Networks and Distributed Systems: Assignment 2

20 February 2020

General Information

Deadline: Monday, 2 March, 15:00 (Amsterdam time)

Submission Guidelines:

- This is an *individual* assignment. In other words, it is OK to discuss, but do not share your answers.
- Combine a single PDF file *s0123456.pdf* containing your written answers with any supplemental files in a single ZIP file *s0123456.zip*, where *s0123456* should be replaced by your actual student number.
- In addition to the above, make sure that each of the deliverables contains both your name and student number.
- Submit the ZIP file via Brightspace (`http://brightspace.ru.nl`).

Marks: You will be graded with marks from 0 to 3 where 0 means not serious, 1 means serious but insufficient, 2 means sufficient and 3 means good. You can have at most one assignment graded 0. To get the mark 1 or higher, you **must** attempt to solve **all** exercises. Even if your solution to a problem is not correct or complete, we expect you to submit your work and explain where you got stuck and why. In other words, leaving an exercise out automatically turns your grade to 0. In your solution, please explain all answers clearly and concisely.

Note for repeat students: Students who were part of this course last year (2018-2019) *may* choose to resubmit (parts of) their work on the assignments, if they have *at least* a sufficient mark. Your new submission (ZIP) must then contain your complete submission (ZIP) from last year. Also include a `resubmit.txt` file, explaining which of the problems in the assignment can be found in the old submission. Failing to include solutions for changed or new problems in an assignment will automatically turn your grade into a 0.

Goals: After completing these exercises successfully you should:

- be able to use the `dig` utility to perform DNS lookups;
- understand DNS encoding and compression techniques;
- understand DNS caching and be able to parse a simple DNS database;
- understand TCP and UDP (de)multiplexing;
- understand how to compute an Internet checksum;
- be able to implement your own ICMP-based `traceroute`.

Exercises

1 Interrogating DNS name servers

In this problem, we use the `dig` tool available on Unix-like systems to explore the hierarchy of DNS servers. If your system is running Windows, then you will have to download the BIND DNS software, which contains the `dig` utility. You can download the latest version of BIND here `http://www.isc.org/downloads/bind/` or you can find `dig` in the Windows subsystem for Linux.

Figure 1 shows the process of a DNS lookup. The resolver sends a recursive query to the name server. This means that it is the responsibility of the name server to find the mapping on behalf of the resolver. This mapping is returned to the resolver at the end. The name server accomplishes this by a series of iterative queries to the various DNS servers in the hierarchy. A DNS server higher in the DNS hierarchy delegates a DNS query to a DNS server lower in the hierarchy by sending back to the name server the address of that lower-level DNS server.

First read `dig`'s man page[1]. then solve the questions. As a remark, the figure was taken from Microsoft's website, which provides a valuable reference about DNS, see [2].

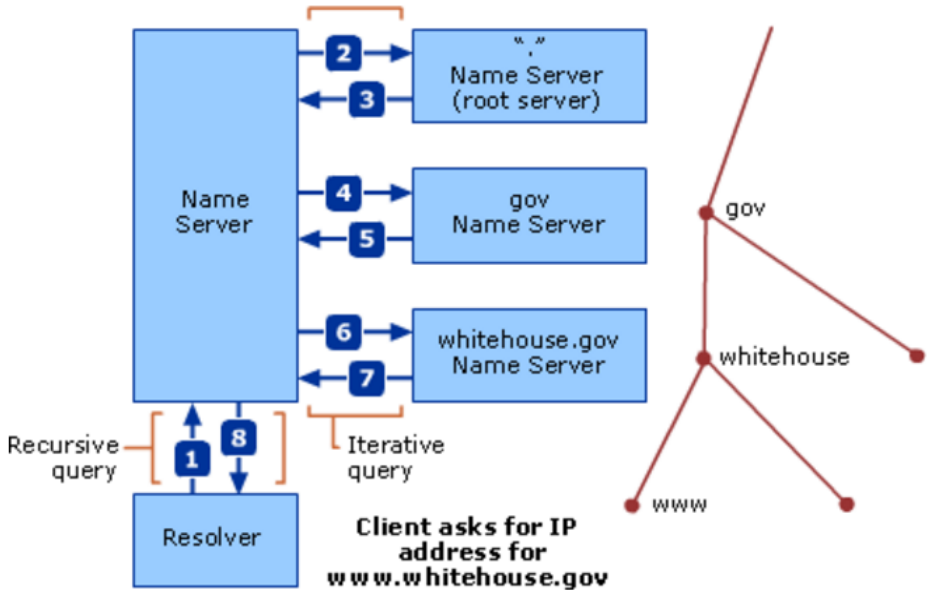


Figure 1: Route from client to server

a) Starting with a root DNS server (from one of the root servers [a-m].root-servers.net), initiate a sequence of iterative queries for *cs.ru.nl* using `dig`. For each query, you have to specify the name server (which is prefixed by @) and the hostname to be resolved. Give a printout of the output of each query in the chain.

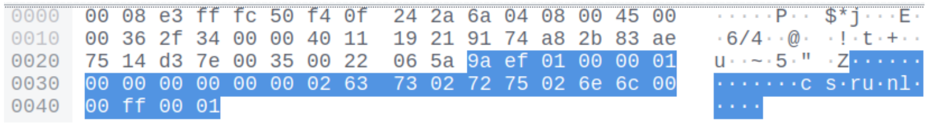
b) For the hostname *cs.ru.nl*, trace the delegation path using the +trace query option. Without giving a printout, explain what happens. Optionally, if unexpected or no output is returned, you can try an online DNS delegation tracer, for example [3].

[1]`https://linux.die.net/man/1/dig`
[2]`https://technet.microsoft.com/en-us/library/dd197461(v=ws.10).aspx`
[3]`http://simpledns.com/lookup-dg.aspx`

2 Decoding DNS messages

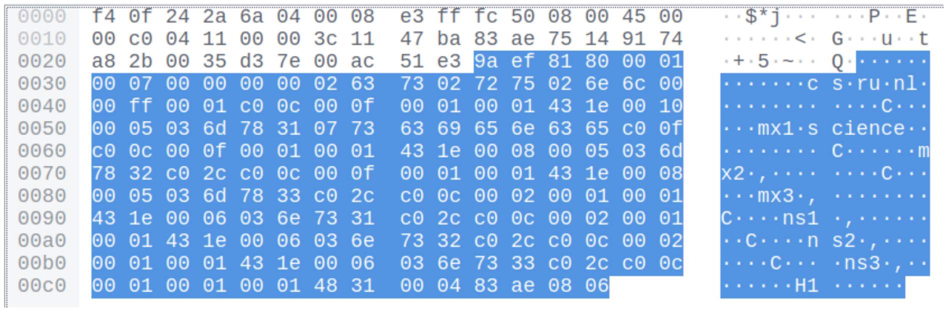
DNS messages follow a strict message format. Moreover, encoding and compression techniques are applied to make the messages understandable to the receiving end, as well as to reduce the message size. These techniques, as well as the message format, are described in chapter 4 of RFC 1035[4]. More accessible explanations for encoding and compression techniques can be found online [5].

Figure 2 presents a Wireshark capture of a DNS Query. Its corresponding response is seen in figure 3. Your task is to figure out what domain is being resolved and to enumerate the types and contents of the returned resource records.



```
0000  00 08 e3 ff fc 50 f4 0f  24 2a 6a 04 08 00 45 00   ·····P·· $*j···E·
0010  00 36 2f 34 00 00 40 11  19 21 91 74 a8 2b 83 ae   ·6/4··@· ·!·t·+··
0020  75 14 d3 7e 00 35 00 22  06 5a 9a ef 01 00 00 01   u··~·5·" ·Z······
0030  00 00 00 00 00 00 02 63  73 02 72 75 02 6e 6c 00   ·······c s·ru·nl·
0040  00 ff 00 01                                        ····
```

Figure 2: A DNS query as captured by Wireshark. Relevant hexadecimal encoding is in blue.



```
0000  f4 0f 24 2a 6a 04 00 08  e3 ff fc 50 08 00 45 00   ··$*j··· ···P··E·
0010  00 c0 04 11 00 00 3c 11  47 ba 83 ae 75 14 91 74   ······<· G···u··t
0020  a8 2b 00 35 d3 7e 00 ac  51 e3 9a ef 81 80 00 01   ·+·5·~·· Q·······
0030  00 07 00 00 00 00 02 63  73 02 72 75 02 6e 6c 00   ·······c s·ru·nl·
0040  00 ff 00 01 c0 0c 00 0f  00 01 00 01 43 1e 00 10   ········ ····C···
0050  00 05 03 6d 78 31 07 73  63 69 65 6e 63 65 c0 0f   ···mx1·s cience··
0060  c0 0c 00 0f 00 01 00 01  43 1e 00 08 00 05 03 6d   ········ C······m
0070  78 32 c0 2c c0 0c 00 0f  00 01 00 01 43 1e 00 08   x2·,···· ····C···
0080  00 05 03 6d 78 33 c0 2c  c0 0c 00 02 00 01 00 01   ···mx3·, ········
0090  43 1e 00 06 03 6e 73 31  c0 2c c0 0c 00 02 00 01   C····ns1 ·,······
00a0  00 01 43 1e 00 06 03 6e  73 32 c0 2c c0 0c 00 02   ··C····n s2·,····
00b0  00 01 00 01 43 1e 00 06  03 6e 73 33 c0 2c c0 0c   ····C··· ·ns3·,··
00c0  00 01 00 01 00 01 48 31  00 04 83 ae 08 06         ······H1 ······
```

Figure 3: A DNS response to the previous query as captured by Wireshark. Relevant hexadecimal encoding is in blue.

3 DNS cache and database

Suppose you access *www.random.cs.nl* via a local DNS server on campus. We assume that the local DNS server knows the address of a root DNS server. When referring to hostnames we use the Fully Qualified Domain Name (FQDN). The device with hostname *www* in the parent domain *random.cs.nl* has FQDN *www.random.cs.nl.* (Note the final dot).

a) Provided that no caching is used and queries are resolved iteratively, enumerate the hostnames that need to be resolved for the given FQDN, stating the order in which they are resolved.

b) Now let's assume that the local DNS server implements caching. Of the hostnames to be resolved, choose the one most *and* least likely to be found stored in the local DNS server's cache. Explain your choices.

c) How can you use DNS to determine whether a website was recently accessed?

d) Every time a hostname is resolved by a query, the local DNS server caches the response in a DNS record. The record, among others, contains the time-to-live (TTL) value which dictates when a

[4]`https://tools.ietf.org/html/rfc1035`
[5]`http://www.tcpipguide.com/free/t_DNSNameNotationandMessageCompressionTechnique.htm`

resource record should be removed from the cache. Relate this TTL value to the likelihood of a cache hit. In other words, does the TTL value tell you something about the frequency with which the record is returned?

Suppose a DNS database contains the following resource records for the domain *random.cs.nl.*:

random.cs.nl.	86400	IN	NS	alpha
random.cs.nl.	86400	IN	MX	1. beta
random.cs.nl.	86400	IN	MX	2. delta
alpha	86400	IN	A	130.122.131.10
beta	86400	IN	A	131.122.131.10
delta	86400	IN	A	132.122.131.10
www	86400	IN	CNAME	alpha
ftp	86400	IN	CNAME	alpha
imap	86400	IN	CNAME	alpha
printer	86400	IN	A	134.122.131.10

e) What is the FQDN of the printer? What is the IPv4 address of the printer?

f) Suppose you send an email to a user within the "random" domain. Which mail server is most likely to be used? List its hostname and IPv4 address.

g) You will notice multiple CNAME entries: for a www server, an ftp server and an imap server respectively. What would be an advantage of using CNAME type entries as opposed to A type entries in this context?

4 TCP and UDP (de)multiplexing

Suppose Client A requests a web page from Server S through HTTP. At about the same time, Client B also requests a web page from Server S through HTTP. Provide possible source and destination port numbers for:

a) The segments sent from A to S.

b) The segments sent from B to S.

c) The segments sent from S to A.

d) The segments sent from S to B.

e) In the answer that you gave to the above, does it matter whether A and B are the same host or not? Explain why (not).

f) Assuming that these are the only ongoing TCP connections, how many open sockets does Server S have?

g) Suppose now that DNS queries are sent instead of HTTP requests. How many open sockets does Server S have in this case?

5 Internet checksum

The checksum used by the standard Internet protocols IP (including ICMP), UDP, and TCP is called the Internet checksum. It is specified in RFC 1071 [6]. Suppose you have the following three bytes: 01010011, 01100110, 01110100. Answer the following:

a) What is the 1s complement of the binary sum of these bytes? (Note that although IP, UDP, and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work.

b) Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum?

c) With the 1s complement scheme, how does the receiver detect errors?

d) Is it possible that a 1-bit error will go undetected? How about a 2-bit error? Explain why (not).

6 ICMP traceroute

In the previous assignment you have experimented with `traceroute`, a tool which allows the user to trace the route from the host running `traceroute` to any host in the world. Included with the assignment, you will find a file called traceroute.py. It contains an incomplete implementation of `traceroute` in Python using ICMP messages. For information about the structure of such messages refer to RFC 792[7]. Provided to you are working routines to build a packet containing an `ICMP_ECHO_REQUEST` message and send it over a raw socket, a socket that allows you to bypass the transport layer and send directly to the network layer. Note that opening such a socket requires root/admin privileges.

The program takes as parameter a hostname, which for our purposes is either an IPv4 address or a domain name. The program should send three probes (`ICMP_ECHO_REQUEST` messages) with a given TTL where the TTL increases with each batch of probes sent, up to some maximum. The routers along the traceroute path will send back an `ICMP_TIME_EXCEEDED` message when the TTL field becomes zero. The final destination will respond with an `ICMP_ECHO_REPLY` message upon receiving the `ICMP_ECHO_REQUEST` message.

For each hop on the network path to the destination host, the program should print its FQDN and IPv4 address, followed by three RTT measurements (corresponding to the three probes). The FQDN and IPv4 address can be extracted from the received packets. The RTT is determined by setting a timer at the sending host.

In case no response for a probe was received the program should print *** in place of the RTT. If for all three probes a response is lacking, then the program does not print the FQDN and IPv4 address, but instead ** * ** is printed. You do not need to worry about the accurracy of the RTT measurements. For example, you do not need to start a timer at the exact moment of sending. The maximum number of hops is 30 and the timeout is set to 1 second.

a) Write the body of the in_cksum method. This method should return the Internet checksum as specified in RFC 1071. Its input is a byte sequence.

b) Finish the body of the main method. Your program is expected to handle incoming packets of type `ICMP_ECHO_REPLY` and `ICMP_TIME_EXCEEDED`.

Include the completed program file in the assignment ZIP file. In addition to the program file, the PDF file with your written answers should include a brief (yet complete) explanation of how your code works.

Please read the instructions at the beginning of this assignment carefully!

[6] `https://tools.ietf.org/html/rfc1071`
[7] `https://tools.ietf.org/html/rfc792`