TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph.D.
Gregor Maier, Jörg Wallerich, Andi Wundsam

3rd Work Sheet Praktikum Protokolldesign WS 07/08

Question 1: (20 points) Latency and Transmission Delay[1]

(a) Calculate, only based on the speed of light ($c = 300,000 km/s$), the minimal time required to send data from here

- to Paris
- via a transatlantic submarine cable to Washington D.C.
- via a geostationary satellite to Washington D.C.

You can use approximate distance estimations!
Some hints for estimating the distances:

- You buy enough material for an experiment setup (quite interesting when it comes to the geostationary satellite ;-))
- You buy enough airline tickets from a big airline and check how many bonus miles you get ;-)
- You buy cheap airline tickets, fly with a GPS receiver, and calculate the distances through the GPS coordinates ;-)
- Use some Internet search engines

(b) Calculate the "transmission delay" for transmitting 400 byte and 1500 byte packets via

- 56Kbit modem
- DSL
- 10 Mbit/s Ethernet
- Gigabit Ethernet

Which value is more significant for each distance and medium combination?
Submit your description and calculation as a file (as pdf, ps, text or html—*no office files!*).

Question 2: (80 points) Simplified DNS client (first part):

In this exercise you will implement a simplified DNS client. The DNS protocol is specified in **RFCs 1034 and 1035**. To solve this exercise sucessfully it is not required to read the RFCs completely. Use the Kurose and Ross book to get an overview of the DNS protocol. For the implementation details we will give references to the relevant sections of the RFCs.

We strongly suggest, that you use subroutines in your code. On the next work sheet, you will enhace the DNS-client.

First you should have a look at the `dig` program. The information returned by your DNS client should convey the same information as the output from `dig`. You **do not** have to clone the output of `dig`.

A DNS packet always has the same structure (see also **RFC 1035, Section 4.1**):

[1] If you are unsure about the definition of these, see the book *Kurose and Ross – Computer Networking.* Online available at `http://www.net.t-labs.tu-berlin.de/` — Login: `student`; Password: `quarter24`

Header
Questions
Answers
Authoritys
Additionals

(a) Write a program that constructs a DNS header and writes the header to a file. The DNS header is always 12 byte long and has the following format:

16	1	4	1	1	1	1	3	4	16	16	16	16
ID	**QR**	**Opcode**	**AA**	**TC**	**RD**	**RA**	**Z**	**Rcode**	**Qdcount**	**Ancount**	**Nscount**	**Arcount**

You can find a detailed explanation of the header fields in **RFC 1035, Section 4.1.1**.

Internally your program should call a subroutine, which fills the header fields based on the parameters passed to the subroutine. You can check the correctness of your function by using different parameters and by checking the results with a hex editor. On the computers in the ZIP pool you can use `hexdump`[2] for example.

Deliverables:

- Sourcecode of your program
- The resulting file, when you call your function with the following parameters `ID` = 1000, `QR` = 1, `AA` = 1, `RD` = 1, `Qdcount` = 1, other parameters = 0.

The file /afs/net.t-labs.tu-berlin.de/home/praktikum/daten/3.uebung/dns.header contains such a header. You can use it to verify your result.

(b) Enhance your program, so that it creates a DNS query (question) and that it writes the query to a file. Your program should get the DNS name (domain name) from the commandline parameters. In general a DNS question has the following format:

multiple octets	16	16
Qname	**Qtype**	**Qclass**

The details are described in **RFC 1035, Section 4.1.2**. Additionally to the subroutine for the header from part (a), you will also need a new subroutine, which fills a question section according to parameters passed to the subroutine. **Note:** you should not use the header field from part (a), rather you should use your own (usefull!) values for the header fields. Keep in mind that the DNS names in the `Qname` field is encoded in a particular format (see **RFC 1035, Sections 2.3.1 and 3.1**). The DNS protocoll knows quite some different kinds of questions. For this exercise it is sufficient, if your program can handle questions with Qtype `A` and Qclass `IN`. You can find descriptions for the types and classes in **RFC 1035, Section 3.2.2-5**.

Deliverables:

- Sourcecode of your program
- The resulting file, if you call your program for `www.heise.de`.

(c) Construct a DNS request, send it to a DNS-Server, receive the reply and write the answer into a file.

Deliverables:

- The source code of your program

[2] Call it like this: `hexdump -C file`

- The output of your program (DNS answer packet) for `www.heise.de` and `www.net.t-labs.tu-berlin.de`. You can use 130.149.220.253 as DNS-server.

(d) Describe the difference between recursive and iterative DNS queries in you own words. What are the typical usages of recursive and iterative DNS queries in practice? Submit your anwser / description as a file (no office documents).

Submission details: look at the FAQ (`http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`)
Due Date: Tuesday, November 13, 2007, 11:59 h s. t.

—End of assignments for the 3rd Work Sheet—

For reference we give you some of the exercises ***for the next worksheets****. The intention is that you get idea where your program should be extensible. However, we do not guarantee that the information below will be on the next worksheet as is. There might be changes!*
The DNS reply packet can contain any number of entries in the `ANSWERS`, `AUTHORITYS`, and `ADDITIONALS` sections. The actual number of entries per section is written in the appropriate header fields by the server. A single entry in a reply section is encoded as a Resource Record (RR).

multiple octets	16	16	32	16	multiple octets
Name	**Type**	**Class**	**TTL**	**Rdlength**	**Rdata**

Details are specified in **RFC 1035, Section 4.1.3**.

- Parse and output the server's reply. First your program should analyze the complete header and output the header fields. If the server returned an error, the program should exit with an appropriate error message. Your program only has to handle entries with type `A` and class `IN`.

  Now you should print the `QUESTIONS` and `ANSWERS` sections. You only have to print the first entry in the `QUESTIONS` and `ANSWERS` section. `AUTHORITYS` and `ADDITIONALS` can be ignored. Your output should contain the same information that the `dig` program prints. *It is not required to clone `dig`'s output format!*

  A problem is, that DNS uses a compression algorithm to save space when encoding DNS names. For the time being it's not required to understand how the compression works. Note that the `Name` field in the `ANSWERS` section is exactly 2 bytes long if the name is compressed. Output the string `compressed` instead of the DNS name in this case.

- Enhance your program, so that it can handle compressed DNS names in answers. A description of the algorithm can be found in **RFC 1035, Section 4.1.4**.

- Complete your program, so that it prints all entries in the `ANSWERS`, `AUTHORITYS`, and `ADDITIONALS` sections. In addtion to type `A` your program should also be able to handle type `NS`. If you encounter an unknown type, your program should print `unknown type`. You should print the same information that the `dig` prints.

- Enhance your program, so that it can do iterative queries and that it can handle the answer packets encountered on the way. I.e., if a server reply does not contain the desired answer, your program has to automatically ask the next server in the chain. Print all reply packets your program receives.

Submission details: look at the FAQ (`http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`)
Due Date: Tuesday, November 13, 2007, 11:59 h s. t.