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 All in a nutshell

In this task, let us trace everything that happens when you access the `www.google.de` web page on your computer. We assume that ARP and DNS caches are still empty in your private network, but any caches from the first router onward can be assumed populated. The network topology is shown in Figure 1.1. 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.

Now for **each link**, i. e., each section between two devices (e. g. from PC to SW), some selected fields of the message headers which are sent via this link in the respective step are to be noted. Since this is a bit of writing, especially for MAC addresses, we abbreviate addresses using `<device name>.<interface>.<type>`, e. g. let `RA.eth0.MAC` stand for the MAC address of interface `eth0` to router `RA`.
You find pre-printed tables in Figures 1.2 — 1.4.
Each line 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 other colums represent the different layers of the ISO/OSI model. These are each divided into the relevant header fields of the commonly used protocols. Depending on the message, not all columns may be applicable. **Cross out unsused fields.** An example is already entered in the table.

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

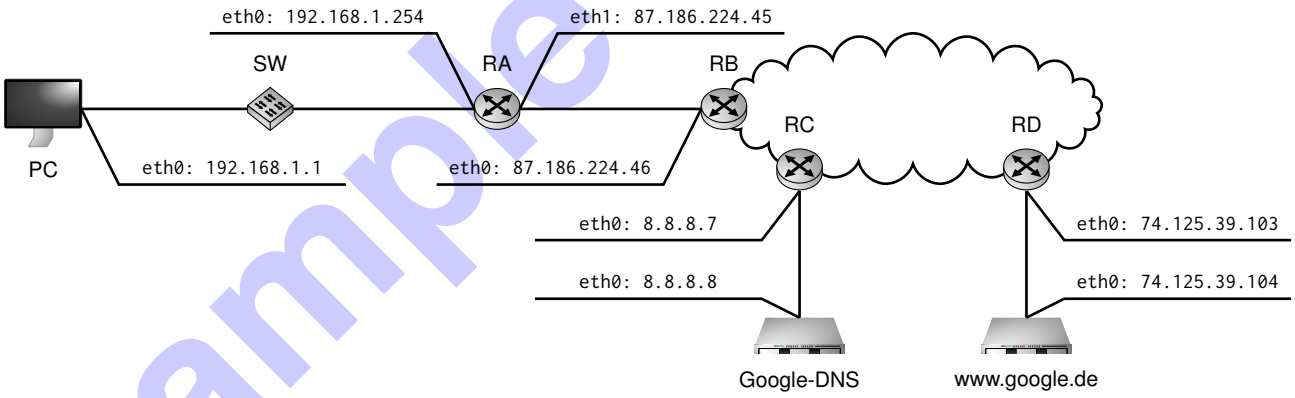


Figure 1.1: Network topology for Problem 1. The relevant links are marked with the numbers 1 – 5.

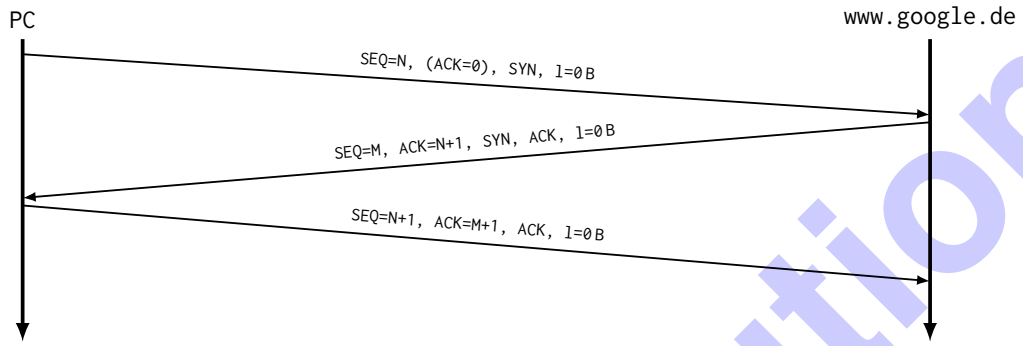
a)* Now fill in the forms in Figures 1.2 – 1.4. Stop after **the first** message transmitted over the link from PC to SW that is directed to `www.google.de`.

Notes:

- We assume that there are a total of 10 other routers between router RB and RC. This is decisive to determine the TTL.
- Enter the application layer protocol by name in the respective column. If applicable, denote the type of the message (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 nodes on between. Serialization time and propagation delay may be neglected. Also assume that no segment losses occur during the entire transmission.

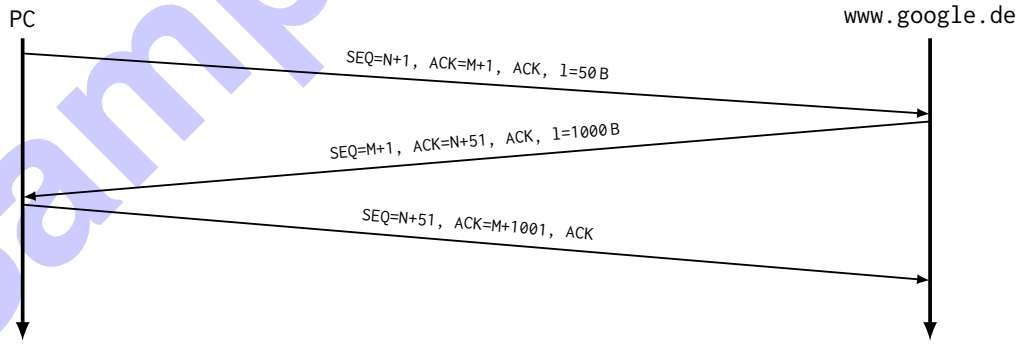
b)* Sketch a path-time diagram representing the TCP connection setup. Specify the sequence number, acknowledgement number, flags set, and the payload length l for each segment.



The acknowledge 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 choosen from $0 \leq N, M, \leq 2^{32} - 1$.

The PC now requests the website stored hosted at `www.google.de`. For this purpose, the PC sends a `HTTP GET` message, which has a useful data 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[1] be larger than l_2.

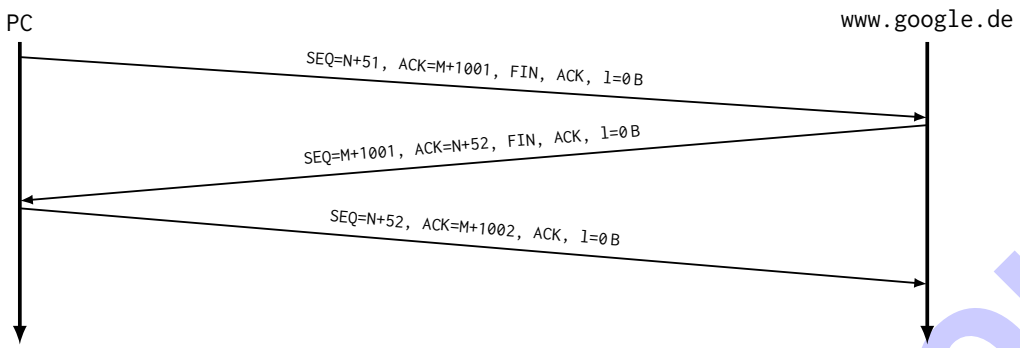
c) Sketch a path-time-diagram which shows the TCP connection phase. Assume the sequence numbers negotiated in Subproblem b). Specify the sequence number, acknowledgement number, flags set, and the payload length l for each segment.



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

[1] The MSS (Maximum Segment Size) specifies the maximum size of a segment. It refers only to the payload from the perspective of layer 4. Plain acknowledgements, for example, are zero-length segments that consist only of a TCP header.

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, flags set, 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 1.2: Vordruck zu Aufgabe 1

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 1.3: Pre-print for Problem 1

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 1.4: Pre-print for Problem 1