Homework 6

Problem 6.1

In theory the domain name should go first to recursive resolver and then the later should query the root server, and from there start querying the authorative servers until we get to resolve the lowest level of address. We query the following command:
dig AAAA @8.8.8.8 +trace grader.eecs.jacobs-university.de.
AAAA - specifies that we are only interested in ipv6 address.
We query to google resolver, since my local resolver didn't want to provide trace.
Last, we indicate +trace to ask for trace query output.
Output:

```
; <<>> DiG 9.11.3-1ubuntu1.11-Ubuntu <<>> AAAA @8.8.8.8 +trace grader.eecs.jacobs-uni
; (1 server found)
;; global options: +cmd
. 16286 IN NS b.root-servers.net.
. 16286 IN NS i.root-servers.net.
. 16286 IN NS g.root-servers.net.
. 16286 IN NS l.root-servers.net.
. 16286 IN NS h.root-servers.net.
. 16286 IN NS a.root-servers.net.
. 16286 IN NS f.root-servers.net.
. 16286 IN NS j.root-servers.net.
. 16286 IN NS m.root-servers.net.
. 16286 IN NS c.root-servers.net.
. 16286 IN NS e.root-servers.net.
. 16286 IN NS d.root-servers.net.
. 16286 IN NS k.root-servers.net.
. 16286 IN RRSIG NS 8 0 518400 20200511170000 20200428160000 48903 . GDayyDzTQl5+8J5U
;; Received 525 bytes from 8.8.8.8#53(8.8.8.8) in 11 ms

de. 172800 IN NS s.de.net.
de. 172800 IN NS n.de.net.
de. 172800 IN NS a.nic.de.
de. 172800 IN NS l.de.net.
de. 172800 IN NS f.nic.de.
de. 172800 IN NS z.nic.de.
de. 86400 IN DS 45580 8 2 918C32E2F12211766BE6226674F447458F2259B9A0D87B44D29D55AF EC
de. 86400 IN RRSIG DS 8 1 86400 20200512170000 20200429160000 48903 . jZ6S7a3B3X4Va0/
;; Received 766 bytes from 202.12.27.33#53(m.root-servers.net) in 28 ms

jacobs-university.de. 86400 IN NS dns.iu-bremen.de.
jacobs-university.de. 86400 IN NS www.jacobs-utils.de.
tjlb7qbojvmlf1s6gdriru7vsms1lg16.de. 7200 IN NSEC3 1 1 15 CA12B74ADB90591A TJLCMU7N3T
7vs4207ndhhgqeqji1f8580hafqjbu7p.de. 7200 IN NSEC3 1 1 15 CA12B74ADB90591A 7VS4TPU20I
tjlb7qbojvmlf1s6gdriru7vsms1lg16.de. 7200 IN RRSIG NSEC3 8 2 7200 20200511104928 2020
7vs4207ndhhgqeqji1f8580hafqjbu7p.de. 7200 IN RRSIG NSEC3 8 2 7200 20200511104928 2020
;; Received 619 bytes from 195.243.137.26#53(s.de.net) in 21 ms

grader.eecs.jacobs-university.de. 3600 IN CNAME cantaloupe.eecs.jacobs-university.de.
cantaloupe.eecs.jacobs-university.de. 3600 IN AAAA 2001:638:709:300::29
cantaloupe.eecs.jacobs-university.de. 3600 IN AAAA 2001:638:709:3000::29
eecs.jacobs-university.de. 3600 IN NS ns1.ibr.cs.tu-bs.de.
eecs.jacobs-university.de. 3600 IN NS dns.jacobs-university.de.
eecs.jacobs-university.de. 3600 IN NS ns.eecs.jacobs-university.de.
;; Received 296 bytes from 212.201.44.22#53(dns.iu-bremen.de) in 2 ms
```

First as we queried 8.8.8.8(Google resolver) we get a list of all options we can query later that will resolve lower level address.
From there we can notice that we went to m.root-servers.net, that provided us again plenty of options for authorative servers that are responsible for de. zone.

In our case the resolver decided to query s.de.net to look for resolution of next level.
We can see two possible options of NS that know the zone of jacobs-university.de.
The resolver contacted dns.iu-bremen.de to get the IP addresses for the gradergrader.eecs.jacobs-university.de..
Grader happened to have an alias name cantaloupe.eecs.jacobs-university.de. as we can see that in CNAME field of output. Out current NS knows the address for the later name, and provides us with AAAA records with 2 addresses that we can use to reach the grader website - 2001:638:709:300::29 and 2001:638:709:3000::29.

b)

SRV is defined in RFC 2782 (https://tools.ietf.org/html/rfc2782).
It is a resource record that allows to specify the location of the server for a specific protocol and domain according to rfc.
The idea is that if the client wants to contact some specific service he would query SRV.
The format of the lookup name is the following:
_ldap._tcp.example.com
Or in more general terms - _Service._Protocol.Name
The RR is in the following format:
_Service._Proto.Name TTL Class SRV Priority Weight Port Target
Service - is the symbolic name of the desired service.
Proto - is the symbolic name of the desired protocol.
Name - is the domain RR refers to.
TTL,Class - defined as in DNS (RFC 1035)
Priority - priority of the targeted host.
Weight - a server selection parameter for hosts with same priority.
Port - port on the target host of this service.
Target - domain name for this name (Must have one other entry (AA/AAAA))
Example:
_sip._tcp.example.de. 80000 IN SRV 0 5 5060 sip1.example.de.
Priority and weight are both 16 bit unsigned values.
Client must attempt to contact the target host with the lowest priority. However, when we have more than 1 entry with same priority, weight comes to play. The bigger weight implies bigger chance of getting chosen. When it comes to weight we do not have a rule which entry to choose, but a algorithm including random factor is used, so weight just describes the probability of choosing the entry in context of them having same priority.
c)
In theory maybe.
There are few potential use cases for it. One would be specific port relocating. That will allow for administrator to host multiple websites on one server, and allocate them to different ports, and the user would be able to retrieve that information from DNS. That was discussed in the following proposal : [Draft-http-srv-02].
Also SRV records allow weight and priority which can be used for load balancing and providing the However, right now browsers do not support it, and do not plan to implement the support of SRV records, so it would be impossible to use benefits of it. According to RFC of SRV it may not be used by protocols whcih do not specify the use of SRV. HTTP sadly hasn't specified that, so that is another reason not to use it.
Possible reason for that might underlay in the fact that that would require either new HTTP scheme, which would be unusable for general wev surfing, while if we use old scheme it would slow down transition times where different clients resolve URLs in different ways.
[Draft-http-srv-05].
d)
EDNS0 is defined in RFC6891.
It targets the issue with the maximum size of the message. Without EDNS0 the maximum limit of the message over UDP is 512 bytes, which is believed to bee too small to efficiently support the additional information that can be conveyed in the DNS (DNSSEC signatures, multiple IPv6 addresses). Also DNS as defined in RFC1035 is missing any way of advertising the capabilities of server to any other actor. [RFC6891].
In EDNS0 OPT RR CLASS has a meaning of requestor's UDP payload size, and TTL has extended RCODE and flags.
e)
Let's try testing it:
First let's query AAAA of facebook.com. to every resolver:

```
dig AAAA @8.8.8.8 facebook.com.
;; ANSWER SECTION:
```

```
facebook.com. 297 IN AAAA 2a03:2880:f13f:83:face:b00c:0:25de

dig AAAA @1.1.1.1 facebook.com.
;; ANSWER SECTION:
facebook.com. 192 IN AAAA 2a03:2880:f11c:8183:face:b00c:0:25de

dig AAAA @9.9.9.9 facebook.com.
;; ANSWER SECTION:
facebook.com. 298 IN AAAA 2a03:2880:f11c:8083:face:b00c:0:25de
```

As we can see the answer differs sometimes.

Let's query A of facebook.com to every resolver:

```
dig A @9.9.9.9 facebook.com.
;; ANSWER SECTION:
facebook.com. 129 IN A 31.13.92.36

dig A @1.1.1.1 facebook.com.
;; ANSWER SECTION:
facebook.com. 174 IN A 31.13.92.36

dig A @8.8.8.8 facebook.com.
;; ANSWER SECTION:
facebook.com. 9 IN A 157.240.27.35
```

It looks like IPv4 also has some diffence in addresses. Facebook has multiple servers with different addresses, so it is not surprising that we recieved different answers.

Problem 6.2

a)
mDNS is defined in RFC6762
If the local network doesn't support name server, that will make it very difficult to access any other device in such network, especially to normal users. mDNS takes a role of DNS here, and allows to locate and communicate with other devices on a local network, such as home network, without any need of administration or configuration.
By the default the top level domain is alway .local. When the client needs to resolve a name he sends a request to indentify to IP multicast address (224.0.0.251 or ff02::fb) using UDP. Target would respond with multicast message having its IP address using the syntax of DNS package. The responses can be locally cached.[https://tools.ietf.org/html/rfc6762]
b)
DNS-Based Service Discovery is defined in RFC6763.
It allows clients to discover a list of named instances of the service, using DNS queries.
Services are described using DNS SRV and TXT rrs. SRV holds target host and port of the service instance, and TXT of the same service provides key/value pairs with additional info. Important to notice that every service has to have also TXT entry, even of size 0.
Client queries PTR record with a name of desired service in a form Service.Domain, and recieves 0 or more PTR records listing names of services associated with DNS SRV/TXT pairs.
Service Instance Name = ¡Instance¿ . ¡Service¿ . ¡Domain¿
[https://tools.ietf.org/html/rfc6763]