Computer Network Assignment 6

Yiping Deng

May 10, 2019

Problem 6.1

a)

There are several steps concerning the lookup of `grader.eecs.jacobs-university.de`. We need to lookup the AAAA records for IPv6 records. We use the following dig command to get the result

```
dig +trace grader.eecs.jacobs-university.de AAAA
```

and we will end up with the following result

```
; <<>> DiG 9.10.6 <<>> +trace grader.eecs.jacobs-university.de AAAA
;; global options: +cmd
. 452315 IN NS a.root-servers.net.
. 452315 IN NS e.root-servers.net.
. 452315 IN NS b.root-servers.net.
. 452315 IN NS j.root-servers.net.
. 452315 IN NS c.root-servers.net.
. 452315 IN NS h.root-servers.net.
. 452315 IN NS i.root-servers.net.
. 452315 IN NS k.root-servers.net.
. 452315 IN NS g.root-servers.net.
. 452315 IN NS f.root-servers.net.
. 452315 IN NS d.root-servers.net.
. 452315 IN NS l.root-servers.net.
. 452315 IN NS m.root-servers.net.
. 489115 IN RRSIG NS 8 0 518400 20190522050000 20190509040000 25266 . yBg5V5MdhPQcGc+FYRa9c7Jxh1tKSr
;; Received 1097 bytes from 10.70.0.20#53(10.70.0.20) in 1 ms

de. 172800 IN NS s.de.net.
de. 172800 IN NS l.de.net.
de. 172800 IN NS z.nic.de.
de. 172800 IN NS n.de.net.
de. 172800 IN NS f.nic.de.
de. 172800 IN NS a.nic.de.
de. 86400 IN DS 39227 8 2 AAB73083B9EF70E4A5E94769A418AC12E887FC3C0875EF206C3451DC 40B6C4FA
de. 86400 IN RRSIG DS 8 1 86400 20190522160000 20190509150000 25266 . b25ARMch60l7X6xtCtG3Z2zrDWtenl
;; Received 738 bytes from 199.9.14.201#53(b.root-servers.net) in 125 ms

jacobs-university.de. 86400 IN NS dns.iu-bremen.de.
jacobs-university.de. 86400 IN NS www.jacobs-utils.de.
H319DM5GC3EDEK691VQBHEHOT7VGGJ2B.de. 7200 IN NSEC3 1 1 15 BA5EBA11 H31EGRUDRBMFSM3HAQ6AMG96SJB4QAVI
H319DM5GC3EDEK691VQBHEHOT7VGGJ2B.de. 7200 IN RRSIG NSEC3 8 2 7200 20190516190105 20190509190105 2629
SFAC58VBFNB14JPD7N2H5MOLE1O2L213.de. 7200 IN NSEC3 1 1 15 BA5EBA11 SFAEPI3LFEOB8NDDLF1S5816FTJPO3AS
SFAC58VBFNB14JPD7N2H5MOLE1O2L213.de. 7200 IN RRSIG NSEC3 8 2 7200 20190516190105 20190509190105 2629
;; Received 611 bytes from 195.243.137.26#53(s.de.net) in 20 ms
```

```
grader.eecs.jacobs-university.de. 3600 IN CNAME cantaloupe.eecs.jacobs-university.de.
cantaloupe.eecs.jacobs-university.de. 3600 IN AAAA 2001:638:709:3000::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 240 bytes from 212.201.44.22#53(dns.iu-bremen.de) in 3 ms
```

The lookup process are:[3]

1. A DNS query is fired to our DNS recursivse resolver provided via DHCP. In this case, it is `10.70.0.20`. The query asked for AAAA records for `grader.eecs.jacobs-university.de`.
2. The local DNS resolve on the network returns 13 root server to us. We use one of the root server `g.root-servers.net` to proceed with the query of `de.`
3. we got 6 nameserver responsible for `de.` top level domains, we choose `s.de.net` to proceed with the next subdomain `jacobs-university.de.`.
4. It gave us two nameserver, `www.jacobs-utils.de.` and `dns.iu-bremen.de.`, we choose `www.jacobs-utils.de.` to proceed with next level of subdomain, `eecs.jacobs-university.de.`
5. We receive 3 nameservers at this level, then we proceed with `dns.jacobs-university.de` to finally lookup the AAAA record for `grader.eecs.jacobs-university.de.`.
6. `dns.jacobs-university.de` found out that `grader.eecs.jacobs-university.de.` contains a CNAME record pointing to `cantaloupe.eecs.jacobs-university.de.`. In the same response, we received AAAA records `cantaloupe.eecs.jacobs-university.de. 3600 IN AAAA 2001:638:709:3000::29` together with additional information. Domain name lookup is finally complete.

b)

SRV resoure record follows the following format:[4]

```
_Service._Proto.Name TTL Class SRV Priority Weight Port Target
```

where

- *Service* denotes the symbolic name of desired service, and an underscore is prepended to avoid the collision with natural DNS labels, and it is case insensitive.
- *Proto* denotes the symbolic name of the protocol, with a underscore prepended. `_TCP` and `_UDP` are the two most commonly used protocols. It is alsos case insensitive.
- *Name* denotes domain name it refers to (will be valid within).
- *TTL* denotes common DNS TTL.
- *Class* denotes common DNS Class.
- *Prority* denotes the priority of this target host.
- *Weight* denotes the relative weight for entries with the same priority.
- *Port* denotes the port number of the service.
- *Target* denotes the domain name of the the host providing the service.

It is designed to "be used by clients for applications where the relevant protocol specification indicates that clients should use the SRV record.".[4] It is commonly used to identify servers that host specfic service with certain protocol. It is properly defined in RFC2782, and it is stored in normal zone file.

For example,

```
_ldap._tcp.example.net. 3600 IN   SRV  0 0 389 phoenix.example.net.
```

denotes that it is serving LDAP on TCP protocol, with port number 389, 0 priority and 0 weight and the server is hosted on `phoenix.example.net.`.

Prority and weight are used in totally different way. Firstly, "a client **must** attempt to contact the target host with the lowest-numbered priority it can reach." When 2 entries have the same priority, the one with **higher** weight should have proportionally higher probability to be selected.

c)

Using SRV resoruce record is a debatable topic. There are several pros and cons for such matters.

Pros	Cons
SRV is designed in a way that we can easily achieve client-side load balancing. This can save a lot of cost for infrastructures cost for website owners. SRV resource record on HTTP will allow websites to be served on any ports, not just port 80. As some ISP will block 80 to save upbound traffic, it is particularly useful to serve website on home ISP network. It can also bypass certain censorship on the network and firewalls. HTTP SRV resource record can redirect traffic to different domains under different port numbers, allowing higher configurability in web services. For distributed system, SRV resource record can be used for service discoverery. This is useful as we are in the age of microservice architecture, and service discoverery is the key to microservice architecture.	Using SRV might end up with non-deterministic visit to the host to open a website. It is hard to aggregate access information and statistics, and it is hard to pinpoint failure servers. The load balancing provided through SRV will lead to security issue. As the internal infrastructures are exposed, targeting a single server has never been easier. The idea of serving HTTP on any ports via SRV resource record is terrifying. Those HTTP traffic might bypass firewalls, leading to security vulnerability. Cutting off traffic for HTTP server will be much hard. Imagine the case of a child pornographic website serving using SRV resource record, port blacklist will not be effective anymore. The idea of redirect traffic to any server on any port can be abused and used to launch denial of service attack. Next level of DNS spoofing can be achieved. You not only are able to redirect to different addresses, you can even redirect to different port number.

Table 1: Pros and cons of using SRV resource record for HTTP

d)

As the Internet grow and introduction of IPv6, as well as the invention of stricter security measures with DNS, more and more bytes are stuffed in DNS messages. However, traditional DNS messages have a upper bound for the size–512 bytes. What's more, previous DNS protocol doesn't contain a mechanism to advertise the capabilities to others. Certainly, this protocol is no longer suitable for the development and the scalability of the Internet.

Hence, in RFC6891, EDNS0 is introduced. "E" stands for extension mechanisms. It provides DNS messages with the capability to extend beyond the previous size limit, and it provides extra data space for flags and return codes.

In the OPT resource record, CLASS field is used to store the requester's UDP payload size. It denotes the number of bytes of the largest UDP payload that can be delivered in the requester's network stack. TTL field is used to store extended RCODE, version and flags. Extended RCODE is 8 bits, used together with the original RCODE. It also contains a version field indicating the lowest implemented level, and full support of RFC6891 is indicated by '0'. There are 2 flags supported, DO flags indicate the usage of DNSSEC. Z flags are usually 0, and are designed for later usage.

e)

We are using dig to find all the different A and AAAA records, and here are the result for amazon.com:

DNS address	Provider	A	AAAA
1.1.1.1	Cloudflare	176.32.103.205 205.251.242.103 176.32.98.166	N/A
8.8.8.8	Google	205.251.242.103 176.32.103.205 176.32.98.166	N/A
9.9.9.9	Quad9	205.251.242.103 176.32.98.166 176.32.103.205	N/A

Table 2: Amazon DNS records on public nameservers

We can see that amazon.com does not support IPv6 yet, which is quite sad. They always have the same answer, quite boring.

Let's test it again on Google.com:

DNS address	Provider	A	AAAA
1.1.1.1	Cloudflare	172.217.22.14	2a00:1450:4001:815::200e
8.8.8.8	Google	172.217.16.206	2a00:1450:4001:821::200e
9.9.9.9	Quad9	172.217.22.46	2a00:1450:4001:821::200e

Table 3: Google DNS records on public nameservers

This time, clearly we can see IPv4 address are different for every server, and IPv6 is different for Cloudflare. As we observe from the IPv6, Cloudflare and Quad9 address are in the same class C subnet, where the Google server report a server in a differnet class C subnet.

Problem 6.2

a)

Multicast DNS provides the capabilities to look up DNS resource record in absence of conventional managed DNS server, and it designs "designates a portion of the DNS namespace to be free for local use".[2] There are three primary benefit of mDNS:

1. It requires little or no administration or configuration to setup.
2. It works with no extra infrastructure.
3. It works on infrastructure failures.

Multicast DNS name will have the following format:[2]

`single-dns-label.local.`

This protocol is defined in RFC6762.[2]

It depart from regular DNS protocol semantics in several ways:

1. It sends requests to multicast addresss `224.0.0.251:5353` (or `[FF02::FB]:5353` in IPv6), and the response is normally in the same multicast fashion. The response have a random delay of up to 500ms to avoid collisions with other responder, and it doesn't have a question section.
2. It then pickup the first response and resolve the addresses.
3. It can also operate on continuous querying, where the querying operation continues until no further responses are required, depending on the type of operation being performed.[2]

4. The TTL values are respected in a probabilistic way. “To avoid the case where multiple Multicast DNS queriers on a network all issue their queries simultaneously, a random variation of 2% of the record TTL should be added, so that queries are scheduled to be performed at 80-82%, 85-87%, 90-92%, and then 95-97% of the TTL.”[2]

b)

DNS-based service discovery is defined in RFC6763. It specifies “how DNS resource records are named and structured to facilitate service discovery”.[1] It allows client to discover a list of named instances with certain service using DNS queries.

Two types of DNS records can be used to facilitate service discovery–SRV and TXT records.

SRV record, for service discovery, has the following form “`<Instance>.<Service>.<Domain>`”[1] and it provides the target host and port where it can reach. Example has been given above (item).

TXT record, in the other hand, has higher flexibility. It is stored as key/value pairs. It can provide more information than just IP addresses and port numbers. For example, a file server might have different volumns, and those information can be conveyed in the TXT record. Client fires a DNS PTR record query, and it will receive a set of zero or more names, which are the names of the DNS SRV/TXT record pairs. Even if all the information are in the SRV record, a TXT record is needed (empty record here).

References

[1] S. Cheshire and M. Krochmal. *DNS-Based Service Discovery.* RFC 6763. `http://www.rfc-editor.org/rfc/rfc6763.txt`. RFC Editor, 2013. URL: `http://www.rfc-editor.org/rfc/rfc6763.txt`.

[2] S. Cheshire and M. Krochmal. *Multicast DNS.* RFC 6762. `http://www.rfc-editor.org/rfc/rfc6762.txt`. RFC Editor, 2013. URL: `http://www.rfc-editor.org/rfc/rfc6762.txt`.

[3] Cloudflare. *What Is DNS? — How DNS Works.* `https://www.cloudflare.com/learning/dns/what-is-dns/`.

[4] Arnt Gulbrandsen, Paul Vixie, and Levon Esibov. *A DNS RR for specifying the location of services (DNS SRV).* RFC 2782. `http://www.rfc-editor.org/rfc/rfc2782.txt`. RFC Editor, 2000. URL: `http://www.rfc-editor.org/rfc/rfc2782.txt`.