Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Hausarbeit

Birger Kamp

Der wissende Aufzug

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Birger Kamp

Der wissende Aufzug

Hausarbeit eingereicht im Rahmen des Moduls Modellierung Dynamischer Systeme

im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer: Prof. Dr. Wolfgang Fohl

Eingereicht am: XXXXXX

Birger Kamp

Thema der Arbeit

Der wissende Aufzug

Stichworte

Aufzug, Fahrstuhl, Warteschlange, Scheduling

Kurzzusammenfassung

Abstract

Inhaltsverzeichnis

1 Einleitung

Man nimmt es als selbstverständlich an, dass ein Fahrstuhl kommt, wenn man auf den *Rufen*-Knopf gedrückt hat. Tut man dies in Gebäuden, in denen ein System mehrere Fahrstühle koordiniert, hat man mit dem bloßen Knopfdrücken einen komplexen Prozess gestartet. Die folgende Arbeit verschafft einen Einblick in die Ablaufplanung eines Fahrstuhls und was es dabei zu beachten gibt. Außerdem wird eine These gestellt und mit einem einfach gehaltenen Modell beantwortet.

1.1 These der Arbeit

Ein einfacher Aufzug setzt sich erst dann in Bewegung, wenn er eine Anfrage erhalten hat. Jemand der den *Rufen*-Knopf drückt, muss also warten bis der Fahrstuhl ihn abholt. Fahrstuhlsysteme sollten grundsätzlich bemüht sein, diese Wartezeit möglichst kurz zu halten.

Daraus leitet sich die These dieser Arbeit ab: "Wenn das Fahrstuhl-System wüsste, dass zum Zeitpunkt x in der Zukunft, eine Anfrage auf dem Stockwerk a gestellt wird, dann müsste der Anfragesteller kürzer warten". Ein Anwendungszenario wäre eine Hochschule bei der dem Fahrstuhlsystem bekannt ist, dass zur Pausenzeit um 9.30 Uhr auf Stockwerk 6 einige Personen zum Erdgeschoss fahren möchten.

1.2 Verwendete Metriken

Um die gestellte These anhand eines Modells evaluieren zu können, müssen einige Metriken bestimmt und verglichen werden.

Die These selbst nennt die *Wartezeit* in als Metrik. Um die Auswirkungen der veränderten Ablaufplanung beurteilen zu können, wird außerdem der *Anfragen-Durchsatz* und die *Fahrtzeit* gemessen. Mit Fahrtzeit wird die Zeit bezeichnet, die ein aufgenommer Fahrgast im Fahrstuhl verbringt bis er auf seinem Ziel-Stockwerk aussteigt.

2 Grundlagen

Hinter der Planung der Fahrstühle stecken Theorien und algorithmische Probleme, die im Folgenden erläutert werden.

2.1 Scheduling-Theorie

Die theoretische Grundlage für eine Fahrstuhlplanung ist die Scheduling-Theorie. In der Scheduling geht es laut Pinedo (2012) darum, dass der Zugriff auf eine Ressource zeitgesteuert auf die Anfragenden verteilt wird.

Ein Beispiel aus der Informatik sind Multi-Threading Prozessoren. Dabei sind die Threads diejenigen Objekte, die den Zugriff auf die Ressource Prozessor erfragen. Der Thread-Scheduler erlaubt jedem anfragenden Thread eine gewisse Zeitspanne den Prozessor zu benutzen. Anschließend erhält ein anderer Thread die Ressource für eine gewisse Zeit.

2.1.1 Online Probleme

Nach Manasse u. a. (1988) ist eine Fahrstuhlablaufplanung ein Problem, das in eine Unterart der Scheduling-Probleme einzuordnen ist, die *Online Probleme* genannt werden. Diese Probleme zeichnen sich dadurch aus, dass dem Scheduler zu einem Zeitpunkt nur eine Teilmenge aller anfallenden Anfragen bekannt ist. Zu jedem Zeitpunkt können weitere Anfragen dazu kommen. Daraus lässt sich folgern, dass der Scheduler nur bedingt optimal planen kann, da er nicht weiß welche Anfragen noch kommen werden.

2.1.2 Offline Probleme

Im Gegensatz zu den Online Problemen stehen die *Offline Probleme*. Diese sind ebenfalls eine Unterart der Scheduling-Probleme. Bei diesen Problemen sind dem Scheduler bereits alle anfallenden Anfragen bekannt, sodass der Scheduler optimal planen kann.

Ein Beispiel für diese Probleme sind Logistik-Unternehmen: Diese können bereits im Voraus die Fahrt der Lieferwagen für den nächsten Tag planen. Denn alle Lieferanfragen gehen bereits im Vortag ein.

3 Modellierung

Das folgende Kapitel beschreibt wie das Modell erstellt wurde und die getroffenen Annahmen und Vereinfachungen.

3.1 Verwendete Tools

Die Modellierung wurde in der Programmiersprache Python vorgenommen. Diese eignet sich besonders für wissenschaftliche Zwecke, da es eine breite Auswahl an Libraries in dieser Richtung gibt.

Für die Simulation von fortschreitender Zeit und Events zu bestimmten Zeitpunkten wurde die Library *SimPy*[1] verwendet.

Um die Ergebnisse in Diagrammen zu visualisieren wurde die Library *matplotlib*[2] verwendet. Diese Library bietet eine MATLAB-ähniche Schnittstelle an, sodass man ohne viel Zeitaufwand ein einfaches Linien-Diagramm konstruieren kann. Mit etwas mehr Aufwand sind auch komplexe 2D- und 3D-Diagramme möglich.

Außerdem wurde ein minimales Python-Interface in *curses*[3] geschrieben. Dies bietet zwar keinen Mehrwert in der Genauigkeit der Simulation, aber es bietet einen Überblick was während der Simulation passiert. Dadurch fallen einige Fehler auf als wenn man die Simulation blind ablaufen lassen würde, das Interface unterstützt daher bei der Fehlerfindung und Verfizierung des Planungsablauf.

3.2 Vereinfachungen und Annahmen

Dieses Modell sich auf das wesentliche des Fahrstuhl-Schedulings beschränken. Um den Aufwand der Modellierung im Rahmen einer Praktikumsaufgabe zu halten, wurden daher folgende Modellierungs-Entscheidungen getroffen:

1. Ein Fahrstuhl kann unendlich viele Personen beinhalten

[1] http://simpy.readthedocs.io/en/latest/
[2] http://matplotlib.org/
[3] https://docs.python.org/2/howto/curses.html

2. Es wird nicht beachtet ob und wieviele Personen bei einem Halt aussteigen
3. Obwohl *keine Personen* in den Fahrstühlen mitfahren, soll keine der Anfragen unnötig lange verzögert werden
4. Folgende Aktionen eines Fahrstuhls benötigen eine Zeiteinheit:
   - Türen öffnen
   - Türen schließen
   - Veränderung der Position um ein Stockwerk
5. Beim Rufen des Fahrstuhls kann der Fahrgast angeben, in welche Richtung er fahren möchte

Die Vereinfachungen 1 und 2 sind auf den ersten Blick nicht relevant für die Planung der Aufzüge, in einigen Fällen ist es jedoch relevant. In der Realität kann ein Fahrstuhl nur endlich viele Personen aufnehmen. Sobald eine konstruktionsbedingte Personen-Anzahl eines Fahrstuhls überschritten wurde, geben moderne Fahrstühle ein Warnsignal, dass der Fahrstuhl zu voll beladen ist. In dem Fall müssen einige Personen aussteigen und auf den nächsten Fahrstuhl warten. Bezogen auf die Ablaufplanung hat dies den Effekt, dass ein Fahrstuhl-Ruf im Scheduling nicht als *erledigt* markiert wird, sondern wieder als eine neue Fahrstuhl-Anfrage eingereiht wird. Da dies jedoch ein Spezialfall ist, wird er in diesem Modell nicht beachtet.

3.3 Das Modell

Das Wesentliche in diesem Modell ist, wann ein Aufzug in welchem Stockwerk hält und in welche Richtung er nach dem Halt weiterfährt. Abhängig davon werden dem Fahrstuhl Anfragen zugeordnet, die er zu bearbeiten hat.

Ein Fahrstuhl wird beschrieben durch:

- Die Queue Q der dem Fahrstuhl zugeordneten Anfragen
- Das Stockwerk $current_floor$ in dem sich der Fahrstuhl aktuell befindet
- Die Richtung r in der sich der Fahrstuhl bewegt
- Der Status $s \in S; S = \{"frei", "belegt"\}$ des Fahrstuhls

Alle vorhandenen Fahrstühle des Modells werden mit F bezeichnet.

Eine Anfrage wird beschrieben durch:

- Das Stock $start_floor$ auf dem nach dem Fahrstuhl gerufen wird
- Die Richtung r in die der Anfragende fahren möchte

In Abbildung 3.1 ist das erstellte Modell dargestellt. Es gibt eine Menge von zufällig generierten Calls, die jeweils zu einem bestimmten Zeitpunkt t an den *ElevatorScheduler* gemeldet werden. Dies ist in der Realität der Zeitpunkt, wenn eine Personen den Fahrstuhl per Knopfdruck ruft.

Der *ElevatorScheduler* prüft zunächst, ob einer der vorhandenen Fahrstühle frei ist oder, ob einer der Fahrstühle in die gewünschte Richtung fährt und auf dem Weg in die Richtung an dem Stockwerk des Rufs vorbeikommt. Beschrieben durch: Sei $f \in F$ der zu prüfende Fahrstuhl, c die neue Anfrage und $check_on_way$ eine Funktion die prüft ob ein Stockwerk in Fahrtrichtung eines Fahrstuhls liegt, dann sind die in Frage kommenden Fahrstühle F_f:

$$f_f \in F_f, F_f \subseteq F : s(f_f) = "frei" \vee (r(f) = r(c) \wedge check\_on\_way(f, c)) \tag{3.1}$$

Falls eine der Bedingungen zutrifft, wird die Anfrage in die aktuelle Queue des Fahrstuhls eingereiht. Es wird berechnet, wann die neue Anfrage von diesem Fahrstuhl bearbeitet wird. Anschließend wird geprüft, wie die neue Anfrage die anderen Anfragen des Fahrstuhls beeinflusst. Aus der Bearbeitungszeit k_d des Fahrstuhls und der Beeinflussung k_l der anderen Anfragen in der Queue wird ein Wert berechnet, der die Kosten k der Anfrage bezogen auf den spezifischen Fahrstuhl darstellt. Die neue Anfrage wird dem Fahrstuhl zugewiesen, der die geringsten Kosten aufweist. Beschrieben durch: Sei $q_n \in Q$ die Anfrage an n-ter Stelle in Q,$n = |Q(f)|$ und $k : k <= n$ die Position von c in Q:

$$k_d = \sum_{i=1}^{k} i = |q_i - q_{i-1}| \tag{3.2}$$

$$k_l = \sum_{i=k+1}^{n} i = |q_i - q_{i-1}| \tag{3.3}$$

$$k = k_d + k_l \tag{3.4}$$

Der Fahrstuhl selbst beinhaltet keine Planungslogik. Er fährt lediglich die Stockwerke ab, die in seiner Queue vom *ElevatorScheduler* eingereiht wurden.

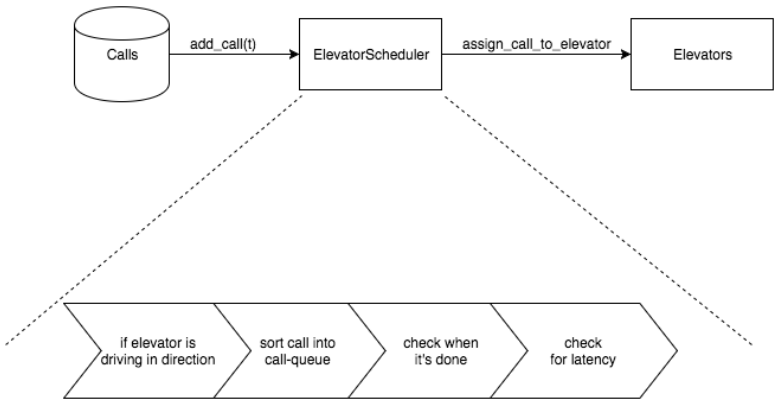

Abbildung 3.1: Modellierung einer Fahrstuhl-Ablaufplanung

4 Auswertung der Ergebnisse

4.1 Fahrstuhl ohne Wissen

4.2 Fahrstuhl mit Wissen

4.3 Analyse

5 Fazit

Literaturverzeichnis

[Manasse u. a. 1988] Manasse, Mark ; McGeoch, Lyle ; Sleator, Daniel: Competitive algorithms for on-line problems. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing* ACM (Veranst.), 1988, S. 322–333

[Pinedo 2012] Pinedo, Michael L.: *Scheduling: theory, algorithms, and systems.* Springer Science & Business Media, 2012