NVS Projekt

Alessandro Crispino

April 2020

Contents

Chapter 1

Einleitung

1.1 Projekt

Dieses Projekt wurde im Zuge der Aubesserung der NVS-Note durchgeführt. Die Aufgabe war es ein Programm zu entwickeln, welches das Verfahren von Cristian simuliert. Hierzu wurde ein Server implementiert, mit welchen sich mehrere Clients verbinden können. Der Server simulierte einen Zeitserver und synchronisierte in mehreren Abständen die verbundenen Clients. Wurde ein Client zum Dritten mal synchronisiert, beendet sich dieser anschließend selbständig.

1.2 Cristian

Einleitung

Wie oben schon erwähnt, handelt es sich beim Algorithmus von Cristian um ein Synchronisationsverfahren von physikalischen Uhren in verteilten Systemen. Dafür wird ein Zeitserver benötigt, mit welchem man sich verbinden kann.

Ablauf

Im folgenden Abschnitt wird der Ablauf genauer beschrieben.

- 1. Client erfragt die Zeit des Servers zum Zeitpunkt t_0.
- 2. Anfrage wird von Server verarbeitet. Dauert gewisse Zeitspanne: I.
- 3. Die Antowort $C_M(t_1)$ wird vom Client zum Zeitpunkt t_1 empfangen.
- 4. Client wird auf die Zeit $C_M(t_1)$ + RTT / 2 gesetzt.

Die RTT wird hierbei folgendermaßen berechnet: RTT = t_0 - t_1 - I.

Chapter 2

Verwendung

2.1 Server

Der Server dient als Zeitsynchronisationsserver. Mit ihn können sich beliebig viele Clients verbinden und Anfragen schicken. Er lässt sich mit dem Kommando "./server" starten und benötigt keine weiteren Parameter. Allerding lässt sich der Port des Servers mit dem Kommandozeilenparamter "-p" ändern. Standartmäßig wird hierfür der Port "9999"verwendet. Ausserdem wird mittels dem Kommando "-h, –help", eine Verwendungsansicht angezeigt. Um das ganze nocheinmal zusammenzufassen werden hier die Parameter aufgelistet:

- -p: Kofiguration des Ports, Integer erwartet
- -h, –help: Anzeige des Verwendungsmenüs

2.2 Client

Ein Client verbindet sich mit dem angegebenen Server, synchronisiert seine Zeit drei mal und beendet sich dann selbst. Der Port des Clients kann ausserdem auch konfiguriert werden. Außerdem muss beim Starten eines Clients, der Name des Clients mit dem Kommando "-n" angegeben werden. Zudem wird auch hier, wie beim Server, mittels "-h, –help" das Hilfemenü angezeigt. Die Paramter welche dem Client übergeben werden können sind:

- -n: Name des Clients, Text erwartet, erforderlich
- -p: Kofiguration des Ports, Integer erwartet
- -h, –help: Anzeige des Verwendungsmenüs

Chapter 3

Uhr

Um auf jedem Endpunkt eine Uhr simulieren zu können, wurd in diesem Fall ein header-only Modul “Clock“ erstellt. Dieses funktioniert wie eine normale Uhr, allerdings kann die “Schnelligkeit“ der Uhr angepasst werden. Es wurden folgende Methoden implementiert:

3.0.1 Attribute

step

- Gibt die Schnelligkeit der Uhr an
- Typ: int

curr_time

- beinhaltet die derzeit eingestellte Uhrzeit
- Typ: time_point

mtx

- dient zum Locken der einzelnen Method in der Klasse Clock
- Typ: mutex

3.0.2 Konstruktoren

- Clock(): Defaultkonstruktor, setzt die Schnelligkeit auf 1000ms und setzt curr_time auf den aktuellen Zeitpunkt
- Clock(int step): setzt die Schnelligkeit auf übergebenen Parameter und setzt curr_time auf den aktuellen Zeitpunkt

3.0.3 Methoden

operator()

In dieser Methode wird er operator “()“überladen. Hierbei wird eine Endlosschleife gestartet, in welcher die “step“ Zeit gewartet wird. Anschließend gelocked, die Zeit um eine Sekunde erhöht und die derzeitige Zeit ausgegeben wird.

set_time(int hours, int minutes, int seconds)

set_time erwartet sich drei Paramete vom Typ integer. Mittels dieser Parameter wird die “cur_time“ überschrieben. Der erste Parameter gibt die Stunden, der zweite die Minuten und der dritte die Sekunden an. Hier wird nichts zurückgegeben.

get_time()

Diese Methode gibt ein tuple zurück. Dieser besteht aus den Werten “hour“, “minutes“ und “seconds der curr_time. Diese Funktion gibt ein Tuple zurück.

to_time()

In dieser Funktion wird der Zeitpunkt “curr_time“ auf ein Objekt vom Typ “time_t“ konvertiert und anschließend auf den Typ “long“ gecasted. Dieser wird zum Schluss auch zurückgegeben.

from_time(time_t tmp)

Setzt die Variable “curr_time“ auf den übergebenen Zeitpunkt. Hierbei kann auch ein long übergeben werden. Diese Methode gibt nichts zurück.

Chapter 4

Server

CLI11

Zur Parameterbehandlung wurde beim Server CLI11 verwendet. Bei CLI11 handelt es sich um ein Befehlszeilenparser für C++ 11 welcher umfangreiche Funktionen mit einer einfachen und intuitiven Benutzeroberfläche bietet. Hierbei wurde, wie im Punkt 2.2 ersichtlich, ein Parameter "-p" hinzugefügt, welcher zur Konfiguration des Ports dient.

Clock

Danach wurde ein thread eines Clock Objekts gestartet, welcher der Simulation einer Uhr dient. Diese brauchte genau 1000ms für eine Sekunde. Ausserdem wurde der Thread mittels der Funktion "detach()" gestartet, um somit den anderen thread nicht zu blockieren.

eigentliche Aufgabe

Mittels asio wurde ein tcp::acceptor implementiert, welcher auf tcp::sockets im eigenem Netzwerk, auf dem angegebenen Port wartet. Hierfür wurde anschließend eine Endlosschleife gestartet, welche eingehende Verbindungen akzeptiert und jeweilige tcp::iostreams erstellt. Sobald ein Client in den Stream schreibt, wird diese Nachricht vom Server ausgegeben. Anschließend wartet der Server ein bis drei Sekunden, sendet dem Client die Wartezeit und danach seine derzeitige Zeit.

Chapter 5

Client

CLI11

Wie auch beim Server wurde auch beim Client CLI11 zur Kommandozeilenverarbeitung verwendet. Hier wurde allerdings zusätzlich zur Konfiguration des Ports noch eine Option hinzugefügt. Diese ist die Konfiguration des Clientnamen, angeben des Namens ist ausserdem verpflichtend.

Clock

Auch die Uhr wurde ähnlich wie die Uhr des Servers gestartet. Der einzige Unterschied lag allerdings am Tempo, zu Anschauungszwecken wurde die Uhr langsamer als die des Servers eingestellt.

eigentliche Aufgabe

Beim Client wurde die Uhrzeit drei mal synchronisiert. Ist dies geschehen hat sich der Client anschließen beendet. In dieser Schleife wurde ein tcp::iostream erstellt mit dem jeweiligen Netzwerk und dem angegebenen Port. Ist der Stream erfolgreich erstellt worden, hat der Client seinen Namen an den Server geschickt und anschließend den derzeitigen Zeitpunkt abgespeichert. Sobald eine Antwort in den Stream geschrieben worden ist, hat sich der Client die Wartezeit und die Serverzeit aus dem Stream geholt und abgespeichert. Anschließend wurde auch noch der eigene derzeitige Zeitpunkt abgespeichert. Danach wurde die neue Zeit, mittels dem Verfahren von Cristian (siehe 1.2) berechnet und für den Client eingestellt. Ist dies geschehen wurde der Stream geschlossen.
Zwischen den Synchronisationen wurde jeweils 10 Sekunden gewartet.
War der Client fertig mit der Synchronisation, hat dieser sich letztendlich selbst beendet.

Chapter 6

Zusätzliches

Im Zuge dieses Projekts sind ausserdem die header only Module “print.h“ und “random.h“ entstanden, welche im folgenden Kapitel genauer erklärt werden.

random.h

Dieses Modul wurde zur Generierung der Wartezeit verwendet. Dafür gibt es in diesem Modul die Funktion “getRandom(double min, double max)“ welche den Bereich, in welchem sich die Zufallszahl befinden soll, entgegen nimmt. Nach Berechnung wurde auch eine Zufallszahl vom Typ double zurückgegeben.

print.h

Bei dem Modul “print.h“ handelt es sich um eine Sammlung an Funktionen, welche eine Ausgabe mittels “spdlog“ ermöglichen, ohne eine Race Condition zu bekommen.
Hierbei wurden folgende Funktionen implementiert:

- inline std::ostream& operator«(std::ostream& out, decltype(std::chrono::system_clock::now()) time): Dieser überladene Operator erwartet sich einen ostream und bekommt einen ostream und einen time_point übergeben. Anschließend wird der übergebene time_point mittels der Funktion “put_time“ in den Stream geschrieben, welcher letztendlich zurückgegeben wird.
- std::string to_string(std::chrono::system_clock::time_point tp): In dieser Funktion wird ein string zurückgegeben und ein time_point übergeben. Dieser time_point wird mittels dem überladenen Operator in eine stringstream geschrieben und anschliessend als string zurückgegeben.
- void printer(std::chrono::system_clock::time_point time): Diese Funktion nimmt einen time_point entgegen und gibt diesen sicher mittels spdlog aus.

- void printer(std::string msg): Um nicht nur time_points auszugeben, wurde die Funktion auch mittels string Parameter implementiert.