Homework2: The Interplay between Web and DNS - Solutions

COM-208: Computer Networks

For most of us, the most common operation we do with our computer is type a URL in the web browser. In this homework, you will explore the underlying messages that this operation triggers.

Whenever you start a new problem, assume that all the DNS, web-browser, and web-server-proxy caches are initially empty.

Also, assume that web browsers and web servers communicate over *persistent* TCP connections, i.e., they use the same TCP connection to exchange multiple HTTP messages (so that they do not have to pay the TCP connection-setup cost for every new HTTP message they exchange).

In most problems, you will be asked to fill in a table, stating all the messages that were transmitted or received as a result of some action. For each message, briefly describe the goal, e.g., is this message an HTTP GET request for a particular URL? is it a DNS request for the IP address of a particular DNS name?

Before you start, a quick recap:

When a process running in the application layer of your computer wants to communicate with another, remote process, it must form the remote process's name; for that, it needs to know the IP address of the network interface behind which the remote process is running.

For example, when you type a URL in your web browser, the latter must form the process name of the web server that stores that URL; for that, it needs to know the IP address of the network interface behind which the web server is running.

To form the process name of the target web server, the web browser extracts the DNS name from the URL and asks a DNS client for the corresponding IP address; the DNS client then asks a local DNS server.

If the local DNS server does not know the answer, it asks another DNS server, according to a DNS hierarchy that consists of three kinds of DNS servers:

- A root server knows the IP address of at least one (typically several) top-level-domain (TLD) servers for each TLD.
- A TLD server knows the IP address of at least one (typically several) authoritative servers for each domain that falls under its TLD.
- An authoritative server knows the IP address of every DNS name that falls under its domain.

DNS clients and servers cache the DNS answers they receive, so that they do not need to send the same DNS questions multiple times. DNS caching significantly affects the amount of DNS traffic on the Internet.

DNS and HTTP messages

You are working on an EPFL computer called `workstation.epfl.ch`. Your local DNS server is `ns.epfl.ch`. This DNS server knows the IP address of root server `a.root-servers.net`, which knows the IP address of .ch TLD server `a.nic.ch`, which knows the IP address of epfl.ch authoritative server `ns.epfl.ch` and unil.ch authoritative server `www.unil.ch`. All these DNS servers perform *iterative* requests. Table 1 shows information about all the servers involved in this problem.

Server	DNS name	IP address
Root DNS server	`a.root-servers.net`	1.1.1.1
`.ch` TLD DNS server	`a.nic.ch`	2.2.2.2
EPFL DNS server	`ns.epfl.ch`	3.3.3.3
UNIL DNS server	`ns.unil.ch`	4.4.4.4
EPFL workstation	`workstation.epfl.ch`	5.5.5.5
UNIL web server	`www.unil.ch`	6.6.6.6

Table 1: Server DNS names and IP addresses.

- You open your web browser and type in `http://www.unil.ch/index.html`. This URL's base file does not reference any other URLs. In Table 2, list all the DNS and HTTP messages that get transmitted as a result of your action.

Packet	Source	Destination	Application protocol	Message
1	5.5.5.5	3.3.3.3	DNS	query: A for www.unil.ch
2	3.3.3.3	1.1.1.1	DNS	query: A for www.unil.ch
3	1.1.1.1	3.3.3.3	DNS	NS for .ch is a.nic.ch with A 2.2.2.2
4	3.3.3.3	2.2.2.2	DNS	query: A for www.unil.ch
5	2.2.2.2	3.3.3.3	DNS	NS for unil.ch is ns.unil.ch with A 4.4.4.4
6	3.3.3.3	4.4.4.4	DNS	query: A for www.unil.ch
7	4.4.4.4	3.3.3.3	DNS	A for www.unil.ch: 6.6.6.6
8	3.3.3.3	5.5.5.5	DNS	A for www.unil.ch: 6.6.6.6
9	5.5.5.5	6.6.6.6	HTTP	GET /index.html HTTP/1.0Host: www.unil.ch
10	6.6.6.6	5.5.5.5	HTTP	200 OK {index.html}

Table 2: Transmitted DNS and HTTP messages.

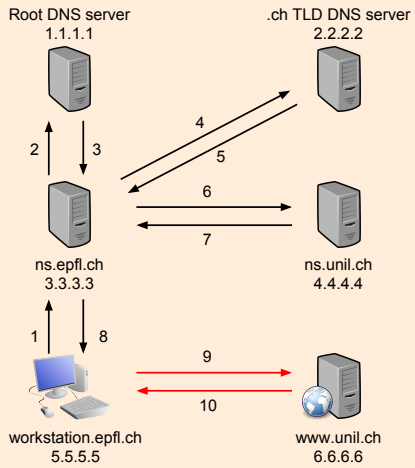

Figure 1: The messages transmitted in Table 2

- Immediately after retrieving this URL, you type in `http://www.unil.ch/logo.png`. In Table 3, list all the DNS and HTTP messages that get transmitted as a result of your action.

Since the DNS client have cached the IP address of the web server (6.6.6.6) , then no DNS lookup will be needed at this stage.

Packet	Source	Destination	Application protocol	Message
11	5.5.5.5	6.6.6.6	HTTP	GET /logo.png HTTP/1.0Host: www.unil.ch
12	6.6.6.6	5.5.5.5	HTTP	200 OK {logo.png}

Table 3: Transmitted DNS and HTTP messages.

Adding a proxy web server

Two users are logged into their respective computers, `user1.epfl.ch` and `user2.epfl.ch`, both located inside the EPFL network. Each user's web browser uses a single persistent TCP connection to communicate with a web server.

EPFL has local DNS server `ns.epfl.ch`, web server `www.epfl.ch`, and proxy web server `proxy.epfl.ch`.

All the computers inside the EPFL network use `ns.epfl.ch` as their local DNS server, which is also the authoritative DNS server for the `epfl.ch` domain. All DNS servers perform *iterative* requests.

- User1 types `http://www.epfl.ch/index.html` in her web browser. This URL's base file references only one other URL, `http://www.epfl.ch/image.png`. User1's web browser does not use any proxy web server.

  In Table 4, list all the application-layer messages and TCP connection-setup packets that are transmitted as a result of this action.

Packet	Source	Destination	Transport protocol	Application protocol	Message
1	`user1.epfl.ch`	`ns.epfl.ch`	UDP	DNS	Name: www.epfl.ch
2	`ns.epfl.ch`	`user1.epfl.ch`	UDP	DNS	IP: `www.epfl.ch`
3	`user1.epfl.ch`	`www.epfl.ch`	TCP		TCP SYN
4	`www.epfl.ch`	`user1.epfl.ch`	TCP		TCP SYN ACK
5	`user1.epfl.ch`	`www.epfl.ch`	TCP	HTTP	GET /index.html
6	`www.epfl.ch`	`user1.epfl.ch`	TCP	HTTP	200 OK ... *index.html*
7	`user1.epfl.ch`	`www.epfl.ch`	TCP	HTTP	GET /image1.png
8	`www.epfl.ch`	`user1.epfl.ch`	TCP	HTTP	200 OK ... *image1.png*

Table 4: Transmitted messages and connection-setup packets.

- User2 types in his web browser `http://www.epfl.ch/help.html`. This URL's base file does not reference any other URLs. User2's web browser uses the EPFL proxy web server.

  In Table 5, list all the application-layer messages and connection setup packets that are transmitted as a result of this action.

Packet	Source	Destination	Transport protocol	Application protocol	Message
1	user2.epfl.ch	ns.epfl.ch	UDP	DNS	Name: proxy.epfl.ch
2	ns.epfl.ch	user2.epfl.ch	UDP	DNS	IP: proxy.epfl.ch
3	user2.epfl.ch	proxy.epfl.ch	TCP		TCP SYN
4	proxy.epfl.ch	user2.epfl.ch	TCP		TCP SYN ACK
5	user2.epfl.ch	proxy.epfl.ch	TCP	HTTP	GET /help.html Host: www.epfl.ch
6	proxy.epfl.ch	ns.epfl.ch	UDP	DNS	Name: www.epfl.ch
7	ns.epfl.ch	proxy.epfl.ch	UDP	DNS	IP: www.epfl.ch
8	proxy.epfl.ch	www.epfl.ch	TCP		TCP SYN
9	www.epfl.ch	proxy.epfl.ch	TCP		TCP SYN ACK
10	proxy.epfl.ch	www.epfl.ch	TCP	HTTP	GET /help.html
11	www.epfl.ch	proxy.epfl.ch	TCP	HTTP	200 OK ... *help.html*
12	proxy.epfl.ch	user2.epfl.ch	TCP	HTTP	200 OK ... *help.html*

Table 5: Transmitted messages and connection-setup packets.

Adding a security twist

Three users, Alice, Bob, and Persa, are logged into their computers, respectively called `alice.ethz.ch`, `bob.ethz.ch`, and `persa.ethz.ch`, all located inside ETHZ's network.

ETHZ has web server `www.ethz.ch` and local DNS server `ns.ethz.ch`, which is also the authoritative server for the `ethz.ch` domain.

EPFL has web server `www.epfl.ch` and local DNS server `ns.epfl.ch`, which is also the authoritative server for the `epfl.ch` domain.

All DNS servers perform *recursive* requests.

Figure 2 illustrates the setup for this problem.

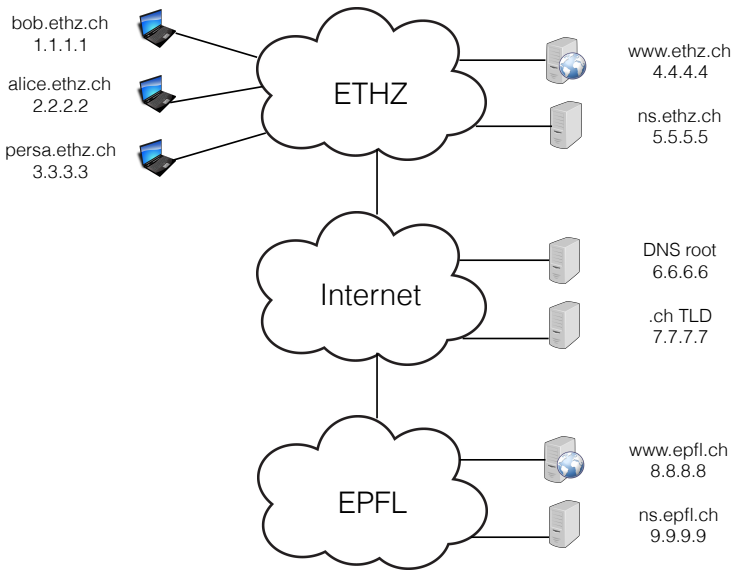

Figure 2: Question Setup

- Alice types in her web browser `http://www.epfl.ch/index.html`. This URL's base file references two other URLs, `http://www.epfl.ch/image.jpg` and `http://www.ethz.ch/file.html` (which does not reference any other URL).

  In Table 6, list all the application-layer messages and connection-setup packets that are transmitted as a result of this action.

Packet	Source IP	Dest. IP	Transport protocol	Application protocol	Purpose
1	2.2.2.2	5.5.5.5	UDP	DNS	www.epfl.ch IP address?
2	5.5.5.5	6.6.6.6	UDP	DNS	www.epfl.ch IP address?
3	6.6.6.6	7.7.7.7	UDP	DNS	www.epfl.ch IP address?
4	7.7.7.7	9.9.9.9	UDP	DNS	www.epfl.ch IP address?
5	9.9.9.9	7.7.7.7	UDP	DNS	www.epfl.ch: 8.8.8.8
6	7.7.7.7	6.6.6.6	UDP	DNS	www.epfl.ch: 8.8.8.8
7	6.6.6.6	5.5.5.5	UDP	DNS	www.epfl.ch: 8.8.8.8
8	5.5.5.5	2.2.2.2	UDP	DNS	www.epfl.ch: 8.8.8.8
9	2.2.2.2	8.8.8.8	TCP		Connection request
10	8.8.8.8	2.2.2.2	TCP		Connection response
11	2.2.2.2	8.8.8.8	TCP	HTTP	GET request for index.html
12	8.8.8.8	2.2.2.2	TCP	HTTP	index.html
13	2.2.2.2	8.8.8.8	TCP	HTTP	GET request for image.jpg
14	8.8.8.8	2.2.2.2	TCP	HTTP	image.jpg
15	2.2.2.2	5.5.5.5	UDP	DNS	www.ethz.ch IP address?
16	5.5.5.5	2.2.2.2	UDP	DNS	www.ethz.ch: 4.4.4.4
17	2.2.2.2	4.4.4.4	TCP		Connection request
18	4.4.4.4	2.2.2.2	TCP		Connection response
19	2.2.2.2	4.4.4.4	TCP	HTTP	GET request for file.html
20	4.4.4.4	2.2.2.2	TCP	HTTP	file.html

Table 6: Transmitted messages and connection-setup packets.

- After Alice has retrieved `http://www.epfl.ch/index.html`, Bob wants to access the same URL.

  Persa is a malicious user who guesses exactly when Bob tries to access `http://www.epfl.ch/index.html`. She wants to trick Bob and make him access a web server running on her own computer, thinking that he is accessing the EPFL web server.

  How can Persa do that by sending DNS traffic to Bob?

  When Bob's DNS client makes a DNS request to `ns.ethz.ch` for `www.epfl.ch`'s IP address, Persa can impersonate `ns.ethz.ch` and respond that `www.epfl.ch`'s IP address is `3.3.3.3` (Persa's IP address). If Persa's response gets to Bob's DNS client before the real

`ns.ethz.ch`'s response, Bob's web browser will connect to the web server running on Persa's workstation instead of `www.epfl.ch`.

Thinking creatively about DNS

You are an ordinary user (not a network/system administrator), and your computer is inside the EPFL network. All computers in this network use the same local DNS server.

Can you find out whether a given external URL, e.g., `www.mit.edu`, was recently accessed by another EPFL user?

> Yes, we can use `dig` to query that website in the local DNS server. For example, `dig example.com` will return the query time for finding example.com. If example.com was just accessed a couple of seconds ago, an entry for example.com is cached in the DNS cache of the university's DNS server, so the query time is negligible (almost 0 msec). Otherwise, the query time is large.

Web page retrieval time

You type a URL in your web browser, and this causes your DNS client to send a DNS query. The number of DNS servers that participate in answering this DNS query is n. The propagation delay between any two DNS servers or a DNS server and client is constant and equal to D_1; the corresponding round-trip time (RTT) is $RTT_1 = 2D_1$. There are no queuing or processing delays.

How much time elapses from the moment your computer transmits the DNS query until it receives a response, in each of the following scenarios:

- All DNS servers perform recursive requests.

> In this case, $2n$ DNS messages are transmitted, and the total time is $2nD_1$. To understand why $2n$ DNS messages are transmitted, see Figure 3.

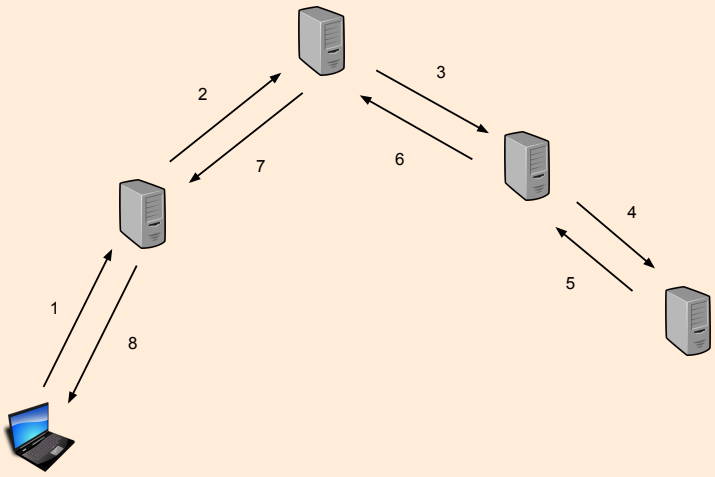

Figure 3: Recursive DNS resolution, with $n = 4$ servers. Each arrow represents a DNS message, and the numbers near the arrows are used to show the order in which the DNS messages are sent. The number of messages sent is $2n$. It can be shown through induction that this is true for any value of n, if $n \geq 1$.

- The local DNS server performs iterative requests.

  In this case, $2n$ DNS messages are transmitted, and the total time is $2nD_1$. To understand why $2n$ DNS messages are transmitted, see Figure 4.

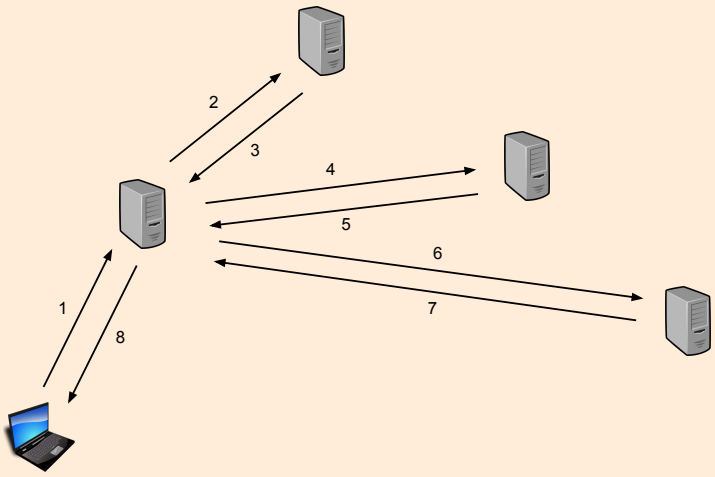

Figure 4: Iterative DNS resolution, with $n = 4$ servers. Each arrow represents a DNS message, and the numbers near the arrows are used to show the order in which the DNS messages are sent. The number of messages sent is $2n$. It can be shown through induction that this is true for any value of n, if $n \geq 1$.

- The local DNS server tries to perform recursive requests, but some of the other DNS servers do not accept such requests (they send an iterative response instead).

  In this case, $2n$ DNS messages are transmitted, and the total time is $2nD_1$. To understand why $2n$ DNS messages are transmitted, see Figure 5.

  Note: this is the most common scenario in the Internet. Normally the root DNS servers and most of the TLD (top-level domain) servers do not allow recursive queries, in order to reduce the load on the servers.
  For servers below the TLDs, the availability of recursion depends on the administrator's decision, but it is normally allowed, since it reduces lookup times. Unlike the scenario shown in this problem, in practice the RTT between the DNS servers at levels below the TLD is small since they usually belong to the same organization. For example, to resolve the name iccluster.epfl.ch, one has to query the root server, the .ch TLD server, then stisun1.epfl.ch and finally ns1.iccluster.epfl.ch. Since the last two servers are located on the same campus, the RTT between them is very small. For a user

querying from outside the campus, it is faster if stisun1.epfl.ch queries recursively instead of requiring the local resolver of the user to send an extra query.

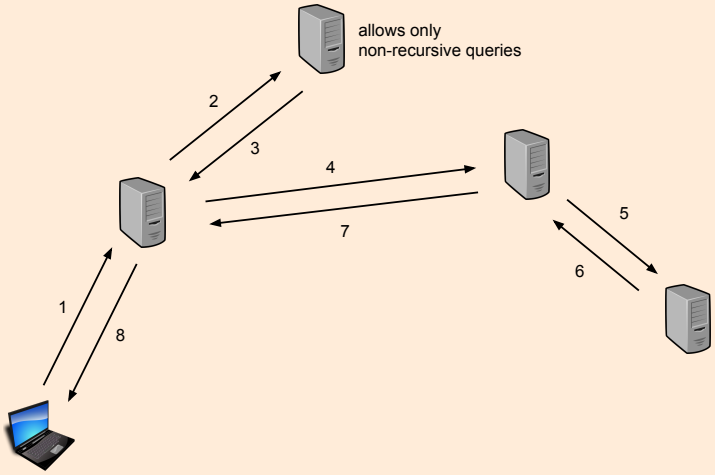

Figure 5: A mix of iterative and recursive DNS resolution, with $n = 4$ servers. Each arrow represents a DNS message, and the numbers near the arrows are used to show the order in which the DNS messages are sent. The number of messages sent is $2n$. *Proof:* If we represent the machines that transmit messages as vertices in a graph, and create an edge in the graph between every two machines that transmit messages, the graph is a tree. This is because it is connected and contains no loops (it is never necessary to query a node twice, and replies always return on the same path on which the query was sent, so it is impossible to form a loop). There are $n + 1$ nodes in the tree (n servers and the computer), and any tree with $n + 1$ nodes has n links. Since two messages are transmitted for each link, the total number of messages is $2n$.

After your computer receives the DNS response, your web browser retrieves the target URL, whose base file references m other URLs, all stored on the same web server. Each retrieved object is small enough that the transmission delays are negligible relative to the propagation delays. The RTT between your computer and the web server is constant and equal to RTT_2. There are no queuing or processing delays.

How much time elapses from the moment you type in the URL until the web browser finishes downloading the entire web page, in each of the following scenarios:

- The web browser and server communicate over a single persistent TCP connection.

With persistent TCP, the web browser opens one TCP connection and reuses it for each request.

First, the browser opens a connection and downloads the base HTML file. This takes two round-trip times.

Then, the browser reads the HTML file to identify the other objects that need to be downloaded. Repeated m times: the browser sends a request, then waits for the server to send back the corresponding reply, as it can be seen in Figure 6; each of this transfer takes one round-trip time plus the transmission delays, which are negligible.

Thus the total download time is $2nD_1 + (2+m)RTT_2$.

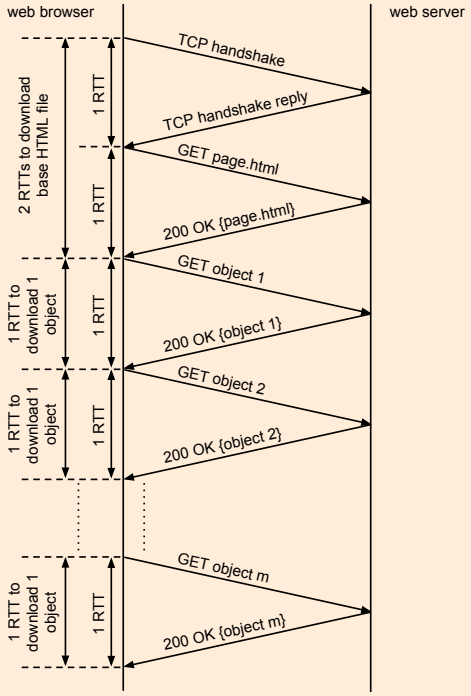

Figure 6: Downloading the web page using single persistent TCP connection.

- The web browser and server communicates over m parallel TCP connections.

With non-persistent TCP, the web browser opens a new TCP connection for each request.

First, the browser uses a connection to download the base HTML file. This takes two round-trip times.

Then, the browser reads the HTML file to identify the other m objects that need to be downloaded. These objects are downloaded using m parallel connections. Hence, they are downloaded in parallel. This also takes two round-trip times. Figure 7 shows an example with $m = 2$.

The total download time is $2nD_1 + 4RTT_2$.

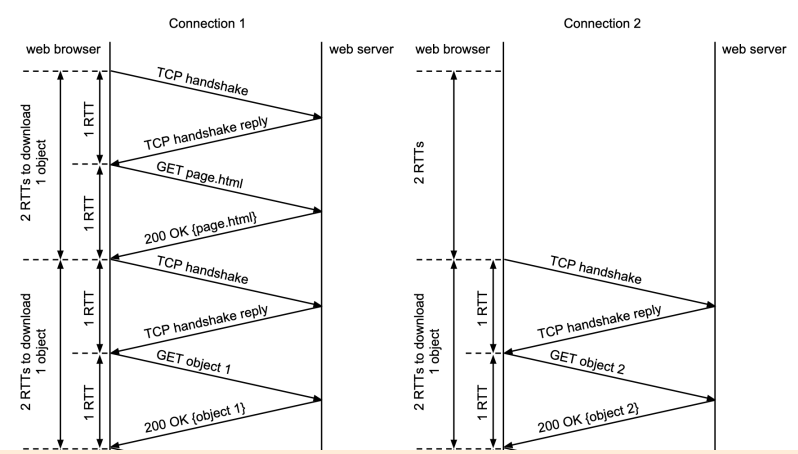

Figure 7: Downloading a web page using non-persistent TCP, with m=2 parallel connections.