION:
stackoverflow.com. 300 IN A 151.101.193.69
stackoverflow.com. 300 IN A 151.101.129.69
stackoverflow.com. 300 IN A 151.101.1.69
stackoverflow.com. 300 IN A 151.101.65.69

;; Query time: 31 msec
```

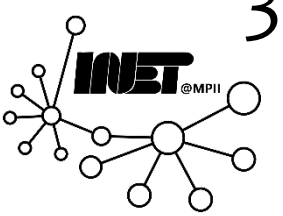
```
~$ dig stackoverflow.com A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17617

;; ANSWER SECTION:
stackoverflow.com. 294 IN A 151.101.129.69
stackoverflow.com. 294 IN A 151.101.1.69
stackoverflow.com. 294 IN A 151.101.65.69
stackoverflow.com. 294 IN A 151.101.193.69

;; Query time: 2 msec
```



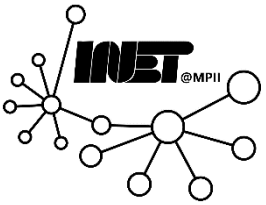
Question 3 (c)

Execute DNS queries for stackoverflow.com a couple of times for the A record with a few minutes intervals between them. What is the difference between the responses in the answer sections? What is the reason for the changes?

- ID: A random ID is picked for every request
- TTL: The time-to-live of the entries is reduced
- Order: Order of entries is not specified. It can be changed, for example, for load balancing purposes
- Query Time: The first request was not in the cache, it took 31ms. Subsequent requests were much quicker



Questions?



Feedback?