LIÈGE université
Sciences Appliquées

University of Liège

Introduction to computer networking (INFO0010-4)

First part of the assignment

Author :
Louis Hogge s192814

Professor : G. Leduc
Assistant : M. Aerts
Year : 2022-2023

1 Software architecture

I have broken down the problem in 5 classes, each one responsible for a specific part of the work :

1. **Client.java :**
   The main class responsible for initiating a new TCP connection with a Socket, setting a time out of 5 seconds, writing the question to stdout in the requested format, creating the message by calling the classes and methods in charge, sending a query in the form of a byte array, retrieving the response length as described in RFC 1035 (4.2.2 TCP usage) and the full response, managing the answer by calling the classes and methods in charge and finally closing input and output streams and socket.
2. **Message.java :**
   A class responsible for handling message objects that contains 2 methods :
   (a) ***public byte[] writeMessage() :***
   Constructs a message byte array by creating header and question thanks to a call to the classes and methods in charge, calculates the length of the message and concatenates it with the header and the question.
   (b) ***public byte[] readMessage() :***
   Separates the answer byte array in 3 parts : header, question and resource records byte arrays. Dispatches these parts to the classes and methods in charge to handle them.
3. **Header.java :**
   A class responsible for handling the header part of a message that contains 2 methods :
   (a) ***public byte[] writeHeader() :***
   Creates a header byte array by concatenating the right bits/bytes in order to have all the required components : ID, flags, QDCOUNT, ANCOUNT, NSCOUNT and ARCOUNT.
   (b) ***public short readHeader(ByteBuffer) :***
   Reads a header byte array and verifies if the ID of the response and the ID of the question match and if some answers have been received.
4. **Question.java :**
   A class responsible for handling the question part of a message that contains 2 methods :
   (a) ***public byte[] writeQuestion() :***
   Creates a question byte array by concatenating the right bits/bytes in order to have all the required components : QNAME, QTYPE and QCLASS. For QCLASS, it splits the domain name in parts delimited by the dots and converts each of these parts into byte arrays. For QTYPE, it matches the question type argument to the right type value.
   (b) ***public void readQuestion(ByteBuffer) :***
   Reads a question byte array.
5. **ResourceRecord.java :**
   A class responsible for handling the resource records part of a message that contains 3 methods :
   (a) ***public void readResourceRecord(byte[], short) :***
   Reads a resource records byte array, handles the domain name compression associated with TCP protocol and prints the answers to stdout by calling the methods in charge.
   (b) ***private void readOffset(byte[], ByteBuffer, byte, byte) :***
   Handles the domain name compression associated with TCP protocol by recognising pointers and labels thanks to the 11 or 00 first bits, in the case of a pointer, by calculating the offset from the start of the message (i.e. the first octet of the ID field in the domain header) to find domain name parts and by concatenating the latter.
   (c) ***private void manageRDATA(short, short, int, int) :***
   Manages RDATA by recognising type and class answers and by adapting method accordingly, then, by printing answers to stdout in the requested format. For type A with class IN, RDATA is a 32 bit Internet address so it must get the different domain name parts,

translate them to String and concatenate them together separated by dots. For type TXT with class IN, RDATA is one or more character strings, each of these preceded by an octet containing its length. So it must get the bytes of the character strings, translate them to String and concatenate them together. For other types and classes, RDATA is thrown away and no answer is printed to stdout.

2 Message-oriented communication using a stream

The recovery of a message-oriented communication scheme using a stream protocol such as TCP is handled in **Client.java** thanks to a socket that initiates a new TCP connection, an output stream in charge of sending data to an internet address and an input stream which job is to read data from an internet address. The input stream retrieves the response length written in 2 bytes, as described in RFC 1035 (4.2.2 TCP usage), then converts bytes to length (data sent over the network is always big-endian) and finally retrieves the full response written in a number of bytes corresponding to the calculated length. This full response must then be parsed following a strict procedure where each bytes has a specific job. That is done in **Message.java**, **Header.java**, **Question.java** and **ResourceRecord.java**.

3 Limits & Possible Improvements

— The only question and answer types allowed are A and TXT. To be complete I would have to add the possibility to send questions and receive/parse answers of all the other types which are : NS, MD, MF, CNAME, SOA, MB, MG, MR, NULL, WKS, PTR, HINFO, MINFO and MX.

— The class used in questions and answers is always IN. To be complete I would have to add the possibility to send questions and receive/parse answers of all the other types which are : CS, CH and HS.

— Additional records are outside the scope of this project. To be complete I would have to add the possibility to handle the latter.

— Resource records in the authority records section are not handled. To be complete I would have to add the possibility to handle the latter.

— If sending the question or receiving the response does not work, the program stops. Setting up a retry system could be interesting.