Chair of Distributed Systems and Security
School of Computation, Information and Technology
Technical University of Munich

Computer Networking and IT Security (INHN0012)

Tutorial 12

Problem 1 Domain Name System (DNS)

The main task of the Domain Name System (DNS) is to map human-readable names to IP addresses, which can then be used to route packets at the network layer. The name `asciiart.grnvs.net.` is a so called *Fully Qualified Domain Name (FQDN)*.

a)* What is the difference between a fully qualified domain name (FQDN) and a non-(fully-) qualified one?

b)* Name the individual components of the FQDN, if common names exist for them.

```
asciiart.grnvs.net.
```

Figure 1.1 shows a PC and a set of servers. We assume that PC1 uses the router as a resolver. The router in turn uses a Google resolver at IP address 8.8.8.8 for name resolution. Furthermore, we assume that the Google resolver has just been restarted (i. e., in particular, it has not cached any resource records) and offers recursive name resolution.
The authoritative name servers for the respective zones are given in Table 1.1.

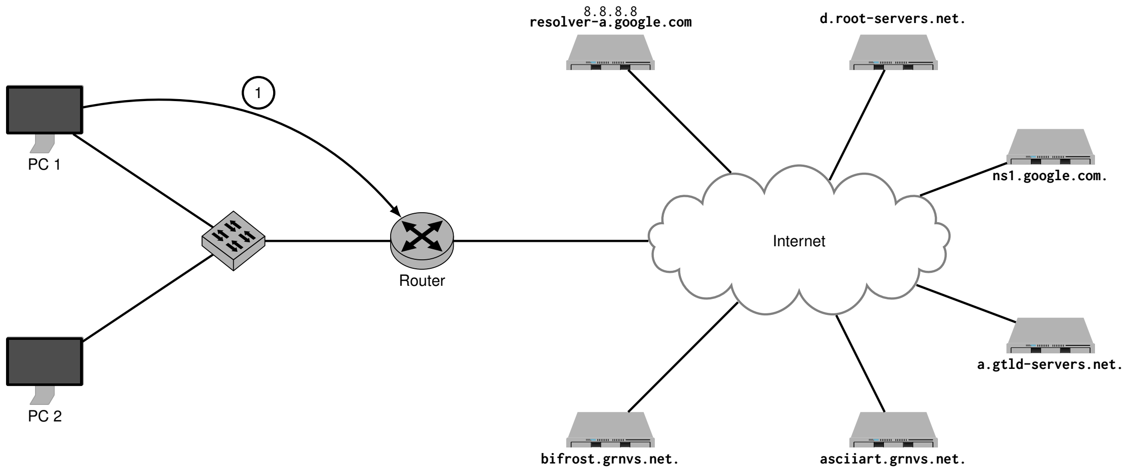


Figure 1.1: Preprint for Problem 1f)

Zone	Authoritative Name Server
.	d.root-servers.net.
com., net.	a.gtld-servers.net.
google.com.	ns1.google.com.
grnvs.net.	bifrost.grnvs.net.

Table 1.1: Zones with corresponding authoritative name servers

c)* Explain the difference between a *resolver* and a *name server*.

d)* What is the function of `d.root-servers.net` and `a.gtld-servers.net`?

e)* Explain the difference between iterative and recursive name resolution.

f) In Figure 1.1, draw all DNS messages (Requests / Responses) that will be exchanged once PC1 accesses `asciiart.grnvs.net.`. Number the messages according to the order in which they are exchanged between the nodes.

g)* How is it ensured in DNS that no malicious name server answers queries for other domains? (We assume that no man-in-the-middle attacks are possible).

Problem 2 All in a nutshell

In this task, we trace everything that happens when you access the `www.google.de` web page on your computer. We assume that the ARP and DNS caches in your private network are empty, while any caches from the first router onward can be assumed to be populated. The network topology is shown in Figure 2.1. Your router translates private IP addresses to public IP addresses as well as port numbers using NAT. On your computer, the Google resolver is configured with the IPv4 address 8.8.8.8, which allows recursive queries.

For **each link**, i. e., each section between two devices (e. g. from PC to SW), you should note some selected fields of the message headers as they are sent over the particular link. Since this involves a considerable amount of writing, especially for MAC addresses, we abbreviate addresses using `<device name>.<interface>.<type>`. For example, `RA.eth0.MAC` denotes the MAC address of interface `eth0` of router `RA`.
You can find pre-printed tables in Figures 2.2 — 2.5.
Each row corresponds to a message transmitted via the respective link. The first column therefore denotes the link, e. g. from the PC to the switch or from the switch to the router. The remaining columns represent the different layers of the ISO/OSI model. These are further subdivided into the relevant header fields of the commonly used protocols. Depending on the message, not all columns may be applicable. **Cross out unused fields.** An example is already provided in the table.

Some headers contain a protocol field that specifies the protocol of the next higher layer. Usually, numerical codes are used to represent these protocols. It is **not** necessary to specify these numerical codes. Instead, it is sufficient to indicate the protocol used, e. g. IPv4, TCP or UDP.
There is some flexibility in choosing certain header fields, e. g. for port numbers or the initial TTL. In these cases, choose **meaningful** values.

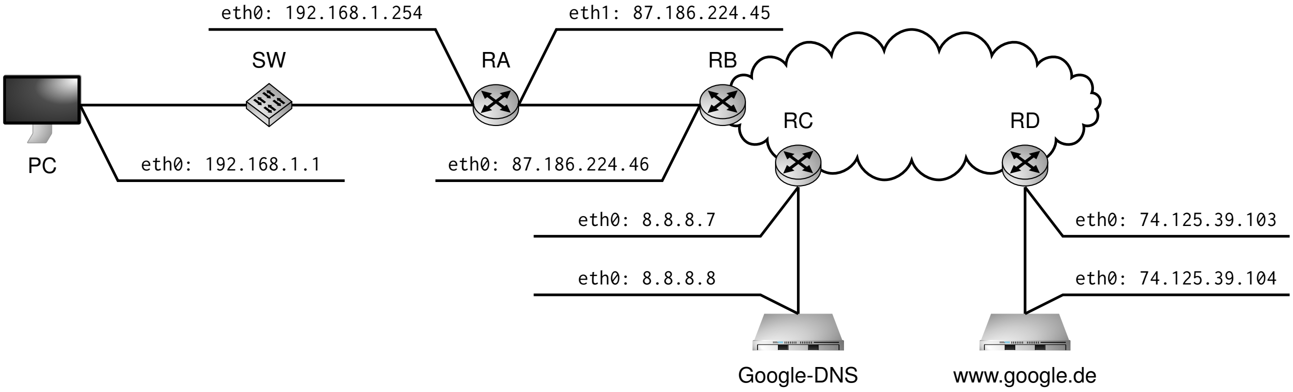


Figure 2.1: Network topology for Problem 2.

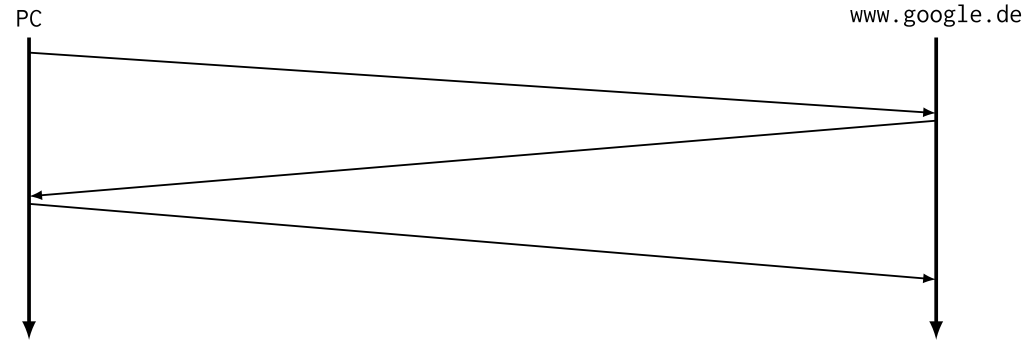
a)* Now fill in the forms in Figures 2.2 – 2.5. Stop after **the first** message transmitted over the link from PC to SW that is directed to `www.google.de`.

Notes:

- We assume that there are a total of 10 additional routers between router RB and RC. This information is required to determine the TTL.
- Enter the application-layer protocol by name in the respective column. If applicable, specify the message type (e. g. request or response) as well as the content of the message in descriptive form (e. g. "DNS request" or "DNS response").

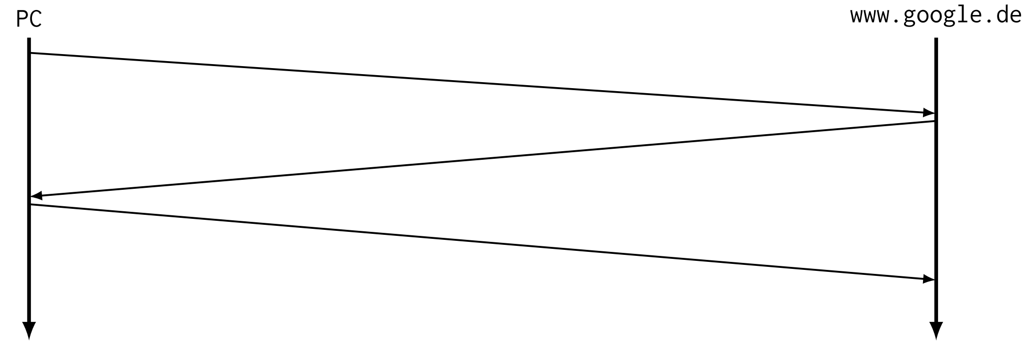
The previous subproblem has presented in detail the processes up to the beginning of the TCP connection construction. In the following, we focus on the TCP connection and data transmission. To this end, we now only consider the logical connection between the PC and `www.google.de` in the form of a simple path-time diagram **without** any intermediate nodes. Serialization time and propagation delay may be neglected. Furthermore, assume that no segment losses occur during the entire transmission.

b)* Sketch a path-time diagram representing the TCP connection setup. Specify the sequence number, acknowledgement number, the set flags, and the payload length *l* for each segment.

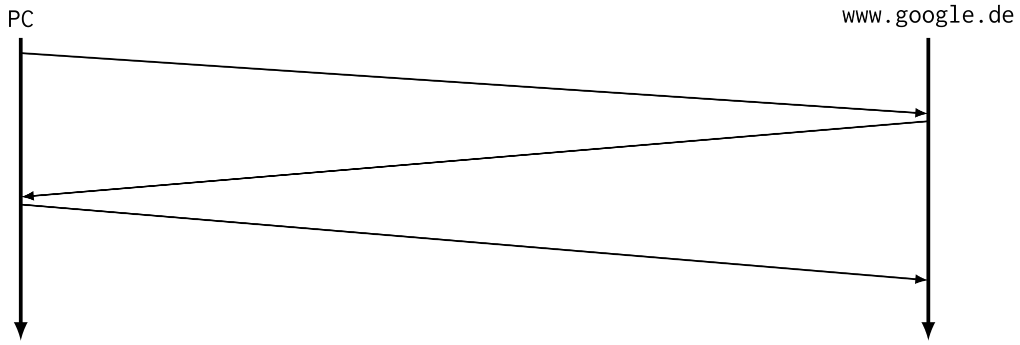


The PC now requests the website hosted at `www.google.de`. For this purpose, the PC sends a `HTTP GET` message, which has a length of $l_1 = 50\,\text{B}$ from the perspective of layer 4. The web server will then send the web page to the PC, which is assumed to be of length $l_2 = 1000\,\text{B}$. Let the negotiated MSS be larger than l_2.

c) Sketch a path-time-diagram which shows the TCP connection phase. Assume the sequence numbers negotiated in subproblem b). Specify the sequence number, acknowledgement number, the set flags, and the payload length *l* for each segment.



d) Sketch a path-time diagram representing TCP connection termination. Assume the PC initiates the teardown. Assume the sequence numbers negotiated in subproblem b). Specify the sequence number, acknowledgement number, the set flags, and the payload length *l* for each segment.



Link		Layer 2		Layer 3		Layer 4		Layer 7
From	PC	Src	PC.eth0.MAC	Src	▨	Src	▨	▨
		Dst	ff:ff:ff:ff:ff:ff	Dst	▨	Dst	▨	
To	SW	Prot	ARP	Prot	▨	Flags	▨	
		Op	Request	TTL	▨	SEQ	▨	
						ACK	▨	
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		

Figure 2.2: Pre-print for Problem 2

Link		Layer 2		Layer 3		Layer 4		Layer 7
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		

Figure 2.3: Pre-print for Problem 2

Link		Layer 2		Layer 3		Layer 4		Layer 7
From		Src		Src		Src		
		Dst		Dst		Dst		
						Flags		
To		Prot		Prot		SEQ		
				TTL		ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
						Flags		
To		Prot		Prot		SEQ		
				TTL		ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
						Flags		
To		Prot		Prot		SEQ		
				TTL		ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
						Flags		
To		Prot		Prot		SEQ		
				TTL		ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
						Flags		
To		Prot		Prot		SEQ		
				TTL		ACK		

Figure 2.4: Pre-print for Problem 2

Link		Layer 2		Layer 3		Layer 4		Layer 7
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		
From		Src		Src		Src		
		Dst		Dst		Dst		
To		Prot		Prot		Flags		
				TTL		SEQ		
						ACK		

Figure 2.5: Pre-print for Problem 2