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 Koblenz

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: Bravo - Shriharsh Ambhore, Kandhasamy Rajasekaran, Daniel Akbari """

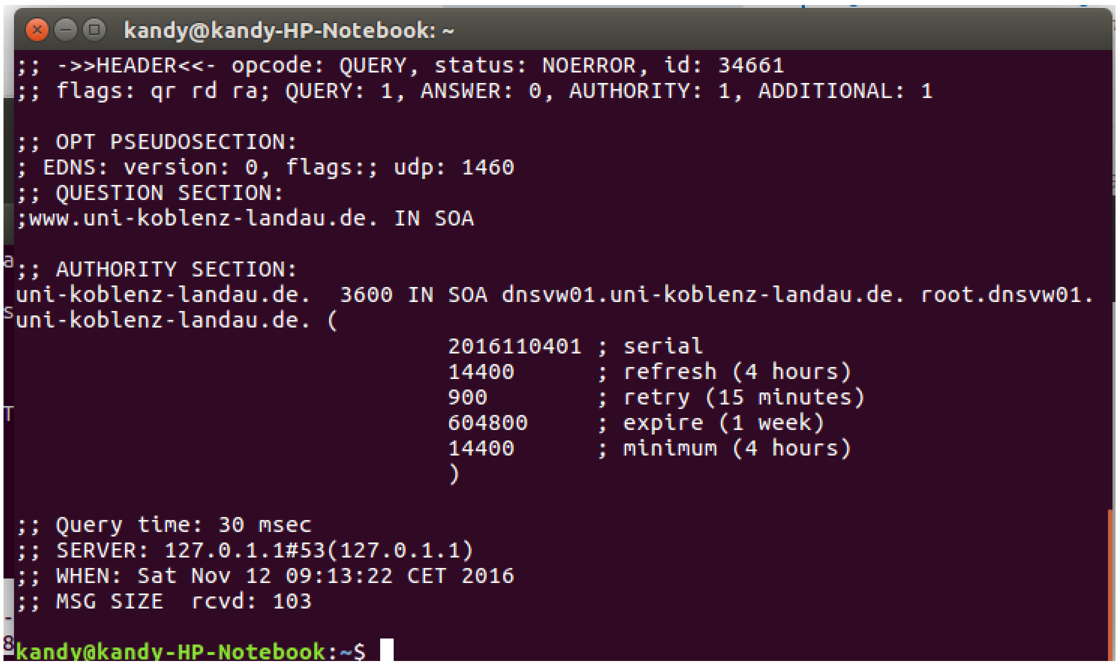
1 DIG Deeper (5 Points)

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 wont 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:

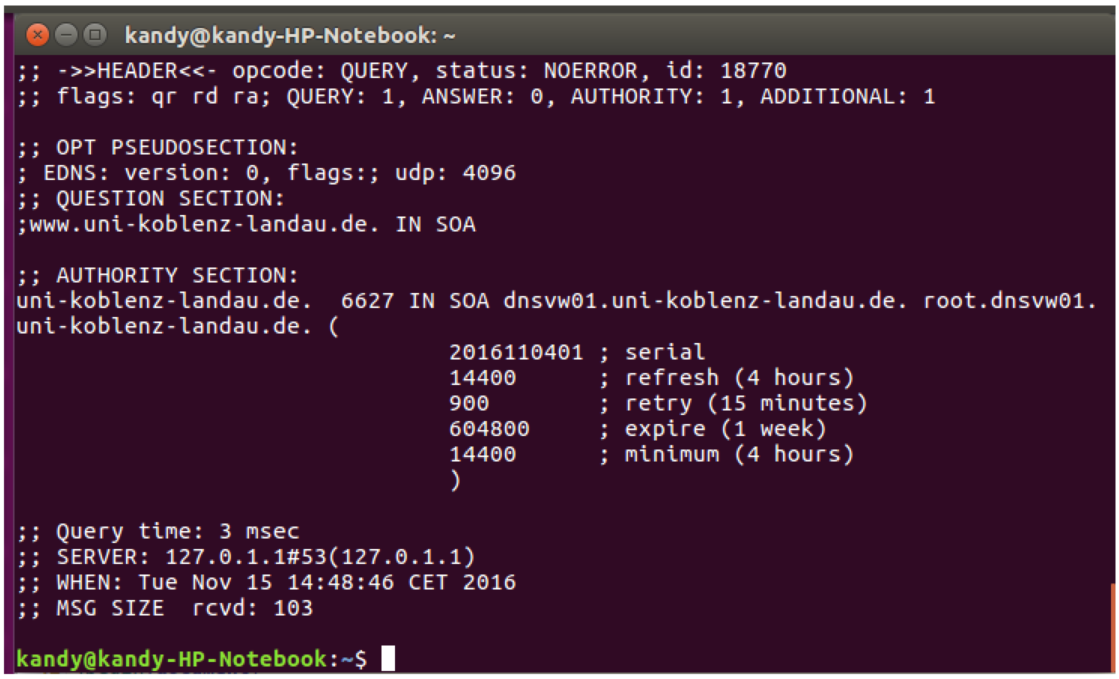


```
kandy@kandy-HP-Notebook: ~
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34661
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1460
;; QUESTION SECTION:
;www.uni-koblenz-landau.de. IN SOA

;; AUTHORITY SECTION:
uni-koblenz-landau.de.  3600 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: 30 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Sat Nov 12 09:13:22 CET 2016
;; MSG SIZE  rcvd: 103

kandy@kandy-HP-Notebook:~$
```

SOA - start of authority - it refers to basic properties of a particular domain name system such as
system which is created, email address who is responsible for maintain the zone file, revision no. to track changes, refresh time, retry time, retry expire time and time to live This is necessary for secondary name servers to refer to and mainly update their records

In the above example
dnsvw01.uni-koblenz-landau.de. - is the host where it is created and maintained
root@dnsvw01.uni-koblenz-landau.de - email address of person/organization who is re-

sponsible for maintaining the zone records
2016110401 - everytime when a change is done this number is increment by 1. (Wow !!! That is a lot of change)
14400 seconds (4 hrs) - refresh time. secondary name servers will update their record within this time
900 seconds - when failed updating, in what interval time the secondary name servers can try
604800 seconds - when the secondary name servers are not updated for this much time then they are bound to expire after that
14400 seconds - similar to refresh time. It is referred to as time to live, but this is sent in response along with zone file which is used by secondary servers to use it as a cache



```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18770
;; 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.  6627 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: 3 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Tue Nov 15 14:48:46 CET 2016
;; MSG SIZE  rcvd: 103

kandy@kandy-HP-Notebook:~$
```

Results from uni

I did not find any difference in the soa details between acessing from home and uni

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.

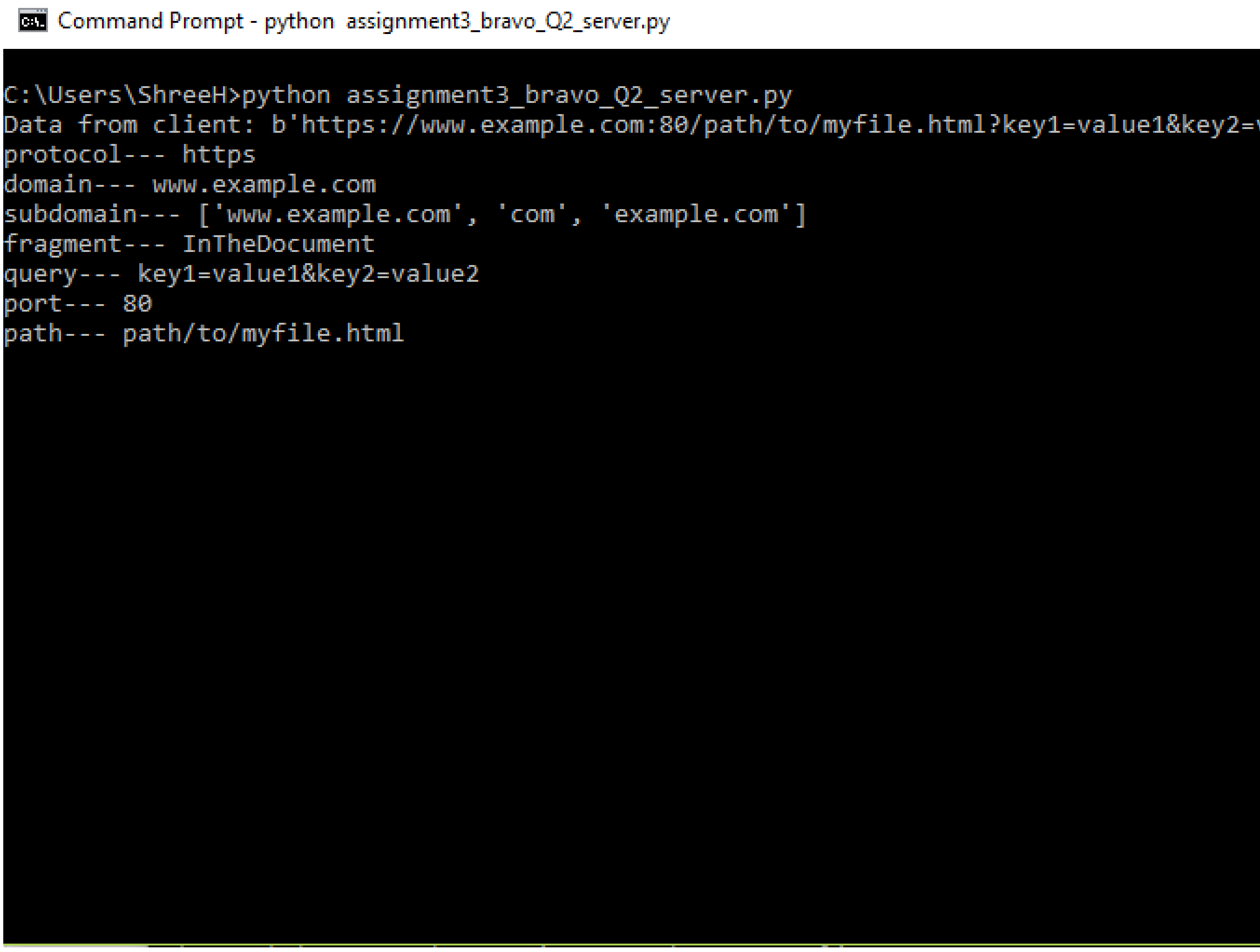
P.S.: You are **not** allowed to use libraries like `urlparse` for this question. You will also not use "Regular Expressions" for this.

Answer:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 10 19:13:29 2016

@author: Shriharsh Ambhore
@author: Kandhasamy Rajasekaran
@author: Daniel Akbari

"""



import socket


s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('localhost', 8080))  ## connect to the server and send the data
s.send('https://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#In
s.close()
```

```
# -*- coding: utf-8 -*-
"""
```

```
server side code

Created on Mon Nov  7 03:36:00 2016

@author: Shriharsh Ambhore
@author: Kandhasamy Rajasekaran
@author: Daniel Akbari


"""

import socket
import sys



server_socket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)


try:
    server_socket.bind(('localhost',8080))
except socket.error as msg:
    print ('Bind to socket failed.'+' Message ' + msg[1])
    sys.exit()

server_socket.listen(1)


while True:
    conn,address=server_socket.accept()
    dataFromClient=conn.recv(4096)
    url=dataFromClient.decode()
    print('Data from client:',dataFromClient) # data received from client
    ## at any point of time  characters  :?#/[]@  are reserved as delimeters

    pos=url.find(":")
    subdomainlist = []
    if pos>0:
        tempvar=url[:pos]
        if url[pos:].find("//")> 0: # if the url contains the protocol then execu
https://www.google.com
            protocol=tempvar #
            print("protocol---",protocol)
            url=url[pos+3:] ## ignoring : and //
        if ":" in url: ## check for : to get the domain
            domain,url=url.split(":")
            print("domain---",domain)
            #subdomain= domain[:(domain.find(".",1))]

            subdomainlist.append(domain)
            gblsd = ""

            while domain.find(".")!=-1:  # find . in the domain
                remainingUrl,sd=domain.rsplit(".",1)    # split from right hand si
                sd="."+sd+gblsd
                subdomainlist.append(sd.lstrip("."))    #add valid string to the l
                gblsd=sd                                # change the string from w
                domain=remainingUrl

            print("subdomain---",subdomainlist)
        if "#" in url:  ## split on # to get the fragment
            url,anchor=url.split("#",1)
            print("fragment---",anchor)
        if "?" in url: ## split on ? to get the query
            url,query=url.split("?",1)
            print("query---",query)
        if "/" in url:   ## split on / to get the port and the path
            port,path=url.split("/",1)
            if port.isdigit():
                print("port---",port)
            else:
                print("domain",port)
                subdomain= port[:(port.rfind(".",2))]
                print("subdomain---",subdomain)
            print("path---",path)

    else:       # if the url does not contain any protocol specified i.e www.googl
        domain=tempvar
        print("domain:",domain)
        if ":" in url:
            domain,url=url.split(":")
            print("domain---",domain)
            #subdomain= domain[:(domain.rfind(".",1))]
            gblsd = ""

            while domain.find(".")!=-1:
                remainingUrl,sd=domain.rsplit(".",1)
                sd="."+sd+gblsd
                subdomainlist.append(sd.lstrip("."))
                gblsd=sd
                domain=remainingUrl

            print("subdomain---",subdomainlist)

        if "#" in url:
            url,anchor=url.split("#",1)
            print("fragment---",anchor)
        if "?" in url:
            url,query=url.split("?",1)
            print("query---",query)
        if "/" in url:
            port,path=url.split("/",1)
            print("port---",port)
            print("path---",path)


    conn.close()
```



Command Prompt - python assignment3_bravo_Q2_server.py

```
C:\Users\ShreeH>python assignment3_bravo_Q2_server.py
Data from client: b'https://www.example.com:80/path/to/myfile.html?key1=value1&key2=
protocol--- https
domain--- www.example.com
subdomain--- ['www.example.com', 'com', 'example.com']
fragment--- InTheDocument
query--- key1=value1&key2=value2
port--- 80
path--- path/to/myfile.html
```

Figure 1: Server output

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

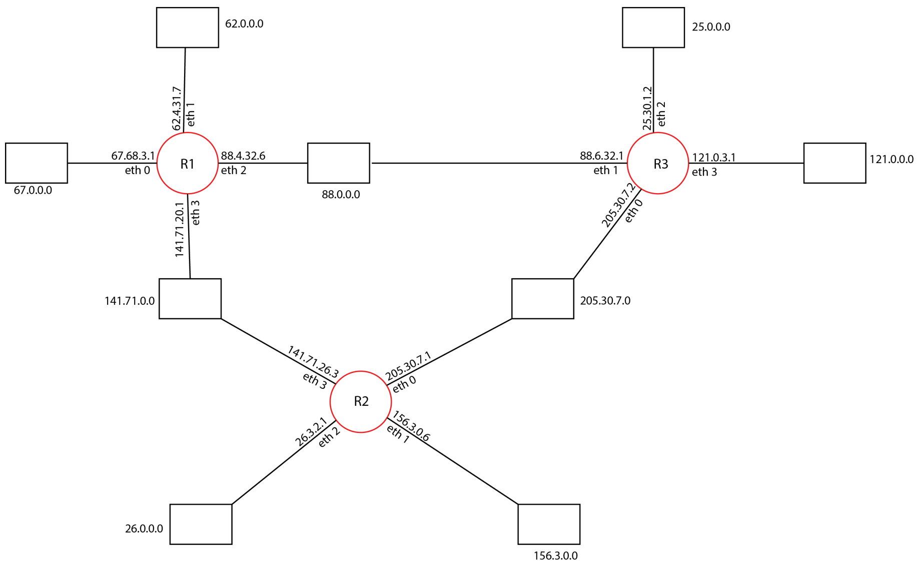


Figure 2: DNS Routing Network

Let us asume a client with the following ip address 67.4.5.2 wants to resolve the following domain `subdomain.webscienceexampledomain.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 `webscienceexampledomain.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. 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.

Hint: You can start like this:

Answer:
Assumption:
We consider the router R1 is the DNS name server for the given client. And it will take care of the recursive DNS resolving

67.4.5.2 creates an IP packet with the source address 67.4.5.2 and destination address 67.68.3.1 inside the payload, we have DNS request for ”subdomain.webscienceexampledomain.com”. This IP packet is send as an ethernet frame to Router1,which is the DNS name server in this case. This name server will have zone file which contains the address for ”root” name server which is 25.8.2.1.

Router1 forwards the encapsulated IP packet with address 25.8.2.1 to 88.0.0.0 through eth2 interface and then to Router3. From Router3 it is directed to 25.8.2.1. This name server will answer back to the DNS name server 67.68.3.1 saying that 156.3.20.2 might be able to resolve it.(156.3.20.2 is related to webscienceexampledomain.com)

Router1 then forwards the encapsulated IP packet with address 156.3.20.2 to 141.71.0.0 through eth3 interface and then to Router2. From Router2 it is directed to 156.3.0.0 through eth1. This name server 156.3.20.2 will answer back to the DNS name server 67.68.3.1 saying that 26.155.36.7 might be able to resolve it.(26.155.36.7 is related to subdomain.webscienceexampledomain.com)

Router1 then forwards the encapsulated IP packet with address 26.155.36.7 to 141.71.0.0 through eth3 interface and then to Router2. From Router2 it is directed to 26.0.0.0 through eth2. This name server 26.155.36.7 will answer the ip address of the given sub-domain to the DNS name server 67.68.3.1

Then the final address will be returned to the client which is 67.4.5.2 by 67.68.3.1

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 LaTeXengine to `LuaLaTeX`.