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

3. Blatt Praktikum Protokolldesign WS 07/08

Aufgabe 1: (20 Punkte) Latenz und Transmission Delay[1]

(a) Berechne, lediglich basierend auf Lichtgeschwindigkeit ($c = 300.000 km/s$), 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!
Einige Anregungen zum Abschätzen der Entfernungen:

- Man kauft sich genügend Material und macht einen geeigneten Versuchsaufbau (vor allem zum Punkt geostationärer Satellit ;-))
- Man kauft sich Flugtickets einer großen Fluggesellschaft und schaut nach wieviel Bonusmeilen man bekommt ;-)
- Man kauft sich Billig-Airline Flugtickets, fliegt selbst mit einem GPS Empfänger und rechnet mit Hilfe der Koordinaten die Differenz nach ;-)
- Man sucht einfach mit ein paar Suchmaschinen im Internet.

(b) Berechne, wie lang das „transmission delay“ ist um 400 Bytes bzw. 1500 Bytes lange Pakete via

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

zu senden.

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

Gib Deine Erklärung und die Berechnung als Datei ab (als ps, pdf, text oder html – keine Office Dateien!).

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, sich zunächst einen Überblick über das DNS-Protokoll anhand des Buches von Kurose und Roos zu verschaffen. Für die entsprechenden Implementierungsdetails wird dann in den einzelnen Aufgabenteilen an die genauen Stellen in den RFCs verwiesen.

Wir empfehlen ganz deutlich, dass ihr Subroutinen verwendet. Auf dem nächsten Aufgabenblatt, werdet ihr den DNS-Client weiterentwickeln.

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).

[1] Falls die Definition unklar ist, schaut im Buch *Kurose und Ross – Computer Networking* nach. Online verfügbar unter `http://www.net.t-labs.tu-berlin.de/` — Login: `student`; Passwort: `quarter24`

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 füllt. 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`[2] verwenden.

Abzugeben sind:

- 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.

Die Datei /afs/net.t-labs.tu-berlin.de/home/praktikum/daten/3.uebung/dns.header enthält so einen Header. Du kannst sie verwenden, um dein Ergebnis zu überprüfen.

(b) Dein Programm ist nun so zu erweitern, dass es eine DNS-Anfrage zusammenbaut und in eine Datei schreibt. Den DNS-Namen (Domainnamen) 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 sind:

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

[2] Aufruf: `hexdump -C datei`

(c) Konstruiere eine DNS-Anfrage, sende sie zum DNS-Server, empfange die Server-Antwort und schreibe sie die Antwort in eine Datei.

Abzugeben sind:

- Der Quelltext Deines Programmes
- Die Ausgabe Deines Programmes (DNS-Anwort Paket) für `www.heise.de` und `www.net.t-labs.tu-berlin.de`. Als DNS-Server kannst du 130.149.220.253 verwenden.

(d) Beschreibe den Unterschied zwischen rekursiven und iterativen DNS-Anfragen mit eigenen Worten.

Wo werden in der Praxis typischerweise rekursive, wo iterative Anfragen benutzt?

Gib Deine Beschreibung als Datei mit ab (keine Office-Dokumente).

Details zur Abgabe der Aufgaben: siehe FAQ (`http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`)
Abgabe: bis Dienstag, den 13. November 2007, 11:59 h s. t.

—Ende der Aufgabenstellung für das 3. Blatt.—

Zur Übersicht hier noch die Aufgabenteile, ***die nächste und evtl. übernächste Woche*** *kommen werden. Damit kannst Du besser abschätzen, an welchen Stellen Dein Programm erweiterbar gestaltet werden muss. Wir geben allerdings keine Garantie, dass es nicht noch Änderungen in den Details dieser zukünftigen Aufgaben geben wird.*

Ein DNS-Antwortpaket 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**.

- Interpretiere und gibt die Serverantwort aus. 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. Es ist ausreichend, wenn Dein Programm nur Antworten vom Type `A` und Class `IN` unterstützt, andere Fälle brauchen nicht betrachtet zu werden.

  Anschließend ist der `QUESTIONS`- und `ANSWERS`-Bereich auszugeben, wobei man nur die erste Question und die erste Answer ausgeben muss. `AUTHORITYS` und `ADDITIONALS` können ignoriert werden. Das Ausgabeformat Deines Programmes sollte möglichst genau die selben Informationen wie die Ausgabe von `dig` enthalten.

  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.

- 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**.

- 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 `NS` unterstützen. Bei unbekannten Typen soll dein Program `unknown type` ausgeben. Beim QUESTIONS-Bereich kann weiterhin von genau einer Question ausgegangen werden. Die ausgegebenen Informationen sollen wieder denen von `dig` entsprechen.

- Erweitere Dein Programm um die Möglichkeit von iterativen Anfragen und die Fähigkeit, mit entsprechenden Antworten umzugehen. Sollte also in einem Antwortpaket nicht die angefragte Information enthalten sein, so muss Dein Programm selbständig beim nächsten Server in der Kette weiterfragen. Gib alle Antwortpakete aus, die du auf dem Weg zum Ziel erhältst.

Details zur Abgabe der Aufgaben: siehe FAQ (`http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`)
Abgabe: bis Dienstag, den 13. November 2007, 11:59 h s. t.