TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph. D.
Harald Schiöberg, Bernhard Ager, Vlad Manilici

3. Übung Protokolldesign WS 08/09

Aufgabe 1: (20 Punkte) Latenz und Transmission delay

(a) Berechne, lediglich basierend auf Lichtgeschwindigkeit ($c = 300.000\,\mathrm{km/s}$), wie lange es mindestens dauern muß, um Daten von hier

- nach Paris
- über ein Transatlantik-Kabel nach Washington D.C. und
- über einen geostationären Satelliten nach Washington D.C. zu übertragen.

Die Entfernungen müssen nur ungefähr abgeschätzt werden!

(b) Berechne, wie lang das "transmission delay" ist um 400 Bytes bzw. 1500 Bytes lange Pakete via 56Kbit Modem , DSL, 10 Mbit/s Ethernet bzw. Gigabit Ethernet zu senden.

Welcher Wert fällt bei den jeweiligen Entfernungen und Übertragungsmedien jeweils am meisten ins Gewicht?

Abzugeben: Deine Erklärung und die Berechnung als Datei (als ps, pdf, txt oder html).

Aufgabe 2: (80 Punkte) Vereinfachter DNS-Client (erster Teil):

In dieser Aufgabe geht es darum, einen einfachen DNS-Client zu implementieren. Das DNS-Protokoll wird in den **RFCs 1034 und 1035** spezifiziert. Um diese Aufgabe erfolgreich zu lösen, ist es nicht nötig, beide RFCs komplett zu lesen. Es sollte ausreichend sein, evtl. die Grundlagen des DNS-Protokols zu wiederholen, z.B. anhand des Buches (Kurose/Ross, siehe Web-Seite). Für die entsprechenden Implementierungsdetails wird dann in den einzelnen Aufgabenteilen an die genauen Stellen in den RFCs verwiesen. Die RFCs sind verfügbar über `http://www.rfc-editor.org/`.

Zu Beginn solltest Du Dir das Programm `dig` genauer anschauen. Die von Deinem DNS-Client ausgegebenen Daten sollten die gleichen Informationen enthalten, wie die von `dig` ausgegebenen (du musst die Ausgabe von `dig` natürlich **nicht** nachmachen!!).

Ein DNS-Paket hat immer folgenden Aufbau (siehe auch **RFC 1035, Abschnitt 4.1**):

Header
Questions
Answers
Authoritys
Additionals

(a) Schreibe ein Programm, das einen DNS-Header zusammenbaut und in eine Datei schreibt. Der DNS-Header ist immer 12 Bytes lang und hat folgendes 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**

Eine genaue Erläuterung der einzelnen Felder findest Du in **RFC 1035, Abschnitt 4.1.1**

Dein Programm sollte so aufgebaut sein, dass intern eine Funktion aufgerufen wird, die die Felder des Headers anhand der übergebenen Parameter ausfüllt. Du kannst die Korrektheit Deiner Funktion überprüfen, indem Du sie mit verschiedenen Parametern aufrufst und Dir die Ausgabedatei in einem Hexeditor anschaust. Auf den Praktikumsrechnern kann man hierzu `hexdump` verwenden.

Abzugeben:

- Der Quelltext Deines Programmes
- Die resultierende Datei, wenn Du Deine Funktion mit den Parametern `ID` = 1000, `QR` = 1, `AA` = 1, `RD` = 1, `Qdcount` = 1, restliche Parameter = 0 aufrufst.

(b) Dein Programm ist nun so zu erweitern, dass es eine DNS-Anfrage zusammenbaut und in eine Datei schreibt. Den DNS-Namen soll das Programm von der Kommandozeile lesen. Allgemein hat eine Question folgenden Aufbau:

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

Details kannst Du **RFC 1035, Abschnitt 4.1.2** entnehmen. Zusätzlich zu der Funktion für den Header aus Teil (a) wirst Du eine neue Funktion brauchen, die die Felder einer Question entsprechend den übergebenen Parametern füllt. **Achtung:** es sollen nicht die Headerfelder aus Teil (a) verwendet werden, sondern eigene (entsprechend sinnvolle) Werte für die Headerfelder verwendet weerden. Es ist darauf zu achten, dass die DNS-Namen im `Qname`-Feld in einem besonderen Format kodiert sind (siehe **RFC 1035, Abschnitt 2.3.1 und 3.1**). Das DNS-Protokoll kennt sehr viele verschiedene Typen von Questions. Für diese Aufgabe ist es ausreichend, wenn Dein Programm nur Questions mit Qtype `A` und Qclass `IN` unterstützt. Erläuterungen zu den einzelnen Typen und Klassen findest Du in **RFC 1035, Abschnitte 3.2.2-5**

Abzugeben:

- Der Quelltext Deines Programmes
- Die resultierende Datei, wenn Du Dein Programm für `www.heise.de` aufrufst.

(c) Beschreibe den Unterschied zwischen rekursiven und iterativen DNS-Anfragen.
Wo werden in der Praxis typischerweise rekursive, wo iterative Anfragen benutzt?

Abzugeben: Deine Beschreibung als Datei.

Details zur Abgabe der Aufgaben: siehe online FAQ.

Abgabedatum: Mi, 12. Nov. 2008, 23:59 Uhr

—Ende der Aufgabenstellung für das 3. Aufgabenblatt.—

Vorschau auf Blatt 4

*Zur Übersicht hier noch die Aufgabenteile, **die nächste Woche** kommen werden. Damit kannst Du besser abschätzen, an welchen Stellen Dein Programm erweiterbar gestaltet werden muss. Wenn du willst kannst du auch schonmal damit anfangen, da die nächste Aufgabe recht umfangreich sein wird, aber gib bitte noch keine Lösungen ab.*

- Nun soll die DNS-Anfrage zu einem DNS-Server geschickt werden und das entsprechende Antwortpaket ausgewertet werden. Dieses kann jeweils beliebig viele Antworten in den Bereichen `ANSWERS`, `AUTHORITYS` und `ADDITIONALS` haben. Die konkrete Anzahl trägt der Server in die entsprechenden Felder im Header ein. Eine einzelne Antwort wird durch einen Resource Record (RR) kodiert:

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

Genaueres findet sich in **RFC 1035, Abschnitt 4.1.3**. Für diesen Aufgabenteil muss Dein Programm nur Antworten mit Type `A` und Class `IN` unterstützen, andere Fälle brauchen nicht betrachtet zu werden. Es reicht außerdem, nur den ersten Resource Record im `ANSWERS`-Bereich zu bearbeiten; `AUTHORITYS` und `ADDITIONALS` können ignoriert werden.

Dein Programm sollte zunächst den kompletten Header des Antwortpaketes auswerten und die einzelnen Felder ausgeben. Hierbei muss darauf geachtet werden, dass man bei eventuellen Fehlercodes vom Server die weitere Bearbeitung mit einer passenden Fehlermeldung beendet.

Anschließend ist der `QUESTIONS`- und `ANSWERS`-Bereich auszugeben, wobei man von genau einer Question und einer Answer ausgehen kann. Das Ausgabeformat Deines Programmes sollte möglichst genau demjenigen von dig entsprechen. Ein Problem ergibt sich noch daraus, dass DNS bei den Antworten einen Kompressionsalgorithmus verwendet, um bei DNS-Namen Platz zu sparen. In diesem Aufgabenteil musst Du noch nicht verstehen, wie diese Kompression funktioniert. Gehe einfach davon aus, dass das Name-Feld im `ANSWERS`-Bereich durch die Kompression genau 2 Bytes lang wird und gib anstatt dem richtigen DNS-Namen einfach die Zeichenkette `compressed` aus.

Abzugeben:

- Der Quelltext Deines Programmes
- Die Ausgabe Deines Programmes, wenn Du es für `www.heise.de` aufrufst. Als Nameserver kannst Du z. B. `130.149.220.253` oder `130.149.2.12` verwenden.

- Erweitere Dein Programm, so dass es auch mit komprimierten DNS-Namen bei den Antworten zurechtkommt. Eine genaue Erklärung des Algorithmus findest Du in **RFC 1035, Abschnitt 4.1.4**.

  Abzugeben:

  - Der Quelltext Deines Programmes
  - Die Ausgabe Deines Programmes, wenn Du es für `www.heise.de` aufrufst.

- Dein Programm ist nun so zu vervollständigen, dass es alle Antworten (`ANSWERS`, `AUTHORITYS` und `ADDITIONALS`) ausgibt. Dabei soll es bei der Ausgabe zusätzlich zu Type `A` auch Type `CNAME` und `NS` unterstützen. Beim QUESTIONS-Bereich kann weiterhin von genau einer Question ausgegangen werden. Ausgabeformat sollte wieder weitgehend dem von `dig` entsprechen.

  Abzugeben:

  - Der Quelltext Deines Programmes
  - Die Ausgabe Deines Programmes für `www.heise.de` und `www.google.de`.

- Erweitere Dein Programm um die Möglichkeit von iterativen Anfragen und die Fähigkeit, mit entsprechenden Antworten umzugehen. Gib alle Antwortpakete aus. Die iterative Auflösung eines Names soll vollautomatisch geschehen, es soll also ein Aufruf deines Programms genügen um die gesamte Auflösung durchzuführen.

  Abzugeben:

  - Der Quelltext Deines Programmes
  - Die Ausgabe Deines Programmes, die eine iterative Auflösung von `www.heise.de` zeigt.