INFO0010 Introduction to Computer Networking
First part of the assignment

Maxime Aerts, Guy Leduc
University of Liège

Academic year 2022 - 2023

Contents

1 DNS Client

1.1 DNS

DNS (Domain Name System) is a service that converts hostnames to IP addresses among others. It is an application layer protocol that allows users and servers to exchange messages. DNS provides another way to identify hosts than their IP address, as IP addresses can change over time and are difficult to remember.

Nowadays, DNS is deployed everywhere and many protocols rely on it for their own operation, making it one of the central pillars of the Internet.

1.2 Client

In this first part of the project, you are going to implement a client able to perform queries to a DNS server, then parse the response from the server.

2 DNS communication

DNS is a well-known protocol described in multiple Requests For Comments (RFCs). These RFCs are text file describing as completely as possible concepts, behaviors, or in this case, protocols. As we are not looking for advanced features, the RFC 1035 is enough for our needs. **In particular the following sections of the RFC 1035 will contain all the information required to implement the client: 3.2. RR definitions, 3.3. Standard RRs, 4.1. Format, 4.2. Transport.**

Your client will be able to:

- Send a DNS compliant query containing one header (4.1.1.) and one question (4.1.2.) crafted from the arguments given at program invocation.
- Parse a DNS compliant response containing one header (4.1.1.) and multiple resource records (4.1.3.) of type A (3.4.1.) or TXT (3.3.14.) (other types of resource can be ignored).
- Usually, DNS messages with a size less than or equal to 512 bytes are sent using UDP, while bigger messages are sent using TCP. **In this project, we only use TCP** and the client always reaches the port 53 on the server (default DNS port).
- Some big DNS servers like `8.8.8.8` or `1.1.1.1` do not serve queries not containing additional records. Additional records are outside de scope of this project, thus you should test your program with other DNS servers, such as `139.165.99.199`.

/!\
- If not mentioned, the default class to use in the questions and to parse in the answers is IN, for the Internet (3.2.4.).

3 Program Input/Output

3.1 Input

Your client can be launched using the following command: `java Client <name server IP> <domain name to query> [question type]`, where the first 2 arguments are mandatory and the last one is optional (if not mentioned the question type is A). For example, these commands are valid and you should use these values to test your program:

- `java Client 139.165.99.199 ddi.uliege.be`
- `java Client 139.165.214.214 ulg.ac.be TXT`

3.2 Output

The output of your program will follow a strict format written to `stdout`. It will be divided in ANCOUNT + 1 lines, where ANCOUNT is the number of answers received in the query response. The first line will describe the query: `Question (NS=<name server IP>, NAME=<domain name to query>, TYPE=<question type>)`, where names between

`<...>` must be replace accordingly. Following the same principle, each additional line of answer is formatted like: `Answer (TYPE=<resource type>, TTL=<TTL value in seconds>, DATA="<text representation of answer data>")`.

For example, the input `java Client 139.165.99.199 ddi.uliege.be` should give this output:

```
Question (NS=139.165.99.199, NAME=ddi.uliege.be, TYPE=A)
Answer (TYPE=A, TTL=3600, DATA="139.165.99.199")
Answer (TYPE=A, TTL=3600, DATA="139.165.214.214")
```

And the command `java Client 139.165.214.214 ulg.ac.be TXT` will return something that looks like (values may differ):

```
Question (NS=139.165.214.214, NAME=ulg.ac.be, TYPE=TXT)
Answer (TYPE=TXT, TTL=3600, DATA="v=spf1 ip4:139.165.32.0/24 mx ~all")
Answer (TYPE=TXT, TTL=3600, DATA="MS=ms98930425")
Answer (TYPE=TXT, TTL=3600, DATA="MS=1E7F6381888F476D870A047B7006051C385F9029")
```

4 TCP: A stream oriented protocol

Your DNS client will rely on the TCP protocol, which is stream-oriented. This has at least three implications:

1. You can be sure that the connection is lossless and that there won't be any packet re-ordering.

2. The sender could be buffering the output, i.e. the data that is requested to be sent might not have been effectively sent yet. This can be solved by calling the `flush()` method on the `OutputStream`. Nagle's algorithm can also prevent a packet from departing immediately. It can be deactivated using `setTcpNoDelay(true)`.

3. **The data requested to be sent with one call to a write method will not necessarily be received with only one read method from the other side. Several reads could be necessary to recover the entire information.** This is because we are reading from a stream, not messages (as we would with UDP, for instance). We thus have to find a way to delimit message boundaries on the stream.

   DNS achieves this by having the first 2 bytes of a new DNS message over TCP giving the total length of the message, without counting these 2 bytes (RFC 1035 - 4.2.2 TCP usage).

5 Guidelines

5.1 General

- You will implement the programs using Java 1.8, with packages `java.lang`, `java.io`, `java.nio`, `java.net` and `java.util`.
- This first part of the project has to be completed alone.
- Your program must stop quickly after writing the response information to the console, or on 5 seconds timeout if there is no reply from the DNS server.
- You will not manipulate any file on the local file system.
- At the root of the submitted archive, you'll have a file `report.pdf` and a directory `src/` containing all your Java code. Other files or directories can be present, but it should not be necessary.
- You will ensure your main class is named `Client`, located in `src/Client.java` from the root of the archive, and does not contain any `package` instruction.
- You can cluster parts of your program (not `Client`) in packages or directories.
- You will ensure that your program is fully operational (i.e. compilation and execution) on the student's machines in the lab (ms8xx.montefiore.ulg.ac.be).

Submissions that do not observe these guidelines could be ignored during evaluation.

Your commented source code will be delivered no later than 2nd of November 2022 (**Hard deadline**) to the Montefiore Submission platform (`http://submit.run.montefiore.ulg.ac.be/`) as a .zip package.

5.2 Report

Your program will be completed with a report called `report.pdf` (in the zip package) addressing the following points:

Software architecture: How have you broken down the problem to come to the solution? Name the major classes and methods responsible for requests processing.

Message-oriented communication using a stream: Explain how you handled the recovery of a message-oriented communication scheme using a stream protocol such as TCP.

Limits & Possible Improvements: Describe the limits of your program, especially in terms of robustness, and what you could have done better.

5.3 Evaluation

Criteria used in the evaluation include: program robustness (e.g. robust stream-oriented socket programming and packet parsing), code readability, coding style, and quality of the report.

Some penalties will be applied when the instructions are not strictly followed (e.g. program does not compile on ms800 machines, generates warnings, uses the wrong port number or forbidden libraries, etc.).