INFO0010 Introduction to Computer Networking
First part of the assignment

Clément Vermeylen– s202897 Manon Gerard – s201354
Bac 3 - civil engineer
University of Liège

Academic year 2022 - 2023

1 Software architecture

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

1. Creating the server:

   In the `Server` class, we created a server socket connected to port 53 with a timeout of 5 seconds. The method `getResponse` of this server listens on the port and when a request is done it accepts it and sends it to a new thread in `RunnableServer`.

2. Multi-thread:

   When a thread is `run`ning, the request will be read from the stream and decoded by the `Request` class. A timeout of 5 seconds was once again chosen. Once the request was parsed, the job of creating the response is dedicated to the `Response` class. And finally, the response is sent to the client socket.

3. Parsing the request:

   As said, this is done by the `Request` class. The method `parseHeader` is responsible for the header and `parseQuestion` for the question. In addition, is the `parseQuestion` method, the tunneled data is decoded from a Base 32 representation to a usual representation where we can identify the URL.

   The RCODE value is updated base on the value received in the header and question fields. A lot of verification methods are there to see if an error happened. If one is detected, RCODE is updated to that error representation.

4. Creating the response:

   This is done by the `Response` class. The class has two main function, which are to perform an HTTP get if an answer is needed and to craft the response based of this.

   (a) HTTP get:
   This HTTP get is made only if the RCODE presents no errors. If the RCODE had an error value, only the crafting of the response without answer field was asked.

   The `getFromURL` method is responsible for performing an HTTP get to the URL decoded from the tunneled data. The answer from this get is `truncate`d to a maximum of 60000 bytes.

   (b) Crafting the response:
   Here, it is the `createResponse` method which is called. It accesses a lot of parameters that were decoded in the `Request` class. Thanks to them, it is able to pass the length of the message, to create a Header, copy the question from the request and if needed to create answer records.

The role of composing the header is assigned to the `Header` class. To increase the readability of the header code, we also created a `Flags` and a `Counts` class to be able to set those specific fields in the header. By calling `responseHeader` we are able to obtain what me must transfer to the client as a header.

The `AnswerRecord` class, as its name indicates, create the answer records. For the filed name of the record, we implemented message compression. The rdata send is encoded in `<character-string>`s.

Once the response is created, the asked print is performed and finally `RunnableServer` is able to send the response to the client.

2 Length limit

In your program, you have a hard limit on the HTTP response size. Explain why and how would you do to overpass this limit.

We chose to impose the limit to the RDATA field to 60000 bytes but it reality the limit is of $2^{16}-$ different fields. There is this limitation because we are using IPv4 which length field is of 16 bits. So to overpass this, we could use tunneling and encapsulation to encapsulate an IPv6 packet in an IPv4 packet.

3 DNS Tunneling in practice

In this project, you performed DNS tunneling in only one hop, initiating a direct connection between your controlled client and your controlled server. How could you perform DNS tunneling if your DNS client had to send its queries to an uncontrolled DNS server such as 8.8.8.8 ? What is lacking to your server in order to handle this scenario ?

Our program listens on port 53 and accepts the request as soon as there is one. It can therefore accept a request that it cannot process. In this case, the code must be able to handle this error by sending the request to another server that will be able to process it.

4 Limits & Possible Improvements

Describe the limits of your program, especially in terms of robustness, and what you could have done better.

We truncated our answer from the HTTP get to 60000 bytes, but we could have been more precise by computing the real available size to not exceed the $2^{16}-1$ bytes limit.

In addition, a problem is that if anything wrong happens, except rcode different from 1, the programs cannot recover and terminates with an exception thrown.

Furthermore, our implementation could also have been better for the message compression. As we realized that the name was always in the 12 fields, we basically set it to 12.