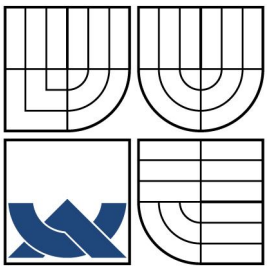
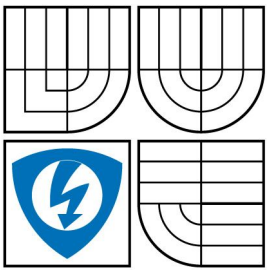


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH UNICASTOVÉHO A MULTICASTOVÉHO SIMULAČNÍHO MODELU V NS2

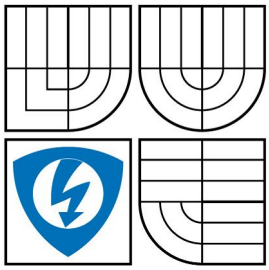
DESIGN OF UNICAST AND MULTICAST MODELS IN NS2

bakalářská práce
bachelor´s thesis

AUTOR PRÁCE — MARTIN VOTAVA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MILAN ŠIMEK
SUPERVISOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

fakulta elektrotechniky a komunikačních technologií

ústav telekomunikací

Bakalářská práce

bakalářský studijní obor

Teleinformatika

Student: Votava Martin **ID:** 78409

Ročník: 3 **Akademický rok:** 2007/2008

NÁZEV TÉMATU:

Návrh unicastového a multicastového simulačního modelu v NS2

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se se simulačním nástrojem Network Simulator 2 a prozkoumejte jeho možnosti pro návrh pevných IP sítí. Popište základní principy a procesy při návrhu sítí v tomto simulačním nástroji. Zaměřte se především na možnosti simulace směrovacích protokolů. Cílem práce je vytvoření dvou laboratorních úloh zabývajících se směrovaním paketů v unicastových a multicastových IP sítích. U multicastového modelu připravte zadání objasňující problematiku přenosu multicastových dat v módu Dense a Sparse.

DOPORUČENÁ LITERATURA:

[1] Information Sciences Institute, "The Network Simulator - ns-2", June 2004, online, dostupné na: <http://www.isi.edu/nsnam/ns/>

[2] M.A. SPORCTAK, Směrování v sítích IP, ComputerPress, a.s.,2004, ISBN: 80-251-0127-4

Termín zadání: 11.2.2008 **Termín odevzdání:** 4.6.2008

Vedoucí práce: Ing. Milan Šimek

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

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:

Bytem:

Narozen/a (datum a místo):

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: .....

Vedoucí/ školitel VŠKP: .....

Ústav: .....

Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů ……..
- □ elektronické formě – počet exemplářů ……..

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: …….

…….. ……

Nabyvatel Autor

ANOTACE

Práce je zaměřena na tvorbu sítí pomocí nástroje Network Simulátor 2. Hlavním tématem práce je simulace unicastového a multicastového směrování.Práce také popisuje protokoly používané na transportní vrstvě, jako je protokol TCP (Transmission Control Protocol) a UDP (User Datagram Protocol). V neposlední řadě práce obsahuje laboratorní úlohy, v nichž si čtenář může ověřit nabité znalosti simulování Unicastu a Multicastu.

Klíčová slova: Network Simulátor 2, Skupinové směrování, Unicast, Směrování, Transportní vrstva, Protokol UDP, Protokol TCP

ABSTRACT

This work is focused on Network Simulator 2 - tool used for simulating network implementations. Focal point of this work is simulating of unicast and multicast routing. This work also describes protocols used on transport layer, such TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Last but not least work includes laboratory part in which reader is able to verify reached knowledge in Unicast and Multicast simulation.

Keywords: Network Simulator 2, Multicast, Unicast, Routing, Transport layer, User Datagram Protocol, Transmission Control Protocol

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma "Návrh unicastového a multicastového simulačního modelu v NS2 " jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdroj, 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 jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona . 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona . 140/1961 Sb.“

V Brn dne …… ……..

(podpis autora)

Použité zkratky:

- **ACK** (Acknowledge): Potvrzovací paket
- **CBR** (Constant Bit Rate):Konstantní bitový tok
- **DRR** (Deficit Round Robin): Typ fronty používané při směrování
- **FIFO** (First in – First out): Typ fronty používané při směrování
- **FTP** (File Transfer Protocol): Protokol aplikační vrstvy
- **FQ** (Fair Queueing): Typ fronty používané při směrování
- **NAM** (Network Animator): Nástroj pro vizualizaci simulací
- **NS2** - Network Simulator 2
- **RED** (Random Early Discard): Typ fronty používané při směrování
- **TCL** (Tool Command Language): Programovací jazyk
- **TCP** (Transmission Control Protocol): Základní protokol protokolové sady internetu
- **TTL** (Time To Live): Parametr životnosti paketu
- **UDP** (User Datagram Protocol): Základní protokol protokolové sady internetu

Obsah:

1. Úvod:

Cílem mé bakalářské práce je seznámení čtenáře s programem Network simulátor a následné vytvoření dvou laboratorních úloh k tomuto programu. První laboratorní úloha má za úkol seznámit čtenáře s možností simulací unicastového směrování. V druhé části se zaměřím na multicastingové směrování pomocí laboratorních úloh.

Toto téma jsem si vybral z toho důvodu, protože v průběhu mého studia na VUT mě nejvíce zaujala tvorba sítí, směrování v sítích a další věci týkající se blíže této problematiky. A právě program NS2 je vhodným nástrojem pro simulaci výše zmíněné problematiky.

Mým cílem je, aby studenti, kteří budou vypracovávat mnou vytvořené laboratorní úlohy, se díky nim naučili pracovat s programem NS2, ale hlavně aby měli možnost vidět, jak směrování pomocí unicatu a multicastu funguje.

Věřím, že se mi vytýčené cíle podaří naplnit. A tato práce přiblíží čtenáři základní možnosti programu Network simulátor.

2. O NS2:

Network Simulátor je založen na dvou programovacích jazycích. Objektově orientovaný simulátor napsaný v C++ a překladač OTcl. NS2 má velice bohatou knihovnu sítí a protokolů. Máme zde dvě třídy hierarchií. Kompilovaná C++ hierarchie a překladač OTcl. Oba dva jsou mezi sebou v souladu.

Hierarchie C++ dovoluje vykonat efektivně simulaci a zrychlení doby vykonání simulace.

Pomocí definice OTcl skriptu, můžeme definovat jednotlivé topologie.

3. Tcl a Otcl programování:

Jazyk tcl (**T**ool **C**ommand **L**anguage) byl vyvinut Johnem Ousterhoutem. Je založen na jazyku Lisp (List procesing). Má velice jednoduchou syntaxi a dovoluje sjednocení s ostatními jazyky. Mezi jeho největší výhody patří :

- Přenositelnost (existují interpreti tcl pro mnoho platforem)
- Rozšiřitelnost
- Dostupnost zdrojových kódů
- Jednoduchost

3.1. Základní programování v tcl:

Pomocí příkazu **set** se zapisuje do proměnné.

Například zápis:

set x 5

Přiřadí proměnné x hodnotu 5.

Pomocí znaku # můžeme za zdrojový kód psát poznámky, aniž by měli vliv na program.

Pokud chceme pracovat s hodnotou proměnné, umístíme před ni znak $

Např.:

set x $a # Proměnné x je přiřazena hodnota proměnné a

Díky tcl můžeme provádět matematické operace.

Součet dvou proměnných zapíšeme následovně:

x [expr $a + $b]

Takto můžeme používat i další matematické operace (odčítání, dělení, násobení, ….)

Pro vypsání výsledku na obrazovku užíváme příkaz **puts.**

Např.:

set x 5

puts $x ; # vytisknutí hodnoty proměnné x

3.1.1 Podmíněné větvení:

V jazyce tcl máme dále možnost tzv.: podmíněného větvení. K tomu nám slouží příkaz if. Jeho použití je následující:

if {výraz 1} {tělo 1} elseif {výraz 2}{tělo 2}…else {výraz n}{tělo n}

Výraz zde znamená podmínku. Pokud je daná podmínka splněna, je provedeno tělo, pokud splněna není, program přechází na další výraz.

Praktické použití příkazu if:

Jednoduchý program na určení znaménka čísla

set x 5

if {$x > 0}{puts "kladné číslo"}

elseif {$x < 0}{puts "záporné číslo"}

else {$x = 0}{puts "nula"}

3.1.2 Cykly:

Pro vytváření cyklů máme v jazyce tcl k dispozici příkaz for a while. Cyklus for se používá následovně:

for start test další tělo

Na začátku je proveden příkaz start, potom je proveden test. Pokud je výsledkem testu true (pravda), provede se tělo a další. Cyklus se potom opakuje. Pokud je výsledkem testu false (nepravda), cyklus je ukončen.

Praktické použití příkazu for:

for {set i 0} {i<10}{incr i}{puts"zvýšení po $i."}

příkaz *incr* (inkrementace) znamená navýšení o 1, znamená to tedy že cyklus proběhne desetkrát.

Příkaz *while* je používán následovně:

while test tělo

Je vykonáváno tělo tak dlouho, dokud test vrací false(nepravda).

Tímto jsme prošli základní možnosti programování v tcl. Z ukázek příkladů je vidět, že jazyk tcl je poměrně jednoduchý a velice podobný jazyku c++.

4.Vytváření topologií v NS2:

Program NS2 je primárně určen na vytváření topologií sítí a to jak pevných tak bezdrátových. My se budeme zabývat simulováním pevných sítí.

Prvním krokem pro vytvoření simulace, je vytvoření objektu, který bude reprezentovat celou simulaci. K tomu použijeme příkaz:

set ns [NewSimulator]

Dalším krokem pro vytvoření topologie je vytvoření uzlů sítě. To se provádí pomocí příkazu:

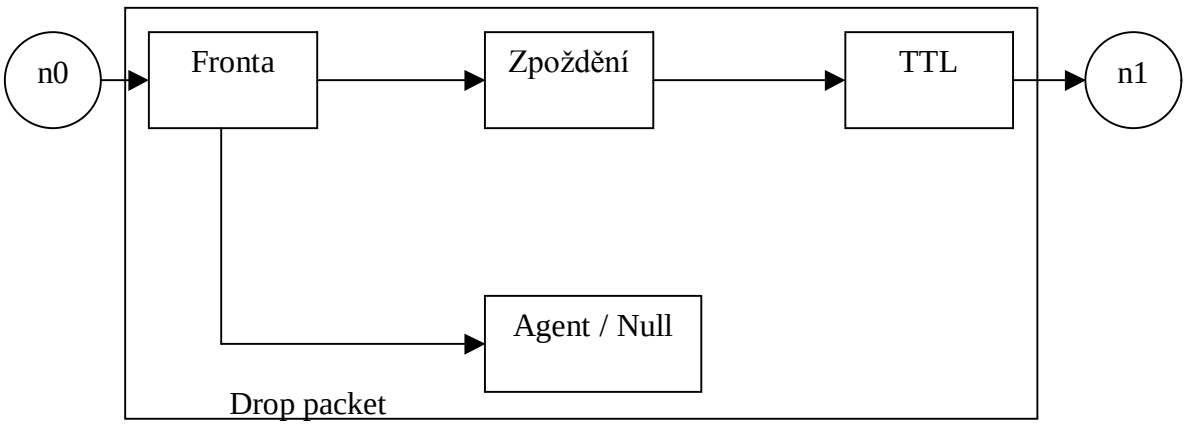
set n0 [$ns node] # vytvoření uzlu 0

Poté co vytvoříme potřebný počet uzlů, můžeme jednotlivé uzly vzájemně propojit. Propojení uzlů se provádí příkazem:

$ns duplex – link $n0 $n1 1Mb 10ms DropTail

Tímto příkazem propojíme uzly 1 a 2. Příkaz *duplex – link* určuje, že na lince může probíhat obousměrný provoz, ten může být nahrazen příkazem simplex – link (jednosměrný provoz). Dalšími parametry určujeme šířku pásma na 1Mb a zpoždění 10ms. Parametr DropTail určuje typ fronty, droptail představuje typ fronty FIFO (First in – First out) z názvu je patrný princip této fronty, ten paket, který vstoupí jako první, jde také první ven. Další možnosti front jsou RED (Random Early Discard), FQ (Fair Queueing), DRR (Deficit Round Robin) a další.

Provoz simplexní linky lze zobrazit pomocí blokového schématu.

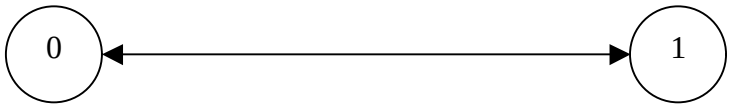


Obr. č.1: Blokové schéma simplexní linky

Přetečení fronty je zde simulováno pomocí zaslání drop packetu do Agent Null. TTL zde znamená time to live. Je to parametr, který nese každý paket a udává dobu "života" paketu. Prakticky je to tedy číslo, které určuje počet uzlů, kterými packet může projít. Po průchodu

uzlem je toto číslo sníženo o jedno, jakmile dosáhne nuly packet je zahozen. Duplexní linku si můžeme představit, jako zapojení dvou simplexních linek paralelně.

Pro vytvoření té nejjednodušší topologie, tedy o dvou uzlech a propojení těchto uzlů linkou, zapíšeme následující sled příkazů:
set n0 [$ns node] # Vytvoření uzlu 0
set n1 [$ns node] # Vytvoření uzlu 1
$ns duplex – link $n0 $n1 1Mb 8ms DropTail # Propojení obou uzlů duplexní linkou
Námi navržená síť by vypadala následovně:



Obr. č.2:Topologie o dvou uzlech

Teď, když máme vytvořené jednotlivé uzly a vytvořeno propojení mezi uzly, je potřeba simulovat provoz na lince. To je prováděno pomocí tzv.: agenta. Ten představuje protokol, kterým se přenos dat řídí.
Agenta vytvoříme příkazem:
set udp0[new Agent/UDP] # Zde je vytvořen agent protokolu UDP
$ns attach–agent $n0 $udp0 ; # přiřadí agenta k uzlu 0
A na závěr je potřeba nastavit provoz na lince a jeho parametry:
 # vytvoření CBR(constant bit rate) konstantní bitový tok
set cbr 0 [new Application / Traffic / CBR]
$cbr0 set packetSize_ 500 # nastavení velikosti packetů
$cbr0 set interval_ 0,005 # interval mezi jednotlivými packety
$cbr0 attach – agent $udp0 #přiřazení CBR k UDP
Další možností provozu na lince je protokol ftp. Ten, viz kapitola o protokolech, pracuje pod protokolem TCP.
Na druhém uzlu je potřeba ještě vytvořit přijímač bitového toku. K tomu slouží následující příkazy:
set null0 [newAgent/Null]
$ns attach – agent $n1 $null0
Agent/Null je ten nejjednodušší přijímač, dochází u něj k pohlcení přijatých paketů bez možnosti další analýzy. Dalším přijímačem je sink, který se přiřazuje k TCP.

Následujícím krokem je propojení obou agentů:

$ns connect $udp0 $null0

Na závěr musíme definovat začátek a konec vysílání:

$ns at 0,5 "$cbr0 start"

$ns at 4,5 "$cbr0 stop"

Vysílání tedy začne 0,5 sekundy od spuštění programu a bude ukončone 4,5 sekundy po spuštění programu. Poběží tedy 4 sekundy.

Posledním řádkem našeho zdrojové kódu je:

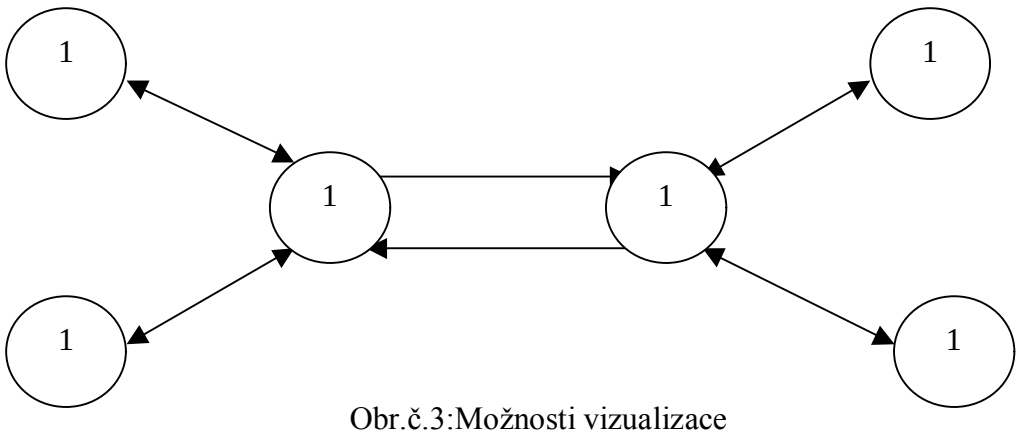
$ns run

Tento příkaz nám zahájí celou simulaci.

4.1. Vizualizace simulací:

Pokud máme navrženou nějakou topologii, můžeme ji převést do grafické podoby. To provedeme mimo NS2, a to pomocí nástroje nam (Network animator). Aby bylo grafické znázornění přehledné, je dobré zvolit rozmístění jednotlivých uzlů. Pokud by jsme tak neučinili, byla by jejich rozmístění náhodná.

Pokud chceme určit rozložení uzlů např. dle obrázku 3.



Obr.č.3:Možnosti vizualizace

Potom píšeme:

$ns duplex-link-op $n0 $n2 orient right - down

$ns duplex-link-op $n1 $n2 orient right – up

$ns simplex-link-op $n2 $n3 orient right

$ns simplex-link-op $n3 $n2 orient left

$ns duplex-link-op $n3 $n4 orient right – up

$ns duplex-link-op $n3 $n5 orient right – down

Na prním řádku je řečeno, že uzly 0 a 2, které jsou propojeny duplexní linkou, jsou vzájemně orientovány tak, že uzel 2 je umístěn vpravo dole od uzlu 0. Definice dalších uzlů je obdobná. Další možností zpřehlednění grafického zobrazení je možnost barevného rozlišení uzlů a linek a změna tvaru uzlů. Pokud chceme odlišit datové toky v naší simulaci, tak si jednotlivé barvy těchto toků nadefinujeme na začátku programu. Tato definice je vidět níže:

$ns color 1 red

$ns color 2 green

Potom barvu přiřadíme k TCP či UDP to se provádí následovně:

$udp set fid_ 1

Pro změnu barvy uzlu píšeme příkaz:

$n0 color red

Jak už bylo řečeno, dalším parametrem, který můžemem měnit, je tvar uzlu. K tomu užijeme následující příkaz:

$n0 shape box # uzel 0 má tvar čtverce

Dalšími tvary jsou hexagon a kruh(circle). Kruh je nastaven u všech uzlů implicitně.

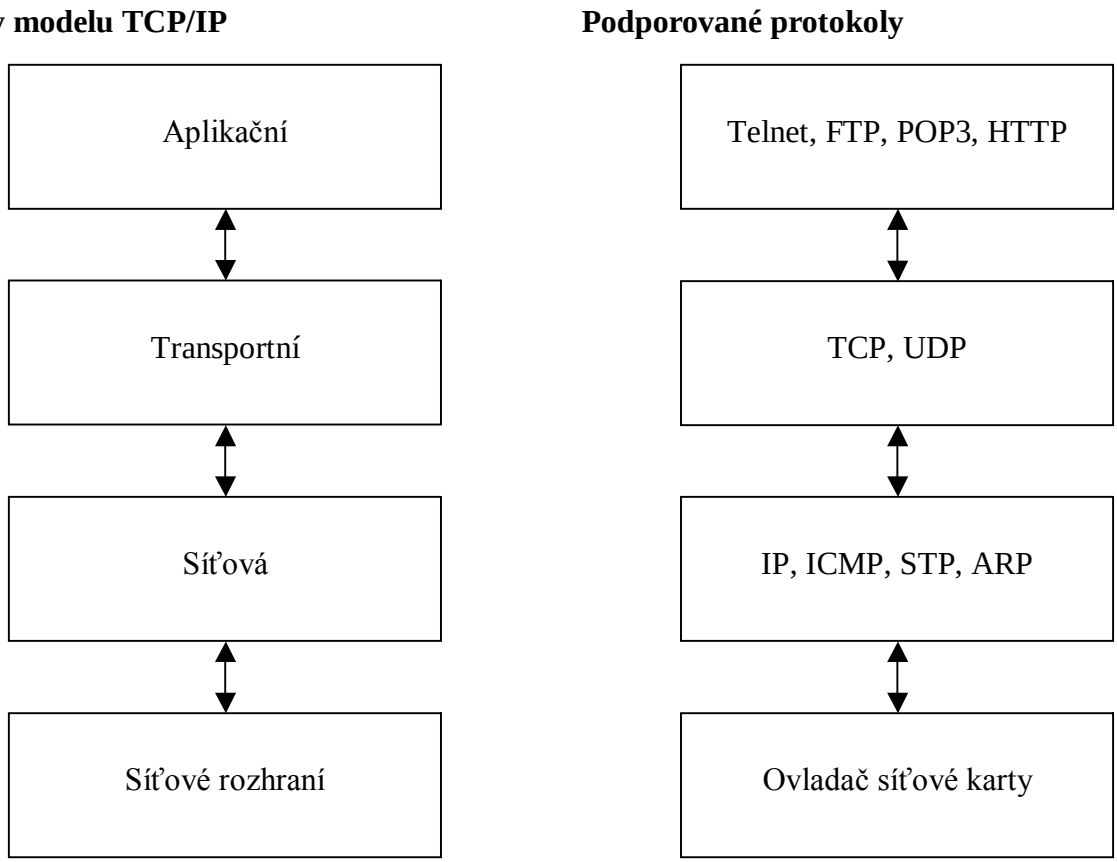
Co se týče změny barvy linky, použijeme tento příkaz:

$ns duplex – link – op $n0 $n2 color "green"

Ještě před tím, než se vrhneme na simulování vlastních sítí, vysvětlíme si, jak pracují jednotlivé protokoly, abychom věděli, kdy jaký protokol použít.

4.2. Protokoly v NS2:

Jedním z nejznámějších a nejrozšířenějších protokolů je protokol TCP, který pracuje na transportní vrstvě. Dalším protokolem pracujícím na transportní vrstvě je protokol UDP. Když se posuneme o vrstvu výš v modelu TCP/IP, viz obrázek 4, na aplikační vrstvu, tak zde můžeme najít protokol ftp(file transfer protocol), http (hypertext transfer protocol), telnet (telcommunicatons network) a další.



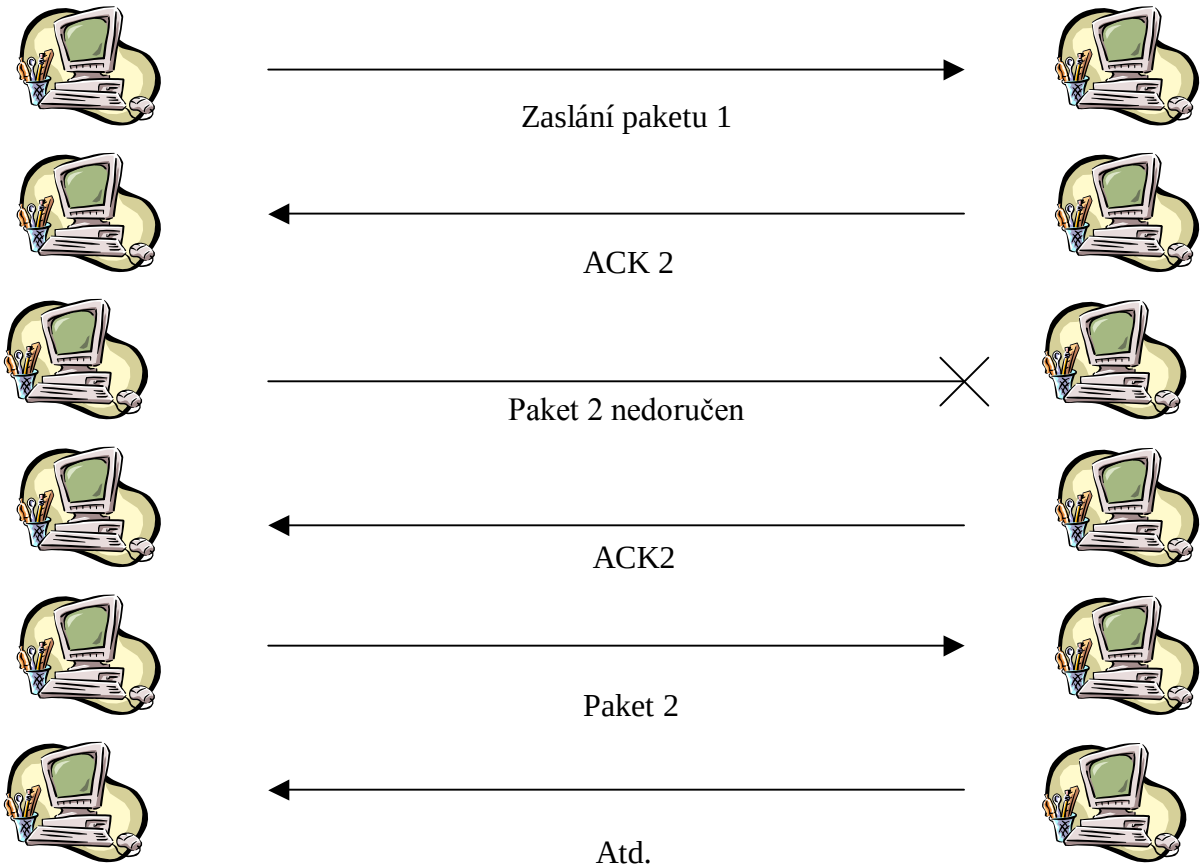
Obr.č.4: Model TCP/IP

4.2.1. Protokol TCP (Transport control protocol):

Tento protokol zajišťuje zhruba 90% veškerého internetového provozu.

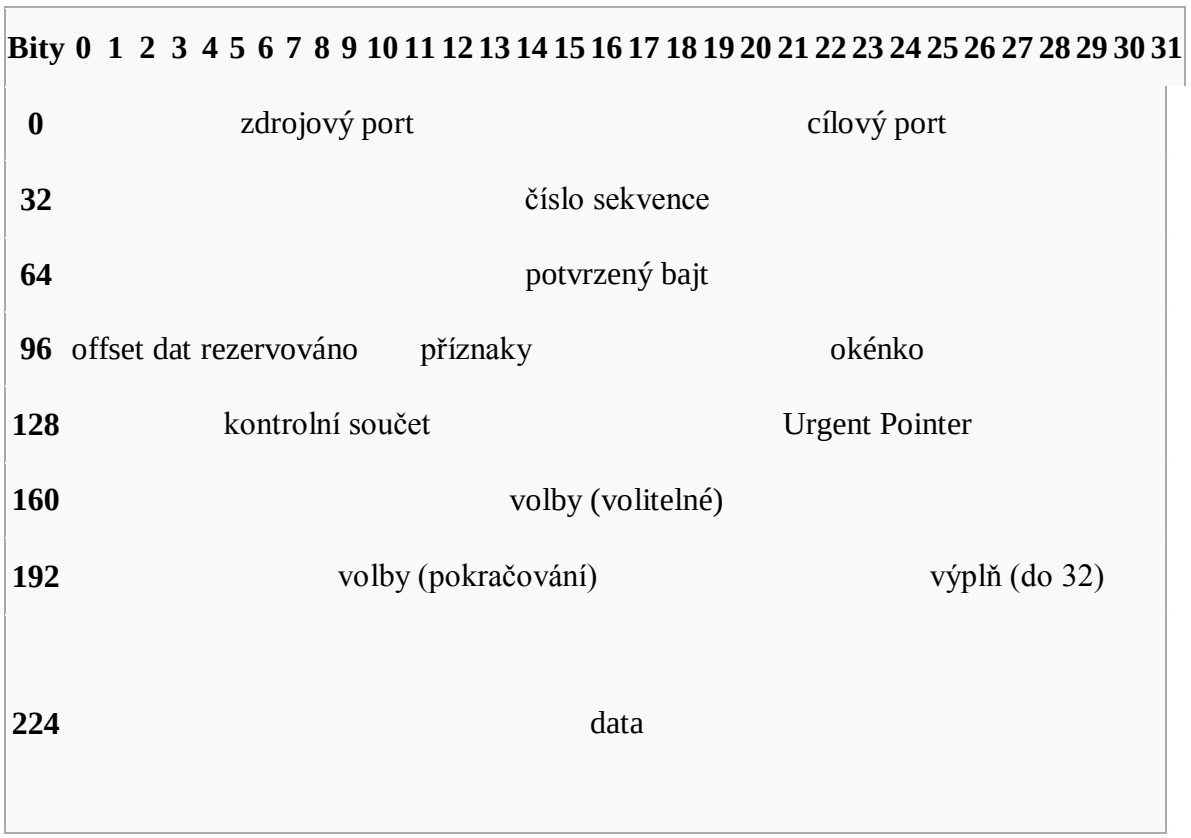
Protokol TCP zajišťuje spojovanou službu, což znamená, že mezi zdrojem dat a cílem je vytvořeno spojení, které je po ukončení přenosu rozvázáno. Dalo by se to přirovnat k telefonu. Když někomu voláme, tak po přijetí telefonátu druhou stranou naváže spojení. To se udržuje po dobu celého hovoru a po ukončení hovoru se spojení ukončí. Jak už jsme si řekli, tento protokol pracuje na transportní vrstvě. Jeho hlavním účelem je rozdělení zprávy na menší části (pakety) a ty na druhé straně, tedy u příjmače, opětovně složit do původní podoby. Díky rozdělení dat na menší úseky je možné kontrolovat jednotlivé pakety a v případě ztráty paketu nebo poškození, není nutné zasílat celou zprávu, ale stačí zaslat pouze ztracený paket. Zajišťuje tedy spolehlivý přenos dat. Princip kontroly paketů je následující:

Pokud je zaslán paket s číslem jedna, tak po jeho doručení edo cíle, je z cíle zaslán paket ACK, tedy packet potvrzující přijetí paketu 1. Paket ACK nese číslo příchozího packetu +1, je to tedy číslo dalšího očekávaného paketu. Tento princip můžeme vidět na obr. 5.



Obr č.5: Princip TCP

Pakety nemusí být potvrzovány jednotlivě. Potvrzování může probýhat i pro skupiny paketů. K tomu slouží tzv.: okno. Komunikace potom probíhá následovně. Zdroj a cíl se dohodnou na velikosti a počtu přenášených paketů. Vysílač poté odešle domluvený počet paketů, jejich velikost odpovídá velikosti okna. Tyto pakety jsou potom potvrzeny příjmačem. Pokud dojde k nějaké chybě, například příjmač nestíhá zpracovat všechny přijaté pakety, potom je okno zmenšeno. Zdroj a cíl si tedy bez potvrzování vymění menší počet paketů.
Podobu TCP paketu můžememem vidět na obrázku 6.



Obr. č.6:Paket TCP

Každý paket tedy nese:

- Číslo zdrojového a cílového portu
- Číslo sekvence udává pořadové číslo odesílaného bajtu
- Potvrzený bajt udává číslo přijatého bajtu
- Okno určuje přírůstek pořadového čísla přijatého bajtu
- Kontrolní součet ověřuje bezchybnost packetů
- Urgent pointer je ukazatel naléhavých dat
- Volby jsou volitelnými položkami záhlaví
- Data

Výhody TCP:

- ✓ Snadné zjištění nedoručených dat
- ✓ Data jsou interpretována ve správném pořadí
- ✓ Nevzniká duplicita dat
- ✓ Zajištěna správnost dat díky kontrolnímu součtu

Nevýhody TCP:

- ✖ Přenášení velkého počtu dat (hlavička protokolu TCP obsahuje velké množství informací)
- ✖ Velké zatížení sítě (nutné potvrzování paketů)

Protokol TCP je tedy vhodný pro přenos dat, u kterých potřebujeme zaručit bezchybnost, jako jsou textové dokumenty, filmy, obrázky atd.

4.2.2 Protokol UDP (User datagram protocol):

Jak už bylo řečeno, protokol UDP pracuje na transportní vrstvě. Narozdíl od protokolu TCP zajišťuje nespojovanou službu. Princip této služby lze přirovnat k listovní poště, kdy je každý dopis(paket) považován za samostatný celek a je doručen nezávisle na ostatních dopisech(paketech). To může způsobit doručení packetů v nesprávném pořadí.
U protokolu UDP jsou přenášeny datagramy. Jak takový UDP datagram vypadá, můžeme vidět na obrázku 7.

+	**bity 0 - 15**	**16 - 31**
0	zdrojový port	cílový port
32	délka	kontrolní součet
64	data	

Obr. č.7: Datagram UDP

UDP datagram nese:

- Zdrojový port je volitelný. Pokud není použit, je nastaven na nulu.
- Cílový port
- Délka udává velikost UDP datagramu včetně dat
- Kontrolní součet je nepovinný

Z UDP paketu, který máme zobrazený na obrázku 7, je vidět, že neobsahuje číslo sekvence ani potvrzený bajt. Z toho tedy vyplývá, že nedochází k potvrzování jednotlivých datagramů. Není tedy zajištěno, že všechny datagramy dorazí do cíle. Dále není zajištěna bezchybnost všech přenášených dat, protože kontrolní součet není povinnou součástí UDP datagramu. Dalším "rizikem" UDP je možnost duplicty. A v neposlední řadě, jak už bylo řečeno výše, může dojít k tomu, že při průchodu sítí mohou datagramy odeslané později dorazit dříve než datagramy odeslané před nimi. Nejsou tedy doručeny ve správném pořadí.

Výhody UDP:

- ✓ Malé množství přenášených dat (malá hlavička UDP datagramu)
- ✓ Nízké zatížení sítě (nejsou zasílány potvrzovací pakety)

Nevýhody UDP:

- ✘ Přenos není nijak zabezpečen proti - ztrátě dat
  - poškození
  - duplicitě

UDP je tedy vhodné používat tehdy, když potřebujeme zajistit rychlý přenos dat a na bezchybnosti přenosu tolik nelpíme. Jsou to například videokonference, internetové hry a další real – timové aplikace.

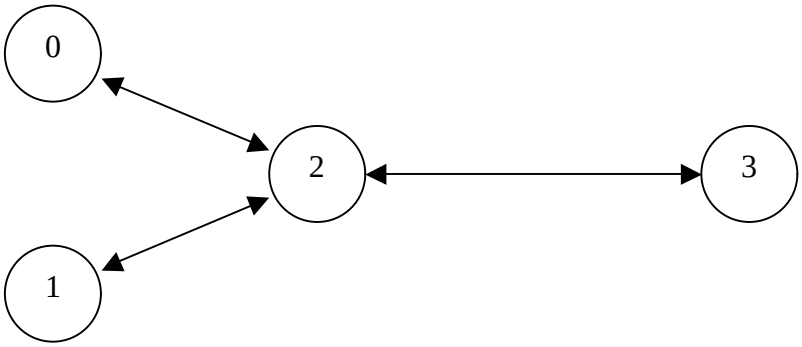
Pokud se ve struktuře TCP/IP, viz obrázek 4, posuneme o vrstvu výš, tedy na vrstvu aplikační, tak zde můžeme nalézt protokol ftp.

4.2.3 Protokol FTP (File transfer protocol):

Ftp je protokol, který slouží pro přenos souborů, jak již napovídá název. Tento protokol běží jen pod protokolem TCP, protože jak jsme již řekli, protokol TCP zajišťuje zabezpečený přenos dat, který je vhodný i pro přenos souborů. Tento protokol pracuje na portech 20 a 21, přičemž port 20 slouží pro přenos dat a port 21 pro odesílání a příjem příkazů.

5. Vlastní simulace:

Poté, co jsme si přiblížili základní protokoly, které budeme nyní používat, můžeme přistoupit k vlastnímu návrhu topologie. Pro začátek si tedy zvolíme jednoduchou topologii o čtyřech uzlech, které budou propojeny následovně:



Obr.č.8: Navržená topologie

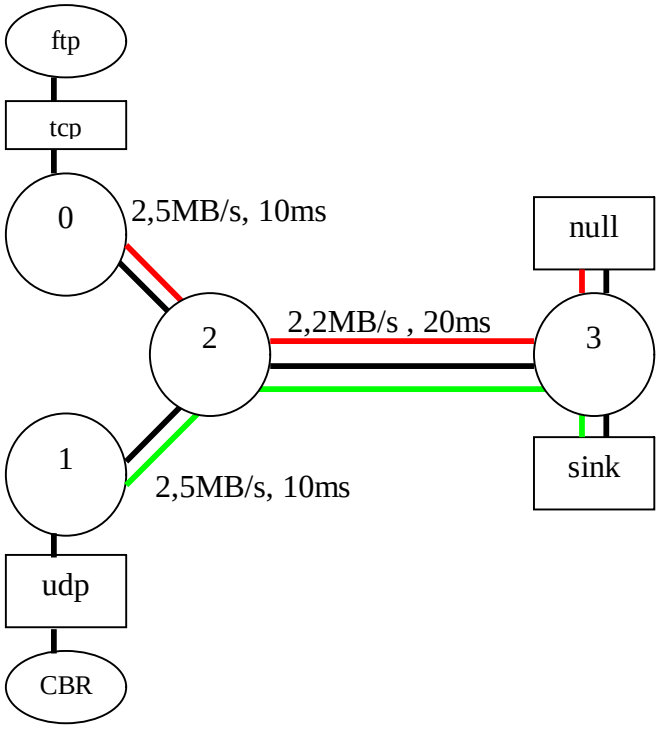
Uzel nula se bude řídit protokolem UDP s konstantním bitovým tokem, který generuje pakety o velikosti 1kB a to rychlosti 1Mbit/s a bude propojen s agentem null na uzlu tři. Jak bylo řečeno výše, tento agent všechny přijaté pakety bez analýzi zahodí, což nám pro naši první simulaci plně stačí. Na uzlu jedna bude protokol TCP s protokolem ftp. Protokol TCP bude propojen s objektem sink na uzlu 3. Ten zajišťuje, že každý obdržený paket bude potvrzen paketem ACK. Propojení mezi uzly 0-2 a 1-2 bude zajištěno duplexní linkou o kapacitě 2,5Mbit a zpoždění zde bude 1Oms. Uzly 2 a 3 budou také propojeny duplexní linkou, ovšem kapacita této linky bude 2,2Mbit a zpoždění 20ms. Druh fronty zvolíme FIFO, to zaručíme parametrem DropTail. Nyní, když jsme si určili všechny potřebné parametry naší simulace, můžeme se pustit do psaní zdrojového kódu.

```
set ns[NewSimulator]
# Určení barvy datových toků
$ns color 1 Green
$ns color 2 Red

# Otevření souboru pro zápis krokovacích informací pro NAM
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

```
# Definice procedury finish
proc finish{}{
        global ns tracefile1 namfile
        $ns flush-trace
        close $tracefile1
        close $namfile
        exec namout.nam &
        exit 0
}

# Definice uzlů
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
# Definice linek mezi jednotlivými uzly
$ns duplex-link $n0 $n2 2.5Mb 10ms DropTail
$ns duplex-link $n1 $n2 2.5Mb 10ms DropTail
$ns duplex-link $n2 $n3 2.2Mb 20ms DropTail

# Rozmístění uzlů pro nam
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

# Definice spojení UDP
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 1 #Přiřazení barvy k provozu
```

```
# Definice aplikace konstantního bitového toku a přiřazení k UDP
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false

# Definice spojení TCP
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 2 #Přiřazení barvy k provozu

# Definice aplikace ftp a přiřazení pod TCP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

# Definice počátku a konce provozu pro ftp a cbr
$ns at 0.5 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.5 "$ftp stop"
$ns at 5.0 "$cbr stop"

# Zavolání procedury finish po pěti sekundách od zahájení simulace
$ns at 6.0 "finish"

# Zahájení simulace
$ns run
```

Naše výsledná simulace potom vypadá následovně:



Obr.č.9:Výsledná topologie

Touto topologií jsme uzavřeli základní možnosti simulování pomoci NS2. Je vidět, že program NS2 skýtá opravdu velké možnosti pro simulaci sítí.

6. Návrh laboratorních úloh:

Nyní, když jsme se seznámili se základními možnostmi NS2. Je možné vyzkoušet si samostatnou práci v tomto programu. K tomuto účelu zde máme dvě laboratorní úlohy. První je zaměřená na unicast a druhá ještě rozšiřuje znalosti studenta a zaměřuje se na multicast.

6.1 Laboratorní úloha číslo 1:

Simulace Unicatového směrováni pomocí nástroje Network Simulátor 2 (NS2)

Zadání:

1. Prostudujte si úvod, ve kterém je nastíněno tvoření simulací v NS2.
2. V textovém editoru vytvořte topologii, která je na obrázku 11.
3. Vámi navržené schéma spusťte.
4. Pozměňte zpoždění jednotlivých linek a zjistěte jaký to má vliv na výpadky paketů.

Úvod:

V současné době existuje spousta programů pro simulaci problematiky sítí. Tyto programy jsou vytvářeny jako náhrada experimentálního testování. Mezi hlavní výhody použití takových programu patří bezesporu nižší finanční náklady.

A právě mezi programy tohoto typu řadíme program Network Simulator 2. Tento program byl vytvořen na universitě v Berkeley, ve dvou programovacích jazycích a to C++ a OTcl. Tento nástroj je velmi flexibilní a je možné jej použít na simulaci jak lokálních tak i rozsáhlých sítí. Tento nástroj patří ovšem mezi složitější systémy a proto se seznámíme pouze se základními funkcemi tohoto nástroje.

Pokud chceme vytvořit novou simulaci, otevřeme si libovolný textový editor. Nejprve vytvoříme objekt, který reprezentuje celý simulátor. To provedeme příkazem *set ns [new*

Simulator]. Základní stavební částí, v každé topologii programu NS2, je uzel. Ten se vytváří příkazem *set n0 [$ns node]*. Jednotlivé uzly jsou propojeny pomocí duplexních či simplexních linek, které se vytváří příkazem $ns duplex-link $n0 $n1 1Mb 10ms DropTail. Tento příkaz vytvoří duplexní linku mezi uzly n0 a n1, která má kapacitu 1Mb a zpoždění 10ms. Poslední parametr určuje typ fronty. V tomto případě je to fronta FIFO (First in fisrt out). Tento typ fronty je jeden s nejjednoduších. Paket, který první vstoupí do fronty, první také odchází. Když jednotlivé uzly propojíme potřebným počtem linek, vytvoříme mezi uzly provoz. To se provede tak, že k uzlu který má být zdrojem vysílání, přiřadíme protokol, kterým se má vysílání řídit. Mezi nejrozšířenější protokoly v dnešní době patří protokol UDP a TCP. Pokud chceme tedy připojit k uzlu n1 protokol TCP, použijeme následující příkazy.

set tcp [new Agent/TCP]

ns$ attach-agent $n1 $tcp

Dalším krokem pro vytvoření simulace je vytvořit příjímač vysílání. K tomu slouží příkaz:
set sink[new Agent/TCPSink]. Tento přijímač poté přidělíme k uzlu n2 příkazem:

ns$ attach-agent $n2 $sink

Posledním krokem je vytvoření provozu. Nejprve musíme určit protokol, kterým se bude provoz řídit. Jako protokol zvolíme protokol ftp a poté jej přiřadíme pod protokol TCP. Tuto operaci provedou následující řádky.

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type_FTP

Posledním krokem je určení počátku a konce vysílání. K tomu slouží následující příkazy.

$ns at 0,5 “$ftp start”

$ns at 3 “$ftp stop”

Číslice udává čas v sekundách,dobu kdy bude simulace spuštěna a kdy zastavena.

Součástí každého zdrojového kódu bývá i procedura finish, v níž jsou definovány akce, které mají být provedeny po ukončení simulace. Jako je uzavření souborů, spuštění vizualizačních nástrojů atd. Tato procedura se spouští obdobně jako začátek a konec provozu, tedy příkazem $ns at 0,5 “finish“. Celý zdrojový kód je uzavřen spuštěním celé simulace a to příkazem *$ns run.*

V praktické části této úlohy budeme simulovat výpadek linky a proto se seznámíme s příkazem, který vyřadí linku s provozu v určenou dobu. Příkaz má podobu *$ns rtmodel at 1.0 down $n1 $n2.* V čase 1 sekunda od spuštění simulace vyřadí s provozu linku mezi uzly

n1 a n2. Pro obnovení činosti této linky píšeme *$ns rtmodel at 3.0 up $n1 $n2* . Linka se tedy obnoví po třech sekundách od začátku simulace.

Poté co máme navrženou celou topologii, spustíme ji z terminálového okna příkazem *ns název_souboru.tcl* .Musíme se ovšem přesunout do adresáře, kde se soubor nachází. Po spuštění simulace máme možnost vidět celou simulaci díky nástroji NAM (Network AniMator). Ten pracuje nezávisle na NS2.

Uvedeme si jednoduchý příklad na topologii o dvou uzlech (Obr. 10), v níž dojde po dvou sekundách k výpadku linky a po třech sekundách k obnovení spojení.

Obr.č. 10

```
set ns[NewSimulator]
# Otevření souborů pro zápis krokovacích informací pro NAM
set namfile [open out.nam w]
$ns namtrace-all $namfile
# Definice procedury finish
proc finish{}{
   global ns tracefile1 namfile
   $ns flush-trace
   close $tracefile1
   close $namfile
   exec namout.nam &
   exit 0
}
  set n0 [$ns node]
set n1 [$ns node]
```

```
# Definice linek mezi jednotlivými uzly
$ns duplex-link $n0 $n1 2.5Mb 10ms DropTail
# Rozmístění uzlů
$ns duplex-link-op $n0 $n1 orient right
# Definice spojení TCP
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
# Definice protokolu ftp a přiřazení pod TCP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
#Přerušení linky a opětné obnovení
$ns rtmodel-at 2.0 down $n0 $n1
$ns rtmodel-at 3.0 up $n0 $n1

# Spuštění a ukončení vysílaní pomocí ftp
$ns at 1.0 "$ftp start"
$ns at 4.5 "$ftp stop"

$ns at 6.0 "finish"

# Zahájení simulace
$ns run
```

Praktická část této úlohy je zaměřena na unicastové směrování, což je směrování, kdy jsou data odesílána pouze jednomu příjemci. Máme několik protokolů, kterými se unicastové směrování může řídit. Mezi ty nejznámější patří protokol RIP (Routing Information Protocol) a OSPF (Open Shortest Path First).

RIP (Routing Information Protocol):

V současné době existují dvě verze protokolu RIP, a to RIPv1 a RIPv2. Oba dva protokoly používají pro rozhodování o nejlepší cestě k cíli k tzv. metriku. Cesta nám udává počet skoků, tedy počet uzlů, kterými paket projde než se dostane do cíle. Maximální počet uzlů byl zvolen na číslo 15. Každý uzel v síti si pravidelně aktualizuje své směrovací tabulky a to každých 30 sekund. Aktualizace získává od okolních uzlů, když příjme aktualizaci od sousedního uzlu, přičte k počtu skoků jedničku a poté aktualizaci posílá dále.

OSPF (Open Shortest Path First):

Protokol OSPF na rozdíl od protokolu RIP nepoužívá jako rozhodovací informaci počet skoků, ale princip sledování stavu linky. Každý uzel v síti, který využívá tento protokol, si vytváří databázi o topologii dané sítě. Využívá u toho algoritmu SPF (Shortest Path First). Směrování v síti je založeno na zasílání zpráv LSA, které nesou informaci o sítích, ke kterým je uzel připojen. Zpráva LSA je šířena záplavově, což znamená, že je posílána všemi směry. Každý uzel si díky této zprávě vytváří topologickou databázi. Tato databáze je pomocí Dijkstrůvova algoritmu převedena na topologii ve formě stromu. Tuto operaci provádí každý směrovač sám a každý směrovač je ve své databázi kořenem stromu.

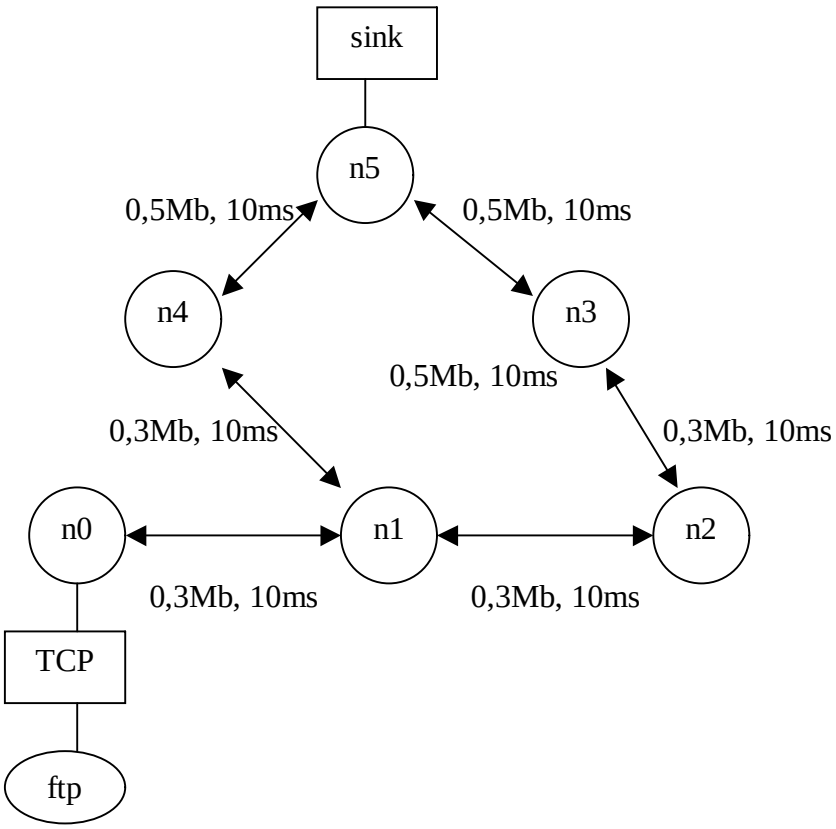
Mezi výhody protokolu OSPF patří rychlá konvergence (budování směrovací tabulky), z které plyne větší odolnost vůči smyčkám. Dále je méně náchylný na vznik chyb při směrování. Na druhou stranu je poměrně náročný na hardware.

Praktická část:

Praktická část se skládá z ověření vlastností unicastového směrování pomocí simulátoru NS2.

Úkol číslo 2:

Pomocí znalostí, které jste získali z teoretického úvodu vytvořte topologii, kterou můžete vidět na obrázku 11. Ta se skládá ze šesti uzlů (n0 – n5). Jednotlivé uzly jsou mezi sebou propojeny duplexní linkou, která má mezi uzly n0 – n1, n1 – n2, n1 – n4 a n2 – n3 kapacitu 0,3 Mb. Duplexní linky mezi uzly n3 – n5 a n4 – n5 mají kapacitu 0,5 Mb. Všechny linky mají zpoždění 10ms. Uzel 0 vysílá směrem k uzlu 5 data pomocí protokolu TCP, nad nímž pracuje protokol ftp. Data jsou nejprve řazeny do fronty FIFO a poté jsou posílány nejkratší cestou přes uzel 4. Spojení mezi uzly 1 a 4 je po jedné sekundě přerušeno a proto se změní cesta paketů a ty dále putují přes uzly 2, 3 a 5. Po 3,5 sekundách se spojení opět obnoví a pakety se vrátí do původní kratší cesty.



Obr.č. 11 vzor topologie

Úkol číslo 3:

Vámi vytvořený zdrojový text spusťte pomocí příkazu ns *název_souboru.tcl*. Pozorujte jak se mění cesta paketů při výpadku jednotlivých linek.

Úkol číslo 4:

Ve zbylém čase si zkuste změnit zpoždění jednotlivých linek a sledovat, jaký vliv má tato úprava na přenos paketů.

6.2 Laboratorní úloha číslo 2:

Simulace Multicastového směrováni v Dense módu a Sparse módu pomocí nástroje Network Simulátor 2 (NS2)

Zadání:

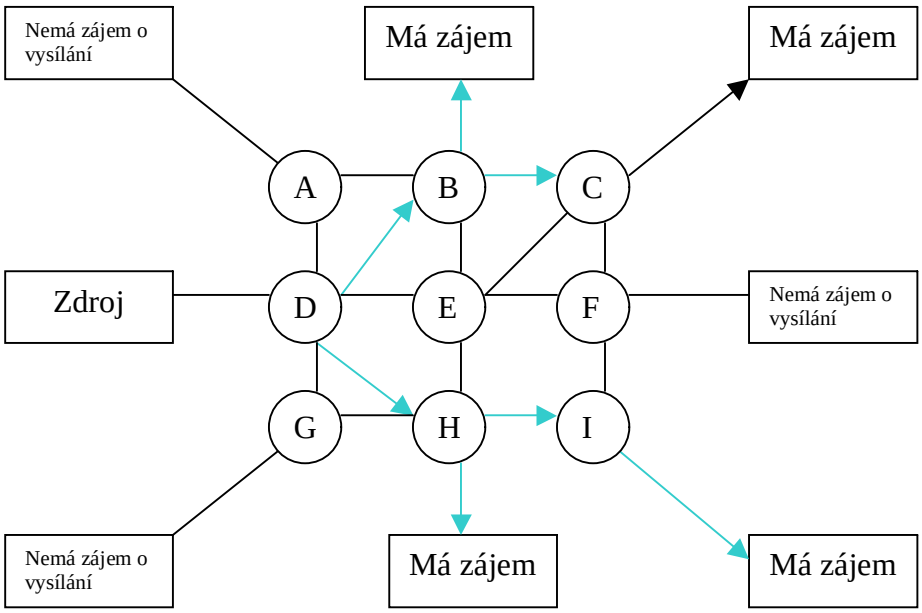
1. Prostudujte si teoretický úvod k multicastu
2. Vytvořte topologii, která je vidět na obrázku 14 tak, aby fungovala v DM.
3. Vytvořenou topologii upravte tak, aby pracovala v SM.

Úvod:

Unicastové směrování, které jsme si ověřili v minulé úloze, je nedostačující pro situace, kdy je potřeba stejná data poslat více příjemcům. Kdyby se taková situace řešila unicastovým vysíláním, musela by se stejná data vyslat tolikrát, kolik je příjemců. To je ovšem značně neefektivní. Z toho důvodu se začalo využívat multicastového (skupinového) směrování. Tato metoda posílá pakety multicastové skupině. Každý packet nese ve své hlavičce multicastovou adresu, která má tvar IP adresy a je v rozsahu 224.0.0.0 až 239.255.255.255, tyto adresy jsou také nazývány adresy třídy D. Multicastové směrování používá nezabezpečeného směrování pomocí UDP protokolu. To znamená, že nedochází k potvrzování jednotlivých paketů.

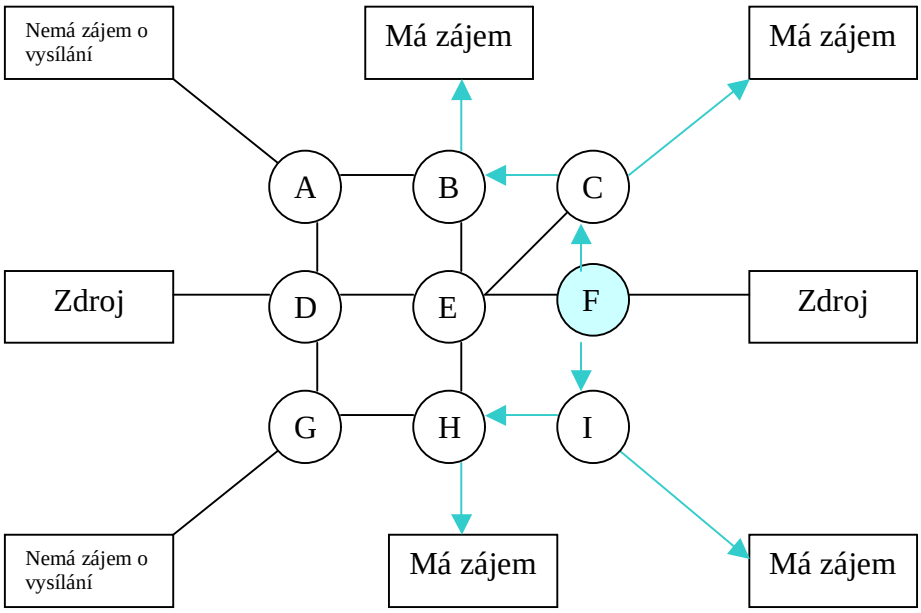
V současné době existují dvě metody multicastového směrování. První metoda je nazývána Dense mode a druhá Sparse mode.

Ještě než si něco povíme o jednotlivých módech, je důležité se seznámit s podobou směrovacích tabulek. Ty mají stromovou strukturu. První možností je tzv. zdrojový strom, u něj je kořenem stromu zdroj vysílání a přijímače vysílání tvoří listy tohoto stromu. Ukázku je možno vidět na obrázku 12. Pro označení zdrojového stromu bývá užívána notace (S,G), kde S udává adresu zdroje a G skupinovou adresu.



Obr.č.12 Zdrojový strom

Druhým typem stromové struktury je sdílený strom. Tento strom má, narozdíl od zdrojového stromu, kořen stromu na jiném místě než je zdroj vysílání. Tomuto kořenu se říká rendesvous point RP. Všechny uzly v síti vědí, kde RP bod leží a pokud mají zájem o nějaké vysílání, obrátí se na RP bod. Ukázka sdíleného stromu je na obrázku 13. Z něj je vidět, že uzel F plní funkci RP bodu. Pro označení tohoto stromu se užívá notace (*,G), zde hvězdička oznamuje, že strom není závislý na zdroji multicastu a G značí multicastovou adresu.



Obr.č.13 Sdílený strom

Dense mode:

Při tomto módu zdroj vysílá na některé s multicastových adres všem účastníkům, kteří jsou v síti. Data jsou tedy posílány i těm stanicím, které o vysílání nemají zájem. A proto tyto stanice mají možnost požádat, aby jim data nebyla posílána. To se provede pomocí takzvané Prune zprávy. Po jejím odeslání zdroj vysílání přestane do daného směru data vysílat. Prune zpráva má ovšem časově omezený účinek a je potřeba ji zasílat opakovně. Jinak by vysíláni dat bylo opět obnoveno. V případě, že stanice, která je odpojena z multicastové skupiny, má zájem o vysílání, vyšle zprávu graft. Stanice je poté ihned znovu připojena do multicastové skupiny.

Sparse mode:

Tato varianta je vhodná tam, kde je málo příjemců vysílání a z toho důvodu není všem uzlům v síti vysílání posíláno. V tomto módu je využíváno tzv.: rendez-vous pointu (RP). Zde má zájemce o vysílání možnost se s vysíláním “potkat“, protože pakety ze zdroje vysílání jsou posílány právě sem. Každý zájemce zašle žádost o zaslání paketů skupinového vysílání. S prvním paketem je zjištěn zdroj vysílání a je možné zažádat o příjem přímo od zdroje vysílání. Pokud příjemce už nemá zájem o vysílání, má možnost se od multicastové skupiny odpojit.

Implementace multicastu do NS2:

V minulé úloze jsme se seznámili se základní syntaxí programu NS2. V dnešním cvičení se seznámíme s příkazy, pomocí nichž implementujeme do simulace multicasting. Nejprve se seznámíme s implemenací Dense módu. Na začátku simulace musíme definovat, že se bude jednat o multicast, to zajistíme příkazem *$ns multicasting*. Poté je nutné přidělit multicastovou adresu, to provedeme příkaze *set group [Node allocaddr]*. Poté vytvoříme potřebný počet uzlů příkazem *set nod 6* , číslice na konci vyjadřuje počet uzlů. Až máme vytvořené uzly, použijeme cyklus for pro vytvoření multicastových uzlů. K tomu slouží příkaz:

for {set i 1}{$i <= $nod}{incr i}{
set n($i) [$ns node]}

Poté definujeme, v jakém módu bude multicast pracovat. Pro DM tedy píšeme:
set mprot DM
V dalším kroku je třeba zajistit, aby všechny uzly pracovaly s multicastovým protokolem, k tomu poslouží příkaz *set mrthandle [$ns mrtproto $mproto]*. Jak jsme si výše řekli, multicast pracuje s UDP protokolem a proto je potřeba vytvořit agenta s protokolem UDP a přiřadit jej k některému z uzlů. To provedeme příkazem *set udp1 [new Agent/UDP]*. Agenta přiřadíme k uzlu, který si zvolíme příkazem $ns attach-agent $n(1) $udp1. Poté vytvoříme zdroj vysílání a přiřadíme jej k agentovy UDP. K tomu použijeme následující řádky.
set src1 [new Application/Traffic/CBR]
$src1 attach/agent $udp1
$udp1 set dst_addr_ $group
$udp1 set dst_port_ 0
Tímto jsme vytvořili zdroj vysílání. Ještě nám zbývá vytvořit přijímací agenty. K tomu slouží následující příkaz set rcvr [new Agent/LossMonitor]
Na závěr musíme určit, kdy se budou uzly přihlašovat k multicastingové skupině, a kdy odhlašovat. K tomu nám poslouží dva příkazy, první, pro přihlášení do skupiny je *$ns at 0.5 "$n(2) join-group $rcvr $group"* a pro přihlášení je tu příkaz *$ns at 1.5 "$n(2) leave-group $rcvr $group.*

Druhým módem, který si v této úloze procvičíme, je sparse mód. Implementace Sparse módu je obdobná jako u Danse módu, jen při definici multicastového protokolu budeme psát následující:
BST set RP_($group) $n(2)

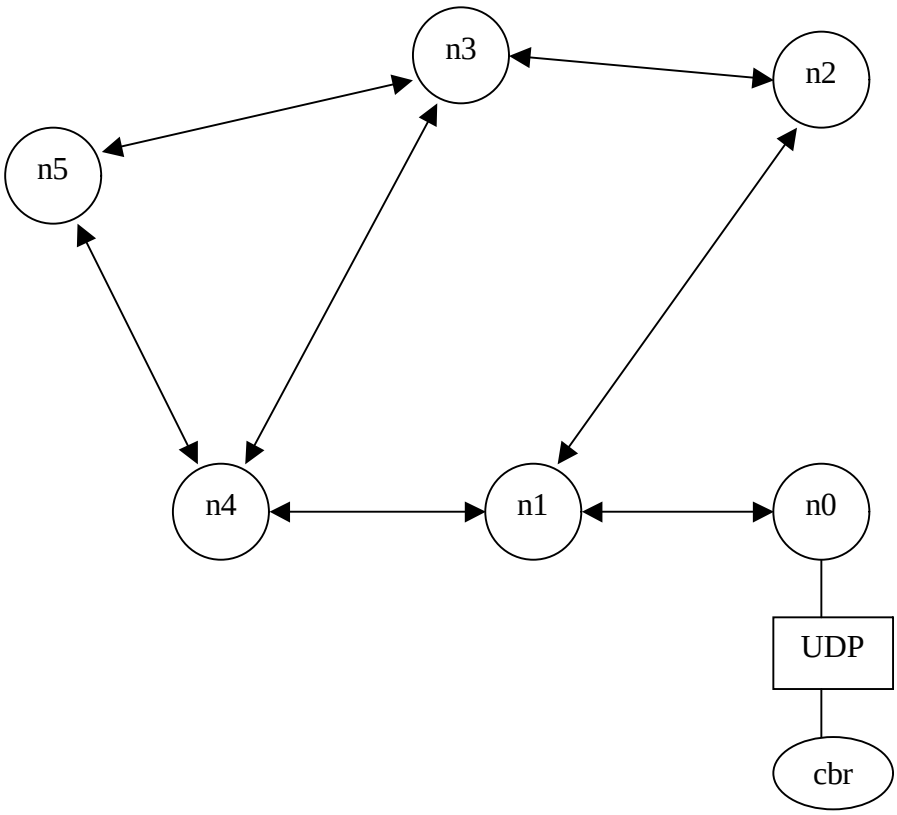
$ns mrtproto BST

V tomto příkazu číslo 2 určuje, že uzel bude sloužit jako RV point. Ostatní příkazy jsou shodné s DM módem.

Praktická část:

Úkol číslo 2:

Pomocí znalostí, které jste získali z předchozího cvičení a z teoretické části, vytvořte topologii, která je na obrázku č.14. Jako zdroj vysílání zvolte uzel n0. K tomuto uzlu se budou uzly n1 až n5 postupně jeden po druhém připojovat v rozmezí 1s. Poté se budou zase v pořadí uzel n5 až n1 odpojovat zase v rozmezí 1 sekundy. Vámi navrženou topologii poté spusťte a sledujte jak komunikace probíhá.



Obrázek č.:14 vzor topologie

Úkol číslo 3:

Vámi vytvořenou topologii upravte tak, aby fungovala ve sparse módu. Jako RV point zvolte uzel číslo n1. Tento zdrojový kód spusťte a porovnejte jak funguje přístup k multicastové skupině oproti DM módu.

7.Závěr:

V této práci jsem se pokusil seznámit čtenáře se základním používáním programu NS2 a tvorbou jednoduchých simulací. Abychom tyto simulace mohly vytvářet, tak jsme se ve čtvrté kapitole seznámili se základními protokoly sady TCP/IP. V poslední části jsem nasimuloval jednoduchou topologii o čtyřech uzlech.

Součástí mé bakalářské práce jsou i dvě laboratorní úlohy. První z nich je zaměřena na unicástové směrování v sítích. Tato úloha je koncipována tak, aby student po přečtení teoretického úvodu úlohy, byl schopen vytvořit první vlastní simulaci v NS2. Seznámí se se základní syntaxí NS2 a následně s tím, jak směrování v unicastu funguje. Druhá úloha plynule navazuje na znalosti získané z první úlohy a rozšiřuje je o možnosti NS2 pro simulaci multicastu. Hlavním cílem druhé úlohy je, aby student pochopil princip jednotlivých variant multicastu.

Věřím, že tato bakalářská práce bude kvalitní učební pomůckou pro všechny, kteří mají zájem o rozšíření znalostí ve směrování v dnes používaných sítích.

8.Použitá literatura:

1) Information Sciences Institute, "The Network Simulator - ns-2", June 2004, [online] URL:<http:// www.isi.edu/nsnam/ns/>
2) M.A. SPORCTAK, Směrování v sítích IP, ComputerPress, a.s.,2004, ISBN: 80-251-0127-4
3) *Wikipedie: Otevřená encyklopedie: TCP* [online]. c2007 [citováno 19. 12. 2007]. URL <http://cs.wikipedia.org/w/index.php?title=TCP&oldid=2026403>
4) *Wikipedie: Otevřená encyklopedie: Sada protokolů Internetu* [online]. c2007 [citováno 19. 12. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Sada_protokol%C5%AF_Internetu&oldid=1896300>
5) *Wikipedie: Otevřená encyklopedie: SYN-flood* [online]. c2008 [citováno 3. 06. 2008]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=SYN-flood&oldid=2208807>
6) *Wikipedie: Otevřená encyklopedie: Sada protokolů Internetu* [online]. c2008 [citováno 3. 06. 2008]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Sada_protokol%C5%AF_Internetu&oldid=2600674>