PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Simulace protokolu BGP v simulátorech NS

2014

Milan Hümmer

Anotace

Síťové simulace jsou přínosným nástrojem nejenom v oblasti projektování počítačových sítí, ale především pomůckou k pochopení a ověření chování komunikačních protokolů. Bakalářská práce obsahuje základní popis teorie simulací, na kterou navazuje popis síťových simulátorů NS-2 a NS-3. Dále součástí této práce je popis směrovacího protokolu BGP, který je poté použit při simulaci počítačové sítě v simulátoru NS-3. Vyhodnocení simulace obsahuje popis chování protokolu BGP při zřizování vazeb mezi směrovači, předávání informací o dostupnosti sítí či aktualizaci směrovacích tabulek.

Děkuji vedoucí bakalářské práce paní doc. Ing. Lence Motyčkové, CSc. za užitečnou metodickou pomoc, připomínky a trpělivost. Dále bych chtěl poděkovat manželce Soni za trpělivost a podporu.

Obsah

Seznam obrázků

1. Úvod

Jedním z hlavních témat této bakalářské práce, je problematika simulace počítačových sítí. V dnešní době, kdy si bez počítačových sítí nedokážeme představit svět, kdy projektanti počítačových sítí musí navrhovat takové řešení sítí, aby byly schopny zvládnout na tyto sítě rychle rostoucí nároky, jsou simulace a modelování počítačových sítí ekonomicky nejlepším řešením.

Druhým hlavním tématem bakalářské práce je popis a simulace směrovacího protokolu BGP. Internet, který je součástí většiny naší společnosti, je rozdělen na několik částí, kterým se říká autonomní systémy. Protokol BGP je v současné době standardem pro směrování mezi těmito autonomními systémy.

Bakalářská práce je rozdělena na dvě části, na část teoretickou a praktickou. V teoretické části se zabývá popsáním protokolu IP, rozdělením a typy směrovacích protokolů, dále se zabývá protokolem UDP a protokolem TCP, popisuje princip navazování a ukončení spojení pomoci tohoto protokolu. Součástí teoretické části je popsání základních vlastností protokolu BGP.

Další část části teoretické se zabývá teorií simulací, za kterou následuje popis síťového simulátoru NS-2, který zatím podporuje větší množství protokolů než simulátor NS-3, o kterém pojednává poslední část části teoretické.

Praktická část se zabývá instalací simulátoru NS-3 s nástrojem pro přímé vykonávání kódu (DCE) a balíčkem Quagga, dále popisem vytvářeného simulačního skriptu s využitím směrovacího protokolu BGP a následným spuštěním. Poslední kapitola části praktické se zaměřuje na rozbor výsledku simulace, především na chování protokolu BGP, na navazování spojení, výměnu zpráv, aktualizaci směrovacích tabulek a chování při výpadku linky v průběhu přenosu informací.

2. Internet protokol

Internet protokol je v dnešní době nejpoužívanější komunikační protokol. Je základním protokolem pracujícím na síťové vrstvě používaným v počítačových sítích a Internetu [11]. Protokol neposkytuje záruky na přenos dat a rozlišuje pomocí IP adresy pouze jednotlivá síťová rozhraní. Existují dvě varianty protokolu IP:

- IP verze 4
- IP verze 6

Trvalejším řešením nedostatku IP adres verze 4 je nová verze protokolu IPv6, která má větší adresový rozsah (2^{128} adres). IPv6 je stále využívána jen menšinou zařízení připojených k internetu, ale postupně probíhá přechod na tento protokol.

2.1. Internetový protokol verze 4

IP protokol dopravuje data mezi dvěma libovolnými počítači či zařízeními v internetu, tj. i přes mnohé LAN. Data jsou od odesílatele k příjemci dopravována (směrována) přes směrovače. IP protokol je protokol, který umožňuje spojit jednotlivé lokální sítě do celosvětového Internetu. Protokol IPv4 poskytuje omezený adresní prostor - teoreticky 2^{32} adres.

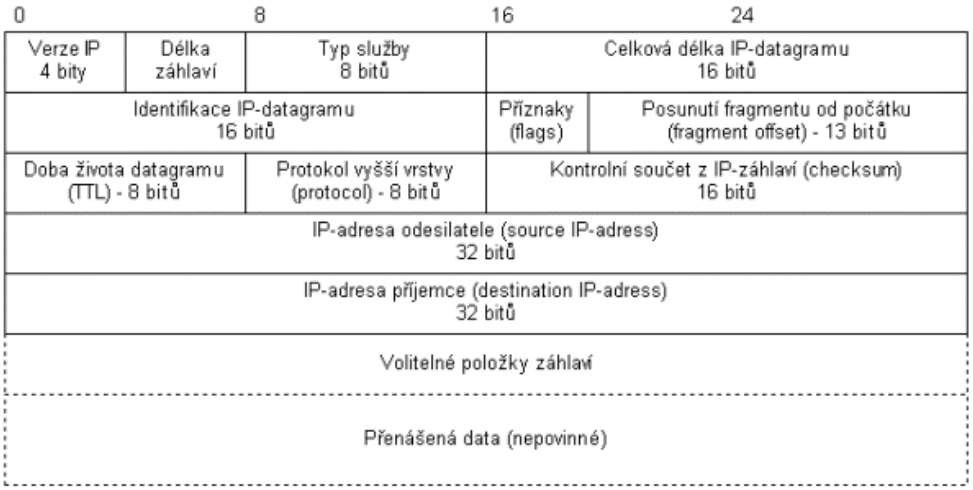
2.1.1. Paket IPv4

Jedná se o základní jednotku přenášených dat. Skládá se ze záhlaví a přenášených dat, viz obrázek 1. Celková délka paketu včetně záhlaví je vždy násobkem 32 bitů. Maximální délka paketu je 65535 oktetů [12]. IP paket je na úrovni síťového rozhraní vždy zabalen do rámce příslušné technologie (Ethernet, PPP, WiFi, atd.), který se mění, tak jak paket prochází přes dílčí sítě. [8]

2.2. Směrování

Směrování představuje proces hledání cest z jednoho bodu do jiných bodů v rámci propojených sítí. Směrování zajišťují směrovače (routery), ale i koncové stanice (při vysílání) a jejich úkolem je doručit paket adresátovi, pokud možno co nejefektivnější cestou.

Každé síťové rozhraní, komunikující prostřednictvím protokolu IP, má přiřazený jednoznačný identifikátor - IP adresu. Na základě IP adres se ve směrovačích vytváří a udržuje směrovací tabulka. Směrovací tabulka obvykle obsahuje záznamy o cestách do různých sítí [12]. Podle vzniku záznamu ve směrovací tabulce hovoříme o statickém, nebo dynamickém směrování.



Obrázek 1. Fornát IPv4 paketu

Statické - směrovací tabulka je plněna ručně administrátorem, který musí přesně určit, co a kudy se bude směrovat. Nevýhodou je nulová dynamika systému a nefunkčnost v případě poruchy v síti, nebo jakékoliv změny. Výhodou nulový provoz v síti v souvislosti s výměnou směrovacích informací. Běžně se používá pouze v malých sítích a konfiguraci koncového směrovače. [8]

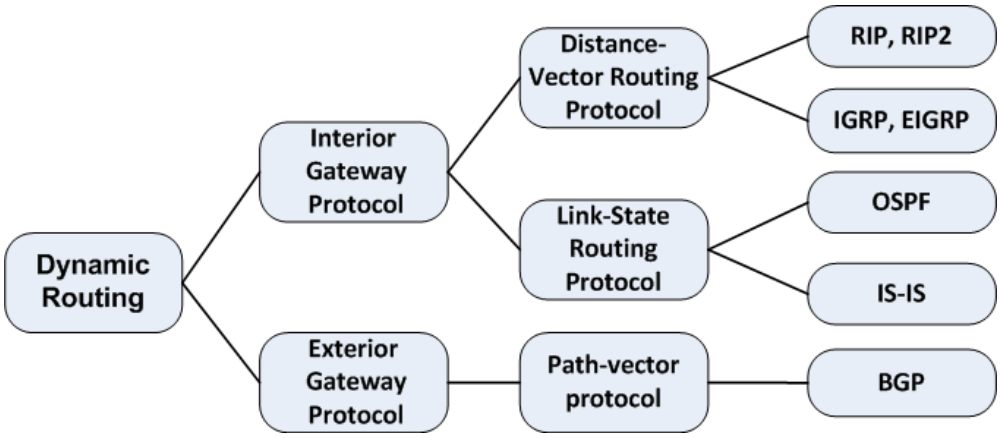
Dynamické - směrovací tabulka je naplněna automaticky, směrovače si dynamicky vyměňují směrovací informace na základě směrovacích protokolů.

2.2.1. Dynamické směrování

Dynamické směrování má za úkol nalézt pro paket procházející sítí nejlepší cestu do sítě cílové. Každý směrovací protokol má dvě komponenty. *Algoritmus*, což jsou definované kroky, kterými protokol postupuje při plnění obsahu směrovací tabulky, kdy vybere nejlepší z cest, a *zprávy*, údajové struktury, kterými sousední směrovače komunikují, aby si navzájem oznámily informace o dostupných sítích a cestám k nim.

Směrovací protokoly je možné definovat podle různých kritérií. Podle prostředí, ve kterém pracují se dělí do dvou skupin:

- *vnitřní směrovací protokoly - Interior Gateway Protocols (IGP)*, které se používají uvnitř autonomních systémů.
- *vnější směrovací protokoly - Exterior Gateway Protocols (EGP)*, které se používají na směrování mezi autonomními systémy.



Obrázek 2. Dělení dynamických protokolů

Podle principu činnosti se vnitřní směrovací protokoly dále dělí na:

- *Distance-Vector Routing Protocols* - hledají nejlepší cestu do vzdálené sítě na základě vzdálenosti. Trasa do sítě, která obsahuje nejnižší počet přeskoků, je považována za optimální. Vektor udává směr do vzdálené sítě. Mezi protokoly s vektorovou vzdáleností patří protokoly RIP, IGRP, EIGRP. Upraveným typem protokolu s vektorovou vzdáleností je *path-vector protocol*, do kterého patří BGP protokol, o kterém pojednává kapitola 4. [13]
- *Link-State Routing Protocols* - také se označují jako protokoly algoritmu nejkratší cesty (Shortest-Path-First Protocol), tzv. Dijskrův algoritmus. Každý směrovač vytváří tři tabulky. Jedna s těchto tabulek sleduje přímo připojené sousedy, druhá určuje topologii celé datové sítě a třetí slouží jako směrovací tabulka. Směrovače se stavem linky mají o datové síti více informací než směrovače, které pracují se směrovacím protokolem o vektorové vzdálenosti. Mezi protokoly se stavem linky patří protokoly OSPF a IS-IS. [13]

3. UDP a TCP protokoly

3.1. TCP(Transmission Control Protocol)

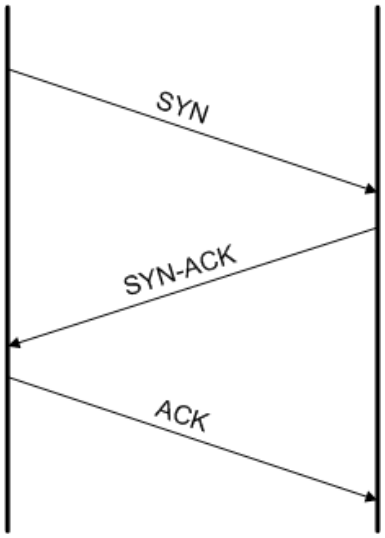
TCP je protokolem transportní vrstvy referenčního modelu ISO/OSI, nebo architektury TCP/IP. TCP je spojovou službou, tj. službou, která mezi dvěma aplikacemi naváže spojení - vytvoří na dobu spojení virtuální okruh. Tento okruh je plně duplexní. Přenášené bajty jsou číslovány. Ztracená nebo poškozená data jsou znovu vyžádána. Integrita přenášených dat je zabezpečena kontrolním součtem. [7]

3.1.1. Navazování a ukončení spojení v TCP

Aby se mohla vysílat data pomocí TCP protokolu, je nejprve třeba vytvořit spojení. Pro navázání spojení se používá třícestný handshake (potřesení ruky). V průběhu navazování spojení se obě strany dohodnou na čísle sekvence (sequence number). Číslo sekvence a odpovědi (sequence, acknowledgement number) jsou 32bitové hodnoty uváděné v TCP hlavičce. Pro navázání spojení se posílá TCP segment, který má nastaveny příznaky (flags) v TCP hlavičce. Jedná se o 8 bitových hodnot CWR (Congestion Window Reduced), ECE (ECN-Echo), URG (Urgent), ACK(Acknowledgement), PSH (Push), RST (Reset), SYN (Synchronize), FIN. [15]

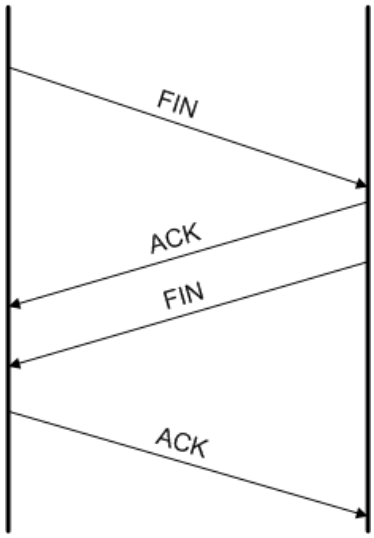
Navázání spojení probíhá ve třech krocích:

1. klient pošle SYN paket s uvedeným číslem sekvence (x), číslo odpovědi 0
2. druhá strana si uloží číslo sekvence (x) a odpoví SYN-ACK, jako číslo sekvence nastaví svoje číslo (y) a do čísla odpovědi vloží (x+1) - další očekávanou hodnotu
3. klient odpoví ACK, číslo sekvence (x+1), číslo odpovědi (y+1)



Obrázek 3. Navazování spojení protokolem TCP

Principy při ukončení spojení jsou podobné jako při jeho navazování. Nejčastěji se používá čtyřcestný handshake, kdy každá strana samostatně uzavře spojení. Zde se používá sekvence FIN s odpovědí ACK. [15]



Obrázek 4. Ukončení spojení protokolem TCP

3.2. UDP(User Datagram Protocol)

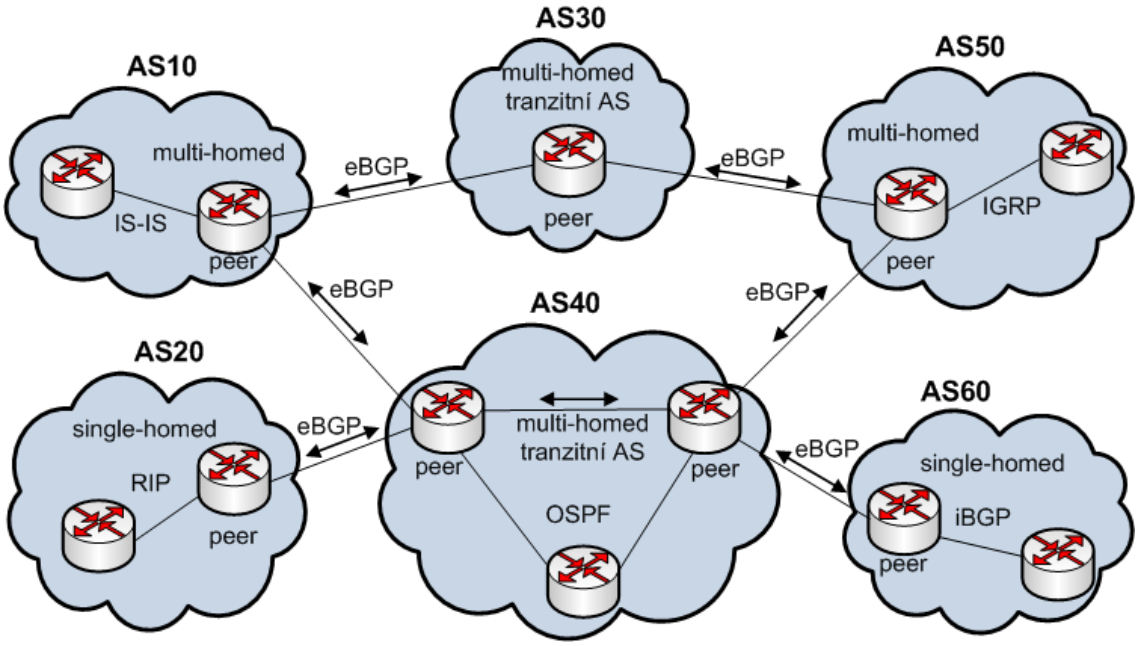
UDP je protokolem transportní vrstvy referenčního modelu ISO/OSI, nebo architektury TCP/IP. UDP je jednoduchou alternativou protokolu TCP. UDP je nespojovaná služba, tj. nenavazuje spojení. Odesílatel odešle UDP datagram příjemci a už se nestará o to, zdali se datagram náhodou neztratil. [7] Svými jednoduchými vlastnostmi se protokol UDP hodí pro aplikace vyžadující rychlý přenos.

4. Směrovací protokol BGP(Border Gateway Protocol)

4.1. Autonomní systémy

Současný internet je natolik rozsáhlý a proměnlivý, že není reálné udržovat ve směrovačích úplnou informaci o jeho topologii. Proto bylo rozhodnuto směrování v Internetu řešit hierarchickým způsobem a byl rozdělen do tzv. autonomních systémů (AS). Autonomním systémem rozumíme souvislou skupinu sítí a směrovačů, které jsou pod společnou správou a řídí se společnou směrovací politikou. Při směrování v rámci jednotlivých AS se používají vnější směrovací protokoly (EGP) [9].

AS mohou být z topologického hlediska rozděleny do dvou kategorií,*single-homed* a *multi-homed*. Single-homed AS je připojen jedinou linkou k jinému AS,



Obrázek 5. Směrování v autonomních systémech

zatím co multi-homed je připojen více linkami. AS může být také *tranzitní* AS, takový systém dovoluje průchod provozu, který v něm nezačíná, ani nekončí [9].

4.2. Protokol BGP

BGP je dynamický směrovací protokol typu EGP (Exterior Gateway Protocol) [8] [9] [10], využívaný ke směrování v Internetu mezi autonomními systémy (AS). BGP je založen na výměně informací mezi BGP sousedy tvz. *peer* směrovače, spojení mezi těmito směrovači je nakonfigurováno ručně a pro přenos se využívá spolehlivý protokol TCP. Tento směrovací protokol patří mezi tzv. Path-vector protokoly, kde obsah směrovacích informací uchovává cenu cesty a zároveň i všechny předešlé skoky jako seznam AS, jimiž cesta prošla (k detekci smyček). BGP podporuje přenos směrovacích informací o sítích s beztřídní maskou.

AS svým sousedním AS prostřednictvím protokolu BGP sděluje k jakým IP sítím je schopen doručit IP pakety. Okolní AS se, podle své nastavené směrovací politiky, rozhodnou, jestli použijí konkrétní AS pro směrování k dané síti.

BGP může být využito i ke směrování v rámci AS, takovýto BGP se nazývá vnitřní (internal) - iBGP.

4.3. Zprávy protokolu BGP

Existují 4 typy zpráv, které si sousedé mohou mezi sebou vyměňovat. Zpráva

protokolu BGP může mít maximální délku až 4096 B.

Typy zpráv:

- OPEN - zpráva vyměňována při zřizování vazby mezi sousedními směrovači. Přenáší se v ní informace o AS, identifikace směrovače, apod. Musí být potvrzena zprávou KEEPALIVE.
- UPDATE - zpráva nesoucí aktualizaci směrovacích informací. Aktualizace může obsahovat informace o zrušení, nebo o vzniku nových cest.
- KEEPALIVE - zpráva posílána pravidelně pro ověřování funkčnosti spojení mezi sousedy. Spojení se považuje za nefunkční, pokud od souseda nepřišla zpráva KEEPALIVE. Ve výchozí konfiguraci se vysílá každých 60 sekund.
- NOTIFICATION - zpráva používaná pro ukončení vazby mezi sousedy rozpojením TCP spojení. Je zasílána v případě chyby, při absenci zprávy KEEPALIVE, nebo jiné mimořádné situace.

4.4. Atributy cesty

Vyhledávání pomocí algoritmu path-vector umožní nalézt nejkratší cesty do všech AS [9]. Abychom byli schopni explicitně ovlivňovat směrovací politiky, je potřebný mechanismus, kterým bychom vyjádřili preferenci, resp. zakázali některé cesty podle na základě různých kritérií. K tomuto účelu v BGP protokolu slouží tzv. atributy, které můžeme každému záznamu o cestě k cílové síti přiřadit. Atributy jsou přenášeny ve zprávě UPDATE. Je definováno několik atributů, z nichž některé jsou povinné a některé nepovinné.

Vybrané typy atributů [8]:

- ORIGIN - říká od kterého původce směrovací informace pochází. Přenášenou hodnotou jsou číselné kódy reprezentující buď IGP (Interior Gateway Protocol), EGP protokol, nebo INCOMPLETE (původ cesty není znám). Hodnota by neměla být měněna žádným jiným směrovačem.
- AS_PATH - cesta k cílové síti, seznam AS kterými zpráva pošla. Pokud se cesta dostane do AS, jehož číslo je již v seznamu uvedeno, cesta se ignoruje. Tímto způsobem se zabrání vzniku smyčky.
- NEXT_HOP - unicastová IP adresa dalšího skoku, který by měl být použit při směrování k cílové síti. Hodnota se při přenosu zprávy zpravidla mění, tak jak prochází mezi AS.
- LOCAL_PREF - volitelný atribut, na základě kterého se rozhoduje, který hraniční směrovač v rámci AS bude preferován, pokud do cílového AS vede více cest.

- MULTI_EXIT_DISC - volitelný atribut kterým lze ovlivnit volbu cesty používanou sousedním AS. V případě existence více cest k jedné síti si směrovač vybere tu s nejnižším výstupním diskriminátorem (MED).
- WEIGHT - jedná se o atribut definovaný firmou CISCO, který používá hodnotu váhy k určení nejlepší cesty (vyšší hodnota má vyšší prioritu), definuje se pouze pro lokální směrovač.

4.5. Směrovací proces protokolu BGP

U IGP protokolů rozhodují o procesu směrování údaje jako rychlost linky [8], vzdálenost, počet skoků apod. Při směrováni mezi AS vystupují i jiné požadavky. Směrovací tabulky obsahují stovky tisíc záznamů nejdůležitějším kritériem nebývá jenom vzdálenost, ale posuzují se nastavitelné parametry například jako cena přenosu a dodatečná pravidla aplikovaná v závislosti na zdroji, cíli seznamu tranzitních autonomních systémů a dalších atributech. Rozhodnutí o politice směrování je často závislé na administrátorovi.

Směrovací informace si uchovává každý BGP směrovač v tabulkách, nazývajících RIB (Routing Information Base) [8] [10]. Tyto tabulky jsou tří druhů:

- Adj-RIBs-In (Adjacent Routing Information Base - Incoming) obsahuje směrovací informace od svého souseda přijaté v příchozích zprávách UPDATE. Informace obsaženy v této tabulce jsou následně využity v rozhodovacím procesu směrovače.
- Loc-RIB (Local Routing Information Base) obsahuje informace o cestách, které byly vybrány z tabulky Adj-RIBs-In na základě směrovacích politik. Tyto informace jsou lokálně využívány BGP směrovačem.
- Adj-RIBs-Out (Adjacent Routing Information Base - Outgoing) tabulka, kde se nachází průběžně připravované informace, které byly vybrány k rozeslání k sousedním BGP směrovačům zprávou typu UPDATE.

K rozhodnutí o směrování využívá protokol BGP tyto parametry [8]:

- druh cesty k dané síti - počet procházených AS - parametr AS_PATH
- skupina pravidel definovaných administrátorem:
  - váha (weight)
  - místní preference (local preference) - parametr LOCAL_PREF
  - výstupní diskriminátor - MED - parametr MULTI_EXIT_DISC

5. Simulace počítačových sítí

5.1. Modelování

Modelování je zjednodušená reprezentace skutečného systému. Modelování je důležité v návrhu systému a vývoje, protože dává představu o tom, co by systém provedl, pokud by byl skutečně realizován.

Systémové parametry v modelování mohou být měněny, zkoušeny a analyzovány. Správné použití a zpracování modelování může ušetřit náklady rozvoje reálného systému.

Existují dva přístupy modelování: analytický a simulační přístup. [2]

5.1.1. Analytický přístup

Obecné pojetí analytického modelování spočívá v nalezení způsobu jak popsat systém matematicky pomocí použití matematických nástrojů, jako je například teorie pravděpodobnosti, a poté aplikovat numerické metody k získání přehledu z rozvinutého matematického modelu. [2]

5.1.2. Simulační přístup

Simulace je široce používáno v systému modelování pro aplikace například v strojírenském výzkumu, obchodní analýze, plánování výroby, nebo experimentální biologii. Ve srovnání s analytickým modelováním simulace obvykle vyžaduje v modelu méně abstrakce, v simulačním modelu může být uveden téměř každý možný detail systému a tak co nejlépe popsat skutečný systém. [2]

5.2. Simulace

Podle R. E. Shannona [3] je simulace proces navrhování modelu reálného systému a provádění experimentů s tímto modelem za účelem pochopení chování systému a nebo vyhodnocování strategie pro provoz systému. [2]

5.2.1. Prvky simulace

Podle R. G. Ingallse [4] se stavební dílce simulace skládají z následujících kroků:

Subjekty

Subjekty jsou objekty, které se vzájemně ovlivňují a způsobují v simulačním programu některé změny stavu systému.

V rámci počítačové sítě mohou subjekty zahrnovat počítačové uzly, pakety, toky paketů, nebo nefyzické předměty jako jsou simulační hodiny.

Prostředky

Prostředky jsou součástí komplexních systémů. Obecně platí, že omezený zdroj prostředků musí být sdílen určitou množinou subjektů. To je obvyklé pro případ počítačové sítě, kde například šířka pásma, vysílací čas, počet serverů představují síťové prostředky, které mají být sdíleny mezi subjekty sítě.

Aktivity a události

Čas od času, se subjekty zapojí do některých aktivit. Toto zapojení vytváří události a spouští změny v systému. Mezi běžné příklady aktivit patří zpoždění, vytváření front.

Plánovač

Plánovač udržuje seznam událostí a čas jejich provedení. Během simulace spustí simulační hodiny, vytváří události a provádí je.

Globální proměnné

V simulaci jsou globální proměnné dostupné z jakékoliv funkce nebo subjektu v systému a v podstatě udržují některé společné hodnoty simulace. V rámci počítačové sítě, by mohly takové proměnné představovat například délku fronty paketů, celkové obsazení vysílacího času v bezdrátové síti, nebo celkový počet paketů.

Generátor náhodných čísel

Je nutné zavést náhodnost do simulačního modelu.
V simulaci počítačové sítě, například proces doručení paketu, proces čekání a proces služby jsou obvykle modelovány jako náhodné procesy.

Statistický sběrač

Hlavní odpovědností statistického sběrače je sběr dat generovaných během simulace tak, aby bylo z těchto údajů možno vyvodit smysluplné závěry. [2]

5.2.2. Časově závislá simulace

Hlavním typem simulace je simulace závislá na čase, která probíhá chronologicky. Tento typ simulace si udržuje simulační hodiny, které udržují přehled o aktuálním čase simulace. Ve většině případů simulace běží, dokud hodiny nedosáhnou předdefinované hodnoty.

Časově závislá simulace může být dále rozdělena *časově řízenou simulaci* a *událostně řízenou simulaci.*

Časově řízená simulace vyvolává a provádí události v daném časovém intervalu. Na druhé straně, událostně řízený simulátor vyvolává události v libovolném okamžiku. Simulace se přesunuje od jedné události k druhé a opět spustí událost (pokud existuje), dokud simulace neskončí. [2]

6. Network Simulator 2 (NS-2)

NS-2 je open-source událostně řízený simulátor, který se používá především v oblasti výzkumu. NS-2 poskytuje značnou podporu pro simulaci TPC, UDP, směrovacích unicastových i multicastových protokolů v drátových i bezdrátových (lokálních a satelitních) sítích. NS byl vyvíjen jako varianta REAL simulátoru vyvíjený na Kalifornské univerzitě v Berkeley od roku 1989. V roce 1995 byl vývoj podporován agenturou DARPA(Agentura pro výzkum pokročilých obranných projektů - Ministerstvo obrany USA). V současné době je NS podporován DARPA a NSF (Národní vědecká nadace - USA). [1]

NS-2 je napsaný ve dvou jazycích, C++ a OTcl. C++ definuje vnitřní mechanizmus, výhodou tohoto jazyka je rychlost při běhu programu, a nevýhodou je pomalost při vyhledávání chyb, změnách v kódu a rekompilaci, z tohoto důvodu bylo zavedení druhého programovacího jazyka.

Jazyk OTcl je objektově orientovaný jazyk, který vyšel z jazyka Tcl. Je to jednoduchý skriptovací jazyk, který se používá pro nastavení simulací, konfiguraci objektů a plánování diskrétních událostí. Výsledky simulace lze graficky interpretovat v nástrojích jako je NAM (Network Animator) nebo XGraph. [2]

6.1. Protokoly a modely podporované NS-2

V NS-2 je implementována celá řada protokolů a zahrnuje komunikaci v drátových, bezdrátových a satelitních sítích. [1]

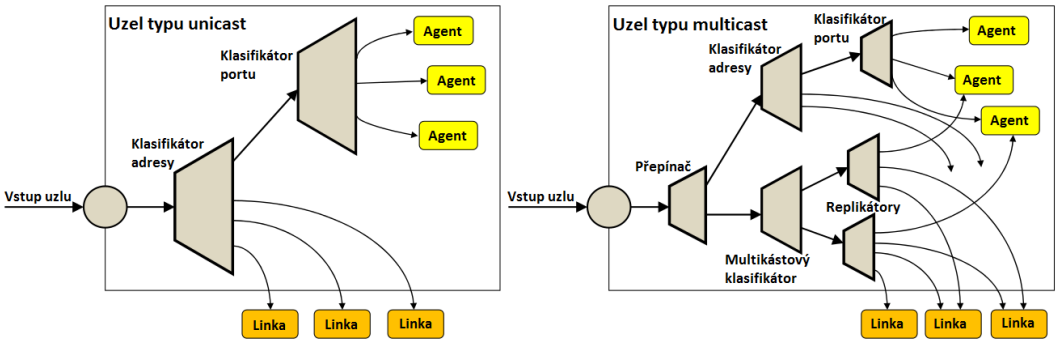
- Aplikační vrstva - telnet, HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), a další.
- Transportní vrstva - TCP, UDP, RTP, SRM
- Síťová vrstva - IPv4, ICMP, STP, ARP
- Práce s frontami - FQ (Fair Queueing), SFQ (Stachastic Fair Queueing), DRR (Deficit Round Robin), FIFO (First In First Out), RED (Random Early Discard), CBQ (Class Based Queuing)
- Linková vrstva - CSMA/CD, MAC
- Směrování - RIP, OSPF, BGP, atd.

6.2. Základní objekty modelu

Základní objekty, které NS-2 obsahuje [6] jsou uzly, linky, agenti a aplikace. Topologii sítě definují uzly a linky. Agenti reprezentují koncové body a vytvářejí, nebo zpracovávají pakety.

6.2.1. Uzel (Node)

Společně s linkou tvoří hlavní kostru v topologii simulované sítě. Uzel je složený objekt, který se skládá z dvou hlavních částí. Přístupu uzlu a klasifikátoru (adresový a portový). Klasifikátor adresy určuje, zda-li příchozí paket je adresován danému uzlu či nikoli, klasifikátor portu určuje na základě čísla portu, které aplikaci či agentu paket náleží. [6]

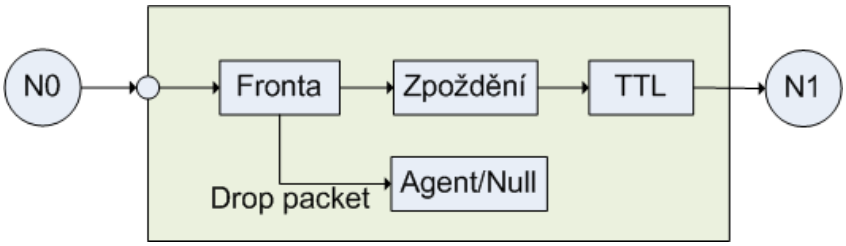


Obrázek 6. Unicastový a multicastový uzel

Struktura uzlu může být definována jako typu unicast, nebo typu multicast. Multicastový uzel obsahuje navíc klasifikátor multicastové adresy.

6.2.2. Linka

Linky tvoří spojení mezi uzly. Podle druhu spojení můžeme nadefinovat typ linky s příslušnými parametry. NS primárně podporuje simplexní spojení. [6]



Obrázek 7. Simplexní linka v NS

Pakety vstupují do fronty (Queue) na jejím výstupu je na ně aplikována simulace zpoždění (Delay) a ztráta paketu (Agent/null). Blok TTL (Time To Live) zjistí a aktualizuje parametr TTL v hlavičce průchozího paketu.

Pro linku je i možné nastavit velikost vstupní paměti, kde v případě přetečení se s pakety zachází podle zvoleného typu fronty (např. DropTail - zahazování paketů).

6.2.3. Agenti

Představují koncové body, kde se vytváří, nebo zpracovává paket síťové vrstvy a používají se při implementaci protokolů vyšších vrstev. Agenti jsou vždy definováni mezi dvěma uzly a ke každému zdrojovému agentu odpovídá určitý druh cílového. [6]

6.2.4. Aplikace

Představují datové simulační zdroje a jsou připojeny nad transportní vrstvou, kde se nachází agenti reprezentující transportní protokoly (TCP, UDP, SCTP,...)

6.3. Error model

ErrorModel [2] je modul, který ukládá chyby do přenosu paketů. Může být vložen mezi dva NS objekty. Simuluje chybu paketu po přijetí paketu. Pokud je simulováno, že paket má chybu, error model paket zahodí, nebo označí s příznakem chyby. Pokud na paketu není simulována chyba, error model předá paket navazujícímu objektu. Error model může být použit jak na drátových, tak na bezdrátových sítích.

7. Network Simulator 3 (NS-3)

NS-3 je událostně řízený síťový simulátor, zaměřený především na výzkum a vzdělávání, který může být použit k simulaci různých typů sítí. Projekt NS-3 byl zahájen v roce 2006, původně jako náhrada za NS-2 s počátečním důrazem na IEEE 802.11 WiFi modely.

Jádro NS-3 obsahuje sadu knihoven, které nabízí široký rámec nástrojů po vývoj síťových simulací. Softwarová struktura NS-3 dovoluje simulovat modely v realistických podmínkách. NS-3 simulační jádro podporuje výzkum jak na obou IP sítích, tak i v sítích bez IP protokolu. Obsahuje mnoho protokolů, které hlavně podporují WiFi, WiMAX, nebo sítě LTE, také obsahuje modely směrovacích protokolů jako je OLSR, nebo AODV. Nemá ale všechny modely, které obsahuje NS-2.

NS-3 je v současné době aktivně vyvíjen, každé 3 měsíce je vydána nová stabilní verze obsahující nové modely, které jsou zdokumentované a ověřené. Momentálně nejnovější verze je NS-3.20. [5]

7.1. Rozdíly mezi NS-2 a NS-3

NS-3 není rozšíření NS-2, je to nový simulátor, který není zpětně kompatibilní s NS-2, je vyvíjen také proto, aby NS-2 nahradil. Část NS-2 je psaná v C++ a část v OTcl, NS-3 je napsaný celý v C++. NS-2 simulační skripty jsou psány v skriptovacím jazyce OTcl a simulační skripty u NS-3 mohou být psány v C++, nebo v Pythonu, což má za následek jednoduššího psaní simulací, tím pádem i menší časovou náročnost. NS-3 se také pokouší vyřešit problémy vyskytující se v NS-2. Mezi velké výhody NS-3 patří emulační mód, který umožňuje integraci s reálnými sítěmi. [5]

7.2. Protokoly podporované v NS-3

V jádru NS3 je implementována celá řada protokolů, která jsou nezbytná ke správné komunikaci v síti. Mezi tyto protokoly patří:

- Transportní vrstva - UDP, TCP
- Síťová vrstva - IPv4, IPv6, globální statické směrování (unicast, multicast),OLSR, AODV, DSDV
- Linková vrstva - PPP, CSMA, 802.11 MAC
- Fyzická vrstva - 802.11a, základní nastavení kabelových linek (ztráta, zpoždění, kapacita linky)

Díky možnosti přímého vykonávání kódu, viz kapitola 7.6., lze tuto podporu rozšířit o další protokoly, jako například RIPng, OSPFv3, BGP-4. Jelikož je NS-3 stále vyvíjen a zdokonalován, počítá se s větší podporou protokolů, jako je tomu u NS-2.

7.3. Diskrétní simulace u NS-3

Podle teorie simulací je diskrétní simulace časově závislá, která se skládá z daných dílců, mezi které například patří:

- entity - zde patří třídy `Node`, `Packed` atd.
- prostředky - třída `Cannal` ,nastavení vysílacího času
- aktivity a události - nastavení zpoždění ve třídě `Cannal`, výpadek linky pomocí Topology Helperu IPv4
- plánovač - možnost nastavení času vysílání paketů, přerušení vysílání
- generátor náhodných čísel - aplikace, která generuje a přijímá pakety
- statický sběrač je možné použít několik typů několik typů sběračů, např. AsciiTraceHelper, nebo PcapHelper viz kapitola 7.5.

7.4. Základní objekty modelu

Základní objekty, které NS-3 obsahuje [5] jsou uzly, kanály, síťové rozhraní a aplikace. Topologii sítě definují uzly a linky. Aplikace vytvářejí, nebo zpracovávají pakety.

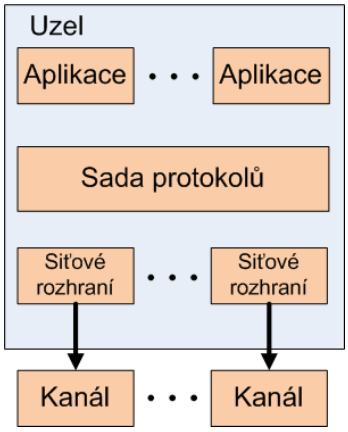
7.4.1. Uzel

V NS-3 je uzel základním abstraktním zařízením. Tato abstrakce je zastoupena v C++ třídou `Node`. Tato třída poskytuje metody pro řízení reprezentace výpočetních zařízení v simulaci.

Uzel si můžeme představit jako počítač, kterému přiřadíme tyto funkce:

- Aplikace
- Sady protokolů
- Síťové rozhraní

[5]



Obrázek 8. NS-3 uzel

7.4.2. Aplikace

Aplikací v se NS-3 nazývá základní abstrakce pro uživatelský program, který generuje nějakou činnost, která má být simulována. Tato abstrakce je v C++ zastoupená třídou `Application`.

Každý uzel může využívat služeb jedné, nebo více aplikací. Mezi aplikace například patří:

- Generátor provozu

- Směrovací agenti
- Vypínání a zapínání simulace

[5]

7.4.3. Kanál

Kanály spojují síťová rozhraní a umožňují uzlům mezi sebou komunikovat. Kanál může reprezentovat drátové spojení, či spojení přes optické vlákno. Specializovaný kanál může také modelovat složitější prostředí, jako je ethernetový přepínač, tří-dimenzionální prostor plný překážek v případě bezdrátové sítě.

V jazyku C++ je tato abstrakce zastoupena třídou `Channel`. [5]

7.4.4. Síťové rozhraní

V případě reálných sítí, pro připojení zařízení je nutné mít v zařízení síťovou kartu (NIC - Netwotk Interface Card). Pro správné fungování této karty je potřeba také softwarové podpory v podobě ovladače. V NS-3 abstrakce síťového rozhraní představuje jak softwarový ovladač, ale také i hardwarovou síťovou kartu. Abstrakce síťového rozhraní je v jazyku C++ zastoupena třídou `NetDevice`. Síťová rozhraní jsou instalována v uzlu, aby tento uzel mohl komunikovat s ostatními uzly v simulaci pomocí kanálů. Stejně jako v reálném počítači, může být uzel připojen k více než jednomu kanálu přes více síťových rozhraní. [5]

7.4.5. Topology Helpers

NS-3 obsahuje tzv. Toplogy Helpers pro zjednodušení konfigurace sítě. V reálné síti má každý síťový prvek MAC adresu, nastavenou IP adresu a nadefinovány parametry přenosu přes fyzické rozhraní. Pro zjednodušení vytváření rozsáhlé sítě v simulátoru NS-3, například při zadávání IP adres síťovým rozhraním, použijeme daný Topology Helper. Použitím Topolgy Helperu se můžeme vyhnout hůře odstranitelným chybám, jako je například špatné zadání IP adresy, kdy tomuto helperu (konkrétně u IPv4 adres `Ipv4AdressHelper`) určíme adresní rozsah zadáním IP adresy sítě, včetně masky. Topology Helper se postará o přidělení jednotlivých IP adres jednotlivým zařízením.

NS-3 obsahuje velkou sadu funkcí Topology Helper,které se starají o to, aby běžné úkony byli co nejjednodušší a nejrychlejší. [5]

7.5. Výstupní soubory a vizualizace

Pro sledování průběhu a sběru výsledku simulace obsahuje NS-3 několik funkcí, jako jsou například funkce AsciiTraceHelper, nebo PcapHelper [5]. Funkce AsciiTraceHelper slouží pro záznam událostí simulace ve formě textu, který se

ukládá do souboru typu txt. Funkce PcapHelper slouží pro záznam událostí simulace do souboru typu pcap. Pro analýzu výstupu simulace u tohoto typu souboru lze použít program Wireshark.

Wireshark - je nástroj pro sledování sítě a síťový analyzátor, tj. nástroj, který nejenom zachycuje pakety v síti, ale navíc umožňuje jejich analýzu, která zahrnuje jak pitvání datových paketů do největších detailů, tak vytváření nejrůznějších statistik a grafů. [7]

Samotný simulátor NS-3 neobsahuje žádné grafické rozhraní, kterým by bylo možné zobrazit návrh simulovaného návrhu modelu, ale jsou dostupné různé grafické nástavby. Nejpoužívanější grafický program je NetAnim.

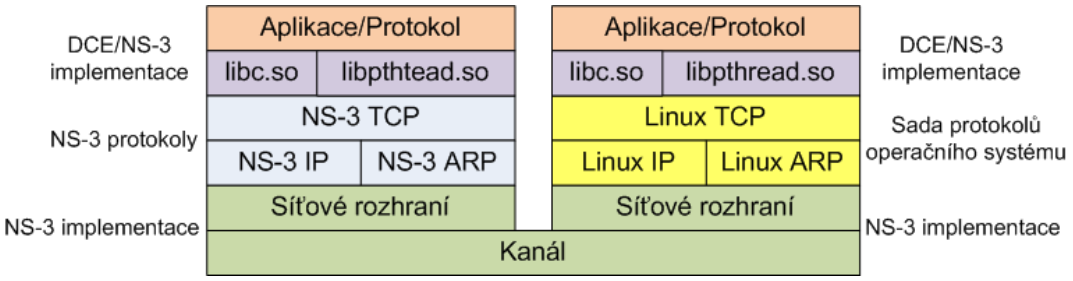
NetAnim - je offline animátor, zobrazuje animace simulace pomocí souboru XML, do kterého byly nashromážděny informace v průběhu simulace. Tento program umožňuje graficky zobrazit umístění uzlů, samotný průběh simulace v čase i přenos paketů mezi uzly.

7.6. Přímé vykonávání kódu

Direct Code Execution (DCE) je nástroj, který umožňuje při použití NS-3 využívat protokolové implementace operačního systému bez nutnosti zasahovat do zdrojového kódu. Podpora DCE ve spolupráci s NS-3 nabízí realističtější simulování sítí s přesnějšími výsledky, než starší simulátory.

DCE v NS-3 nabízí dva operační módy:

- Základní - používá sadu protokolů TCP/IP implementovanou v NS-3
- Pokročilý - používá sadu síťových protokolů implementovaných v jádru operačního systému (Linux)



Obrázek 9. Vrstvový model DCE pro základní a pokročilý operační mód v NS-3

Pro DCE byla umožněna spolupráce s dalšími aplikacemi, které umožňují větší využití simulátoru NS-3. Mezi takové aplikace patří aplikace Quagga. [5]

7.6.1. Quagga

Quagga je směrovací softwarový balík, poskytující implementace směrovacích protokolů OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP-4 a síťové protokoly IPv4 a IPv6 pro operační systémy Unix, Linux, Solaris.

Díky programu Quagga je možné v systému využívat démonů, zajišťující funkci směrování podobně jako směrovač. [5]

8. Praktická část

Tato část bakalářské práce se zaměřuje na instalaci simulátoru NS3 a dodatečných balíků DCE a Quagga, na vytváření modulu simulace se zaměřením na protokol BGP.

8.1. Instalace NS-3

NS3 je primárně vyvíjen pro platformy GNU/Linux. Pro použití NS3 v systému Windows je možné využít prostředí Cyrwin, které ale už není v dnešní době podporováno. Další alternativou je nainstalování virtualizačního nástroje, jako je například VirtualBox a nainstalování virtuálního operačního systému Linux.

Pro potřeby bakalářské práce je použita linuxová distribuce Ubuntu 13.10.

Před samotnou instalací simulátoru NS3 je potřeba nainstalovat požadované nástroje a balíčky:

- Mercurial - nástroj pro správu organizace a změn zdrojového kódu a dokumentace
- Waf - nástroj pro kompilaci a instalaci aplikací. Nahrazuje nástroj make, který není příliš vhodný pro použití velkých a vysoce konfigurovatelných systémů.Nástroj Waf je založený na jazyku Python.
- Vývojové prostředí - z důvodu použití jazyka C++ při psaní skriptů, je nutné nainstalovat kompilátor pro C++

Tyto základní balíčky získáme zadáním příkazu v terminálu:

```
sudo apt-get install gcc g++ python python-dev mercurial
```

Z důvodu podpory protokolu BGP programem Quagga, bude dále popsána instalace simulátoru NS3 včetně DCE-Quagga.

DCE s podporou Quaga vyžaduje několik balíčků: autoconf, automake, flex, git-core, wget, g++, libc-dbg, bison, indent, pkgconfig, libssl-dev, libsysfs-dev, gawk.

Při instalaci NS3 DCE Quagga, můžeme využít instalačního nástroje Bake, vyvinutého pro projekt NS3. Nejprve si stáhneme a nainstalujeme nástroj Bake pomocí příkazu `hg` programu Mercurial do složky bake.

```
hg clone http://code.nsnam.org/bake bake
```

Definujeme proměnné, pro možnost použití Bake mimo adresář, do kterého byl nainstalovaný.

```
export BAKE_HOME=‘pwd‘/bake
export PATH=$PATH:$BAKE_HOME
export PYTHONPATH=$PYTHONPATH:$BAKE_HOME
```

V dalším kroku si nainstalujeme simulátor ns-3-dce-quagga do složky, kterou si vytvoříme. Po zadání `-e dce -linux-|verze|` v parametrech `bale.py configure`, bude nainstalována verze, která bude používat protokolové implementace linuxového systému. Pokud bychom chtěli používat protokoly ns3 simulátoru, jako parametry zadáme `-e dce -ns3-|verze|`.

```
mkdir dce
cd dce
bake.py configure -e dce- linux-|verze| -e dce-quagga-|verze|
bake.py download
bake.py build
```

Po úspěšné instalaci je vhodné provést kontrolu programu, zda byl správně sestaven. Kontrolu provedeme příkazem:

```
cd source/ns-3-dce ./test.py -s dce-quagga
```

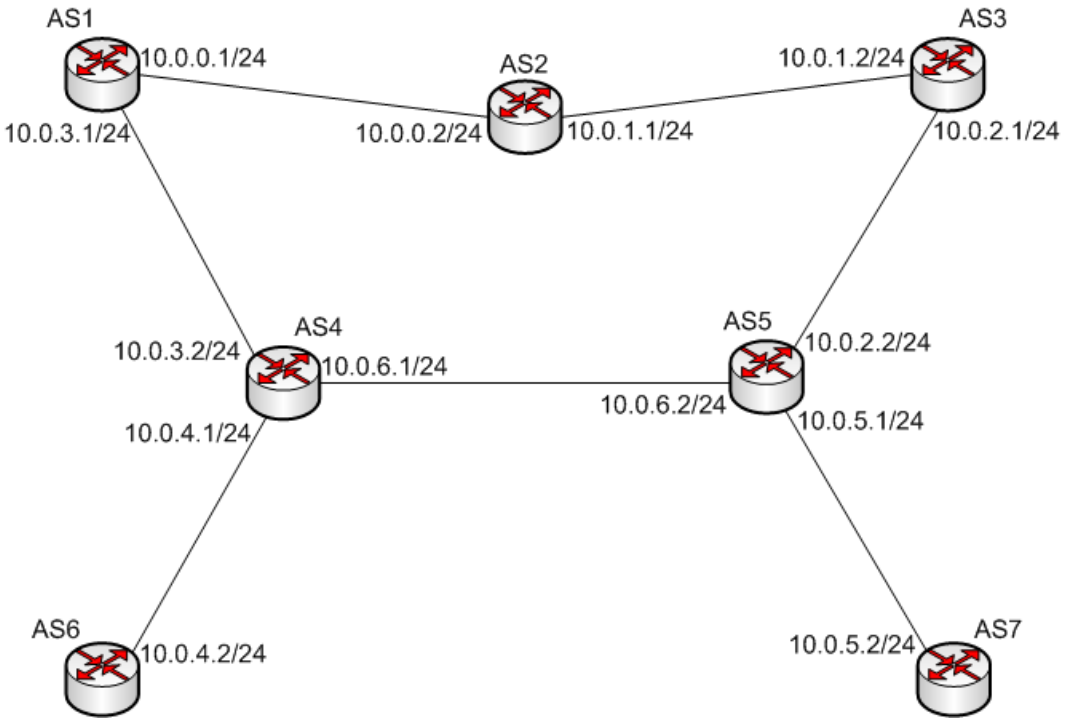
[14]

8.2. Vytváření skriptu simulace

Na obrázku č. 10. můžeme vidět návrh sítě, kterou použijeme při simulaci. Skládá se ze sedmi uzlů, které budou simulovat peer směrovače jednotlivých autonomních systémů. Nejdůležitější uzly - peer směrovače jsou AS3 a AS6. Směrovači AS3 přidělíme roli klienta, který bude vysílat data v podobě UDP paketů směrovači AS6, kterému přidělíme roli serveru. Velikost přenášených dat v jednom paketu bude nastavena na 1024 bytů. Použitým směrovacím protokolem bude BGP. Délka trvání simulace bude 300s, klient začne vysílat UDP pakety po 100 sekundách běhu simulace, délka vysílání bude 30s. Linky spojující uzly budou typu poin-to-point, každé spojení uzlů se bude nacházet v samostatné síti. Všechny linky budou mít nastavenou stejnou kapacitu a to 5Mb/s a zpoždění na linkách bude nastaveno na 2ms.

Síť byla navržena takovým to způsobem, abychom mohli ověřit funkčnost směrování, výpadek při provozu a nalezení alternativní cesty k cílovému uzlu.

Simulaci budeme provádět, aby jsme pomocí nastavených událostí zjistili a ověřili, jaké je chování protokolu BGP při navazování spojení a odesílání aktualizací pro směrovací tabulky pomocí zpráv UPDATE (vytváření a rušení směrovacích informací). Jelikož nebudeme v simulaci nastavovat parametry jako, jsou váha (WEIGHT), nebo místní preference (LOCAL_PREF), měla by při zahájení přenosu dat, s možnosti výběru více cest, být zvolena cesta s nejnižším počtem procházených AS (AS_PATH). Dále se zaměříme na chování směrovačů při výpadku linky v průběhu přenosu dat.



Obrázek 10. Návrh simulované sítě

Nyní přejděme k samotnému návrhu simulace. Skript budeme psát v jazyku v jazyce C++. Pro vytvoření, editaci skriptu můžeme použít jeden s velkého množství textových editorů, nebo vývojových platforem, jako jsou například Eclipse, nebo Geany.

Na začátku programu připojíme externí moduly z knihovny NS-3. Není to nejefektivnější řešení, jelikož moduly mohou obsahovat i hlavičkové soubory, které v programu nepoužijeme, ale psaní scénáře tímto způsobem je mnohem jednodušší.

```
#include "ns3/netanim-module.h"
#include "ns3/network-module.h"
#include "ns3/core-module.h"
```

```
#include "ns3/internet-module.h"
#include "ns3/dce-module.h"
#include "ns3/quagga-helper.h"
#include "ns3/point-to-point-helper.h"
#include "ns3/applications-module.h"
```

Moduly zajišťují funkčnost a správné chování simulovaných sítí.

Dále definujeme obor názvů `namespase`. Potom následuje hlavní funkce programu, do které nadefinujeme celý scénář simulace.

```
using namespace ns3;
int main (int argc, char *argv[])
```

K definování uzlů použijeme helper `NodeContainer`, který nám slouží k jednoduchému vytvoření, zorganizování a přístupu k uzlům. Metodou `Create` vytvoříme požadované množství uzlů:

```
NodeContainer nodes;
nodes.Create (7);
```

Dále nadefinujeme linku a její parametry. K tomuto použijeme topology helper `PoinToPointHelper`. Jelikož parametry linek mezi uzly budou stejné, bude stačit pouze jedna definice:

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

Dalším krokem bude přiřazení linek mezi uzly. Pro přiřazení využijeme pomoci helperu `NetDeviceCointainer` vytvořením kontejneru, který můžeme chápat jako síťové rozhraní na daných uzlech. Pomocí metody `Install` přiřadíme linky mezi dvojice uzlů:

```
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes.Get (0), nodes.Get (1));
devices = pointToPoint.Install (nodes.Get (1), nodes.Get (2));
```

Jelikož budeme chtít při simulaci využívat sadu protokolů implementovaných přímo v linuxovém jádru, musí být tato sada nastavena. Pro toto nastavení využijeme pomoci helperu `DceManagerHelper`. Metodou `SetNetworkStock` určíme sdílenou knihovnu, pomocí které bude simulátor využívat protokolů jádra.

```
DceManagerHelper processManager;
processManager.SetTaskManagerAttribute ("FiberManagerType",
                                                EnumValue (0));
processManager.SetNetworkStack ("ns3::LinuxSocketFdFactory",  "Library",
                                              StringValue ("liblinux.so"));
processManager.Install (nodes);
```

Nyní nastavíme IP adresy na jednotlivých uzlech, k čemuž využijeme funkce `AddAdress` a `RunIP`. Konfiguraci IP adres a aktivování rozhraní zajišťuje linuxový program `ip`, který má podobu:

```
ip -f inet addr add adress dev name
```

K volání tohoto příkazu slouží funkce `RunIP` a `AddAdress`.

```
AddAddress (nodes.Get (0), Seconds (0.1), "sim0", "10.0.0.1/24");
RunIp (nodes.Get (0), Seconds (0.11), "link set lo up");
RunIp (nodes.Get (0), Seconds (0.11), "link set sim0 up");
```

Dalším krokem je nastavení směrovacího protokolu BGP pomocí helperu `QuaggaHelper`. Metodou `EnableHelper` přiřadíme daný směrovací protokol uzlům, na kterých tento protokol požadujeme. Pomocí metody `BgpAddNeighbor` nastavíme na jednotlivých uzlech směrovací informace o sousedních směrovačích.

```
QuaggaHelper quagga;
quagga.EnableBgp (nodes);
quagga.BgpAddNeighbor (nodes.Get (0), "10.0.0.2",
                       quagga.GetAsn (nodes.Get (1)));
quagga.BgpAddNeighbor (nodes.Get (0), "10.0.3.2",
                       quagga.GetAsn (nodes.Get (3)));
quagga.Install (nodes);
```

Dále vytvoříme aplikace pro generování paketů, kterou přiřadíme klientovi a aplikace pro přijímání paketů kterou přiřadíme serveru. Pro vytvoření těchto aplikací slouží topology helper `DceApplicationHelper`, který umožňuje spustit binární kód linuxového programu. Pro generování paketů a určení příjemce bude využíván program `udp-perf`. Pomocí metody `AddArgument` nastavíme délku trvání aplikací a čísla portů. U klienta, který bude generovat UDP datagramy, nastavíme množství přenášených dat v jednom datagramu a IP adresu cílového uzlu. Pomocí metody `Start` nastavíme čas spuštění aplikací.

```
DceApplicationHelper process;
 ApplicationContainer apps;
```

```
std::ostringstream oss;
process.SetBinary ("udp-perf");
process.AddArgument ("--duration=100");
process.AddArgument (oss.str ().c_str ());
oss.clear ();
process.AddArgument ("--port=9");
process.AddArgument (oss.str ().c_str ());
apps = process.Install (nodes.Get(5));
apps.Start (Seconds (100.0));

process.SetBinary ("udp-perf");
process.ResetArguments ();
process.AddArgument ("--pktsize=500");
process.AddArgument (oss.str ().c_str ());
oss.clear ();
process.AddArgument ("--port=9");
process.AddArgument (oss.str ().c_str ());
oss.clear ();
process.AddArgument ("--client");
process.AddArgument (oss.str ().c_str ());
oss.clear ();
oss.str ("");
oss << "--host=" << Ipv4AddressToString ("10.0.4.2");
process.AddArgument (oss.str ().c_str ());
oss.clear ();
oss.str ("");
process.AddArgument ("--duration=30");
apps = process.Install (nodes.Get(2));
apps.Start (Seconds (100.0));
```

Pro simulace výpadku linky využijeme možnosti funkce `RunIP`, kdy parametrem `Down` deaktivujeme rozhraní na uzlu. Čas výpadku nastavíme na uzlu č.4 (AS5), přes který bude datový tok směrován.

```
RunIp (nodes.Get (4), Seconds (110.0), "link set sim1 down");
```

Posledním krokem v definování skriptu bude nastavení uložení výsledků simulace do výstupních souborů. Pro tuto činnost využijeme pomoci topology helperů `AsciiTraceHelperForDevice`, `PcapHelperForDevice` a třídu `AnimationInerface` modulu `netanim-module`.

```
pointToPoint.EnableAsciiAll ("vysledky-txt/quagga-bgpd");
pointToPoint.EnablePcapAll ("vysledky-pcap/quagga-bgpd");
AnimationInterface anim ("vysledky-xml/quagga-bgpd.xml");
```

První metoda ukládá výsledky simulace jako textový soubor, druhá metoda jako soubor typu pcap. Soubor typu pcap je možné otevřít v programu Wireshark. Poslední metoda ukládá výsledky simulace do souboru xml, který je spustitelný v programu NetAnim.

8.3. Spuštění simulace

Pro sestavení, kompilaci a spuštění se používá program Waf. Před spuštěním je nejdříve nutné skript uložit do složky example v adresáři myscripts/ns-3-quagga a poté upravit konfigurační soubor wscript uložený ve složce myscripts/ns-3-quagga. Do konfiguračního souboru je potřeba nadefinovat modely použité v simulaci, název souboru, cílový adresář a jméno zkompilovaného souboru. Simulace se spustí příkazem:

```
./waf -run quagga-bgpd
```

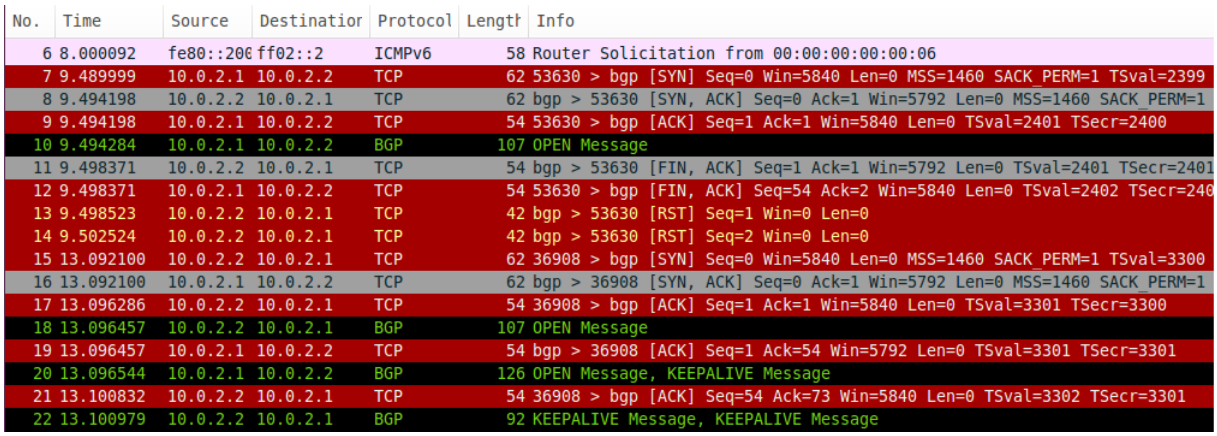
[5] [14]

8.4. Vyhodnocení simulace

Pro vyhodnocení simulace použijeme výsledky simulace zaznamenané na peer směrovačích AS3 a AS5. U AS3 budeme vyhodnocovat provoz na obou linkách (AS5, AS2), z důvodu možnosti vidět přenos zpráv UPDATE, konkrétně nás zajímají údaje o směrovacích informacích do sítě 10.0.4.0, kde se nachází server, do kterého je směrován přenos dat. U směrovače AS5 nás bude zajímat chování směrovače při výpadku linky.

Nyní si přiblížíme výstupy simulace.

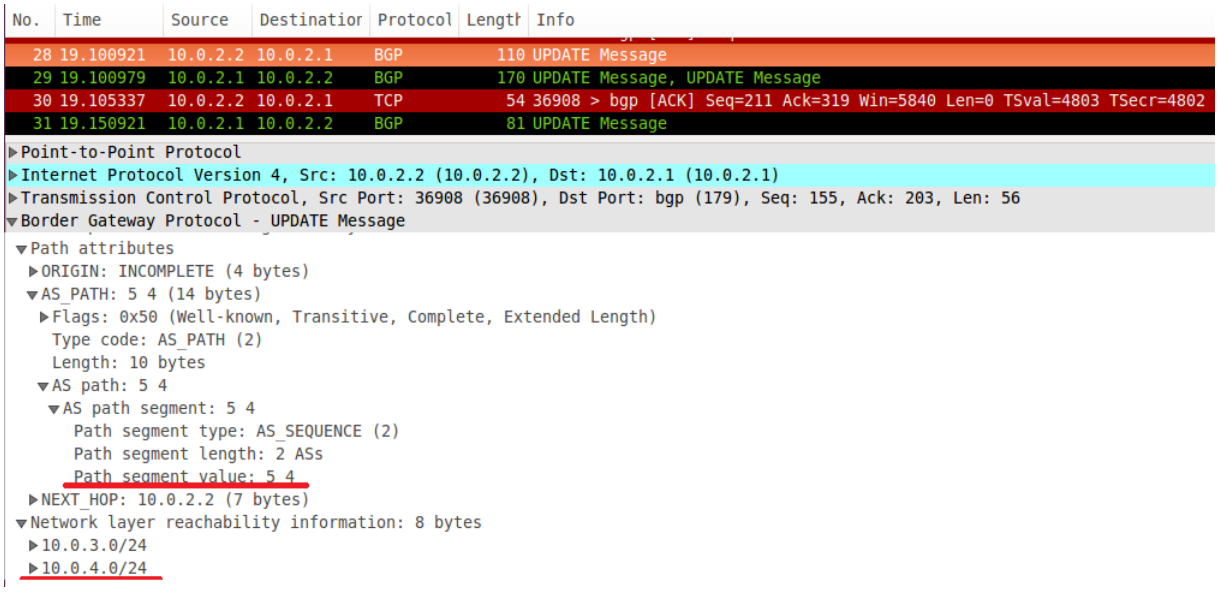
Na obrázku č. 11. je zachyceno navazování TPC spojení vysláním segmentu s příznakem SYN ze strany směrovače AS3 do směrovače AS5 v čase 9,489s, vyslání odpovědi SYN-ACK ze strany AS5 v čase 9,494s a následné odpovědi vysláním ACK ze strany AS3 v čase 9,494s. Po navázání spojení mezi směrovači AS3 a AS5, můžeme vidět v čase 9,494s vyslání zprávy OPEN protokolu BGP a poté v čase 9,498s ukončení spojení vysláním segmentu s příznaky FIN a ACK. Dále zde můžeme vidět opětovné navázání spojení mezi uzly a vyslání zprávy OPEN ze strany AS5, následné znovu vyslání zprávy OPEN ze strany AS3 a potvrzení zprávy OPEN vyslanou směrovačem AS5 vysláním zprávy KEEPALIVE. V čase 13,1s můžeme vidět potvrzení obou zpráv OPEN vyslaných z AS3 zprávami KEEPALIVE vyslaných z AS5.



No.	Time	Source	Destinatior	Protocol	Lengtl	Info
6	8.000092	fe80::206	ff02::2	ICMPv6	58	Router Solicitation from 00:00:00:00:00:06
7	9.489999	10.0.2.1	10.0.2.2	TCP	62	53630 > bgp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=2399
8	9.494198	10.0.2.2	10.0.2.1	TCP	62	bgp > 53630 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1
9	9.494198	10.0.2.1	10.0.2.2	TCP	54	53630 > bgp [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=2401 TSecr=2400
10	9.494284	10.0.2.1	10.0.2.2	BGP	107	OPEN Message
11	9.498371	10.0.2.2	10.0.2.1	TCP	54	bgp > 53630 [FIN, ACK] Seq=1 Ack=1 Win=5792 Len=0 TSval=2401 TSecr=2401
12	9.498371	10.0.2.1	10.0.2.2	TCP	54	53630 > bgp [FIN, ACK] Seq=54 Ack=2 Win=5840 Len=0 TSval=2402 TSecr=240
13	9.498523	10.0.2.2	10.0.2.1	TCP	42	bgp > 53630 [RST] Seq=1 Win=0 Len=0
14	9.502524	10.0.2.2	10.0.2.1	TCP	42	bgp > 53630 [RST] Seq=2 Win=0 Len=0
15	13.092100	10.0.2.2	10.0.2.1	TCP	62	36908 > bgp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=3300
16	13.092100	10.0.2.1	10.0.2.2	TCP	62	bgp > 36908 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1
17	13.096286	10.0.2.2	10.0.2.1	TCP	54	36908 > bgp [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=3301 TSecr=3300
18	13.096457	10.0.2.2	10.0.2.1	BGP	107	OPEN Message
19	13.096457	10.0.2.1	10.0.2.2	TCP	54	bgp > 36908 [ACK] Seq=1 Ack=54 Win=5792 Len=0 TSval=3301 TSecr=3301
20	13.096544	10.0.2.1	10.0.2.2	BGP	126	OPEN Message, KEEPALIVE Message
21	13.100832	10.0.2.2	10.0.2.1	TCP	54	36908 > bgp [ACK] Seq=54 Ack=73 Win=5840 Len=0 TSval=3302 TSecr=3301
22	13.100979	10.0.2.2	10.0.2.1	BGP	92	KEEPALIVE Message, KEEPALIVE Message

Obrázek 11. Navazování spojení mezi peer směrovači AS3 a AS5

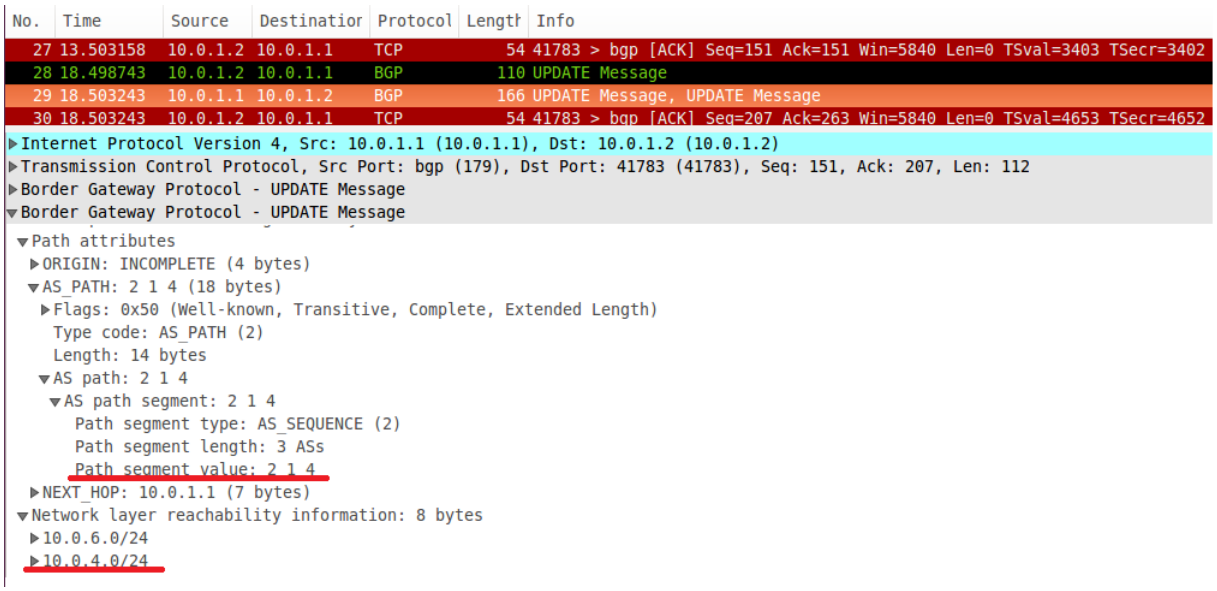
Na obrázcích č. 12. a č. 13. můžeme vidět distribuci zpráv UPDATE ze směrovačů AS2 a AS5 do směrovače AS3 s informacemi o dostupnosti sítí. Konkrétně nás zajímá údaj o vzdálenosti do sítě 10.0.4.0 (atribut AS_PATH), na základě kterého bude rozhodnuto o směrování přenosu dat z klienta do serveru. Můžeme zde vidět zaznamenané AS, kterými zpráva prošla (Path segment value). Po vyhodnocení přijatých zpráv UPDATE, by měl směrovač AS2 směrovat přenos dat přes AS5 z důvodu menší vzdálenosti.



No.	Time	Source	Destinatior	Protocol	Lengtl	Info
28	19.100921	10.0.2.2	10.0.2.1	BGP	110	UPDATE Message
29	19.100979	10.0.2.1	10.0.2.2	BGP	170	UPDATE Message, UPDATE Message
30	19.105337	10.0.2.2	10.0.2.1	TCP	54	36908 > bgp [ACK] Seq=211 Ack=319 Win=5840 Len=0 TSval=4803 TSecr=4802
31	19.150921	10.0.2.1	10.0.2.2	BGP	81	UPDATE Message

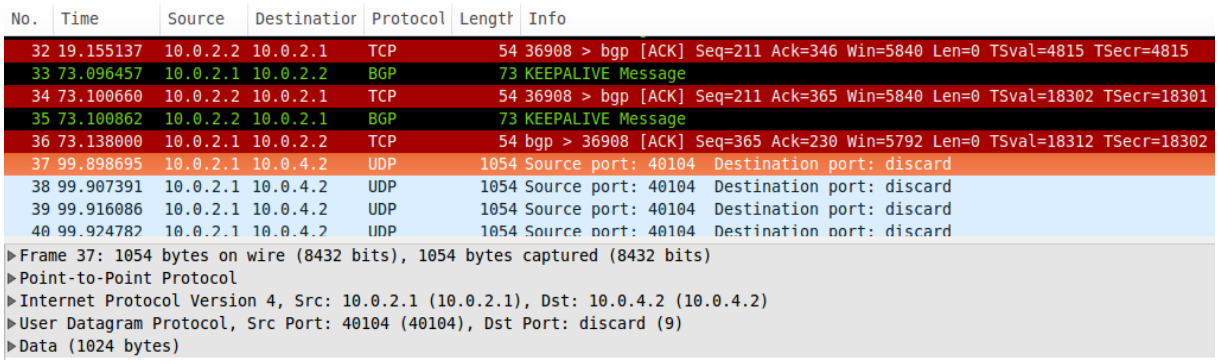
```
▶ Point-to-Point Protocol
▶ Internet Protocol Version 4, Src: 10.0.2.2 (10.0.2.2), Dst: 10.0.2.1 (10.0.2.1)
▶ Transmission Control Protocol, Src Port: 36908 (36908), Dst Port: bgp (179), Seq: 155, Ack: 203, Len: 56
▼ Border Gateway Protocol - UPDATE Message
  ▼ Path attributes
    ▶ ORIGIN: INCOMPLETE (4 bytes)
    ▼ AS_PATH: 5 4 (14 bytes)
      ▶ Flags: 0x50 (Well-known, Transitive, Complete, Extended Length)
        Type code: AS_PATH (2)
        Length: 10 bytes
      ▼ AS path: 5 4
        ▼ AS path segment: 5 4
            Path segment type: AS_SEQUENCE (2)
            Path segment length: 2 ASs
            Path segment value: 5 4
    ▶ NEXT_HOP: 10.0.2.2 (7 bytes)
  ▼ Network layer reachability information: 8 bytes
    ▶ 10.0.3.0/24
    ▶ 10.0.4.0/24
```

Obrázek 12. Zpráva UPDATE vyslaná z AS5 do AS3



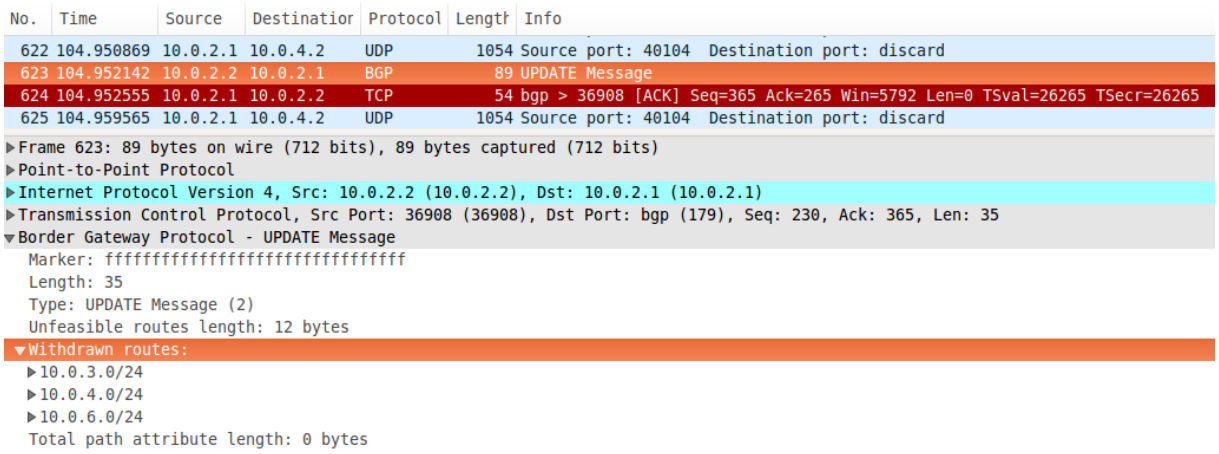
Obrázek 13. Zpráva UPDATE vyslaná z AS2 do AS3

Na obrázku č. 14. vidíme informace o provozu mezi AS2 a AS5, můžeme zde zaznamenat vysílání a přijetí zprávy KEEPALIVE. Směrovače mají nastavený časovač na vysílání těchto zpráv po každých 60-ti sekundách. Dále v čase 99,8986s je zachyceno zahájení plánovaného vysílání UDP paketů z AS2 do AS6, ve kterém se nachází server.



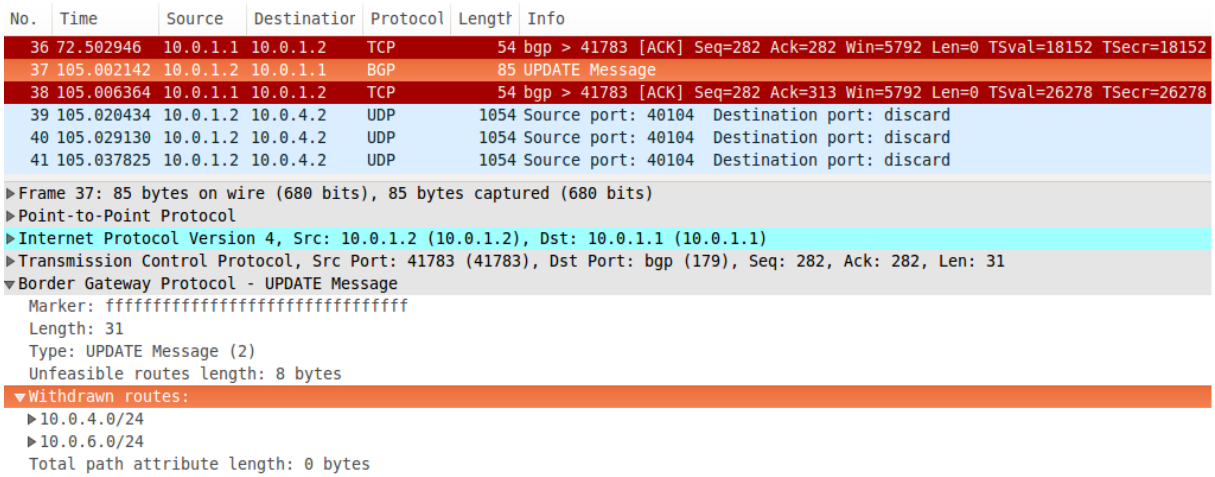
Obrázek 14. Zahájení vysílání z AS3 do AS6

Na obrázku č. 15. vidíme plánovaný výpadek linky mezi směrovači AS4 a AS5 a také zde můžeme v čase 104,95 vidět odesílání zprávy UPDATE ze směrovače AS5 do směrovače AS3 s aktualizací do směrovací tabulky.



Obrázek 15. Plánovaný výpadek linky mezi směrovači AS4 a AS5

Na obrázku č. 16. můžeme v čase 105,002s vidět odesílání UPDATE zprávy do směrovače AS2 ze směrovače AS3 a v čase 105,02 následné přesměrování datového toku odesílaného do AS6.



Obrázek 16. Přesměrování vysílání UDP paketů z AS3 přes peer směrovač AS2

Závěr

Cílem bakalářské práce bylo prostudovat problematiku simulace počítačových sítí, se zaměřením na síťové simulátory NS-2 a NS-3. Dalším cílem byl popis směrovacího protokolu BGP a tento protokol použít při simulaci počítačové sítě na simulátoru NS-3.

První část části teoretické se zaměřuje na popis protokolu IP, rozdělení směrovacích protokolů, na protokol TCP, navazování a ukončování spojení tímto protokolem, na protokol UDP. Dále je v práci rozebráno, co jsou to autonomní systémy a směrovací protokol BGP.

Druhá část teoretické části práce obsahuje popis teorie simulací, na kterou navazuje popis simulátorů NS-2 a NS-3, jaké protokoly podporují, jaké jsou základní objekty těchto simulátorů. U simulátoru NS-3 je popsán nástroj pro přímé vykonávání kódu (DCE) a balíček Quagga, které jsou použity v praktické části bakalářské práce.

V praktické části bakalářské práce je detailně popsáno vytvoření simulačního skriptu pro použití nástroje pro přímé vykonávání kódu a balíčku Quagga. Součástí scénáře byl použit výpadek linky jako simulace chyby při přenosu dat, aby bylo možné zaznamenat chování směrovacího protokolu BGP, který byl při simulaci použit.

Poslední kapitola se zaměřuje na rozbor výsledků provedené simulace, na popis navazování spojení pomocí protokolu TCP, chování protokolu BGP, konkrétně na zřizování vazby mezi směrovači pomocí zprávy OPEN, dále na zasílání aktualizací do směrovacích tabulek pomocí zprávy UPDATE, zvolení směrování na základě atributu AS_PATH při přenosu dat, na udržování spojení pomocí zprávy KEEPALIVE a chování protokolu při výpadku linky v průběhu přenosu dat.

Conclusions

The aim of this thesis was to study the issue of computer network simulation; with a focus on network simulators NS-2 and NS-3. The next objective was to describe the BGP routing protocol and the protocol used in the simulation of computer networks with the NS-3 simulator.

The first theoretical part focuses on the description of the IP protocol, distribution of routing protocols, the TCP protocol connection establishment and termination of this protocol and the UDP protocol. This study also discusses what autonomous systems and routing protocol BGP are.

The second theoretical part of the paper contains a description of theory simulations; followed by NS-2 and NS-3 simulator descriptions including which protocols they support and the basic objects of these simulators. The NS-3 simulator description explains a tool for direct code execution (DCE) and Quagga package, which is used in the practical part of this thesis.

The practical part of the thesis explains in detail how to create a simulation script to use tools for DCE and Quagga package. Part of the scenario used a line outage simulation error during data transmission in order to record the behaviour of BGP routing protocol.

The last chapter focuses on the analysis of the results of the simulation, the description establishing a connection using TCP, BGP protocol behavior, namely the establishment of links between routers using the OPEN message, then the sending updates to routing tables using UPDATE messages, select the route based on the attribute AS_PATH during data transmission, to maintain connections using keepalive messages and protocol behavior during power lines during data transfer.

The last chapter focuses on the analysis of the results of the practical section. This includes an explanation on establishing a connection using TCP, BGP protocol behavior, establishment of links between routers using OPEN message, routing table updates utilizing UPDATE messages, choosing a route during transmission using the attribute AS_PATH, maintaining connections using KEEPALIVE messages and protocol behaviour of data transfers during a transmission failure.

Reference

[1] *The Network Simulator - ns-2*, [online] *http://www.isi.edu/nsnam/ns/*

[2] T. Issariyakul, E. Hossain : *Introduction to Network SimulatorNS2.*

[3] R. E. Shannon : *Introduction to the art and science of simulation.* in Proc. ofthe 30th conference on Winter simulation (WSC'98), 1989.

[4] R. G. Ingalls : *Introduction to simulation.* Proceedings of the 34th conference on Winter simulation. Winter Simulation Conference, 2002.

[5] *The network simulator NS-3* [online] *http://www.nsnam.org/*

[6] E. Altman, T. Jimenéz : *NS Simulator for beginners, Lecture notes.* Universita de Los Andes, Venezuela, 2003.

[7] L. Dostálek, A. Kabelová : *Velký průvodce protokoly TCP/IP a systémem DNS.*Computer Press,Brno, 2012

[8] J. Jeřábek : *Skripta k předmětu Pokročilé komunikační techniky.*Vysoké učení technické v Brně, Brno 2009.

[9] P. Grygárek : *Směrovací protokol BGP*, [online] *http://www.cs.vsb.cz/grygarek/SPS/lect/BGP/BGP.html*

[10] S. H. Y. Rekhter, T. Li : *A Border Gateway Protocol 4 (BGP-4).* 2006. [online] *http://www.faqs.org/rfcs/rfc4271.html*

[11] Wikipedia - Internet Protocol, [online] *http://www.cs.wikipedia.org/wiki/Internet_Protocol*

[12] O. Wendell : *CCENT/CCNA ICND1 640-822 Official Cert Guide.* Third Edition. Cisco Press, 2011.

[13] T. Lammle : *CCNA výukový průvodce přípravou na zkoušku 640-802.* Computer Press, 2010.

[14] *NS-3 Quagga Documentation*, [online] *https://www.nsnam.org/docs/dce/manual-quagga/html/getting-started.html*

[15] P. Bouška : *samuraj-cz Navázání spojení v TCP* [online] *http://www.samuraj-cz.com/clanek/tcpip-navazani-a-ukonceni-spojeni/*