VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SIMULACE MOBILNÍ AD HOC SÍTĚ V PROSTŘEDÍ NS-3

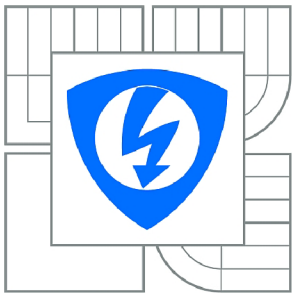
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE LUKÁŠ KOLAJA
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SIMULACE MOBILNÍ AD HOC SÍTĚ V PROSTŘEDÍ NS-3

SIMULATION OF MOBILE AD HOC NETWORK IN NS-3 ENVIRONMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

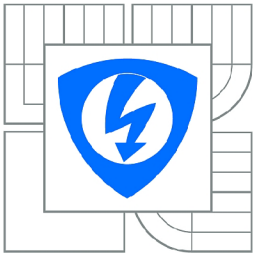
AUTOR PRÁCE — LUKÁŠ KOLAJA

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ HOŠEK, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Lukáš Kolaja	***ID:***	134336
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Simulace mobilní ad hoc sítě v prostředí NS-3

POKYNY PRO VYPRACOVÁNÍ:

V rámci bakalářské práce nejprve nastudujte problematiku mobilních ad hoc sítí (MANET). Zaměřte se zejména na princip komunikace v těchto sítích a nejpoužívanější směrovací protokoly. V rámci praktické části se seznamte se simulačním prostředím NS-3 a vytvořte v něm základní model MANET sítě tvořený minimálně 10 mobilními uzly. Výstup simulace uložte do formátu XML. Následně proveďte analýzu výstupního souboru a vytvořte program pro zpracování a separaci důležitých dat z výstupního souboru. Pro ukládání dat použijte databázový server.

DOPORUČENÁ LITERATURA:

[1] ILYAS, M.: The Handbook of Ad Hoc Wireless Networks. Boca Raton: CRC Press, 2003, ISBN: 0-8493-1332-5.
[2] MOHAPATRA, P., KRISHNAMURTH, S.: Ad Hoc Networks: Technology and Protocols. Boston: Springer Press, 2005, ISBN: 0-387-22689-3.

Termín zadání:	11.2.2013	***Termín odevzdání:***	5.6.2013

Vedoucí práce: Ing. Jiří Hošek, Ph.D.
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cílem této bakalářské práce je teoreticky rozebrat principy bezdrátových mobilních ad hoc sítí. V práci jsou popsány typy směrovacích protokolů jako je např. AODV, DSDV nebo OLSR. Dále jsou popsány vybrané typy modelů pohybu často používané u MANET sítí. Práce se také zabývá tématy jako je využití, bezpečnost a vývoj mobilních ad hoc sítí.
Cílem praktické části je pomocí vytvořené aplikace zpracovat výstupní XML dokument, který byl vygenerován již vytvořenou simulací. Aplikace je psána v jazyce PHP a výstupní data jsou pomocí SQL příkazů touto aplikací ukládána na databázový server, ze kterého lze jednoduše číst získané informace z daného XML dokumentu.

KLÍČOVÁ SLOVA

bezdrátové mobilní ad hoc sítě, MANET, simulace, NS-3, databáze, analýza XML

ABSTRACT

The goal of this bacherlor thesis is theoretically analyze the principles of wireless mobile ad hoc networks. The thesis describes the types of routing protocols such as AODV, DSDV or OLSR. Further describes the types of mobility models which is typically used in MANET. The thesis also deals with issues like use, safety and development of mobile ad hoc networks.
The goal of the practical part is create an application which can parse an XML document that generates simulation that is already created. The application is written in PHP language and output data from this application is inserted to the databse by using SQL commands. From the database is easy to read the data from the original XML document.

KEYWORDS

wireless mobile ad hoc networks, MANET, simulation, NS-3, database, XML analysis

LUKÁŠ, Kolaja *Simulace mobilní ad hoc sítě v prostředí NS-3*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 68 s. Vedoucí práce byl Ing. Jiří Hošek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Simulace mobilní ad hoc sítě v prostředí NS-3“ jsem vypracoval 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 uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom 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)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Jiřímu Hoškovi, Ph.D. a Ing. Pavlu Vajsarovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

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)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V době, kdy převládají prodeje přenosných zařízení, které v drtivé většině používají bezdrátovou komunikaci ať už s vnějším světem nebo mezi sebou, je potřeba těmto požadavkům vyhovět i v místech s neexistující, špatnou či zničenou síťovou infrastrukturou. Bezdrátové technologie však neslouží pouze ke shlédnutí multimediálního obsahu či čtení zpráv kdekoliv a kdykoliv, ale také pro životně důležité účely jako je komunikace záchranných složek, bezpečnostních systémů či k rychlému nasazení při válečných konfliktech.

Takové možnosti nám nabízí tzv. mobilní ad-hoc sítě (Mobile ad-hoc Networks, dále jen MANET), které nepotřebují ke svému provozu existující infrastrukturu, dovedou se samy nakonfigurovat, dynamicky se přizpůsobovat při změnách v síti a pokrývat rozsáhlá území. Typickou ukázku MANET sítě představuje obrázek 1.1 V tomto síťovém modelu jsou přenosná zařízení samy součástí sítě a musí spolupracovat k zajištění její funkčnosti, směrování v rámci sítě a její bezpečnosti.

Tato práce se v teoretické části zabývá principem komunikace MANET sítí, přehledem směrovacích protokolů a modelů pohybu v nich používaných. V praktické části je zaměřena na zpracování výsledků již nakonfigurované simulace. Výstupní data ze simulace se pomocí vytvořené aplikace roztřídí a uloží do databáze. Z hodnot získaných z výstupního souboru simulace jsou také vypočítány některé parametry simulované sítě.

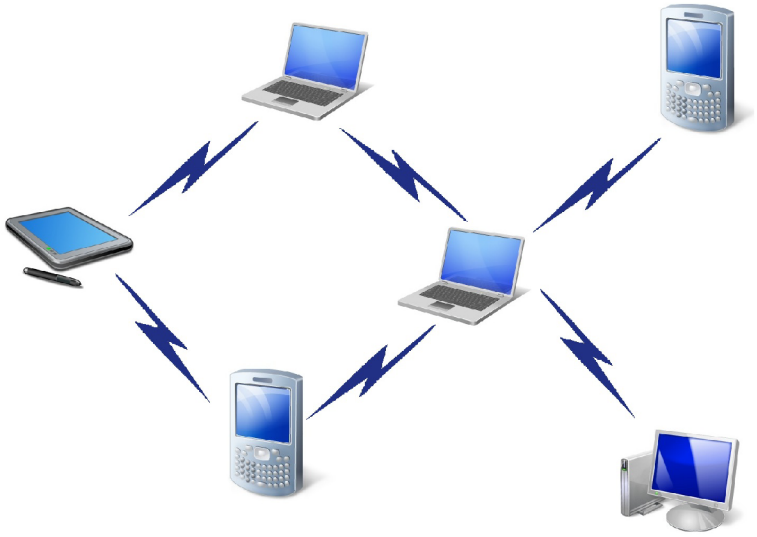
1 MANET SÍTĚ

1.1 Mobilní bezdrátové sítě

Počátky bezdrátového a mobilního internetového připojení sahá až k počátkům samotného zrodu internetu, který zpočátku sloužil výhradně k armádním účelům. Potenciál v takovém typu komunikací byl na pohled zřejmý a tak se v 70. Letech zrodil projekt DARPA Packet Radio, což se dá nazvat jako první bezdrátová ad-hoc síť. Jedná se o technologii pro přenos informací bez existující či nedostatečné infrastruktury (rozvojové země, válečné konflikty, místo přírodní katastrofy apod.) nebo kde by výstavba nové infrastruktury byla příliš nákladná a komplikovaná.

Dříve byl takový typ spojení určen pouze pro speciální nebo improvizované účely pro konkrétní aplikace, což vlastně vychází z překladu latinského „ad-hoc“-pro tento účel. Dnes se již však hojně využívá pro veřejné účely, jako jsou videokonference, vysílání streamovaného videa, sběr informací z bezpečnostních čidel, použití v letecké a lodní dopravě, záchranné složky atd. Konkrétní uživatelé však mohou být mobilní a prostředí se může velmi rychle změnit. Kvůli tomu musí být ad-hoc sítě být schopny na tyto změny dynamicky reagovat a přizpůsobit se, aniž by koncový uživatel zaznamenal výpadky v komunikaci či úplné odpojení od zbytku sítě.

Aby bylo možné vytvořit takovou síť, je nutné splnit několik požadavků jako je například automatická konfigurace uzlů (adresy, směrování,. . .), efektivní skupinová komunikace v reálném čase v závislosti na pohybu uzlů, nároky na pokrytí požadovaného území, energetická úspornost a nízké nároky na výkon. Realizací návrhu mobilních ad-hoc sítí se zabývají výzkumní pracovníci již od 70. let, ale až v poslední dekádě díky velkému rozvoji a pokrokům v radiokomunikaci byly zaznamenány velké úspěchy jak v civilní, tak ve vojenské sféře. Ve vojenské sféře můžeme jmenovat například použití bezpilotních letounů, které jsou schopny mapovat a monitorovat rozsáhlá území a předávat si informace jak s velícím střediskem, tak navzájem mezi sebou.[5]



Obr. 1.1: Příklad typického využití technologie MANET.

1.2 Charakteristické vlastnosti MANET sítí

MANET je systém bezdrátových mobilních zařízení jako jsou notebooky, tablety, telefonní přístroje, vysílačky, dálkově ovládané systémy, senzory různých veličin apod. které budeme označovat jako uzly. MANET může pracovat samostatně jako uzavřená síť nebo může být propojen s dalšími sítěmi pomocí bran. Uzly jsou vybaveny bezdrátovými přijímači a vysílači používající antény v závislosti na typu použití (všesměrové, směrové, nastavitelné nebo kombinované).[1]

MANET má několik charakteristických vlastností:
Dynamická topologie - uzly se v rámci sítě mohou volně pohybovat a tak se topologie sítě může měnit velmi rychle v různých intervalech a síť musí být schopná na tyto změny velmi rychle reagovat např. změnou směrovací tabulky.
Omezená šířka pásma a proměnné přenosové rychlosti spojů - bezdrátové spojení má nižší přenosové rychlosti než spojení realizované kabely a také jsou mnohem citlivější na vnější vlivy, jako jsou překážky, rušení od dalších signálů či šum. To může ve výsledku způsobit zahlcení sítě a její nefunkčnost.
Energetická náročnost - velmi často jsou uzly v MANET síti závislé na baterii či jiné vyčerpatelné energii a proto je nutné optimalizovat komunikační systém, aby co nejméně zatěžoval tyto zdroje.
Zabezpečení komunikace - bezdrátová komunikace je obecně mnohem náchylnější k útokům zvenčí, odposlechu, falšování zpráv než spojení kabelem. Proto se používají stávající technologie jako je šifrování, ověřovací klíče, hesla apod.[1]

1.3 Směrovací procesy a protokoly

Dvě hlavní rodiny směrovacích protokolů pro MANET jsou reaktivní a proaktivní protokoly.

Reaktivní protokoly vyhledávají a sestavují trasy pro pakety až je cesta vyžadována. V angličtině se označují jako „on-demand“ protokoly, v českém překladu „na požádání“.
Výhodou těchto reaktivních směrovacích protokolů je tvorba cest podle potřeb reálné komunikace, takže tolik nezvyšují zatížení sítě svojí režií. Jakmile komunikace skončí, komunikační trasu nadále neudržují.
Nevýhodou reaktivních protokolů je fakt, že jakmile je trasa vyžadována, tak může být síť najednou zaplavena požadavky na vyhledání trasy a data, která takovou cestu potřebují, musí čekat na ustanovení této trasy. Mezi reaktivní protokoly patří například AODV (Ad-hoc On Demand Distance Vector) a DSR (Dynamic Source Routing).[2]

Druhým typem směrovacích protokolů v MANET sítích jsou protokoly proaktivní, nazývané také jako protokoly řízené pomocí tabulek. Na rozdíl od reaktivních směrovacích protokolů stále aktivně sledují stav sítě, změny v topologii a stav jednotlivých komunikačních tras.
Výhodou pak jsou již připravené cesty pro data, takže nemusí čekat, než je spojení sestaveno. Z toho vyplývá, že tyto směrovací protokoly najdou uplatnění v sítích s velkou komunikační režií.
Nevýhodou ale jsou režijní náklady na sledování stavu sítě. Do proaktivních směrovacích protokolů patří například OLSR (Optimized Link State Routing) a DSDV (Destination-Sequenced Distance Vector routing).[2]

Existují však také tzv. hybridní směrovací protokoly, které kombinují funkce jak proaktivních, tak reaktivních směrovacích protokolů. Příkladem takového protokolu může být proaktivní vyhledávání tras v síti, které se následně udržují reaktivním způsobem. Příkladem hybridních směrovacích protokolů je Hybrid Wireless Mesh Protocol, na jehož základě funguje síť laptopů projektu „One Laptop per Child“.[4]

1.3.1 Ad-hoc On-demand Distance Vector

Vznikl jako jeden z prvních směrovacích protokolů používaných v mobilních ad-hoc sítích. Mezi dvěma mobilními uzly, které si potřebují sestavit spojení AODV algoritmus umožňuje dynamicky směrovat pakety přes více uzlů. AODV také umožňuje mobilním uzlům rychle najít trasy k novým cílům a mezi uzly, které spolu už nekomunikují, nadále neudržuje spojení. V případě ztráty linky mezi uzly, které se vyskytují na používané cestě je směrovací protokol AODV schopen rychle reagovat a sestaví novou trasu.
Aby se zabránilo zacyklení paketů v síťové smyčce, používá AODV protokol pro každou cestu sekvenční čísla. Toto číslo vytváří cílový uzel a je přiloženo ke každé směrovací zprávě, kterou uzel posílá jiným uzlům, které žádají o ustanovení cesty.[4]

1.3.2 Destination-Sequenced Distance Vector

DSDV je prokativní směrovací protokol, který vznikl jako modifikace Bellman-Fordova směrovacího algoritmu. DSDV protokol ke každé směrovací zprávě přiloží podobně jako AODV sekvenční číslo pro jednotlivé trasy a zabrání tak vzniku síťových smyček. Směrovací tabulky se udržují v každém uzlu, který podle ní vybírá nejvhodnější trasy k cílovému uzlu. Každý uzel periodicky zasílá svým sousedům zprávy o změně směrovací tabulky, která se s pohybem uzlů v síti může velmi rychle měnit.
Výhodou DSDV je ošetření smyček v síti pomocí sekvenčních čísel. DSDV tabulky obsahují pouze záznamy o nejlepších cestách, takže tabulka neklade takové nároky na paměť.
Kvůli pravidelnému rozesílání směrovacích zpráv potřebuje DSDV protokol část šířky pásma, což se dá považovat jako nevýhoda. V případě rozsáhlejších sítí však může tato režie způsobit i její zahlcení.[3]

1.3.3 Optimized Link State Routing

Jedná se o proaktivní směrovací protokol, který má trasy pro pakety již předem připraveny, takže samotnou komunikaci nezdržuje sestavováním cesty jak je tomu u reaktivních protokolů.
Protokol OLSR nezávisí na žádném centrálním uzlu a nevyžaduje spolehlivý přenos řídících zpráv. Každý uzel zasílá tyto zprávy periodicky, takže ztráta jistého množství řídících zpráv není při provozu sítě zásadně omezující. Vlastností protokolu OLSR je využití tzv. „multipoint relays“ (dále jen MPR). To znamená, že každý uzel vybírá ze svých na 1 skok vzdálených sousedů takové uzly, které budou jeho zprávy přeposílat dál do sítě. Jednotlivé uzly a jejich počet zajišťuje podmínka, že množina těchto zvolených uzlů musí umožňovat komunikaci se všemi sousedy, kteří jsou od něj vzdálení na 2 skoky. Takto vybrané uzly se označují jako MPRN (MPR množina uzlu N).
Jednotlivé uzly si zároveň udržují svou množinu uzlů, které ho aktuálně považují za svoje MPR, tzv. „množinu MPR voličů“. Přijde-li na uzel broadcast zpráva, která by se měla šířit celou sítí a přišla od člena „množiny MPR voličů“, tak ji uzel lokálně zpracuje a zašle dál do sítě. Přijde-li na uzel broadcast zpráva, kterou už v minulosti zpracoval, tak tuto zprávu dál nezpracovává ani dál neodesílá.[4]

1.4 Pohyb uzlů v MANET síti

Pohyb uzlů je klíčovým prvkem mobilních ad-hoc sítí a musí být navržen a realizován v závislosti na požadavcích vytvářené mobilní sítě. Faktem je, že systém pohybu uzlů v reálném čase může být velmi složitý a závisí na úkolech jednotlivých uzlů, které jsou součástí celého autonomního systému. Čím více je složitější model pohybu, tak tím náročnější je ho vytvořit, protože je nutné zahrnout velké množství informací do daného modelu. Pomocí virtuálních prostředí a simulací můžeme zkoumat závislosti jednotlivých modelů pohybu a jejich dopad na síť, ale i tak nemusí být jednoduché najít správné řešení, protože nelze zahrnout veškeré faktory, které se při reálném provozu mohou vyskytnout.

Pohyb uzlů v síti má také za následek změnu síťové topologie, která se mění s časem a tak musí být síť schopna dynamicky reagovat a opět se s těmito změnami nastavit aby byla plně funkční a zajišťovala potřebné služby. A proto je síť a použitý aplikační protokol z mobilních ad-hoc sítí značně ovlivněn frekvencí, s jakou se síťová topologie mění, a také se mohou výkony mobilní sítě výrazně lišit při použití různých modelů pohybu. Tím že se neustále mění různé parametry sítě (vzdálenost uzlů, síla signálu, počet uzlů v dosahu jednotlivého uzlu apod.) může výkon mobilní sítě značně kolísat.

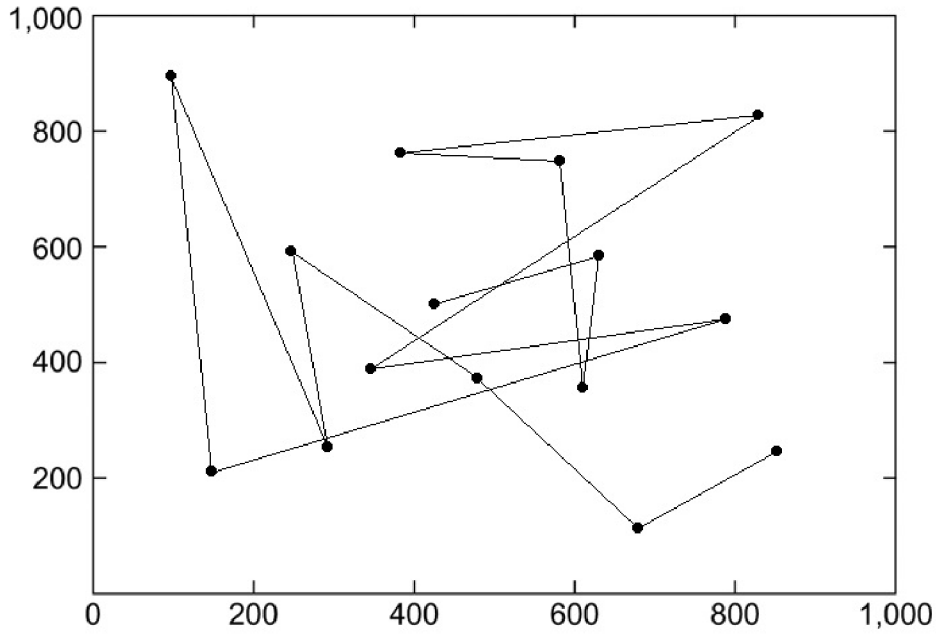
Pro zvolení modelu pohybu je nutné znát komunikační vzor mezi uzly v celé síti, která právě ovlivňuje výslednou funkčnost a výkon sítě. Například MANET obsahuje dvě či více skupin používající stejný protokol pro komunikaci jak uvnitř skupiny, tak mezi ostatními skupinami. Výkon použitého protokolu ale může být značně odlišný při změně v komunikaci mezi skupinami, než když se změní komunikace uvnitř skupiny. Zvolený model pohybu musí být co nejvíce podobný tomu jaký očekáváme při reálném nasazení, a nároky na výkon sítě musí být při výběru modelu náležitě zhodnoceny. Je zřejmé, že značnou část času zabere právě zjišťování a testování, aby se zjistilo, jaký bude model pohybu ve skutečnosti nejlépe vyhovovat požadavkům vytvářeného scénáře.

Každý uzel v mobilní ad-hoc síti je považován za samostatně fungujícího a vzory jejich náhodných pohybů musí být analyzovány pro měření dopadu na výkon při změnách topologie sítě. Navíc MANET je obvykle síť s omezenými komunikačními zdroji, jako je zdroj elektrické energie, výpočetní výkon a paměť a to také musí být bráno v potaz při jeho návrhu. Během let se mnoho modelů pohybu používalo právě pro zjištění parametrů dané mobilní ad-hoc sítě. Mnoho modelů pohybu je navrženo tak, aby bylo jeho reálné použití nejlepší pro konkrétní použití MANET sítě. Statistické vlastnosti jednotlivých modelů jsou analyzovány navrhováním různých metrik pohybů a studiem vlivů modelů pohybu při použití různých síťových protokolů včetně směrování, nalezení trasy a jednotlivých peer-to-peer aplikací.[6]

1.4.1 Modely pohybu

Metoda náhodně zvoleného cíle

Tento model pohybu je velmi hojně používán v mobilních ad-hoc simulacích. Tento model je jednoduchý a přímočarý náhodný model. V tomto modelu se mobilní uzly pohybují v uzavřené ploše z jejich současné pozice do nové pozice náhodně zvoleným směrem, rychlostí pohybu a množstvím času, který čeká, jakmile dosáhne svého cíle, než se začne opět pohybovat. Jak může takový pohyb v simulaci vypadat znázorňuje obr. 1.2. V cílovém bodě se uzel zastaví na nějaký čas podle náhodné proměnné a celý proces se po vypršení zase opakuje - uzel si vybere nový cíl cesty, rychlost a čas „zastávky“. Pohyb uzlu z výchozí polohy do jejího cíle je definován jako jeden pohyb epochy, perioda přesunu nebo přechodný čas. Překonaná vzdálenost mezi výchozí polohou a cílovým bodem je definován jako přechodná délka. Cílové body jsou rovnoměrně náhodně vybrány v systémové oblasti.[6]



Obr. 1.2: Vzor cesty uzlu při použití metody náhodně zvoleného cíle.

Metoda náhodně zvolené trasy

Individuální model pohybu a jeho princip pramení z Brownova pohybu, který jako první popsal matematicky Albert Einstein.
Brownův pohyb je náhodný pohyb mikroskopických částic v kapalném nebo plynném médiu a je limitou náhodné trasy. To znamená, že když vezmeme náhodnou trasu s velmi malými kroky tak se dostaneme přibližně k Brownovu pohybu. Tento pohyb může být i ve vícerozměrném prostoru. V simulaci pak tento pohyb vypadá velmi podobně jako u metody náhodně zvoleného cíle.[6]

Charakteristické vlastnosti:

- uzly mění svoji rychlost a směr každý časový interval a v cílové oblasti není žádná pauza
- rychlost uzlu může být náhodně zvolena z daného rozsahu $[v_{min}, v_{max}]$
- podobně jako rychlost, tak směr je náhodně zvolen z rozsahu $[0, 2\pi]$
- může mít deklarované hranice, přes které nemůže projít
- může mít také pohlcující hranice, ze kterých se uzel už nedostane ven
- ke každému pohybu dochází buď ve stálý časový interval nebo při konstantní uražené vzdálenosti nebo po určitém počtu dokončených pohybů s určitou vzdáleností, kde se na konci vypočítá nový směr a rychlost
- každý mobilní uzel se odrazí od hranice simulace s úhlem, který se určí příchozím směrem, kterým narazil na hranici
- model je bez paměti, protože nikde nezachovává parametry jako je rychlost a směr a budoucí trasy na těchto parametrech nezávisí [6]

Metoda zeměpisně omezené mobility

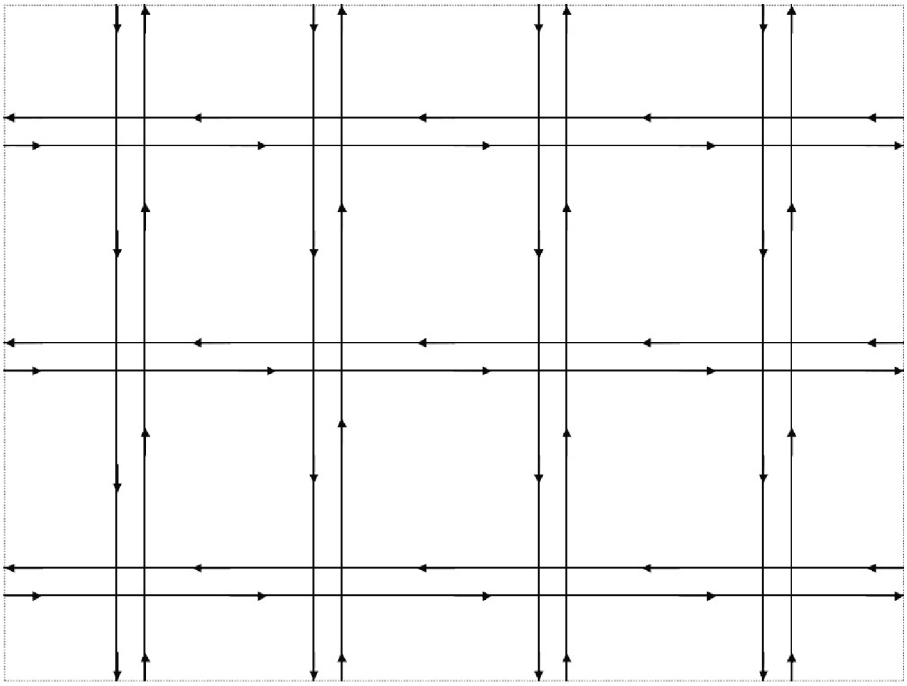
Nevýhody metod náhodně zvoleného cíle nebo trasy je zřejmé z jeho nerealistického modelování např. městské oblasti či dálniční infrastruktury. Chování mobilních uzlů je ovlivněno těmito zeměpisnými omezeními v reálných situacích, protože např. člověk jdoucí po ulici nemůže libovolně procházet budovami, ale musí se pohybovat na určitém geografickém podkladu, jako jsou mapy či satelitní navigace, které v tomto modelu slouží jako omezení pohybu mobilních uzlů a uzly se mohou pohybovat náhodným způsobem pouze na těchto předdefinovaných cestách. Ve skutečnosti mapy zprostředkovávají souhrnný zeměpisný pohled mobilní sítě a definuje tak jeho topologii.
Metoda zeměpisně omezené mobility zahrnuje několik skupin podle způsobu použití jako je pohyb vozidel (VANET- Vehicular Ad-hoc Networks), pohyb překážek, Voronoivův pohyb apod. Zde si ukážeme tzv. Manhattan model, která patří do skupiny pohybu vozidel.[6]

Model Manhattan

Tento model emuluje vzor pohybu mobilních uzlů na ulicích definovaných podle map a jako topologie je použitá mřížka (obr. 1.3), kterou cesty ve městech vytváří (toto pravidlo však neplatí celosvětově, nejvíce lze uplatnit v Severní Americe).
Model Manhattan využívá pravděpodobnostní přístup k výběru pohybu uzlů, jelikož v každém křížení mobilní uzel rozhodne, jestli pojede dál rovně (pravděpodobnost P1) nebo zahne doleva či doprava (pravděpodobnost P2). Sledování rychlosti uzlů je podobné, jako je u dálničního modelu, kde všechny mobilní uzly cestující ve stejném pruhu mají stejnou rychlost. Tento model však není vhodný pokud jde o starší města jako je např. Londýn či Paříž, protože cesty v těchto městech nutně netvoří mřížku, cesty se neprotínají pod 90° úhlem a mají různé tvary a velikosti.[7]

Model Manhattan lze popsat těmito parametry:

- průměrná rychlost
- minimální rychlost
- pravděpodobnost směny rychlosti v závislosti na pozici
- pravděpodobnost změny směru v křížení cesty [7]



Obr. 1.3: Manhattan model. Linky reprezentují jeden pruh na silnici.

2 ZPRACOVÁNÍ VÝSTUPU SIMULACE

2.1 NS-3 síťový simulátor

NS-3 simulátor sítě je zaměřen především na výzkum a vzdělávací použití a je univerzální simulační nástroj, který může být použit k simulaci různých typů sítí.
Jádro NS-3 je sada knihoven, které poskytují široký rámec nástrojů pro vývoj síťových simulací.[9]
Kromě toho také balíček NS-3 obsahuje několik modelů běžných síťových protokolů, které můžeme pomocí funkce jednoduše použít v simulacích. Díky tomu že simulátor NS-3 je populární, volně šiřitelný a jeho obsah lze editovat, tak jsou v něm implementovány i modely, které jsou momentálně v raném vývoji. Tyto modely nejsou omezené pouze pro síťové protokoly, ale obsahuje například modely pohybu, šíření ztrát v sítí nebo modely spotřeby energie.[8]

K získání výsledků simulace existují v modelu simulátoru záchytné systémy a díky nim můžeme poté přistupovat k výsledkům simulace, tato funkce se nazývá trasování.
Jádro simulátoru dále také obsahuje logovací záznamy či testovací jednotku, která se používá pro kontrolu stávajícího kódu a detekci chyb. Jednotka také testuje, jestli se model bude chovat očekávaným způsobem.[8]

NS-3 je psán výhradně v jazyce C++ (na rozdíl od jeho předchůdce NS-2, který používal kombinaci C++ a OTcl), ale nabízí také možnosti psaní skriptů a simulací v Pythonu. Umožňuje také běh v reálném čase při připojení k virtuálním jednotkám běžících na skutečných zařízeních. NS-3 má velkou podporu využití pod operačním systémem Linux, ale existují i metody použití pod systémem MS Windows.[8]

2.2 Parametry a popis simulace

Síť, jejímž výstupním souborem je zpracovávaný XML soubor, byla sestavena z 10 pohyblivých uzlů, mezi kterými byl nakonfigurován směrovací protokol DSDV což je proaktivní směrovací protokol, který periodicky rozesílá informace o směrovacích tabulkách, které jsou navíc doplněny o sekvenční číslo, aby se zabránilo smyčkám. DSDV je vhodné k použití pro sítě s malým počtem uzlů, nízkou mobilitou a kde zároveň probíhá více datových relací.
Modelem pohybu uzlů je v simulaci nakonfigurovaná Metoda náhodně zvolené trasy (více v kap. Jednotlivé modely pohybu), takže můžeme očekávat nízký pohyb uzlů pramenící z principu této metody pohybu. V simulaci probíhají dva typy provozu- VoIP a FTP. VoIP služba využívá protokol UDP což je nespojově orientovaný síťový protokol, který se využívá především pro služby pracující v reálném čase (telefonní hovory, videokonference apod.). Služba VoIP tímto klade velké nároky na zpoždění a na kolísání zpoždění paketů.
Naopak služba FTP využívá spojově orientovaný protokol TCP, který zajišťuje spolehlivé doručení paketů avšak za cenu další režie potřebné k ověřování správnosti paketů a sestavování spojení. FTP protokol se používá například k přenosu většího množství dat, nebo pro přístup k souborům v síti.
Pro optimalizaci provozu v síti je nakonfigurována v simulaci podpora kvality služeb, která např. rozdělí šířku pásma a řídí datové toky jednotlivých spojení tak, aby bylo možné používat více služeb zároveň, aniž by docházelo ke snížení kvality této služby (např. výpadky hovoru při stahování dat z FTP serveru).
Velikost FTP paketu byla 614 bajtů a velikost VoIP paketu byla 566 bajtů.
Šířka pásma jedné linky byla nastavena na 1 Mb.
Délka simulace byla 20 sekund, avšak FTP provoz probíhal pouze 3,5 sekundy a VoIP provoz trval 1 sekundu.

Výstupem simulace zpracovávané v této práci je XML soubor, ve kterém je zaznamenán např. pohyb uzlů a jejich počáteční pozice, údaje o topologii a záznamy vyslaných paketů.

2.2.1 Ukázka tvorby simulace v NS-3

Při tvorbě simulace jsou nejdříve načteny veškeré moduly, které budou v rámci simulace použity, jako jsou například:

```
#include "ns3/core-module.h"
#include"ns3/point-to-point-module.h"
#include"ns3/mobility-module.h"
```

Dále se vytváří samotné mobilní uzly v počtu, který deklarujeme.

```
NodeContainer wifiStaNodes;
wifiStaNodes.Create (5);
```

Mezi uzly je potřeba pro funkční komunikaci nadefinovat směrovací protokol a jeho parametry jako je například interval zasílání směrovacích tabulek.

```
DsdvHelper dsdv;
dsdv.Set("PeriodicUpdateInterval", TimeValue (Seconds(3)));
```

Implementace modelu pohybu a jeho hranic a následná instalace do uzlů je deklarována tímto způsobem:

```
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
"Bounds", RectangleValue (Rectangle (-30, 30, -30, 30)));
mobility.Install (wifiStaNodes);
```

Uzlům je také nutné přiřadit IP adresy.

```
Ipv4AddressHelper address;
address.SetBase ("192.168.1.0", "255.255.255.0");
address.Assign (staDevices);
```

Samostné spuštění a ukončení simulace po veškeré konfiguraci a její délka se provádí příkazy

```
Simulator::Run ();
Simulator::Stop(Seconds(40));
Simulator::Destroy ();
```

Tato ukázka znázorňuje pouze základní záležitosti, co se vytváření simulace týče. Dále je možné mezi uzly definovat provoz, implementovat a nakonfigurovat kvalitu služeb, logovací záznamy apod. Pro vyhodnocení simulace nebo k jejímu zobrazení průběhu v animátoru se využívá trasování, které po dobu běhu simulace zachycuje informace o uzlech a komunikaci v síti, a po ukončení jako výstup vytvoří soubor, který lze následně zpracovávat různými způsoby.

2.3 Struktura XML dokumentu

Po otevření XML souboru (je doporučeno použít některý z alternativních programů k editaci textových souborů např. PSPad) lze vidět strukturu podobnou značkovacímu jazyku HTML. Hlavní značky v celém tomto záznamu simulace jsou *topology*, *wpacket* a *packet*. Ve značce *topology* je navíc značka se záznamem jednotlivých uzlů- *node*. Ve značce *wpacket* jsou vnořeny značky *rx* a *meta*. Ve značce *packet* je ještě značka *rx*.
Každá tato značka obsahuje jednotlivé atributy, ve kterých jsou již jednotlivě zaznamenané údaje o simulaci. Takový soubor se dá vytřídit metodou parsování (viz kap. 2.4 Parsování XML souboru).
V ukázce jsou obsaženy příklady se všemi jednotlivými značkami.

Ukázka struktury XML dokumentu:

```
<topology minX = "-3.1" minY = "-2.9" maxX = "29.9"...>
<node id = "0" descr="STA" locX = "10" locY = "10" />
<node id = "1" descr="STA" locX = "15" locY = "10" />
</topology>

<packet fromId="0" fbTx="0" lbTx="0" aux="DummyPktIgnore"
range="1.5568">
<rx toId="0" fbRx="0" lbRx="0"/>
</packet>

<wpacket fromId="8" fbTx="0.000223" lbTx="0.000223" ...>
<rx toId="5" fbRx="0.000223006" lbRx="0.000223006"/>
<meta info="ns3::WifiMacHeader(QOSDATA)..." />
</wpacket>
```

2.3.1 Význam značek a atributů

Topology - v této značce jsou uloženy informace o topologii sítě, kterou tvoří tyto atributy:
minX, *minY*, *maxX*, *maxY* - souřadnice rohů plošného obrazce (v tomto případě obdélníku), který ohraničuje prostor simulace.

Node - vnořen uvnitř značky *topology*.
locX a *locY* - údaje o souřadnicích jednotlivých uzlů.
Id - identifikační číslo uzlu.

Packet - neměl by se v simulaci vyskytovat, jelikož poskytuje záznam paketů poslaných po kabelovém spojení. V tomto případě je u všech *packet* záznamů vidět že odesilatelem a příjemcem je vždy uzel s identifikačním číslem 0 a z jeho atributu *aux=„DummyPktIgnoreThis“* můžeme odvodit, že jde o prázdný paket, který nenese žádnou informaci.
fromId - identifikační číslo odesílatele
fbTx (first bit transmit time) a *lbTx* (last bit transmit time) - údaj o čase, který uplynul od začátku simulace, ve kterém se začal vysílat první a poslední bit daného paketu.

Uvnitř značky *packet* se nachází značka *rx*, která obsahuje následující atributy:
toId - identifikační číslo uzlu příjemce paketu.
fbRx (first bit receive time) a *lbRx* (last bit receive time) - časový údaj o příjmu prvního a posledního bitu stejně jako v případě *fbTx* a *lbTx*.

WPacket - záznam paketů které byly poslány bezdrátově. Obsahuje stejně jako značka *packet* informaci o odesílateli paketu a časové záznamy o odeslání a příjmu prvního a posledního bitu paketu.
range - informace o vzdálenosti uzlů, mezi kterými probíhala komunikace.

WPacket obsahuje také značku *meta*, která obsahuje jediný atribut - *info*.
info obsahuje dlouhý řetězec, ze kterého lze například vyčíst typ síťového protokolu, hlavičku bezdrátové MAC vrstvy nebo směrovacího a IP protokolu.

2.3.2 Typy přenášených paketů

Během simulace je v rámci komunikace přenášeno několik typů paketů, v XML souboru definováno jako záznamová značka *wpacket* (více o značce *wpacket* v kapitole 2.3.1 Význam značek a atributů). Některé typy přímo souvisí s typem přenášeného provozu, další zase slouží k přenosu informací o směrování v síti. Tyto záznamy se liší jen informacemi v řetězci atributu *meta_info*.

Paket směrovacího protokolu DSDV:

```
<meta info="ns3::WifiMacHeader (QOSDATA )
ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP
Default ECN Not-ECT ttl 1 id 0 protocol 17 offset (bytes) 0
flags [none] length: 40 10.1.3.9 > 10.1.3.255)
ns3::UdpHeader (length: 20 269 > 269)
ns3::dsdv::DsdvHeader (DestinationIpv4: 10.1.3.9 Hopcount: 1
SequenceNumber: 2) ns3::WifiMacTrailer ()"/>
```

Z řetězce v *meta_info* je vidět výpis záznamů jednotlivých několika hlaviček daného paketu jako je WifiMacHeader, LlcSnapHeader, Ipv4Header, UdpHeader a DsdvHeader. Z MacHeaderu lze zjistit typ paketu pro danou hlavičku, z LlcSnapHeaderu typ jeho struktury a specifikace typu paketu. Konkrétně type 0x800 je ethertype což značí, že je použit IP protokol přes Ethernet a 802.2 LLC/SNAP zapouzdření. Tato podvrstva linkové vrstvy zajišťuje navazování, správu a ukončování logického spojení, řízení bezpečného přenosu dat mezi dvěma uzly sítě a identifikaci vyšších protokolů. IPv4Header obsahuje důležitou informaci jako je typ protokolu, délku hlavičky a IP adresy zdroje a cíle. Z UdpHeaderu lze vyčíst jeho délku. V případě DsdvHeaderu lze zjistit informace o IP adrese cíle, počtu přeskoků a sekvenční číslo. Hlaviček DSDV protokolu může být v jednom paketu několik.

Pakety VoIP přenosu:

```
 <meta info="ns3::WifiMacHeader (QOSDATA )
ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP
Default ECN Not-ECT ttl 64 id 21 protocol 17 offset (bytes) 0
flags [none] length: 528 10.1.3.1 > 10.1.3.6)
ns3::UdpHeader (length: 508 49156 > 5060) Payload (size=500)
ns3::WifiMacTrailer ()"/>
```

U VoIP jsou stejně jako v předchozím případě hlavičky WifiMacHeader, LlcSnapHeader, Ipv4Header a UdpHeader. Informace v těchto hlavičkách obsahují stejné typy údajů jako u DSDV paketu. Navíc je zde atribut Payload, který značí velikost paketu datové části.

Potvrzovací rámec síťové podvrstvy MAC:

```
<meta info="ns3::WifiMacHeader (CTL_ACK Duration/ID=0us,
RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()"/>
```

Jedná se o potvrzovací rámec podvrstvy MAC v linkové vrstvě, který je vyslán přijímačem zpět zdroji po každém úspěšném přijetí FTP nebo VoIP paketu. Kromě hlavičky WifiMacHeader, kde je uvedeno, že se jedná právě o ACK rámec, neobsahuje žádné další užitečné informace.

Pakety FTP přenosu:

```
<meta info="ns3::WifiMacHeader (QOSDATA )
ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP
Default ECN Not-ECT ttl 64 id 283 protocol 6 offset (bytes) 0
flags [none] length: 576 10.1.3.1 > 10.1.3.6)
ns3::TcpHeader (49153 > 20 [ ACK ] Seq=32697 Ack=1 Win=65535)
Payload Fragment [496:1032] ns3::WifiMacTrailer ()"/>
```

Hlavičky WifiMacHeader, LlcSnapHeader, Ipv4Header jsou opět stejné jako v předchozích případech, včetně obsažených informací. Nově je zde hlavička TcpHeader, která obsahuje sekvenční číslo, potvrzovací číslo (ack) a velikost okna (win). Dále je zde položka Payload Fragment, která značí část paketu, který byl rozdělen na části LFI mechanizmem. Položek Payload Fragment se může v jednom paketu vyskytovat i více.

Potvrzovací paket FTP_ACK:

```
 <meta info="ns3::WifiMacHeader (QOSDATA )
ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP
Default ECN Not-ECT ttl 64 id 52 protocol 6 offset (bytes) 0
flags [none] length: 40 10.1.3.6 > 10.1.3.1)
ns3::TcpHeader (20 > 49153 [ ACK ] Seq=1 Ack=33769 Win=65535)
ns3::WifiMacTrailer ()"/>
```

Zcela stejný případ jako u FTP paketu, jen s rozdílem že zde není položka Payload Fragment. Jedná se o potvrzovací paket vyplývající z podstaty spojově orientovaného protokolu, který FTP provoz používá.

ARP (Address Resolution Protocol) pakety:

Request (žádost):

```
<meta info="ns3::WifiMacHeader (QOSDATA )
ns3::LlcSnapHeader (type 0x806) ns3::ArpHeader (request
source mac: 00-06-00:00:00:00:00:01 source ipv4: 10.1.3.1
dest ipv4: 10.1.3.6) ns3::WifiMacTrailer ()"/>
```

Reply (odpověď):

```
<meta info="ns3::WifiMacHeader (QOSDATA )
ns3::LlcSnapHeader (type 0x806) ns3::ArpHeader (reply
source mac: 00-06-00:00:00:00:00:06 source ipv4: 10.1.3.6
dest mac: 00-06-00:00:00:00:00:01 dest ipv4: 10.1.3.1)
ns3::WifiMacTrailer ()"/>
```

ARP protokol slouží pro nalezení MAC adresy zařízení podle jeho cílové IP adresy. Zdrojová stanice vyšle požadavek (request) ke zjištění MAC adresy cílové stanice a odešle ji jako broadcast. Všechny stanice v segmentu, které mají požadovanou IP adresu tuto žádost přijmou, sestaví odpověď (reply) do které vloží svoji MAC adresu a odešle ji jako unicast na zdrojovou stanici.

Z ARP hlavičky lze získat informace v případě žádosti o MAC a IP adrese zdroje a IP adrese cíle. V případě odpovědi je možné získat i MAC adresu cíle.

2.4 Parsování XML dokumentu

Při tvorbě programů, simulací apod. je výstupem soubor, jehož obsahem je jeden řetězec, ve kterém je zakódované velké množství informací uvnitř značek, které potřebujeme dostat do proměnných a zobrazit je tak, aby se uložené informace daly jednoduše číst. Metoda, která tyto informace dokáže zpracovat a uzpůsobit je tak, aby měl uživatel v informacích přehled, se nazývá parsování.
Existují 2 základní principy parsování dokumentu - DOM (Document Object Model) a SAX (Simple API for XML). DOM parser, jako je např. SimpleXML použitý v této práci, nejdříve načte celý dokument do paměti a poté přistupuje k jednotlivým prvkům a může se k nim během procesu vracet a modifikovat záznamy v dokumentu. Mezitím SAX načítá jednotlivé značky postupně od shora dolů, takže nemá takové nároky na paměť, ale oproti DOM parseru neumí modifikovat obsah XML dokumentu.[10]

Parsování umožňují provádět různé programovací jazyky. V této práci byl zvolen programovací jazyk PHP5 díky jeho snadné manipulaci právě s takovými soubory a následnému ukládání informací do databáze. Metody parsování jsou již v knihovnách PHP5 implementovány.

Ke psaní kódu PHP byl použit program PSPad v. 4.5.7. Dále byl použit balíček nástrojů EasyPHP v. 12.1 obsahující webový server Apache v. 2.4.2, PHP knihovny v. 5.4.6 a lokální MySQL v. 5.5.27 databázi.

2.4.1 SimpleXML

Struktura XML dokumentů je podobně jako HTML tvořena párovými značkami, ve kterých jsou jednotlivé atributy. Ve značce může být vnořená další značka a spolu tak tvoří stromovou strukturu celého dokumentu. Knihovna SimpleXML využívá vlastnosti DOM parsování, takže nejdříve načte celý XML dokument do paměti a objekty uloží jako strukturu. Názvy objektů pak mají stejný název jako značky zpracovávaného dokumentu. Z takových objektů poté můžeme vyčíst jednotlivé informace, které obsahují.[10]

Funkce `mysql error();` je v aplikaci používaná pro jednodušší vyhledávání a detekci chyb, které se mohou při běhu aplikace vyskytnout. Funkce při chybě zobrazí příslušnou chybovou zprávu.

```
$xml = simplexml_load_file("manetrouting.xml");
```

Tímto příkazem je celý XML soubor načten a uložen ve stromu, kde ho lze následně dále zpracovávat. Dále cyklem

```
foreach($xml->children() as $element)
```

je postupně procházena každá značku v XML dokumentu, která je následně uložena do proměnné element a její název je porovnáván pomocí funkce `getName()` se značkami, které XML soubor obsahuje a provádí se následujícím způsobem:

```
if ($element->getName() === "topology")
{
$attr = $element->attributes();
if (! mysql_query("INSERT INTO TOPOLOGY(minX,...)"))

 foreach($element->children() as $elementChild)
 {
  if ($elementChild->getName() === "node")
  {
  $attrNode = $elementChild->attributes();
  if (! mysql_query("INSERT INTO NODE(id,...)"))
  }
 }
}
```

Podmínka porovnává aktuální značku v dokumentu s daným řetězcem a nastane-li shoda, provede se kód, který je ve složených závorkách. Do proměnné $attr se uloží všechny atributy ve značce *topology*, atributy jsou uloženy v poli. Indexy polí začínají nulou. Podmínkou

```
if (! mysql_query("INSERT INTO TOPOLOGY(minX,minY,maxX,maxY)
VALUES(".$attr[0].",".$attr[1].",".$attr[2].",".$attr[3].")"))
```

je vykonán samotný zápis hodnot do tabulky databáze (více v kap. 2.5 Databáze uložených hodnot). Jelikož se ve značce *topology* může vyskytnout více značek *node*, je potřeba cyklem

```
foreach($element->children() as $elementChild)
```

projít každý záznam, načíst jeho atributy do pole a zapsat je do tabulky stejným způsobem jako v případě zápisu atributů ve značce *topology*.

```
elseif ($element->getName() === "packet")
{
  $attr = $element->attributes();
  $elementChild = $element->children();
  $attrChild = $elementChild->attributes();
  if (! mysql_query("INSERT INTO PACKET(fromId,toId,aux...)"))
}
```

Jakmile jsou dokončeny všechny řádky předchozího kódu, přejde se na další řádek v XML dokumentu a pokud se jeho značka jmenuje *packet*, provede se opět načtení atributů značky *packet* do pole. Jelikože značka *packet* obsahuje pouze jednu značku *rx*, tak není nutné zavádět další cyklus, jako tomu bylo u značky *node*, ale stačí ve stromu klesnout příkazem

```
$elementChild = $element->children();
```

do značky *rx* a následně její atributy zapsat do proměnné $attrChild. Jednotlivé atributy, které jsou uloženy v proměnných, se do tabulky zapíší stejným způsobem jako tomu bylo u značek *node* či *topology*.

```
elseif ($element->getName() === "wpacket")
{
  $attr = $element->attributes();
  $elementChild = $element->children();
  $attrChildRx = $elementChild[0]->attributes();
  $attrChildMeta = $elementChild[1]->attributes();

  if (strpos($attrChildMeta[0],"DsdvHeader"))
     { ... ...
     if (! mysql_query("INSERT INTO DSDV (fromId,...)
}
```

Přístup k jednotlivým atributům ve značce *wpacket* je téměř shodný s postupem u značky *packet* s tím rozdílem, že ve značce *wpacket* jsou vnořeny 2 další značky, takže jejich atributy jsou načteny do pole proměnných $elementChild[0] a $elementChild[1]. Princip zápisu hodnot do tabulky je opět stejný jako v předchozích případech.
Wpacket obsahuje několik typů záznamův atributu *meta_info*, proto je pro *wpacket* vytvořeno několik samostatných tabulek, každá reprezentující jeden typ daného paketu.

2.4.2 Regulární výrazy

Pro vyhledávání konkrétních výrazů, jejich přesných částí nebo jejich nahrazení lze vytvořit pomocí regulárních výrazů. Vyhledávané pasáže jsou pak definovány pomocí vzorů. Jednoduchým regulárním výrazem jsou písmena a jejich skupiny, tedy slova. Při vyhledávání požadovaného výrazu je v textu zjištěno, zda je v textu obsaženo písmeno, skupina písmen nebo slovo, které je definováno pomocí regulárního výrazu. Takové řešení však nepřináší mnoho možností, jelikož často není známa přesná sekvence znaků nebo konkrétní slovo či písmeno. Proto se v regulárních výrazech používají tzv. metaznaky. Například znakem tečky (.) lze nahradit ve vyhledávaném výrazu libovolný znak, takže pomocí regulárního výrazu `.olo` najdeme slova jako molo nebo kolo. Dalším významným metaznakem jsou hranaté závorky [], pomocí nich můžeme vybrat jen určité vyhledávané znaky, například máme výraz `[mk]olo`, tím definujeme že vyhledáváme jeden ze znaků `m` a `k` a za nimi `olo`, takže výsledkem nemůže být například řetězec `polo`. [16]

Metaznaků v regulárních výrazech existuje celá řada, lze je do sebe kombinovat a vyhledávat tak složitější řetězce. Jednotlivé použité regulární výrazy jsou rozebrány u příslušných typů analyzovaných řetězců *meta_info*.

2.4.3 Parsování atributu meta_info

Jak již bylo psáno v kapitole 2.3.2 Typy přenášených paketů, v záznamu simulace existuje 6 typů paketů, které se liší informacemi obsaženými v atributu *meta_info*. Jelikož je nutné nějakým způsobem nejdříve rozeznat o jaký typ paketu se jedná, tak je deklarována podmínka, jestli je v daném *meta_info* obsažen daný řetězec, který je jedinečný pro každý typ paketu. To je zajištěno funkcí `strpos();` která porovnává daný řetězec s předepsaným vzorem.

Jedinečné části jednotlivých typů:
DSDV-"DsdvHeader"
FTP-"protocol 6"a zároveň "Fragment"
FTP_ACK-"protocol 6"a zároveň nesmí být "Fragment"
VoIP-"protocol 17"a zároveň "Payload"
ARP-"ArpHeader"
CTL_ACK-"CTL_ACK"

Prvním krokem je u každého *meta_info* řetězce provedeno rozdělení celého řetězce na části, které jsou odděleny řetězcem "ns3::". Tyto části jsou uloženy do pole a poté k je nim jednotlivě přistupováno. To je prováděno funkcí `preg_split();`.

```
$array = preg_split("/ *ns3:: */", $attrChildMeta[0], -1,
PREG_SPLIT_NO_EMPTY);
```

Tím je celý řetězec rozdělen po jednotlivých hlavičkách v poli. Pořadí hlaviček se u všech paketů nemění, takže lze přistupovat ke každé hlavičce zvlášť a v daném pořadí. Tato skutečnost zjednodušuje pozdější vyhledávání jednotlivých informací pomocí regulárních výrazů.

DSDV

```
preg_match("/[(][^)]+[)]/", $array[0], $tmp);
$WifiMacHeader = preg_replace("/[(|)]/","",$tmp[0]);
```

V hlavičce WifiMacHeader je hledán regulárním výrazem symbol závorky-"[(]", po ní následuje neprázdný řetězec , kde nesmí být znak konce závorky. Znak "+"je tzv. kvantifikátor a určuje, že předchozí znak tam musí být minimálně jednou. Nakonec má následovat znak ukončení závorky-"[)]". Nalezený výraz, v tomto případě je nalezen řetězec `(QOSDATA )`, je uložen do proměnné $tmp. Na druhém řádku je provedeno nahrazení nalezeného výrazu tak, aby odpovídal formě vhodné pro uložení do databáze. Z `(QOSDATA )` se tak stane jen `QOSDATA`, protože výrazem "[(|)]"je definováno, že znak "("nebo ")"má být odstraněn.

```
preg_match("/[(][^)]+[)]/", $array[1], $tmp);
$LlcSnapHeader = preg_replace("/[(|)]/","",$tmp[0]);
```

Z hlavičky LlcSnapHeader jsou data získána stejným způsobem jako u MAC hlavičky, protože řetězec je opět celý uvnitř kulatých závorek. Nalezeným výraz je `(type 0x800)` a opět je uložen do pomocné proměnné. Druhý řádek opět upravuje formát získané informace, provede se odstranění závorek.

```
preg_match("/protocol \d+/", $array[2], $tmp);
$protocol = preg_replace("/protocol /","",$tmp[0]);
if($protocol="17")
   $protocol="UDP";
```

Jako první hodnotu z hlavičky IPv4Header je získáno číslo použitého protokolu. To je definováno výrazem "/protocol d+/", kde "protocol: "je známá část řetězce, která předchází samotnému číslu protokolu, které je definováno výrazem "d+". Písmeno "d"zde označuje libovolnou číslovku a "+"je již známý kvantifikátor, takže je hledána posloupnost minimálně jedné číslovky. Nalezeným výrazem je zde `protocol 17`, další řádek pak řetězec "protocol "odstraní, takže zůstane pouze číselná hodnota

daného protokolu. Následující podmínkou je pak z daného čísla vytvořena slovní zkratka protokolu, aby bylo pro uživatele snažší rozpoznat použitý protokol v databázi.

```
preg_match("/length: \d+/", $array[2], $tmp);
$length = preg_replace("/length: /","",$tmp[0]);
```

Další položkou v IPv4Headeru je délka hlavičky. Vyhledávaný výraz je podobně jako v předchozím případě posloupnost známého řetězce "length: "a sekvence číslovek. Poté je známý řetězec odstraněn a dostaneme pouze číselnou hodnotu délky hlavičky.

```
preg_match("/\d+[.]\d+[.]\d+[.]\d+ > \d+[.]\d+[.]\d+[.]\d+/",
$array[2], $tmp);
$ip = preg_replace("/ > /"," ",$tmp[0]);
$ips = preg_split("/ /", $ip);
$src_IP = $ips[0];
$dst_IP = $ips[1];
```

Posledními zjišťovanými údaji v IPv4 hlavičce jsou IP adresy zdroje a cíle. Tyto údaje lze získat pomocí jediného výrazu, který hledá 4 po sobě jdoucí sekvence číslovek oddělené tečkou (IP adresa zdroje), následuje mezera, předem známý znak »", mezera a poté opět 4 sekvence číslovek oddělené tečkou. Z nalezeného výrazu `10.1.3.6 > 10.1.3.1` je pak nahrazena část řetězce mezi IP adresami jednou mezerou. Pro oddělení IP adresy zdroje je pak využita funkce `preg_split();` která rozdělí tento řetězec na dvě části a dělícím znakem je právě mezera. Oba řetězce jsou touto funkcí uložené do pole $ip a následně pak zvlášť uloženy do jednotlivých proměnných.

```
preg_match("/length: \d+/", $array[3], $tmp);
$UdpLength = preg_replace("/length: /","",$tmp[0]);
```

Z hlavičky UdpLength je získán údaj o její velikosti. Provádí se stejným způsobem jako u IPv4 hlavičky.

S tabulkou DSDV je také provázána tabulka DSDV_header, která obsahuje údaje z hlavičky DsdvHeader. Jelikož se hlaviček DsdvHeader může v paketu vyskytovat i více, bylo nutné tuto druhou tabulku vytvořit, jinak by v tabulce DSDV byly všechny tyto konkrétní záznamy v jedné buňce a porušilo by se tak pravidlo první normální formy.

DSDV_header

```
preg_match_all("/DestinationIpv4: \d+[.]\d+[.]\d+[.]\d+/",
$attrChildMeta[0], $tmp);
$i = 0;
foreach($tmp[0] as $t)
{
       $dsdv_dst_IP[$i] = preg_replace("/DestinationIpv4: /","",$t);
       $i++;
}
```

Funkcí jsou hledány všechny cílové IP adresy pomocí kombinace již známého řetězce "DestinationIpv4: "a výrazu, který odpovídá formě IP adresy. Všechny takové nalezené záznamy jsou pak uloženy do pole a cyklem se ze záznamů odstraní přebytečný kus řetězce.

```
preg_match_all("/Hopcount: d+/", \$attrChildMeta[0], \$tmp);
$i = 0;
foreach($tmp[0] as $t)
{
        $hop[$i] = preg_replace("/Hopcount: /","",$t);
        $i++;
}
```

Další vyhledávanou hodnotou je v DsdvHeaderu počet přeskoků. Vyhledává se opět pomocí kombinace známého řetězce a sekvence čísel. Jako v předchozím případě je cyklicky z každého nalezeného záznamu odstraněn přebytečný řetězec.
Zcela stejné jako v případě vyhledávání počtu přeskoků se provádí i hledání sekvenčních čísel.
Tyto 3 hodnoty (IP adresa, počet přeskoků a sekvenční číslo) jsou pak uloženy v tabulce DSDV_header.

Zápis hodnot do tabulky DSDV_header se částečně liší od ostatních případů zápisu do tabulky. Jelikož s každým sql příkazem `INSERT INTO` se vytvoří nový řádek v tabulce, tak v případě zápisu tří hodnot, které jsou uloženy v polích, musíme postupně procházet a zapisovat jednotlivé údaje a při použití této sql funkce by došlo k „rozházení“ dat, které by byly každá na vlastním řádku a ostatní sloupce v takovém řádku by byly bez údajů. Pro první hodnotu, což je IP adresa, lze použít standardní sql příkaz `INSERT INTO`, takže s každou novou IP adresou je vytvořen nový řádek. Hodnoty počtu přeskoků a sekvenčních čísel je však potřeba dostat na

stejný řádek. To je částečně zajištěno pomocí pomocného čítače ve sloupci Counter. Každému řádku odpovídá jeho jedinečná číselná hodnota. Právě pomocí těchto číselných hodnot lze upravit konkrétní řádek. To umožňuje sql příkaz `UPDATE`, kde je uveden název tabulky, pak ve kterém sloupci bude změna provedena a na jakou hodnotu. Dále je v příkazu specifikováno číslo řádku, ve kterém bude změna provedena. Pro zápis hodnot počtu přeskoků a sekvenčních čísel jsou zavedeny pomocné proměnné $hop_count a $seq_count, jejichž počáteční hodnota je jednička. S každým zápisem hodnoty je inkrementována příslušná pomocná proměnná a příští záznam dané hodnoty je pak aktualizován na následujícím řádku. Vše je zajištěno následující částí kódu:

```
foreach($dsdv_dst_IP as $dst_IP_element)
{
   $pk_count++;
   if (! mysql_query("INSERT INTO DSDV_header (Foreign_key, Counter,
   Dsdv_Destination_IP)
   VALUES (".$fk_dsdv.", ".$pk_count.", '".$dst_IP_element."')"))
}

foreach($hop as $hop_element)
{
   $hop_count++;
   if (! mysql_query("UPDATE DSDV_header
   SET Dsdv_Hopcount='".$hop_element."'
   WHERE Counter = ".$hop_count.""))
}

foreach($seq as $seq_element)
{
   $seq_count++;
   if (! mysql_query("UPDATE DSDV_header
   SET Dsdv_Sequence_Number='".$seq_element."'
   WHERE Counter = ".$seq_count.""))
}
```

Hodnoty z hlaviček WifiMacHeader, LlcSnapHeader a Ipv4Header jsou hledány v následujících typech paketů stejně, jako je tomu u hodnot DSDV paketů. Následující popis analýzy dalších typů paketů již obsahuje jen takové atributy, které ještě nebyly pomocí regulárních výrazů popsány.

FTP

```
preg_match("/Seq=\d+/", $array[3], $tmp);
$Seq = preg_replace("/Seq=/","",$tmp[0]);

preg_match("/Ack=\d+/", $array[3], $tmp);
$Ack = preg_replace("/Ack=/","",$tmp[0]);

preg_match("/Win=\d+/", $array[3], $tmp);
$Win = preg_replace("/Win=/","",$tmp[0]);
```

Hodnoty sekvenčního čísla, potvrzovacího čísla a velikosti okna jsou získány vyhledáním kombinace známého řetězce, v případě sekvenčního čísla je to "Seq=", kterému následuje sekvence čísel. Rozdíl je pouze v použitých proměnných a v hledaném řetězci pro danou hodnotu.

```
preg_match_all("/Payload[^\]]*\]/", $array[3], $tmp);
$i = 0;
foreach($tmp[0] as $t)
{
  $payload[$i] = preg_replace("/Payload[^\[]*/","",$t);
  $i++;
}
```

Jako u DSDV, tak u FTP se může vyskytovat jeden záznam s hodnotou v paketu i vícekrát. V případě FTP paketu je to hodnota Payload Fragment. Je vyhledávána kombinace známého řetězce "Payload"a dále řetězec znaků, který je zakončen hranatou závorkou "]". Takové pravidlo v tomto případě nalezne záznam `Payload Fragment [496:1032]`. Poté je cyklem řetězec upraven pouze na tvar `[496:1032]`, což je vhodný tvar pro uložení do databáze.

VoIP

```
preg_match("/size=\d+/", $array[3], $tmp);
$payload_size = preg_replace("/size=/","",$tmp[0]);
```

Jedinou hodnotou u VoIP paketů, která ještě nebyla rozebírána, je hodnota Payload. Zápis v *meta_info* je `Payload (size=500)`, takže vyhledat tento údaj stačí pomocí kombinace řetězce "size="a sekvenci čísel.

ARP

```
preg_match("/[(]\w+/", $array[2], $tmp);
$ARP_Type = preg_replace("/[(]/","",$tmp[0]);
```

Jako první údaj z hlavičky ARP zjistíme, o jaký typ ARP zprávy se jedná. Vyhledává se podle znaku kulaté závorky, kde následuje sekvence písmen. Znak závorky je následně z vyhledaného řetězce odstraněn.

```
preg_match("/source mac: \w+[-]\w+[-]\w+[:]\w+[:]\w+..",$array,$tmp);
$src_MAC = preg_replace("/source mac: /","",$tmp[0]);
```

Další údaj, který ARP paket obsahuje je údaj o zdrojové MAC adrese. Postup vyhledávání je opět kombinace známého řetězce a sekvencí znaků (písmena nebo čísla).

```
if ($ARP_Type=="reply")
{
  preg_match("/dest mac: \w+[-]\w+[-]\w+[:]\w+[:]\...",$array,$tmp);
  $dst_MAC = preg_replace("/dest mac: /","",$tmp[0]);
}
else
  $dst_MAC = "-";
```

V případě, že se jedná o ARP odpověď, tak je v ARP hlavičce obsažena i cílová MAC adresa. Z tohoto důvodu je zavedena podmínka. Pokud se jedná o odpověď, je z hlavičky získána MAC adresa cíle, jedná-li se o ARP žádost, tak je do buňky cílové MAC adresy zapsána pouze pomlčka.

CTL_ACK

Tento typ paketu neobsahuje žádné užitečné informace, pouze je napevno přiřazen jako typ MAC hlavičky řetězec "CTL_ACK".

2.5 Databáze uložených hodnot

Databáze je zjednodušeně úložiště dat a informací sloužící jako například evidence školní knihovny, sklad zboží apod. tedy přehledný záznam, ze kterého lze jednoduše získat konkrétní potřebné údaje. Ve skutečnosti je databáze soubor tabulek, ve kterých jsou uložené jednotlivé položky a mezi tabulkami a jejich jednotlivými záznamy mohou vznikat vazby.[11]
V případě zpracovávaného XML dokumentu lze pozorovat, že značka *node* je ve všech případech vnořená ve značce *topology* a mnohdy jde o několik záznamů značky *node* uvnitř jediného záznamu *topology*. Z toho můžeme vyvodit, že mezi těmito tabulkami a jejich záznamy vznikne relace typu 1:N. Dalším takovým případem jsou tabulky DSDV a DSDV_header, dále také FTP a FTP_payload.

2.5.1 Definice relačních klíčů

Abychom bylo možné mezi dvěma tabulkami vytvořit relaci, je nutné mít v jedné z nich data, která nám umožní vyhledat související záznamy v tabulce druhé. K takovému řešení slouží primární a cizí klíče. [14]

Primární klíče

Primárním klíčem bývá jeden nebo více atributů v dané tabulce, a jeho hodnoty slouží pro identifikaci příslušných řádků v tabulce druhé.
Požadované vlastnosti klíčů:

- v každém záznamu nesmí mít prázdnou hodnotu
- ve všech záznamech musí být tato hodnota jedinečná
- hodnoty primárního klíče se během práce s databází nesmí změnit nebo vyprázdnit
- pro daný řádek v tabulce může být přidělen pouze jeden primární klíč

Výhodou primárních klíčů je také fakt, že při jedinečné identifikaci záznamu pomáhá při zpětném vyhledávání, protože s přiřazením klíče se k danému atributu vygeneruje automaticky jeho index. Těchto indexů pak využívají vyhledávací algoritmy a práce s databází je pak rychlejší a efektivnější. [14]

Cizí klíče

Cizím klíčem se rozumí atribut, který zajišťuje vazbu mezi jednotlivými tabulkami. Hodnota cizího klíče odpovídá na hodnotu klíče primárního v tabulce, která je odkazovaná. Vytváří také jistou datovou integritu, která slouží pro orientaci mezi jednotlivými tabulkami celé databáze. [14]

Spojení s databází, její vytvoření a zapisování, je zjištěno pomocí jazyka SQL. SQL je deklarativní programovací jazyk, což znamená, že kód se nepíše v samostatném zdrojovém kódu, ale píšeme jej přímo do jiného programovacího jazyka (v tomto případě PHP), který provádí samotné procesy. V jazyce SQL jsou již implementovány nástroje pro tvorbu databází a manipulaci s daty jako je vkládání, aktualizace, mazání a vyhledávání. S jazykem SQL lze pracovat pouze tehdy, je-li aplikace spojená s SQL serverem.[11]

2.5.2 Normální formy databáze

Normalizací se rozumí v oblasti databázových systémů odstranění opakujících se dat a omezení složitosti (vytvoření více jednodušších tabulek než jednu složitou) a zabránění tzv. aktualizačním anomáliím (např. abychom smazáním všech záznamů s danou IP adresou zdroje nepřišli o všechny data s touto adresou spojená). Tyto pravidla a zásady by měly databázi zpřehlednit, umožní její lepší rozšiřitelnost a zefektivní její aplikační výkonnost.
Normalizace obecně vede ke vzniku takových tabulek, které můžeme jednoduše spravovat a vyhledávání v nich je jednodušší. Normalizovaná databáze by měla být navržena tak, aby se všechny závislosti zachovaly stejně jako u původního návrhu. Relace musí také zachovat původní data. Správně vytvořené tabulky splňují základní normální formy, které vymysleli pánové Codd a Boyce v 70. letech 20. století. Normalizace jsou rozděleny na několik úrovní a snahou je dosáhnout co nejvyšší úrovně. [12][13]

Normální formy:
1.NF - První normální forma
2.NF - Druhá normální forma
3.NF - Třetí normální forma
BCNF - Boyce-Coddova normální forma
4.NF - Čtvrtá normální forma
5.NF - Pátá normální forma

V praxi se nejčastěji využívají první tři normální formy. Databáze vyhovuje vyšší normální formě, pokud jsou splněna všechna normalizační pravidla nižších norem a zároveň je splněno pravidlo aktuální normální formy (každá následující normální forma zpřísňuje tu předchozí).[13]

1. normální forma (1.NF)

Relace splňuje první normální formu tehdy, pokud každý atribut, který tabulka obsahuje, již neobsahuje dále nedělitelné hodnoty. Například v relaci obsahující databázové údaje o simulaci: [12]

Tab. 2.1: Tabulka nesplňující 1. NF

fromId	toId	meta_info
5	1	DsdvHeader (10.1.3.4) :: DsdvHeader (10.1.3.8)
3	8	DsdvHeader (10.1.3.2) :: DsdvHeader (10.1.3.6)
6	5	DsdvHeader (10.1.3.5)

S takto navrženou tabulkou by mohlo docházet k řadě problémů, například by se obtížně zjišťovaly IP adresy v jednotlivých DSDV hlavičkách a také by nebylo jednoduché takové záznamy vyhledat.
Aby tabulka splňovala 1.NF, musíme buďto rozdělit atribut meta_info do více atributů, ale pouze za předpokladu, že se počet záznamů uvnitř atributu nezvýší, nebo jednotlivé IP adresy ukládat do samostatné tabulky, která bude s první tabulkou provázána pomocí klíčů. Výsledek pak může vypadat následovně: [12]

Tab. 2.2: Příklad 1. NF

Primary_Key	fromId	toId
1	5	1
2	3	8
3	6	5

Primary_Key	Foreign_Key	DsdvHeader_IP
1	1	10.1.3.4
2	1	10.1.3.8
3	2	10.1.3.2
4	2	10.1.3.6
5	3	10.1.3.5

2. normální forma (2.NF)

Relace splňuje druhou normální formu tehdy, pokud splňuje první normální formu a zároveň každý neklíčový atribut v tabulce zcela závisí na primárním klíči. Primární klíč však může být složen z více částí, takže aby relace splňovala druhou normální formu, musí pak každý neklíčový atribut záviset na celém klíči, nejen na jedné z jeho dílčích částí. Z takového pravidla pak vyplývá, že zavedení druhé normální formy použijeme v případě kdy je primární klíč složený z více hodnot.
Příkladem je tabulka využívaných zařízení ve firmě, kde jsou položky zařízení, výrobce, IP adresa zařízení a emailový kontakt na výrobce v případě nutnosti jej kontaktovat: [12]

Tab. 2.3: Tabulka nesplňující 2. NF

Zařízení	Výrobce	IP adresa	Podpora
FTP Server	IBM	10.0.0.2	erchelp@uk.ibm.com
HTML Server	IBM	10.0.0.5	erchelp@uk.ibm.com
Hraniční Směrovač	Cisco	10.0.0.1	rthelp@cisco.com
Počítač	Dell	10.0.0.9	support@dell.cz

Klíčem této tabulky je kombinace atributů Zařízení a Výrobce. Adresa podpory na výrobce ale není závislá na celém klíči, ale pouze na samotném atributu Výrobce. Takový návrh by mohl vést k tzv. aktualizační anomálii. Pokud by se odstranily veškeré zařízení od výrobce IBM, tak přijdeme o kontakt na tohoto výrobce, což nemusí být žádané. Řešením je rozložení na dvě tabulky. Při smazání zařízení od některého výrobce pak nepřijdeme o kontaktní údaje, které jsou nyní uložené v jiné tabulce: [12]

Tab. 2.4: Příklad 2. NF

Zařízení	ID_Výrobce	IP adresa
FTP Server	1	10.0.0.2
HTML Server	1	10.0.0.5
Hraniční Směrovač	2	10.0.0.1
Počítač	3	10.0.0.9

ID_Výrobce	Výrobce	Podpora
1	IBM	erchelp@uk.ibm.com
2	Cisco	rthelp@cisco.com
3	Dell	support@dell.cz

2.5.3 Normalizace vytvářené databáze

Původní návrh databáze uložených hodnot ze simulace obsahuje 4 tabulky (node, topology, packet, wpacket), tabulky NODE a TOPOLOGY jsou provázány klíčem, který se skládá z jediného atributu, takže v tomto případě není nutné řešit vyšší normální formu než je 1.NF. Každá tabulka v databázi také obsahuje primární klíč.

Tab. 2.5: Normalizace - tabulka Topology

Primary_key	minX	maxX	minY	maxY
1	-3.1	29.9	-2.9	24.1
2	-3.1	29.9	-2.9	24.1
3	-3.1	29.9	-2.9	24.1
4	-3.1	29.9	-2.9	24.1

Tab. 2.6: Normalizace - tabulka Node

Primary_key	Foreign key	Id	descr	locX	locY
1	1	4	STA	14.6235	12.58
2	2	6	STA	11.293	12.58
3	3	0	STA	8.78727	9.24233
4	3	1	STA	16.7859	8.74932

Tab. 2.7: Normalizace - tabulka Packet

Primary_key	fromId	toId	aux	fbTx	lbTx	fbRx	lbRx
1	0	0	DummyPkt	1.4846	1.4846	1.4846	1.4846
2	0	0	DummyPkt	1.5	1.5	1.5	1.5
3	0	0	DummyPkt	1.5589	1.5589	1.5589	1.5589
4	0	0	DummyPkt	1.5821	1.5821	1.5821	1.5821

Tab. 2.8: Normalizace - tabulka Wpacket

Primary_key	fromId	toId	range	fbTx	lbTx	fbRx	lbRx
1	0	4	20.3185	12.6193	12.6193	12.6193	12.6193
2	7	3	3.5956	13.8399	13.8399	13.8400	13.8400
3	3	8	7.6901	15.1455	15.1455	15.1455	15.1455
4	9	6	14.0632	23.6463	23.6463	23.6464	23.6464

Záznam z tabulky WPACKET obsahuje také položku *meta_info*, ale pro vložení do dokumentu je záznam příliš dlouhý. Příklad záznamu pro první řádek z tabulky je vypsaný zde:

```
ns3::WifiMacHeader (QOSDATA ) ns3::LlcSnapHeader (type 0x800)
ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 1 id 1
protocol 17 offset (bytes)
```

Z výpisu je patrné, že kromě tabulky WPACKET jsou všechny atributy v tabulkách již dále nedělitelné, takže splňují první normální formu. V případě tabulky WPACKET nesplňuje požadavky pro 1.NF atribut *meta_info*, ve kterém jsou uloženy informace například o typu protokolu, IP adresy zdroje a cíle nebo jestli se jedná o datový paket nebo paket směrovacího protokolu. Tento atribut je tedy nutné rozdělit na více částí.

Další záležitostí jsou typy paketů (kapitola 2.3.2 Typy přenášených paketů). V analyzované simulaci se vyskytuje 6 typů paketů, které mají zcela stejné XML atributy, ale v řetězci *meta_info* se nachází několik důležitých informací, které budou z řetězce parsovány a právě v těchto informacích se jednotlivé záznamy WPACKET značně liší. Například v DSDV směrovacích paketech se může vyskytovat několik hlaviček s cílovými adresami a počtem přeskoků, naopak v paketech ARP protokolu tyto informace nejsou obsaženy, ale jsou zde zdrojové, případně i cílové MAC adresy apod. Řešením je vytvoření samostatných tabulek pro každý typ paketu.

V DSDV záznamech se může vyskytovat jedna a více hlaviček, které obsahují cílové IP adresy, počet přeskoků a sekvenční číslo. Počet DSDV hlaviček není dopředu znám a s každým paketem se může lišit. Aby tato tabulka splňovala první normální formu, je obsah všech hlaviček zapisován do druhé tabulky DSDV_header. S hlavní tabulkou pro směrovací pakety DSDV je pak tato tabulka provázána pomocí klíče, takže pro každý řádek v DSDV tabulce je vazba na ten počet DSDV hlaviček, jaký se v daném paketu vyskytoval.

Stejný případ nastává u FTP paketů, kde se může vyskytovat několik informací Payload Fragment. Řešení je zcela totožné jako v případě DSDV paketů, opět je zde vytvořena druhá tabulka FTP_payload se záznamy Payload Fragment.

Tabulka DSDV pak po hlubší analýze vypadá následovně (tabulky bylo nutné kvůli své velikosti rozdělit v dokumentu na více částí, v databázi je vše v jedné souvislé tabulce):

Tab. 2.9: Normalizace - tabulka DSDV

Primary_key	**fromId**	**toId**	**reception_range**	**fbTx**
1	7	3	5.385550164	0.0040070
2	5	0	10.21500399	0.0040070

lbTx	**fbRx**	**lbRx**	**MAC_header**	**LlcSnapHeader**
0.0040070	0.0040070	0.0040070	QOSDATA	type 0x800
0.0040070	0.0040070	0.0040070	QOSDATA	type 0x800

IPv4_protocol	**IPv4_length**	**IPv4_source**	**IPv4_dst**	**UDP_length**
UDP	40	10.1.3.9	10.1.3.255	20
UDP	148	10.1.3.4	10.1.3.255	128

K tomu druhá tabulka DSDV_header s vazbou na DSDV tabulku:

Tab. 2.10: Normalizace - tabulka DSDV_header

Primary_key	**Counter**	**Foreign_key**	**DSDV_IP**	**Hop**	**Seq**
1	1	1	10.1.3.9	2	2
2	2	2	10.1.3.8	2	2
3	3	2	10.1.3.9	1	2
4	4	2	10.1.3.4	1	4

U dalších tabulek paketů jiných typů je nebylo dělit, protože pro každou informaci existoval v daném paketu jen jeden údaj nebo hodnota. Všechny tabulky jsou zobrazeny v příloze. Tímto je splněna podmínka první normální formy, jelikož všechny atributy v tabulkách již nejsou dále dělitelné. Celá databáze tak celkově splňuje 1.NF, s ohledem na podmínky dalších úrovní není nutná jejich aplikace v této databázi. Přehled všech tabulek, které se v celé databázi vyskytují jsou zobrazeny v příloze.

2.5.4 Vytváření databáze v aplikaci

Jelikož délka běhu aplikace je větší jak 1 minuta a může se měnit v závislosti na aplikačním výkonu zařízení na kterém aplikace běží, tak je funkcí `ini set('max execution time', 0);` deklarován nekonečný časový interval, který může aplikace běžet, protože defaultně je tato hodnota nastavená na 60 sekund a po uplynutí této doby zobrazí chybovou hlášku o vypršení časového limitu běhu aplikace.

Příkazy

```
$con = mysql_connect("localhost", "root");
mysql_select_db("kolaja");
```

je aplikace připojena k samotnému databázovému serveru do dané databáze. V tomhle případě je použita místní databáze, která je součástí balíčku EasyPHP a běží přímo na vlastním zařízení.
Jelikož bylo při sestavování aplikace nutné opakovaně zkoušet její funkčnost, tak příkazem

```
mysql_query("DROP TABLE x",$con);
```

jsou odstraněny tabulky vytvořené dříve (písmeno x představuje název tabulky která je odstraněna).

Následujícím příkazem je vytvořena samotná tabulka a jsou deklarovány typy proměnných pro jednotlivé sloupce. Postup je stejný pro vytváření všech tabulek. V případě chyby se provede funkce `die();`, která vypíše chybovou hlášku a provádění skriptu se zastaví.

```
$sql = "CREATE TABLE TOPOLOGY(Primary_key,minX real,...)";
if (! mysql_query($sql,$con))
die("Chyba pri vytvareni tabulky TOPOLOGY");
```

Jakmile jsou tabulky vytvořeny, tak následuje deklarace primárních a cizích klíčů, které zajišťují vazby mezi tabulkami NODE-TOPOLOGY, DSDV_header-DSDV a FTP_payload-FTP. Primární klíč je nenulová hodnota začínající číslem 1 a s každým novým záznamem se jeho hodnota automaticky zvýší o 1. Hodnota cizího klíče je řešena přes pomocné proměnné jako je např. $fk_node, jejíž hodnota je zvětšena o 1 s každým nově nalezeným záznamem v nadřazené tabulce. Pro každý takový záznam v podřazené tabulce je zapsána aktuální hodnota cizího klíče, která odpovídá hodnotě primárního klíče.

```
mysql_query("ALTER TABLE TOPOLOGY ADD CONSTRAINT PK_TOPOLOGY
PRIMARY KEY (Primary_key)", $con);
mysql_query("ALTER TABLE TOPOLOGY MODIFY Primary_key int
AUTO_INCREMENT", $con);
mysql_query("ALTER TABLE NODE ADD CONSTRAINT FK_NODE_TOPOLOGY
FOREIGN KEY (Foreign_key) REFERENCES TOPOLOGY(Primary_key)",
$con);
```

Samotný zápis hodnot do databáze je prováděn v části kódu, který vyhledává a zjišťuje atributy jednotlivých značek (viz kap. 2.4.1 SimpleXML).

3 VÝPOČET PARAMETRŮ SIMULACE

3.1 Kvalita služby - QoS

Internetová síť byla vybudována bez cíle zajištění doručení všech odeslaných zpráv do určité doby nebo zajištění dané přenosové rychlosti po dobu trvání určité komunikace. Základní pravidla stanovená v počátcích internetu deklarovala, že žádný typ provozu nebude při přístupu do sítě zamítnut a s veškerým provozem bude zacházeno stejně. Jedinou garancí byl fakt, že bude provoz přenesen co nejlepším způsobem tzv. „best effort“, jaký v daný moment byl k dispozici, takže nebude na cestě nijak uměle zpožděn a nesmí docházet ke zbytečným ztrátám jednotlivých zpráv.

Tento typ služby ve skutečnosti ve většině případů může být dostatečný nejen například pro elektronickou poštu, kde zpoždění nebo jeho kolísání nemá na výsledek vliv, ale i pro hlasové služby, ale pouze jen do té doby, než na komunikační cestě dojde k přetížení. V případě „best effort“ neexistuje žádná jistota v parametrech dané služby, protože v době komunikace dochází k různému zatížení sítě a to může způsobit delší zpoždění nebo i ztrátu datagramů. Zpoždění a ztráty jsou tak nezajištěné v rámci nějakých mezí.

V jádru Internetu dochází neustále k navyšování přenosové kapacity, a proto zde nedochází ke ztrátám datagramů a lze dosáhnout velmi malého zpoždění. Naopak na okrajových částech Internetu, kde infrastruktura nemá takovou kapacitu, může docházet k nedostatečné šířce pásma a následnému zahlcení.

Podle ITU-T E.800 je kvalita služby definována jako „souhrnný výsledek výkonnosti služby, který určuje stupeň spokojenosti uživatele dané služby“. Kvalita služby se v prostředí IP definuje výkonností toku paketů jednou nebo více sítěmi. Cílem je splnit jistá kritéria při doručování paketů mezi koncovými účastníky sítě. Jakým způsobem se daná aplikace bude chovat, závisí na časových charakteristikách komunikačního kanálu, jako je například propustnost a zpoždění. Aplikacemi se rozumí například videokonference, hlasové služby, přenos dat či procházení WWW stránek. Osoba využívající takovou aplikaci pak očekává určitou kvalitu takové služby na aplikační úrovni jako je například doba trvání přenosu souborů, počet zobrazených obrázku za sekundu nebo srozumitelná hlasová komunikace s jinou osobou. [15]

3.1.1 Parametry QoS

Kvalita služeb závisí na několika základních veličinách, mezi které patří zpoždění (souvisí se šířkou pásma), proměnlivost zpoždění, ztrátovost paketů (spolehlivost), dostupnost služby nebo přenosová rychlost.

Každá aplikace vyžaduje své nároky na konkrétní veličiny, na některých však nemusí vůbec záležet. Přísné podmínky mají například hlasové služby a video v reálném čase kde je vysoká citlivost na zpoždění a jeho kolísání, naopak u datových přenosů jsou nároky nejnižší, avšak zde je nutná jistá spolehlivost a potřebná šířka pásma. [15]

3.1.2 Zpoždění

Označuje se také pod pojmem latence a jde o dobu, za kterou se daný paket dostane od zdroje do cíle. Zpoždění se skládá z několika částí, které vyjadřují, jak dlouho se v daném úseku paket zdržel. Máme tedy zpoždění kódováním a serializací (příprava paketů pro přenos v síti na daném médiu) a zpoždění při přenosu (vychází z rychlosti světla a vzdálenosti). Toto jsou pevně dané hodnoty. Dále se skládá ze zpoždění ve frontě a zpoždění při přepínání (nalezení cesty ve směrovači), tyto hodnoty se v průběhu komunikace stále mění.

Obecně platí, že při krátkých paketech bývá zpoždění velmi malé. Je-li paket velký, může docházet k dlouhé serializaci při čekání za tímto datovým paketem. Proto se používá systém fragmentace a prokládání (LFI- Link Fragmentation and Interleaving), to umožní paket rozdělit na několik menších částí, aby se zmenšila doba čekání při vysílání dlouhého paketu.

Zpoždění při čekání ve frontě je další důležitou složkou celkového zpoždění v síti. Závisí na aktuální zátěži sítě, daného spoje a portu zařízení. Dlouhé fronty mohou vést k pomalému zpracování což má za následek velké zpoždění, příliš krátké fronty pak způsobují ztrátu paketů, protože při pomalém odbavování se fronta zaplní a už není schopna další pakety pojmout a zahodí je.

Zpoždění má největší vliv na hlasovou komunikaci, zpoždění větší jak 150 ms má za následek trhání hlasu a komunikace pak může být uživateli znemožněna. [15]

3.1.3 Kolísání zpoždění - Jitter

Po dobu komunikace nemusí pakety ze zdroje přicházet všechny ve stejný časový interval. To je způsobeno zpožděním které nastává na pomalých spojích, rozdílných délkách front a také souvisí s aktuálním zahlcením sítě a rozdílech v použitých frontových mechanizmech.

Nejcitlivější aplikací na kolísání zpoždění jsou opět hlasové služby. Většinou se tento problém řeší vyrovnávacími zásobníky v telefonech a VoIP bránách, které dokážou přizpůsobit hlasový provoz při kolísání zpoždění 20–50 ms. Mimo tyto hodnoty už paměti nestačí a kvalita hovoru může být ovlivněna. [15]

Konkrétní maximální a doporučené hodnoty QoS pro VoIP:
Ztrátovost paketů: 1%
Zpoždění: 150 ms
Kolísání zpoždění: 30 ms
Minimální šířka pásma: 12–106 Kb/s (závisí na kodeku apod.)

3.2 Algoritmy výpočtů parametrů

3.2.1 Výpočet zpoždění paketu

Pro výpočet zpoždění paketů se budou využívat hodnoty časů odesílání a přijímání prvního a posledního bitu daného paketu, v XML konkrétně hodnoty atributů *fbTX, lbTx, fbRX a lbRx*. Hodnoty prvního a posledního bitu odesílání nebo přijímání se nejdříve zprůměrují a následně od výsledného času příjmu paketu se odečte čas odeslání paketu, čímž dostaneme hodnotu zpoždění pro jeden konkrétní paket a tato hodnota se uloží do proměnné $avg. Tyto výpočty jsou prováděny následujícím řádkem kódu:

```
$avg = (((float)$attrChildRx[1] + (float)$attrChildRx[2])/2) -
 (((float)$attr[1] + (float)$attr[2])/2);
```

Aby byl údaj o zpoždění paketu rozlišen pro každý typ provozu, tak následující instrukce kódu jsou uvnitř podmínky, která zjišťuje pomocí funkce `strpos();` (více v kapitole 2.4.3 Parsování atributu meta_info), o jaký typ paketu se jedná. Výpočty tedy probíhají pouze u FTP a VoIP paketů.

```
$tmp_delay_FTP[$packet_counter_FTP_all] = $avg;
$packet_counter_FTP_all++;
```

Do pole $tmp_delay_FTP je uložena s každým paketem aktuální hodnota vypočítaného zpoždění pro konkrétní paket. Pro zajištění zápisu každé hodnoty vypočteného zpoždění do svého indexu pole je s každým nalezeným paketem inkrementována proměnná $packet_counter_FTP_all, která v tomto případě představuje index v poli. Tato proměnná zároveň slouží jako čítač paketů, tak získáme na konci běhu aplikace počet FTP paketů, který využijeme k finálním výpočtům zpoždění a jitteru FTP a VoIP provozu.

```
if ($packet_counter_FTP_all != 0){
  for($i=0; $i<$packet_counter_FTP_all; $i++)
  {
  $delay_FTP += $tmp_delay_FTP[$i];
  }
  $delay_FTP /= $packet_counter_FTP_all;
```

Jakmile aplikace provede analýzu dat a zápis do databáze, tak se podmínkou ošetří, jestli byl nalezen alespoň jeden FTP paket, aby nedošlo k dělení nulou. Cyklem jsou poté sečteny všechny vypočtené hodnoty zpoždění do jedné proměnné s názvem $delay_FTP. Celkový součet hodnot zpoždění je poté vydělen počtem nalezených paketů a tak je vypočítaná průměrná hodnota zpoždění pro daný provoz. V proměnné $delay_FTP je již tedy výsledná hodnota zpoždění pro FTP provoz, která se později zapíše do databáze nebo vypíše v okně webového serveru.

Výpočet zpoždění pro VoIP provoz probíhá totožně, jen s rozdílem názvu proměnných, kde místo části "_FTP"je "_VoIP".

3.2.2 Výpočet kolísání zpoždění - Jitter

Pro výpočet kolísání zpoždění je využita již vypočítaná hodnota zpoždění, která bude sloužit jako střední hodnota.

```
for($i=0; $i<$packet_counter_FTP_all; $i++)
{
   $jitter_FTP += abs($delay_FTP - $tmp_delay_FTP[$i]);
}
   $jitter_FTP /= $packet_counter_FTP_all;
```

Se střední hodnotou zpoždění jsou porovnány jednotlivé hodnoty zpoždění, které jsou uložené v poli $tmp_delay_FTP a vypočítá se rozdíl v absolutní hodnotě. Zároveň je každá vypočítaná hodnota kolísání zpoždění sčítána do jediné proměnné $jitter_FTP. Nakonec je stejně jako u zpoždění tento součet vydělen počtem FTP paketů a tak je získána průměrná a tedy konečná hodnota kolísání zpoždění.

Výpočet jitteru pro VoIP provoz je opět shodný jako pro FTP provoz.

3.2.3 Výpočet propustnosti - šířky pásma

Propustnost daného komunikačního spojení mezi dvěma uzly se dá z XML souboru vypočítat tak, je-li znám počet paketů pro daný typ provozu mezi konkrétními dvěma uzly, velikost paketu pro daný provoz a čas, po který komunikace probíhala. Velikost FTP paketu je 614 bajtů, velikost VoIP paketu je 566 bajtů. Délka simulace byla 20 sekund, avšak FTP provoz probíhal pouze 3,5 sekundy a VoIP provoz trval 1 sekundu. Pro čítač paketů mezi konkrétními dvěma uzly je využito dvourozměrné pole $packet_counter_FTP. Nejříve je nutné zjistit ID zdroje a cíle, které se budou využívat jako indexy dvojrozměrného pole, do kterého se budou ukládat hodnoty počtu nalezených paketů. Tímto je zajištěn výpočet pro všechny možné komunikační kombinace, jaké mohou při 10 uzlech nastat.

```
$src = (int)$attr[0];
$dst = (int)$attrChildRx[0];
```

V proměnné $src je uložen ID zdroje a v proměnné $dst je ID cíle pro daný paket. Následně je, stejně jako v případě výpočtu zpoždění, podmínkou rozlišen typ provozu k oddělení výsledků pro FTP a VoIP.

```
if (!isset($packet_counter_FTP[$src][$dst]))
```

```
$packet_counter_FTP[$src][$dst] = 1;
else
$packet_counter_FTP[$src][$dst]++;
```

Touto podmínkou a funkcí isset(); je zjištěno, je-li v proměnné $packet_counter_FTP již nějaký záznam na konkrétním indexu v tomto dvojrozměrném poli. Pokud se jedná o první zápis, zapíše se jednička. V opačném případě se hodnota v poli zvýší o jedna. Tato proměnná slouží k zaznamenání počtu paketů pro daný typ provozu a mezi konkrétními dvěma uzly, které jsou vyjádřeny indexy pole a slouží k finálnímu výpočtu propustnosti.

```
foreach ($packet_counter_FTP as $src => $v1)
{
   foreach ($v1 as $dst => $v2)
   {
      $a_FTP++;
      $speed_FTP += (($packet_counter_FTP[$src][$dst]*614*8)/3.5);
   }
}
$fin_speed_FTP = $speed_FTP / $a_FTP;
```

Oba cykly projdou celé pole $packet_counter_FTP postupně buňku po buňce. Každá buňka obsahuje počet nalezených paketů mezi dvěma konkrétními uzly, který je využit pro mezivýpočet přenosové rychlosti. Do proměnné $speed_FTP se postupně přičítají vypočtené rychlosti z konkrétních uzlů. Výpočet je počet paketů vynásobený jeho velikostí a to celé poděleno délkou trvání FTP provozu. Pro finální výsledek $fin_speed_FTP je ještě využita proměnná $a_FTP, kterou je podělen součet rychlostí. Konečná rychlost je pak průměrem všech rychlostí pro daný typ provozu mezi konkrétními dvěma uzly. Propustnost je rozlišena pro oba typy provozu.

3.2.4 Zobrazení výsledků a zápis do databáze

Všechny vypočtené výsledky jsou po ukončení běhu aplikace zobrazeny v okně webového serveru na adrese, kde byla aplikace spuštěna. To slouží pro rychlejší orientaci ve výsledcích. Výsledky jsou v okně zaokrouhleny na méně destinných míst. Do databáze se zapíše výsledek bez zaokrouhlování do tabulky RESULTS. Jednotky zpoždění a kolísání zpoždění jsou v sekundách, jednotky přenosové rychlosti jsou v kilobitech za sekundu.

Zobrazení výsledků v okně je zajištěno následující částí kódu:

```
printf("<br>FTP:<br>");
printf("Zpozdeni: %.12f s<br>",$delay_FTP);
printf("Prumerny jitter: %.12f s<br>",$jitter_FTP);
printf("Prumerna rychlost: %.2f Kbit/s<br>",$fin_speed_FTP);

echo "<br><br>";
printf("VoIP:<br>");
printf("Zpozdeni: %.12f s<br>",$delay_VoIP);
printf("Prumerny jitter: %.12f s<br>",$jitter_VoIP);
printf("Prumerna rychlost: %.2f Kbit/s<br>",$fin_speed_VoIP);
```

Zápis do databáze se již provádí standardním SQL příkazem:

```
if(!mysql_query("INSERT INTO RESULTS(zpozdeni_FTP,zpozdeni_VoIP,...)
VALUES(".$delay_FTP.",".$delay_VoIP.", ...)"))
echo mysql_error() . "</br>";
```

3.3 Výsledné hodnoty parametrů simulované sítě

3.3.1 Zpoždění paketu

Zpoždení FTP: 0.000000031269 s = 31,3 ns
Zpoždění VoIP: 0.000000030799 s = 30,8 ns

Rozdíly mezi hodnotami zpoždění obou provozů jsou nepatrné. Vypočtené hodnoty neodpovídají teoretickým předpokladům, hodnota zpoždění bývá alespoň řádově jednotek milisekund. Hodnoty časů v XML dokumentu jsou pravděpodobně zaznamenány z nejnižší vrstvy TCP/IP modelu, takže zde není zahrnuto zpoždění způsobené zapouzdřením a rozbalováním, bezpečnostními algoritmy v jednotlivých vrstvách apod. Vypočítané hodnoty tedy odpovídají době zpoždění na fyzickém médiu. Naměřené hodnoty byly vypočítány i ručně z několika vzorků z XML dokumentu a s výpočty z aplikace souhlasí.

3.3.2 Proměnlivost zpoždění-Jitter

Průměrný jitter FTP: 0.000000008967 s = 8,97 ns
Průměrný jitter VoIP: 0.000000011360 s = 11,36 ns

Opět jako v případě zpoždění jsou hodnoty kolísání zpoždění velmi malé z důvodu záznamu časových hodnot simulace z nejnižšího síťového modelu. Z tohoto důvodu nelze říci, zda simulovaná síť splňuje požadavky na kvalitu jednotlivých služeb. Pro objektivní určení těchto hodnot je lepší varianta využít již implementovaných nástrojů pro měření parametrů, jako je například FlowMonitor, který je v simulátoru zabudován.

3.3.3 Přenosová rychlost

Průměrná rychlost FTP: 609.09 Kbit/s
Průměrná rychlost VoIP: 905.60 Kbit/s

Hodnoty přenosové rychlosti vypočítané aplikací již odpovídají teoretickým předpokladům. Teoretická maximální propustnost linky byla v simulaci nastavena na 1 Mbit/s, takže tato hodnota nebyla překročena. Menší přenosová rychlost u FTP oproti rychlosti u VoIP je pravděpodobně způsobena podstatou spojově orientovaného protokolu, který FTP provoz používá. FTP provoz má oproti VoIP nutnost ověřovat, zda-li daný paket byl skutečně v pořádku přijat. K tomu se využívají potvrzovací pakety (FTP_ACK) a jejich režie způsobuje pokles přenosové rychlosti.

4 ZÁVĚR

V bakalářské práci byly popsány principy a charakteristické vlastnosti bezdrátových mobilních ad hoc sítí. Také jsou uvedeny příklady jejich využití a budoucí vývoj. Dále byla popsána problematika směrování v mobilních ad hoc sítích. Blíže byly představeny některé reaktivní a proaktivní směrovací protokoly jako je AODV, DSDV nebo OLSR. Práce se také zabývá metodami pohybu uzlů v MANET sítích, které se používají jako modely předpokládaného pohybu při skutečném nasazení. Blíže byly popsány modely pohybu, jako je model náhodně zvoleného cíle, náhodně zvolené trasy nebo model Manhattan, který patří do skupiny metody zeměpisně omezené mobility. V práci je také popsána bezpečnost a potřeby zabezpečení mobilních bezdrátových sítí. K praktické části byla také popsána problematika kvality služeb, regulárních výrazů a normálních forem pro tvorbu databáze.

V praktické části bylo cílem zpracovat výstupní XML dokument, který obsahuje informace zachycené v průběhu simulace. Tento soubor vytvářela již nakonfigurovaná simulace bezdrátové mobilní ad hoc sítě. V práci je XML dokument rozebrán a popsána je jeho vnitřní struktura a obsah. Samotný dokument zpracovává vytvořená aplikace napsaná v jazyce PHP, která využívá knihovnu SimpleXML pro parsování XML dokumentu. Pro podrobnější analýzu některých atributů jsou využívány regulární výrazy. Jednotlivé datové informace jsou touto aplikací odeslány pomocí SQL příkazů do databázového systému, ve kterém jsou přehledně uloženy do tabulek. Databáze je navržena s ohledem na normální formy databází. Forma a deklarace tabulek a jejich atributů se provádí ve vytvořené aplikaci. Jelikož některé záznamy nebylo možné sjednotit do jediné tabulky, tak byly uloženy do různých tabulek, které se v aplikaci provázaly pomocí primárních a cizích klíčů. Ukázky záznamů jednotlivých tabulek jsou v příloze tohoto dokumentu. Z hodnot získaných z XML dokumentu jsou vypočítány některé základní parametry simulované sítě jako je zpoždění, proměnlivost zpoždění a přenosová rychlost. Parametry jsou vypočítány pro jednotlivé typy provozu.

LITERATURA

[1] MACHATA, Tomáš a Jiří HOŠEK. Popis směrovacího protokolu AODV pro MANET sítě a jeho následné rozdělení o nový typ zprávy v prostředí OPNET Modeler. *Elektrorevue.* Fakulta elektrotechniky a komunikačních technologií VUT v Brně, 2011, roč. 2011, č. 47, s. 10. ISSN 1213-1539.

[2] CALAFATE, Carlos, Pablo GARRIDO, José OLIVER a Manuel MALUMBRES. *Mobile ad hoc networks.* Technical University of Valencia, Spain.

[3] GORANTALA, Krishna. *Routing Protocols in Mobile Ad-hoc Networks.* Umea, Sweden, 2006. Master Thesis. Umea University, Department of Computing Science. Vedoucí práce Thomas Nilsson.

[4] VÁVRA, Michal. *Mobilní komunikační sítě.* Brno, 2010. Diplomová práce. Masarykova Univerzita, Fakulta informatiky. Vedoucí práce doc. RNDr. Eva Hladká, Ph.D. Dostupné z: <`http://is.muni.cz/th/172871/fi_m/thesis.pdf`>.

[5] MACHATA, Tomáš. *Analýza modelů směrovacích protokolů OLSR a AODV pro MANET sítě v prostředí OPNET modeler.* Brno, 2011. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Ing. Jiří Hošek.

[6] ROY, Radhika Ranjan. *Handbook of Mobile Ad Hoc Networks for Mobility Models.* Fort Monmouth, NJ 07703, USA: Springer, 2011. ISBN 978-1-4419-6048-1.

[7] ABUSALAH, L. a A. KHOKHAR. TARP Performance in a Mobile World. *Globecom / IEEE Global Telecommunications Conference.* Department of Electrical and Computer Engineering University of Illinois at Chicago, 2007, s. 5. ISSN 1930-529X.

[8] ns-3 project. *ns-3 Manual, Release ns-3-dev.* Dostupné z: <`http://www.nsnam.org/docs/release/3.13/manual/ns-3-manual.pdf`>.

[9] KACHAN, Dmitry. *Integration of ns-3 with MATLAB/Simulink.* Lulea University of Technology, Sweden, 2010. ISSN: 1653-0187. Dostupné z: <`http://epubl.ltu.se/1653-0187/2010/062/LTU-PB-EX-10062-SE.pdf`>. Master Thesis.

[10] KOSEK, Jiří. *PHP a XML.* Grada Publishing, 2009, 368 s. ISBN 978-80-247-1116-4.

[11] OPPEL, Andrew J. *SQL bez předchozích znalostí: [průvodce pro samouky].* Vyd. 1. Brno: Computer Press, 2008, 240 s. ISBN 978-80-251-1707-1.

[12] WOHLGEMUTH, Jan. *ANALÝZA, NÁVRH A IMPLEMENTACE INFORMAČNÍHO SYSTÉMU.* Brno, 2009. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská.

[13] GILFILLAN, Ian. *Myslíme v jazyce MySQL 4.* Vyd. 1. Praha: Grada, 2003, 750 s. ISBN 80-247-0661-X.

[14] AGARWAL, Vidya Vrat a James HUDDLESTON. *Databáze v C# 2008: průvodce programátora.* Vyd. 1. Překlad Lukáš Krejčí. Brno: Computer Press, 2009, 424 s. ISBN 978-80-251-2309-6.

[15] PUŽMANOVÁ, Rita. *TCP/IP v kostce.* 2. upr. a rozš. vyd. České Budějovice: Kopp, 2009, 619 s. ISBN 978-80-7232-388-3.

[16] LACKO, Ľuboslav. *PHP 5 a MySQL 5: hotová řešení.* Vyd. 1. Brno: Computer Press, 2007, 320 s. ISBN 978-80-251-1695-1.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

MANET Mobile Ad-hoc Networks - Mobilní ad-hoc sítě

DARPA Defense Advanced Research Projects Agency

AODV Ad-hoc On Demand Distance Vector

DSR Dynamic Source Routing

OLSR Optimized Link State Routing

DSDV Destination-Sequenced Distance Vector

MPR Multipoint Relays

VANET Vehicular Ad-hoc Network

MAC Media Access Control - řízení přístupu média

QoS Quality of Service - Kvalita služby

XML Extensible Markup Language - Rozšiřitelný značkovací jazyk

TCP Transmission Control Protocol

UDP User Datagram Protocol

FTP File Transfer Protocol

VoIP Voice over Internet Protocol

IP Internet Protocol

LLC Logical link control

SNAP Subnetwork Access Protocol

CSMA/CD Carrier Sense Multiple Access with Collision Detection

ACK Acknowledgement - Potvrzení

LFI Link Fragmentation and Interleaving - Fragmentace a prokládání paketů

ARP Address Resolution Protocol

DOM Document Object Model

SAX Simple API for XML

SQL Structured Query Language - Standardizovaný dotazovací jazyk

SAX Simple API for XML

SEZNAM PŘÍLOH

A UKÁZKY DATABÁZOVÝCH ZÁZNAMŮ

Některé tabulky jsou pro vysázení do dokumentu příliš veliké, proto byly rozděleny na více částí pod sebou. Tabulky, které jsou mezi sebou pravázány klíčem jsou vysázeny do stejné tabulkové kapitoly.

A.1 ARP

Tab. A.1: Ukázka záznamů ARP

Primary_key	fromId	toId	reception_range	fbTx
1	1	5	2.179481523	2.003814697
2	5	1	9.172945388	2.004604736

lbTx	fbRx	lbRx	MAC_header	LlcSnapHeader
2.003814697	2.003814704	2.003814704	QOSDATA	type 0x806
2.004604736	2.004604766	2.004604766	QOSDATA	type 0x806

ARP_type	ARP_source_MAC	ARP_source_IP
request	00-06-00:00:00:00:00:01	10.1.3.1
reply	00-06-00:00:00:00:00:01	10.1.3.6

ARP_destination_MAC	ARP_destination_IP
-	10.1.3.6
00-06-00:00:00:00:00:01	10.1.3.1

A.2 CTL_ACK

Tab. A.2: Ukázka záznamů CTL_ACK

Primary_key	fromId	toId	reception_range	fbTx
1	7	3	5.385550164	0.004007021
2	5	0	10.21500399	0.004007045

lbTx	fbRx	lbRx	MAC_header
0.004007021	0.004007066	0.004007066	CTL_ACK
0.004007045	0.004007087	0.004007087	CTL_ACK

A.3 DSDV

Tab. A.3: Ukázka záznamů DSDV

Primary_key	fromId	toId	reception_range	fbTx
1	6	9	4.573164289	0.001189016
2	3	4	6.664288917	0.002115037

lbTx	fbRx	lbRx	MAC_header	LlcSnapHeader
0.001189016	0.001189022	0.001189022	QOSDATA	type 0x800
0.002115037	0.002115072	0.002115072	QOSDATA	type 0x800

IPv4_protocol	IPv4_length	IPv4_source	IPv4_dst	UDP_length
UDP	40	10.1.3.9	10.1.3.255	20
UDP	148	10.1.3.4	10.1.3.255	128

A.3.1 DSDV_header

Tab. A.4: Ukázka záznamů DSDV_header

Primary_key	Counter	Foreign_key	DSDV_IP	Hop	Seq
1	1	1	10.1.3.9	2	2
2	2	2	10.1.3.8	2	2
3	3	2	10.1.3.9	1	2
4	4	2	10.1.3.4	1	4

A.4 FTP

Tab. A.5: Ukázka záznamů FTP

Primary_key	fromId	toId	reception_range	fbTx
1	0	5	5.179548523	3.547162972
2	0	2	3.326945388	3.566947194

lbTx	fbRx	lbRx	Application_layer	MAC_header
3.547162972	3.547162995	3.547162995	FTP	QOSDATA
3.566947194	3.566947217	3.566947217	FTP	QOSDATA

LlcSnapHeader	IPv4_protocol	IPv4_length	IPv4_source
type 0x800	TCP	576	10.1.3.1
type 0x800	TCP	576	10.1.3.1

IPv4_destination	TCP_sequence	TCP_Ack	TCP_Win
10.1.3.6	1	1	65535
10.1.3.6	537	1	65535

A.4.1 FTP_payload

Tab. A.6: Ukázka záznamů FTP_payload

Primary_key	Foreign_key	TCP_Payload_Fragments
1	1	[0:536]
2	1	[536:1072]
3	2	[1072:1400]
4	2	[0:208]

A.5 FTP_ACK

Tab. A.7: Ukázka záznamů FTP_ACK

Primary_key	fromId	toId	reception_range	fbTx
1	5	2	5.179548523	3.446365991
2	5	0	3.326945388	3.746255824

lbTx	fbRx	lbRx	Application_layer	MAC_header
3.446365991	3.446366023	3.446366023	FTP	QOSDATA
3.746255824	3.746255855	3.746255855	FTP	QOSDATA

LlcSnapHeader	IPv4_protocol	IPv4_length	IPv4_source
type 0x800	TCP	40	10.1.3.1
type 0x800	TCP	40	10.1.3.6

IPv4_destination	TCP_sequence	TCP_Ack	TCP_Win
10.1.3.6	0	1	65535
10.1.3.1	1	537	65535

A.6 TOPOLOGY

Tab. A.8: Ukázka záznamů Topology

Primary_key	minX	maxX	minY	maxY
1	-3.1	29.9	-2.9	24.1
2	-3.1	29.9	-2.9	24.1
3	-3.1	29.9	-2.9	24.1
4	-3.1	29.9	-2.9	24.1

A.6.1 NODE

Tab. A.9: Ukázka záznamů Node

Primary_key	Foreign_key	Id	descr	locX	locY
1	1	4	STA	14.6235	12.58
2	2	6	STA	11.293	12.58
3	3	0	STA	8.78727	9.24233
4	3	1	STA	16.7859	8.74932

A.7 PACKET

Tab. A.10: Ukázka záznamů Packet

Primary_key	fromId	toId	aux	fbTx	lbTx	fbRx	lbRx
1	0	0	DummyPkt	1.4846	1.4846	1.4846	1.4846
2	0	0	DummyPkt	1.5	1.5	1.5	1.5
3	0	0	DummyPkt	1.5589	1.5589	1.5589	1.5589
4	0	0	DummyPkt	1.5821	1.5821	1.5821	1.5821

A.8 VoIP

Tab. A.11: Ukázka záznamů VoIP

Primary_key	fromId	toId	reception_range	fbTx
1	0	4	5.179548523	3.547162972
2	0	9	3.326945388	3.566947194

lbTx	fbRx	lbRx	Application_layer	MAC_header
3.547162972	3.547162995	3.547162995	VoIP	QOSDATA
3.566947194	3.566947217	3.566947217	VoIP	QOSDATA

LlcSnapHeader	IPv4_protocol	IPv4_length	IPv4_source
type 0x800	UDP	528	10.1.3.1
type 0x800	UDP	528	10.1.3.1

IPv4_destination	UDP_length	Payload_size
10.1.3.6	508	500
10.1.3.6	508	500

A.9 RESULTS

Tab. A.12: Výsledné hodnoty parametrů simulace

zpozdeni_FTP	zpozdeni_VoIP	jitter_FTP	jitter_VoIP
0.000000031269	0.000000030799	0.000000008966	0.000000011359

rychlost_FTP	rychlost_VoIP
609.088	905.6

B OBSAH CD

Jednotlivé soubory jsou samostatně v odpovídajících složkách.
Obsahem přiloženého CD jsou tyto soubory:

- Hlavní dokument-PDF soubor s bakalářskou prací
- Zdrojový kód PHP aplikace
- Analyzovaný XML dokument
- Kompletní exportovaná databáze