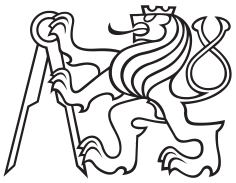
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Decentralizované řízení zahlcení komunikace v inteligentních dopravních systémech pro OS Linux

Bc. Vít Vančura

Vedoucí práce: Ing. Michal Sojka, Ph.D.
Květen 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Vít Vančura**

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: **Decentralizované řízení zahlcení komunikace v inteligentních dopravních systémech pro OS Linux**

Pokyny pro vypracování:

1. Seznamte se se standardy pro bezdrátovou komunikaci v inteligentních dopravních systémech a zejména mechanismem decentralizovaného řízení zahlcení (DCC).
2. Implementujte algoritmus "Decentralized congestion control", zejména vrstvu "access layer" pro OS Linux a jeho "Wi-Fi stack". Předpokládá se, že vrstva "access layer" bude implementována jako modul jádra, ostatní pomocné části jako uživatelské aplikace.
3. Otestujte implementovaný algoritmus jak pomocí simulace, tak se skutečným hardwarem řady Atheros 9000.
4. Výsledky pečlivě zdokumentujte.

Seznam odborné literatury:

- IEEE 802.11-2012, zejména část 802.11p
- ETSI EN 302 663 Access layer specification for 5 GHz band
- ETSI EN 302 637-2 Specification of Cooperative Awareness Basic Service
- ETSI EN 302 637-3 Specification of Decentralized Notification Basic Service
- ETSI TS 102 687 DCC Access layer
- ETSI TS 103 175 DCC Management Entity
- Linux Wireless Wiki: https://wireless.wiki.kernel.org/

Vedoucí: Ing. Michal Sojka, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

prof. Dr. Michal Pěchouček, MSC.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická
L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 2. 2016

Poděkování

Chtěl bych poděkovat vedoucímu práce, Ing. Michalu Sojkovi, Ph.D. za cenné rady a připomínky a pánům Ing. Janu Kaisrlíkovi a Ing. Přemyslu Houdkovi za ochotu a odborné rady při psaní praktické části práce. Dále bych rád poděkoval rodině a přátelům, kteří mě po celou dobu studia podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 27. 5. 2016

_____

Abstrakt

Tato práce se zabývá mechanismy decentralizovaného řízení zahlcení (DCC) při komunikaci inteligentních dopravních systémů (ITS) a jejich návrhem a implementací pro OS Linux s použitím WiFi síťové karty řady Atheros 9000. V práci je popsána funkce DCC systému, jednotlivých DCC mechanismů a aplikování DCC systému na OS Linux.

Klíčová slova: IEEE 802.11p, DCC, ITS-G5, WiFi, Linux OS

Vedoucí práce: Ing. Michal Sojka, Ph.D.
Katedra řídicí techniky - K13135
Fakulta elektrotechnická
České vysoké učení technické v Praze

Abstract

This thesis describes decentralized congestion control mechanisms for intelligent transport systems and its design and implementation for OS Linux using WiFi card Atheros 9000. The thesis also includes scheme of design of the DCC system implemented in OS Linux.

Keywords: IEEE 802.11p, DCC, ITS-G5, WiFi, Linux OS

Title translation: Decentralized congestion control in intelligent transportation systems for Linux

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Moderní technologie zasahují do našeho každodenního života stále více a více. V době malých výkonných počítačů, vysokorychlostního mobilního internetu a rychlé bezdrátové komunikace, bylo jen otázkou času, kdy tyto technologie budou využity ke zlepšení plynulosti a bezpečnosti provozu v osobní automobilové přepravě.

Způsob zlepšení plynulosti a bezpečnosti přepravy je založena na pravidelné výměně informačních zpráv mezi jednotlivými automobily. Pro tyto účely komunikace se nabízí využít technologii ITS-DSRC (Intelligent Transport System - Dedicated Short Range Communication), pracující v pásmu 5 GHz, která se používá při výběru mýta.

Pro zajištění bezpečnosti je důležitá spolehlivost a rychlost komunikace. Za předpokladu, že každý automobil bude vybaven touto technologií a bude schopen komunikovat s ostatními, je zde velmi pravděpodobné vzájemné rušení. V dopravní zácpě při ranní či odpolední špičce, je na úseku silnice dlouhém jeden kilometr, což odpovídá dosahu radiového signálu, vedle sebe nahuštěno několik desítek automobilů. V úvahu také musíme brát ony zmíněné mýtné brány. Navíc frekvenční pásmo 5 GHz je využíváno i v jiných oblastech, nepřímo spojených s přepravou. Při takovém počtu vzájemně se rušících signálů nebude ve vysílacím pásmu dostatek kanálů a nastanou výpadky komunikace způsobené zahlcením pásma. Součástí ITS-DSRC technologie je DCC (Dedicated Congestion Control) systém obsahující mechanismy zabraňující zahlcení, jehož úkolem je zabránit právě zmíněným výpadkům v komunikaci.

Cílem této diplomové práce je implementovat DCC mechanismy proti zahlcení při komunikaci ITS stanic v 5 GHz WiFi pásmu. Systém bude implementován pro OS Linux a jeho "WiFi stack". DCC systém je tvořen komponentami, které pracují na různých vrstvách ISO/OSI modelu. Implementována bude zejména komponenta pracující ve vrstvě "access layer". Předpokládá se, že hlavní část DCC systému bude implementována v jádře OS Linux jako zásuvný modul. Podpůrné komponenty DCC systému budou implementovány jako aplikace a poběží v uživatelském prostředí systému. Součástí práce bude také testování implementovaných DCC mechanismů pomocí simulace i na reálném hardware řady Atheros 9000.

Bohužel jsem nemohl pracovat na této diplomové práci déle, jak jsem měl původně v plánu a byl jsem nucen ukončit studium již nyní. Z tohoto důvodu není implementační část práce zcela dokončena. Dokument obsahuje rozbor problematiky v kapitole 3 a popis všech částí DCC systému s teoretickým návrhem na řešení, v kapitole 4, Architektura DCC pro Linux. V kapitole 5 je popsána implementace, která je v

kapitole 6 následně otestována.

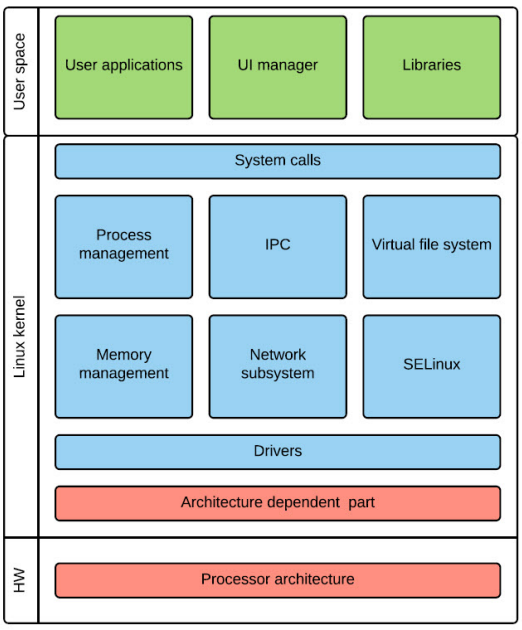
Kapitola 2
Použité technologie

V této kapitole jsou uvedeny a popsány technologie použité při úvaze návrhu řešení a následně využité i v samotné implementaci.

2.1 Operační systém Linux

Operační systém (OS) Linux se dělí na dvě základní části, uživatelské prostředí a jádro OS. V uživatelském prostředí jsou spouštěny uživatelské aplikace a systémové programy uživatelského prostředí. Pro účely programování nabízí uživatelské prostředí OS mnoho standardních knihoven.

Jádro OS obsahuje moduly a rutiny pro obsluhu HW a poskytuje služby aplikacím v uživatelském prostředí (obsluha souborového systému, správa procesů, správa připojeného HW, aj.).



Obrázek 2.1: Struktura OS Linux, dle zdroje [14]

2.1.1 Jádro OS Linux

Jádro OS Linux lze rozdělit na tři podskupiny. První podskupinou jsou rozhraní pro komunikaci s uživatelským prostředím (systémová volání). Druhá podskupina obsahuje podpůrné moduly nezávislé na architektuře systému, tj. moduly pro správu paměti, správu souborového systému, správu procesů a síťový podsystém. Poslední podskupinu v jádře OS tvoří podpůrné moduly a ovladače závislé na architektuře systému, tj. mimo jiné ovladače k hardwaru. Z hlediska programátora má jádro OS Linux jistá omezení. Jedním z těchto omezení je vypnutá jednotka pro počítání v plovoucí řádové čárce (FPU). V jádře OS tedy nelze používat datové proměnné s plovoucí řádovou čárkou. Dále nejsou k dispozici standardní knihovny, na které jsme zvyklí z uživatelského prostředí. Některé standardní knihovny mají v operačním jádře ekvivalentní náhradu.

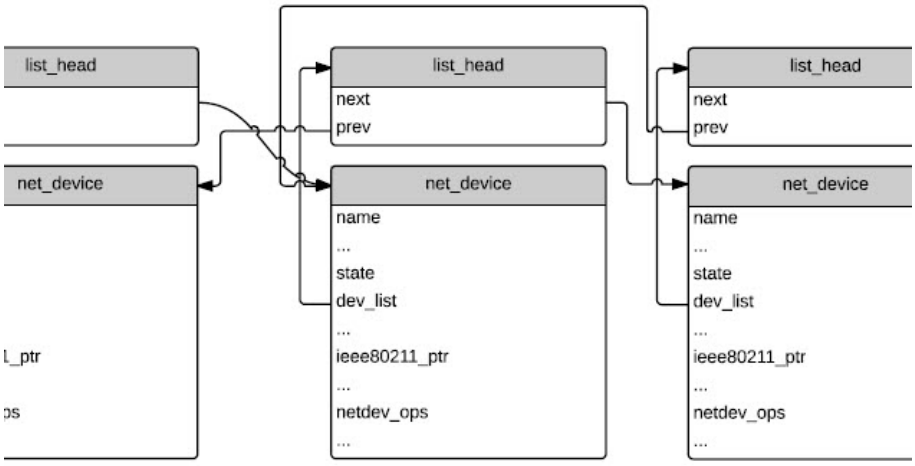
2.2 Síťová karta

Síťová karta je komponenta v počítači, která odesílá a přijímá pakety ze sítě. Obvykle je tvořena softwarovou a hardwarovou částí, nicméně některé síťové karty jsou jen softwarové. Například softwarová síťová karta loopback odesílá pakety sama sobě.

2.2.1 Reprezentace v OS Linux

Každá síťová karta je v jádře OS Linux reprezentována datovou strukturou **net_device** a registrována funkcí *register_netdev* do systému při zavedení jádra.

Síťové karty jsou v jádře uspořádány ve spojovém seznamu *dev_list*.



Obrázek 2.2: Datová struktura **net_device**, dle zdrojových kódů OS Linux

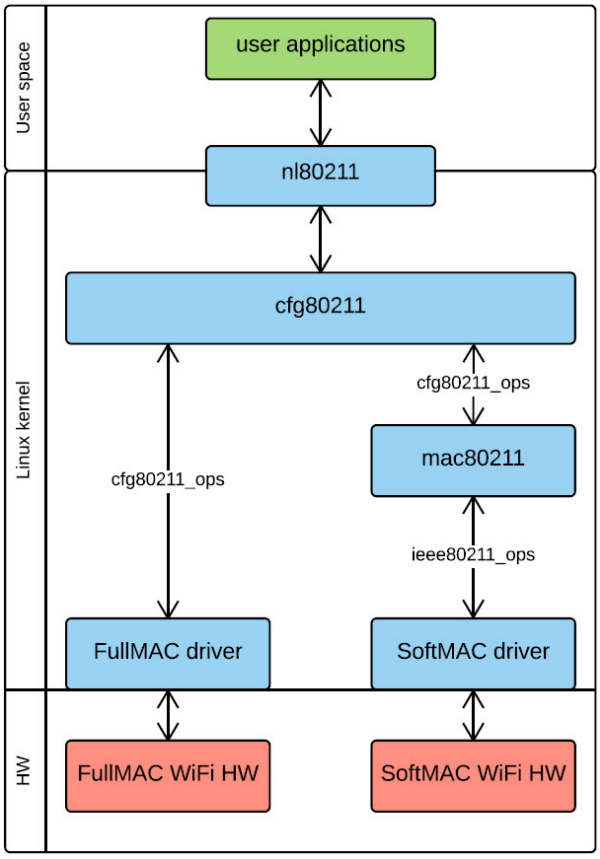
Struktura **net_device** obsahuje informace vztažené k hardwaru (ukazatele na funkce ovladače, obsluhy přerušení, aj.), dále prostředky použité ovladačem (paměť, statistiky přenosů) a také informace použité různými síťovými protokoly (specifická data protokolů). V případě, že se jedná o bezdrátovou síťovou kartu, obsahuje ukazatel *ieee80211_ptr*, který ukazuje na strukturu **wireless_dev**.

2.3 Bezdrátová síťová karta

Bezdrátové síťové karty se dle implementace správy MAC vrstvy dělí na FullMAC a SoftMAC. U FullMAC síťové karty je správa MAC vrstvy implementována přímo v hardware či firmware síťové karty, zatímco v případě SoftMAC síťové karty správu MAC vrstvy obstarává software, v OS Linux je to komponenta mac80211 podsystému IEEE-80211.

2.3.1 Reprezentace v OS Linux

Bezdrátová síťová karta je v jádře OS Linux reprezentována strukturou **wireless_dev**. Tato struktura obsahuje informace specifické pro bezdrátové síťové karty (podporované operační režimy, frekvenční pásma, rychlosti přenosu, aj.). Bezdrátové síťové karty jsou řízeny podsystémem IEEE-80211. Ten se skládá z komponent mac80211, cfg80211 a nl80211. Spolupráce těchto komponent a struktura IEEE-80211 podsystému je znázorněna na obrázku 2.3.



Obrázek 2.3: Struktura IEEE-802.11 podsystému, dle zdrojových kódů OS Linux

Komponenta mac80211 je framework, který pomáhá SoftMAC bezdrátovým síťovým kartám se správou MAC vrstvy. Výrobci SoftMAC bezdrátových síťových karet mohou použít mac80211 při vývoji ovladačů pro jejich karty.

Konfigurace FullMAC i SoftMAC bezdrátových síťových karet se provádí komponentou cfg80211. Tato komponenta poskytuje jednotné API pro komunikaci s HW (řízení přenosu paketů, nastavení parametrů karty, získání statistiky ze zařízení, aj.). Ovladač karty musí všechny požadované funkce implementovat a poskytnout je komponentě cfg80211 při registraci do systému. V případě SoftMAC bezdrátové síťové karty jsou v cfg80211 komponentě zaregistrovány funkce z mac80211 frameworku. Tímto je zaručen jednotný přístup k ovládání karet bez ohledu na implementaci HW funkcí.

Protože ovladače síťových karet jsou umístěny v jádře OS, uživatelské programy k nim nemohou přímo přistupovat. Proto je součástí IEEE-80211 podsystému komponenta nl80211. Poskytuje komunikační API pro aplikace v uživatelském prostředí a převádí příkazy do komponenty cfg80211. Ke komunikaci mezi jádrem OS a uživatelským prostředím používá Netlink protokol.

2.3.2 Operační režimy

Bezdrátové síťové karty mohou pracovat v různých operačních režimech (MASTER, MANAGED, AD-HOC a MONITOR).

V *MASTER* režimu je bezdrátová síťová karta použita pro vytvoření WiFi sítě se specifikovaným názvem a chová se jako klasický přístupový bod. Ostatní bezdrátové síťové karty se k této síti mohou připojit.

Aby byla bezdrátová síťová karta schopna se připojit k WiFi síti, musí být nastavena do režimu *MANAGED*. Po připojení k síti je bezdrátová síťová karta autorizována dle požadavků sítě a v případě úspěšné autorizace je jí dovolena komunikace. Komunikace v síti probíhá přes hlavní přístupový bod.

Režim chodu *AD-HOC* vytváří WiFi síť, ve které každá bezdrátová síťová karta může komunikovat s ostatními kartami přímo, bez hlavního přístupového bodu. Bezdrátové síťové karty musí být ve vzájemném dosahu a musí mít nastavený stejný název a kanál WiFi sítě.

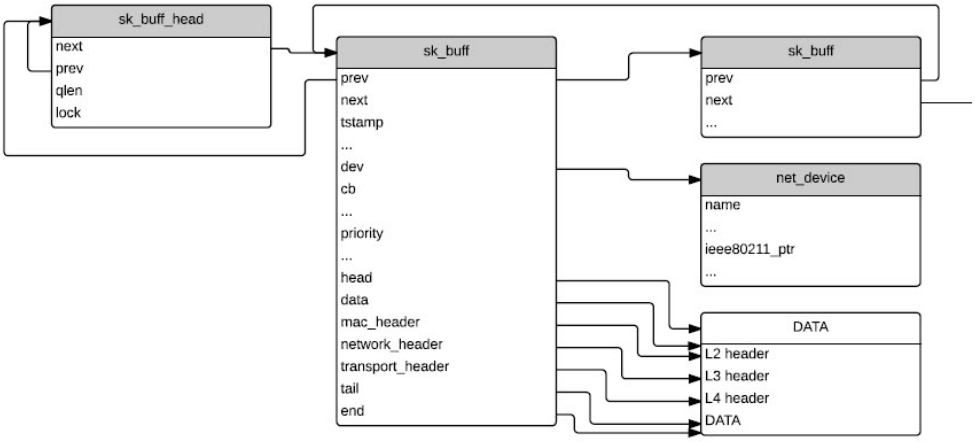
Posledním z operačních režimů je režim *MONITOR*. Bezdrátová síťová karta se přepne do pasivního režimu a poslouchává komunikaci na WiFi síti. V tomto režimu dokáže zachytit pakety vysílané ze všech ostatních bezdrátových síťových karet a také zaznamenat metadata s informacemi o stavu HW při příjmu paketu (metadata jako jsou síla signálu, rychlost přijímání, aj.). Pro uložení metadat je použita speciální struktura přijatá spolu s paketem.

Některé bezdrátové síťové karty mohou pracovat v *MANAGED* i *MONITOR* režimu zároveň. Bezdrátová síťová karta je tedy schopna odposlouchávat síťovou komunikaci a zároveň se jí i účastnit a odesílat pakety.

2.4 Síťový podsystém OS Linux

Linuxový síťový podsystém pracuje nezávisle na použitém protokolu. Data jsou posílaná síťovým podsystémem jádra OS včetně hlaviček jednotlivých protokolů, a jsou mezi vrstvami protokolového zásobníku předávána pomocí datové struktury

sk_buff. Tím je minimalizována potřeba kopírování dat a operací s tím spojených. Datová struktura **sk_buff** je vytvořena vždy, když chce aplikace odeslat data, nebo když síťová karta přijme paket ze sítě.



Obrázek 2.4: Datová struktura **sk_buff**, dle zdrojových kódů OS Linux

Struktury **sk_buff** jsou uspořádány ve spojovém seznamu s počátkem ve struktuře **sk_buff_head**. Struktura **sk_buff** obsahuje ukazatele na předchozí a následující struktury *prev* a *next*, časové razítko přijetí paketu *tstamp*, *dev* ukazatel na síťovou kartu pracující s paketem, proměnné pro vnitřní použití, dále ukazatele na hlavičky všech použitých protokolů a mimo jiné samozřejmě i část s přenášenými daty.

Při odesílání paketu je pro každý paket alokována paměť a naplněna daty. Paket je postupně předáván dolů vrstvami protokolového zásobníku a doplňován hlavičkou z každé vrstvy. Poté je odeslán pomocí síťového karty do sítě.

Přijatá data ze sítě jsou umístěna ovladačem síťové karty do datové struktury **sk_buff** a odeslána do protokolového zásobníku. Každá vrstva zásobníku přečte relevantní data a předá strukturu výše. Tímto způsobem se data dopraví až do koncové aplikace.

Paket je cestou protokolovým zásobníkem několikrát kontrolován. V síťové vrstvě je předán do Netfilter frameworku OS, kde může být zpracován a případně zahozen. Dále, před odesláním vložením do odesílací fronty síťové karty, je paket předán příslušné Queueing disciplíně ke zpracování. Paket je zařazen dle určitých parametrů a podmínek do odesílací fronty k odeslání.

2.5 Intelligent Transport Systems

Intelligent Transport Systems, neboli ITS, jsou systémy podporující přepravu zboží a lidí pomocí komunikačních technologií k zajištění bezpečnosti a efektivity. ITS se vztahuje ke všem druhům přepravy, letecké, námořní, vlakové a také osobní. Zajištěním bezpečnosti a efektivity v osobní přepravě se zabývá projekt Dedicated Short-Range Communications (DSRC) a jeho část Wireless Access in Vehicular

Environment (WAVE). DSRC definuje komunikaci mezi ITS stanicemi ve vozidlech a také komunikaci mezi ITS stanicí ve vozidle a pevným bodem jako je mýtná brána, či informační bod u silnice. Rychlá výměna informačních zpráv a znalost aktuálního pohybu okolních vozidel přispívá ke zvýšení bezpečnosti a zlepšení efektivity dopravy. WAVE je skupina standardů popisující systém pro podporu výměny zpráv mezi ITS stanicemi a tvoří jádro DSRC. Jedním ze skupiny standardů je IEEE 802.11p (kapitola 2.6).

2.5.1 Komunikace v DSRC systému

Komunikace v DSRC systému probíhá mezi stanicemi dvou základních typů, Road-side Unit (RSU) a On-Board Unit (OBU). RSU je pevně umístěná stanice, mýtná brána či informační bod, ke které se připojují projíždějící vozidla. OBU se nachází ve vozidle a je napojena k interní komunikační síti vozidla. Jak se vozidlo pohybuje, OBU se připojuje k jednotlivým RSU stanicím. Maximální dosah RSU je dle okolního prostředí omezen na 1km a OBU stanice by se měla připojit vždy k té nejbližší RSU. Stanice mohou mezi sebou komunikovat rychlostmi dle standardu IEEE 802.11a. Rychlost komunikace je volena dle aktuální rychlosti vozidla. Komunikační pásmo 5 GHz je rozděleno do 10MHz širokých pásem pro řídící kanál a kanál pro služby (Control Channel CCH a Service Channel SCH). Zprávy obsahující informace o situaci kolem vozidla a jiné důležité zprávy využívají řídící kanál. Kanál pro služby je využívaný například při placení mýta, benzínu a v kritických situacích i jako kopie řídícího kanálu.

2.5.2 WAVE Short Messages Protocol

WAVE Short Messages Protocol, neboli WSMP, popisuje formát komunikace mezi ITS stanicemi. Při komunikaci je použit systém priorit Quality of Service (QoS). Priority jsou použity pro odlišení důležitých zpráv a jejich rychlé doručení. Každá WSMP zpráva dále obsahuje z aplikace nastavenou přenosovou rychlost, číslo kanálu a sílu vysílacího signálu pro odeslání paketu.

2.6 IEEE 802.11p

IEEE 802.11p je dodatek standardu IEEE 802.11, rozšiřující Media Access Control (MAC) a specifikaci fyzické vrstvy. Dodatek definuje mechanismus umožňující použít IEEE 802.11 v automobilovém prostředí. V prostředí se rychle mění přenosové podmínky a také je potřeba rychlého spojení a výměny dat. Součástí dodatku je nový přenosový mód Outside the Context of a BSS (OCB). Tento (OCB) mód umožňuje všem stanicím v síti vzájemně komunikovat bez použití autorizace, a bezpečnostních procedur. Díky tomu je dosažena vysoká rychlost komunikace. Před samotnou komunikací je potřeba nastavit kanál, tj. střední frekvence a šířka kanálu. Toto nastavení je provedeno při inicializaci.

2.7 PCAP API

PCAP je API umožňující odchytávání síťové komunikace. Zachyceny jsou všechny pakety procházející zařízením. Při použití bezdrátové síťové karty v operačním režimu *MONITOR* je takto možné odchytit i pakety určené jiným zařízením. Zachycené pakety nejsou nějak zpracované a pokud nebyly šifrované, je možné přečíst jejich obsah. Ke každému paketu může být připojena struktura *RADIOTAP*, obsahující informace o podmínkách přijetí paketu (rychlost přenosu, RX síla signálu a jiné informace).

2.8 Netfilter framework

Netfilter framework je nástroj pro filtrování IPv4 a IPv6 paketů v síťové komunikaci. Definuje několik záchytných bodů v síťovém podsystému jádra OS. K těmto bodům se registrují obslužné funkce, které jsou volány při průchodu paketu. Díky tomu je možné manipulovat s příchozími i odchozími pakety, kontrolovat jejich obsah a případně zamezit jejich průchodu. Na Netfilter frameworku je postaveno mnoho podpůrných modulů jádra, například Network Address and Port Translation (NAPT), firewally a také moduly QoS.

2.9 Queueing disciplína

Po přenesení paketu do jádra OS, je paket zkontrolován vstupním firewallem a je postupně zpracován síťovým podsystémem. Pokud je paket určen pro aktuální stanici, je přenesen zpět do vyšších vrstev protokolového zásobníku. Je-li určen pro jinou stanici, vybere se výstupní zařízení a paket je uložen do jeho fronty ke zpracování pomocí Queueing disciplíny (Qdisc).

Qdisc je součástí síťového podsystému a slouží pro řízení síťového provozu. Tvoří ho algoritmus, který řídí zařazení paketu do příslušné odesílací fronty. Qdisc se dělí na *CLASSLESS* a *CLASSFUL*. Hlavním rozdílem mezi nimi je, že *CLASSFUL* Qdisc obsahují konfigurovatelné části a mohou mít na sebe připojené další Qdisc. Každá Qdisc v systému je určena dvojicí čísel *MAJOR:MINOR*. Všechny registrované Qdisc jsou uspořádané v stromové struktuře a dělí se na fronty, třídy a filtry. Pro fronty platí, že *MINOR* číslo je rovno 0. Třídy patřící k jedné frontě mají stejné *MAJOR* číslo. Hlavní Qdisc (*root*) má dvojici čísel *MAJOR:MINOR* rovno 1:0. K *root* mohou být připojeny třídy a filtry, nebo další fronty.

Kromě směrování paketů do jednotlivých odesílacích front, je možné pomocí Qdisc kontrolovat četnost odesílání paketů a tak předcházet zahlcení vysílacího pásma.

2.9.1 Struktura qdisc

Každá qdisc má definované funkce určené k manipulaci s pakety či s Qdisc parametry. Všechny tyto funkce ve struktuře *Qdisc_ops* jsou registrované v Qdisc API. Po vytvoření Qdisc je potřeba inicializovat její parametry a nastavit je do výchozích hodnot. K tomu slouží funkce *init* a *reset*. K manipulaci s příchozím paketem slouží

funkce *enqueu*. Přijme paket a zařadí ho do příslušné odesílací fronty síťové karty. Pakety čekají v odesílací frontě síťové karty dokud nejsou z fronty odebrány funkcí *dequeue* a přesunuty k samotnému odeslání. Při pokusu odeslat paket rozhraním síťové karty je možné, že síťová karta je zaneprázdněná a nemůže paket odeslat. V takovém případě je paket vrácen k opětovnému zařazení do odesílací fronty, nebo je zahozen. V tabulce 2.1 je seznam funkcí z *Qdisc_ops* struktury s jejich popisem.

Funkce	Popis
enqueue	zařazení paketu do fronty
dequeue	odebrání paketu z fronty
peek	načtení paketu z fronty, bez jeho odebrání
drop	zahození paketu z fronty
init	inicializace Qdisc
reset	nastavení výchozích hodnot parametrů Qdisc
destroy	deinicializace Qdisc
change	nastavení hodnot parametrů Qdisc
attach	připojení k nadřazené Qdisc

Tabulka 2.1: Struktura **Qdisc_ops** - seznam funkcí, dle zdrojových kódů OS Linux

Pro manipulaci s třídami a filtry musí mít Qdisc API k dispozici následující funkce ve struktuře *Qdisc_class_ops*. Funkce *graft* připojí novou Qdisc k nadřazené (vymění novou za starou). Při odebírání Qdisc je použita funkce *delete*. Pro vnitřní použití se udržuje čítač počtu použití Qdisc, aby bylo jasné, kolik jiných Qdisc ji využívá. Qdisc není možné smazat, dokud tento čítač není nulový. K manipulaci s čítačem jsou určeny funkce *get* a *put*. Další funkce struktury *Qdisc_class_ops* jsou zmíněny v tabulce 2.2.

Funkce	Popis
select_queue	výběr cílové fronty na základě tříd
graft	připojení či nahrazení staré Qdisc novou
leaf	manipulace s třídami, vrací okrajový filtr
glen_notify	indikace délky fronty
get	manipulace s třídami
put	manipulace s třídami
change	úprava parametrů třídy
delete	smazání třídy či filtru
walk	iterace přes všechny třídy
tcf_chain	manipulace s filtry
bind_tcf	manipulace s filtry
unbind_tcf	manipulace s filtry

Tabulka 2.2: Struktura **Qdisc_class_ops** - seznam funkcí, dle zdrojových kódů OS Linux

2.10 Rate control algoritmus

Každá bezdrátová síťová karta používá při odesílání paketů Rate Control Algoritmus (RCA). Ten je součástí MAC vrstvy a může být implementován v hardwaru či firmwaru bezdrátové síťové karty (FullMAC), nebo je použit RCA implementovaný v komponentě mac80211 (SoftMAC). Algoritmus nastavuje pro každý paket, který má být odeslán do sítě, přenosovou rychlost. Bezdrátová síťová karta poskytuje seznam podporovaných přenosových rychlostí, ze kterých algoritmus vybírá. Účel algoritmu je přizpůsobovat podmínky vysílání aktuálnímu stavu prostředí a tak zajistit nejlepší možnou kvalitu komunikace. V komponentě mac80211 je výchozí RCA Minstrel algoritmus.

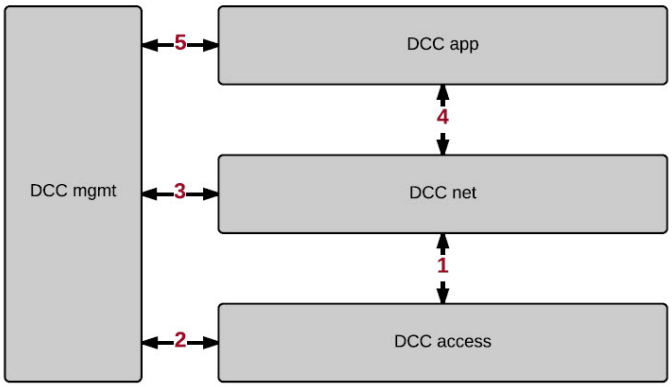
Kapitola 3

Decentralized Congestion Control

Tato kapitola obsahuje popis DCC systému a jeho mechanismů dle zdroje [5]. DCC (Decentralized Congestion Control) je systém zabraňující zahlcení při komunikaci ITS stanic. DCC systém zaručuje stabilitu sítě, spravedlivé dělení vysílacího pásma mezi všemi ITS stanicemi v síti a udržuje zatížení komunikačního kanálu periodickými zprávami pod stanoveným limitem.

3.1 Struktura DCC

Správná funkce DCC systému závisí na spolupráci několika komponent, které pracují v různých vrstvách protokolového zásobníku.

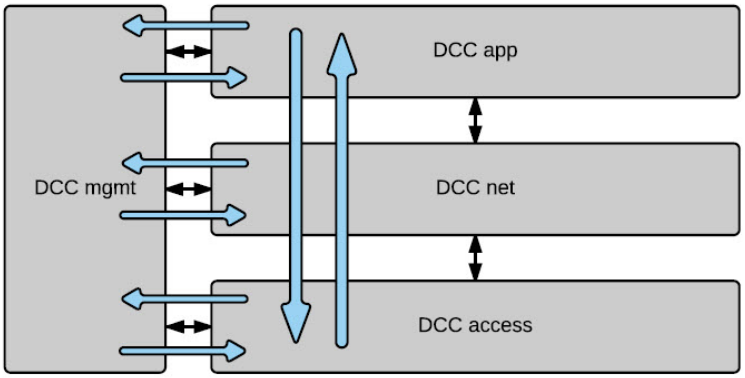


Obrázek 3.1: Struktura DCC systému včetně komunikačních rozhraní, dle zdroje [5], str. 9

Podobně jako je model ISO/OSI tvořen vrstvami, je i DCC systém tvořen komponentami. Systém konkrétně obsahuje aplikační komponentu (DCC app), síťovou a transportní komponentu (DCC net) a přístupovou komponentu (DCC access).

Všechny tyto komponenty DCC systému jsou propojeny s management komponentou (DCC mgmt). Celý DCC systém je nastaven a řízen dle parametrů uložených v Network Design Limits (NDL) tabulce a v tabulce DCC profilů. Obě tyto tabulky jsou součástí komponenty DCC mgmt.

3.2 Komunikace DCC komponent



Obrázek 3.2: Tok komunikace mezi DCC komponentami, dle zdroje [12], strana 19

Komunikace mezi všemi komponentami DCC systému je znázorněna na obrázku 3.2. V systému je celkem 5 komunikačních rozhraní (*DCC app <-> DCC net, DCC net <-> DCC access, DCC mgmt <-> DCC app, DCC mgmt <-> DCC net, DCC mgmt <-> DCC access*).

Komunikační cestou **DCC app <-> DCC net <-> DCC access** jsou odesílány pakety do sítě. Každý paket odeslán z ITS aplikace je ve formátu **IN-UNITDATA.request**, (viz tabulka 3.1) a obsahuje jedinečné referenční číslo zprávy *CommandRef* (v rozsahu od 0 do 255), které je pro každý nový paket inkremenováno. Dále obsahuje zdrojovou a cílovou adresu *SourceAddress, DestinationAddress* a volitelnou informaci o protokolu třetí vrstvy protokolového zásobníku. Následují data, identifikační číslo DCC profilu (*DP-ID* dle kapitoly 3.3) a parametry odesílání doporučené z aplikace.

Parametr	Definice
CommandRef	referenční číslo zprávy
Protocol	info o protokolu L3
SourceAddress	zdrojová adresa
DestinationAddress	cílová adresa
Data	data z aplikace
DP-IP	ID DCC profilu
TxParameters -> ServiceClass	potvrzování na QoS řídící zprávu
TxParameters -> UseRTS	použití RTS/CTS
TxParameters -> TxPower	vysílací síla, v jednotkách 0.5 dBm
TxParameters -> MCS	modulace a kódování

Tabulka 3.1: Struktura zprávy **IN-UNITDATA.request**, dle zdroje [10], str. 9, tabulka 1

Po úspěšném odeslání dat je aplikace informována od DCC access komponenty zprávou **IN-UNITDATA.status**, (viz tabulka 3.2). Zpráva obsahuje potvrzované

referenční číslo *CommandRef* a informace o stavu odeslání (zdrojovou a cílovou adresu, kanál a TX parametry použité při odeslání).

Parametr	**Definice**
CommandRef	referenční číslo zprávy
SourceAddress	zdrojová adresa
DestinationAddress	cílová adresa
Channel	kanál použitý pro odeslání dat
TxStatus	stav odeslání zprávy
TxParameters -> ServiceClass	potvrzování na QoS řídící zprávu
TxParameters -> UseRTS	použití RTS/CTS
TxParameters -> TxPower	vysílací síla
TxParameters -> MCS	modulace a kódování

Tabulka 3.2: Struktura zprávy **IN-UNITDATA.status**, dle zdroje [10], str. 11, tabulka 5

Při příchodu zprávy od jiné ITS stanice, je do aplikace přenesena informační zpráva ve formátu **IN-UNITDATA.indication**, (viz tabulka 3.3). Zpráva obsahuje kromě příchozích dat také zdrojovou a cílovou adresu a RX parametry použité při přijetí.

Parametr	**Definice**
Protocol	info o protokolu L3
SourceAddress	zdrojová adresa
DestinationAddress	cílová adresa
Data	příchozí data
RxParameters -> Channel	kanál použitý pro příjem dat
RxParameters -> RSSI	indikace síly příchozího signálu
RxParameters -> MCS	modulace a kódování

Tabulka 3.3: Struktura zprávy **IN-UNITDATA.indication**, dle zdroje [10], str. 10, tabulka 3

Komunikace mezi komponentami DCC mgmt a DCC access, (rozhraní **DCC mgmt <-> DCC access**) probíhá způsobem request-response. Zprávou **MI-COMMAND.request** (viz tabulka 3.4) může komponenta DCC mgmt požádat o vykonání příkazu v komponentě DCC access. Zpráva obsahuje ID rozhraní *MAC-ID*, které příkaz vykoná, referenční číslo zprávy *CommandRef*, ID příkazu a jeho parametry. Po přijetí žádosti a jejím případném vykonání, komponenta DCC access odpoví zprávou **MI-COMMAND.confirm** (viz tabulka 3.5). Komponenta DCC access může také požádat o vykonání příkazu v komponentě DCC mgmt. Požadavek odešle zprávou **MI-REQUEST.request** (viz tabulka 3.6), na kterou DCC mgmt odpoví zprávou **MI-REQUEST.confirm** (viz tabulka 3.7), obsahující informace o stavu vykonání příkazu.

Parametr	Definice
MAC-ID	ID komunikačního rozhraní
CommandRef	referenční číslo zprávy
MI-Command.No.Value	id vybraného příkazu
MI-Command.Value	data pro vybraný příkaz

Tabulka 3.4: Struktura zprávy **MI-COMMAND.request**, dle zdroje [9], str. 11, tabulka 1

Parametr	Definice
MAC-ID	ID komunikačního rozhraní
CommandRef	referenční číslo zprávy
ErrStatus	kód chyby

Tabulka 3.5: Struktura zprávy **MI-COMMAND.confirm**, dle zdroje [9], str. 11, tabulka 2

Parametr	Definice
MAC-ID	ID komunikačního rozhraní
CommandRef	referenční číslo zprávy
MI-Request.No.Value	id vybraného příkazu
MI-Request.Value	data pro vybraný příkaz

Tabulka 3.6: Struktura zprávy **MI-REQUEST.request**, dle zdroje [9], str. 12, tabulka 3

Parametr	Definice
MAC-ID	ID komunikačního rozhraní
CommandRef	referenční číslo zprávy
ErrStatus	kód chyby

Tabulka 3.7: Struktura zprávy **MI-REQUEST.confirm**, dle zdroje [9], str. 13, tabulka 4

Také je možné nastavit parametry rozhraní v DCC access komponentě, případně získat jejich aktuální hodnoty pomocí zpráv **MI-SET.request** (viz tabulka 3.8) a **MI-GET.request** (viz tabulka 3.10). Po jejich přijetí komponenta DCC access nastaví, nebo přečte požadované parametry a vrátí informaci o stavu zpět do komponenty DCC access zprávami **MI-SET.confirm** (viz tabulka 3.9) a **MI-GET.confirm** (viz tabulka 3.11).

Parametr	Definice
MAC-ID	ID komunikačního rozhraní
CommandRef	referenční číslo zprávy
NumberOfParams	počet následujících parametrů
I-Param.No	ID vybraného parametru
I-Param.Value	data pro vybraný parametr

Tabulka 3.8: Struktura zprávy **MI-SET.request**, dle zdroje [9], str. 14, tabulka 5

Parametr	Definice
MAC-ID	ID komunikačního rozhraní
CommandRef	referenční číslo zprávy
NumberOfParams	počet následujících parametrů
Errors.I-Param.No	ID vybraného parametru
Errors.ErrStatus	kód chyby

Tabulka 3.9: Struktura zprávy **MI-SET.confirm**, dle zdroje [9], str. 15, tabulka 6

Parametr	Definice
MAC-ID	ID komunikačního rozhraní
CommandRef	referenční číslo zprávy
NumberOfParams	počet následujících parametrů
I-Param.No	ID požadovaného parametru

Tabulka 3.10: Struktura zprávy **MI-GET.request**, dle zdroje [9], str. 16, tabulka 7

Parametr	Definice
MAC-ID	ID komunikačního rozhraní
CommandRef	referenční číslo zprávy
NumberOfParams	počet následujících parametrů
I-Param.No	ID požadovaného parametru
I-Param.Value	data pro požadovaný parametr

Tabulka 3.11: Struktura zprávy **MI-GET.confirm**, dle zdroje [9], str. 16, tabulka 8

Kromě výše zmíněných zpráv pro standardní komunikaci, DCC access a DCC mgmt používají zprávy určené ke správě tabulky DCC profilů. Komponenta DCC access odešle zprávu **IM-DP-COMMAND.request** (viz tabulka 3.12) do komponenty DCC mgmt, jako požadavek k vykonání příkazu týkajícího se tabulky DCC profilů. Zpráva obsahuje ID DCC profilu, referenční číslo zprávy *CommandRef* a ID příkazu. Příkazy jsou přesně definované a popsané ve zdroji [9].

Parametr	Definice
DP-ID	ID DCC profilu
CommandRef	referenční číslo zprávy
IM-DP-Command.No.Value	ID vybraného příkazu
IM-DP-Command.Value	data pro vybraný příkaz

Tabulka 3.12: Struktura zprávy **IM-DP-COMMAND.request**, dle zdroje [9], str. 18, tabulka 9

Komponenta DCC mgmt informuje o stavu přijatého požadavku odesláním zprávy **IM-DP-COMMAND.confirm** (viz tabulka 3.13) zpět do DCC access. Zpráva obsahuje ID DCC profilu, potvrzované referenční číslo zprávy *CommandRef* a stav vykonání příkazu.

Parametr	Definice
DP-ID	ID DCC profilu
CommandRef	referenční číslo zprávy
ErrStatus	kód chyby

Tabulka 3.13: Struktura zprávy **IM-DP-COMMAND.confirm**, dle zdroje [9], str. 19, tabulka 10

Komponenta DCC mgmt může také odeslat do komponenty DCC access požadavek na vykonání příkazu týkajícího se tabulky DCC profilů. Požadavek provede odesláním zprávy **IM-DP-REQUEST.request** (viz tabulka 3.14). DCC access odpoví na požadavek zprávou **IM-DP-REQUEST.confirm** (viz tabulka 3.15).

Parametr	Definice
DP-ID	ID DCC profilu
CommandRef	referenční číslo zprávy
IM-DP-Request.No.Value	ID vybraného příkazu
IM-DP-Request.Value	data pro vybraný příkaz

Tabulka 3.14: Struktura zprávy **IM-DP-REQUEST.request**, dle zdroje [9], str. 20, tabulka 11

Parametr	Definice
DP-ID	ID DCC profilu
CommandRef	referenční číslo zprávy
ErrStatus	kód chyby

Tabulka 3.15: Struktura zprávy **IM-DP-REQUEST.confirm**, dle zdroje [9], str. 20, tabulka 12

Zprávy odeslané z komponenty DCC app obsahují v hlavičce ID DCC profilu. DCC profil definuje určité podmínky a hodnoty parametrů odesílání. DCC profily

jsou uložené v tabulce DCC profilů v komponentě DCC mgmt. Aby bylo možné odeslat paket s parametry zvoleného DCC profilu, musí o ně komponenta DCC access požádat. Požadavek je proveden odesláním zprávy **IM-DP-GET.request** (viz tabulka 3.16) do komponenty DCC mgmt. Komponenta DCC mgmt odešle zpět požadované parametry profilu zprávou **IM-DP-GET.confirm** (viz tabulka 3.17).

Parametr	Definice
DP-ID	ID DCC profilu
CommandRef	referenční číslo zprávy
NumberOfParams	počet následujících parametrů
D-Param.No	ID požadovaného parametru

Tabulka 3.16: Struktura zprávy **IM-DP-GET.request**, dle zdroje [9], str. 21, tabulka 13

Parametr	Definice
DP-ID	ID DCC profilu
CommandRef	referenční číslo zprávy
NumberOfParams	počet následujících parametrů
D-Param.No	ID požadovaného parametru
D-Param.Value	hodnota požadovaného parametru

Tabulka 3.17: Struktura zprávy **IM-DP-GET.confirm**, dle zdroje [9], str. 22, tabulka 14

Komunikace mezi ostatními komponentami je detailněji popsána v příslušných ETSI dokumentech k jednotlivým komunikačním rozhraním, viz tabulka 3.18.

číslo standardu	popis
ETSI TS 102 723-4	rozhraní mezi DCC mgmt a DCC net
ETSI TS 102 723-5	rozhraní mezi DCC mgmt a DCC app
ETSI TS 102 723-6	rozhraní mezi DCC mgmt a DCC security
ETSI TS 102 723-7	rozhraní mezi DCC security a DCC access
ETSI TS 102 723-8	rozhraní mezi DCC security a DCC net
ETSI TS 102 723-9	rozhraní mezi DCC security a DCC app
ETSI TS 102 723-11	rozhraní mezi DCC net a DCC app

Tabulka 3.18: Specifikace dalších rozhraní DCC systému

3.3 Tabulka DCC profilů

Komponenta DCC mgmt obsahuje tabulku DCC profilů. Každý DCC profil má definované *ID*, číslo odesílací fronty $Q\#$, časový interval mezi odesláním dvou paketů na stejném kanálu T_{off}, výkon odesílání P_{TX} a také rychlost odesílání D_{TX}. Tabulka

je rozdělena na tři části dle aktuálního stavu DCC systému (*RELAXED, ACTIVE a RESTRICTIVE*). Detailní popis tabulky DCC profilů je ve zdroji [12].

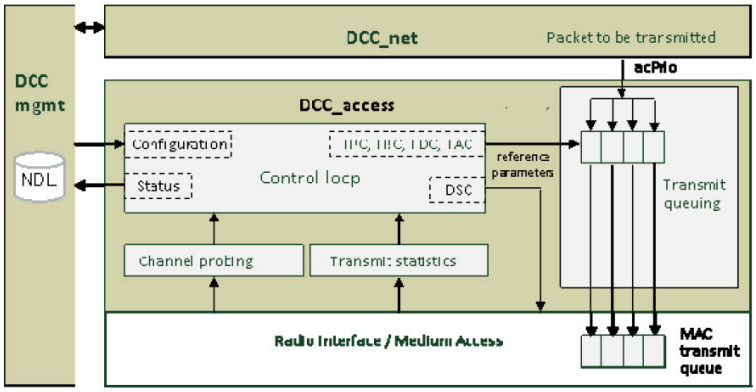
V době psaní této práce je tabulka DCC profilů ve vývoji a bude doplněna o další parametry.

3.4 NDL tabulka

Network Design Limits (NDL) tabulka obsahuje informace o aktuálním stavu DCC systému a také informace potřebné k jeho konfiguraci a řízení. Tabulka je součástí komponenty DCC mgmt a informace z ní i do ní jsou sdíleny v rámci DCC systému. Data v tabulce mají definovaný formát a také limitní hodnoty, kterých mohou nabývat. Všechny výchozí hodnoty, včetně definovaného formátu každého atributu tabulky jsou podrobně popsány ve zdroji[5]. Důležitou částí NDL tabulky je část s referenčními hodnotami. Při změně stavu DCC systému jsou jeho parametry nastaveny dle referenčních hodnot nového stavu. NDL tabulka je využívána převážně komponentou *DCC access*, ale ostatní komponenty k ní mohou také přistupovat.

3.5 DCC access

Komponenta DCC access provádí několik mechanismů řízení a měření. Řídí sílu vysílacího signálu pomocí Transmit Power Control (TPC) mechanismu, četnost vysílání jednotlivých paketů Transmit Rate Control (TRC) mechanismem. Dále mechanismus Transmit Datarate Control (TDC) volí rychlost odesílání jednotlivých paketů. Komponenta DCC access dále shromažďuje statistiky a jiné parametry přenosu v Transmit a Receive modelu. Na obrázku 3.3 je popsána architektura DCC access komponenty. Hlavní částí DCC access komponenty je řídící smyčka.



Obrázek 3.3: Funkční blok DCC access komponetny, dle zdroje [5], str. 10

Každý paket určený k odeslání obsahuje hlavičku nastavenou z DCC app. DCC access hlavičku zkontroluje a vyžádá si parametry odesílání zvoleného DCC profilu od DCC mgmt. Parametry porovná s referenčními hodnotami aktuálního stavu a dle pravidel DCC mechanismů následně upraví parametry odesílání v paketu a zařadí ho do určité odesílací fronty, případně zahodí.

3.5.1 DCC řídící smyčka

Řídící smyčka DCC access komponenty může být popsána jako stavový automat znázorněn na obrázku 3.4. Stavový automat je tvořen několika stavy celkem tří typů, *RELAXED*, *ACTIVE* a *RESTRICTIVE*. Stavů typu *ACTIVE* může být několik. Přechod mezi jednotlivými stavy závisí na vypočítaném zatížení kanálu *channelLoad*, viz tabulka 3.28. V NDL tabulce jsou udržovány parametry řídící smyčky pro všechny typy stavů, popsané v tabulce 3.19.

Parametr	Definice
asStateId	id stavu
asChanLoad	referenční hodnota zatížení kanálu
asDcc	maska popisující povolené DCC mechanismy
asTxPower	referenční síla TX signálu
asPacketInterval	referenční interval mezi přenosem paketů
asDatarate	referenční rychlost odesílání paketu
asCarrierSense	referenční citlivost přijímače

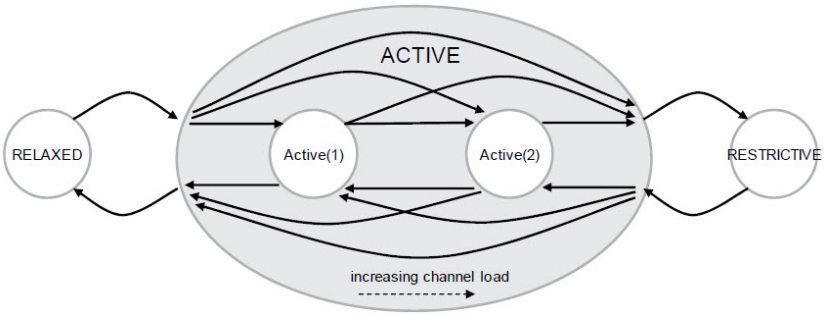
Tabulka 3.19: Parametry řídící smyčky, dle zdroje [5], str. 22, tabulka 15

Přechod do jiného stavu S je podmíněný hodnotou parametru *asChanLoad*. Pokud je hodnota vypočítaného zatížení *channelLoad* větší než hodnota parametru *asChanLoad* definovaná v NDL tabulce pro aktuální stav S_N, automat přejde do následujícího stavu S_{N+1}. V případě, že je zatížení *asChanLoad* menší než hodnota *asChanLoad* definovaná v NDL tabulce pro předchozí stav S_{N-1}, přejde do onoho předchozího stavu S_{N-1}.

Parametr	Definice
NDL_TimeUp	maximální doba reakce na zvýšení aktuálního zatížení
NDL_TimeDown	maximální doba reakce na snížení aktuálního zatížení
NDL_minDccSampling	minimální interval kontroly podmínek pro změnu stavu

Tabulka 3.20: Časové parametry řídící smyčky, dle zdroje [5], str. 21, tabulka 14

Kontrola podmínky pro přechod do jiného stavu stavového automatu je prováděna dle parametru *NDL_minDccSampling*. Při přechodu do nového stavu jsou hodnoty referenčních parametrů nahrazeny hodnotami řídící smyčky z NDL tabulky pro nový stav S_N, např. *NDL_refTxPower = asTxPower(id)*. Změnou referenčních parametrů řídící smyčka nepřímo ovládá níže zmíněné mechanismy.



Obrázek 3.4: Funkční blok DCC access řídící smyčky, dle zdroje [5], str. 23 obrázek 5

Řídící smyčka z NDL tabulky přijímá konfigurační parametry dle kterých upravuje chování DCC access komponenty.

3.5.2 Transmit Power Control

Mechanismus Transmit Power Control (TPC) upravuje *effTxPower* sílu vysílacího signálu pro každý paket. Hodnota *effTxPower* je porovnána s referenční hodnotou *NDL_refTxPower(acPrio)* příslušné odesílací fronty a následně upravena dle rovnice 3.3 níže.

Parametr	Definice
NDL_minTxPower	minimální síla signálu, kterou TPC může zvolit
NDL_maxTxPower	maximální síla signálu, kterou TPC může zvolit
NDL_defTxPower	výchozí síla vysílacího signálu
NDL_refTxPower	síla signálu dle referenční tabulky

Tabulka 3.21: Parametry **TPC**, dle zdroje [5], str. 11, tabulka 1

V tabulce 3.21 jsou vypsány NDL parametry používané TPC mechanismem. Parametry NDL_refTxPower a NDL_defTxPower splňují rovnice 3.1 a 3.2.

$$NDL\_minTxPower \leq NDL\_refTxPower \leq NDL\_maxTxPower \tag{3.1}$$

$$NDL\_minTxPower \leq NDL\_defTxPower \leq NDL\_maxTxPower \tag{3.2}$$

Úprava síly vysílacího signálu

$$effTxPower = \min(NDL\_refTxPower(acPrio), effTxPower) \tag{3.3}$$

3.5.3 Transmit Rate Control

Příjem paketů z komponenty DCC net a jejich řazení do odesílacích front musí splňovat časové omezení definované Transmit Rate Control (TRC) mechanismem. Časové omezení se vztahují na dobu přenosu paketu a také na interval mezi odesíláním paketů patřící do jedné odesílací fronty.

Parametr	Definice
NDL_maxPacketDuration	maximální doba přenosu paketu
NDL_minPacketInterval	minimální interval mezi přenosem jednotlivých paketů
NDL_maxPacketInterval	maximální interval mezi přenosem jednotlivých paketů
NDL_defPacketInterval	výchozí interval mezi přenosem jednotlivých paketů
NDL_refPacketInterval	interval mezi přenosem jednotlivých paketů dle referenční tabulky

Tabulka 3.22: Parametry **TRC**, dle zdroje [5], str. 12, tabulka 2

Tabulka 3.22 popisuje NDL parametry použité TRC mechanismem. Parametry NDL_refPacketInterval a NDL_defPacketInterval jsou omezeny dle rovnic 3.4 a 3.5.

$$NDL\_minPacketInterval \leq NDL\_refPacketInterval \leq NDL\_maxPacketInterval \tag{3.4}$$

$$NDL\_minPacketInterval \leq NDL\_defPacketInterval \leq NDL\_maxPacketInterval \tag{3.5}$$

Doba přenosu paketu (T_{AIR}) je spočítána z doby přenosu hlavičky paketu, počtu bitů v OFDM (N_{DBPS}) dle zvolené přenosové rychlosti (12 bitů na každých 3Mit/s) dle rovnice 3.7.

$$N_{PR} = (T_{PREAMBLE} + T_{SIGNAL})/T_{SYMBOL} \tag{3.6}$$

$$T_{AIR} = (N_{PR} + N_{SYMBOL}) \times T_{SYMBOL} \tag{3.7}$$

Dle zdroje [1] a [5], jsou hodnoty $N_{PR} = 5$, $N_{SYMBOL} = (8 \times pkt\_len/N_{DBPS})$ a $T_{SYMBOL} = 8\mu s$ pro 10 MHz kanál.

Například pro paket dlouhý 100B (pkt_len = 100), odeslaný rychlostí 3Mbit/s je doba přenosu $T_{AIR} = 568\mu s$. Výpočet dle rovnice 3.8.

$$\begin{aligned} T_{AIR} &= (N_{PR} + N_{SYMBOL}) \times T_{SYMBOL} \\ &= (N_{PR} + (8 \times pkt\_len/N_{DBPS})) \times T_{SYMBOL} \\ &= (5 + (8 \times 100/12)) \times 8 \\ &= 568 \end{aligned} \tag{3.8}$$

Intervaly přenosů mezi pakety jsou počítány mezi začátky vysílání jednotlivých paketů.

Po přijetí paketu z DCC net, je z délky paketu a zvolené rychlosti vypočítán T_{AIR}. Pokud je hodnota T_{AIR} menší než NDL_maxPacketDuration, je paket vložen do odesílací fronty dle zvoleného DCC profilu. Pokud ne, je paket zahozen. TDC může zvýšit přenosovou rychlost a snížit tak předpokládanou dobu přenosu.

3.5.4 Transmit Datarate Control

DCC využívá k odesílání paketů OFDM systém s šířkou kanálu 10 MHz. Možné přenosové rychlosti v 10MHz kanálu jsou 3, 4.5, 6, 9, 12, 18, 24 a 27 Mbit/s. Tyto rychlosti jsou vyjádřeny MCS indexem.

Parametr	Definice
NDL_minDatarate	minimální rychlost přenosu
NDL_maxDatarate	maximální rychlost přenosu
NDL_defDatatrate	výchozí rychlost přenosu
NDL_refDatarate	rychlost přenosu dle referenční tabulky

Tabulka 3.23: Parametry **TDC**, dle zdroje [5], str. 13, tabulka 3

Parametry NDL_defDatatrate a NDL_refDatarate z tabulky 3.23 musí splňovat omezení rovnic 3.10 a 3.9.

$$NDL\_minDatarate \leq NDL\_refDatarate \leq NDL\_maxDatarate \tag{3.9}$$

$$NDL\_minDatarate \leq NDL\_defDatarate \leq NDL\_maxDatarate \tag{3.10}$$

V případě, že má paket z DCC app nastavenou rychlost přenosu *effTxDatarate*, je tato rychlost porovnána s referenční hodnotou NDL_refDatarate(acPrio) a upravena dle vztahu 3.11. Pokud rychlost přenosu *effTxDatarate* není z DCC app nastavena, je automaticky použita rychlost přenosu určená referenční hodnotou.

Jestliže doba přenosu paketu vypočítaná v TRC (T_{AIR}) přesahuje NDL_maxPacketDuration, TDC může zvolit vyšší přenosovou rychlost, aby se zamezilo zahození paketu.

$$effTxDatarate = \max(NDL\_refDatarate(acPrio), effTxDatarate) \tag{3.11}$$

3.5.5 DCC Sensitivity Control

Mechanismus DCC Sensitivity Control (DSC) indikuje využití vysílače při odesílání a řídí citlivost přijímače. Při příjmu paketu bez preambule s úrovní signálu větší než NDL_refCarrierSense, je vysílán *busy* signál.

Parametr	Definice
NDL_minCarrierSense	minimální citlivost přijímače
NDL_maxCarrierSense	maximální citlivost přijímače
NDL_defCarrierSense	výchozí citlivost přijímače
NDL_refCarrierSense	citlivost přijímače dle referenční tabulky

Tabulka 3.24: Parametry **DSC**, dle zdroje [5], str. 14, tabulka 4

DSC upravuje NDL parametry popsané v tabulkce 3.24. Parametry NDL_refCarrierSense a NDL_defCarrierSense jsou omezeny dle vztahů 3.12 a 3.13.

$$NDL\_minCarrierSense \leq NDL\_refCarrierSense \leq NDL\_maxCarrierSense \tag{3.12}$$

$$NDL\_minCarrierSense \leq NDL\_defCarrierSense \leq NDL\_maxCarrierSense \tag{3.13}$$

3.5.6 Transmit Access Control

Mechanismus Transmit Access Control (TAC) reguluje přístup stanice ke komunikačnímu kanálu. Pokud je komunikační kanál vytížen, TAC omezí možnost vysílání stanicím s největším počtem odeslaných paketů (za předpokladu, že všechny stanice mají TAC).

Parametr	Definice
NDL_numQueue	počet odesílacích front
NDL_refQueueStatus	aktuální stav fronty

Tabulka 3.25: Parametry **TAC**, dle zdroje [5], str. 15, tabulka 5

TAC omezuje vysílání změnou aktuálního stavu odesílacích front. Odesílací fronty mohou být ve stavu *OPEN*, nebo *CLOSED*. Fronty jsou řazené dle priority q, kde nejvyšší priorita je q = 0. Aktuální statistiky odesílání *txChannelUse* pro každou odesílací frontu, jsou porovnány se statistikami *NDL_tmChannelUse* v Transmit modelu. Stavy odesílacích front jsou nastavovány dle rovnice 3.14.

$$NDL\_refQueueStatus = \begin{cases} CLOSED & txChannelUse \geq NDL\_tmChannelUse \\ OPEN & txChannelUse < NDL\_tmChannelUse \end{cases} \tag{3.14}$$

Paket určený k odeslání, který má být zařazen do odesílací fronty ve stavu *CLOSED* je zahozen.

3.5.7 Transmit model

Komponenta DCC access odhaduje předpokládané využití kanálu při odesílání paketů pomocí *Transmit Modelu* (TM). Tento model je porovnáván s aktuálními statistikami odesílání paketů pro účely výše zmíněných mechanismů.

Parametr	Definice
NDL_tmPacketArrivalRate	očekávaný počet paketů za sekundu
NDL_tmPacketAvgDuration	očekávaná průměrná doba přenosu paketu
NDL_tmPacketAvgPower	očekávaná průměrná síla vysílacího signálu
NDL_tmChannelUse	souhrnné očekávané využití kanálu
NDL_maxChannelUse	maximální využití kanálů

Tabulka 3.26: Parametry **Transmit Modelu**, dle zdroje [5], str. 16, tabulka 6

V tabulce 3.26 jsou popsány NDL parametry *Transmit modelu*. Souhrnné očekávané využití kanálu *NDL_tmChannelUse* je spočítáno z parametrů *NDL_tmPacketAvgDuration* a *NDL_tmPacketArrivalRate* dle rovnice 3.15. Parametr *NDL_maxChannelUse* je vypočítán z rovnice 3.16.

$$NDL\_tmChannelUse(acPrio) = \\ = \sum_{n=0}^{acPrio} NDL\_tmPacketAvgDuration(n) \times NDL\_tmPacketArrivalRate(n) \tag{3.15}$$

$$NDL\_maxChannelUse = NDL\_tmChannelUse(NDL\_numQueue-1) \tag{3.16}$$

3.5.8 Receive model

Receive model (RM) se v DCC systému používá pro odhad komunikační a detekční vzdálenosti. DCC access poskytuje tyto odhady DCC mgmt komponentě. Vzdálenosti jsou počítány mezi přijímačem a vysílačem v prostředí se ztrátovým parametrem NDL_refPathloss.

Parametr	Definice
NDL_defDccSensitivity	výchozí citlivost přijímače
NDL_maxCsRange	maximální CS vzdálenost detekování
NDL_refPathloss	ztrátový parametr
NDL_minSNR	minimální SNR pro dekódování rychlosti MSC 0.

Tabulka 3.27: Parametry **Receive Modelu**, dle zdroje [5], str. 17, tabulka 7

Ztrátový parametr je závislý na aktuálním prostředí a pohybuje se v rozmezí daném rovnicí 3.17.

$$1.8 \leq NDL\_refPathloss \leq 4.0 \tag{3.17}$$

Výpočty odhadované komunikační a detekční vzdálenosti jsou uvedeny v rovnicích 3.18 a 3.19. Například odhadovaná komunikační vzdálenost při ztrátovém parametru $NDL\_refPathloss = 2.5$, použité síle a rychlosti vysílání $txPower = 13dBm$ a $datarate = 12Mbit/s$ je rovna 76m. Výpočet dle rovnice 3.19.

$$csRange(txPower) = \\ maxcsRange \times 10^{((txPower-NDL\_maxTxPower)\times 10/NDL\_refPathloss)} \tag{3.18}$$

$$estCommRange(txPower, datarate) = \\ csRange(txPower) \times 10^{((-\Delta SNR(datarate))\times 10/NDL\_refPathloss)} \tag{3.19}$$

Výpočty jsou provedeny pro každý paket a výsledky jsou poskytnuté DCC mgmt komponentě na vyžádání.

3.5.9 Channel probing

DCC systém shromažďuje informace o aktuálním stavu komunikačního kanálu. Sledované a měřené parametry jsou vypsány v tabulce 3.28. Metody měření těchto parametrů nejsou standardem definované, standard pouze definuje jejich význam.

Parametr	Definice
channelLoad	zatížení kanálu
loadArrivalRate	míra zatížení příchozími pakety
loadAvgDuration	průměrná doba zatížení
packetArrivalRate	četnost příchozích paketů
packetAvgDuration	průměrná doba příchodu paketu
channelBusyTime	doba po kterou je kanál v BUSY stavu

Tabulka 3.28: Měřené parametry stavu kanálu, dle zdroje [5], str. 20, tabulka 11

Všechny měřené parametry jsou závislé na hodnotě citlivosti přijímače *NDL_defCarrierSense* a pro účely měření a výpočtů jsou použity jen pakety s větší sílou přijímacího signálu než je *NDL_defCarrierSense*. Zaznamenává se jejich počet, síla přijímacího signálu, přenosová rychlost a také jejich délka. U paketů se silou přijímacího signálu menší než je *NDL_defCarrierSense*, se zaznamenává jen jejich počet. Měření se provádí vždy v pevných časových intervalech. Výpočty statistik jsou prováděny z několika posledních intervalů měření. Tím je docílena rychlejší odezva na změnu stavu komunikačního kanálu.

3.5.10 Transmit packet statistics

Komponenta DCC access udržuje statistiky týkající se odesílání paketů. Četnost odesílání paketů příslušné priority, průměrná doba přenosu odesílaného paketu, průměrná síla vysílacího signálu a doba využití kanálu jsou zaznamenávány komponentou DCC access a dále distribuovány do NDL tabulky v DCC mgmt komponentě.

Parametr	Definice
txPacketArrivalRate	četnost odesílání paketů
txPacketAvgDuration	průměrná doba přenosu paketu
txSignalAvgPower	průměrná síla vysílacího signálu
txChannelUse	poměrná doba využití kanálu

Tabulka 3.29: Měřené statistiky odesílání, dle zdroje [5], str. 20, tabulka 12

Výpočty parametrů se provádí vždy v pevně daném časovém intervalu. Četnost odesílání paketů je spočítána jako celkový počet odeslaných paketů vydělený délkou sledovaného časového intervalu. Při výpočtu průměrné doby přenosu paketu je použita rovnice 3.7 pro každý paket, a součet těchto časů je dělen celkovým počtem odeslaných paketů. Podobně je vypočítána i průměrná síla TX signálu. Součet síly vysílacího signálu pro všechny odeslané pakety je vydělen jejich počtem. Poslední

parametr je spočítán jako suma součinů *txArrivalRate* a *txPacketAvgDuration* přes všechny priority.

Kapitola 4

Architektura DCC pro Linux

DCC systém je rozsáhlý a skládá se z několika spolupracujících komponent. Tato kapitola obsahuje rozbor jednotlivých komponent, jejich potřeb a také možností OS Linux.

Nejprve bude nastíněno rozložení DCC komponent v OS Linux a podoba komunikačních zpráv a datových struktur použitých v systému. Následovat bude popis odesílání a příjmu dat z aplikace, měření a výpočty statistik a celkové schéma DCC systému v OS Linux.

4.1 Základní popis systému

V této kapitole jsou popsány základní prvky DCC systému. Rozmístění komponent v OS Linux, formáty použitých zpráv a popis tabulek DCC profilů a NDL.

4.1.1 Rozložení DCC komponent v OS Linux

Každá komponenta v DCC systému má svůj úkol, který v systému plní. Dle tohoto úkolu a potřeb komponenty bude každá komponenta umístěna v OS Linux.

DCC app, jako aplikace sloužící pro odesílání a příjem zpráv, které následně reprezentuje v uživatelském prostředí, bude umístěna do uživatelské části OS.

Komponenta DCC access má za úkol řídit odesílání paketů, kontrolovat parametry systému a na základě toho upravovat jeho chování. Pro svou činnost potřebuje přístup k ovladačům HW a také k odesílacímu procesu OS Linux. DCC access bude implementována jako modul jádra OS.

Komponenta DCC mgmt je z hlediska informací centrálním prvkem systému. Obsahuje tabulky s informacemi pro všechny komponenty. Každá komponenta je s ní spojena komunikačním rozhraním. Z tohoto důvodu bude nejlepší ji rozdělit na dvě části, jednu část implementovat jako modul jádra OS a druhou jako uživatelskou aplikaci, či aplikačního démona.

4.1.2 Formát použitých zpráv

Ve standardu je definováno několik typů zpráv, které se mohou vyskytovat v různých komponentách. Všechny komponenty tedy musí znát jejich formát. Zprávy budou reprezentovány datovou strukturou obsahující definované parametry dle kapitoly

3.2. Navíc bude každá struktura obsahovat proměnnou určující typ zprávy. Tuto proměnnou budou mít i datové struktury použité pro interní výměnu dat. Tím se sjednotí systém pro zpracování struktur, který bude záviset na typu zprávy.

4.1.3 Tabulka DCC profilů a NDL

Tabulka NDL bude stejně jako tabulka DCC profilů umístěna v komponentě DCC mgmt a budou reprezentovány poli datových struktur. Formát těchto struktur je definován ve zdroji [5] a [12]. Standard, viz zdroj [5], definuje kódování a rozsahy hodnot. Pro jednodušší zápis a čtení hodnot bude implementována sada pomocných funkcí. Protože je možné, že do tabulky bude přistupováno z více míst programu, bude nutné přístup k tabulce omezit zámkem. Proces čtení ani zápisu do tabulek nebude časově náročný, zpomalení systému z důvodu čekání na odemčení tabulky je tedy zanedbatelné. Protože se bude vždy číst jen z jedné zamknuté oblasti, nenastane ani podmínka uváznutí (deadlocku).

4.2 Komunikace mezi komponentami

V DCC systému je důležitá komunikace mezi komponentami. Ať už se jedná o předávání parametrů, nastavování rozhraní, či o samotné odesílání zpráv, komunikace musí být rychlá a spolehlivá. Komunikační rozhraní mezi komponentami DCC systému jsou dle standardu, viz zdroj [5], definované jako oboustranná, tedy obě komunikující strany mohou začít komunikovat. Pro komunikaci mezi procesy jádra OS bude použita dvojice *FIFO* front, každá pro jeden směr komunikace. Jejich použití je jednoduché a za předpokladu, že bude vždy jeden producent a jeden konzument, není potřeba je zamykat.

4.3 Odesílání zpráv z aplikace

V této části kapitoly je popsána posloupnost událostí a akcí v DCC systému při odesílání zpráv z aplikace. Aplikace připraví paket s daty a s nastavenými parametry odesílání ji odešle. Paket je předán do DCC access komponenty, kde proběhne kontrola a dodatečné nastavení parametrů před samotným odesláním. Poté je paket předán do síťové karty k odeslání do sítě.

4.3.1 Příprava zprávy v DCC app

Komponenta DCC app vloží data do datové struktury definované v tabulce 3.1. Spolu s daty vyplní i ostatní parametry struktury. Zvolí aktuální referenční číslo zprávy, vyplní adresy, číslo DCC profilu a také rychlost odesílání a sílu vysílacího signálu. Tyto parametry jsou buď určené aplikací dle typu zprávy, například důležité zprávy budou mít vyšší rychlost odesílání než běžné zprávy, a nebo je DCC app nechá nevyplněné. Pro nastavení skutečné priority paketu je možné využít políčko *PRIORITY* v paketu. Hodnota tohoto políčka je brána jako priorita IPv6 při směrování paketu. Takto naplněnou strukturu odešle aplikace pomocí standardního

UDP/IPv6 na adresu *BROADCASTu*. Bezdrátová síťová karta bude nastavena v režimu OCB, zprávy se tedy budou posílat na *BROADCAST* adresu do nastaveného pásma.

4.3.2 Zpracování zpráv v DCC access

Komponenta DCC access potřebuje zpracovávat pakety z vyšších vrstev a následně je rozřazovat do určitých vysílacích front dle jejich parametrů. Z toho plyne, že část komponenty DCC access musí být queueing disciplína. Tím získá přístup ke všem paketům určených k odeslání a také bude mít možnost je pak zařadit do určitých vysílacích front.

Protože DCC access získá přístup ke všem odchozím paketům, po přijetí paketu k odeslání bude potřeba zjistit, zda se jedná o paket DCC systému, nebo jiné komunikace. Pro tyto účely bude na začátku datové části DCC paketu proměnná, která bude identifikovat DCC paket. Pokud se bude jednat o jiný paket, zvolí se odesílací fronta s nejnižší prioritou a paket se do ní zařadí, tím bude vliv ostatní komunikace na funkci DCC systému minimální.

V případě DCC paketu, bude přečtena hlavička DCC struktury a tím budou zjištěny parametry odesílání nastavené z DCC app (ID DCC profilu, rychlost odesílání a síla vysílacího signálu). Nyní je potřeba získat od DCC mgmt komponenty informace ohledně zvoleného DCC profilu. K tomu se použije komunikační *FIFO*, která spojuje DCC access a DCC mgmt komponenty. Po získání parametrů DCC profilu, budou spuštěny DCC mechanismy. Nejprve se pomocí mechanismu TAC, viz kapitola 3.5.6, zjistí aktuální stav odesílací fronty určené DCC profilem a v případě stavu *CLOSED* bude paket zahozen. Dále bude spuštěn mechanismus TRC, viz kapitola 3.5.3, který nejprve zaznamená dobu příchodu paketu do DCC access a spočítá čas uplynulý od příchodu předchozího paketu stejné priority. Tento čas porovná s parametrem *NDL_maxPacketInterval*. Poté spočítá dobu přenosu paketu dle rovnice 3.7 a porovná ji s parametrem *NDL_maxPacketDuration*. Pokud některá z vypočítaných hodnot bude větší než definované maximální limity, paket bude zahozen. Nakonec se spustí mechanismy TDC, viz kapitola 3.5.4, a TPC, viz kapitola 3.5.2. Tyto mechanismy pouze upraví parametry odesílání dle rovnic 3.11 a 3.3. Upravený paket bude zařazen do vybrané odesílací fronty a zpět do aplikace bude odeslána informace o úspěšném odeslání paketu, obsahující aktuální hodnoty parametrů odesílání. V případě zahození paketu v některém předchozím kroku, aplikace bude také informována, jen bude stav odeslání označen jako neúspěšný.

Tato informace by mohla být do DCC app dopravena pomocí paketu. DCC access by vytvořila normální paket a odeslala ho na lokální IPv6 adresu na určený port aplikace.

Pakety budou z odesílacích front odebírány dle priority, od nejvyšší. Po odebrání z fronty, bude paket postoupen dále k fyzickému odeslání do sítě.

4.3.3 Odeslání paketu do sítě

Před samotným odesláním do sítě, podsystém IEEE-80211 OS Linux určí přenosovou rychlost odesílání a sílu signálu. Součástí podsystému IEEE-80211 je Minstrel algoritmus. Ten vybírá přenosovou rychlost pro každý paket.

Pro účely DCC systému je potřeba zaručit, aby paket byl odeslán přenosovou rychlostí určenou v DCC access komponentě. To je s použitím stávajícího Minstrel algoritmu problém.

Jako možné řešení se nabízí tři možnosti:

1. změnit nastavení bezdrátové síťové karty a povolit jen požadované přenosové rychlosti
2. upravit Minstrel algoritmus
3. upravit ovladač síťové karty ath9k

Možnost číslo 1, změna nastavení bezdrátové síťové karty a povolení jen požadované přenosové rychlosti, je z hlediska implementace ta nejjednodušší. Bohužel ale není možné zaručit, že bude pro vybraný paket použita právě změněná přenosová rychlost. Také doba reakce na změnu povolených přenosových rychlostí není dostačující.

Jako další možnost se nabízí úprava Minstrel algoritmu, možnost číslo 2. Upravený algoritmus by mohl číst hlavičky paketů a pro DCC pakety změnit přenosovou rychlost dle nastavení DCC access komponenty. Ostatní pakety by byly zpracovány normálně dle algoritmu Minstrel.

Poslední varianta, číslo 3, úprava ovladače síťové karty ath9k. Z hlediska paketu a jeho přenosové rychlosti, poslední kdo ji může změnit, je samotný ovladač síťové karty. Informace o konečných parametrech přenosu jsou ukládány v proměnné **cb** ve struktuře paketu *sk_buff*, viz obrázek 2.4. Úprava ovladače karty by byla podobná jako úprava Minstrel algoritmu, tedy pro pakety DCC systému ignorovat nastavení od RCA a nastavit přenosovou rychlost dle komponenty DCC access.

Možnost číslo 1 nemůže zaručit nastavení přenosové rychlosti pro určitý paket. Není proto vyhovující pro použití v DCC systému. Možnost číslo 3 je závislá na použití specifické síťové karty. Po této úvaze vidím jako jedinou možnost řešení číslo 2, tedy upravit stávající Minstrel algoritmus dle návrhu výše.

Nastavení síly vysílacího signálu by mohlo být provedeno ve stejném kroku, jako nastavení přenosové rychlosti. Potřebné informace jsou uložené v proměnné **cb** ve struktuře *sk_buff*. Upravený Minstrel algoritmus by je tedy mohl změnit dle hodnot uvedených v paketu.

Po nastavení přenosové rychlosti a síly vysílacího signálu je paket fyzicky odeslán do sítě.

4.4 Příjem zpráv od jiné ITS stanice

Tato kapitoly, podobně jako kapitola 4.3, popisuje posloupnost událostí a akcí v DCC systému při příjmu zpráv od jiné ITS stanice.

4.4.1 Zpracování zpráv v DCC access

Standard definuje různé formáty zpráv pro zprávu odeslanou a zprávu přijatou, viz tabulky 3.1 a 3.3.Z jedné ITS stanice je paket ve formátu *IN-UNITDATA.request*

odeslaný do sítě a následně přijatý v druhé ITS stanici, dle standardu by bylo nutné jeho formát po přijetí změnit na formát *IN-UNITDATA.indication*. Tuto změnu by bylo možné provést v komponentě DCC access s použitím Netfilter frameworku. Přijatý paket by byl upraven dle požadavků a pokračoval by v cestě do DCC app. Do paketu by byly přidány informace o přijímací rychlosti, síly přijímacího signálu a čísle kanálu, získané od síťové karty. Tyto informace jsou použity při výpočtu statistiky příchozích paketů.

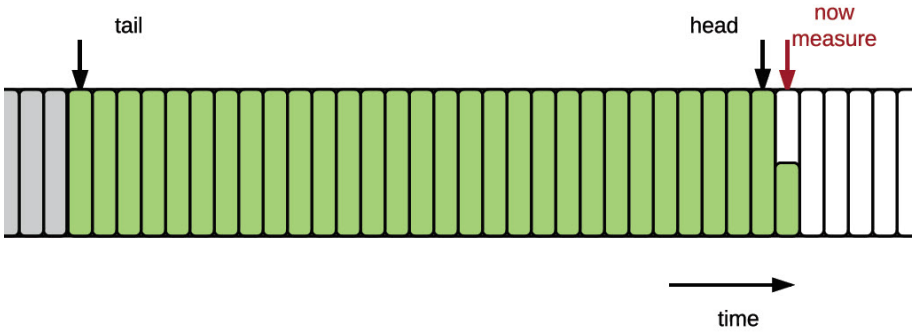
Osobně v tomto kroku nevidím žádnou výhodu. Obě zprávy obsahují v podstatě stejné informace. Zmíněné informace navíc mohou být poslány s paketem jako metadata ve struktuře *RADIOTAP* pomocí PCAP API, viz kapitola 2.7. Komponenta DCC app by mohla při přijetí paketu tyto informace získat z právě zmíněné struktury. Navíc pro počítání statistik a zatížení kanálu, je potřeba zaznamenat všechny pakety kolující ve vysílacím pásmu, ne jen pakety určené pro lokální ITS stanici.

4.4.2 Zpracování zpráv v DCC app

Komponenta DCC app bude připojena na definovaný port, na kterém bude očekávat příchozí pakety, které po přijetí zpracuje. Když vezmu v úvahu předchozí nápad s informacemi o síle a rychlosti přijímaného signálu ve struktuře *RADIOTAP* v podobě metadat k přijímanému paketu, byly by tyto informace zpracovány společně s paketem. Tím by se ušetřil čas potřebný k úpravě paketu v jádře OS.

4.5 Měření statistik

DCC access shromažďuje informace o paketech a počítá z nich statistiky přenosů. Jak je popsáno v kapitolách 3.5.9 a 3.5.10. V této kapitole nejprve popíši způsob měření a výpočtů statistik přijímaných paketů a následně i výpočty statistik odesílaných paketů.



Obrázek 4.1: Nákres způsobu zaznamenávání statistik

4.5.1 Statistiky přijímaných paketů

Pro účely statistik přijímaných paketů se započítávají všechny pakety přítomné v komunikačním kanálu, nejen ty určené pro aktuální ITS stanici. Komunikační

kanál lze odposlouchávat a tím získat všechny přenášené pakety. Odposlouchávání komunikačního kanálu je možné díky *MONITOR* režimu bezdrátové síťové karty. V tomto režimu není možné vysílat pakety do sítě, to lze pouze v režimu *MANAGED*. Některé bezdrátové síťové karty podporují provoz *MONITOR* a *MANAGED* režimu současně. Bezdrátová síťová karta Arheros 9000 bohužel tento současný provoz nepodporuje. Pokud tedy budeme sbírat pakety za účelem počítání statistik, nebudeme schopni vysílat.

Jako řešení tohoto problému mě napadají dvě možnosti.

1. v pravidelných intervalech střídat režim standardní komunikace a režim sbírání dat pro statistiky
2. jednu WiFi kartu použít pro sběr dat pro statistiky a využít další WiFi kartu pro standardní komunikaci

Způsob řešení problému číslo 1 není vhodný z důvodu pravidelných časových intervalů, ve kterých není možné komunikovat. To v kritických situacích není přípustné. Ani rychlost přepínání WiFi karty mezi režimy *MONITOR* a *MANAGED* není dostačující.

Z těchto důvodů bych zvolil možnost řešení číslo 2. Zde je problém jen absence druhé WiFi karty. To lze ovšem vyřešit připojením další WiFi karty do USB konektoru.

Zaznamenávané a počítané parametry jsou popsány v kapitole 3.5.9. Statistiky je potřeba sledovat pravidelně v daných časových úsecích. Zvolím sledovaný úsek dlouhý 10 sekund. Sledovaný úsek bude rozdělen do menších měřených úseků, každý dlouhý 1 milisekundu. To znamená, že naměřená data budou zaznamenána každou milisekundu. Statistiky budou počítány z celkového sledovaného úseku 10 sekund, který se po milisekundách bude posouvat v čase.

Na obrázku 4.1 je znázorněn způsob zaznamenávání údajů pro statistiky. Sledovaný úsek mezi ukazatelem *head* a *tail* bude rozdělen na menší měřené úseky. Sledovaný úsek se bude pohybovat v čase a statistiky se budou přepočítávat s každým pohybem. Způsob výpočtů statistik bude popsán v následující kapitole 5.

4.5.2 Statistiky odesílaných paketů

Statistiky odesílaných paketů budou počítány v komponentě DCC access před zařazením jednotlivých paketů do odesílacích front pro každou prioritu *acPrio*. Pro každý paket bude zaznamenána síla vysílacího signálu, zvolená přenosová rychlost, čas příchodu do DCC access a také jeho délka. Způsob výpočtů statistik je popsán v kapitole 3.5.10 a v kapitole 5 bude upřesněn.

4.6 Řídící smyčka DCC access

Chování řídící smyčky komponenty DCC access bylo popsáno v kapitole 3.5.1. Doplním, že smyčka bude implementována jako vlákno a bude spouštěna v časových intervalech daných parametrem *NDL_minDccSampling* z NDL tabulky.

4.7 Schéma návrhu DCC systému

Z výše uvedených úvah nad realizací DCC systému v OS Linux, jsem vytvořil celkové schéma rozložení DCC systému v OS, viz obrázek 4.2.

4.7.1 DCC app

Implementována bude jen základní funkcionalita, žádné uživatelské prostředí. Aplikace bude obsahovat dva procesy. První proces bude periodicky v daných časových úsecích vytvářet a odesílat zprávy do sítě. Druhý proces bude přijímat zprávy na portu "47474"a vypisovat informaci o jejich přijetí.

4.7.2 DCC mgmt

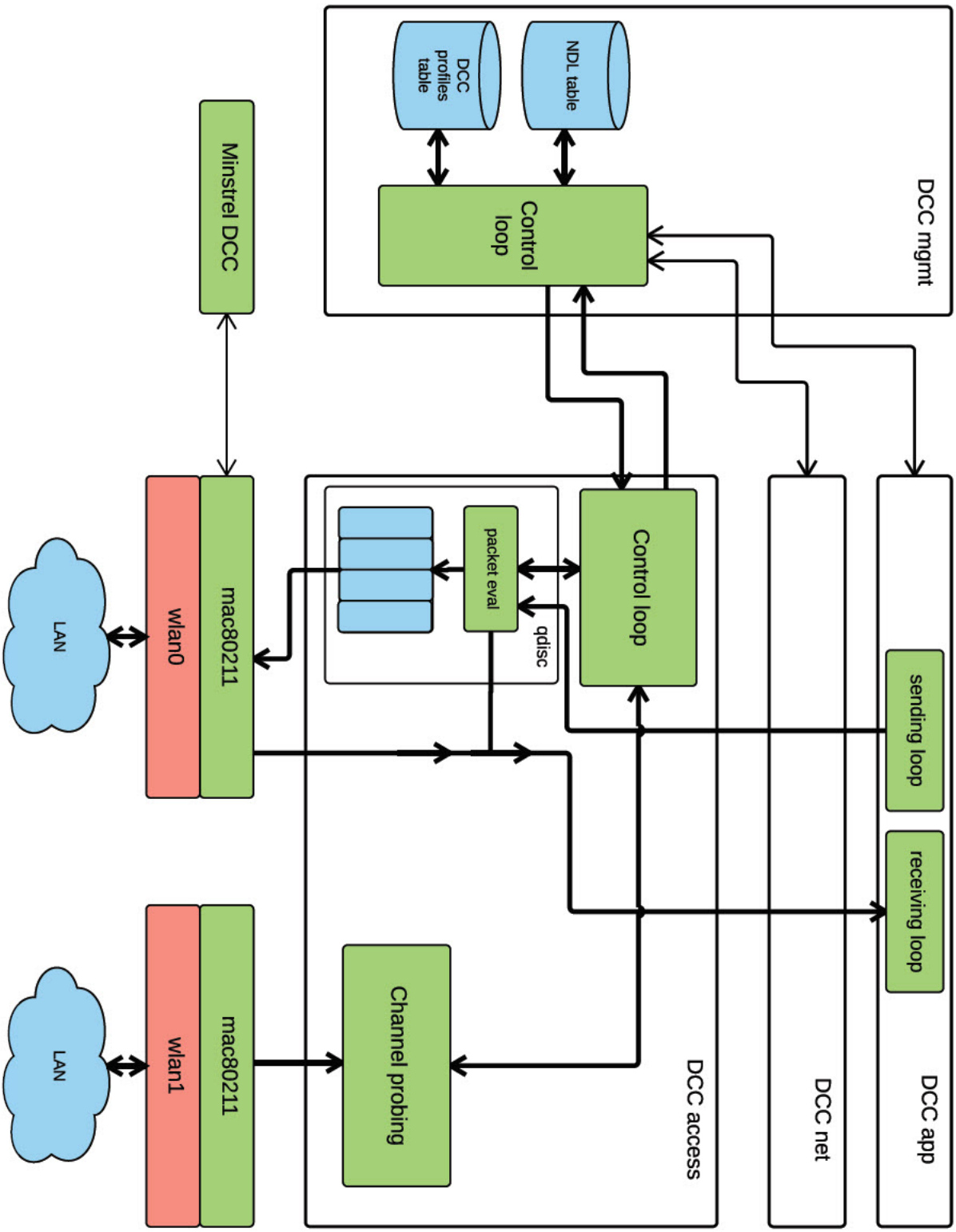
Součástí DCC mgmg budou tabulka DCC profilů a NDL tabulka. Komponenta bude obsahovat řídící smyčku, která bude kontrolovat příjem požadavků z DCC access či DCC app a obsluhovat je, tj. číst a zapisovat data z tabulek.

4.7.3 DCC access

Komponenta DCC access bude rozdělena na dvě základní části. První část bude tvořena řídící smyčkou qdisc. Qdisc přijme paket a pomocí řídící smyčky, která bude mít na starosti mimo jiné i komunikaci s DCC mgmt, požádá o parametry odesílání z tabulky DCC profilů. Poté provede kontrolu podmínek pro zařazení paketu do odesílacích front a zařadí jej. Následně odešle informační zprávu zpět do DCC app. Druhá část bude sbírat a počítat statistiky, které následně pomocí řídící smyčky přenese do NDL tabulky.

4.7.4 Struktura HW

V systému budou zapojeny dvě WiFi karty. Jedna poběží v režimu *MONITOR* a bude sbírat pakety pro statistiky. Druhá bude sloužit pro standardní komunikaci v OCB módu. Jako RCA bude použit upravený Minstrel algoritmus.



Obrázek 4.2: Schéma návrhu DCC systému v OS Linux

Kapitola 5
Implementace

Tato kapitola obsahuje popis implementace DCC systému pro OS Linux. Nejprve je popsána implementace struktur a změny oproti návrhu v kapitole 4. Následuje popis implementace každé komponenty a také způsobu jejich spolupráce. Z velké části navazuji na návrhy z kapitoly 4. Nakonec je popsána celková struktura implementovaného DCC systému a část implementace, kterou jsem nedokončil.

5.1 Struktury tabulek a DCC zpráv

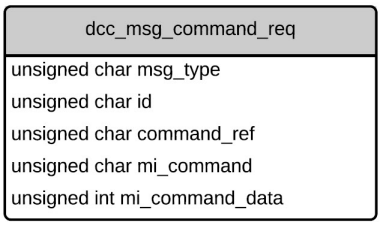
V implementaci jsou použité následující datové struktury. Všechny struktury mají na začátku proměnnou *msg_type*, to sjednotí čtení a zjednoduší rozpoznání typu zprávy.

- dcc_msg_command_req
- dcc_msg_command_conf
- dcc_msg_params
- dcc_msg_data
- dpType_dp_table
  - dpType_dcc_profile
- ndlType_ndl_database
  - ndlType_transmit_power_tresholds
  - ndlType_packet_timing_tresholds
  - ndlType_packet_datarate_tresholds
  - ndlType_receive_signal_tresholds
  - ndlType_receive_model_parameters
  - ndlType_demodulation_model_parameters
  - ndlType_transmit_model_parameters
  - ndlType_channel_load_tresholds

- ndlType_transmit_queue_parameters
- ndlType_communication_ranges
- ndlType_channel_load_measures
- ndlType_transmit_packet_statictics
- ndlType_general_configuration
- ndlType_state_configuration

5.1.1 Struktura dcc_msg_command_req

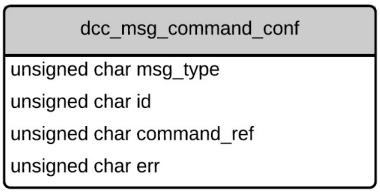
Zprávy *MI-COMMAND.request*, *MI-REQUEST.request*, *IM-DP-COMMAND.request* a *IM-DP-REQUEST.request* jsou implementovány strukturou **dcc_msg_command_req**.



Obrázek 5.1: Struktura dcc_msg_command_req

5.1.2 Struktura dcc_msg_command_conf

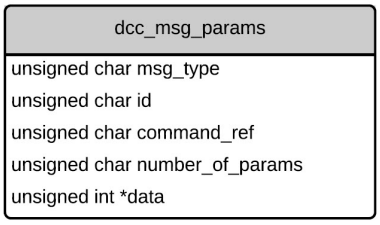
Zprávy *MI_COMMAND.confirm*, *MI_REQUEST.confirm*, *IM-DP_COMMAND.confirm* a *IM-DP_REQUEST.confirm* jsou v aplikaci implementovány strukturou **dcc_msg_command_conf**.



Obrázek 5.2: Struktura dcc_msg_command_conf

5.1.3 Struktura dcc_msg_params

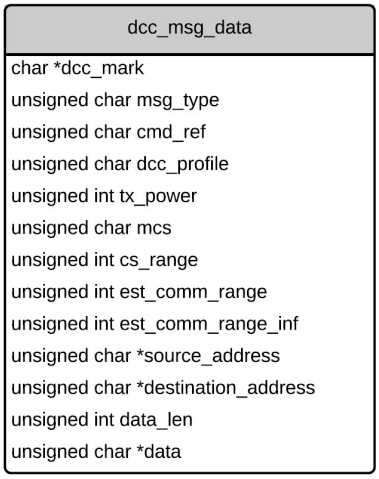
Zprávy *MI_SET.request*, *MI_SET.confirm*, *MI_GET.request*, *MI_GET.confirm*, *IM-DP_GET.request* a *IM-DP_GET.confirm* jsou v aplikaci implementovány strukturou **dcc_msg_params**.



Obrázek 5.3: Struktura dcc_msg_params

5.1.4 Struktura dcc_msg_data

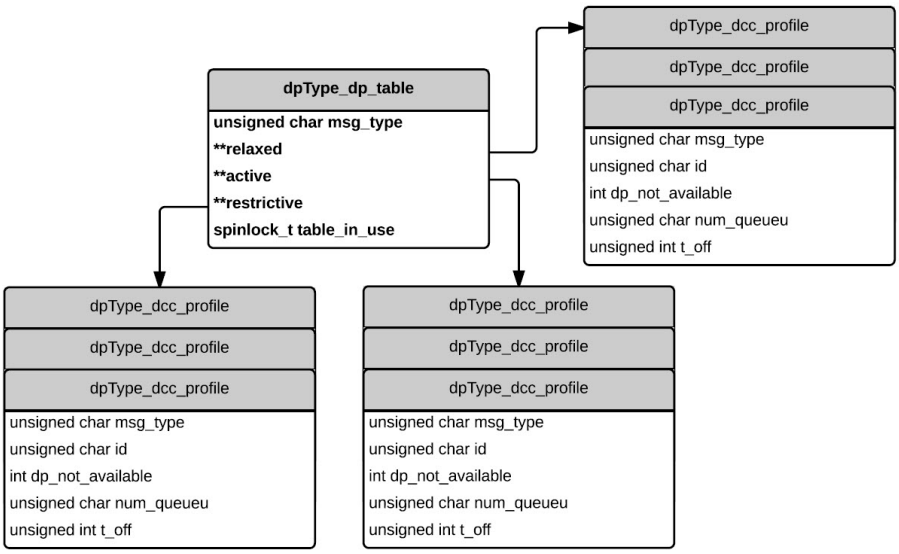
Pro zprávy *IN-UNITDATA.request*, *IN-UNITDATA.status* a *IN-UNITDATA.indication*, které jsou odesílané a přijímané komponentou DCC app, je implementována struktura **dcc_msg_data**.



Obrázek 5.4: Struktura dcc_msg_data

5.1.5 Struktura dpType_dp_table

Struktura tabulky DCC profilů je znázorněna na obrázku 5.5. Obsahuje ukazatele na pole struktur *dpType_dcc_profile*. Schéma struktury *dpType_dcc_profile* je také na obrázku 5.5.



Obrázek 5.5: Struktura dpType_dp_table a podstruktura dpType_dcc_profile

Kromě ukazatelů na podstruktury, tabulka obsahuje také mutex *table_in_use*.

5.1.6 Struktura ndlType_ndl_database

Na obrázku 5.6 je zobrazeno schéma struktury *ndlType_ndl_database* reprezentující NDL tabulku. NDL tabulka je hlavní informační zdroj, proto je tato struktura takto rozsáhlá. Každý parametr v NDL tabulce je zakódován dle standardu, viz zdroj [5]. V aplikaci jsou proto implementovány podpůrné funkce pro manipulaci s takto zakódovanými hodnotami. Struktura se zamyká pomocí mutexu *table_in_use*.

5.2 Změny a úpravy oproti návrhu v kapitole 4

V průběhu implementace jsem učinil některé změny oproti předchozímu návrhu.

5.2.1 Lokální tabulky

První úprava se týká umístění NDL tabulky a tabulky DCC profilů. V implementaci jsou tyto tabulky umístěny ve všech komponentách. DCC mgmt nadále obsahuje a spravuje hlavní, neboli globální tabulky, ale pro zrychlení odezvy a ukládání hodnot byly do ostatních komponent přidány lokální tabulky. Tyto lokální tabulky se synchronizují s globálními jednou za sekundu. To je interval odpovídající reakční době uvedené ve standardu, viz [5], strana 21.

5.2.2 Měření a výpočty příchozích statistik v DCC app

Měření a výpočty příchozích statistik, dle kapitoly 4.5 měla vykonávat komponenta DCC access. Z důvodu komplexní matematiky (rovnice 3.18 a 3.19) při měření a výpočtu statistik s návazností na omezení jádra OS Linux, dle kapitoly 2.1.1, jsem tyto výpočty přesunul do DCC app. Spolu s příchozími pakety jsou přijímána i *METADATA* ve struktuře *RADIOTAP*, jak jsem uvažoval v kapitole 4.4. Z této struktury jsou získávány hodnoty pro výpočty statistik.

Výsledky výpočtů jsou rozesílány pomocí Netlink zpráv do DCC mgmt.

5.3 Komunikace mezi komponentami

Komponenty které jsou umístěny v jádře OS, mezi sebou komunikují pomocí *FIFO* front. Komunikační rozhraní mezi DCC mgmt a DCC access, číslo 2 dle obrázku 3.1, je tvořeno dvěmi *FIFO* frontami, dle návrhu v kapitole 4.2. V DCC access i v DCC mgmt komponentě je obslužný proces, který se stará o čtení zpráv z příchozích front. Do front jsou zařazeny ukazatele na struktury, které jsou vytvořeny v jedné komponentě a uvolněny ve druhé. Není tedy potřeba kopírovat data do fronty celá, ale jen jejich ukazatele.

Komunikace mezi komponentou DCC app a DCC mgmt je implementována pomocí Netlink protokolu. Komponenta DCC mgmt po zavedení do jádra OS zaregistruje Netlink rodinu s názvem *"DCC_PROTOCOL"*. Implementovány jsou tři typy zpráv. Zprávy na výměnu NDL parametrů a také zpráva na odesílání naměřených statistik z DCC app.

5.4 Implementace DCC app

Aplikace je rozdělena do tří procesů. Proces odesílání dat, proces příjmu dat a proces zpracování příchozích statistik. Po spuštění se inicializuje lokální NDL tabulka a tabulka DCC profilů. Dále se spustí výše zmíněné procesy a hlavní proces očekává příkaz na příkazové řádce. Příkazem *exit* je aplikace ukončena. Aplikaci je nutné spouštět s příkazem *sudo*.

5.4.1 Proces odesílání dat

Pro účely této práce bylo implementováno jen základní odesílání paketů. Odesílání je prováděno v časových intervalech 300ms. Proces vytvoří paket, nastaví cílovou adresu *BROADCASTu* a port *4747*. Dále nastaví prioritu a naplní paket daty. Data jsou tvořena strukturou *dcc_msg_data*, viz kapitola 5.1.4. Referenční číslo ve struktuře je inkrementováno pro každý odesílaný paket. Takto naplněný paket odešle pomocí funkce *sendto*.

5.4.2 Proces příjmu dat

Proces pro přijímání paketů naslouchá na portu *4747*. Po příchodu paketu zkontroluje zda se jedná o DCC paket a následně paket zpracuje. Vypíše informaci o přijetí paketu s určitým referenčním číslem.

5.4.3 Proces zpracování statistik

Získávání dat k výpočtům příchozích statistik je prováděno pomocí PCAP API v procesu pro zpracování statistik. Po spuštění aplikace je PCAP API nastaveno na odchytávání paketů z *wlan0* v *MONITOR* režimu. Následně je proces měření a výpočtů spuštěn. Proces měření probíhá způsobem popsaným v kapitole 4.5.1. Sledovaný časový úsek je 10 sekund a je rozdělen na měřící úseky dlouhé jednu milisekundu. U každého paketu se zkontroluje síla přijímacího signálu a pokud je větší než *NDL_defDccSensitivity*, pokračuje ke zpracování. Při zpracování se z každého paketu zaznamená síla přijímacího signálu, rychlost a také celkový počet přijatých paketů. Aritmetický průměr takto zaznamenaných hodnot je uložen do pole na pozici ukazatele *head*. Ukazatel se posouvá po každém cyklu měřícího procesu (jako cyklická fronta). Statistiky se počítají každou sekundu z celého úseku dlouhého 10 sekund. Vypočítané statistiky jsou následně protokolem Netlink odeslány do globální NDL tabulky v DCC mgmt.

5.5 Implementace DCC access

Komponenta DCC access je rozdělena do dvou částí, qdisc a řídící smyčku. Qdisc zpracovává pakety k odeslání pomocí DCC mechanismů a připravuje data pro výpočet statistik odesílání. Řídící smyčka synchronizuje lokální tabulky s globálními, počítá statistiky odesílání a upravuje aktuální stav DCC access komponenty dle zatížení kanálu.

5.5.1 Qdisc

Po zavedení modulu do jádra OS a zaregistrování qdisc k příslušné síťové kartě, je spuštěna inicializace qdisc, ve které jsou vytvořeny 4 odesílací fronty typu *FIFO*. Qdisc je připravena na příjem paketů k odeslání. Paket k odeslání je nejprve v záchytném bodu Netfilter frameworku zastaven a jde-li o DCC paket, je časově označkován. Poté pokračuje do qdisc. Qdisc přijme paket k odeslání a přečte jeho hlavičku. Provede kontrolu zda se jedná o DCC paket a pokud ne, vybere odesílací frontu s nejmenší prioritou a paket do ní přesune. Pokud se jedná o DCC paket, qdisc zkopíruje všechny informace do struktury *dcc_msg_data*. Z lokální tabulky DCC profilů si zjistí parametry odesílání (číslo odesílací fronty a interval odesílání). Dále je paket zpracován DCC mechanismy, které upraví jeho parametry a v případě nesplnění některé z podmínek, paket zahodí. Tento proces je podrobně popsán v kapitole 3.5. Následuje část získávání dat pro statistiky. Qdisc vezme parametry odesílání aktuálního paketu a přičte je do tabulky na pozici ukazatele *head*, pro zpracování řídící smyčkou DCC access, podobně jako jsou počítány statistiky v DCC app. Nakonec paket zařadí do odesílací fronty.

Pakety jsou odebírány z odesílacích front k odeslání dle priority fronty, od nejvyšší.

5.5.2 Řídící smyčka DCC access

Hlavním úkolem řídící smyčky komponenty DCC access, je úprava parametrů komponenty dle aktuálního stavu zatížení kanálu.

Interval smyčky je jedna sekunda, dle standardu viz zdroj [5]. V každém cyklu je zkontrolován stav zatížení získaný z lokální NDL tabulky a je porovnán s limitní hodnotou pro přechod do jiného stavu. Pokud jsou podmínky pro přechod splněny, změní hodnoty všech referenčních parametrů dle výchozích hodnot nového stavu. Pokud ne, pokračuje ve své činnosti bez změny.

Dalším úkolem řídící smyčky je počítání statistik. Data připravená od qdisc jsou, podobně jako v DCC app, uspořádána v poli. Pole znázorňuje sledovaný časový úsek 10 sekund a je rozděleno na menší měřené úseky. Jeden měřený úsek představuje, v případě statistik odesílání, jednu sekundu. Princip použití pole je znázorněn na obrázku 4.1

Řídící smyčka se také stará o obsluhu komunikačního rozhraní s DCC mgmt, tedy front FIFO. V každém cyklu vloží do fronty pro DCC mgmt zprávu se spočítanými statistikami a také zkontroluje, zda ve frontě od DCC mgmt není nová příchozí zpráva.

5.6 Implementace DCC mgmt

DCC mgmt je implementována jako modul jádra OS. Obsahuje globální tabulku DCC profilů a NDL tabulku. Skládá se z řídící smyčky a obsluhy příchozích zpráv Netlink protokolu. Po zavedení modulu do jádra OS, je inicializována globální tabulka DCC profilů a NDL tabulka. Také je spuštěna řídící smyčka a zaregistrována Netlink rodina *"DCC_PROTOCOL"*.

5.6.1 Řídící smyčka DCC mgmt

Řídící smyčka komponenty DCC mgmt kontroluje příchozí *FIFO* frontu od DCC access, v časových rozmezí jedna sekunda. Každý cyklus zkontroluje nové zprávy, které případně zpracuje. Poté do DCC access pomocí *FIFO* fronty odešle ukazatel na globální tabulku DCC profilů a také ukazatel na tabulku NDL. Tím se synchronizují lokální tabulky v DCC access s globálními tabulkami v DCC mgmt.

5.6.2 Obsluha Netlink protokolu

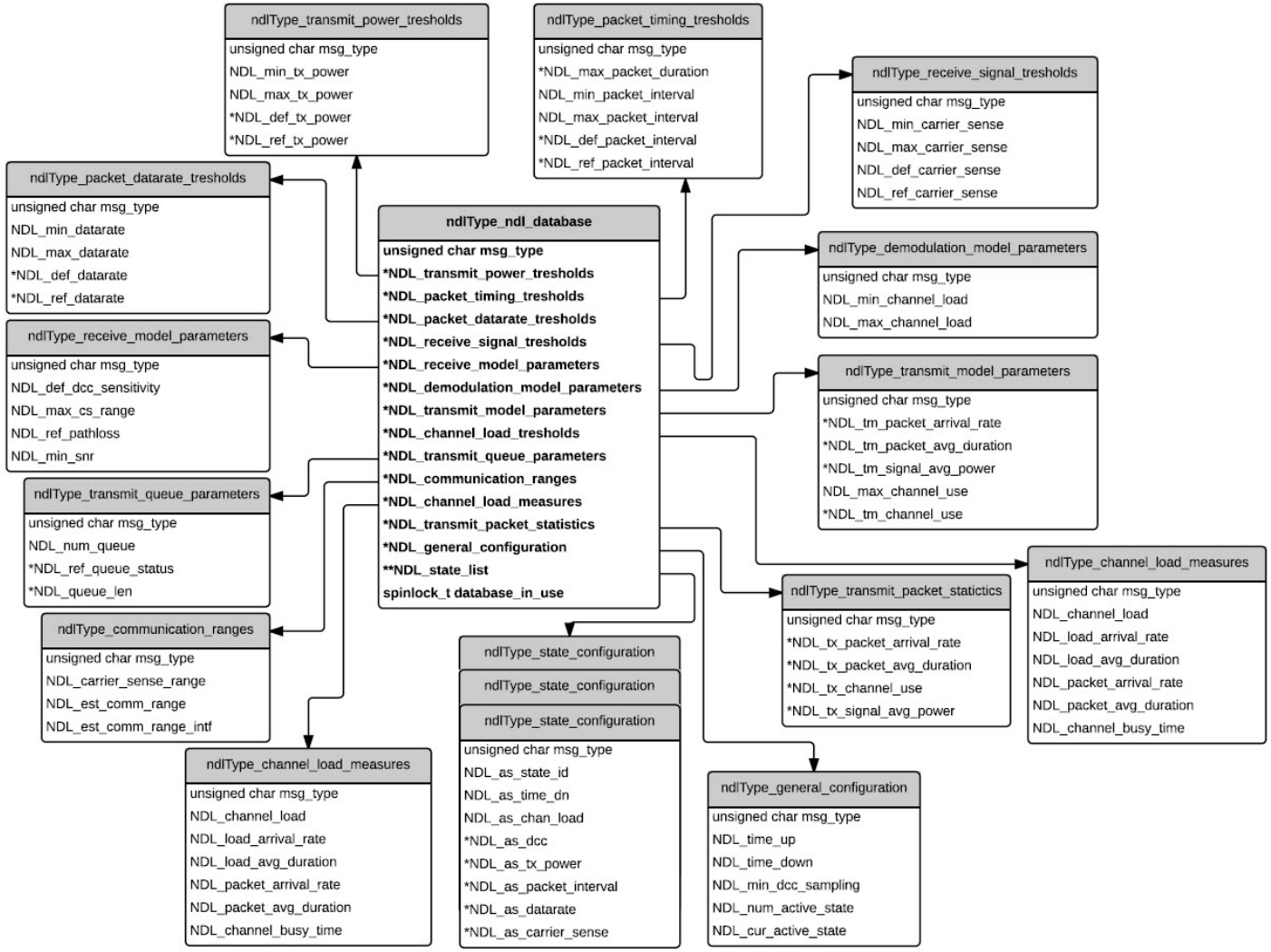
Po příchodu zprávy Netlink protokolu do obsluhy Netlink komunikace je přijatá zpráva rozpoznána a dle jejího typu zpracována. Zatím jsou implementovány tři druhy zpráv, sloužící pro výměnu informací s DCC app. Zprávy pro získání a odeslání parametrů NDL tabulky a zpráva pro odeslání naměřených statistik do DCC mgmt.

5.7 Nedokončená část implementace

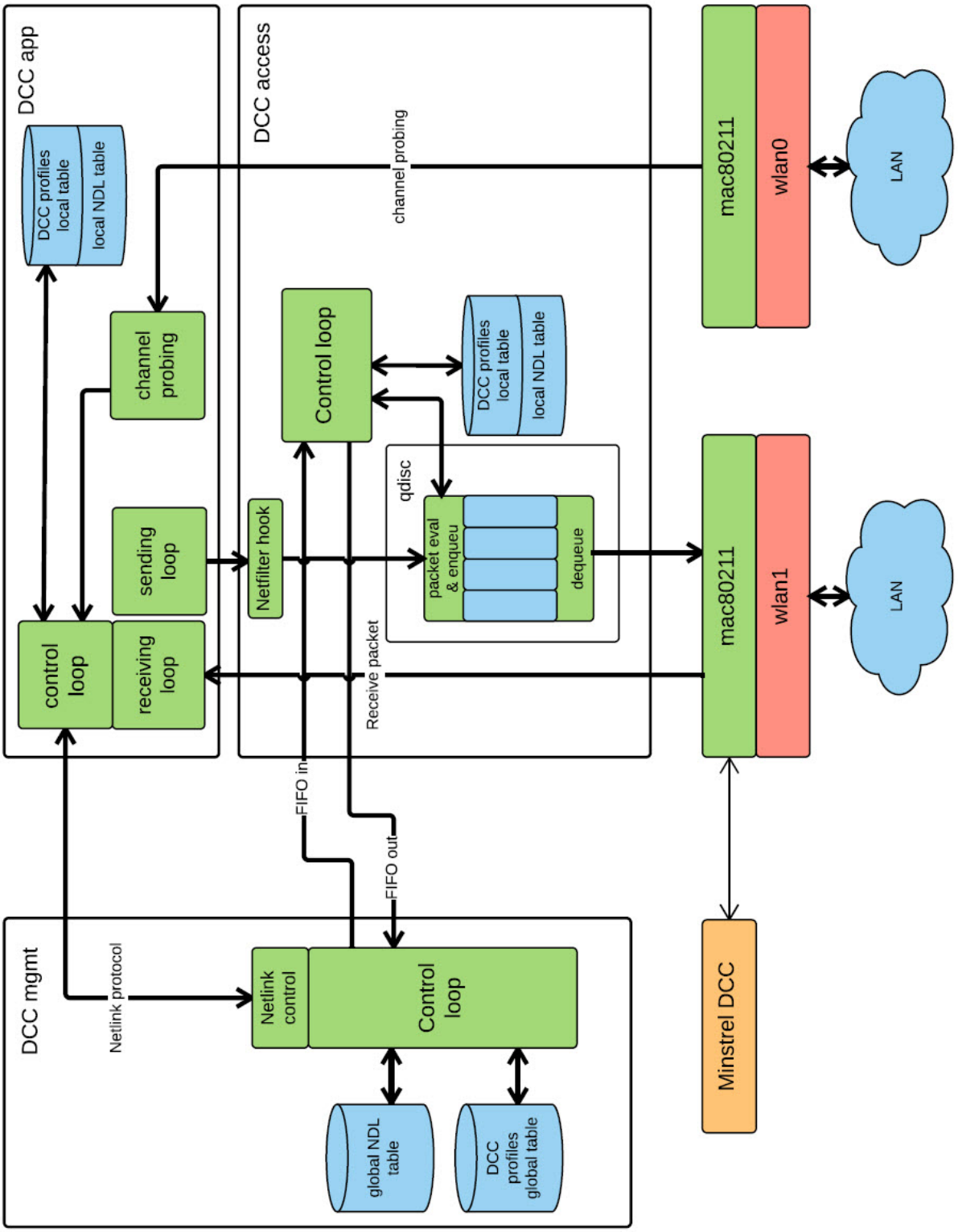
Jak jsem již zmínil v úvodu této diplomové práce, z důvodů dřívějšího ukončení studia jsem bohužel zcela nedokončil implementaci DCC systému. Část DCC systému, která má zaručit odeslání paketu zvolenou silou vysílacího signálu a přenosovou rychlostí není implementována. Nicméně v kapitole 4.3.3 jsem uvedl úvahy o úpravě Minstrel algoritmu, což by dle mého názoru vedlo ke správné funkci odesílání paketu.

5.8 Schéma implementovaného DCC systému

Na obrázku 5.7 je znázorněno schéma DCC systému implementovaného v OS Linux.



Obrázek 5.6: Struktura ndlType_ndl_database včetně podstruktur ndlType_transmit_power_tresholds, ndlType_packet_timing_tresholds, ndlType_packet_datarate_tresholds, ndlType_receive_signal_tresholds, ndlType_receive_model_parameters, ndlType_demodulation_model_parameters, ndlType_transmit_model_parameters, ndlType_channel_load_tresholds, ndlType_transmit_queue_parameters, ndlType_communication_ranges, ndlType_channel_load_measures, ndlType_transmit_packet_statictics, ndlType_general_configuration a ndlType_state_configuration



Obrázek 5.7: Schéma implementovaného DCC systému v OS Linux

Kapitola 6
Testování

Implementované části DCC systému byly otestovány pomocí simulace i v reálném prostředí. S ohledem na nedokončenou část implementace, byly vybrány testovací scénáře, které nejsou závislé na skutečné rychlosti odesílání paketu a síle odesílacího signálu.

6.1 Testování simulací

V první fázi testování implementace je použita simulace aktuálních podmínek komunikačního kanálu. Testuje se odezva řídící smyčky na aktuální zatížení a také manipulace s pakety dle jejich parametrů a stavu systému.

6.1.1 Testovací podmínky

Komponenta DCC app je upravena tak, aby přijímala generované hodnoty simulující provoz v komunikačním kanálu. Generovány jsou informace o přenosové rychlosti, síly přijímacího signálu a velikosti přenášených zpráv.

6.1.2 Testovací scénář #1

Do DCC app jsou náhodně generovány informace simulující komunikaci v komunikačním kanálu. Po přijetí těchto informací, jsou řídící smyčkou komponenty DCC app vypočítány statistiky a zatížení komunikačního kanálu a následně odeslány do globální NDL tabulky v DCC mgmt. Lokální tabulky v komponentě DCC access se každou sekundu synchronizují s globálními a tím komponenta DCC access získává informace o vypočítaném zatížení kanálu. Dle hodnoty tohoto zatížení vyhodnotí podmínky pro přechod do jiného stavu. V případě splnění podmínek pro přechod do nového stavu, nastaví nové referenční hodnoty dle hodnot platných pro nový stav z NDL tabulky.

6.1.3 Testovací scénář #2

V komponentě DCC app jsou v pravidelných intervalech vytvářeny pakety, které jsou plněny a následně posílány přes DCC access komponentu k odeslání. Každý paket obsahuje hlavičku s parametry odesílání a data. Po přijetí paketu komponentou DCC

access jsou vyhodnoceny parametry z hlavičky paketu a spuštěny DCC mechanismy. Pokud paket nesplňuje některé podmínky, je zahozen. Nakonec jsou zaznamenány informace o paketu pro účely výpočtu statistik a paket je odeslán do odesílací fronty. V řídící smyčce DCC access jsou každou sekundu počítány statistiky odchozích paketů. Synchronizací lokálních tabulek s globálními se vypočítané statistiky dostanou do DCC mgmt.

6.2 Testování na reálném HW

Ve druhé fázi testování implementace je použit skutečný hardware, WiFi karta řady Atheros 9000. Opět je testována odezva řídící smyčky na aktuální zatížení a také manipulace s pakety dle jejich parametrů.

6.2.1 Testovací podmínky

Komponenta DCC app odposlouchává pomocí PCAP API provoz v komunikačním kanálu. Pro každý paket je zaznamenána informace o přenosové rychlosti, síly přijímacího signálu a velikosti přenášených zpráv.

6.2.2 Testovací scénář #3

Řídící smyčka komponenty DCC app z odposlechnutých paketů počítá statistiky a zatížení komunikačního kanálu. Vypočítané hodnoty odesílá do globální NDL tabulky. Po synchronizaci lokálních tabulek v DCC access komponentě, se vypočítaná hodnota zatížení kanálu dostane do řídící smyčky DCC access. Pokud hodnota zatížení kanálu splňuje podmínky pro přechod do jiného stavu, nastaví referenční hodnoty do nových hodnot a přejde do nového stavu.

6.3 Výsledky testů

Testovací scénáře byly spouštěny nejprve každý samostatně a poté v následujících kombinacích:

- testovací scénář #1 a #2 současně
- testovací scénář #2 a #3 současně

Při použití testovacího scénáře #3, byla také měněna synchronizační doba mezi DCC mgmt a DCC access.

6.3.1 Výsledky testovacích scénářů

Testovací scénáře #1 a #3 byly určeny k otestování implementovaného DCC systému, konkrétně části řídící smyčky a její reakce na změnu zatížení kanálu. Řídící smyčka úspěšně měnila svůj stav dle simulovaného zatížení. V reálném prostředí bylo i s různými intervaly synchronizace docíleno podobných výsledků a řídící smyčka měnila stav DCC access s přijatelnými časovými odchylkami. DCC access komponenta

v testovacím scénáři #2 fungovala dle očekávání. Při velkém počtu paketů byly omezovány pakety s menší prioritou.

Při spuštění testovacích scénářů #1 a #2, či testovacích scénářů #2 a #3 současně, nebyla objevena výrazná změna v chování systému a výsledky byly podobné jako při spuštění jednotlivých testovacích scénářů samostatně.

Kapitola 7

Závěr

V moderních automobilech se pro zajištění bezpečnosti a plynulosti provozu bude využívat systém vzájemně komunikujících ITS stanic. Při komunikaci mezi ITS stanicemi může docházet, zejména v dopravní zácpě či na parkovišti, k zahlcení komunikačního kanálu a tím k výpadkům komunikace.

Cílem mé diplomové práce bylo implementovat DCC mechanismy zabraňující zahlcení komunikačního kanálu do OS Linux. V kapitole 7.1 jsou popsané části DCC systému, které jsem implementoval a úspěšně otestoval. V kapitole 7.2 jsem uvedl nedokončené části, které implementované nejsou.

Standard definující DCC systém, konkrétně komponentu DCC access, je v testovací fázi a v roce 2018 je očekávána jeho aktualizace.

7.1 Implementované části DCC systému

Komponenta DCC app se skládá ze dvou hlavních částí. Jedná se o části pro příjem a odesílání paketů a část měření a výpočtů příchozích statistik. DCC app odesílá pakety, naplněné specifikovanými parametry odesílání a daty, do komponenty DCC access, kde jsou dále zpracovány a odeslány do sítě. Měřící část komponenty odposlouchává komunikační kanál pomocí PCAP API a z parametrů odchycených paketů počítá příchozí statistiku a zatížení kanálu. Spočítané hodnoty odesílá do komponenty DCC mgmt.

DCC mgmt obsahuje tabulku DCC profilů a tabulku NDL. Přijaté informace od DCC app ukládá do NDL tabulky, kterou řídící smyčka pravidelně odesílá pomocí *FIFO* fronty do komponenty DCC access. Spolu s NDL tabulkou se odesílá i tabulka DCC profilů.

Hlavní komponenta DCC access přijímá pakety k odeslání z DCC app. Každý přijatý paket zkontroluje a aplikuje na něj DCC mechanismy. Následně zaznamená informace pro výpočet odchozích statistik a paket zařadí do odesílací fronty. Pakety z odesílacích front následně odebírá a předává k odeslání do ovladače síťové karty. Řídící smyčka v každém cyklu spočítá statistiky odesílání a synchronizuje lokální tabulky s globálními z DCC mgmt komponenty.

7.2 Neimplementované části DCC systému

Dle kapitoly 5.7, v implementaci chybí část DCC systému, která zaručí odeslání paketu zvolenou rychlostí a silou vysílacího signálu. Na základě úvah v kapitole 4.3.3 věřím, že úpravou Minstrel algoritmu by bylo možné nastavit přenosovou rychlost a sílu vysílacího signálu pro každý paket, jak je potřeba pro správnou funkci DCC systému.

7.3 Možné rozšíření práce

Implementace DCC systému pracuje pouze na jednom vybraném kanálu. Specifikace standardu předpokládá možné využití také ve vícekanálovém ITS zařízení. Úprava implementace je otázkou přidání dalších procesů a lokálních tabulek NDL a DCC profilů pro každý kanál.

Některé části implementovaného DCC systému mají jen základní funkcionalitu nezbytně nutnou pro chod systému. V komponentě DCC mgmt je možné rozšířit implementované komunikační rozhraní s ostatními komponentami.

Literatura

[1] *IEEE Std 802.11^{TM}-2012*: IEEE Standard for Information technology – Telecommunications and information exchange between systems Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications [online], (2012) [cit. 2016-05-01].
URL `http://standards.ieee.org/about/get/802/802.11.html`

[2] *ETSI EN 302 663*: ITS - Access layer specification for ITS operating in the 5 GHz frequency band [online], (draft V1.2.0-2012) [cit. 2016-05-01].
URL `http://www.etsi.org/standards-search`

[3] *ETSI EN 302 637-2*: ITS - Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service [online], (final draft V1.3.1-2014) [cit. 2016-05-01].
URL `http://www.etsi.org/standards-search`

[4] *ETSI EN 302 637-3*: ITS - Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service [online], (final draft V1.2.1-2014) [cit. 2016-05-01].
URL `http://www.etsi.org/standards-search`

[5] *ETSI TS 102 687*: ITS - Decentralized Congestion Control Mechanisms for ITS operating in the 5 GHz range; Access layer part [online], (V1.1.1-2011) [cit. 2016-05-01].
URL `http://www.etsi.org/standards-search`

[6] *ETSI TS 103 175*: ITS - Cross Layer DCC Management Entity for operation in the ITS G5A and ITS G5B medium [online], (V1.1.1-2015) [cit. 2016-05-01].
URL `http://www.etsi.org/standards-search`

[7] *ETSI EN 302 665*: ITS - Communications Architecture [online], (V1.1.1-2010) [cit. 2016-05-01].
URL `http://www.etsi.org/standards-search`

[8] *ETSI TS 102 723-1*: ITS - OSI cross-layer topics; Part 1: Architecture and addressing schemes [online], (V1.1.1-2012) [cit. 2016-05-01].
URL `http://www.etsi.org/standards-search`

[9] *ETSI TS 102 723-3*: ITS - OSI cross-layer topics; Part 3: Interface between management entity and access layer [online], (V1.1.1-2012) [cit. 2016-05-01]. URL `http://www.etsi.org/standards-search`

[10] *ETSI TS 102 723-10*: ITS - OSI cross-layer topics; Part 10: Interface between access layer and networking & transport layer [online], (V1.1.1-2012) [cit. 2016-05-01]. URL `http://www.etsi.org/standards-search`

[11] *Linux Wireless Wiki* [online], (2015) [cit. 2016-05-01]. URL `https://wireless.wiki.kernel.org/`

[12] *ETSI TS 102 724*: ITS - Harmonized Channel Specifications for ITS operating in the 5 GHz frequency band & transport layer [online], (V1.1.1-2012) [cit. 2016-05-01]. URL `http://www.etsi.org/standards-search`

[13] WEHRLE, Klaus. *The Linux networking architecture*: design and implementation of network protocols in the Linux kernel. Upper Saddle River, N.J.: Pearson Prentice Hall, (2005). ISBN 0131777203.

[14] MAUERER, Wolfgang. *Professional Linux kernel architecture.* (2010). ISBN 0470343435.

Příloha A

Použité pojmy a zkratky

Symbol	Význam
AC	Access Category
AP	Access Point
DCC	Decentralized Congestion Control
DSC	DCC Sensitivity Control
DSRC	Dedicated Short-Range Communications
GNU C	kompiler jazyka C z projektu GNU
ID	identifikační číslo
IEEE	Institute of Electrical and Electronics Engineers
ITS	Intelligent Transport System
LAN	Local Area Network
MAC	Media Access Control
NAPT	Network Address and Port Translation
NDL	Network Design Limits
NIC	Network Interface Controller
OBU	On-Board Unit
OCB	Outside the Context of a BSS
OS	Operační Systém
Qdisc	Queueuing discipline
QoS	Quality of Service
RM	Receive Model
RSU	Road-side Unit
SNR	Signal to Noise Ratio
TAC	Transmit Access Control
TDC	Transmint Datarate Control
TM	Transmit Model
TPC	Transmit Power Control
TRC	Transmit Rate Control
WAVE	Wireless Access in Vehicular Environment
WLAN	Wireless Local Area Network

Příloha B

Zdrojové kódy a spuštění aplikace

Aktuální zdrojový kód je k dispozici v git repozitáři:

```
$ git clone git@gitlab.fel.cvut.cz:vancuvit/dcc_access.git
```

Aplikace byla vyvíjena v IDE CodeBlocks v OS Debian 4.5.0-rc2.
Struktura adresáře se zdrojovými kódy:

```
/
└── src
    ├── Makefile
    ├── dcc_access.c
    ├── dcc_access.cbp
    ├── dcc_access.h
    ├── dcc_app.c
    ├── dcc_app.h
    ├── dcc_cross.c
    ├── dcc_cross.h
    ├── dcc_err.h
    ├── dcc_mgmt.c
    ├── dcc_mgmt.h
    ├── dcc_queue.c
    ├── dcc_queue.h
    ├── platform.h
    ├── radiotap.c
    ├── radiotap.h
    └── radiotap_iter.h
└── README
```

Soubory platform.h, radiotap.c, radiotap.h a radiotap_iter.h jsou z knihovny RADIOTAP

Ke správné kompilaci jsou potřebné následující balíčky (knihovny libnl3, lnl-genl-3, lnl-3, lpthread, lpcap):

```
$ sudo apt-get install libnl-utils
$ sudo apt-get install libnl-3-dev
$ sudo apt-get install libnl-3-200
$ sudo apt-get install libnl-genl-3-dev
$ sudo apt-get install libnl-genl-3-200
```

```
$ sudo apt-get install build-essentials
$ sudo apt-get install libpthread-stubs0-dev
$ sudo apt-get install libpcap-dev
```

Kompilace je provedena příkazem make:

```
$ make
```

Instalace a spuštění pro stanici s dvěma WiFi kartami, měření je prováděno wlan0, wlan1 slouží pro komunikaci.

```
$ make kernel-module-install-wifi
$ dmesg        # verify if modules are loaded
$ sudo dcc_run
```

Instalace a spuštění pro stanici s jednou WiFi kartou a jedním Eth portem, měření je prováděno wlan0, eth0 slouží pro komunikaci.

```
$ make kernel-module-install-eth
$ dmesg        # verify if modules are loaded
$ sudo dcc_run
```

V případě problémů se spuštěním aplikace, zkuste resetovat síťové rozhraní.

```
$ sudo ifdown eth0
$ sudo ifdown wlan0
$ sudo ifup eth0
$ sudo ifup wlan0
```