

Introduction to Web Science

Assignment 3

Prof. Dr. Steffen Staab

staab@uni-koblenz.de

René Pickhardt

rpickhardt@uni-koblenz.de

Korok Sengupta

koroksengupta@uni-koblenz.de

Institute of Web Science and Technologies

Department of Computer Science

University of Luxembourg

Submission until: November 16, 2016, 10:00 a.m.

Tutorial on: November 18, 2016, 12:00 p.m.

The main objective of this assignment is for you understand different concepts that are associated with the "Web". In this assignment we cover two topics: 1) DNS & 2) Internet.

These tasks are not always specific to "Introduction to Web Science". For all the assignment questions that require you to write a code, make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.

Group name: uniform

Group members: Pradip Giri, Jalak Arvind Kumar Pansuriya, Madhu Rakhal Magar

1 DIG Deeper (5 Points)

Assignment 1 started with you googling certain basic tools and one of them was "dig".

1. Now using that dig command, find the IP address of www.uni-koblenz-landau.de
2. In the result, you will find "SOA". What is SOA?
3. Copy the SOA record that you find in your answer sheet and explain each of the components of SOA with regards to your find. Merely integrating answers from the internet won't fetch you points.

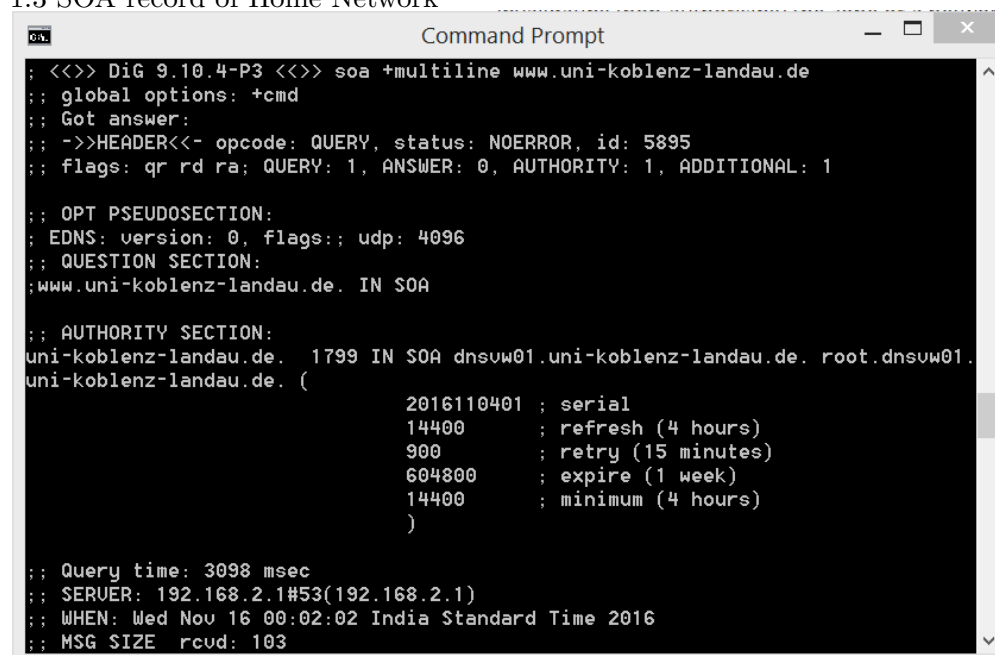
Try the experiment once from University network and once from Home network and see if you can find any differences and if so, clarify why.

Answers:

1.1 The IP address of www.uni-koblenz-landau.de is 141.26.200.8

1.2 DNS authority record(SOA) specifies authoritative information about a DNS zone, including the primary name server, the email of the domain administrator, the domain serial number, and several timers relating to refreshing the zone.

1.3 SOA record of Home Network



```
Command Prompt
: <<>> DiG 9.10.4-P3 <<>> soa +multiline www.uni-koblenz-landau.de
: global options: +cmd
: Got answer:
: ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5895
: flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
:
: OPT PSEUDOSECTION:
: EDNS: version: 0, flags:; udp: 4096
: QUESTION SECTION:
: www.uni-koblenz-landau.de. IN SOA
:
: AUTHORITY SECTION:
: uni-koblenz-landau.de. 1799 IN SOA dnsuw01.uni-koblenz-landau.de. root.dnsuw01.
: uni-koblenz-landau.de. (
:                               2016110401 ; serial
:                               14400      ; refresh (4 hours)
:                               900       ; retry (15 minutes)
:                               604800    ; expire (1 week)
:                               14400    ; minimum (4 hours)
:                               )
:
: Query time: 3098 msec
: SERVER: 192.168.2.1#53(192.168.2.1)
: WHEN: Wed Nov 16 00:02:02 India Standard Time 2016
: MSG SIZE rcvd: 103
```

First of all, we discussed about components of SOA record based on output of Home Network:

- a) Name :- Here, [uni-koblenz-landau.de](#) is the domain name for the primary name server where original data of [www.uni-koblenz-landau.de](#) comes from. All changes can be made to the primary name server only.
- b) TTL :- Time-To-Live shows the duration for that record will be cached on other name servers. TTL for this SOA record is *1799* seconds which means this server has to remove the entry from its cache after 1799 seconds.
- c) Class :- We got this record by Internet. So, the address class of this record is *IN*.
- d) Record Type :- This field shows the type of DNS record is *SOA*.
- e) NameServer :- [dnsvw01.uni-koblenz-landau.de](#) - This domain name is followed by dot which indicates this is an external server to store all the data files.
- f) E-mail :- It shows the email of domain name administrator is [root.dnsvw01.uni-koblenz-landau.de](#) which follows [username.domail.tdl](#) format. In this email, sign '@' is replaced by '.' (period). For Example, [root.dnsvw01.uni-koblenz-landau.de](#) is used instead of [root@dnsvw01.uni-koblenz-landau.de](#) to overcome some displaying problem.
- g) Serial Number :- *2016110401* - This serial number shows that primary name server lastly updated on 4th November, 2016 twice.
- h) Refresh :- In every *14400* seconds, secondary name server will check for update to primary name server if serial number has increased.
- i) Retry :- When secondary name server failed to check for updates at primary name server, it should wait for *900* seconds after this failure to reconnect with this primary name server.
- j) Expiry :- When primary name server will go offline, secondary name server would still be able to handle all queries related to DNS server using its cache information until *1 week*. After this duration, it will be no longer authoritative.
- k) Minimum :- This server should cache the primary name server record minimum for *4 hour*.

After that, we have run same command from university network also.

SOA record of University Network

```
C:\Users\mepra>dig soa www.uni-koblenz-landau.de
; <<> DiG 9.9.6 <<> soa www.uni-koblenz-landau.de
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 43385
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.uni-koblenz-landau.de.      IN      SOA
;; AUTHORITY SECTION:
uni-koblenz-landau.de.  387     IN      SOA      dnsvw01.uni-koblenz-landau.de. root.dnsvw01.uni-koblenz-landau.de. 2016110401 14400 900 604800 14400
;; Query time: 15 msec
;; SERVER: 141.26.64.60#53(141.26.64.60)
;; WHEN: Tue Nov 15 12:32:46 W. Europe Standard Time 2016
;; MSG SIZE rcvd: 103
```

At last, we can see from above both screenshots of outputs in different networks are almost same except current TTL value. TTL values change by experienced administrators periodically according increased load on name servers.

2 Exploring DNS (10 Points)

In the first part of this assignment you were asked to develop a simple TCP Client Server. Now, using **that** client server setup. This time a url should be send to the server and the server will split the url into the following:

`http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#InTheDocument`

1. Protocol
2. Domain
3. Sub-Domain
4. Port number
5. Path
6. Parameters
7. Fragment

The Protocol for sending the URL will be a string terminated with `\r \n`.

P.S.: You are **not** allowed to use libraries like `urlparse` for this question. You will also not use "Regular Expressions" for this.

Answer:

Listing 1 Supporting File

```
1: # pylint: disable-msg=C0103
2: """
3: Exploring more DNS
4: """
5:
6:
7: def extract_url_and_fragemnt(input_url):
8:     """ extract fragment from the url
9:     return fragment and url without fragment
10:    """
11:     clean_url = None
12:     frag = None
13:     if input_url.find("#") > 0:
14:         clean_url, frag = input_url.split("#", 1)
15:     else:
16:         clean_url = input_url
17:     return (clean_url, frag)
18:
19:
```

```
20: def extract_query_from_url(input_url):
21:     """ extract query from the url
22:     returns query and url with out query
23:     if query is not present the empty string
24:     is returned
25:     """
26:     query_path = ''
27:     without_path = input_url
28:     if input_url.find("?") > 0:
29:         without_path, query_path = input_url.split("?")
30:     return(query_path, without_path)
31:
32:
33: def extract_domain_and_subdomain(raw_domain):
34:     """
35:     extract domain and sub domain name
36:     from given url and returns it
37:     """
38:     sub_domain_inner = None
39:     main_domain = None
40:     splitted_domain = raw_domain.split(".")
41:     if splitted_domain[0] == "www":
42:         main_domain = raw_domain
43:     else:
44:         sub_domain_inner = splitted_domain[:1][0]
45:         main_domain = ".".join(splitted_domain[1:])
46:     return(main_domain, sub_domain_inner)
47:
48:
49:
50:
51:
52:
53: def extract_domain_and_port_info(input_path):
54:     """
55:     extract full domain name and port number
56:     from given url and returns it
57:     """
58:     inner_port = None
59:     # determine if there is inner_port
60:     if input_path.find(":") > 0:
61:         rawdomain, rawport = input_path.split(":")
62:         if rawport.find("/"):
63:             rawport = rawport.split("/")
64:             try:
65:                 inner_port = int(rawport[0])
66:             except AttributeError:
67:                 inner_port = None
68:     # sometimes there won't be inner_port
```

```
69:     elif input_path.find("/") > 0:
70:         rawdomain = input_path.split("/")[0]
71:     else:
72:         rawdomain = input_path
73:     main_domain, inner_sub_domain = extract_domain_and_subdomain(rawdomain)
74:     return (main_domain, inner_sub_domain, inner_port)
75:
76:
77: def extract_path_from_url(input_url):
78:     """ extract input_path from given url
79:     and returns path and url without path
80:     """
81:     clean_url = input_url.split("://", 1)[0]
82:     if clean_url.find('/') >= 0:
83:         return "/" + clean_url.split("/", 1)[1]
84:     return ''
```

Listing 2 Server Side Code

```
1: # pylint: disable-msg=C0103
2: """ Simple Web Server in Python
3:     using socket
4: """
5: import socket
6: from assignment_3_2 import extract_path_from_url
7: from assignment_3_2 import extract_query_from_url
8: from assignment_3_2 import extract_url_and_fragemnt
9: from assignment_3_2 import extract_domain_and_port_info
10:
11:
12: HOST = '127.0.0.1'
13: PORT = 8080
14: BUFFER_SIZE = 1024
15:
16: s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17: s.bind((HOST, PORT))
18: s.listen(1)
19: conn, addr = s.accept()
20: while True:
21:     data = conn.recv(BUFFER_SIZE)
22:     if not data:
23:         break
24:     raw_url = data.decode('utf-8')
25:     url, fragment = extract_url_and_fragemnt(raw_url.strip())
26:     protocol, url_without_protocol = url.split("://")
27:     domain, s_domain, port = extract_domain_and_port_info(url_without_protocol)
28:     query, url_without_path = extract_query_from_url(url_without_protocol)
29:     path = extract_path_from_url(url_without_path)
30:     print("path: {}".format(path))
31:     print("domain: {}".format(domain))
```

```
32:     print("subdomain: {}".format(s_domain))
33:     print("port: {}".format(port))
34:     print("query: {}".format(query))
35:     print("protocol: {}".format(protocol))
36:     print("fragment: {}".format(fragment))
37:     conn.send(data)
38: conn.close()
```

```
~/d/a/w/u/assignment-3 git|master >>> python server.py
path: /en/index.html
domain: www.uni-koblenz.de
subdomain: None
port: 80
query: webscience=1
protocol: http
fragment: head
```

Listing 3 Client Side

```
1: #!/usr/bin/env python
2: # pylint: disable-msg=C0103
3: """
4:     Simple python program which asks for
5:     url whose information should be processed.
6: """
7: import socket
8:
9: BUFFER_SIZE = 1024
10: HOST = '127.0.0.1'
11: PORT = 8080
12:
13: print("Enter web url")
14: url = input()
15: s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
16: s.connect((HOST, PORT))
17: s.send(url.encode('utf-8'))
18: response = s.recv(BUFFER_SIZE)
19: s.close()
```

```
~/d/a/w/u/assignment-3 git|master >>> python client.py
Enter web url
http://www.google.de:80/en/?hello=world
```


3 DNS Recursive Query Resolving (5 Points)

You have solved the "Routing Table" question in Assignment 2. We updated the routing tables once more, resulting in the following tables creating the following topology

Table 1: Routing Table

Router1			Router2			Router3		
Destination	Next Hop	Interface	Destination	Next Hop	Interface	Destination	Next Hop	Interface
67.0.0.0	67.68.3.1	eth 0	205.30.7.0	205.30.7.1	eth 0	205.30.7.0	205.30.7.2	eth 0
62.0.0.0	62.4.31.7	eth 1	156.3.0.0	156.3.0.6	eth 1	88.0.0.0	88.6.32.1	eth 1
88.0.0.0	88.4.32.6	eth 2	26.0.0.0	26.3.2.1	eth 2	25.0.0.0	25.03.1.2	eth 2
141.71.0.0	141.71.20.1	eth 3	141.71.0.0	141.71.26.3	eth 3	121.0.0.0	121.0.3.1	eth 3
26.0.0.0	141.71.26.3	eth3	67.0.0.0	141.71.20.1	eth 3	156.3.0.0	205.30.7.1	eth 0
156.3.0.0	88.6.32.1	eth 2	62.0.0.0	141.71.20.1	eth 3	26.0.0.0	205.30.7.1	eth 0
205.30.7.0	141.71.26.3	eth 3	88.0.0.0	141.71.20.1	eth 3	141.71.0.0	205.30.7.1	eth 0
25.0.0.0	88.6.32.1	eth 2	25.0.0.0	205.30.7.2	eth 0	67.0.0.0	88.4.32.6	eth 1
121.0.0.0	88.6.32.1	eth 2	121.0.0.0	205.30.7.2	eth 0	62.0.0.0	88.4.32.6	eth 1

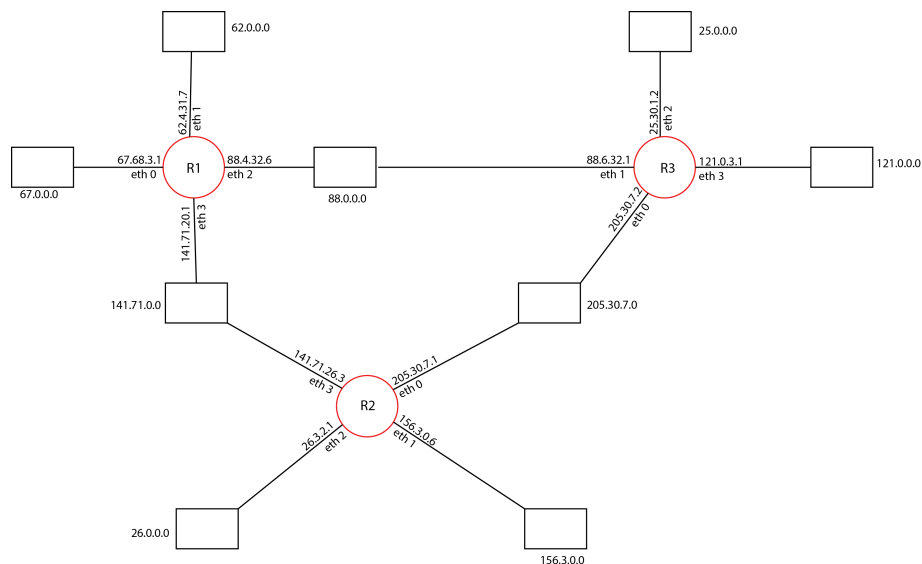


Figure 1: DNS Routing Network

Let us assume a client with the following ip address 67.4.5.2 wants to resolve the following domain `subdomain.webscienceexampdomain.com` using the DNS.

You can further assume the root name server has the IP address of 25.8.2.1 and the name-server for `webscienceexampdomain.com` has the IP address 156.3.20.2. Finally the sub-domain is handled by a name server with the IP of 26.155.36.7.

Please explain how the traffic flows through the network in order to resolve the recursive DNS query. You can assume ARP tables are cached so that no ARP-requests have to be made.

Hint: You can start like this:

67.4.5.2 creates an IP packet with the source address XXXXXX an destination address YYYYYY inside there is the DNS request. This IP packet is send as an ethernet frame to ZZZZZ. ZZZZZ receives the frame and forwards the encapsulated IP packet to

Also you can assume the DNS requests and responses will fit inside one IP packet. You also don't have to write down the specific DNS requests and responses in hex.

Answer:

1. 67.4.5.2 creates an IP packet with the source address 67.4.5.2(itself) an destination address 67.68.3.1 inside there is the DNS request. This IP packet is sent as an ethernet frame to router R1 via eth0 whose IP address is 67.68.3.1 In our case let us assume R1 is the ISP(Internet Service Provider).
2. Router R1 sends packet to R3 using eth2 whose IP address is 88.4.32.6
3. R3 receives the packet via 88.0.0.0 network using eth1 whose IP address is 88.6.32.1. Now it sends the packet to the 25.8.2.1 (i.e root name server) via eth2 whose IP address is 25.30.1.7
4. The root name server sees the DNS request and returns the response to router R3 via same port by adding the information to find out the domain name server.
5. Then, router R3 receives the response from root name server and sends it back to router R1(i.e ISP in our case) via same path from where it came.
6. Router R1 receives the response and make the request again. It sends the request to router R2 via 141.71.0.0 network using eth3 whose IP address is 141.71.20.1
7. Router R2 receives the request via eth3 whose IP address is 141.71.26.3. After that it forwards the request to 156.3.20.2 (name serve) using eth1 whose IP address is 156.3.0.6
8. Now the name server(156.3.20.2) sees the DNS request and returns the response to router R2 via same port by adding the information to find out the sub- domain.
9. The router R2 sends the response to router R1 via same path from where it came before.
10. Again, the router R1 receives the response and makes a request to get sub- domain name via same previous path to router R2 and R2 receives request via same previous port.
11. The router R2 sends the packet to the sub-domain(26.155.36.7) via eth2 whose IP address is 26.3.2.1
12. Now, the sub-domain gives the IP address of subdomain.webscienceexampledomain.com. After that it forwards the response to router R2 via same port. And R2

sends back to R1 via same path.

13. Router R1 receives the response via eth3 whose IP address is 141.71.20.1 and it forwards via eth0 whose IP address is 67.68.3.1 to the client 67.4.5.2
14. Finally, the client gets the response with IP address of subdomain.webscienceexampledomain.com

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment3/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use UTF-8 as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
 - Make sure you code has consistent [indentation](#).
 - Make sure you comment and document your code adequately in English.
 - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of "Web Information Retrieval".

\LaTeX

Currently the code can only be build using [LuaLaTeX](#), so make sure you have that installed. If on Overleaf, there's an error, go to settings and change the \LaTeX engine to LuaLaTeX.