

# Introduction to Web Science

## Assignment 3

Prof. Dr. Steffen Staab

[staab@uni-koblenz.de](mailto:staab@uni-koblenz.de)

René Pickhardt

[rpickhardt@uni-koblenz.de](mailto:rpickhardt@uni-koblenz.de)

Korok Sengupta

[koroksengupta@uni-koblenz.de](mailto: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.

Team Name: Tango

1. Mariya Chkalova  
[mchkalova@uni-koblenz.de](mailto:mchkalova@uni-koblenz.de)
2. Arsenii Smyrnov  
[smyrnov@uni-koblenz.de](mailto:smyrnov@uni-koblenz.de)
3. Simon Schauß  
[sschauss@uni-koblenz.de](mailto:sschauss@uni-koblenz.de)

## 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](http://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. 141.26.200.8
2. The SOA is an abbreviation for *State Of Authority* and defines how to handle zone transfers, which synchronize data in a DNS name server cluster.
3.
  - a) `uni-koblenz-landau.de.` is the owner name (in this case the parent domain of `www.uni-koblenz-landau.de.`), the dot at the end indicates a FQDN (fully qualified domain name)
  - b) 3600 is the duration a resolver may cache the record
  - c) IN the class of the record, Internet in this case
  - d) `dnsvw01.uni-koblenz-landau.de.` is the FQDN of the name server which is authoritatively responsible for the queried domain
  - e) `root.dnsvw01.uni-koblenz-landau.de.` is the email address to be used for reporting errors. As the @ has another purpose in the zone file, the @ is replaced by a dot. Therefore `root.dnsvw01.uni-koblenz-landau.de` is `root@dnsvw01.uni-koblenz-landau.de`
  - f) 2016110401 Is the serial number which gets increased every time the zone file is updated. Here the serial number consists of 10 digits, which indicates that the BIND implementation is used. A convention is to use date based serial numbers followed by a sequential two digit number. Therefore this zone file has last been updated once on the fourth of November in 2016.
  - g) 14400 Is the refresh rate in seconds a slave DNS server will try to refresh the zone file with the one from the master. 4 hours in this case.
  - h) 900 Defines how long (15 minutes) a slave DNS server should wait to retry a refresh if it fails to reach its master.

- i) 604800 Is the time (1 week) the zone file is authoritative. A slave server doesn't respond to queries if the zone file is expired.
- j) 14400 Is the negative caching time (4 hours). This negative caching time defines how long an unresolved name query is answered with a negative answer from the cache.

As seen in 1 and 2 there is now difference in the SOA records.

```
x schauboga@schauboga ~ dig SOA +multiline www.uni-koblenz-landau.de

; <<>> DiG 9.11.0-P1 <<>> SOA +multiline www.uni-koblenz-landau.de
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 57756
;; 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. 8678 IN SOA dnsvw01.uni-koblenz-landau.de. root.dnsvw01.uni-koblenz-landau.de. (
                                2016110401 ; serial
                                14400      ; refresh (4 hours)
                                900        ; retry (15 minutes)
                                604800     ; expire (1 week)
                                14400     ; minimum (4 hours)
                                )

;; Query time: 1009 msec
;; SERVER: 141.26.64.2#53(141.26.64.2)
;; WHEN: Tue Nov 15 13:14:07 UTC 2016
;; MSG SIZE rcvd: 103
```

Figure 1: dig SOA @ university

```
schauboga@schauboga ~$ dig SOA +multiline www.uni-koblenz-landau.de

; <<>> DiG 9.11.0-P1 <<>> SOA +multiline www.uni-koblenz-landau.de
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62813
; 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. 3542 IN SOA dnsvw01.uni-koblenz-landau.de. root.dnsvw01.uni-koblenz-landau.de. (
    2016110401 ; serial
    14400      ; refresh (4 hours)
    900        ; retry (15 minutes)
    604800     ; expire (1 week)
    14400      ; minimum (4 hours)
)

; Query time: 18 msec
; SERVER: 192.168.2.1#53(192.168.2.1)
; WHEN: Sat Nov 12 11:40:00 UTC 2016
; MSG SIZE rcvd: 103
```

Figure 2: dig SOA @ home

## 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`

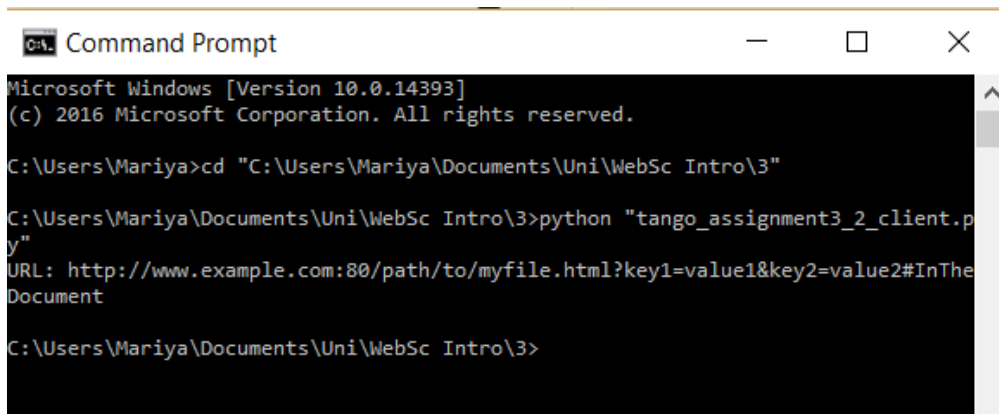
1. Protocol
2. Domain
3. Sub-Domain
4. Port number
5. Path
6. Parameters
7. Fragment

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:**

```
1:
2: # coding: utf-8
3:
4: # In[ ]:
5:
6: #!/usr/bin/python
7:
8: import socket
9:
10: IP = '127.0.0.1'
11: PORT = 8080
12: BUFFER_SIZE = 1
13:
14: client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15: client_socket.connect((IP, PORT))
16: try:
17:     url = input('URL: ')+'\r\n'
18:     data=url.encode('UTF-8')
19:     client_socket.sendall(data)
20: finally:
21:     client_socket.close()
```



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Mariya>cd "C:\Users\Mariya\Documents\Uni\WebSc Intro\3"

C:\Users\Mariya\Documents\Uni\WebSc Intro\3>python "tango_assignment3_2_client.py"
URL: http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#InTheDocument

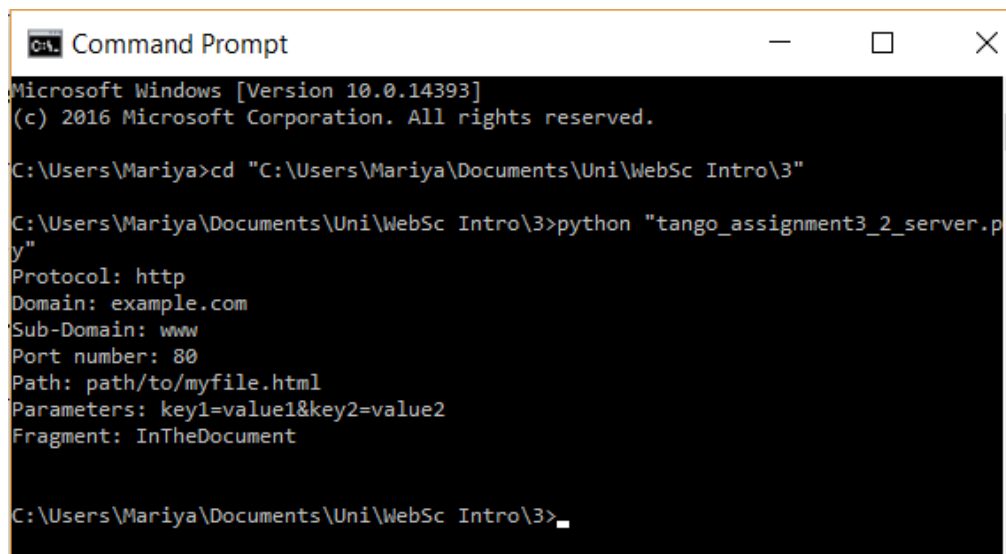
C:\Users\Mariya\Documents\Uni\WebSc Intro\3>
```

Figure 3: TCP client in action

```
1:
2: # coding: utf-8
3:
4: # In[ ]:
5:
6: #!/usr/bin/python
7: import socket
8:
9: IP = '127.0.0.1'
10: PORT = 8080
11: BUFFER_SIZE = 1
12:
13: server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14: server_socket.bind((IP, PORT))
15: server_socket.listen(1)
16: connection, address = server_socket.accept()
17: data = ''
18: def parseUrl (url):
19:
20:     protocol_rest=url.split('://')
21:
22:     dict_url={}
23:     dict_url['protocol']=protocol_rest[0]
24:     host_rest=protocol_rest[1].split('/',1)
25:
26:     port='Undefined'
27:     domains=host_rest[0]
28:     if ':' in domains:
29:         port_split=domains.split(':')
30:         domains=port_split[0]
31:         port=port_split[1]
32:     dict_url['port']=port
33:     domain_split=domains.split('.')

```

```
34:     subdomain='Undefined'
35:     domain=''
36:
37:     if (len(domain_split)!=2):
38:         domain_parts=domains.split('.',1)
39:         subdomain=domain_parts[0]
40:         domain=domain_parts[1]
41:     else:
42:         domain=domains
43:     dict_url['subdomain']=subdomain
44:     dict_url['domain']=domain
45:
46:     path_rest=host_rest[1]
47:     fragment='Undefined'
48:     if '#' in path_rest:
49:         rest_fragment=path_rest.split('#')
50:         path_rest=rest_fragment[0]
51:         fragment=rest_fragment[1]
52:     dict_url['fragment']=fragment
53:
54:     path=''
55:     parameters='Undefined'
56:     if '?' in path_rest:
57:         path_parameters=path_rest.split('?')
58:         path=path_parameters[0]
59:         parameters=path_parameters[1]
60:     else:
61:         path=path_rest
62:     dict_url['path']=path
63:     dict_url['parameters']=parameters
64:     return dict_url
65:
66: try:
67:     while True:
68:         buffer = connection.recv(BUFFER_SIZE)
69:         data += buffer.decode('UTF-8')
70:         if not buffer:
71:             dict = parseUrl(data)
72:             print('Protocol: %s' % dict['protocol'])
73:             print('Domain: %s' % dict['domain'])
74:             print('Sub-Domain: %s' % dict['subdomain'])
75:             print('Port number: %s' % dict['port'])
76:             print('Path: %s' % dict['path'])
77:             print('Parameters: %s' % dict['parameters'])
78:             print('Fragment: %s' % dict['fragment'])
79:             break
80: finally:
81:     server_socket.close()
```



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Mariya>cd "C:\Users\Mariya\Documents\Uni\WebSc Intro\3"

C:\Users\Mariya\Documents\Uni\WebSc Intro\3>python "tango_assignment3_2_server.py"
Protocol: http
Domain: example.com
Sub-Domain: www
Port number: 80
Path: path/to/myfile.html
Parameters: key1=value1&key2=value2
Fragment: InTheDocument

C:\Users\Mariya\Documents\Uni\WebSc Intro\3>
```

**Figure 4:** TCP server in action

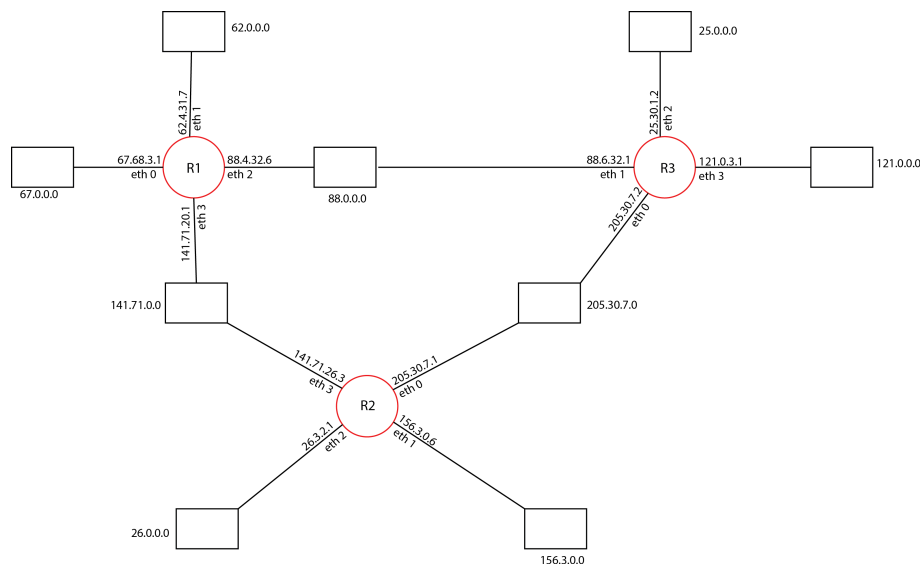


### 3 DNS Recursive Query Resolving (5 Points)

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 5: 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 and destination address YYYYYY inside there is the DNS request. This IP packet is sent 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 ip-packet. Source address – 67.4.5.2; destination address – 67.68.3.1 (R1 – ISP).
2. Ip-packet is sent to R1 over eth0 (67.68.3.1).
3. R1 sends the request to R3 over eth2 (88.4.32.6).
4. R3 (88.6.32.1) receives the request and sends it to the 25.8.2.1 (root name server) over eth2 (25.30.1.7).
5. R3 receives the response from root name server (doesn't have subdomain.webscience-exampldomain.com) and sends it to R1.
6. R1 sends the request to R2 over eth3 (141.71.20.1).
7. R2 receives the request and sends it over eth0 (156.3.0.6) to 156.3.20.2 (webscienceex-ampdomain.com name server).
8. Name server (156.3.20.2) returns the response to R2.
9. R2 sends the response to R1.
10. R1 sends packet to the 26.155.36.7 (sub-domain) over eth2 (26.3.2.1).
11. Sub-domain sends the IP-address of subdomain.webscienceexampdomain.com.
12. R1 receives the response at eth3 (141.71.20.1) and sends it to 67.4.5.2 over eth0 (67.68.3.1).
13. 67.4.5.2 gets IP-address of subdomain.webscienceexampdomain.com.
14. IP-address of subdomain.webscienceexampdomain.com is cached by ISP.

## 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".

### **L**A<sub>T</sub>E<sub>X</sub>

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 **L**A<sub>T</sub>E<sub>X</sub>engine to **LuaLaTeX**.