VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Luka Zarković

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

AUTO RATE FALLBACK (ARF) ALGORITMUS PRO BEZDRÁTOVÉ SPOJENÍ STANDARDU 802.11G.

AUTO RATE FALLBACK (ARF) ALGORITHM FOR WIRELESS CONNECTION ACCORDING TO THE STANDARD 802.11G.

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Luka Zarković

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Anna Kubánková, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Luka Zarković ***ID:*** 186241
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Auto Rate Fallback (ARF) algoritmus pro bezdrátové spojení standardu 802.11g.

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte standard 802.11g a Auto Rate Fallback (ARF) algoritmus. Seznamte se s prostředím NS-3 (Network Simulator 3). Vytvořte model bezdrátové sítě standardu 802.11g, kde bude jeden přístupový bod a několik pohybujících se stanic. Popište postup při návrhu modelu. Analyzujte ARF algoritmus a vliv různých provozních parametrů na funkčnost tohoto algoritmu. Porovnejte další Rate Adaption algoritmy mezi sebou.

DOPORUČENÁ LITERATURA:

[1] ns-3. [online]. [cit. 2016-09-13]. Dostupné z: https://www.nsnam.org/

[2] IEEE standard 802.11g, “Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band,” 2003.

Termín zadání: 18.6.2018 ***Termín odevzdání:*** 15.8.2018

Vedoucí práce: Ing. Anna Kubánková, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Táto bakalářská práce se zabývá problematikou adaptačních protokolů a jejich implementaci. V této bakalářské práci je proveden popis fyzické a MAC vrstvy standardu 802.11g a teoretický rozbor adaptačních protokolů ARF a AARF. Dále práce popisuje postup tvorby modelu sítě vhodné pro testování a ověření vlastnosti adaptačních protokolů v simulačním prostředí NS-3 (Network Simulator 3). V poslední části jsou popsány výsledky simulace a zpracovány do vhodných grafů.

KLÍČOVÁ SLOVA

ARF, IEEE 802.11, NS-3, 802.11g

ABSTRACT

This bachelor thesis deals with problems of adaptation protocols and their implementation. In this bachelor thesis are described physical and MAC layers of standard 802.11g and theoretical analysis of adaptation protocols ARF and AARF. Further, the thesis describes the process of creating a network model suitable for testing and verifying the properties of adaptive protocols in the simulation environment NS-3 (Network Simulator 3). In the last part, the simulation results are described and processed into suitable graphs.

KEYWORDS

ARF, IEEE 802.11, NS-3, 802.11g

ZARKOVIĆ, Luka. *Auto Rate Fallback (ARF) algoritmus pro bezdrátové spojení standardu 802.11g*. Brno, 2017, 53 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Anna Kubánková, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Auto Rate Fallback (ARF) algoritmus pro bezdrátové spojení standardu 802.11g“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucí bakalářské práce paní Ing. Anně Kubánkové, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

IEEE 802.11 je standard pro Wi-Fi s dalšími doplňky pro lokální bezdrátové sítě (Wireless LAN, WLAN) vyvíjený 11. pracovní skupinou IEEE LAN/MAN standardizační komise (IEEE 802). Výraz 802.11x je používán pro množinu doplňků k tomuto standardu. IEEE 802.11 je nejpoužívanějším WLAN systémem v dnešním světě a pravděpodobně bude hrát důležitou roli v příští generaci bezdrátové a mobilní komunikace. Původně IEEE 802.11 nabízel pouze dvě fyzické datové rychlosti: přenos se prováděl na 1Mbps nebo 2Mbps. V roce 1999 definovala IEEE dvě rozšíření: 802.11b založené na technologii DSSS, se datovými rychlostmi až 11Mbps v pásmu 2,4GHz a 802.11a na základě OFDM (Orthogonal Frequency Division Multiplexing) technologie, s přenosovou rychlostí až 54 Mb / s v pásmu 5 GHz. V roce 2003, standard 802.11g, který rozšiřuje vrstvu PHY 802.11b na podporu datových rychlostí až 54 Mb / s v pásmu 2,4 GHz byl dokončen.

Existuje mnoho důvodů pro velmi nestálou povahu bezdrátového média používaného standardem IEEE 802.11: útlum, rušení z jiných zdrojů záření, rušení z jiných zařízení 802.11 v síti ad hoc, atd. Tyto změny v kvalitě přenosu lze klasifikovat jako přechodné krátkodobé modifikace bezdrátového připojení v přenosovém prostředí.

Algoritmy, které upravují parametry přenosu na kanálové podmínky mohou být navrženy tak, aby optimalizovaly množství parametrů v závislosti na topologii sítě a typu zařízení. Jeden z takových algoritmů je ARF(Auto-Rate Fallback).

Táto bakalářska práce je nástroj na lepší teoretické pochopení standardů IEEE 802.11 a adaptačních algoritmů. Práce také obsahuje praktickou část, ve které je vysvětleno programové řešení tj. samotný kód, který byl napsán v jazyce C++. Součastí je dále obrázek simulace v programu NetAnim a zkoumání některých parametrů ve Wireshark-u. Důležité výsledky byly zpracovány do vhodných grafů.

1 IEEE 802.11 STANDARD

Lokální bezdrátové sítě dnes implementují IEEE 802.11 standard. Standard bezdrátové sítě IEEE 802.11 byl vyvinutý za účelem poskytnutí bezdrátových lokálních sítí přes nelicencované pásmo ISM.

802.11 definuje protokol ve dvou vrstvách:

- Fyzická vrstva(PHY)
- Spojová vrstva(MAC)

Část (1.1) uvádí důležité informace fyzické vrstvy 802.11, zatímco část (1.2) představí spojovou vrstvu.

1.1 Fyzická vrstva

Fyzická vrstva standardů 802.11 je s první vrstvou ISO/OSI modelu provázaná stejným názvem.

802.11 používá několik modulačních a kódovacích schémat na fyzické vrstvě, a to DSSS, FHSS, OFDM a HR-DSSS.V části (1.1.1) budeme popisovat různé modulační techniky které se používají. Část (1.1.2) uvádí hlavní změny pro IEEE 802.11 včetně IEEE 802.11g. V části (1.1.3) uvádíme různé mechaniky preambule 802.11 (dlouhé a krátké).

1.1.1 Modulační techniky

Standard IEEE 802.11 implementuje tři hlavní modulační techniky, Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS) a Infrared (IR). Následuje popis prvních dvou.

Direct Sequence Spread Spectrum (DSSS)

Technika přímého rozprostřeného spektra je jednou z metod pro rozšíření spektra při bezdrátovém přenosu dat. Pracuje tak, že každý jednotlivý bit určený k přenosu, je nejprve nahrazen určitou početnější sekvencí bitů (tzv. chipů). Tyto sekvence mají nejčastěji pseudonáhodný charakter. Pro jejich vytváření se využívají například Goldovy či Barkerovy kódy. Skutečně přenášena (modulována na nosný signál) je pak tato sekvence bitů. Jde tedy vlastně o umělé zavedení nadbytečnosti (redundance). Signál, který je rozprostřen do větší části rádiového spektra, je méně citlivý vůči rušení (což zvyšuje spolehlivost přenosu). Signál se ostatním uživatelům jeví jako náhodný šum, a bez znalosti mechanismu vytváření původní pseudonáhodné sekvence je pro ně obtížné zpět získat (demodulovat) přenášená data. Jedná

se o modulační techniku používanou například v bezdrátové technologii Wi-fi či v navigačním systému GPS.[9]

Hopping Spread Spectrum (FHSS)

FHSS je další modulační technika používaná v bezdrátových sítích 802.11. Tato technika používá pojem přeskakování z jednoho kanálu na druhý v rovnoměrně rozložených frekvencích a časových úsecích. Kdykoliv je vysílání obsazeno určitou frekvencí po stanovenou dobu, komunikace se přesune na jinou frekvenci. Počítač vysílá všesměrové zprávy(broadcast), kterými oznamuje, které kanály používá pro vysílání nosných. Tento proces probíhá v pseudonáhodném režimu. Stanice, která se chce připojit k počítači obvykle poslouchá pro tyto vysílané zprávy, a když zpráva přijde, může začít skákání mezi kanály, aby se data dostala k počítači. Výhodou využití spektra šíření frekvencí (FHSS) je především schopnost obejít šum na určitých frekvencích. Pokud schéma FHSS přeskakuje na kanál, který je silně ovlivněný šumem, ovlivní pouze komunikaci mezi stanicemi během času stráveného v aktuální frekvenci. Když PC přechází na jiný kanál, šum z předchozího kanálu bude s největší pravděpodobností zanedbatelný a provoz se může obnovit a využívat vyšší přenosové rychlosti. Nevýhodou této techniky je potřeba přesného načasování na všech stanicích. Abychom byli schopni sledovat veškerou komunikaci v kanálu, každá stanice se musí přepnout na novou frekvenci ve stejném čase. Pokud ne, mohlo by dojít ke ztrátě a opětovnému přenosu dat, které by mohly snížit propustnost kanálu.[9]

1.1.2 Rozšířené modulační techniky

Krátkou dobu po objevení standardu 802.11, řídící rada IEEE 802.11 schválila dvě rozšíření už existujícího standardu 802.11. To jsou standardy 802.11a a 802.11b, které poskytují mechanismy pro zvýšení přenosové rychlosti v bezdrátovém mediu. Samozřejmě, existuje ještě řada dalších standardů, viz tabulku(1.1). Některé standardy budou popsány podrobněji.

Tab. 1.1: Standardy 802.11

Standard	**Rok vydání**	**Pásmo[GHz]**	**Max. rychlost [Mbit/s]**	**Modulace**
IEEE 802.11	1997	2,4	2	DSSS a FHSS
IEEE 802.11a	1999	5	54	OFDM
IEEE 802.11b	1999	2,4	11	DSSS
IEEE 802.11g	2003	2,4	54	OFDM
IEEE 802.11n	2009	2,4 nebo 5	600	MIMO OFDM
IEEE 802.11y	2008	3,7	54	
IEEE 802.11ac	2013	2,4 a 5	1000	MU-MIMO OFDM
IEEE 802.11ad	2012	2,4 , 5 a 60	7000	

802.11a

Modifikace 802.11a pro bezdrátovou síť LAN podle původního standardu 802.11 byla schválená v roce 1999. Standard 802.11a je založen na stejném principu jako původní standard, pracuje v pásmu 5 GHz a využívá multiplexování ortogonálních kmitočtových dělení (OFDM) s maximální přenosovou rychlostí 54 Mbit/s, která dosahuje realistické propustnosti 20 Mbit/s. Rychlost dat je snížená na 48, 36, 24, 18, 12, 9 a 6 Mbit/s v případě potřeby. 802.11a původně měla 12 až 13 nepřekrývajících se kanálů, z toho 12 kanálů lze použít uvnitř a 4 až 5 z 12 kanálů lze použít při konfiguraci venkovních bodů. 802.11a není interoperabilní s 802.11b, protože pracuje na samostatných pásmech, s výjimkou zařízení, která mají kapacitu dvou pásem.[4]

802.11b

Tento standard má maximální datovou rychlost 11 Mbit/s a používá stejnou metodu přístupu na medium CSMA/CA definovanou v původním standardu. Vzhledem k režii protokolu CSMA/CA je v praxi maximální přenosová rychlost přibližně 5,9 Mbit/s pomocí TCP a 7,1 Mbit/s pomocí UDP.

Produkty 802.11b se objevily na trhu v polovině roku 1999, protože 802.11b je přímým rozšířením modulační techniky DSSS(Direct sequence spread spectrum) definované v původním standardu. Apple iBook byl první počítač prodávaný se standardem 802.11b. Z technického hlediska standard 802.11b používá jako svou modulační techniku CCK(complementary code keying). Výrazný nárůst propustnosti 802.11b(ve srovnání s původním standardem) spolu se současným podstatným snížením cen vedly k rychlému přijetí technologie 802.11b.

Zařízení s technologií 802.11b trpí interferencí jiných přístrojů pracujících v pásmu 2,4 GHz. Zařízení pracující v rozmezí 2,4 GHz zahrnují: mikrovlnné trouby, bluetooth, bezdrátové telefony atd. Problémy s rušením a problémy s hustotou uživatelů v pásmu 2,4 GHz se staly hlavním problémem a frustrací pro uživatele.[5]

802.11g

IEEE 802.11g je doplněním specifikace IEEE 802.11, která rozšířila propustnost až na 54 Mbit/s pomocí stejného pásma 2,4 GHz jako u 802.11b. 802.11g je třetí modulační standard pro bezdrátové sítě LAN. Pracuje v pásmu 2,4GHz, ale s maximální rychlostí 54 Mbit/s. Použitím přenosového schématu CSMA/CA je maximální přenosová rychlost 31,4 Mbit/s. V praxi nemusí mít přístupové body ideální implementaci, proto nemohou dosáhnout dokonce propustnosti 31,4 Mbit/s s 1500 bitovými pakety.

Hardware 802.11g je plně zpětně kompatibilní s hardwarem 802.11b. V síti 802.11g však přítomnost účastníka 802.11b významně sníží rychlost celkové sítě 802.11g. Některé směrovače 802.11g využívají zpětný kompatibilní režim pro klienty 802.11b nazývané 54g LRS (Limited Rate Support).

Modulační schéma použitá v 802.11g je širokopásmová modulace využívající frekvenční dělení kanálu (OFDM) zkopírováno z 802.11a s datovými rychlostmi 6, 9, 12, 18, 24, 36, 48 a 54 Mbit/s. I když 802.11g pracuje ve stejném kmitočtovém pásmu jako 802.11b, může dosáhnout vyšších přenosových rychlostí kvůli jeho dědictví na 802.11a.[3]

1.1.3 Podvrstvy fyzické vrstvy

Ve všech standardech 802.11 je fyzická vrstva rozdělená do dvou podvrstev:

- **PLCP (Physical Layer Convergence Procedure)**– v této podvrstvě se k datovým rámcům MAC (Medium Access Control) podvrstvy přikládají informace o použitém přenosovém mechanismu a modulaci. Díky této podvrstvě je přenášený datový rámec nezávislý na typu fyzické vrstvy. Do této podvrstvy je implementována rovněž funkce CCA (Clear Channel Assessment), která poskytuje odezvu pro MAC vrstvu o připravenost přenosového média.
- **PMD (Physical Medium Dependent)**– tato podvrstva je zodpovědná za přenos dat mezi jednotlivými vysílači a přijímači. Z podvrstvy PLCP jsou data v závislosti na použitém přenosovém mechanismu ve vysílači vysílána do bezdrátového prostředí, kde jsou na straně přijímače pomocí PMD přijímána a předávána podvrstvě PLCP.

V následujícím textu bude postupně popsána struktura PLCP podvrstvy dvou dnes nejpoužívanějších standardu 802.11a/g a její návaznost na vyšší vrstvy (MAC, LLC atd.).

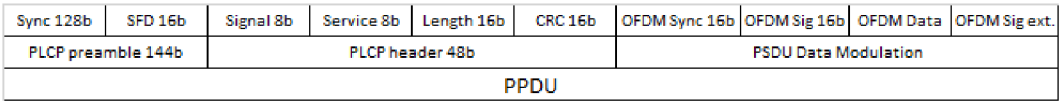
PPDU

PPDU rámec (1.1) dle standardu IEEE 802.11a/g se skládá z následujících polí:

- **PLCP preambule**– tvoří dvanáct OFDM symbolů, kdy je prvních 10 krátkých (0,8 us) a další dva jsou dlouhé (4 us). Oba uvedené časy odpovídají přenosu OFDM symbolu v kanálu širokém 20 MHz.
- **PLCP hlavička**- je rozdělena na dvě části. První část, která má 24 bitů, je umístěna v bloku SIGNAL. Druhá je umístěna v datové části bloku SERVICE a její délka je 16 bitů. Údaje, důležité pro správné přečtení datové části, jsou

uloženy v blocích RATE a LENGHT. Blok RATE specifikuje přenosovou rychlost datového bloku (DATA) a LENGHT jeho délku. Zbylé bloky jsou určeny k řízení a synchronizaci komunikujících zařízení.

- **DATA**– v tomto bloku jsou přenášena nejen uživatelská data z vyšších vrstev (PSDU), ale i servisní informace. Celá datová část může být přenášena na rozdíl od předchozích částí (PLCP preambule a hlavička) vyššími rychlostmi.



Obr. 1.1: PPDU rámec

1.2 Spojová vrstva (MAC)

Vrstva řízení přístupu k médiu (MAC) používaná skupinou standardů 802.11 je základem přenosového protokolu. Základním přístupovým mechanismem, neboli distribuční koordinační funkcí je CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance).[1]

Tab. 1.2: 802.11 rámec

Preambule	PLCP Header	MAC Data	CRC

U klasického Ethernetu, např. na koaxu, může každá stanice slyšet vysílání jiné stanice a detekovat kolizi. Tento základní předpoklad pro detekování kolizí u bezdrátových sítí neplatí. Stanice může detekovat volné médium ve svém okolí, to však neznamená, že je volné i u přijímače. Jak je uvedeno výše, stanice komunikují prostřednictvím AP a nemusí se tak vůbec přímo slyšet s jinou stanicí, ani detekovat její vysílání. Proto je použit mechanismus předcházení kolizím spolu s kladným potvrzováním. Lokalní bezdrátové sítě se často nazývají pojmem bezdrátový ethernet, ale to neznamená, že tyto technologie mají stejnou strukturu rámce. Maximalní délka WLAN rámce je 2346B oproti 1518B u klasického ethernetu. WLAN rámec předchází preamble vytvořený střídavým sledem 0 a 1. Preamble zajišťuje bitovou synchronizaci. Jsou dva typy preamble rámce lišící se v délce. Tzv. dlouhý preamble se skládá ze 128 bitů a povinně musí být podporován u technologie 802.11. Krátký preamble má pouze 56 bitů, je volitelný u technologie 802.11 a využívá se např. pro účely přenosu hlasu přes WLAN, kde je snaha minimalizovat zpoždění. Vzájemně komunikující zařízení musí používat stejný typ preamble rámce.[16]

1.2.1 Typy rámců

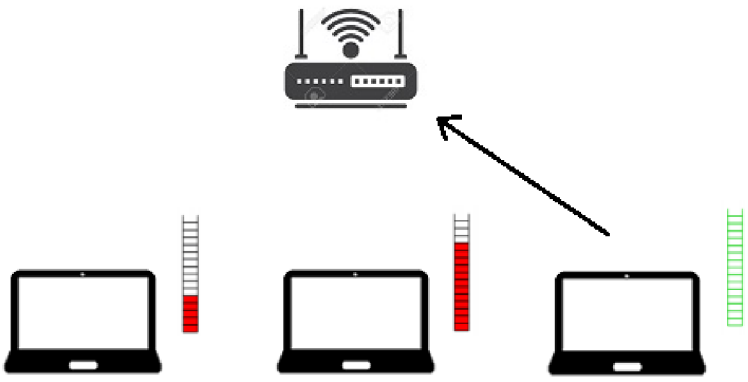
Aktuální normy 802.11 specifikují typy rámců pro použití v přenosu dat, stejně jako pro správu a řízení bezdrátových spojení. Používají se 3 základní typy rámců:

- **Datové**: přenášejí samotná data
- **Řídící**: slouží pro řízení přenosu dat (RTS, CTS, ACK, PS Poll, CF End, CF End + ACK)
- **Managmentové**: slouží pro synchronizaci a připojení stanic do buňky (Association request, Association response, Reassociation request, Reassociation response, Beacon, Disassiociation, Authentication, Deauthentication)

1.2.2 Koordinační funkce

Pro sítě WLAN jsou definovány dva typy koordinačních funkcí, distribuované a centralizované.

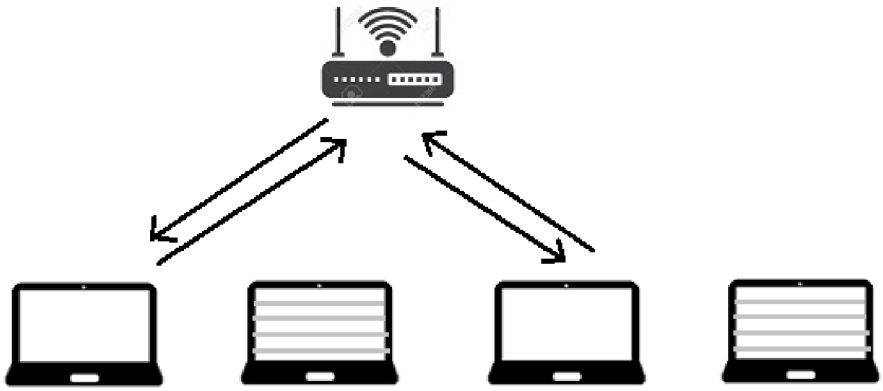
- Distribuovaná koordinační funkce (DCF - Distributed Coordination Function) je definována v kapitole 9.3 standardu IEEE 802.11 a je de facto výchozí nastavení pro hardware Wi-Fi. Využívá náhodnou přístupovou metodu, což znamená, že stanice soutěží o přístup k médiu, viz obrazek (1.2).



Obr. 1.2: Distribuovaná koordinační funkce (DCF)

- Centralizovaná koordinační funkce (PCF - Point Coordiantion Fuction) je přístupová metoda bez soutěžení. Tato metoda vyžaduje přístupový bod, který se pravidelně dotazuje všech stanic, zda nemají data k vysílání, viz obrazek

(1.3). Takové řešení je vhodné pro zajištění kvality služeb. Tato metoda se používá vždy v kombinaci s DCF. PCF se nachází přímo nad distribuovanou koordinační funkcí (DCF) v architektuře MAC IEEE 802.11.



Obr. 1.3: Centralizovaná koordinační funkce (PCF)

V komunikaci přes síť WLAN hrají důležitou roli čekací doby označené pojmem mezirámcová mezera. Existují tři typy mezirámcových mezer:

- Krátká mezirámcová mezera (Short Interframe Space – SIFS) má nejvyšší prioritu. V případě, že médium není volné, stanice musí čekat na jeho uvolnění. Podle toho, jakou operaci chce stanice provést, tak je stanoveno, jak dlouho se po uvolnění média může pokoušet o přístup k médiu. Protože SIFS je nejkratší dobou (Obr. 6.28), zajišťuje největší pravděpodobnost přístupu k médiu. Před nebo za SIFS mohou být rámce RTS (určený k rezervaci média), CTS (odezva na RTS, která zajišťuje, že všechny stanice budou čekat na avizovaný přenos) a rámce ACK (potvrzení přijatého rámce).
- Mezirámcová mezera centralizované koordinační funkce (Point Coordination Function Interframe Space – PIFS) je středně dlouhá (Obr. 6.28) a má vyšší prioritu, než datové rámce. Využívá se pouze v případě nastavené PCF, kde je nutné pravidelné dotazování stanic.
- Mezirámcová mezera distribuované koordinační funkce (Distributed Coordination Function Interframe Space – DIFS) je základem DCF. Každá stanice využívající DCF musí čekat minimálně DIFS a potom může začít soutěžit o médium. Po uplynutí DIFS musí stanice čekat ještě náhodně dlouhou dobu,

což slouží jako ochrana proti kolizi. Po DIFS následuje soutěžící interval (Contention Period – CP). Po uplynutí DIFS si stanice vygeneruje náhodné číslo, které vynásobí délkou kanálového intervalu (slot time), aby zjistila, jak dlouho má čekat. Každá stanice postupně odpočítává kanálové intervaly a první stanice, která odpočítá celý čekací interval, může obsadit médium. Ostatní stanice zjistí, že médium už není volné a zapamatují si zbylý počet kanálových intervalů. V následujícím soutěžení bude stanice pokračovat v odpočítávání od zapamatované hodnoty.[16]

1.2.3 Skrytý uzel(zavedení RTS/CTS)

Kromě dvou přístupových metod DCF a PCF existuje i třetí metoda, která umožňuje stanici získat nerušený přístup ke sdílenému médiu. Tuto přístupovou metodu inicializuje přímo stanice, která vyžaduje výhradní přístup a trvá až do ukončení přenosu, tj. až do přenosu rámce ACK. Tato přístupová metoda je zajištěna pomocí rámců Request to Send (RTS) a Clear to Send (CTS).

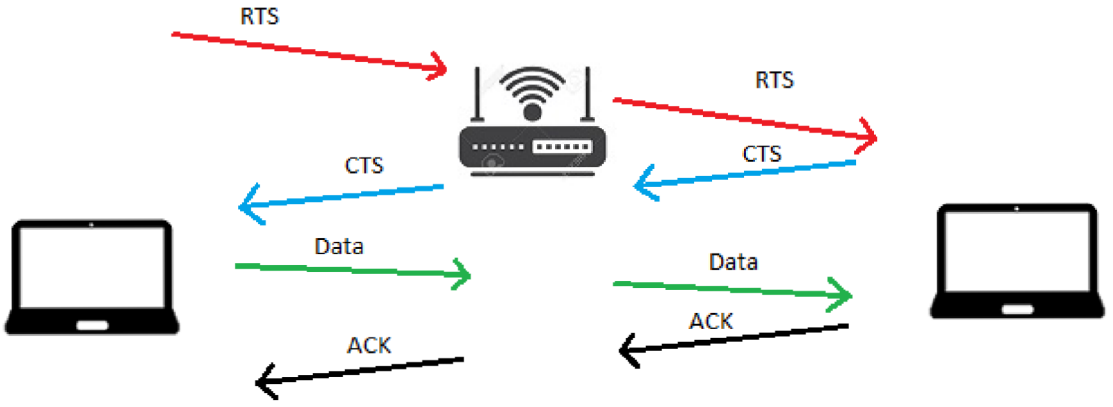
Při používání WiFi ve venkovním prostředí je "skrytý uzel" velkým problémem. Skrytý uzel se prakticky vždy objevuje tam, kde se WiFi používá ve venkovním prostředí pro distribuci signálu způsobem Point – Multipoint. Příčinou je fakt, že zatímco přípojný bód vidí všechny stanice, protože používá všesměrovou nebo alespoň sektorovou anténu, klientské stanice (tedy uzly) se navzájem nevidí, protože je s přípojným bódem pojí směrová anténa.

Jak funguje RTS/CTS mechanismus bránící kolizím ve WiFi sítích?

- stanice, která chce vysílat si ověří, zda-li je síť po určitou dobu (DIFS - Distributed Inter Frame Space) volná
- jestliže síť je (stane se) v průběhu DIFS obsazená, tak se přenos dat odloží
- v opačném případě je vyslán krátký paket RTS - Request To Send, který mimo jiné obsahuje informaci o době, kterou bude následující přenos trvat
- cílová stanice odpovídá (po krátkém okamžiku - SIFS) packetem CTS - Clear To Send, který opět mimo jiné obsahuje dobu, po kterou bude následující přenos trvat
- všechny stanice, které slyší RTS nebo CTS si nastaví vlastní indikátor NAV - Network Allocation Vector na dobu přenášenou v těchto paketech a nebudou se v jejím průběhu snažit přistupovat k síti
- CTS a RTS jsou krátké pakety a výše uvedený mechanismus dovoluje podstatným způsobem snížit pravděpodobnost kolize
- celá transakce je (v případě úspěšného přenosu dat) ukončena zasláním paketu ACK - Acknowledge

- jestliže přenos není potvrzen paketem ACK, pak je situace vyhodnocena jako kolize a přenos se opakuje

Tento proces je zobrazen na obrázku (1.4).



Obr. 1.4: RTS/CTS

1.3 Datové (přenosové) rychlosti v 802.11g

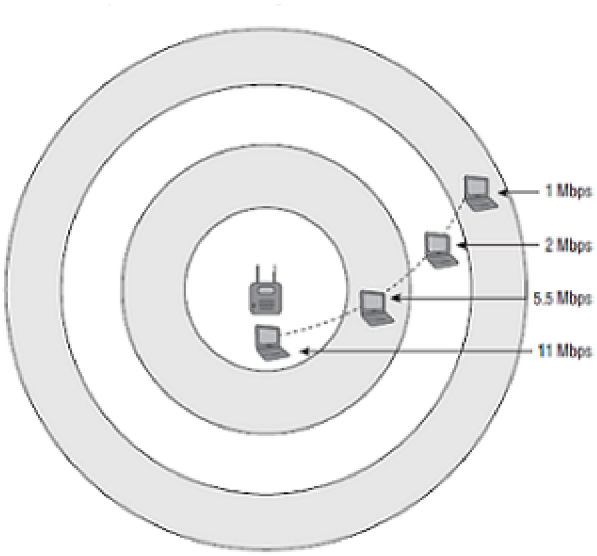
Norma IEEE 802.11 definuje sadu datových rychlostí, které se mohou používat při přenosu dat přes bezdrátový kanál. Dostupné datové rychlosti závisí na verzi bezdrátového rozhraní IEEE 802.11 sítě. Tabulka (1.3) uvádí rychlosti dostupné pro bezdrátové sítě implementující IEEE 802.11a/b/g.

Tab. 1.3: Rychlosti 802.11

Datová rychlost	802.11	802.11a	802.11b	802.11g
1	x			
2	x			
5,5			x	
6		x		x
9		x		x
11			x	
12		x		x
18		x		x
24		x		x
36		x		x
48		x		x
54		x		x

1.3.1 Výběr rychlosti

Volba správné rychlosti(rate selection) viz obrázek (1.5) při přenosu rámců je důležitá. Pokud zařízení zvolí rychlost, která je nevyhovující, propustnost kanálu je vážně ovlivněna. Další kapitola se zaměří na přizpůsobení rychlosti a způsobu jeho implementace a použití v IEEE 802.11 bezdrátových sítích.[13]



Obr. 1.5: Ilustrace výběru rychlosti

2 ADAPTAČNÍ ALGORITMY

Adaptace rychlosti v IEEE 802.11 je dobře známý a hluboce studovaný problém. Z algoritmů, které byly navrženy v literatuře část nemůže být realizována v reálném síťovém rozhraní, protože nejsou kompatibilní. Kanálové podmínky ve WLAN sítích jsou proměnné kvůli pohybu předmětů v prostoru a interferencí se dalšími zařízeními v prostoru. Autorate algoritmy jsou běžně používany ve WLAN sítích pro maximalizaci datové rychlosti přes proměnný fyzický kanál. Obecně platí, že podle metod pro odhadování existujících kanálových podmínek adaptační schémata mohou být rozděleny do dvou skupin:

- poměr signálu a šumu (SNR)
- ztrátovost rámců

Vzhledem k tomu, že účinnost SNR schématu je v praxi omezená, schémata založená na ztrátovosti rámců se stávají atraktivní alternativou. Jeden z nejdříve vymýšlených adaptačních algoritmů, který řídí datovou rychlost na základě ztrátovosti rámců byl Auto Rate Fallback (ARF, Kamerman a Monteban 1997), který byl nejvíce zaváděn do praxe díky své jednoduchosti.[2] [6]. V této části jsou popsány nejznámější algoritmy adaptace rychlosti.[2]

2.1 ARF algoritmus

ARF je široce adoptovaný a dobře známý v praxi. Nicméně, široce použitý Auto Rate Fallback algoritmus nefunguje správně v „multiacces“ sítích (síť, do které se připojuje více zařízení). Rozhodnutí o zvýšení nebo snížení přenosové rychlosti je založená na počtu/ech po sobě jdoucích úspěšných nebo neúspěšných pokusů o přenos.

Tento algoritmus je široce používán, protože je jednoduchý. Hlavním problémem tohoto algoritmu je, že nemůže rozlišit mezi ztrátami způsobenými kolizí v kanálu ze ztrát způsobených rušením v kanálu, tak dosahuje špatného výkonu ve scénářích pro více uživatelů.[8] [6]

2.1.1 Samotný algoritmus

Jak to funguje?

Pokud dva po sobě jdoucí rámce ACK nejsou přijaty správně($N_d = 2$),

- Použij další dostupnou nižší rychlost pro druhý pokus a další vysílání a nastav časovač(rate-up timer).

Je-li posledních 10 přenosů ($N_u = 10$) úspěšných (přijatých ACK) nebo časovač vypršel

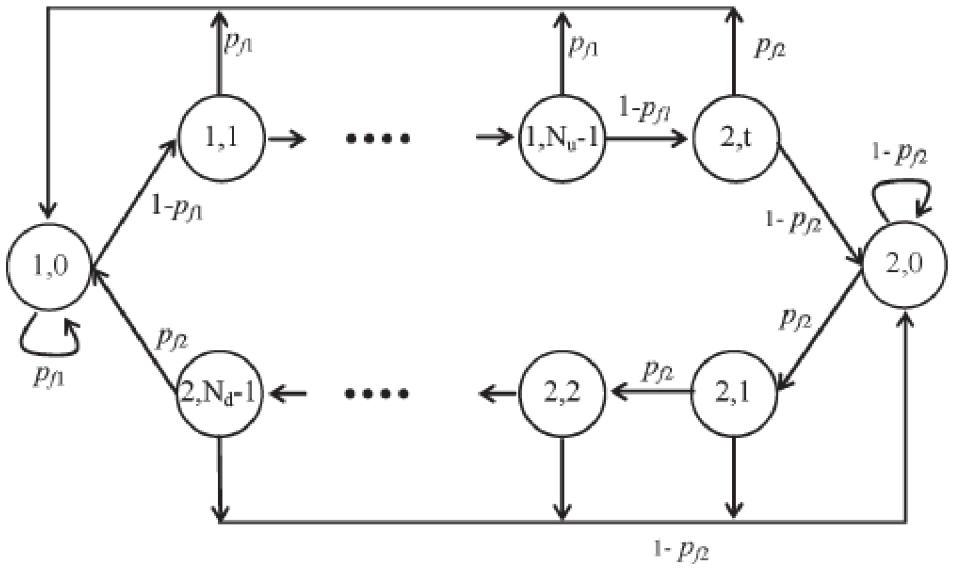
- Pošli zkušební rámec při další dostupné vyšší rychlosti

- Pokud však pro tento rámec NENÍ přijat ACK, rychlost je snížená zpět a časovač se restartuje.[6]

Pseudokód ARF algoritmu se nachází v příloze, viz (Obr.1).

2.1.2 Problém s ARF-em

Navzdory širokému uplatnění ARF, lze prokázat, že přizpůsobení přenosové rychlosti pomocí ARF, může být chybné, pokud vice stanic soutěží o přístup na přenosové medium. Zvažme bezdrátovou lokální síť , ve které řada aktivních stanic odesílá datové rámce. Následující tvrzení lze dokázat.



Obr. 2.1: Diagram přechodu stavů ARF-u

Tvrzení: *Vzhledem k souboru parametrů řízení (Nd, Nu a rate-up timer) pro ARF, pokud je počet aktivních stanic velký, průměrná výkonnost s ARF může být horší než případ, kdy každá stanice používá pevnou přenosovou rychlost, která byla náhodně vybraná.*

Předpokládáme, že existuje n aktivních stanic v bezdrátové lokální síti (WLAN) a stanice mohou vysílat na dvou přenosových rychlostech R_1 a R_2 kde je $R_1 \leq R_2$. Nechť je ER_1 efektivní rychlost $R1$ definovaná jako $ER1 = R1 \times (1 - p_{e1})$, kde je p_{e1} pravděpodobnost ztráty datových rámců při první rychlosti R1. Stejně tak máme ER_2 pro přenosovou rychlost $R2$. Zvážme případ kdy je stav kanálu dostatečně dobrý, to znamená $ER_1 \leq ER_2$. Nechť $\{1, i\}$ $(0 \leq i \leq Nu - 1)$ je stav stanice, která vysílá rychlostí R_1 a předchozích i paketů bylo úspěšně přeneseno. Nechť $\{2, t\}$ je stav, kdy se přenosová rychlost stanice právě změnila z R_1 na R_2. V tomto okamžiku následující přenos musí být úspěšný, aby se rychlost přenosu udržela na R2.

Nechť $\{2, j\}\,(0 \leq j \leq Nd-1)$ je stav stanice která vysílá rychlostí R_2 a předchozích j paketů nebylo úspěšně přeneseno. Poďme nejprve ignorovat vliv rate-up timeru. Nechme, aby $pf1$ a $pf2$ reprezentovaly pravděpodobnost neúspěchu (ztráty) přenášených datových rámců rychlostmi R_1 a R_2. Zanedbejme pravděpodobnost kolize datových rámců způspbených p_c. Potom máme:

$$p_{f1} = p_c + p_{e1} - p_c \times p_{e1} \tag{2.1}$$

$$p_{f2} = p_c + p_{e2} - p_c \times p_{e2} \tag{2.2}$$

Stavy $\{1, i\}\,(0 \leq i \leq Nu-1)$, a $\{2, j\}\,(0 \leq j \leq Nd-1)$ tvoří Markův řetězec, viz (Obr.2.1). Pravděpodobnost změny stavů v Markově řetězci krok za krokem/krok po kroku jsou zobrazeny níže:

$$\begin{aligned}
&P(1, i+1 \mid 1, i) = 1 - p_{f1}, && 0 \leq i < N_u - 1 \\
&P(1, 0 \mid 1, i) = p_{f1}, && 0 \leq i < N_u - 1 \\
&P(2, t \mid 1, N_u - 1) = 1 - p_{f1} && \\
&P(2, j+1 \mid 2, j) = p_{f2}, && 0 \leq i < N_d - 1 \\
&P(2, 0 \mid 2, j) = 1 - p_{f2}, && 0 \leq i < N_d - 1 \\
&P(1, 0 \mid 2, N_d - 1) = p_{f2} && \\
&P(2, 0 \mid 2, t) = 1 - p_{f2} && \\
&P(1, 0 \mid 2, t) = p_{f2}. &&
\end{aligned} \tag{2.3}$$

Z (2.3) a (Obr.2.1) máme:

$$P_{2,t} = P_{1,0}(1 - p_{f1})^{N_u} \tag{2.4}$$

$$P_{2,t} + \sum_{j=0}^{N_d-1} P_{2,j} = P_{2,t}\left[1 + \frac{(1-p_{f2})(1 + p_{f2} + p_{f2}^2 + ... + p_{f2}^{N_d-1})}{1 - (1-p_{f2}) - ... - p_{f2}^{N_d-1}(1-p_{f2})}\right] = \frac{P_{2,t}}{p_{f2}^{N_d}} = \frac{P_{1,0}(1-p_{f1})^{N_u}}{p_{f2}^{N_d}} \tag{2.5}$$

$$\sum_{i=0}^{N_u-1} P_{1,i} = P_{1,0}\left[1 + (1-p_{f1}) + (1-p_{f1})^2 + ... + (1-p_{f1})^{N_u-1}\right] \tag{2.6}$$

Abychom prokázali tvrzení, musíme pouze ukázat, že když n je dostatečně velký, průměrný počet stanic, které vysílají rychlostí R1 je větší než n / 2 při uplatnění ARF.

$$\sum_{i=0}^{N_u-1} P_{1,i} > P_{2,t} + \sum_{j=0}^{N_d-1} P_{2,j} \tag{2.7}$$

anebo

$$\frac{1 + (1-p_{f1}) + (1-p_{f1})^2 + ... + (1-p_{f1})^{N_u-1}}{(1-p_{f1})^{N_u}} > \frac{1}{P_{f2}^{N_d}} \tag{2.8}$$

Lze prokázat, že $\frac{1}{P_{f2}^{N_d}}$ klesá z $\frac{1}{P_{e2}^{N_d}}$ (při $p_c = 0$) až na 1 (při $p_c = 1$) a $\frac{1+(1-p_{f1})+(1-p_{f1})^2+...+(1-p_{f1})^{N_u-1}}{(1-p_{f1})^{N_u}}$ se zvyšuje na ∞ (při $p_c = 1$). U n aktivních stanic v síti WLAN je p_c dána hodnotou:

$$p_c = 1 - (1-\tau)^{n-1} \tag{2.9}$$

kde τ je pravděpodobnost, že stanice vysílá v náhodně vybraném časovém slotu. Hodnota τ závisí na průměrné velikosti okna soutěžení (CW). V praxi platí, že pro stanice, které implementují 802.11 standard WLAN, existují dva parametry, které mohou omezit maximální hodnotu velikosti okna soutěžení (CW): 1) CWmax a 2) CWmin. Pokud jedno z obou omezení bylo dosaženo, zvýšování velikosti okna soutěžení je zrušeno. Proto mame:

$$\tau \geq \tau_{min} \equiv \frac{2}{min\left\{CW_{max}, CW_{min} \times 2^m\right\} + 1} \tag{2.10}$$

a

$$p_c \geq 1 - (1-\tau_{min})^{n-1} \tag{2.11}$$

Výše uvedená analýza předpokládá homogenní bezdrátovou komunikaci (stejné p_{e1} a p_{e2} pro všechny stanice). Když různé stanice mají různé podmínky kanálu (např. existuje x typů kanálových podmínek), stanice mohou být zařazeny do tříd x. Z výše uvedeného pro každé k ($1 \leq k \leq x$) existuje Nk takové, že když je počet stanic $n > Nk$, tvrzení platí pro třídy k. Pokud je celkový počet stanic dostatečně velký (větší než maximální Nk), tvrzení platí pro všechny stanice. Nyní zvážme rate-up timer. Funkce rate-up timeru je zvýšit pravděpodobnost přechodu z $R1$ na $R2$. Nicméně, pro předem stanovenou hodnotu časovače (obvykle řadu vteřin nebo minut), zvýšení pravděpodobnosti přechodu je poměrně omezená. Na základě (2.6) a (2.11), dokud počet stanic je dostatečně velký, účinek rate-up timeru, nemůže kompenzovat fakt, že kolize způsobují, aby více stanic vysílalo rychlostí $R1$ než rychlostí $R2$. Pro zjednodušení se výše uvedená analýza zabývá pouze dvěma datovými rychlostmi přenosu, ale domníváme se, že je lze rozšířit na standard, který podporuje více datových rychlostí (např. 802.11g). Analýza ukazuje, že za dobrých kanálových podmínek, které by měly upřednostňovat vyšší datovou rychlost, kdy je

počet stanic dostatečně velký, kolize by způsobily, že většina stanic zůstane s nižší přenosovou rychlostí. Proto, ARF může dát horší výkon, než kdybychom nepoužívali ARF. ARF algoritmus není schopen přizpůsobit se oběma změnám stavu kanálu a kolizím mezi velkým počtem aktivních stanic.[18]

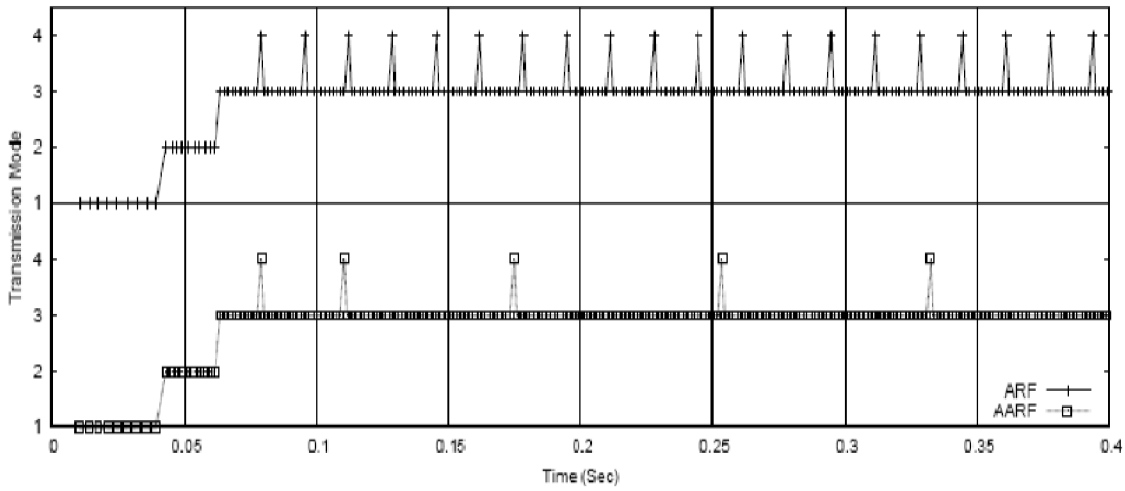
2.1.3 AARF (Adaptive Auto Rate Fallback) algoritmus

Systémy s nízkou latencí (zpožděním) nám umožňují implementovat „per-packet adaptation“. To znamená, že pro každý odeslaný paket musíme získat zpětnou informaci o stavu přenosu tohoto paketu před odesláním dalšího paketu. MAC 802.11 protokol označuje úspěšnost přenosu zasláním zpět ACK (potvrzení) k vysílači a neúspěšnost přenosu je zjištěna chybějící ACK (neexistuje Negativní ACK).

AARF funguje tak, že když přenos zkušebního paketu novou rychlostí selže, tak se rychlost přenosu okamžitě sníží na předchozí používanou (jako u ARF), ale na rozdíl od ARF dvakrát vynásobíme počet po sobě jdoucích úspěšných přenosů (s maximální hraniční hodnotou 50) požadovaných pro přechod na vyšší rychlost (proměnná success_threshold). Prahová hodnota (success_threshold) je resetována na počáteční hodnotu 10, když je rychlost snížená z důvodu dvou po sobě jdoucích neúspěšných přenosů. Účelem tohoto adaptačního algoritmu je prodloužit období mezi následujícími neúspěšnými pokusy o použití vyšší rychlosti. Méně neúspěšných přenosů a opakovaných přenosů zlepšují celkovou propustnost. (Obr.2.2) ukazuje dobu, kdy je nejúčinnější přenosový režim 3. ARF se pokusí použít režim 4 po 10 úspěšných přenosů v režimu 3, zatímco AARF zvyšuje počet potřebných přenosů pro odeslání zkušebního paketu vyšší rychlosti.[13][17] Seznám proměnných, které jsou použité v AARF pseudo kódu, viz přílohu (Obr.2):

- **timer**: Tato proměnná je inkrementovaná po každém přeneseném paketu (bez ohledu na úspěch nebo neúspěch). Tato proměnná je resetována na nulu, když se provede dva nebo více po sobě jdoucích pokusů o přenos, nebo když bylo úspěšně přeneseno 10 po sobě jdoucích paketů.
- **success**: počet úspěšně přenesených po sobě jdoucích paketů.
- **recovery**: pokud je nastavena na hodnotu TRUE, znamená to, že jsme právě odeslali zkušební paket, abychom vyzkoušeli vyšší přenosovou (datovou) rychlost. V opačném případě, proměnná Recovery je nastavená na FALSE.
- **retry**: počet po sobě jdoucích pokusů pro daný paket.
- **success_threshold**: počet po sobě jdoucích úspěšných přenosů, které jsou požadované k odeslání zkušebního paketu (pro zkušení vyšší přenosové rychlosti).
- **timeout**: Pokud rate-up timer (časovač) dosáhne tuto hodnotu, zkušební paket bude odeslán.

- **retry_threshold**: Počet po sobě jdoucích neúspěchů požadovaných pro zahození paketu.
- **min_success_threshold**: Hodnota, která je použitá k inicializaci proměnné success_threshold, když je datová rychlost snížená z důvodu dvou po sobě neúspěšných přenosů.
- **max_success_threshold**: Maximální možná hodnota success_threshold.
- **success_k**: Multiplikativní faktor, který se používá pro výpočet nové hodnoty success_threshold-u, když zkušební paket selže
- **min_timeout**: Minimalní hodnota timeout-u.
- **timeout_k**: U AARF-u, proměnná timeout je nastavená buď na mintimeout, anebo success_threshold x timeout_k.[17]



Obr. 2.2: Srovnání výběru režimu mezi ARF-em a AARF-em.

3 NETWORK SIMULATOR 3

Ns-3 (network simulator) je síťový simulátor určený převážně pro výzkumné a vzdělávací účely. Jedná se o open-source program dostupný pod licencí GNU GPLv2. Vývoj tohoto programu začal v roce 2006. Hlavní cíl, který si program klade, je tvorba simulací v souladu s potřebami moderního síťového výzkumu. Software ns-3 umožňuje vývoj realistických simulačních modelů. Podporována je i emulace sítí v reálném čase a implementace protokolů a jejich opětovné použití.[15]

3.1 Úvod do NS3

Na vývoji síťového simulátoru se podílela skupina lidí, mezi nimiž byli Tom Henderson a George Riley, kteří se snažili vytvořit náhradu za simulátor ns-2. Na projektu spolupracovalo výzkumné centrum INRI. První vydaná verze byla ns-3.1 v červnu 2008, a každé čtvrtletí pokračovalo vydání nové verze softwaru.

Síťový simulátor je postaven na základě programovacích jazyků C ++ a Python. Liší se od simulátoru ns-2, který používá jazyk OTcl namísto jazyka Python. Ns-3 není zpětně kompatibilní s ns-2 a nepodporuje aplikační programové rozhraní (API).

Ns-3 poskytuje modely, které ukazují jak pracují paketové sítě. Rovněž umožňuje realizaci náročnějších nebo těžko uskutečnitelných studií na reálných systémech pro testování chování systému ve vysoce kontrolovaném prostředí. Možné je reprodukování simulace pro zjištění jak pracuje daná síť. Dostupné modely v ns-3 se soustřeďují na modelování internetových protokolů, ty však nelimitují použití simulátoru.

Design ns-3 sestává ze sady vzájemně kombinovatelných knihoven, které jsou spojitelné s dalšími externími softwarovými knihovnami. Grafické rozhraní ns-3 je modulární a tvořené několika externími animačními a vizualizačními nástroji.

Podporovány jsou i nástroje pro analýzu dat. Primárně je simulátor využíván na Linuxových systémech. Není oficiálně podporovaný produkt žádné společnosti. Simulátor ns-3 je celý napsán v jazyce C ++, s možnou vazbou na Python. Z toho vyvozujeme, že zdrojový kód může být napsán v jazyce C ++ nebo Python. Jelikož ns-3 generuje trasovač pakety typu PCAP, pro analýzu trasy mohou být použity i další programy.

Tvorba kvalitního síťového simulátoru s dostatečným počtem kvalitních a udržovaných modelů vyžaduje značné množství práce. Ns-3 simulátor se snaží rozdělit práci mezi velkou komunitou uživatelů a vývojářů, aby dosáhla tohoto cíle. Každé tři měsíce se vydává nová stabilní verze ns-3 s nově vytvořenými modely, které jsou zdokumentovány a udržovány týmem výzkumníků.

3.2 Waf

Zdrojový kód je nutné po stažení zkompilovat pro vytvoření použitelných programů. Kompilování, jinak řečeno překlad, je transformací zdrojového kódu napsaného v programovacím jazyce do strojového kódu, čili nižšího jazyka. Existuje mnoho nástrojů určených pro tuto činnost. Známým je nástroj make, avšak pro jeho složité použití jsou vyvíjeny alternativy. Jednou z nich je **Waf** vyvinut použitím jazyka Python.

3.3 Vývojové prostředí

Skriptování v ns-3 je prostřednictvím jazyka C ++ a Python. Modely jsou napsány v C ++ a většina aplikačního programového rozhraní simulátoru je dostupná v jazyce Python. Systém ns-3 používá několik součástí z GNU sady nástrojů (gcc, gdb), ale nepoužívá nástroj make ani GNU kompilační nástroj. Pro tento případ slouží WAF.

Ns-3 je určen pro Linuxové prostředí. Typické je pro uživatele operačního systému Windows instalování virtuálního stroje a do něj nainstalování Linuxu

3.4 Hlavní abstrakce

3.4.1 Uzel

Ns-3 je síťový simulátor a nejedná se specificky o simulátor Internetu. Výpočtové zařízení připojené do sítě se běžně označuje jako hostitel (host), ale tento název se vztahuje na síť Internet a protokoly s ním spojené. My budeme používat pojem uzel (node), který je obecnější a používán i u jiných simulátorů. Abstrakcí základního výpočetního zařízení je tedy uzel, který je reprezentován třídou Node v C ++. Pro řízení výpočetního zařízení v simulaci nám slouží metody třídy Node. Uzel si můžeme představit jako počítač s přidanými funkcemi, jako jsou aplikace.

3.4.2 Aplikace

Počítačový software se rozděluje na systémový a aplikační software. Hranice, která jich rozděluje je vymezenou změnou privilegované úrovně, která se děje ve voláních operačního systému. Ns-3 nepracuje s reálným konceptem operačního systému, privilegovaných úrovní nebo systémovými volanmi. Stejně jako aplikace běží na reálném počítači, běží ns-3 aplikace na uzlech. Aplikace je základní abstrakcí pro uživatelský program, který vytváří aktivitu pro simulaci. Aplikace třída reprezentuje tuto

abstrakci v jazyce C ++. Třída Aplikace obsahuje metody pro správu uživatelských aplikací v simulacích.

3.4.3 Kanál

Médium, pomocí něhož data dostáváme do sítě, označujeme jako kanály. Připojením síťového kabelu do zásuvky připojíte počítač ke komunikačnímu kanálu. V ns-3 simulaci připojujeme uzel k objektu reprezentujícímu komunikační kanál. Abstrakce základní podsítě pro komunikaci se nazývá kanál a je určena třídou Channel v jazyce C ++. Tato třída poskytuje metody pro řízení objektů podsítě a připojuje k nim uzly. Kanál v ns-3 může být použit pro reprezentaci jednoduché linky, ale může také modelovat zařízení jako přepínač.

3.4.4 Pomocníci tvorby topologie

Ve velkých simulovaných sítích je třeba zajistit velké množství spojení mezi uzly, síťovými zařízeními a kanály. Následně je nutné nastavení IP adres a mnoha dalších operací. Ns-3 poskytuje pomocníky tvorby topologie (topology helpers) pro zjednodušení těchto úkonů.

3.5 Zdroje

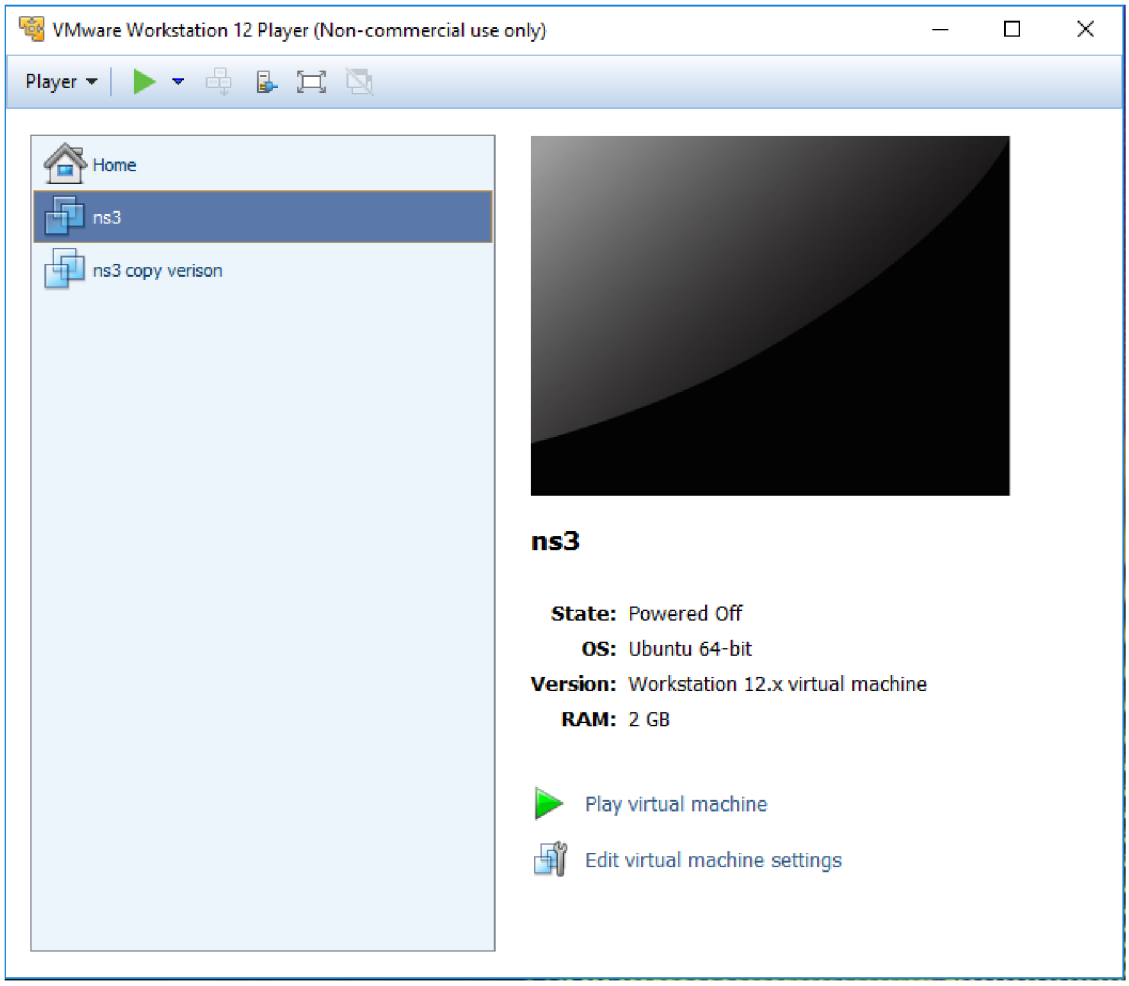
Primárním zdrojem informací pro uživatele ns-3 simulátoru je webová stránka <`http://www.nsnam.org`> Poskytuje přístup k hlavním informacím o ns-3 simulátoru. Prostřednictvím další webové stránky <`http://www.nsnam.org/documentation/`> se dostaneme k detailní dokumentaci a popisu systémové architektury. Na stránce popisující dokumentaci se nachází odkaz pro "API Documentation", který nás přivede na stránku dokumentace aplikačního programového rozhraní. Prostřednictvím nástroje pro generování online dokumentace ze zdrojového kódu je zaznamenána API dokumentace a organizovaná napříč různými moduly.

4 SIMULACE V NS3

V této kapitole bude uveden podrobný popis vytvořeného programu v simulačním prostředí NS-3. Úkolem bylo vytvořit model bezdrátové sítě standardu 802.11g, kde bude jeden přístupový bód a několik pohybujících se stanic. V následujících částech práce bude vysvětlen samotný kód simulace který byl napsán v jazyce C++.[7]

4.1 Předpoklady nutné k simulaci

Pro vytvoření simulace bylo za potřeby nainstalovat několik programů a nastavit Linuxové prostředí. Typické je pro uživatele operačního systému Windows instalování virtuálního stroje a do něj nainstalování Linuxu (4.1), což je taký případ této práce.



Obr. 4.1: Ubuntu linux

4.2 Programové řešení simulace 1

Kód začíná načítáním modulů tj. naimportováním knihoven které se budou používat.

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"
```

Defaultní topologie je následující. Podle zadání semestrální práce, simulace by měla obsahovat několik pohybujících se uzlů a jeden Access Point (AP), se kterým budou komunikovat. Navíc je na AP přípojená LAN síť z důvodu, že je AP v praxi vždycky připojen na pevnou síť, viz níže.

```
// Defaultní Topologie
//
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n2   n3   n4   n0 -------------- n1    (udp echo server)
// (udp echo client) point-to-point  |
//       n2
//                                      LAN 10.1.2.0
```

Po ilustraci se používá jmenný prostor(namespace) `ns-3` a je definována logovací součást, viz níže.

```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ns3arfExample");
```

Hlavní program začíná přidáním některých parametrů příkazového řádku pro povolení nebo zakázání logg zpráv a pro změnu počtu vytvořených zařízení, viz další strana.

```
int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 0;
  uint32_t nWifi = 3;
  uint32_t TimerThreshold = 15;
  uint32_t SuccessThreshold = 10;
  bool tracing = true;

  CommandLine cmd;
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);
  cmd.AddValue ("timerThreshold", "timer threshold", TimerThreshold);
  cmd.AddValue ("successThreshold", "success threshold", SuccessThreshold);

  cmd.Parse (argc,argv);



  if (verbose)
    {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }
```

Dalším krokem je vytvoření dvou uzlů, které budou propojení přes point-to-point linku. Tímto je AP spolu s dalšími uzly připojen na LAN síť, viz níže.

```
NodeContainer p2pNodes;
p2pNodes.Create (2);
```

Inicializujeme `PointToPointHelper` a nastavíme přidružené výchozí atributy, abychom vytvořili vysílač s rychlostí 5 megabitů za sekundu na zařízeních vytvořených pomocí pomocníka a zpoždění dvou milisekund na kanálech vytvořených pomocníkem. Pak následuje instalace uzlů a kanál mezi nimi.

```
PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer p2pDevices;
  p2pDevices = pointToPoint.Install (p2pNodes);
```

V dalším kroku budeme vytvářet uzly, které budou součástí sítě `Wifi`. Budeme vytvářet řadu uzlů "stanic", jak jsou specifikovány v argumentu příkazové řádky, a jako uzel pro přístupový bod budeme používat "nejvzdálenější"uzel propojení bod-bod(point-to-point), viz další strana.

```
NodeContainer wifiStaNodes;
  wifiStaNodes.Create (nWifi);
  NodeContainer wifiApNode = p2pNodes.Get (0);
```

Další část kódu vytváří zařízení `Wifi` a propojovací kanál mezi těmito wifi uzly. Nejprve konfigurujeme pomocníky `PHY`(fyzická vrstva) a kanálu, viz níže.

```
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
  phy.SetChannel (channel.Create ());
```

Jakmile je `PHY` pomocník nakonfigurován, můžeme se zaměřit na vrstvu `MAC`. Metoda `SetRemoteStationManager` říká pomocníkovi jak se bude řídit rychlost. Zde požádá pomocníka, aby použil konstantní rychlost a `OFDM` jako typ modulace.[10]

```
phy.SetPcapDataLinkType(YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
  WifiHelper wifi;
  wifi.SetStandard(WIFI_PHY_STANDARD_80211g);
  wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager",
                    "DataMode",StringValue("ErpOfdmRate6Mbps"));
```

Další část kódu nejprve vytvoří objekt identifikátoru sady služeb 802.11 (SSID), který bude použit pro nastavení hodnoty atributu `Ssid`, viz níže.

```
WifiMacHelper mac;
  Ssid ssid = Ssid ("ns-3-ssid");
  mac.SetType ("ns3::StaWifiMac",
               "Ssid", SsidValue (ssid),
               "ActiveProbing", BooleanValue (false));
```

Jakmile jsou všechny parametry specifické pro stanici plně nakonfigurovány, a to jak ve vrstvě `MAC`, tak ve vrstvě `PHY`, můžeme vyvolat metodu instalace, abychom vytvořili bezdrátová zařízení, viz níže.

```
NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);
```

Máme nakonfigurovanou `Wifi` pro všechny uzly `STA` a nyní je potřeba nakonfigurovat uzel `AP` (přístupový bod). Tento proces začíná změnou výchozích atributů `ApWifiMacHelper` tak, aby odpovídaly požadavkům `AP`, viz níže.

```
NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);
```

Nyní přidáme model mobility. Chceme, aby uzly `STA` byly mobilní, pohybovaly se kolem ohraničovacího boxu a chceme, aby `AP` uzel byl stacionární. Používáme službu `MobilityHelper`, která nám to usnadní.

```
MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
  "MinX", DoubleValue (0.0),
  "MinY", DoubleValue (0.0),
  "DeltaX", DoubleValue (5.0),
  "DeltaY", DoubleValue (10.0),
  "GridWidth", UintegerValue (3),
  "LayoutType", StringValue ("RowFirst"));
```

Chceme, aby přístupový bod zůstal během simulace v pevné poloze. Dosahujeme toho nastavením modelu `mobility` tohoto uzlu jako `ns3 :: ConstantPositionMobilityModel`, viz níže.

```
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);
```

Použijeme `Ipv4AddressHelper` k přiřazení adres IP do našich rozhraní na zařízeních. Nejprve použijeme síť 10.1.1.0 pro vytvoření dvou adres potřebných pro naše dvě zařízení typu point-to-point. Potom použijeme síť 10.1.2.0 k přiřazení adres do pevné sítě a potom přidělíme adresy ze sítě 10.1.3.0 jak zařízením `STA`, tak `AP` v bezdrátové síti.

```
Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);
```

`Udp` server je umístěn na stanici, která patří LAN síti tj. na nejvzdálenější uzel na pravé straně.

```
UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps=echoServer.Install(csmaNodes.Get (1))
    ;
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

`Udp` klient je umístěn na jeden ze `STA` uzlů směřující na server v pevné síti.

```
ApplicationContainer clientApps =
    echoClient.Install (wifiStaNodes.Get (nWifi - 1));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));
```

Vytvářime pcap soubory pro pokrytí všech tří sítí.

```
if (tracing == true)
    {
      pointToPoint.EnablePcapAll ("ns3arf");
      phy.EnablePcap ("ns3arf", apDevices.Get (0));
      csma.EnablePcap ("ns3arf", csmaDevices.Get (0), true);
    }
```

Tyto tři řádky kódu spustí trasování pcap na oba uzly typu point-to-point, které slouží jako páteř, spustí trasování pcap v síti `Wifi` a v síti `CSMA`(pevná sít). To umožňuje sledovat veškerou komunikaci s minimálním počtem pcap souborů.

4.2.1 NetAnim

V simulačním prostředí NS-3 není standardně možnost graficky zobrazit pohyb mobilních uzlů nebo například komunikaci mezi jednotlivými mobilními uzly. Velmi rozšířeným vizualizačním nástrojem je NetAnim. Jedná se o multiplatformní animátor, který je založený na Qt 4 toolkit. NetAnim využívá pro zobrazení simulace XML (eXtensible Markup Language) trasovací soubor, který je vytvořen v průběhu kompilace zdrojového kódu.[14]

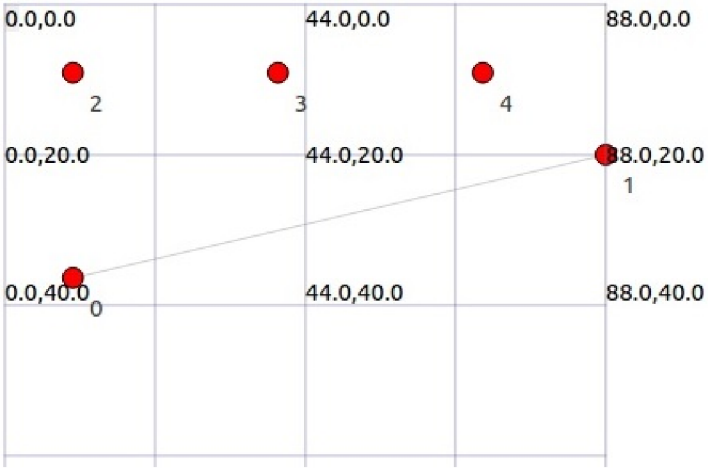
```
#include "ns3/netanim-module.h"
AnimationInterface anim ("ns3arf1.xml");
```

4.3 Výsledky simulace

4.3.1 Zobrazení simulace

V tomto bodě budou ukázány některé situace z průběhu simulace zobrazené programem NetAnim.

Na obrázku (4.2) je vidět jednotlivé uzly před spouštěním simulace. Uzel číslo 0 hraje rolí UDP serveru, který je připojen LAN kabelem na AP (uzel číslo 1). Ostatní uzly představují jednotlivé klienty (stations), který komunikuji s UDP serverem přes AP.



Obr. 4.2: Topologie simulace

4.3.2 RSSI v závislosti na vzdálenosti

Received Signal Strength Indicator anebo v češtině výkon přijatého signálu vyjadřuje jak dobře přístroj „slyší“ signál z přístupového bodu. Je to hodnota, která je užitečná pro zjištění, zda existuje dostatek signálu pro získání dobrého bezdrátového připojení, viz tabulka (4.1).

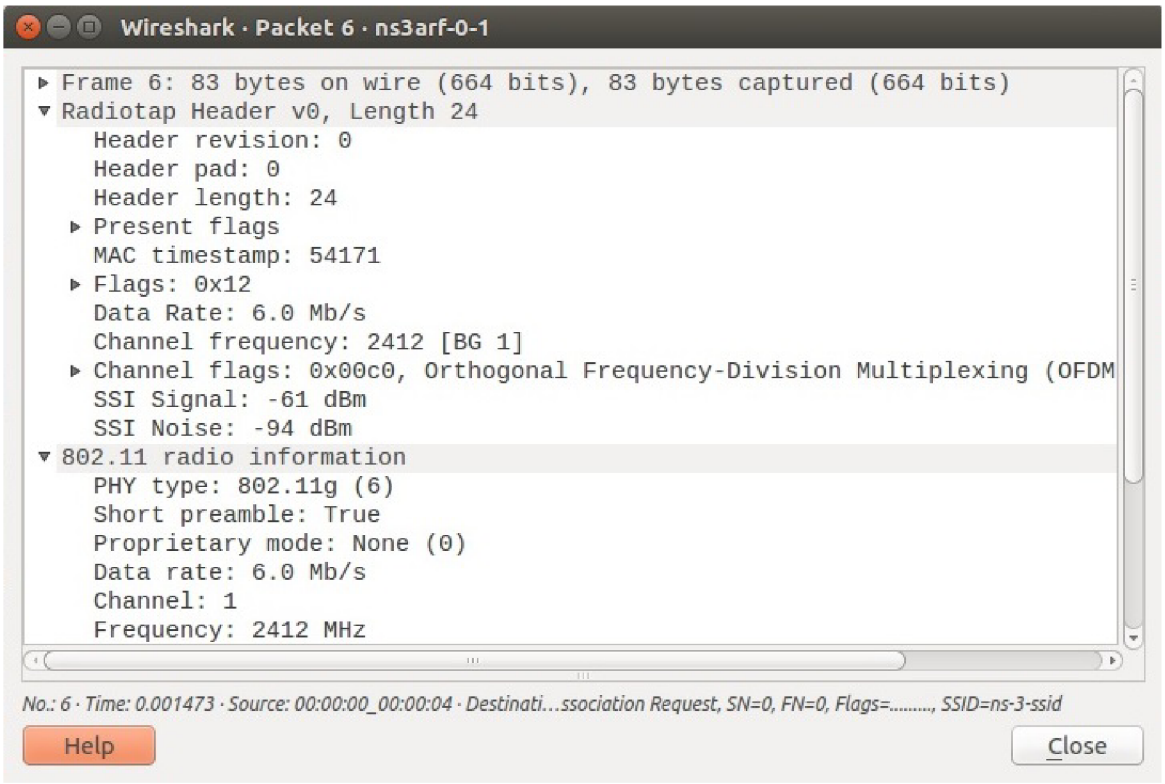
Tab. 4.1: Hodnoty RSSI

Výkon signálu	Kvalita	Komentář	Požadované pro
-35dBm	Perfektní	Maximální dosažitelná síla signálu.	N/A
-67dBm	Velmi dobrý	Minimální síla signálu pro aplikace	VoIP/VoWiFi,streaming video
-70dBm	OK	Minimální síla signálu pro spolehlivé doručení paketů.	Email, web
-80dBm	Špatný	Minimální síla signálu pro základní připojení.	N/A
-90dBm	Nepoužitelný	Veškeré funkce jsou velmi nepravděpodobné.	N/A

V teto části následuje porovnání parametru RSSI na třech úzlech, který se náhodně připojují na přístupový bod. Bude se měnit vzdálenost mezi AP a úzly.

Vzdálenost od AP cca 10m

Výsledky pro uzel č. 4 se zdrojovou mac adresou 00 : 00 : 04 jsou zobrazený na obrázku (4.3). Výkon přijatého signálu z mac adresy 00 : 00 : 07, což je adresa AP, můžeme ohodnotit jako velmi dobrý (-63 dBm), viz (4.1). Tato hodnota by měla stačit na streaming videa tj. na přenos UDP datagramů přes sít. Toto potvrzuje obrázek (4.6). Samozřejmě v záhlaví radiotap se nachází řada dalších parametrů. Je vidět že je použitá rychlost přenosu 6 Mb/s, což odpovídá standardu 802.11g. Rychlost přenosu se během simulace nemění, protože je pevně nastavena na 6 Mb/s. Použita modulace je OFDM.



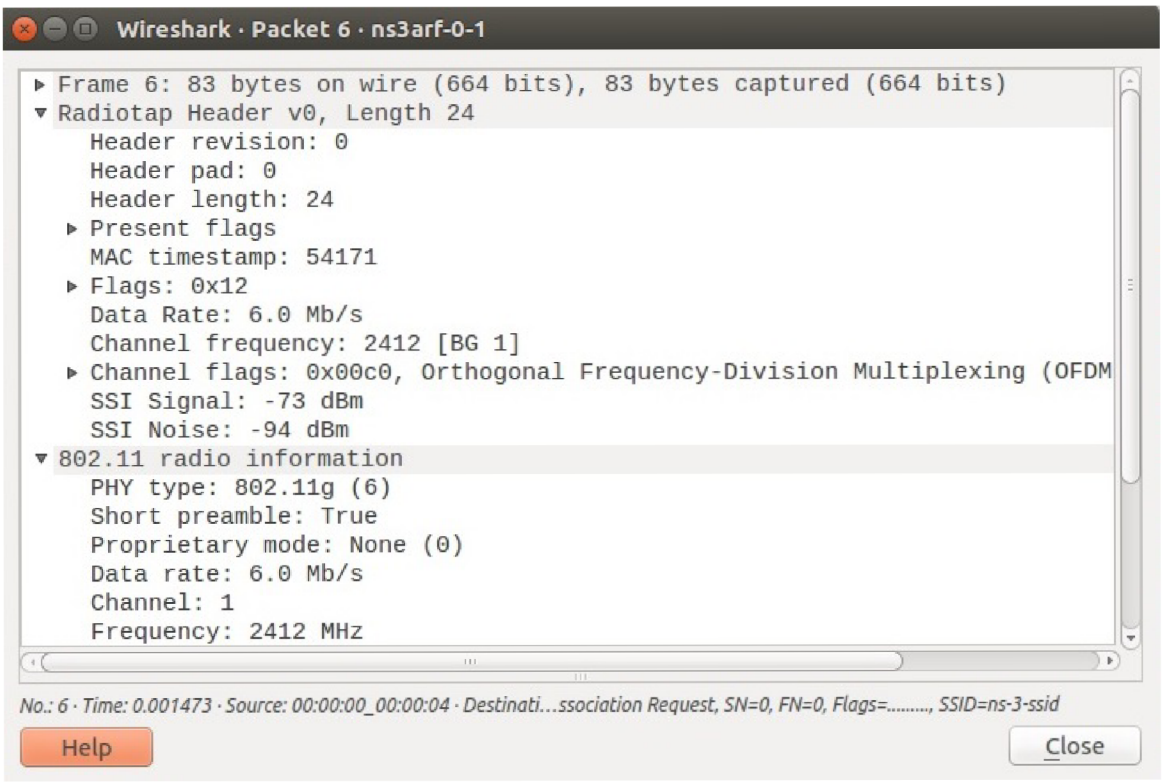
Obr. 4.3: Uzel č. 4, vzdálenost 10 m od AP

Vzdálenost od AP cca 25m

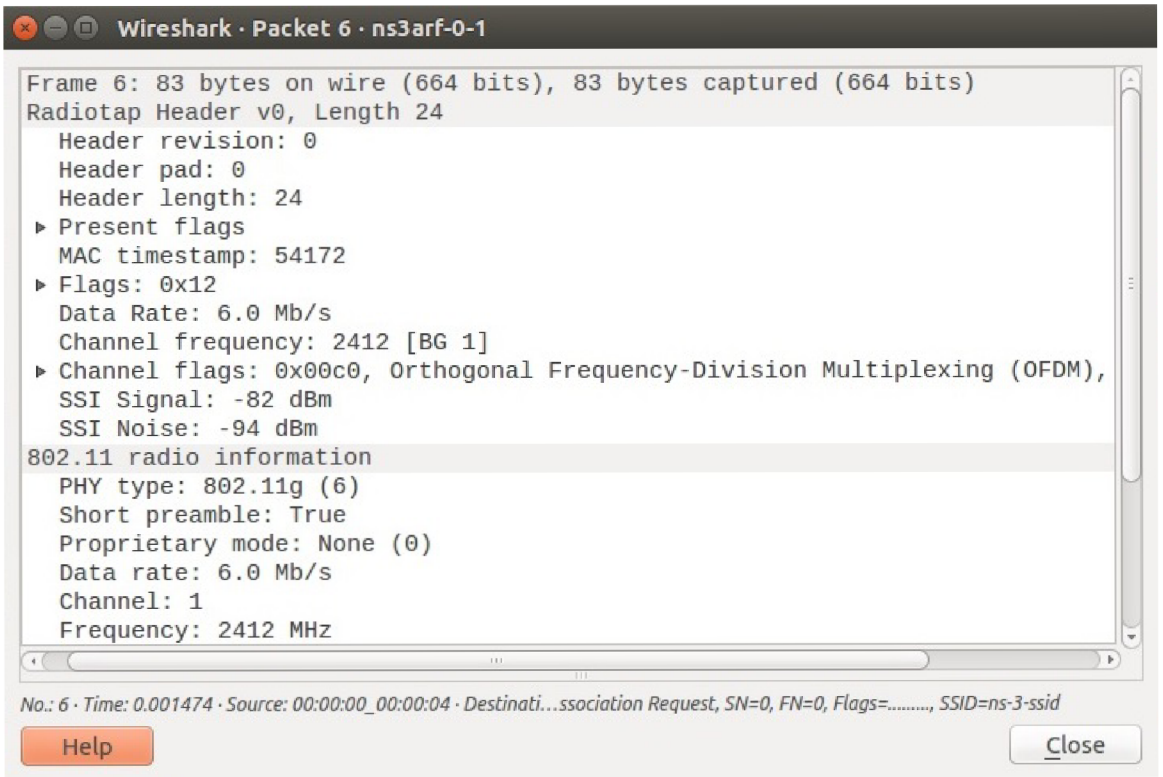
Výsledky pro stejný úzel (4), ale na jiné vzdáleností od AP jsou zobrazený na obrázku (4.4). Výkon přijatého signálu v tomto případě můžeme ohodnotit jako OK (-73 dBm), viz tabulka (4.1). Teoretický by hodnota -73 dBm měla stačit na přenos UDP datagramů, což bylo potvrzeno testem na obrázku (4.6). V záhlaví radiotap se nic nezměnilo, kromě parametru SSI.

Vzdálenost od AP cca 50m

Výsledky pro stejný uzel (4), ale na největší použité vzdálenosti v simulaci jsou zobrazeny na obrázku (4.5). Výkon přijatého signálu v tomto případě můžeme ohodnotit jako špatný anebo nepoužitelný (-82 dBm), viz tabulka (4.1). Teoreticky by hodnota -82 dBm neměla stačit na přenos UDP datagramů, což bylo potvrzeno testem na obrázku 4.7. V záhlaví radiotap se nic nezměnilo, kromě parametru SSI.



Obr. 4.4: Uzel č. 4, vzdálenost 25 m od AP



Obr. 4.5: Uzel č. 4, vzdálenost 50 m od AP

UDP server-klient

Přenos UDP dat je zobrazen na obrázku (4.6). Data jsou přenášena od mobilního uzlu 4 (IP adresa 10.1.3.3, zdrojový port: 49153) k cílovému AP č. 1 (IP adresa 10.1.2.1, cílový port: 9). Tento způsob posílaní dat byl zvolen, aby bylo prokázáno, že parametr RSSI má velký význam pro spolehlivý přenos dat v standardu 802.11g. Přenos dat u vzdálenosti cca 50m nebyl úspěšný, viz obrázek (4.7). Je vidět, že

```
ns3@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$ ./waf --run scratch/ns3arf
Waf: Entering directory `/home/ns3/Desktop/ns-allinone-3.27/ns-3.27/build'
[ 956/2719] Compiling scratch/ns3arf.cc
[2661/2719] Linking build/scratch/ns3arf
Waf: Leaving directory `/home/ns3/Desktop/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (25.799s)
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
At time 2s client sent 1024 bytes to 10.1.2.1 port 9
At time 2.00847s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.00847s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.02369s client received 1024 bytes from 10.1.1.2 port 9
ns3@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$
```

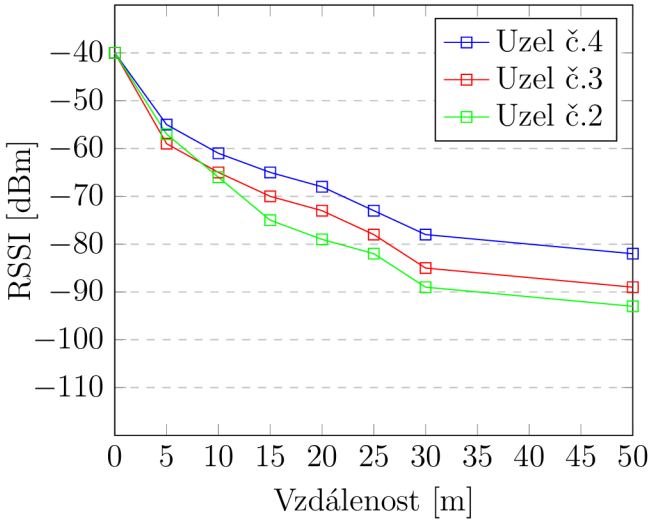
Obr. 4.6: Úspěšný přenos UDP

mobilní uzel 4 (IP adresa 10.1.3.3, zdrojový port: 49153) poslal data k cílovému AP č. 1 (IP adresa 10.1.2.1, cílový port: 9), který nedostal žadná data, a to z důvodu malého výkonu přijatého signálu, čímž přenos neúspěšně skončil.

```
ns3@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$ ./waf --run scratch/ns3arf
Waf: Entering directory `/home/ns3/Desktop/ns-allinone-3.27/ns-3.27/build'
[2333/2719] Compiling scratch/ns3arf.cc
[2705/2719] Linking build/scratch/ns3arf
Waf: Leaving directory `/home/ns3/Desktop/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (17.382s)
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
At time 2s client sent 1024 bytes to 10.1.2.1 port 9
ns3@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$
```

Obr. 4.7: Neúspěšný přenos UDP

Graf závislosti RSSI na vzdálenosti



Graf ukazuje sílu přijatého signálu jako funkci vzdálenosti pro mobilní uzly 4, 3 a 2. Hodnoty RSSI pro jednotlivé uzly se trochu liší kvůli tomu, že byl zvolen model pohybu `ns3::RandomWalk2dMobilityModel`. Jedná se o 2D model, který umožňuje pro každou instanci (mobilní uzel) nastavit náhodnou hodnotu rychlosti a změny směru. Uzly 3 a 2 měly větší počáteční vzdálenost od AP než uzel 4, viz obrázek (4.2).

4.4 Programové řešení simulace 2

Programové řešení první simulace bylo částečně změněno a přizpůsobeno tak, aby odpovídalo druhé simulaci. Druhá simulace se z největší části zabývá adaptačními protokoly a jejich funkcemi. V následující části bude prozkoumáno několik scénářů. Výsledky simulace budou zpracovány do vhodných grafů.

4.4.1 Rozšíření kódu

Aby bylo možné docílit požadované výsledky, tak bylo zapotřebí některé části kódu rozšířit anebo změnit. Třida `UdpEchoClient` byla změněna na `OnOffHelper` z důvodu že, `OnOffHelper` lze použít jako generátor UDP provozu během celé simulace. Pomoci třídy `UdpEchoClient` bylo možné nastavit jenom určitý počet paketů. Přenosová rychlost je nastavená na 54 Mb/s, což je nejvyšší podporovaná rychlost standardu 802.11g, viz další strana.

```
PacketSinkHelper sink ("ns3::UdpSocketFactory", InetSocketAddress (sinkAddress,
    port));
  ApplicationContainer apps_sink = sink.Install (wifiStaNodes.Get (0));

  OnOffHelper onoff ("ns3::UdpSocketFactory", InetSocketAddress (sinkAddress, port)
      );
  onoff.SetConstantRate (DataRate ("54Mb/s"), 1420);
  onoff.SetAttribute ("StartTime", TimeValue (Seconds (0.5)));
  onoff.SetAttribute ("StopTime", TimeValue (Seconds (simuTime)));
  ApplicationContainer apps_source = onoff.Install (wifiApNodes.Get (0));

  apps_sink.Start (Seconds (0.5));
  apps_sink.Stop (Seconds (simuTime));
```

Také bylo zapotřebí změnit mobilitu stanic. Aby bylo zjištěno jaky vliv na přenos dat má vzdálenost stanice od AP-u, tak byl zvolen přímočarý pohyb stanic, viz níže.

```
MobilityHelper mobility;
  Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ()
      ;
  //Initial position of AP and STA
  positionAlloc->Add (Vector (ap1_x, ap1_y, 0.0));
  positionAlloc->Add (Vector (sta1_x, sta1_y, 0.0));
  mobility.SetPositionAllocator (positionAlloc);
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (wifiApNodes.Get (0));
  mobility.Install (wifiStaNodes.Get (0));
```

Tato třída implementuje tzv. ARF algoritmus, který se používá pro detekci kolizí. Pro použití jiného algoritmu stačí změnit název v závorce, viz níže.[12]

```
wifi.SetRemoteStationManager ("ns3::ArfWifiManager");
```

Aby bylo možné rozlišit ztráty způsobené kolizi paketů ze ztrát způsobených špatným kanálem (malý výkon signálu, SNR atd.), tak bylo zapotřebí udělat dvě metody které to umožňují. První metoda sleduje každý paket v kanálu (pomocí smart pointer-u) a inkrementuje se v případě, že chyba nastala na MAC vrstvě. Když chyba nastala na MAC vrstvě to znamená, že je důsledkem kolize paketů, viz níže.

```
void MacTxDrop(Ptr<const Packet> p)
{
NS_LOG_INFO("Packet␣Drop");
MacTxDropCount++;

}
```

Druhá metoda sleduje každý paket v kanálu (pomocí smart pointer-u) a inkrementuje se v případě, že chyba nastala na fyzické vrstvě (PHY). Když chyba nastala na PHY vrstvě to znamená, že je důsledkem rušení v kanálu, malý výkon signálu atd., viz níže.

```
PhyTxDrop(Ptr<const Packet> p)
{
  NS_LOG_INFO("Packet Drop");
  PhyTxDropCount++;
}
```

Nejdůležitější je metoda `AdvancePosition`, která obsahuje všechny potřebné parametry pro výpis do konzole. Např. okamžitá pozice uzlu, přenosová rychlost, ztrátovost paketů atd., viz níže.

```
void
NodeStatistics::AdvancePosition (Ptr<Node> node, int stepsSize, int stepsTime)
{
  Vector pos = GetPosition (node);
  double mbs = ((m_bytesTotal * 8.0) / (1000000 * stepsTime));
  std::cout<<"STA-Time: "<< Simulator::Now().GetSeconds() << "At position: " << pos
      .x <<"has Data Rate = " <<mbs<<std::endl;
  std::cout<<"STA-RXDrop Time: "<<Simulator::Now().GetSeconds()<<"At position: "<<
      pos.x<<"has number of packets STARxDrop= "<<MNPhyRxDropCount<<std::endl;

  double RXmbs=((m_RXbytesTotal * 8.0) / (1000000 * stepsTime));     // Mbps
      Throughput
  double TXmbs = ((m_TXbytesTotal  * 8.0) / (1000000 * stepsTime));    // Mbps
      Throughput
  std::cout<<"AP-RX Time: "<<Simulator::Now().GetSeconds()<<"has Data Rate = "<<
      RXmbs<<std::endl;
  std::cout<<"AP-TX Time: "<<Simulator::Now().GetSeconds()<<"has Data Rate = "<<
      TXmbs<<std::endl;
  std::cout<<"AP-TXDrop Time: "<<Simulator::Now().GetSeconds()<<" has number of
      packets APTxDrop = " <<PhyRxDropCount<<std::endl;
  std::cout<<"AP-RXDrop Time: " <<Simulator::Now().GetSeconds()<< "has number of
      packets APRxDrop = " <<PhyTxDropCount<<"\n"<<std::endl;
  m_RXbytesTotal = 0;
  m_TXbytesTotal = 0;
  m_bytesTotal = 0;
  m_output.Add (pos.x, mbs);
  pos.x += stepsSize;
  SetPosition (node, pos);
  Simulator::Schedule (Seconds (stepsTime), &NodeStatistics::AdvancePosition, this,
       node, stepsSize, stepsTime);
}
```

Do hlavní funkce byly přidáné další proměnné, které určují počáteční pozice uzlů a hraniční vzdálenost, do které se stanice budou pohybovat, viz níže.

```
int main (int argc, char *argv[])
{
  uint32_t rtsThreshold = 2346;
  std::string manager = "ns3::ArfWifiManager";
  std::string outputFileName = "arf";
  int ap1_x = 0;
  int ap1_y = 0;
  int sta1_x = 5;
```

```
int sta1_y = 0;
int steps = 200;
int stepsSize = 1;
int stepsTime = 1;
```

4.4.2 Výsledky simulace

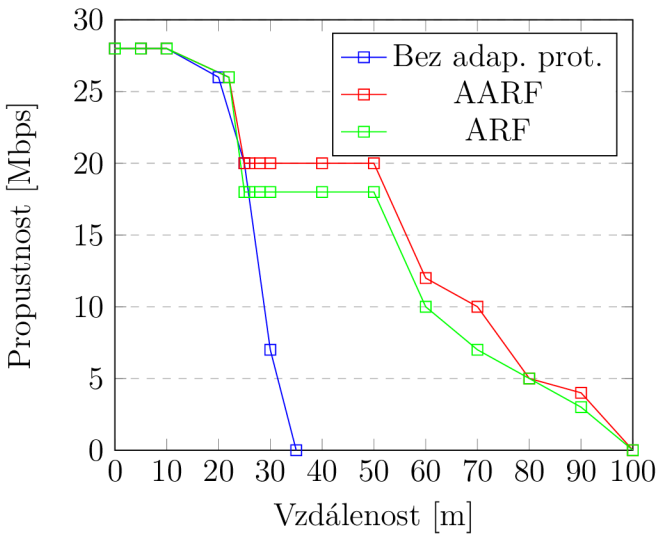
V této části budou prezentované výsledky několik scénářů, kde hlavní roli hrají adaptační protokoly a parametry důležité pro ně. Byl vyhodnocen výkon zkoumaných adaptačních protokolů pomocí simulací prostřednictvím síťového simulátoru ns-3. V simulaci se zajímáme především o propustnost (throughput) a ztrátovost paketů pod proměnným počtem uzlů a různých kanálových podmínek. Pro srovnání byl hodnocen výkon ARF a AARF.

Byl považován scénář s proměnným počtem „soupeřících“ uzlů, kde každý uzel může přímo komunikovat pouze s přístupovým bodem (AP) a vzdaluje se od AP krokem 1 meter každou sekundu. Simulace trvá zhruba 200 sekund. Všechny uzly jsou vybaveny rozhraním IEEE 802.11g a používají stejné adaptační schéma. Datový provoz je generován zdrojem přenosu UDP s konstantní přenosovou rychlostí a velikost paketu je 1420 bajtů. Počáteční rychlost je nastavená na 54 Mb/s. Každý uzel přenáší v nasyceném režimu, tj. jeho datová fronta není nikdy prázdná.

Jediná stanice vzdalující se od AP

Jako první bod testování, byl uvažován nejjednodušší scénář, ve kterém pouze jeden uzel přijímá data z AP a současně zvyšuje vzdálenost od AP. Na obrázku (4.9) je ukázka výpisu do konzole, kde je vidět, že rychlost přenosu klesá se zvyšující se vzdálenosti. Taktéž je zřejmě, že se počet chyb zvyšuje. Závislost propustnosti jako funkce vzdálenosti od AP je vynesená do grafu. Obecně, propustnost všech testovaných protokolů (schemat) klesá se zvyšující se vzdálenosti. Kvůli téměř žádné korekci chyb v scénáři bez adaptačního protokolu propustnost drasticky klesá, když se vzdálenost zvyšuje a stává se nulou, pokud je vzdálenost větší než 35 m, viz graf (4.8).

Podle očekávání, adaptační protokoly ARF a AARF dosahují přibližně stejnou propustnost. V podstatě, v tomto scénáři, AARF může dosáhnout trochu vyšší výkon než ARF. Je to proto, že pro korekci chyb může AARF dynamicky upravit prahovou hodnotu (threshold) pro zvýšení přenosové rychlosti, i když existuje pouze jeden uzel, viz graf (4.8).



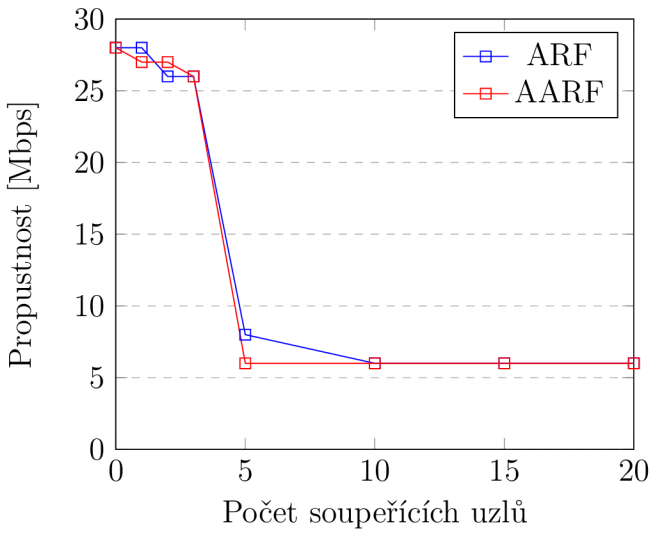
Obr. 4.8: Závislost propustnosti jako funkce vzdálenosti

```
ns3@ubuntu: ~/Desktop/ns-allinone-3.27/ns-3.27
AP-TXDrop  Time: 16.5 has number of packets APTxDrop = 0
AP-RXDrop  Time: 16.5 has number of packets APRxDrop = 0

STA-Time: 17.5  At position: 21 has Data Rate = 28.4227
STA-RXDrop  Time: 17.5  At position: 21 has number of packets STARxDrop = 7
AP-RX  Time: 17.5 has Data Rate = 0
AP-TX  Time: 17.5 has Data Rate = 29.1549
AP-TXDrop  Time: 17.5 has number of packets APTxDrop = 0
AP-RXDrop  Time: 17.5 has number of packets APRxDrop = 0

STA-Time: 18.5  At position: 22 has Data Rate = 26.2302
STA-RXDrop  Time: 18.5  At position: 22 has number of packets STARxDrop = 114
AP-RX  Time: 18.5 has Data Rate = 0
AP-TX  Time: 18.5 has Data Rate = 26.8836
AP-TXDrop  Time: 18.5 has number of packets APTxDrop = 0
AP-RXDrop  Time: 18.5 has number of packets APRxDrop = 0

STA-Time: 19.5  At position: 23 has Data Rate = 22.027
STA-RXDrop  Time: 19.5  At position: 23 has number of packets STARxDrop = 374
AP-RX  Time: 19.5 has Data Rate = 0
AP-TX  Time: 19.5 has Data Rate = 22.5971
AP-TXDrop  Time: 19.5 has number of packets APTxDrop = 0
AP-RXDrop  Time: 19.5 has number of packets APRxDrop = 0

STA-Time: 20.5  At position: 24 has Data Rate = 24.424
STA-RXDrop  Time: 20.5  At position: 24 has number of packets STARxDrop = 487
AP-RX  Time: 20.5 has Data Rate = 0
AP-TX  Time: 20.5 has Data Rate = 25.0316
AP-TXDrop  Time: 20.5 has number of packets APTxDrop = 0
AP-RXDrop  Time: 20.5 has number of packets APRxDrop = 0

STA-Time: 21.5  At position: 25 has Data Rate = 20.4026
STA-RXDrop  Time: 21.5  At position: 25 has number of packets STARxDrop = 751
AP-RX  Time: 21.5 has Data Rate = 0
AP-TX  Time: 21.5 has Data Rate = 20.9315
```

Obr. 4.9: Výpis do konzole 1

Více stanic vzdalujících se od AP

Aby byl analyzován vliv kolize na výkon systému, v tomto scénáři byl vytvořen různý počet soupeřících uzlů, které se pohybují kolem AP. Na grafu je ukázka závislosti propustnosti na počtu soupeřících uzlů, které se zvyšují z 1 na 20. Políčko `STARxDrop` vypisuje do konzole počet ztrácených paketů na MAC vrstvě. Ztrátovost paketů na MAC vrstvě je způsobená kolizemi mezí soupeřícími uzly. Oba protokoly dosahují přibližně stejných výsledků tj. propustností. ARF a AARF jsou téměř stejné, když se počet soupeřících uzlů zvyšuje od 1 do 5, což je kvůli nízké kolizi paketů v kanálu. Vzhledem k tomu, že ARF a AARF se nemohou zbavit efektu kolize, jejich propustnost se drasticky zhoršuje, jelikož se počet uzlů zvětšuje. Na druhou stranu, s nárůstem počtu soupeřících uzlů, pravděpodobnost výskytu 10 úspěšných přenosů za sebou stejnou přenosovou rychlostí je velmi obtížné a vede k výsledku, že AARF dosahuje téměř stejné propustnosti jako ARF.



Obr. 4.10: Závislost propustnosti na počtu/ech uzlů

Simulace s chybovým kanálem

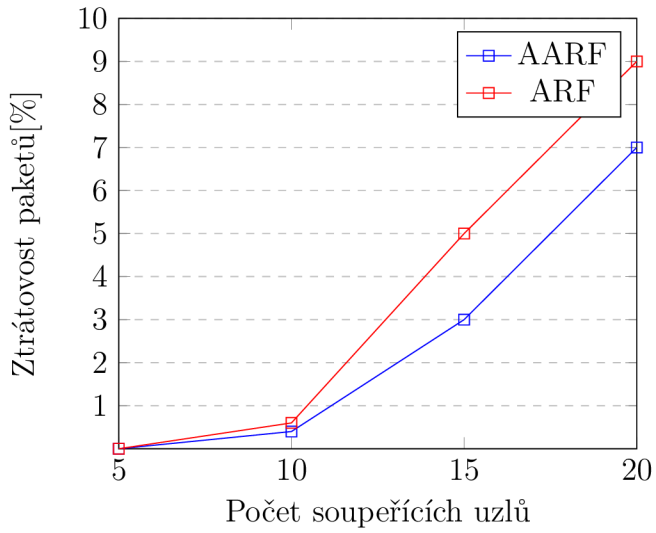
V této části byly záměrně zhoršené podmínky v kanálu. Pomoci LogDistancePro pagationLossModel-u je možné měnit vlastnosti kanálu. Tento model vypočítá přijatý výkon pomocí takzvaného „log-distance propagation model“.[19] Vzorec pro výpočet je následující: $L = L_0 + 10nlog_{10}(\frac{d}{d_0})$ kde:

- n : exponent ztrátovosti cesty
- d_0 : referenční vzdálenost
- L_0 : ztrátovost cesty na referenční vzdálenosti
- d : vzdálenost

- L :¨ztrátovost cesty
  Implementace do programového řešení, viz níže:

```
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
  channel.SetPropagationDelay ("ns3::
      ConstantSpeedPropagationDelayModel");
  channel.AddPropagationLoss ("ns3::
      LogDistancePropagationLossModel",
                                  "Exponent", DoubleValue (3.0),
                                  "ReferenceLoss", DoubleValue
                                      (40.0459));
```

Počet uzlů se zvyšuje jako v předchozí části. Z grafu (4.11) je zřejmě, že se zvyšujícím se počtem uzlů, také roste ztrátovost paketů v procentech. AARF dosahuje o trochu lepších výsledků než ARF a to z důvodu, že sleduje historii kanálu. Dělá to tím, že si pamatuje počet neúspěšných pokusů o posílání zkušebního paketů vyšší rychlosti, a pokaždé pokus selže, algoritmus vynásobí počet po sobě jedoucích úspěšných přenosů o dvou, až do maximálně 50. Pokud paket selže dvakrát při aktuální rychlosti, sníží přenosovou rychlost o krok (např. z 12 Mbps na 6 Mbps) a zresetuje čítač na 10 jako ARF.



Obr. 4.11: Závislost ztrátovosti paketů na počtu/ech uzlů

5 ZÁVĚR

V teoretické části práce byl uveden popis standardů IEEE 802.11 z pohledu charakteristických vlastností. Dále byl popsán ARF algoritmus pro řízení přenosových rychlostí v médiu. Na konci teoretické části byl uveden podrobnější popis NS-3 spolu se hlavními abstrakcemi programu.

V praktické části v první fázi byla vytvořena topologie obsahující AP připojený na LAN síť a několik pohybujících se uzlů (WLAN síť). Dále byl simulován UDP přenos v sítí a byly zjištěny kritické hodnoty RSSI pro úspěšný přenos UDP dat. Z dat získaných ze simulace lze konstatovat, že je parametr RSSI, kromě přenosové rychlosti a odstupu signálu od šumu, jeden z nejdůležitějších parametrů pro přenos dat.

Druhá fáze praktické části se zabývá implementací adaptačních protokolů. Dále byl upraven kód z první fáze, kde byl změněn model pohybu stanic a pomocí metody `SetRemoteStationManager` byly přidany adaptační protokoly. Výsledky simulace byly zpracovány do grafů, který byly vhodně okomentovaný.

LITERATURA

[1] SPONSOR a LAN/MAN STANDARDS COMMITTEE OF THE IEEE COMPUTER SOCIETY. *IEEE standard for Information technology telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements.* New York, N.Y: Institute of Electrical and Electronics Engineers, 2003. ISBN 0738137014.

[2] P.A.K. Acharya, A. Sharma, E.M. Belding, K.C. Almeroth, and K. Papagiannaki. *Rate adaptation in congested wireless networks through real-time measurements. Mobile Computing, IEEE Transactions*, 9(11):1535–1550, nov. 2010.

[3] IEEE 802.11 Working Group. *IEEE 802.11g-2003 - further higher data rate extension in the 2.4 ghz band*, 2003.

[4] IEEE 802.11 Working Group. *IEEE 802.11a-1999 - ofdm in the 5ghz band*, 1999.

[5] IEEE 802.11 Working Group. *IEEE 802.11b-1999 - high rate dsss in the 2.4ghz band*, 1999.

[6] Dr. Ramana. *Rate Adaption Algorithms in 802.11 Networks* [online]. Dostupné z URL: <`http://home.iitj.ac.in/~ramana/802-11-rate`>

[7] *ns-3* [online]. Dostupné z URL: <`https://www.nsnam.org/`>.

[8] Jianhua He, Wenyang Guan, Lin Bai, and Kai Chen. *Theoretic analysis of IEEE 802.11 rate adaptation algorithm samplerate. Communications Letters, IEEE,* 15(5):524–526, may 2011.

[9] GROOM, Frank M., Kevin M. GROOM a Stephan JONES. *The basics of 802.11 wireless LANs. Chicago,* Ill.: International Engineering Consortium, c2005. ISBN 9781931695329.

[10] *Wifi Models: Devices. Ns-3 Documentation [online].* Dostupné z URL: <`http://netdb.cis.upenn.edu/rapidnet/doxygen/html/group___wifi.html`>.

[11] *NetAnim.* NS-3 [online]. 2017 Dostupné z URL: <`http://netdb.cis.upenn.edu/rapidnet/doxygen/html/group___wifi.html`>.

[12] LACAGE, M., MANSHAEI, M.H., TURLETTI, T.: *IEEE 802.11 Rate Adaptation: A Practical Approach.* [online] 2004, s. 9 [cit. 2012-11-17]. Dostupné z URL: <`http://cutebugs.net/files/mswim04.pdf`>

[13] NAYANAJITH, R.: *CWAP – Dynamic Rate Selection* [online] [cit. 2017-11-17]. Dostupné z URL: <https://mrncciew.com/2014/11/03/cwap-dynamic-rate-selection/>

[14] *NetAnim*. NS-3 [online]. 2017 [cit. 2017-11-25] Dostupné z URL: <http://www.nsnam.org/wiki/index.php/NetAnim#Feature-set_in_NetAnim_3.0>

[15] *What is ns-3*. NS-3 [online]. 2017 [cit. 2017-11-25] Dostupné z URL: <https://www.nsnam.org/overview/what-is-ns-3/>

[16] MOLNAR, K.: *Hardware počítačových sítí*. [online]. 2018 [cit. 2018-07-25] Vysoké učení technické v Brně, 2012 ISBN: 978-80-214-4449-2

[17] LACAGE, M., MANSHAEI, M.H., TURLETTI, T.: *IEEE 802.11 Rate Adaptation: A Practical Approach.* [Research Report] RR-5208, INRIA. 2004, pp.25. <inria-00070784> Dostupné z URL: <https://hal.inria.fr/inria-00070784/document>

[18] QIXIANG, P., VICTOR C. M. LEUNG *An Enhanced Autorate Algorithm for Wireless Local Area Networks Employing Loss Differentiation* IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 57, NO. 1, JANUARY 2008

[19] *ns-3* [online]. Dostupné z URL: <https://www.nsnam.org/doxygen/classns3_1_1_log_distance_propagation_loss_model.html>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AARF	Adaptive ARF
ACK	Acknowledgment
AP	Access Point
ARF	Auto Rate Fallback
CBR	Constant Bit Rate
CCK	Complementary Code Keying
CTS	Clear to Send
DCF	Distributed Coordination Function
DSSS	Direct Sequence Spread Spectrum
FHSS	Frequency Hopping Spread Spectrum
IEEE	Institute of Electrical and Electronics Engineers
LRS	Limited Rate Support
LAN	Local Area Network
LTS	Long Term Support
MAC	Medium Access Control
NS-3	Network Simulator 3
OFDM	Orthogonal Frequency Division Multiplexing
PLCP	Physical Layer Convergence Procedure
PMD	Physical Medium Dependent
PPDU	PLCP Protocol Data Unit
RTS	Request to Send
TDMA	Time Division Multiple Access
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity
WLAN	Wireles Local Area Network
XML	eXtensible Markup Language
Yans	Yet Another Network Simulator
SNR	Signal-to-noise ratio
DIFS	Distributed Inter Frame Space

A.1 Přílohy k BP

A.1.1 Na CD-u jsou k dispozici tyto soubory:

zarkovic-bp.pdf

programové-řešení1.txt

programové-řešení2.txt

programové-řešení3.txt

A.1.2 ARF pseudokód

```
Parametry:
int Xu = 10; - počet po sobě jdoucích úspěšných přenosů (byl přijat ACK rámec), které
jsou potřebné pro zvýšení rychlosti.
int Xd = 2; - počet po sobě jdoucích neúspěšných přenosů (nebyl přijat ACK rámec), které
jsou potřebné pro snížení rychlosti.
int uprate = 0; - pomocná proměnná (pokud správně přišel rámec ACK, proměnná je inkrementována)
int downrate = 0; - pomocná proměnná (pokud nepřišel rámec ACK, proměnná je dekrementována)
int phyrate; - přenosová (datová) rychlost
rate_up; - časováč
bool ACK; - ACK rámec

int main() {
if (ACK = true || rate_up = 0){
downrate = 0;
uprate++;

    if(uprate >= Xu){

        phyrate++;
        uprate = 0;
        rate_up = stop;

        }

}else {

uprate = 0;
downrate ++;

    if(downrate >= Xd){

        phyrate--;
        downrate = 0;
        rate_up = start;
}
}
}
```

Obr. 1: ARF pseudokód

A.1.3 AARF pseudokód

```
int success = 0;
int timer = 0;
int recovery = 0;
int success_thershold = 10;
int min_success_threshold = 10;
int max_success_threshold = 50;
int timeout = 15;
int retry_threshold = 4;
double min_success_k = 2;
double timeout_k = 1.5;
bool packet_status;

#define report_recovery_failure() {
   success_threshold = min (success_threshold * success_k, max_success_threshold);
   timeout = max (timeout_k * success_threshold, min_timeout);

}
#define report_failure(){
   success_threshold = min_success_threshold;
   timeout = min_timeout;

}

int send_packet (Packet packet, int rate){

int retry = 0;
Status packet_status;

while (retry < retry_threshold){


packet_status = send_packet (packet, rate);

   if(packet_status == SUCCESS){
        success++;
        retry = 0;
        if ((success == success_threshold || timer == timeout)
               && !is_max_rate (rate)) {
               rate++
               timer = 0;
               success = 0;
               recovery = true;

             } else {
               timer++;
               recovery = false;
             }
              break;

             } else {
               timer++;
               retry++;
               success = 0;
               if (recovery) {
                  timer = 0;
                  if (retry == 1){
                     report_recovery_failure();
                     if (!is_min_rate(rate)) {
                       rate--;
                      }
                    }
                 }else {
                  if(retry == 2
                     || retry == 4
                     || retry == 6
                     || retry == 8
                    || retry == 10){
                  report_failure();
                  if (!is_min_rate()){
                     rate--;
                   }
                  }
                }
                 if (retry >=2){
                  timer = 0;
                }
              }
            }
             return rate;
        }
```

Obr. 2: AARF pseudokód