

Wide-Area-Network Simulator

mit Flask/Bootstrap

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science in Engineering (BSc)
eingereicht am
Fachhochschul-Studiengang Internettechnik

Betreuer: DI Takashi Linzbichler

eingereicht von: Georg Markowitsch
Personenkennzahl: 1310418065

September 2016

**Assignment for the bachelor thesis of
Georg Markowitsch
Matr. no. 1310418065**

**Subject:
“Wide-Area-Network Simulator with Flask / Bootstrap”**

Zusammenfassung

Weltweit werden immer mehr Websites, Webservices und Webanwendungen über das World Wide Web angeboten. Auch der Datenverkehr wächst immer weiter, da immer mehr Menschen über Smartphone oder Personal Computer (PC) Zugriff erhalten. Das bedeutet, dass man als Entwickler nicht nur von optimalen Verbindungen ausgehen kann, sondern seine Website, Webservice oder Webanwendung auch an schlechte Verbindungsverhältnisse anpassen sollte um so eine größere Anzahl an Benutzer zu sichern. Um solche schlechten Verbindungsverhältnisse zu simulieren gibt es in jedem Betriebssystem verschiedene Wege und Programme, die teilweise recht kompliziert zu konfigurieren sind.

Ziel dieser Bachelorarbeit ist es daher, einen eigenständigen Netzwerksimulator als externe Hardware, die man einfach zwischen PC oder Laptop und Server oder Internetverbindung schließt und den Datenverkehr manipuliert, zu verwirklichen.

Im Zuge dessen wurden bereits einige Komponenten ausgewählt. So wird als Plattform ein Raspberry Pi dienen. Der Raspberry Pi besitzt bereits einen Netzwerkanschluss, der zweite benötigte Anschluss wird einfach über einen Universal Serial Bus (USB) zu Local Area Network (LAN) Adapter an einem der USB Anschlüsse am Raspberry Pi hinzugefügt. Zur Verwaltung der Verbindung wird ein sieben Zoll Touch Display direkt am Raspberry Pi verwendet auf dem ein lokales Webuserinterface mittels Flask zur Verfügung gestellt wird. Flask ist ein Python Microframework zum Bereitstellen eines Webservers.

Ehrenhausen a.d.W., 27.04.2016

Betreuer:

DI Takashi Linzbichler

Georg Markowitsch

Abstract

More and more websites, web services and web applications are offered over the World Wide Web and the number keeps increasing. Because of that, the worldwide traffic also grows, as more and more people get access via Smartphones or PCs. This means for developers, that they cannot assume that every user has a perfect connection to their product and it should be optimized to still work properly even in bad connection conditions. There are already a number of ways to simulate such connection condition on every operating system, but some of them are quite complicated to configure and handle.

The aim of this thesis is therefore to realize a stand-alone network simulator as external hardware, which is simply connected between a PC and the server to manipulate the data traffic conditions.

With this aim in mind a few components have been pre-selected. As the hardware platform, the popular Raspberry Pi. The Raspberry Pi already has one network socket and the second will be supplied with a simple USB to LAN adapter to one of the USB sockets on the Raspberry Pi. To manage the network conditions, a seven-inch touch screen is used directly in connection with the Raspberry Pi to display a webuserinterface provided with Flask. Flask is a Python Micro Framework for providing a web server.

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende Bachelorarbeit/Diplomarbeit/Masterarbeit selbstständig angefertigt und die mit ihr verbundenen Tätigkeiten selbst erbracht habe. Ich erkläre weiters, dass ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle aus gedruckten, ungedruckten oder dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte sind gemäß den Regeln für gutes wissenschaftliches Arbeiten zitiert und durch Fußnoten bzw. durch andere genaue Quellenangaben gekennzeichnet.

Die vorliegende Originalarbeit ist in dieser Form zur Erreichung eines akademischen Grades noch keiner anderen Hochschule vorgelegt worden. Diese Arbeit wurde in gedruckter und elektronischer Form abgegeben. Ich bestätige, dass der Inhalt der digitalen Version vollständig mit dem der gedruckten Version übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Georg Markowitsch

Ort, Datum

Inhaltsverzeichnis

1	Einleitung	1
1.1	Wide-Area-Network	1
1.2	Ziel dieser Bachelorarbeit	1
2	Hardware Komponenten	2
2.1	Raspberry Pi 2	2
2.2	Raspberry Pi Touch Display	4
3	Software Komponenten	5
3.1	Python	5
3.2	Flask	6
3.3	Bootstrap	6
4	Benutzeroberfläche	8
4.1	Übersicht	8
4.2	Netzwerkbrücke	10
4.3	Netzwerkgeschwindigkeit	11
4.4	Latenz	12
4.5	Paketverlust	13
4.6	Paketduplicierung	14
4.7	Paketverfälschung	15
5	Webservice	16
5.1	Flask	16
5.1.1	Initialisierung	16
5.1.2	Routen	17
5.1.3	HTML-Templating System	17
5.1.4	Rendern	18
6	Umsetzung	19
6.1	Netzwerkbrücke	19
6.2	Netzwerkmanipulation	19
7	Versuch	20

7.1	Versuchsaufbau	20
7.2	Tests	20
8	Ergebnisse	21
	Quellenverzeichnis	23

Tabellenverzeichnis

Abbildungsverzeichnis

2.1	Raspberry Pi Modell 2, Quelle: <i>Raspberry Pi Model 2</i>	2
2.2	Raspberry Pi Touch Display, Quelle: <i>Raspberry Pi touch display</i>	4
3.1	Python Logo	5
3.2	Flask Logo	6
3.3	Bootstrap Logo	7
4.1	Übersicht bei konfigurierter Netzwerkbrücke	9
4.2	Übersicht ohne Netzwerkbrücke	9
4.3	Konfiguration der Netzwerkbrücke	10
4.4	Konfiguration der Netzwerkgeschwindigkeit	11
4.5	Konfiguration der Latenz	12
4.6	Konfiguration des Paketverlust	13
4.7	Konfiguration der Paketduplicierung	14
4.8	Konfiguration der Paketverfälschung	15

Einleitung

1.1 Wide-Area-Network

Unter dem Begriff Wide-Area-Network, zu Deutsch Weitverkehrsnetz, versteht man ein Netzwerk welches zur Sprach- oder Datenübertragung über sehr weite Strecken konzipiert ist. Diese Netze sind weltweit flächendeckend aufgebaut und können unbegrenzt für verschiedene Zwecke der Kommunikation verwendet werden. Weitverkehrsnetze können eine Ausdehnung von mehreren 1.000 Kilometern bis zu 10.000 Kilometern haben.

Aufgrund dieser großen Ausdehnung kann es zu einer Verschlechterung der Übertragungsrate und anderen Werten in der Netzwerkkommunikation kommen, die die Verbindung negativ beeinflussen.

1.2 Ziel dieser Bachelorarbeit

Ziel dieser Bachelorarbeit ist es daher, einen eigenständigen Netzwerksimulator als externe Hardware, die man einfach zwischen PC oder Laptop und Server oder Internetverbindung schließt und den Datenverkehr manipuliert, zu verwirklichen.

Kapitel 2

Hardware Komponenten

Genauerer Einblick in die für diese Bachelorarbeit ausgewählten Hardware Komponenten.

2.1 Raspberry Pi 2

Ein Raspberry Pi ist ein Miniatur Computer im Scheckkartenformat der von der britischen Raspberry Pi Foundation entwickelt wurde. Die erste Version kam Anfang 2012 auf den Markt.

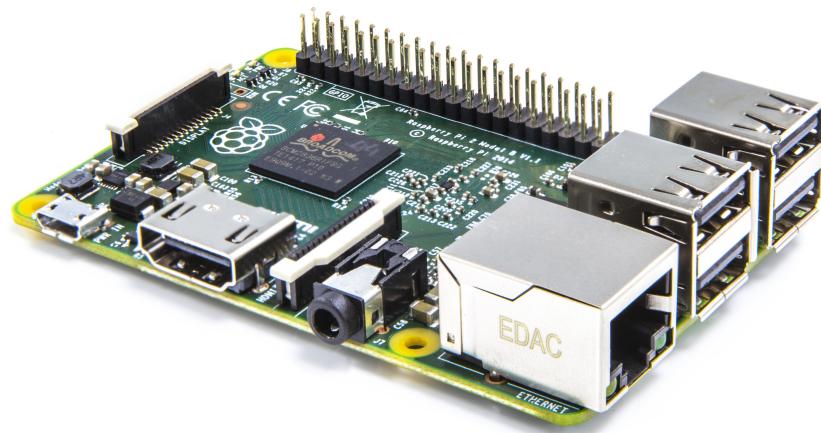


Abbildung 2.1: Raspberry Pi Modell 2, Quelle: *Raspberry Pi Model 2*

Das primäre Ziel der Raspberry Pi Foundation ist es, einen preislich günstigen aber trotzdem leistungsstarken Computer für vielseitige Einsatzbereiche herzustellen. Der

Raspberry Pi kann beinahe alles was ein herkömmlicher PC auch kann, angefangen von Textverarbeitung, Internetbrowsen bis hin zum Abspielen von High Definition Videos.

Zum Betreiben eines Raspberry Pi wird, wie auch bei einem herkömmlichen PC zusätzlich eine Maus, eine Tastatur und ein Monitor oder TV-Gerät mit einem High Definition Multimedia Interface (HDMI) Anschluss und eine Secure Digital (SD)-Karte oder bei den neueren Modellen eine microSD-Karte auf der sich das Betriebssystem befindet, benötigt.

Zusätzlich zu den normalen Funktionen und Schnittstellen eines PC's, hat ein Raspberry Pi General Purpose Input Output (GPIO)-Schnittstellen die frei programmierbar sind. Diese Schnittstellen ermöglichen es über den Raspberry Pi verschiedenste Aktoren zu steuern oder Sensoren auszulesen.

Die verschiedenen Versionen des Raspberry Pi unterscheiden sich im allgemeinen durch Prozessorleistung, Größe des Arbeitsspeichers und Anzahl der Schnittstellen. Für diese Bachelorarbeit wurde der Raspberry Pi 2 gewählt, da er für diese Bachelorarbeit an angemessener Leistung verfügt.

Quelle: *What is a Raspberry Pi*

2.2 Raspberry Pi Touch Display

Das Raspberry Pi Touch Display ist eine eigens für den Raspberry Pi angefertigter Touchscreen der von der Raspberry Pi Foundation entwickelt wurde.

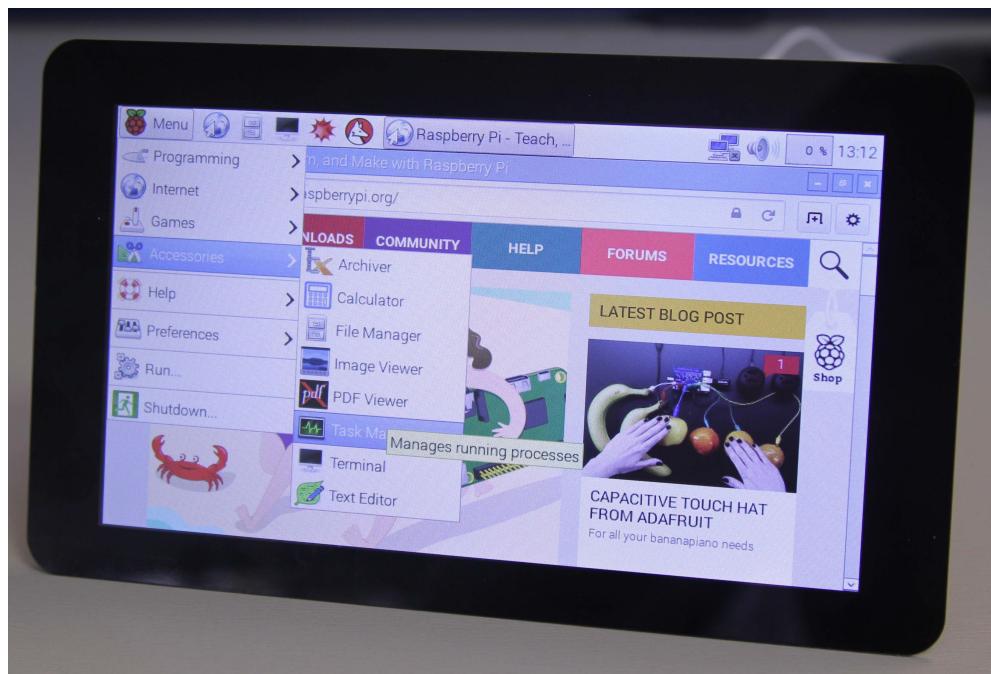


Abbildung 2.2: Raspberry Pi Touch Display, Quelle: *Raspberry Pi touch display*

Das Display selbst ist 7 Zoll groß und verfügt über eine Auflösung die mit 800x480 Pixel nicht besonders groß ist, dafür aber den Preis senkt. Da das Display von der Raspberry Pi Foundation selbst unterstützt wird, reicht zum Betrieb eine einfache Aktualisierung des Betriebssystem im Gegensatz zu Touchscreen von Drittherstellern.

Um das Display mit dem Raspberry Pi benutzen zu können muss lediglich ein flaches Display-Kabel verbunden werden. Nun hat man die Möglichkeit das Display über die GPIO-Schnittstellen oder über ein eigenes Netzteil über Micro-USB mit Strom zu versorgen. Der Raspberry Pi kann auch gleich an der Rückseite des Displays festgeschraubt werden.

Quelle: *Raspberry Pi touch display product information*

Kapitel 3

Software Komponenten

Genauerer Einblick in die für diese Bachelorarbeit ausgewählten Software Komponenten.

3.1 Python

Python ist eine Objektorientierte Programmiersprache, vergleichbar zu Perl, Ruby oder auch Java.

Python bietet unter anderem folgende Funktionen:

- Plattformunabhängigkeit
- Open-Source
- leicht zu lesende Syntax, besonders gut geeignet zur Entwicklung von Prototypen
- eine große Anzahl an Standard-Bibliotheken für alle gängigen Programmieraufgaben, wie zum Beispiel das Schreiben und Lesen von Dateien, Volltextsuche mit Regular Expressions
- Interaktiver Modus zum einfachen und schnellen Testen von Code-Ausschnitten
- bereits in C oder C++ compilierte Module können einfach in Python integriert und verwendet werden



Abbildung 3.1:
Python Logo

- Unterstützung objektorientierter Programmierung mit Klassen und Mehrfachvererbung
- eine Vielzahl an Basisdatentypen wie zum Beispiel, Zeichenketten (strings), Listen (lists) und Wörterbücher (dictionaries)
- Code kann einfach in Module und Pakete aufgeteilt werden
- Java-ähnliche Fehlerbehandlung
- automatische Speicherverwaltung
- Interpreter-Sprache, Code wird zur Laufzeit in Maschinencode übersetzt und ausgeführt

Quelle: *Python Overview*

3.2 Flask

Die Autoren von Flask nennen es ein micro Web-Framework, also eine Sammlung an Tools zum Bereitstellen einer Webanwendung inklusive Service, Ressourcen und API. Flask ist in Python geschrieben und basiert auf dem Werkzeug toolkit und der *Jinja2* Template-Engine.

Flask wird micro framework genannt, weil es Entwicklern die es verwenden nicht vorschreibt welche Tools oder Werkzeuge dieser für bestimmte Funktionen verwenden muss. Stattdessen gibt es eine große Anzahl an Flask-Erweiterungen die von Flask unterstützt werden. Somit können beliebig weitere Funktionen hinzugefügt werden. Die Integration der Erweiterungen funktioniert sogar so gut, dass man oft glaubt, sie wären in Flask selbst implementiert.



Abbildung 3.2:
Flask Logo

Quelle: *Flask*

3.3 Bootstrap

Bootstrap ist ein Open-Source Front-End Web-Framework, also ein Rahmenwerk zum Erstellen von Benutzeroberflächen für Webseiten. Es beinhaltet HyperText Markup Language (HTML) und Cascading Style Sheets (CSS) basierte Design Vorlagen für Formulare, Buttons, Navigation und viele weitere Oberflächen Komponenten, sowie optional JavaScript Erweiterungen.

Der große Vorteil durch die Benutzung von Bootstrap besteht darin, dass man mit einer einzigen Code-Basis auf den verschiedensten Geräten die optimale Ansicht bekommt. Egal ob Smartphone, Tablet oder Desktop-PC, Bootstrap skaliert die Benutzeroberfläche automatisch auf die perfekte Größe.



Abbildung 3.3:
Bootstrap
Logo

Quelle: *Bootstrap*

Kapitel **4**

Benutzeroberfläche

Bei der Benutzeroberfläche wurde speziellen Wert darauf gelegt, alle Funktionen und Daten übersichtlich auf dem doch eher kleinen Touchscreen des Raspberry Pi anzuzeigen. So wurden die verschiedenen Einstellungen auf eigene Webseiten ausgelagert und Einstellungen bei denen man Zahlenwerte eingeben muss mit Schiebereglern versehen.

Als Basis für die Benutzeroberfläche Diente eine Vorlage von *Bare*, welche später an die in dieser Bachelorarbeit benötigten Anforderungen angepasst wurde.

4.1 Übersicht

In der Übersicht wird der aktuelle Status der Netzwerkbrücke angezeigt. Ist eine Netzwerkbrücke konfiguriert, so werden hier der Name und die in der Brücke enthaltenen Netzwerkschnittstellen aufgelistet. Zusätzlich gibt es noch eine Anzeige der von Netzwerkinformationen zu Übertragungsgeschwindigkeit und gesendeten/verlorenen Paketen.

Sollte keine Netzwerkbrücke konfiguriert sein, so wird das ebenfalls hier angezeigt.

The screenshot shows a user interface for monitoring network traffic. At the top, there is a navigation bar with tabs: Overview, Bridge, Network Speed, Latency, Loss, Duplication, and Corruption. The 'Overview' tab is currently selected. Below the navigation bar, the title 'Overview of current Bridge' is displayed. Underneath the title, detailed network statistics are listed:

- Current Bridge: br0
- Contained Interfaces: eth0, eth1,
- Network Stats:
 - Kilobytes sent: 2351kb
 - Kilobytes received: 86211kb
 - Packets sent: 32184
 - Packets received: 60411
 - Errors total: 0
 - Dropped packets total: 0

Abbildung 4.1: Übersicht bei konfigurierter Netzwerkbrücke

The screenshot shows a user interface for monitoring network traffic. At the top, there is a navigation bar with tabs: Overview, Bridge, Network Speed, Latency, Loss, Duplication, and Corruption. The 'Overview' tab is currently selected. Below the navigation bar, the title 'Overview of current Bridge' is displayed. A message 'No Bridge currently available!' is shown.

Abbildung 4.2: Übersicht ohne Netzwerkbrücke

4.2 Netzwerkbrücke

Die Konfiguration erfolgt über einfaches auswählen der Netzwerkschnittstellen die verbunden werden sollen. Hier werden ausschließlich kabelgebundene Netzwerke zur Auswahl angezeigt. Hat man alle Netzwerkschnittstellen die man verbinden möchte ausgewählt, so werden beim Drücken von *Create Bridge* eine Netzwerkbrücke erstellt und die zuvor ausgewählten Netzwerkschnittstellen hinzugefügt.

Durch *Delete Bridge* kann eine bestehende Netzwerkbrücke nach kurzer Bestätigung wieder aufgelöst werden. Hierbei werden auch alle anderen Einstellungen zurückgesetzt.

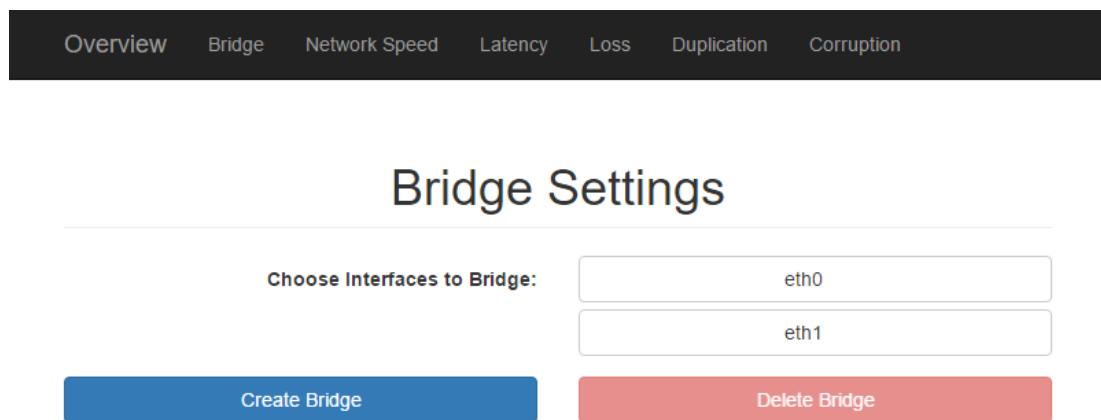


Abbildung 4.3: Konfiguration der Netzwerkbrücke

4.3 Netzwerkgeschwindigkeit

Hier kann man die maximale Download und/oder Upload Geschwindigkeit die über die Netzwerkbrücke verfügbar sein soll einstellen. Beide Konfigurationswerte werden in der Einheit Kilobits¹ pro Sekunde dargestellt.

Die Upload Geschwindigkeit kann mit einem Wert zwischen null und 10.000, und die Download Geschwindigkeit zwischen null und 50.000 Kilobits pro Sekunde eingestellt werden.

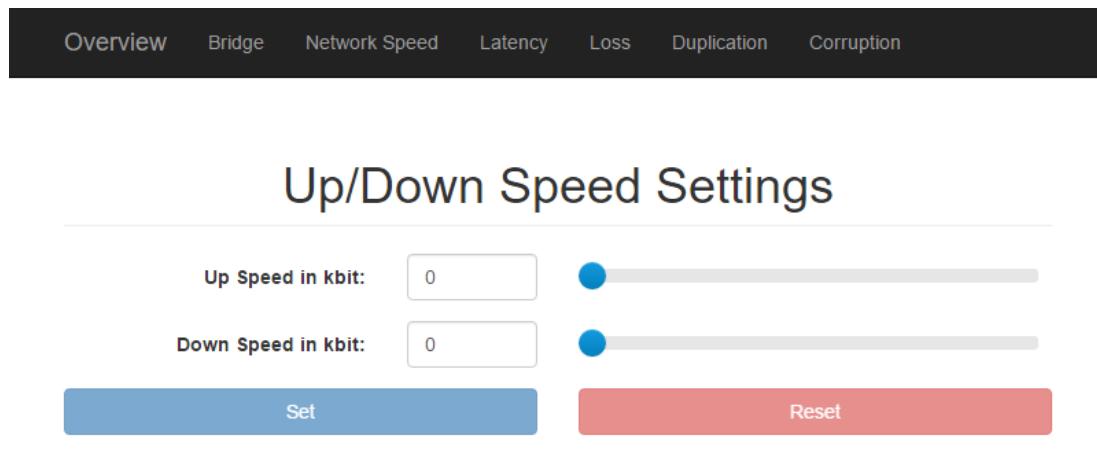


Abbildung 4.4: Konfiguration der Netzwerkgeschwindigkeit

¹ein Bit ist die kleinste digitale Informationseinheit, kann die Werte 0 oder 1 einnehmen, ein Kilobit sind 1024 Bits

4.4 Latenz

Hier gibt es drei Parameter die eingestellt werden können:

- die Latenz¹ in Millisekunden (0-2000ms)
- der Jitter² der Latenz in Millisekunden (+/- 0 bis 250ms)
- und eine Abhängigkeit in Prozent

Alle Parameter können über Schieberegler am Touchscreen oder falls vorhanden über eine Tastatur eingestellt werden.

Die Latenz kann mit einem Wert zwischen null und 2000 Millisekunden (2 Sekunden) eingestellt werden.

Die Variation wird zufällig generiert und kann so die Latenz mit Plus/Minus null bis 250 Millisekunden beeinflussen.

Der letzte Parameter bildet eine Abhängigkeit zwischen den generierten Werten. So kann man zum Beispiel einstellen, dass die nächste generierte Latenz mit Variation zu 25 Prozent von der vorherigen abhängt.

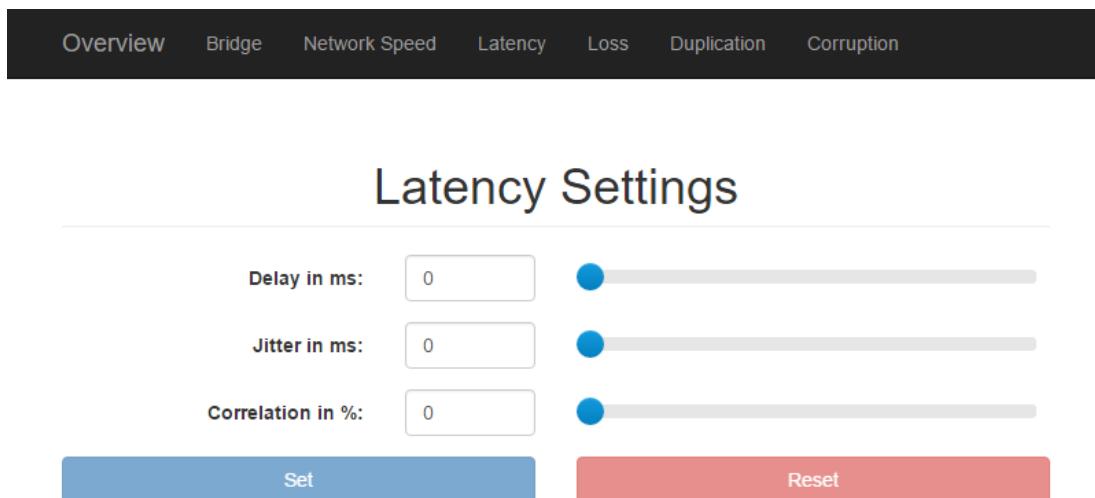


Abbildung 4.5: Konfiguration der Latenz

¹auch Verzögerungszeit, im Netzwerkbereich versteht man darunter die Übertragungszeit eines Netzwerkpakets von Start(z.B. Server) bis Ziel(z.B. Client)

²zufällige Schwankung

4.5 Paketverlust

Der Paketverlust¹ kann hier in Prozent zwischen null und zehn eingestellt werden. Zusätzlich gibt es wieder eine Option zur Abhängigkeit zwischen den Paketen über die *Correlation*. Stellt man hier zum Beispiel fünfzig Prozent ein, so hat das Nächste Datenpaket nach einem Paketverlust eine fünfzig prozentige Chance ebenfalls verloren zu gehen.

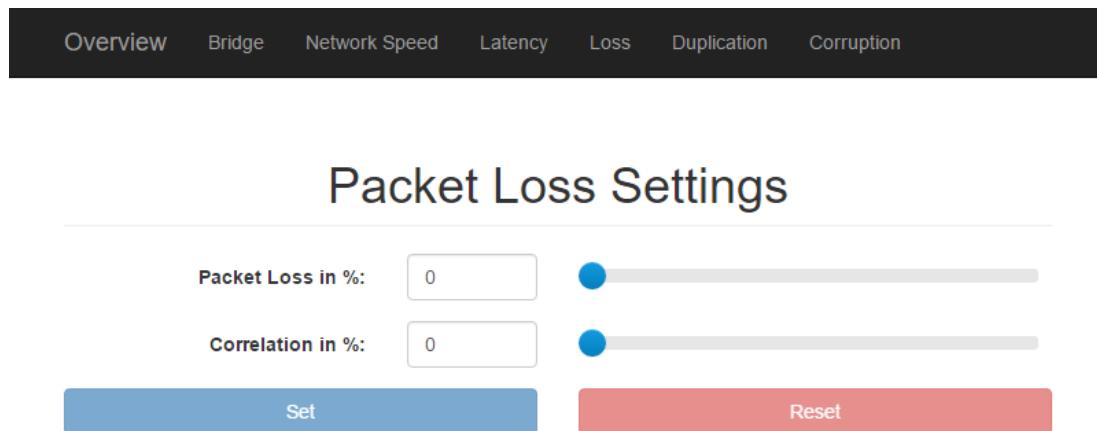


Abbildung 4.6: Konfiguration des Paketverlust

¹Pakete die bei der Datenübertragung verloren gehen

4.6 Paketduplicierung

Die Paketduplicierung¹ kann zwischen null und 100 Prozent eingestellt werden. Damit die Chance in der eine Paketduplicierung auftreten kann manipuliert. Zusätzlich gibt es wieder eine Option zur Abhängigkeit zwischen den Paketen über die *Correlation*. Stellt man hier zum Beispiel fünfzig Prozent ein, so hat das Nächste Datenpaket nach einer Paketduplicierung eine fünfzig prozentige Chance ebenfalls dupliziert zu aufzutauchen.

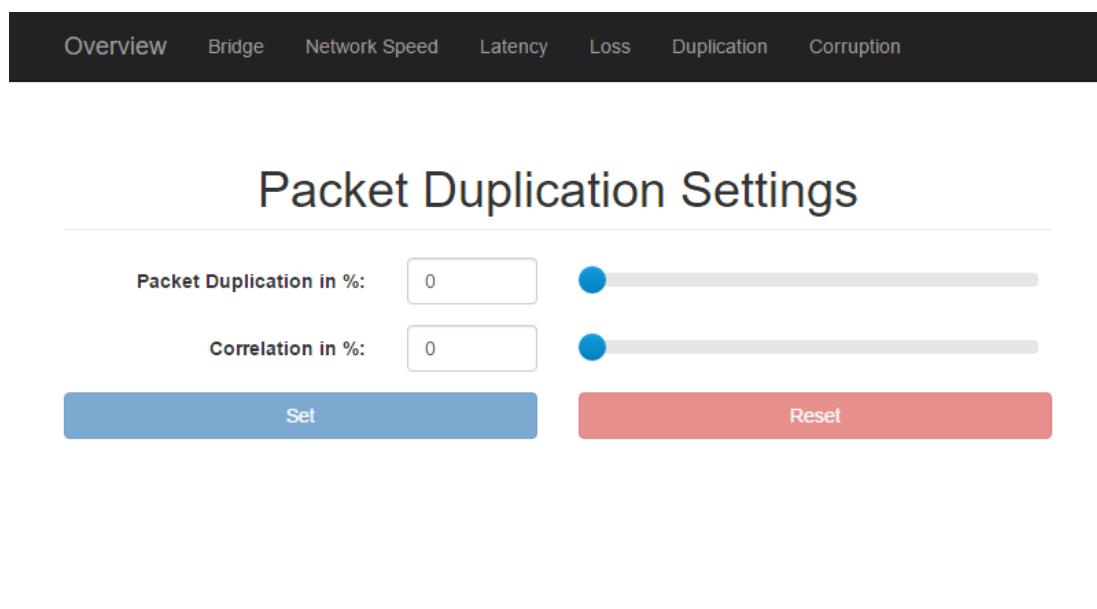


Abbildung 4.7: Konfiguration der Paketduplicierung

¹dasselbe Netzwerkpaket wird zwar nur einmal gesendet aber öfter erhalten

4.7 Paketverfälschung

Bei der Paketverfälschung wird ein einziges zufälliges Bit¹ im Datenpaket verfälscht. Die Chance zur Paketverfälschung kann zwischen null und zehn Prozent eingestellt werden. Zusätzlich gibt es wieder eine Option zur Abhängigkeit zwischen den Paketen über die *Correlation*. Stellt man hier zum Beispiel fünfzig Prozent ein, so hat das Nächste Datenpaket nach einer Paketverfälschung eine fünfzig prozentige Chance ebenfalls verfälscht zu werden.

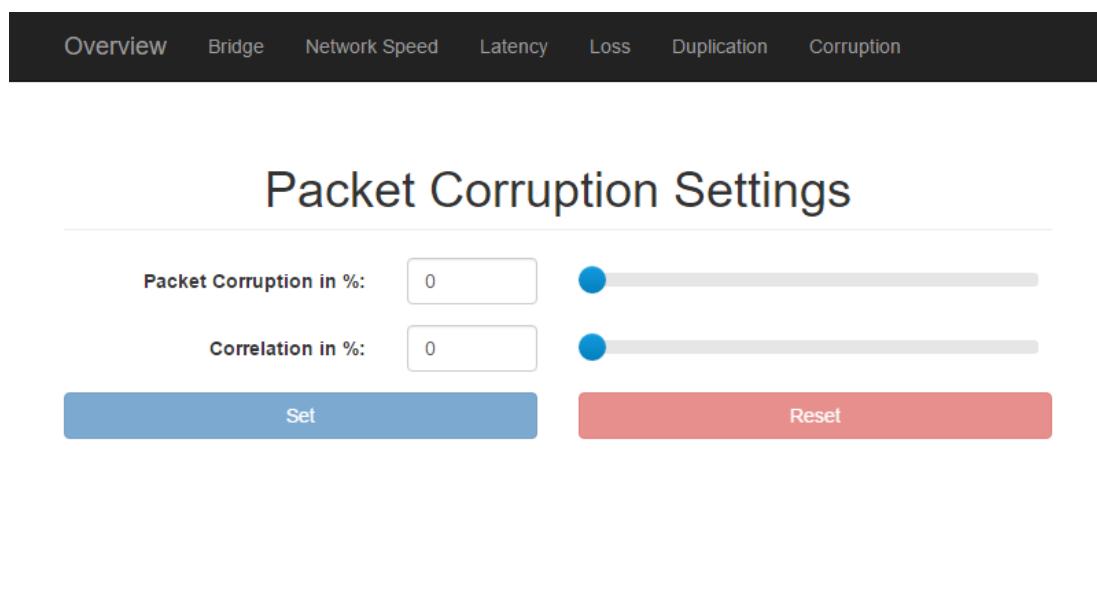


Abbildung 4.8: Konfiguration der Paketverfälschung

¹ kleinste digitale Informationseinheit, kann die Werte 0 oder 1 einnehmen

Kapitel 5

Webservice

5.1 Flask

Flask übernimmt in dieser Bachelorarbeit alle Funktionen eines Webservices. Es werden mit Flask alle Routen zu bereitgestellten HTML-Vorlagen oder dynamisch erzeugten HTML-Inhalten bereitgestellt.

5.1.1 Initialisierung

Folgender Code initialisiert in dieser Bachelorarbeit den Flask Webservice.

```
1 app = Flask(__name__)
2 app.config.from_object(__name__)
3
4 # cleans up network configuration if necessary before starting up
5 # the webservice
6 @manager.command
7 def runserver():
8     # register cleanup function atexit of program
9     atexit.register(cleanup)
10    cleanup()
11    app.run()      #starts flask webservice
12
13 #this is executed when the file is started with python
14 #it starts up the manager and the manager checks for extra
15 #parameters
16 #available paramaters:
17 #    runserver    #will start the flask webserver
```

```
16 if __name__ == "__main__":
17     manager.run()
```

Hier wird ein Manager von der Erweiterung Flask-Script verwendet. Dieser Manager wird als Startpunkt festgelegt, um vor dem Start des Webservice noch weitere Funktionen ausführen zu können. So werden beispielsweise die Netzwerkschnittstellen über die Funktion `cleanup()` im Falle eines unsauberer Absturzes zurückgesetzt.

Zum Start des Webservice gibt man in einer Kommandozeile im Ordner in dem sich die Projektdaten befinden folgende Befehle ein.

```
$ python start.py runserver
```

Dadurch wird die Funktion `runserver()` vom Manager gestartet. In `runserver()` wird dann der Flask Webservice mit den Standardparametern, `host=127.0.0.1` und `port=5000` gestartet.

5.1.2 Routen

Routen können durch das hinzufügen des Decorators¹ `@app.route('/route-name')` vor einer beliebigen Funktionen definiert werden.

```
1 # route to bridge creation
2 @app.route('/bridge', methods=['GET'])
3 def create():
4     return render_template('bridge.html', devs=getIfs(),
5                           bridgeActive=bridgeActive)
```

In diesem Beispiel wird eine Route zur Erstellung der Netzwerkbrücke festgelegt, die zunächst eine Liste aller Netzwerkschnittstellen über die Funktion `getIfs()` holt. Diese Liste wird dann mit der Funktion `render_template()` dynamisch in die entsprechende HTML-Vorlage eingebaut.

5.1.3 HTML-Templating System

Templating Systeme ermöglichen es Platzhalter für Daten in eine HTML-Vorlage einzubauen. Diese Platzhalter werden erst beim Aufruf der jeweiligen Webseite von der Templating-Engine mit den den aktuellen Daten ersetzt. Die Vorlage selbst enthält daher den gemeinsamen Stil und alle visuellen Elemente die dargestellt werden sollen.

¹ermöglicht das dynamische Hinzufügen von Fähigkeiten zu einer Klasse oder Funktion

Durch Templating-Engines kann man außerdem Logik mithilfe von Schleifen und Wenn/Dann-Statements in eine Vorlage integrieren. Dies ermöglicht etwa eine dynamische Größe bei der Darstellung von Daten in einer Tabelle.

Erst durch die Kombination von Daten und Vorlage durch die Templating-Engine erhält man die vollständige Webseite.

5.1.4 Rendern

In diesem Projekt wurde die in *Flask* enthaltende Templating-Engine *Jinja2* verwendet.

Nachfolgend ein Beispiel zum dynamischen Aufbau beliebig vieler Buttons auf Checkbox-Basis, die eine Netzwerkschnittstelle enthalten.

```
1 {%- for dev in devs %} 
2     <div class="btn-group btn-block" data-toggle="buttons">
3         <label class="btn btn-default btn-block">
4             <input type="checkbox" name="check" value="{{ dev }}"
5                 autocomplete="off">{{ dev }}
6             <span class="glyphicon glyphicon-ok"></span>
7         </label>
8     </div>
9 {%- endfor %}
```

Kapitel

6

Umsetzung

todo

6.1 Netzwerkbrücke

todo brctl und python brctl modul

6.2 Netzwerkmanipulation

todo netem todo tc

Kapitel **7**

Versuch

todo

7.1 Versuchsaufbau

todo

7.2 Tests

Kapitel 8

Ergebnisse

todo

Acronyme

HDMI High Definition Multimedia Interface

GPIO General Purpose Input Output

PC Personal Computer

SD Secure Digital

LED Light-Emitting Diode

USB Universal Serial Bus

I2C Inter Integrated Circuit Bus

SPI Serial Peripheral Interface

HTML HyperText Markup Language

CSS Cascading Style Sheets

BSD Berkeley Software Distribution

DIP Dual In-Line Package

MHz Megahertz

IDE Integrated Development Environment

HTTP Hypertext Transfer Protocol

LAN Local Area Network

Quellenverzeichnis

- Armin Ronacher (2016a). *Flask*. Verfügbar von: <<http://flask.pocoo.org/>> [Feb. 2016].
- (2016b). *Jinja2*. Verfügbar von: <<http://jinja.pocoo.org/>> [Feb. 2016].
- Bootstrap Team und Contributors (2016). *Bootstrap*. Verfügbar von: <<http://getbootstrap.com/>> [Feb. 2016].
- Raspberry Pi Foundation (2016a). *Raspberry Pi Model 2*. Verfügbar von: <https://www.raspberrypi.org/wp-content/uploads/2015/01/Pi2ModB1GB_-comp.jpeg> [Feb. 2016].
- (2016b). *Raspberry Pi touch display*. Verfügbar von: <<https://www.raspberrypi.org/wp-content/uploads/2015/09/front-centred.jpg>> [Aug. 2016].
- (2016c). *Raspberry Pi touch display product information*. Verfügbar von: <<https://www.raspberrypi.org/products/raspberry-pi-touch-display/>> [Aug. 2016].
- (2016d). *What is a Raspberry Pi*. Verfügbar von: <<https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>> [Feb. 2016].
- Start Bootstrap (2016). *Bare*. Verfügbar von: <<https://startbootstrap.com/template-overviews/bare/>> [Aug. 2016].
- Steve Holden (2016). *Python Overview*. Verfügbar von: <<https://wiki.python.org/moin/BeginnersGuide/Overview>> [Feb. 2016].