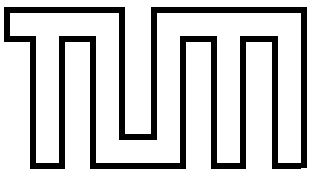


Technische Universität München

Fakultät für Informatik
Forschungs- und Lehreinheit Informatik VI

Übung zur Vorlesung Echtzeitsysteme

Aufgabe 3

Simon Barner
barner@in.tum.de

Irina Gaponova
gaponova@in.tum.de

Stephan Sommer
sommerst@in.tum.de

Wintersemester 2009/10

Simple Network Time Protocol (SNTP)

SNTP steht für *Simple Network Time Protocol* und ist in RFC 2030 definiert (siehe http://www.faqs.org/rfcs/rfc2030.html) und ist eine vereinfachte Variante des NTP Protokolls. NTP ist ein Standard Netzwerkprotokoll für Uhrensynchronisierung. NTP verwendet UDP für die Nachrichtenübertragung (UDP Port 123). Die Genauigkeit der Zeitsynchronisierung beträgt 10 Millisekunden in einem globalen Netzwerk - 200 Mikrosekunden in einem lokalen Netzwerk. NTP wird meist mit einer UTC-Zeitskala eingesetzt. UTC (*Coordinated Universal Time*) beachtet Schaltsekunden im Gegensatz zu z.B. der TAI Zeitskala (*International Atomic Time*, zu unterschiedliche Zeitskalas siehe http://stjarnhimlen.se/comp/time.html). Sowohl SNTP als auch NTP verwenden das gleiche Netzwerkpaketformat. Die Unterschiede bestehen darin, dass die Genauigkeit des SNTP Protokolls fast immer schlechter ist, als die einer vollwertigen NTP Implementierung, da SNTP nur einen Zeitserver verwendet, während NTP mehrere verwendet. Ausserdem kann SNTP plötzliche Zeitsprünge (wegen Schaltsekunden) nicht vermeiden. Für die Übung ist das SNTP Protokoll ausreichend, um die Grundlagen zu verdeutlichen.

Ziel

In der Übung wird ein SNTP Client implementiert, der die aktuelle Zeit von einem NTP Server (ntp.in.tum.de) bezieht und basierend darauf den Offset der eigenen Uhr berechnet. Neben der Funktionsweise eines (S)NTP Clients wird hierbei auch die Socket Schnitstelle (Schnitstelle zur Netzwerk-Kommunikation) betrachtet.

Die Aufgabe wird in der Programmiersprache C implementiert. Zum übersetzen wird Microsoft Visual Studio verwendet.

Implementierungsumgebung

Zum erstellen eines Projekts gehen Sie wie folgt vor. Nach dem Starten von Visual Studio:

File → *New* → *Project* → *Win32* → *Win32 Console Application*

Project → *Properties* → *Configuration properties* → *C/C++* → *Advanced* → *Compile as „compile as C code“*

Anschliessend muss die Socket Bibliothek zum erstellten Projekt gelinkt werden:

Project → *Properties* → *Linker* → *CommandLine*

und tippen Sie „wsock32.lib“ ein.

Die Programmiersprache C bietet fast die gleiche Socket Schnittstelle unter Windows wie unter VxWorks. Die Aufgabe wird unter Windows gelöst, weil der Simulator in VxWorks nur begrenzte Netzwerk-Funktionalität hat.

Aufgabe 3: SNTP Client

Aufgabe 3.1: Netzwerk-Kommunikation

Um mit einem NTP Server kommunizieren zu können, muss zuerst die Kommunikation implementiert werden. Ergänzen Sie dazu das Rahmenprogramm SNTPClient.c um die Netzwerkfunktionalität.

Implementierungshinweis

Verwenden Sie für die UDP Kommunikation als Zielserver „ntp.in.tum.de“ und als Port 123.
Um eine Nachricht an den Server zu schicken benutzen Sie folgende Aufrufe:

- socket(); Erzeugt einen Socket für z.B. UDP
- sendto(); Schickt eine Nachricht zu einer angegeben Adresse
- ...
- closesocket(); Schliesst das angegebene Socket

Um eine Nachricht von dem Server empfangen zu können, sind folgende Funktionen nötig:

- socket(); Erzeugt ein Socket, das auf z.B. UDP gebunden ist
- recvfrom(); Empfängt eine Nachricht von einem Socket und speichert die Senderadresse
- ...
- closesocket(); Schliesst den angegebenen Socket

Zum Senden und Empfangen kann der gleiche Socket verwendet werden. Die (S)NTP Nachricht wird mit der Funktion createNTPMessage() erzeugt. Als Parameter benötigt diese Funktion unter anderem die Sendezeit des Paketes. Die aktuelle Zeigt bekommt man under Windows mit dem Aufruf von time(), die einen time_t Wert zurückliefert.

Je nach Betriebssystem werden folgende Datentypen für die Zeitrepräsentation verwendet:

- time_t : stellt die Zeit als Anzahl von Sekunden (Windows, Linux, VxWorks) item struct timeval : stellt die Zeit als 2 Werte dar: Sekunden und Mikrosekunden (Linux)

item struct timespec : stellt die Zeit als 2 Werte dar: Sekunden und Nanosekunden (VxWOrks)

Aufgabe 3.2: Erzeugen und Parsen von den NTP Nachrichten

Implementieren Sie eine Funktion zum Parsen eines Zeitstempels in einer (S)NTP Nachricht *decodeTimestamp()* und eine Funktion zum parsen einer NTP Nachricht *parseNTPMessage()*.

Implementierungshinweise decodeTimestamp

Die Zeitstempel von NTP werden als 64-bit Festkomma Werte codiert. Dabei umfasst der Ganzzahl Anteil 4 Byte und der Nachkommaanteil ebenfalls 4 Byte. Zu beachten ist, dass „Big Endian“ bei der Anordnung der Bytes verwendet wird. Dies bedeutet, dass das „most significant“ Byte die niedrigste Adresse hat.

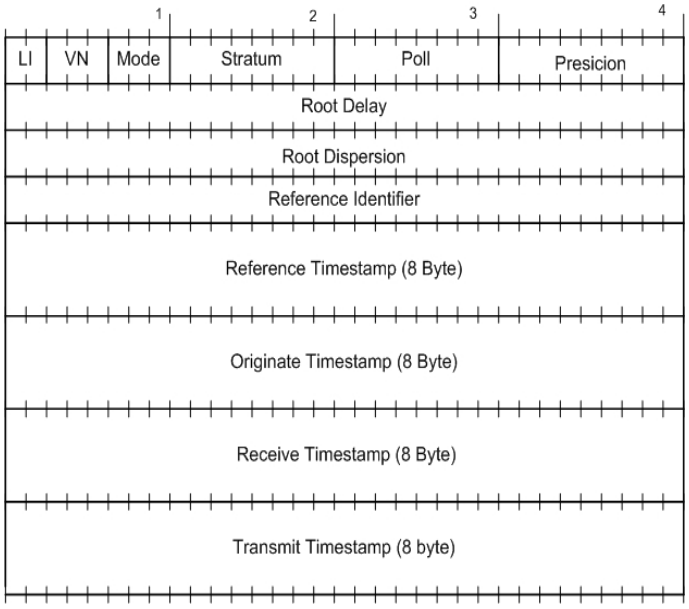
Beispielscode zur Umrechnung eines Byte-Arrays im Big-Endian Format in einen double Datentyp:

```
//char pointer[] is the given array
int i; // variable to iterate in the array
short byte;
double base;
double timestamp = 0.0;

for (i = 0; i < 8; i++) {
        /* byte is a signed data type */
        byte = *(i + pointer);
        /* short is a signed data type */
        if (((*(i + pointer)) & 0x80) == 0x80)
                byte = (short) (128 + ((*(i + pointer)) & 0x7f));
        /* 2^24, 2^16, 2^8, 2^0, 2^-8, 2^-16, 2^-24, 2^-32 */
        base = pow((double)2, (double)(3 - i) * 8);
        timestamp = timestamp + (double) (byte * base);
}

return timestamp;
```

Testen Sie Ihre Implementierung von *parseNTPMessage()* mittels der vorgegebenen Funktion *int decodeTimestamp_test()*.

Implementierungshinweise parseNTPMessage

Die Struktur des NTP Paketes ist im Bild dargestellt. Eine Nachricht wird dabei als ein Array bestimmter Länge repräsentiert.



Genauere Informationen bezüglich des Aufbaus einer SNTP Message finden Sie unter `http://www.networksorcery.com/enp/protocol/sntp.htm`. Zum Versenden der SNTP Nachricht müssen folgende Felder gesetzt werden:

- Leap Indicator - LI
- Version Number - VN
- Mode
- Transmit Timestamp - ist nicht zwingend notwendig, empfohlen

Aufgabe 3.3: Roundtrip Delay und Local Clock Offset

Bestimmen Sie den „Roundtrip Delay“ und den „Local Clock Offset“ und geben Sie diese Daten aus. Nähere Informationen hierzu können Sie der RFC2030 entnehmen.

Implementierungshinweis

Die Uhrzeit des lokalen Rechners kann man durch den time() Aufruf ermittelt werden. Merken Sie die aktuelle Uhrzeit in der Variable localTimestamp vor dem Senden des (S)NTP Packetes und übergeben Sie diese Uhrzeit der Funktion createNTPMessage(). Merken Sie

die Empfangszeit des (S)NTP Paketes in der Variable localTimestamp, sie wird bei der späteren Berechnung gebraucht.

(S)NTP unter Windows und Linux haben eine UTC Zeitskala. Es gilt aber zu beachten, dass sie unterschiedliche „Epochen“ nutzen. NTP zählt die Zeit seit dem 1. Januar 1900, und Windows/Linux seit dem 1. Januar 1970. Um diese Zeitstempel umzurechnen müssen 2208988800 Sekunden addiert oder subtrahiert werden.

Wie sich der Round Trip Delay und Offset zur lokalen Uhr berechnen lassen, kann der RFC 2030 entnommen werden.