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)

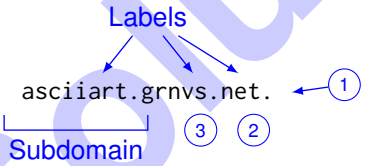
The main task of the Domain Name System (DNS) is to map human-readable names to IP addresses, which can then be used to route packets at the network layer. The name `asciiart.grnvs.net.` is a so called *Fully Qualified Domain Name (FQDN)*.

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

An FQDN always ends with ., i. e. the root of the name space. A non-qualified domain name, on the other hand, may consist of a single label or an ordered list of labels separated by dots and interpreted relative to a root other than ".".

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

1. Root (start of the namespace)
2. Top-Level Domain (TLD)
3. Second-Level Domain



Since in everyday life usually no explicit distinction is made between a „FQDN" (i. e. with terminating dot) and „Domain Name" (i. e. without terminating dot), since it is clear from the context which of the two is intended. Hence, we will also only set the root dot in the following subproblems if we want to emphasize or make this particularly clear.

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.

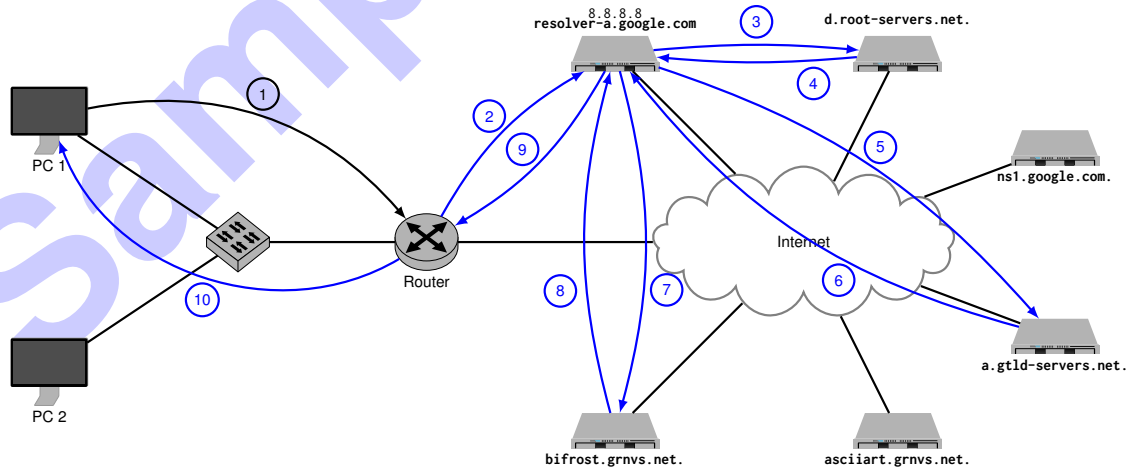


Figure 1.1: Preprint for Problem 1f)

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*.

Name servers are authoritative for one or more zones („areas“), i.e. they maintain a valid and up-to-date copy of the entire zone for which they are authoritative.
Resolvers, on the other hand, use a series of iterative queries to the respective authoritative name servers to extract the required information from the DNS and return it to the requesting client. Resolvers can cache records for a limited time so that the process does not have to be repeated when the same resource records are requested again.

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

The root name server is authoritative for the root zone, i. e. it knows the name servers responsible for the individual TLDs, e.g. `a.gtld-servers.net` as one of the authoritative name servers for net domains.
`a.gtld-servers.net` in turn knows the responsible name servers for all second-level domains below the net TLD.

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

Recursive name resolution means that a DNS query is made to a resolver. The latter will return the final result.
With iterative resolution, on the other hand, the authoritative name servers of the individual zones are requested step by step.

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).

This is only indirectly ensured by the fact that only the authoritative name servers are contacted during iterative name resolution. Provided that the

- response of the root server was reliable and
- the response was not modified on the way from the root server to the requesting name server

a malicious name server cannot provide incorrect answers — precisely because it is never asked.
Of course, this does not prevent DNS responses from being intercepted and modified by means of man-in-the-middle attacks. Only cryptographic procedures, such as those found in the DNSSEC extension (not covered in the lecture), can help against this.

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 — 2.4.
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.

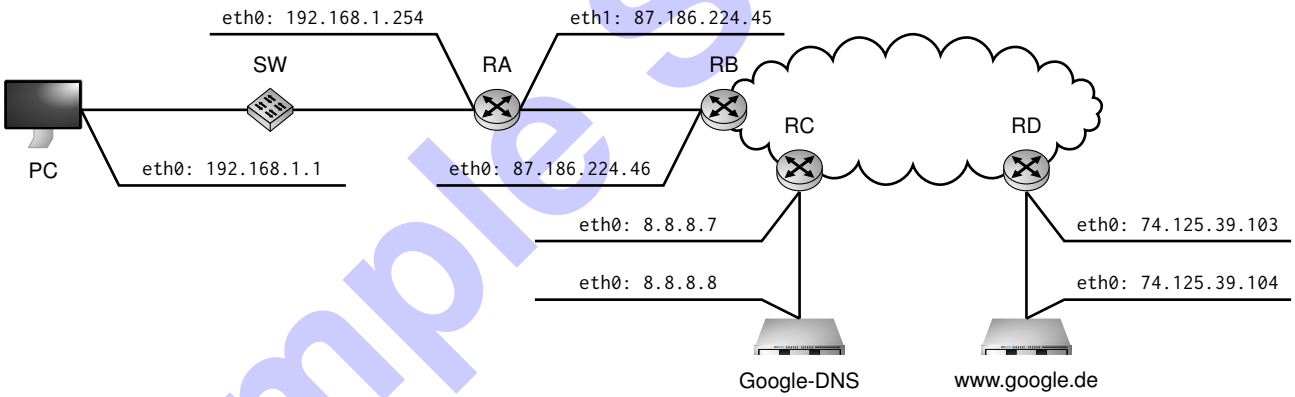


Figure 2.1: Network topology for Problem 2.

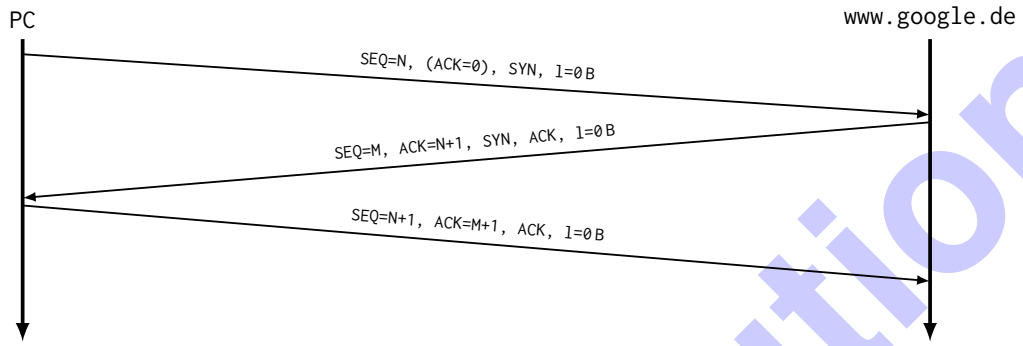
a)* Now fill in the forms in Figures 2.2 – 2.4. Stop after **the first** message transmitted over the link from PC to SW that is directed to `www.google.de`.

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.

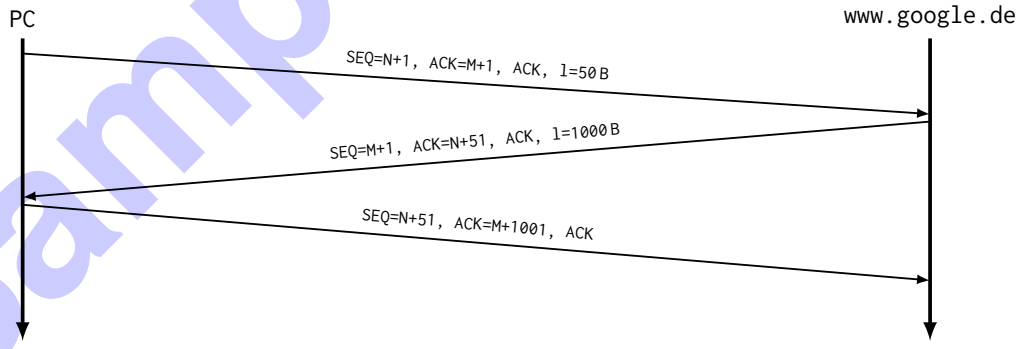
b)* Sketch a path-time diagram representing the TCP connection setup. Specify the sequence number, acknowledgement number, the set flags, and the payload length l for each segment.



The acknowledgement number in the first segment has no influence since the acknowledgement flag is not set (the field for the acknowledgement number is always present in the header). The initial sequence numbers for both sides are randomly chosen from $0 \leq N, M, \leq 2^{32} - 1$.

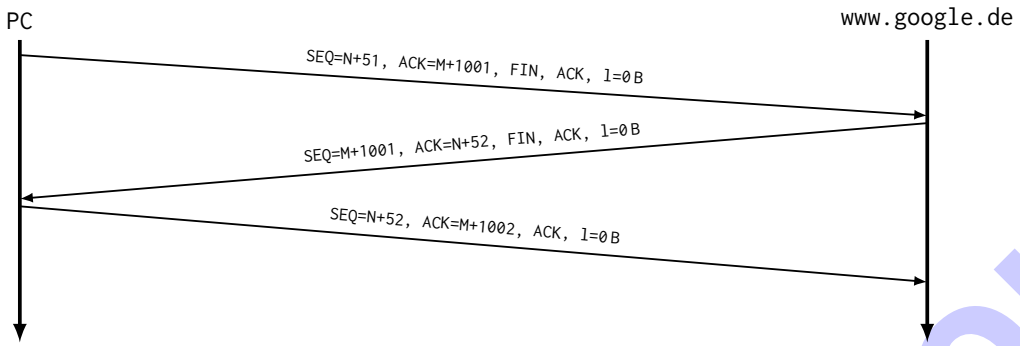
The PC now requests the website hosted at `www.google.de`. For this purpose, the PC sends a `HTTP GET` message, which has a length of $l_1 = 50\,\text{B}$ from the perspective of layer 4. The web server will then send the web page to the PC, which is assumed to be of length $l_2 = 1000\,\text{B}$. Let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



Note: If the PC wants to terminate the connection immediately after receiving the message, the third segment may have already set the FIN flag.

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.



From this and the previous two subtasks it can be seen once again that TCP confirms individual bytes, but not segments. Segments which have a SYN or FIN flag set are treated as segments with exactly 1 B payload.
The termination can also take the form of four messages instead of three, i. e., `www.google.de` first acknowledges receipt of the `FIN` flag without setting the `FIN` flag itself. This could be the case, for example, if the `PC` has no more data to send to `www.google.de`, but `www.google.de` still has data to send to the `PC`. A TCP connection in that state is called “half-open” as data transfer in one of the two directions is still possible.

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	SW	Src	PC.eth0.MAC	Src		Src		
		Dst	ff:ff:ff:ff:ff:ff	Dst		Dst		
To	RA	Prot	ARP	Prot		Flags		
		Op	Request	TTL		SEQ		
						ACK		
From	RA	Src	RA.eth0.MAC	Src		Src		
		Dst	PC.eth0.MAC	Dst		Dst		
To	SW	Prot	ARP	Prot		Flags		
		Op	Reply	TTL		SEQ		
						ACK		
From	SW	Src	RA.eth0.MAC	Src		Src		
		Dst	PC.eth0.MAC	Dst		Dst		
To	PC	Prot	ARP	Prot		Flags		
		Op	Reply	TTL		SEQ		
						ACK		
From	PC	Src	PC.eth0.MAC	Src	192.168.1.1	Src	51827	DNS Request
		Dst	RA.eth0.MAC	Dst	8.8.8.8	Dst	53	Who is www.google.de?
To	SW	Prot	IPv4	Prot	UDP	Flags		
				TTL	64	SEQ		
						ACK		

Figure 2.2: Pre-print for Problem 2

Link		Layer 2		Layer 3		Layer 4		Layer 7
From	SW	Src	PC.eth0.MAC	Src	192.168.1.1	Src	51827	DNS Request Who is www.google.de?
		Dst	RA.eth0.MAC	Dst	8.8.8.8	Dst	53	
To	RA	Prot	IPv4	Prot	UDP	Flags		
				TTL	64	SEQ		
						ACK		
From	RA	Src	RA.eth1.MAC	Src	87.186.224.45	Src	38218	DNS Request Who is www.google.de?
		Dst	RB.eth0.MAC	Dst	8.8.8.8	Dst	53	
To	RB	Prot	IPv4	Prot	UDP	Flags		
				TTL	63	SEQ		
						ACK		
From	RC	Src	RC.eth0.MAC	Src	87.186.224.45	Src	38218	DNS Request Who is www.google.de?
		Dst	DNS.eth0.MAC	Dst	8.8.8.8	Dst	53	
To	Google DNS	Prot	IPv4	Prot	UDP	Flags		
				TTL	51	SEQ		
						ACK		
From	Google DNS	Src	DNS.eth0.MAC	Src	8.8.8.8	Src	53	DNS Reply www.google.de is 74.125.39.104!
		Dst	RC.eth0.MAC	Dst	87.186.224.45	Dst	38218	
To	RC	Prot	IPv4	Prot	UDP	Flags		
				TTL	64	SEQ		
						ACK		
From	RB	Src	RB.eth0.MAC	Src	8.8.8.8	Src	53	DNS Reply www.google.de is 74.125.39.104!
		Dst	RA.eth1.MAC	Dst	87.186.224.45	Dst	38218	
To	RA	Prot	IPv4	Prot	UDP	Flags		
				TTL	52	SEQ		
						ACK		

Figure 2.3: Pre-print for Problem 2

Link		Layer 2		Layer 3		Layer 4		Layer 7
From	RA	Src	RA.eth0.MAC	Src	8.8.8.8	Src	53	DNS Reply
		Dst	PC.eth0.MAC	Dst	192.168.1.1	Dst	51827	www.google.de is
To	SW	Prot	IPv4	Prot	UDP	Flags		74.125.39.104!
				TTL	51	SEQ		
						ACK		
From	SW	Src	RA.eth0.MAC	Src	8.8.8.8	Src	53	DNS Reply
		Dst	PC.eth0.MAC	Dst	192.168.1.1	Dst	51827	www.google.de is
To	PC	Prot	IPv4	Prot	UDP	Flags		74.125.39.104!
				TTL	51	SEQ		
						ACK		
From	PC	Src	PC.eth0.MAC	Src	192.168.1.1	Src	58392	
		Dst	RA.eth0.MAC	Dst	74.125.39.104	Dst	80	
To	SW	Prot	IPv4	Prot	TCP	Flags	SYN	
				TTL	64	SEQ	0	
						ACK	0	
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 2.4: Pre-print for Problem 2