UNIVERSITÄT DUISBURG-ESSEN
FAKULTÄT FÜR INGENIEURWISSENSCHAFTEN
ABTEILUNG INFORMATIK UND ANGEWANDTE KOGNITIONSWISSENSCHAFT

Bachelorarbeit

An Algorithm for Dependency-Preserving Smart Home Updates

Koray Uzun
Matrikelnummer:3081690
Angewandte Informatik (Bachelor)

Fachgebiet Verteilte Systeme, Abteilung Informatik
Fakultät für Ingenieurwissenschaften
Universität Duisburg-Essen

23. August 2021

Erstgutachter: Prof. Dr-Ing. Torben Weis
Zweitgutachter: Peter Zdankin
Zeitraum: 1. September 2042 - 1. Januar 2043

Abstract

Heutzutage erfreuen sich Smart Home Systeme immer größerer Popularität. Ein Smart Home System ist ein Netzwerk von mehreren Geräten, meist innerhalb eines Wohnraumes. Wie jede andere Technologie auch, unterliegen Smart Homes einem ständigen Wandel. Dabei gibt es einen wichtigen Punkt zu beachten. Während andere Technologien, wie z.B. Smartphones oft einfach nach Jahren aufgrund ihres Alters ausgetauscht werden, wird von Smart Home Systemen eine lange Lebensdauer erwartet. Die meist komplexen und teuren Installationen sind nicht dazu ausgelegt alle paar Jahre ausgetauscht zu werden. Dennoch erwartet man, wie von anderen Technologien auch, dass die Systeme sich den Gegebenheiten der Zeit anpassen. Neue Funktionen oder oft auch Sicherheitslücken erfordern ein ständiges updaten der Systeme oder einzelner Geräte. Mit diesen Updates geht jedoch ein großes Risiko einher. Aufgrund der riesen Diversität verschiedener Smart Home Systeme kann vom Hersteller meist nicht garantiert werden, dass ein Update keine Schäden anrichtet. Lediglich das Update im Einzelnen wird auf mögliche Fehlerquellen überprüft, nicht jedoch das Zusammenspiel des Updates mit anderen Geräten. Dies wäre aufgrund der riesen Diversität an Smart Home Systemen für den Hersteller auch gar nicht umsetzbar. Dennoch benötig man eine Möglichkeit Updates präventiv zu überprüfen, um so größere Schäden zu meiden und die Langlebigkeit von Smart Home Systemen zu gewährleisten. Diese Abschlussarbeit beschäftigt sich mit der Implementation und Evaluation eines Algrorithmus, welcher Abhängigkeiten zwischen Geräten untersucht und basierend auf diesen Abhängigkeiten Updatekonfigurationen findet, die keine Probleme verursachen. Der Algorithmus basiert auf dem Paper "An Algorithm for Dependency-Preserving Smart Home Updates" von Peter Zdanking, Matthias Schaffeld, Marian Waltereit, Oskar Carl und Torben Weis.

[1]

[1] Wikipedia: `https://en.wikipedia.org/wiki/Abstract_(summary)`

Contents

Chapter 1

Einführung

In den letzten Jahren stieg die Anzahl der Smart Home Geräte stark. Laut Verified Market Research wird der Smart Home Markt im Jahr 2019 auf bereits 80 Milliarden Dollar geschätzt. Prognostiziert werden 207.88 Milliarden Dollar im Jahr 2027. Dies entspricht einer jährlichen Wachstumsrate von 13.52 Prozent. [**https://www.verifiedmarketresearch.com/produ**
Zudem ist es so, dass innerhalb von Smart Home Systemen die Anzahl der Geräte ebenfalls stetig steigt. Während 2016 durchschnittlich 5.8 Geräte über ein Smart Home System vernetzt waren, waren es 2018 schon bereits 8.1 Geräte. [**https://www.homeandsmart.de/**]
Es ist zu vermuten, dass sich dieser Trend vorerst so weiterentwickeln wird. Logischerweise steigt mit der Anzahl der Geräte in einem Smart Home System auch die Anzahl der Updates und somit insgesamt die Komplexität von Smart Home Systemen. Das Problem hierbei ist, dass mit steigender Komplexität zum einen automatisch eine höhere Fehleranfälligkeit bezüglich Updates einhergeht, zum anderen wird es schwieriger Fehler zu beheben, die durch Updates verursacht werden. Updates im Einzelnen besitzen bereits viele Fehlerquellen, wie zum Beispiel Sicherheitslücken oder generell fehlerhafte Implementationen. Dies sind jedoch Aspekte die bereits vom Entwickler weitestgehend vermieden werden. Viel Problematischer ist das Zusammspiel verschiedener Geräte und den dazugehörigen Updates. Ein an sich fehlerfreies Update kann bei der Installtion in einem Smart Home System, dennoch große Probleme verursachen. Was ist wenn zum Beispiel ein Update eine Funktion eines Geräts so verändert, dass ein anderes Gerät nicht mehr darauf zugreifen kann. Durch solche kleine Änderungen können Abhängigkeiten zwischen Geräten zerstört werden, wodurch ganze Teile eines Smart Home Systems ausfallen können. Solch ein Ausfall ist für den Nutzer meist ein großer finanzieller Schaden, da Updates oft nicht rückgängig gemacht werden können und somit Ersatz beschafft werden muss. Aus diesen Gründen sind präventive Maßnahmen notwendig, um die Langlebigkeit von Smart Home Systemen gewährleisten zu können. Langlebigkeit wird nämlich von den meisten Usern erwartet, wie man an der Resonanz zum Abschalten der HueBridge V1 von Philips sehen kann [**Paper [4]**]

1.1 Ziel der Arbeit

Ziel dieser Arbeit ist die Implementation und Evaluation eines Algorithmus, welcher Abhaengigkeiten zwischen Geraeten untersucht und basierend auf diesen Abhängigkeiten Updatekonfigurationen findet, die keine Probleme verursachen. Mit diesem Algorithmus wäre es möglich Updates bereits vor der Installation zu überprüfen und so sicherzugehen, dass sie keine Ausfälle verursachen. Problem hierbei ist, wie bereits erwähnt, dass es etliche Smart Home Anbieter mit verschiedener Software und Geräten gibt. Meist findet man selbst innerhalb eines Smart Home Systems bereits Geräte, die nicht vom selben Hersteller sind. Dies erschwert die Entwicklungs eines Algorithmus, welcher allgemein an allen Arten von Systemen anwendbar sein soll. Für die Implementation eines Algorithmus ist es daher notwendig gewisse Dinge vorauszusetzen und zu definieren. Zunächst die Definition eines Smart Homes:

[**Paper [12]**] A smart home has a set of devices that are connected through a platform. Each device has a certain software version and a set of available updates. Each software version has a set of available predefined services. Devices can use services of other devices which creates dependencies. An update configuration is one of the finite states of the nondeterministic finite automaton (NFA) that can be constructed by using the configurations as states and connecting them using individual updates as transitions.

Nun benötigt man noch eine Möglichkeit die Informationen über die Services und Updates eines Geräts leicht abfragen zu können. Eine solche Abfrage müsste in der Realität für jedes Gerät individuell angepasst werden. Dies ist jedoch nicht umsetzbar, weswegen es nötig ist einen einheitlichen Weg zu finden. Geräte besitzen üblicherweise Metadaten, welche Informationen über Firmware Version, Größe usw. enthalten. Diesen Metadaten könnte man zusätzlich Informationen über Services und Updates hinzufügen, sodass man einen schnellen Überblick über alle Geräte innerhalb eines Netzwerks erhalten kann.

Der entwickelte Algorithmus wird nicht an realen Smart Home Systemen getestet, weswegen sich diese Frage zunächst erübrigt. Es bieten sich zwei Möglichkeiten an den Algorithmus zu testen. Eine Möglichkeit ist eine statische Liste, welche man man manuell mit Geräten und deren Updates füllt, oder ein Generator, welcher abhängig von Parametern ein "künstliches" Smart Home System kreiert. Parameter wären dann zum Beispiel die Anzahl der Geräte oder die Anzahl der Updates, die ein einzelnes Gerät besitzt. Der Generator bietet sich mehr an, da der Algorithmus nicht nur getestet, sondern auch evaluiert werden soll. Die Evaluation kann dann an unterschiedlich komplexen Smart Home Systemen durchgeführt werden, indem man die Parameter dementsprechend verändert. Parameter wären zum Beispiel die Anzahl der Geräte, die Anzahl der Updates die ein Gerät besitzt oder auch die Anzahl der Services.

Chapter 2

Related Work

This chapter presents the basis on which the system and later evaluation are built. It usually contains short descriptions of previous research papers and puts them into the context of the thesis.

2.1 Code Listings

If there is some interesting code you would like to show in order to ease the understanding of the text, you can just include it using the `lstlisting` environment. Have a look at the source of this page to see how this is included:

```
x := from(42);
```

You could also put the code into an external file and include it in this document using the `lstinputlisting` command:

```
1 var x = new Node
2 if luck() {
3     var y = new Node
4     x.next = y
5 }
6 return
```

Be careful not to include large files as it hampers readability. If there is a short excerpt from a large file you would like to show, you can also extract an explicit range of lines from it without the need to modify the source file. This next listing only shows the conditional from the previous code:

```
2 if luck() {
3     var y = new Node
4     x.next = y
5 }
```

To use more advanced syntax highlighting have a look at the available options of the *listings* package or use the *minted* package[1], which has more extensive language support and additional themes. Both can be configured in the main file.

2.2 Math

In case you need to include some math, the *amsmath* package[2] is already included in this document.

To properly display some short formula like $e^{i\pi} = -1$, you can use the `\( \)` inline command. For larger formulas, the `math` environment is more appropriate. If you need to reference the formula multiple times, e.g. in case it is used in theorems, you should use the `equation` environment:

$$\vec{\nabla} \times \vec{B} = \mu_0 \vec{j} + \mu_0 \varepsilon_0 \frac{\partial \vec{E}}{\partial t} \tag{2.1}$$

To reference it as 2.1 using the `\ref{}` command, remember to use a `\label{}`.

2.3 Miscellaneous

You can use the `\todo{}` command to put obvious reminders on the side of the document.

TODO: like this!

[1] `https://texdoc.net/texmf-dist/doc/latex/minted/minted.pdf`
[2] `https://texdoc.net/texmf-dist/doc/latex/amsmath/amsmath.pdf`

Chapter 3

Evaluation

The evaluation usually consists of three main steps: first it defines the goals intended to be achieved by the software in detail; next is a description of the methodology used to measure the satisfaction of the software in relation to these goals; finally, the measurements are depicted and assessed.

Chapter 4

Conclusion

The conclusion quickly summarizes the results of the paper in relation to the previously defined goals and hypotheses. It usually also includes some information on next steps and further research that might be required or possible on the presented subject.

Versicherung an Eides Statt

Ich versichere an Eides statt durch meine untenstehende Unterschrift,

- dass ich die vorliegende Arbeit - mit Ausnahme der Anleitung durch die Betreuer - selbstständig ohne fremde Hilfe angefertigt habe und
- dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus fremden Quellen entnommen sind, entsprechend als Zitate gekennzeichnet habe und
- dass ich ausschließlich die angegebenen Quellen (Literatur, Internetseiten, sonstige Hilfsmittel) verwendet habe und
- dass ich alle entsprechenden Angaben nach bestem Wissen und Gewissen vorgenommen habe, dass sie der Wahrheit entsprechen und dass ich nichts verschwiegen habe.

Mir ist bekannt, dass eine falsche Versicherung an Eides Statt nach §156 und §163 Abs. 1 des Strafgesetzbuches mit Freiheitsstrafe oder Geldstrafe bestraft wird.

Duisburg, 23. August 2021
(Ort, Datum)

(Vorname Nachname)