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.

Team Name: **papa** Brigitte Aznar
Bonasmitha Behura
Ilia Tugushi

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:

1. The IP address is 141.26.200.8

```
; <<>> DiG 9.9.5-3ubuntu0.8-Ubuntu <<>> www.uni-koblenz-landau.de
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53152
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 6, ADDITIONAL: 11

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.uni-koblenz-landau.de.     IN      A

;; ANSWER SECTION:
www.uni-koblenz-landau.de. 4200 IN      A       141.26.200.8
```

Figure 1: Dig IP address

2. SOA stands for Start of Authority, contains important information about the management of the zone, in particular zone transfer. It indicates the server(s) that are the ultimate authority for answering DNS queries about that domain.
   It includes
   - The root(name) of the zone: This specifies that the zone file is for the **www.uni-koblenz-landau.de.** domain.
   - Class zone: IN (stands for Internet)
   - SOA: indicator that this is a Start of Authority record.

- The primary master name server for this domain: **dnsvw01.uni-koblenz-landau.de.**
  Name servers can either be master or slaves, and if dynamic DNS is configured one server needs to be a "primary master", which goes here. If you haven't configured dynamic DNS, then this is just one of your master name servers.
- Email address of the administrator for this zone: **root.dnsvw01.uni-koblenz-landau.de.**
  The "@" is replaced with a dot in the email address. If the name portion of the email address normally has a dot in it, this is replace with a "\" in this part (your.name@domain.com becomes your\name.domain.com).
  In this case the email would be root.dnsvw01@uni-koblenz-landau.de
- Serial number for the zone file: **2016110401**
  Every time you edit a zone file, you must increment this number for the zone file to propagate correctly. Slave servers will check if the master server's serial number for a zone is larger than the one they have on their system. If it is, it requests the new zone file, if not, it continues serving the original file.
- Refresh interval for the zone: **14400(4 hours)**
  This is the amount of time that the slave will wait before polling the master for zone file changes.
- Retry interval for this zone: **900(15 minutes)**
  If the slave cannot connect to the master when the refresh period is up, it will wait this amount of time and retry to poll the master.
- Expiry period: **604800(1 week) 900**
  If a slave name server has not been able to contact the master for this amount of time, it no longer returns responses as an authoritative source for this zone.
- Cache error time: **14400(4 hours)**
  This is the amount of time that the name server will cache a name error if it cannot find the requested name in this file.

```
;; AUTHORITY SECTION:
uni-koblenz-landau.de.  1367 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)
                              )
```

Figure 2: SOA Results

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:

```
import socket
import json

def Main():

    socket_server = socket.socket()
    socket_server.bind(('localhost', 8080))

    socket_server.listen(1)
    conn, addr = socket_server.accept()
    print("Connection from: " + str(addr))
    data = conn.recv(1024)
    data = data.decode('utf-8')
    if not data:
        print('no data received')
        return

    def protocol(url):
        protocol = url.split("://")
        return protocol

    def domain(url):
```

```
23:         domain = url.split(":")
24:         domain_string = domain[0].strip("www.")
25:         if domain_string.count(".") == 1:
26:             parts  = domain_string.split('.')
27:             domain = parts[0] + "." +parts[1]
28:         elif domain_string.count(".") > 1:
29:             parts = domain_string.split('.')
30:             total = len(parts)
31:             domain = parts[total - 2] + "." + parts[total - 1]
32:         return domain
33:
34:     def subdomain(url, domain):
35:         subdomain = url.split(domain)
36:         subdomain = subdomain[0].strip("www.")
37:         return subdomain
38:
39:     def port(url):
40:         port = url.split(":")
41:         port = port[1]
42:         port = port.split("/")
43:         port = port[0]
44:         return port
45:
46:     def path(url, port):
47:         path = url.split(port)
48:         path = path[1].split("?")
49:         path = path[0]
50:         return path
51:
52:     def params(url, path):
53:         params = url.split(path)
54:         params = params[1]
55:         params = params.split("#")
56:         params = params[0]
57:         return params
58:
59:     def fragment(url):
60:         fragment = url.split("#")
61:         fragment = fragment[1]
62:         fragment = fragment.strip("\\r\\n")
63:         return fragment
64:
65:     protocol  = protocol(data)
66:     url       = protocol[1]
67:     protocol  = protocol[0]
68:     domain    = domain(url)
69:     subdomain = subdomain(url, domain)
70:     port      = port(url)
71:     path      = path(url, port)
```

```
72:     params     = params(url,path)
73:     fragment   = fragment(url)
74:
75:     print('Protocol: ', protocol)
76:     print('Subdomain: ', subdomain)
77:     print('Domain: ', domain)
78:     print('Port: ', port)
79:     print('Patha: ', path)
80:     print('Params: ', params)
81:     print('Fragments: ', fragment)
82:     conn.close()
83:
84:
85: if __name__ == '__main__':
86:     Main()
```

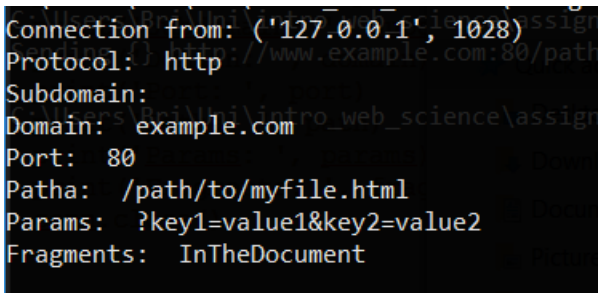

Figure 3: URL parts

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

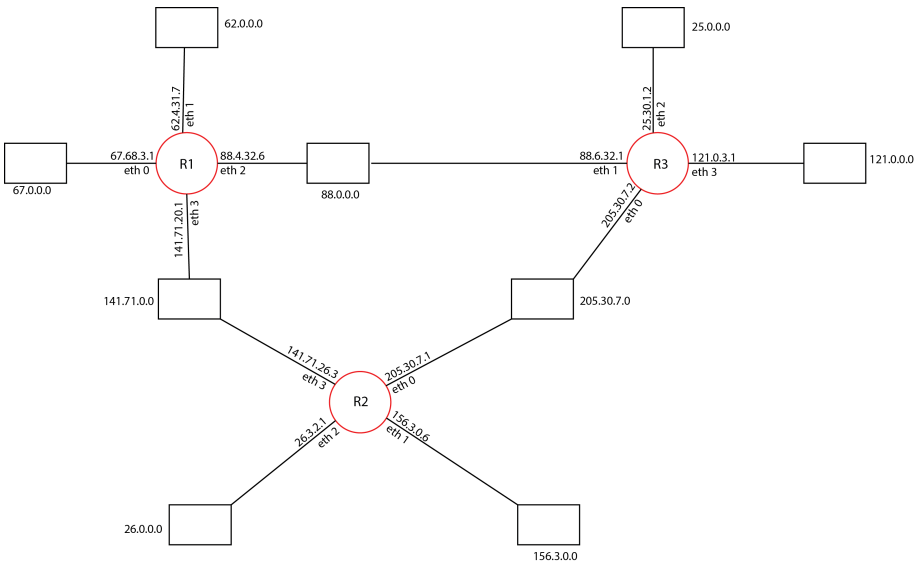

Figure 4: DNS Routing Network

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.

Hint: You can start like this:

67.4.5.2 creates an IP packet with the source address XXXXXX an destination address YYYYY 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.

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`.