Technische
Hochschule
Rosenheim

Faculty of computer science

Course of study Master of computer science

Wi-Fi for time critical applications: Practical measurements and evaluation of influencing variables on real time capability

Master Thesis

by

Jakob Hasche

Date of submission: tt.mm.jjjj
First examiner: Prof. Dr. Wolfgang Mühlbauer
Second examiner: Prof. Dr. Florian Künzner

Eigenständigkeitserklärung / Declaration of Originality

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Rosenheim, the July 2, 2025

Jakob Hasche

Kurzfassung

text

Schlagworte:

Contents

List of Figures

List of Tables

Listings

1 Introduction

1.1 Problem description

1.2 Objective

1.3 Motivation

1.4 Layout of the work

2 Basics

2.1 Used Hardware

2.2 Yocto Linux

2.3 Wi-Fi standards

2.3.1 Quality of Service

2.3.2 Scheduling

2.3.3 IEEE 802.11n

2.3.4 IEEE 802.11ac

2.3.5 IEEE 802.11ax

2.3.6 IEEE 802.11be

2.4 Medium Access Mechanisms

2.5 Disruptive factors in Wi-Fis

2.6 Real Time Capability of Wi-Fi

3 Related Work

3.1 RT-Wi-Fi

3.2 Quality of Service in Wi-Fi

3.3 Disruptive factors in Wi-Fi's

3.4 Categorization of this work

4 Test Environment

4.1 Test concept

4.1.1 Requirements

4.1.2 Used Technologies

4.1.3 Server and Client Implementation

4.1.4 Monitoring

4.1.5 Test Automatization

4.1.6 Access point configuration

4.2 Operating System

4.2.1 Requirements

- As less background noise on the system as possible
- Capable for hard real time
- For my test environment the programming languages Python, Rust and the bpftrace library
- Capability to use the Wi-Fi Card
- Network tools, to manage my Wi-Fi interface and generate the workload

4.2.2 Implementation

- local.conf
  - set machine to raspberrypi 5
  - set variable *RUST_LIBC*
  - set preferred linux version to 6.12
  - activated license synaptics-killswitch
- distro (version 5.2.1 on the walnascar branch)
  - for mt7925e is a kernelversion of at least 5.7 necessary
  - walnascar was the newest one to the time of june 2025
  - CONFIG_MT7925E=m
- image

  - hostapd
  - dnsmasq
  - python3, python3-pip
  - net-tools, iproute2, wpa-supplicant, iw und networkmanager (Als Netzwerktools um interfaces, etc steuern zu können)
  - werden mittels *IMAGE_INSTALL* hinzugefügt

- Other Layers
  - Poky
  - meta-openembedded (oe, python, networking)
  - meta-raspberrypi
  - meta-rust in layer.conf the branch walnascar had to be added and the *RUST_LIBC* has to replaced with ${@d.getVar('RUST_LIBC').lower()}, and the variable had to be set in the local.conf

- Own layer
  - bb and bbappend files like normal
  - higher priority
  - set layer compatability to walnascar

- RT-Patch
  - from meta-raspberrypi layer only Linux 6.12.1 is given
  - the normal patch is 6.12.28, this is not working
  - from older patches 6.12.8 is working, but QA had to be added
  - also the kernel config needed to be updated, especially the preemt rt and rcu boost, to read copy update and synchronization in the kernel for rt behaviour

- device-tree overlay
  - download pcie-32bit-dma-pi5.dtbo from https://github.com/raspberrypi/firmware/tree/master/bo
  - write own recipe which installs the file at /boot/overlays/
  - License file has to be created, because it's an own recipe, so the md5 checksum has to be added
  - since Distro 5.1 for S *WORKDIR/sources* has to be set, in this path the files are saved
  - with do install first the folder is created and then the file is saved in the temporary root file system D with the rights 644
  - With Files:$PN the file is set from the temporary file system to the final one
  - in the local.conf with *RPI_USE_OVERLAYS* and *RPI_EXTRA_CONFIG* the overlay is added to the config.txt
  - in meta-raspberrypi/conf/machine/include/rpi-base.inc it has to be added in the *RPI_KERNEL_DEVICETREE_OVERLAYS*

- Custom Files
  - hostapd
    - bbappend in recipes-connectivity/hostapd
    - all configs are located in /etc/hostapd
    - same procedure like for the device tree overlay
    - files are not located in workdir, but in workdir/sources-unpack
  - Rust Server + BPF
    - Create bitbake blueprint with cargo bitbake git clone https://github.com/meta-rust/cargo-bitbake.git
    - Add License
    - set always to newest commit SRCREV = "$AUTOREV"
    - Adjust workdir with S = "$WORKDIR/git/code/server"
    - It depends on DEPENDS (Build-time dependency) += "clang-native kernel-devsrc pkgconfig-native zlib elfutils bpftool-native"
    - (Run Time Dependency) RDEPENDS_$PN += "libbpf"
    - in a do_compile prepand() $exportKERNEL\_HEADERS = "\$STAGING\_KERNEL\_DIR", VML$ "$TOPDIR/tmp/work/raspberrypi5-poky-linux/linux-raspberrypi/6.12.1+git/linux-raspberrypi5-standard-build/", buildvmlinuxwith\$BPFTbtfdumpfile"$VMLINU$ $S/src/bpf/vmlinux.h, preopareClanandBindgen, sothatclangistthecompilerandbindgenhasttheo$ $clangexportBINDGEN_EXTRA_CLANG_ARGS = " --target = bpf -I$S/src/bpf", rusttargetba$ $--release --target = $TARGET_SYS --target-dir = B
    - add configs (add pahole to recipe)
    - config can be seen in /yocto-rpi5/build/tmp/work/raspberrypi5-poky-linux/linux-raspberrypi/6.12.1+git/linux-raspberrypi5-standard-build/.conf
    - with bitbake -c menuconfig virtual/kernel the kernel config can be seen and also the depedencies of the single configs and which are forefilled and which not
    - in source/.kernel-meta/cfg/merge_config_build.log can be seen if the configuration was noticed ant if it was really taken
    - libbpf, clang, cargo, rust, libbpf-dev and elfutils had to be added to the image
    - CONFIG_COMPILE_TEST=y CONFIG_DEBUG_INFO=y, CONFIG_DEBUG_INFO_BTF=y, CONFIG_FRAME_POINTER=y, CONFIG_DEBUG_INFO_DWARF4=y

```
1 def hallo():
2     print("Hallo Welt")
```

5 Wi-Fi Modifications

5.1 Quality of Service

5.2 Change of Wi-Fi Standards

5.3 Frequency

5.4 Bandwidth

6 Evaluation

6.1 Test procedure

6.2 Results of low disturbance test

6.3 Results of normal use test

6.4 Results of high disturbance test

6.5 Discussion

7 Conclusion

A Erstes Kapitel des Anhangs

Wenn Sie keinen Anhang benötigen, dann bitte einfach rausnehmen.

Bibliography