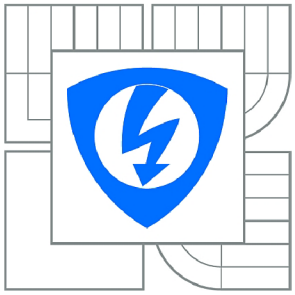
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SIMULAČNÍ ÚLOHY V NS2 OSVĚTLUJÍCÍ ZNALOSTI Z PŘEDMĚTU MPKT

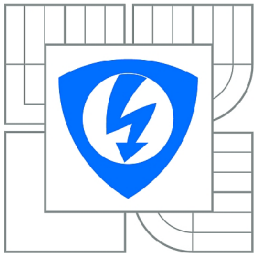
SIMULATION SCENARIOS IN NS2 DEMONSTRATING KNOWLEDGE OBTAINED IN THE MPKT COURSE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN MAŠEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN JEŘÁBEK
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Jan Mašek ***ID:*** 106624
Ročník: 3 ***Akademický rok:*** 2009/2010

NÁZEV TÉMATU:

Simulační úlohy v NS2 osvětlující znalosti z předmětu MPKT

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte problematiku simulace počítačových sítí a zaměřte se především na Network Simulator 2 (NS2). Prostudujte strukturu laboratorních úloh v předmětu Pokročilé komunikační techniky a chování protokolů popisovaných v přednáškách (zejména se zaměřte na http protokol). V rámci bakalářské práce navrhněte a připravte minimálně dvě kompletní laboratorní úlohy na vybranou problematiku. Úlohy vytvořte tak, že umožní studentům ověření znalostí získaných na přednáškách a doplňte je vzorovým řešením s detailním popisem.

DOPORUČENÁ LITERATURA:

[1] Altman, E.; Jimenéz T.: NS Simulator for beginners, Lecture notes, Universita de Los Andes, Venezuela, 2003.
[2] Fall, K.: The NS Manual (Notes and Documentation), UC Berkeley, USA, 2009.
[3] Jeřábek, J.: Skripta k předmětu Pokročilé komunikační techniky, Vysoké učení technické v Brně, 230 s., Brno 2009.

Termín zadání: 29.1.2010 ***Termín odevzdání:*** 2.6.2010

Vedoucí práce: Ing. Jan Jeřábek

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

UPOZORNĚNÍ:

Abstrakt

V bakalářské práce jsou navrženy simulační úlohy datových sítí pro potřeby předmětu Pokročilé komunikační techniky (MPKT). Sítě jsou simulovány v prostředí programu Network Simulator 2 (NS2). V úvodu je stručný popis programového prostředí NS2 a dále základy jazyka Tcl, používaného k simulaci. Uvedeny jsou i možnosti simulace protokolu HTTP s použitím OTcl tříd PagePool. Následuje návrh celkem šesti simulačních úloh. První úloha využívá třídu PagePool/Math a zobrazuje nejjednodušší komunikaci mezi klientem, serverem a cache. Další úloha, využívající třídu PagePool/CompMath zobrazuje komunikaci mezi klientem, cache, webovým serverem, FTP serverem a FTP klientem. Úloha využívající třídu PagePool/WebTraf simuluje vzájemnou komunikaci mezi dvěma klienty a třemi servery. Poslední tři úlohy využívající třídy PagePool/ProxyTrace se od sebe liší rozdílnou topologií a průběhem komunikace. Úlohy této třídy jsou nejsložitější a jejich vstupní soubory umožňují nadefinovat celou komunikaci. Výstupy úloh jsou grafy, logovací soubory HTTP komunikace a k poslední úloze je zpracován podrobný návod.

Klíčová slova

HTTP, Network Simulator 2, cache, webový server, webový klient, PagePool, Tcl

Abstract

The aim of the bachelor´s thesis is to develop simulation tasks for modeling of data networks. The tasks will be used in the course Modern communication technique (MKPT). The networks are simulated in the environment of Network Simulator 2 (NS2). A brief description of program environment NS2, as well as basics of Tcl language is mentioned in the introduction. Described is the possibility of simulation HTTP protocol in NS2 using OTcl PagePool classes. Totally six proposed tasks form the core of the thesis. The first task uses PagePool/Math and displays the simplest communication between client, server and cache. Next task uses PagePool/CompMath and it shows the communication between client, cache, Web server, FTP server and FTP client. The task, that uses PagePool/WebTraf class, simulates communication between two clients and three servers. The last three tasks are of different topologies and course of communication using PagePool/ProxyTrace. The exercises of this class are the most complicated and their input files allow defining whole communication. The outputs of exercises are given in graphs, log files of HTTP communication and the last task contains detailed instructions.

Key words

HTTP, Network Simulator 2, cache, web server, web client, PagePool, Tcl

Citace

MAŠEK, J. Simulační úlohy v NS2 osvětlující znalosti z předmětu MPKT. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 63 s. Vedoucí bakalářské práce Ing. Jan Jeřábek.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Simulační úlohy v NS2 osvětlující znalosti z předmětu MPKT jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny 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 č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne . . . .

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Janu Jeřábkovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování práce.

V Brně dne . . . .

podpis autora

Obsah

Úvod

Cílem práce je navrhnout laboratorní úlohy k předmětu Pokročilé komunikační techniky, které se zabývají problematikou protokolu HTTP. Laboratorní úlohy budou navrženy v prostředí Network Simulator verze 2 (NS2).

Práce se zabývá popisem NS2 a některých aplikací, které NS2 obsahuje, popřípadě instalací NS2 do různých operačních systémů. Dále jsou zde zmíněny základní vlastnosti a syntaxe jazyka Tcl, který je klíčový při vytváření simulačních úloh v NS2. V práci jsou podrobně rozepsány možnosti simulace protokolu HTTP a následně okomentovány všechny simulace, které jsem vytvořil. K jedné vybrané úloze je vypracován i podrobný návod.

1 Network Simulator 2 (NS2)

Network Simulator 2 je objektově orientovaný volně šiřitelný simulační nástroj řízený diskrétními událostmi, který se používá k simulaci síťového provozu [1, 2]. Byl vyvinut na Univerzitě v Berkeley jako nástupce programu REAL Network Simulator, který byl vyvíjen od roku 1989. V roce 1995 byl NS2 podporován agenturou DARPA (agentura pro výzkum pokročilých obranných systémů v USA) skrz VINT projekt. Později byl podporován nadací NSF (Národní vědecká nadace v USA) skrz projekt CONSER (Collaborative Simulation for Education and Research). V současné době už existuje nová verze síťového simulátoru, o které se zamýšlí, že by měla v budoucnu nahradit NS2. Tato verze nese označení NS3 a nabízí oproti NS2 více funkcí. Přesto se NS2 v dnešní době používá stále velmi intenzivně a to především díky tomu, že je výborně odladěn a jeho chování a výstupy jsou dostatečně prověřeny.

1.1 Funkce NS2

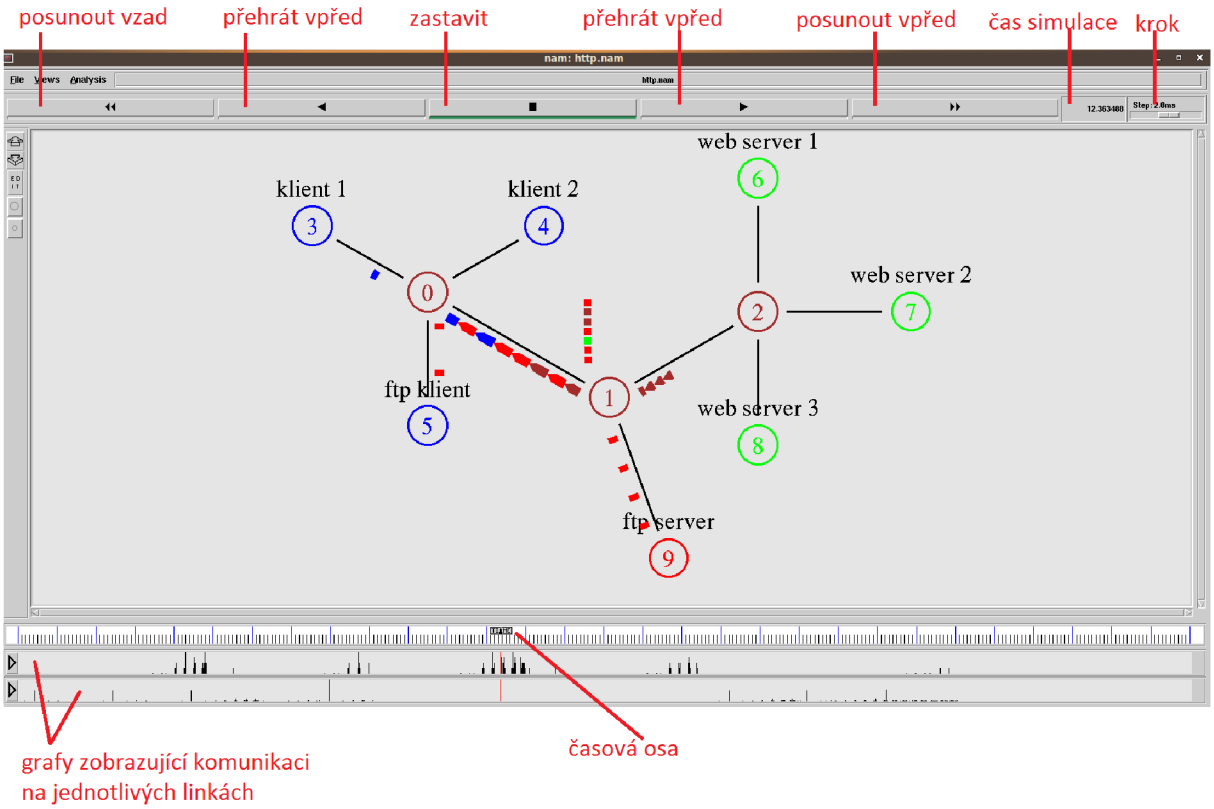
NS2 byl naprogramován v jazycích C++ a OTcl. Jazyk C++ se používá pro operace s daty a jazyk OTcl zase pro operace řízení. V NS2 je implementována celá řada protokolů a zahrnuje komunikaci v drátových, bezdrátových a satelitních sítích. Protože je NS2 stále vyvíjen a zdokonalován, tak je možné do programu doinstalovat i další typy protokolů. Na jednom modelu lze simulovat i více protokolů.

Program zahrnuje ve standardní instalaci tyto funkce vrstev ISO/OSI modelu [3]:

- Aplikační vrstva zahrnuje tyto zdroje síťového provozu: HTTP (Hypertext Transfer Protocol), telnet, FTP (File Transfer Protocol), CBR (Constant Bit Rate),...
- Transportní vrstva používá protokoly: UDP (User Datagram Protocol), TCP (Transmission Control Protocol), RTP (Real-time Transport Protocol),…
- Síťová vrstva provádí směrování v sítích užitím algoritmů: PIM-SM (Protocol Independent Multicast – Sparse Mode), AODV (Ad hoc On-Demand Distance Vector), DSR (Dynamic Source Routing Protocol),….
- 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 pracuje s: CSMA/CD (Carrier Sense Multiple Access with Collision Detection), MAC (Medium Access Control protocol),... [4, 5]

1.2 NAM (nástroj Network Animator)

NAM je nástroj, který se používá pro zobrazení grafické reprezentace výsledků simulace. Je založený na Tcl/TK, kde TK je knihovna sloužící pro práci s grafickým prostředím [6]. Jako vstup mu slouží trasovací soubor, který je vytvořen pomocí Tcl skriptu. NAM může zobrazit uzly, linky, datový tok paketů, jejich řazení do front nebo zahazování. Dále je možné nastavit barvy jednotlivých uzlů, popisky linek a paketů, což přispívá k přehlednosti výstupu. Po startu se NAM přesune na start simulace do času $t = 0.0s$. Nyní se zvolí krok simulace, který je v rozsahu od 1µs do 794,3ms a určuje měřítko zobrazení snímků v simulaci. Pro lepší orientaci toku paketů se mohou zapnout grafy, které zobrazují datový tok na jednotlivých linkách nebo zahozené pakety jednotlivých linek. Pokud zobrazená topologie není moc dobře přehledná, tak se dá ještě dodatečně editovat například přiblížením, oddálením topologie nebo změnou polohy uzlů a linek popřípadě změnou velikosti zobrazení uzlu. Okno aplikace NAM spolu s popisem ovládacích prvků je zobrazeno na obr. 1.2. Simulace se spustí tlačítkem přehrát vpřed. Lze využít i zpětné přehrávání tlačítkem přehrát vzad. Tlačítka posunout vpřed a vzad slouží ke zrychlenému posunu v čase a tento krok je pětadvaceti násobkem kroku simulace.



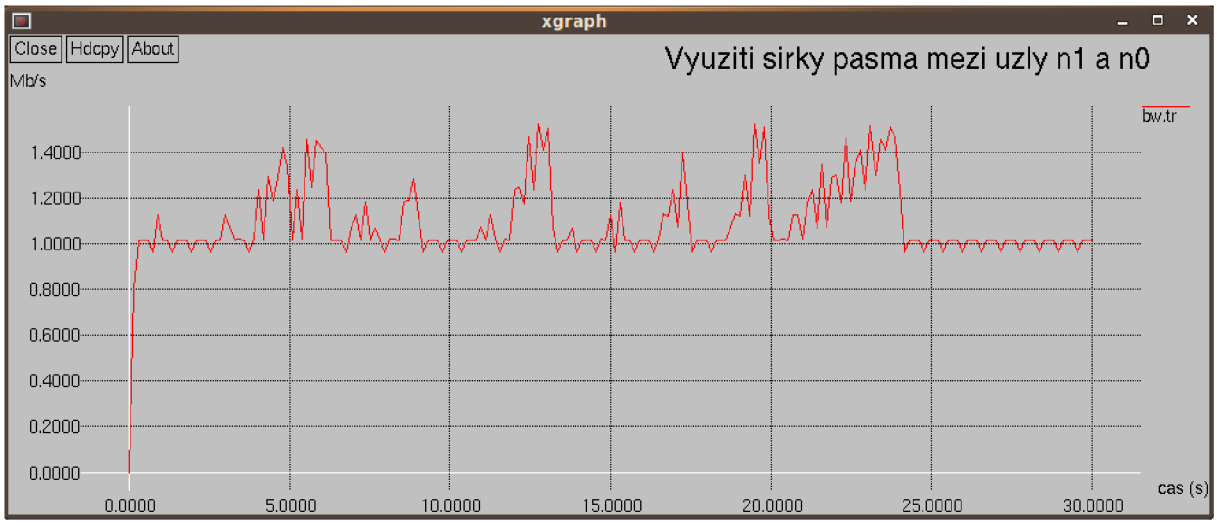
Obr. 1.2: Okno aplikace NAM.

1.3 XGRAPH (nástroj pro zobrazení grafů)

Program XGRAPH slouží k zobrazení grafických závislostí, které si nadefinujeme v Tcl skriptu. Narozdíl od grafů, které zobrazuje aplikace NAM (zobrazuje pouze datový tok na jednotlivých linkách), dokáže XGRAPH zobrazit mnoho druhů grafických závislostí, které si nadefinujeme. Mezi další výhody XGRAPHu patří například zobrazení názvu grafu a popisků jednotlivých os a zobrazení jednotek na osách. Při spuštění Tcl skriptu je vytvořen trasovací soubor, který je interpretován programem XGRAPH. Okno aplikace je zobrazeno na obr. 1.3. Aplikace dokáže výstupní graf uložit jako soubor typu *.ps (postscript), což usnadňuje jeho další využití. K zobrazení grafu, jeho názvu, popisků os a určení zobrazené velikosti grafu slouží příkaz, napsaný v Tcl skriptu, který spustí XGRAPH s následujícími parametry: `exec xgraph bw.tr -x "cas (s)" -y "Mb/s" -t "Vyuziti sirky pasma mezi uzly n1 a n0" -geometry 1000x400 &`

Vysvětlení jednotlivých parametrů příkazu:

- `exec xgraph bw.tr` – Spustí program XGRAPH, který otevře vstupní trasovací soubor bw.tr.
- `-x "cas (s)"` –Nastavení popisku pro osu x.
- `-y "Mb/s"` –Nastavení popisku pro osu y.
- `-t "Vyuziti sirky pasma mezi uzly n1 a n0"` – Název grafu
- `-geometry 1000x400`– Nastavení rozměrů okna po spuštění aplikace XGRAPH, první hodnota určuje vodorovný rozměr a druhá hodnota rozměr svislý.



Obr. 1.3: Okno aplikace XGRAPH.

2 Instalace programu NS2

Program lze instalovat do unixových operačních systémů (FreeBSD, Linux, SunOS, Solaris) nebo pomocí emulátoru unixového prostředí Cygwin do operačního systému Windows. Před instalací programu NS2 je nejprve nutné nainstalovat tyto balíčky: *xorg, gcc, g++*. Pro instalaci jsem stáhl nejnovější verzi programu NS2 (ns-allinone-2.34.tar.gz) z [http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.34]. Tento balík obsahuje všechny potřebné programy a po rozbalení archivu se provede instalace příkazem `./install`. NS2 jsem testoval jak z prostředí Cygwinu, tak i z vizualizovaného systému Ubuntu 9.10 s pomocí programu VirtualBox [http://www.virtualbox.org/]. Pracování s NS2 je pohodlnější v Ubuntu 9.10 než v emulačním prostředí Cygwin.

Sada NS2 allinone ve verzi 2.34 obsahuje tyto verze balíčků:

- nam-1.14 (aplikace Network Animator),
- ns-2.34 (aplikace Network Simulator),
- otcl-1.13 (objektově orientovaný jazyk Tcl),
- tcl8.4.18 (jazyk Tcl),
- tclcl-1.19 (Tcl s třídami),
- tk8.4.18 (knihovna rozšiřující Tcl o grafické prostředí),
- xgraph-12.1 (aplikace XGRAPH),
- zlib-1.2.3 (knihovna potřebná pro funkci aplikace NAM).

3 Jazyk Tcl

Tcl (Tool Command Language) je interpretovaný programovací jazyk, jehož předchůdcem byl jazyk Lisp (List Processing), a který byl vyvinut Johnem Ousterhoutem v osmdesátých letech minulého století. Tcl používá knihovnu TK, která slouží pro práci s grafickým prostředím.[7]

Výhody Tcl:

- jednoduchost
- rychlá tvorba aplikací
- snadná návaznost na jiné programovací jazyky (C, C++, JAVA)

3.1 Základy programování v Tcl

Kód programu se může psát do obyčejného textového editoru, ale vhodnější je použít některý z textových editorů s pokročilými funkcemi jako jsou například automatické barevné odlišení různých částí kódu podle jeho typu. Kód programu je pak mnohem přehlednější a lépe se v něm orientuje. Mezi tyto textové editory patří v linuxovém operačním systému program gedit a ve windowsovém operačním systému se dají použít volně stažitelné programy Notepad++ [8] nebo PSPad [9]. Vytvořený program má příponu *.tcl. Základní příkazy a syntaxe jsou vysvětleny v ukázkách kódu níže.

Do proměnné zapisujeme pomocí příkazu „set“.

```
set a 3
```

Proměnná „a” nabývá hodnoty 3.

Jednořádkové poznámky k programu se píší za znak „#“.

```
#Zapíše řetězec "Hello world" do proměnné "c"
set c "Hello world"
```

Zrušení proměnné se pak provádí příkazem „unset“.

```
unset c
```

Substituce je nahrazení jména proměnné její hodnotou. Používá se znak „$“.

```
set c $a
```

Proměnné „c“ je přiřazena hodnota „a“. Tedy $c = 3$.

Matematické operace se provádějí pomocí příkazu „expr“.

```
set x [expr $a + $c]
```

Do proměnné „x“ se uloží hodnota součtu proměnných „a“ a „c“.

Mezi další matematické a logické operace, které se mohou použít, patří:

- rozdíl (-), součin (*), podíl (/),
- bitové posuny doleva a doprava (<<, >>), bitová negace (~), logická negace (!),
- bitový operátor AND (&), OR (|), XOR (^),
- logický součin (&&), logický součet (||),
- větší než (>), menší než (<), rovnost (==), nerovnost(!=).

4 Simulace protokolu HTTP

Většina aplikací, které jsou implementovány v OTcl (aplikace FTP, telnet, CBR), jsou simulovány NS2 pouze jako jakési „virtuální “ aplikace, které nepřenášejí svá reálná data, ale jediné co lze u těchto aplikací konfigurovat je velikost dat a čas, kdy jsou data přenesena. Výjimkou je právě web caching, který posílá v simulaci podobná data, jaká jsou posílána v reálném síťovém provozu. V simulaci HTTP server posílá HTTP hlavičky do cache a klientům. HTTP hlavičky obsahují důležité informace, které jsou následně využívány cache pamětí. [10]

4.1 Přenos dat mezi aplikacemi

Pro přenos dat mezi aplikacemi a transportními agenty se používá speciální jednotka ADU (application-level data unit). Transportní agenti posílají přijatá data dále formou paketů. Posílání dat lze realizovat pomocí transportních protokolů UDP a TCP, čemuž pak odpovídají vlastnosti a chování přenosu.

4.2 HTTP objekty v NS2

Simulovat protokol HTTP se dá pomocí tří HTTP objektů, což jsou servery, cache a klienti. Klient je vlastně webový prohlížeč v počítači, cache (vyrovnávací paměť, která je součástí počítače) se používá pro uložení již navštívených webových stránek a pro rychlé znovunačtení stránek do prohlížeče. Klient zasílá požadavky o stáhnutí určité webové stránky webovému serveru a ten odesílá data zpět. Jako transportní agent v NS2 se standardně používá agent SimpleTcp, který je součástí třídy TcpApp. SimpleTcp má velmi podobné vlastnosti jako agent UDP. To znamená, že neopravuje chyby, neřídí datový tok a nepodporuje segmentaci paketů. Agent SimpleTcp se tedy nedá použít v sítích se ztrátou paketů pro simulaci HTTP. Výhoda agenta SimpleTcp spočívá ve zjednodušení trasovacího souboru a tím pádem k lepší orientaci ve výstupních souborech.

4.3 Page pool

Generátor webových stránek PagePool a jeho odvozené třídy jsou používány servery a klienty ke generování údajů o webových stránkách. Třída PagePool obsahuje tyto odvozeniny: PagePool/Math, PagePool/CompMath, PagePool/ProxyTrace, PagePool/Client, PagePool/WebTraf. Některé z nich budou podrobněji popsány dále. Pro simulaci s generátorem stránek PagePool je potřeba nastavit vlastnosti web serveru, klienta i cache a vzájemně je propojit. [10]

Klient, který je jednoduchým webovým prohlížečem, je popsán třídou Http/Client. Následující příkazy vytvoří proměnnou klient a nastaví její vlastnosti.

```
set client [new Http/Client $ns <uzel>]
                #Vytvoření proměnné definující klienta
                a jeho přiřazení k uzlu

$client set-interval-generator <rv>
                #Nastavení hodnoty  časového intervalu, kdy
                bude generátor připojený ke klientovi vysílat
                požadavky o stáhnutí webové stránky,kde <rv>
                je náhodná proměnná

$client set-page-generator <PagePool>
                #Připojení zvoleného generátoru stránek ke
                klientovi

$client log <logovací soubor>
                #Zaznamenání HTTP událostí souvisejících
                s klientem do předem vytvořeného
                logovacího souboru

$client connect <cache>
                #Klient se spojí s cache

$client start-session <cache> <server>
                #Klient začne komunikovat se serverem a
                komunikace jde přes cache
```

Chování webového serveru je popsáno třídou Http/Server. Konfigurace je jednoduchá. Stačí vytvořit server, připojit vybraný generátor stránek a pak už jen server čeká, dokud mu nepřijde požadavek na webovou stránku. Server přijímá požadavky typu GET a IMS (If-Modified-Since). Při přijmutí požadavku GET odešle server požadovanou stránku. Při přijmutí požadavku IMS se srovnává čas modifikace stránky v požadavku s časem stránky v generátoru stránek. Pokud je k dispozici novější stránka, tak ji server pošle do cache. Pokud

stránka k dispozici není, tak server pošle o tom jen informaci do cache. Příkazy na vytvoření serveru a přiřazení generátoru stránek jsou uvedeny níže.

```
set server [new Http/Server $ns <uzel>]
                    #Vytvoření proměnné definující server
                    a jeho přiřazení k uzlu

$server set-page-generator <PagePool>
                    #Připojení zvoleného generátoru stránek
                    k serveru

$server log <logovací soubor>
                    #Zaznamenání HTTP událostí, souvisejících
                    se serverem, do předem vytvořeného
                    logovacího souboru
```

Jednoduchá HTTP cache je popsána třídou Http/Cache. Vytvoření HTTP objektu cache se provede stejným příkazem, jako se vytvářel objekt serveru.

```
set cache [new Http/Cache $ns <uzel>]
                    #Vytvoření proměnné definující cache a
                    přiřazení k uzlu

$cache log <logovací soubor>
                    #Zaznamenání HTTP událostí, souvisejících
                    s cache, do předem vytvořeného logovácího
                    souboru


$cache connect <server>
                    #Cache se spojí se serverem
```

Všechny události, které se stanou na klientovi, serveru a cache se pokud to bude nastaveno tak, jako v předcházejících příkladech kódu, zaznamenávají do logovacího souboru. Tento soubor obsahuje informace o komunikaci mezi HTTP objekty. Zaznamenán je čas, kdy se událost stala, pak na kterém HTTP objektu nastala, dalšími informacemi jsou identifikační číslo stránky nebo jiné údaje, které jsou součástí určité HTTP události.

4.3.1 PagePool/Math

Generátor PagePool/Math je nejjednodušším generátorem webových stránek. Generuje pouze jednu stránku pomocí několika náhodných proměnných. Pro simulaci komunikace mezi klientem, vyrovnávací pamětí a webovým serverem za použití generátoru stránek PagePool/Math, se využijí obecné příkazy uvedené níže.

```
ranvar-size <rv>        #Nastavení velikosti stránky v bytech
                        pomocí náhodné proměnné <rv> = random
                        variable

ranvar-age <rv>         #Nastavení životnosti stránky v
                        sekundách pomocí náhodné proměnné <rv>
```

4.3.2 PagePool/CompMath

Generátor webových stránek PagePool/CompMath má obdobnou funkci jako PagePool/Math. Hlavním rozdílem je, že PagePool/CompMath používá vložené objekty jako součást webových stránek. Vloženým objektem může například být obrázek, který webová stránka obsahuje. Jako parametry se nastavují: velikost hlavní stránky, velikost objektu, počet objektů, doba životnosti objektů. Tyto specifické parametry jsou uvedeny níže jako obecné příkazy.

```
set main_size_ <hodnota>    #Nastavení velikosti hlavní
                            stránky v bytech
set comp_size_ <hodnota>    #Nastavení velikosti objektu
                            v bytech
set num_pages_ <hodnota>    #Nastavení počtu objektů na
                            jednu stránku
ranvar-obj-age <rv>         #Nastaví životnost objektu v
                            sekundách
ranvar-main-age <rv>        #Nastaví životnost hlavní
                            stránky v sekundách
```

4.3.3 PagePool/ProxyTrace

V předchozích dvou třídách PagePool se při simulaci posílá vždy jeden typ stránky v definovaném intervalu. V této třídě PagePool/ProxyTrace může být simulovaný provoz značně složitější. Mohou se například využít trasovací soubory z reálných proxy cache serverů, které se pomocí programu (webcache-trace-conv.tar.gz), který se může stáhnout z [http://www.isi.edu/nsnam/dist/webcache-trace-conv.tar.gz], dají převést na dva soubory, jež jsou použity při simulaci s využitím této třídy. Prvním vstupním souborem je pglog, který obsahuje informace o webových stránkách jako je ID webové stránky a její velikost. Druhým vstupním souborem je reqlog, který popisuje komunikaci mezi serverem cache a klienty. Díky němu je možné definovat čas, kdy událost stažení určité stránky z určitého webového serveru nastane, dále určuje, který klient tento požadavek vyslal. Soubor pglog je tedy využíván webovým server a soubor reqlog mají k dispozici webový klienti. Při simulaci je možné, aby

cache představovala vyrovnávací paměť v počítači, ve kterém klient představuje jednoduchý webový prohlížeč. Další možností je proxy cache server, který je využíván jako úložiště webových stránek pro větší počet klientů. Webové stránky se dají rozdělit na dva typy podle intervalu jejich modifikace v cache. Prvním typem jsou stránky dynamické, které musejí být v cache aktualizovány velmi často (například v intervalech minut). Dalším typem jsou stránky statické, které mohou být uchovány v cache i měsíce bez nutnosti aktualizace webovým serverem. V simulaci jsou tedy dva generátory modifikačních intervalů pro statické i dynamické webové stránky. Počet dynamických a statických stránek se dá zvolit.

```
set-reqfile <soubor>        #Připojení vstupního souboru
                            "reqlog". Soubor využívají klienti.

set-pagefile <soubor>       #Připojení vstupního souboru
                            "pglog". Soubor využívá server.

set-client-num <hodnota>    #Nastavení počtu HTTP klientů.


bimodal-ratio <hodnota>     #Nastavení hodnoty dvourežimového
                            modelu (v rozmezí 0 - 1). Hodnota
                            (po vynásobení 100) určuje, kolik
                            procent stránek bude dynamických.
ranvar-dp <rv>              #Nastaveni generátoru modifikačních
                            intervalů pro dynamické stránky.

ranvar-sp <rv>              #Nastaveni generátoru modifikačních
                            intervalu pro statické stránky.
```

4.3.4 PagePool/WebTraf

Třída PagePool/WebTraf se používá pro simulaci webového provozu. Tato třída nemá nic společného s třídami HttpApp na rozdíl od ostatních odvozenin třídy PagePool. To znamená, že ve výsledné simulaci je brán důraz hlavně na webový provoz a že se zde nepřenášejí HTTP hlavičky a tím pádem zde nejsou žádné HTTP události, které by se zaznamenaly do logovacího souboru. Také se zde nepoužívá cache paměť k ukládání dat. Hlavní funkcí třídy WebTraf je možnost uskutečnění komunikace mezi vetšími počty klientů a serverů, kde může každý klient komunikovat s různými servery a požadovat z nich stáhnutí různých webových stránek. WebTraf využívá dvě třídy naprogramované v C++. První třída

class WebTrafSession definuje relace, které musí být vytvořeny pro spuštění komunikace mezi servery a klienty. A druhá třída *class WebPage* definuje vlastnosti webových stránek. Obecné příkazy, které se použijí, jsou uvedeny níže[10].

```
set-num-session <počet relací>   #Nastavení počtu relací
                                 ve třídě PagePool/WebTraf.

set-num-server <počet serverů>   #Nastavení počtu web
                                 serverů.

set-num-client <počet klientů>   #Nastavení počtu
                                 klientů.

set-client <id> <uzel>           #Přiřazení klienta k uzlu,
                                 kde <id> určuje pořadové číslo
                                 klienta.

set-server <id> <uzel>           #Přiřazení serveru k uzlu,
                                 kde <id> určuje pořadové číslo
                                 serveru.

create-session  <id> <počet stránek> <čas startu relace>
                <interPage> <pageSize> <interObj> <objSize>
          #Vytvoří webovou relaci, kde <id> je pořadové číslo
          relace. Následující zmíněné proměnné se musí
          definovat jako náhodné proměnné (random variable).
          Proměnná <interPage> definuje časový interval mezi
          dvěma po sobě jdoucími stránkami, proměnná
          <pageSize> určuje počet objektů, které obsahuje
          webová stránka, proměnná <interObj> definuje časový
          interval mezi po sobě jdoucímu objekty a proměnná
          <objSize> určuje velikost jednoho objektu.


set-interPageOption <option>
          #Při nastavené hodnotě <option> = 1 je hodnota
          proměnné $interPage brána jako časový úsek mezi
          posledním přijatým paketem první stránky a prvním
          přijatým paketem stránky následující.

          #Při nastavené hodnotě <option> = 0 je hodnota
          proměnné $interPage brána jako časový úsek mezi
          prvním přijatým paketem první stránky a prvním
          přijatým paketem stránky následující.

          #Defaultně je zvolena možnost 1.
```

5 Navržené simulační úlohy

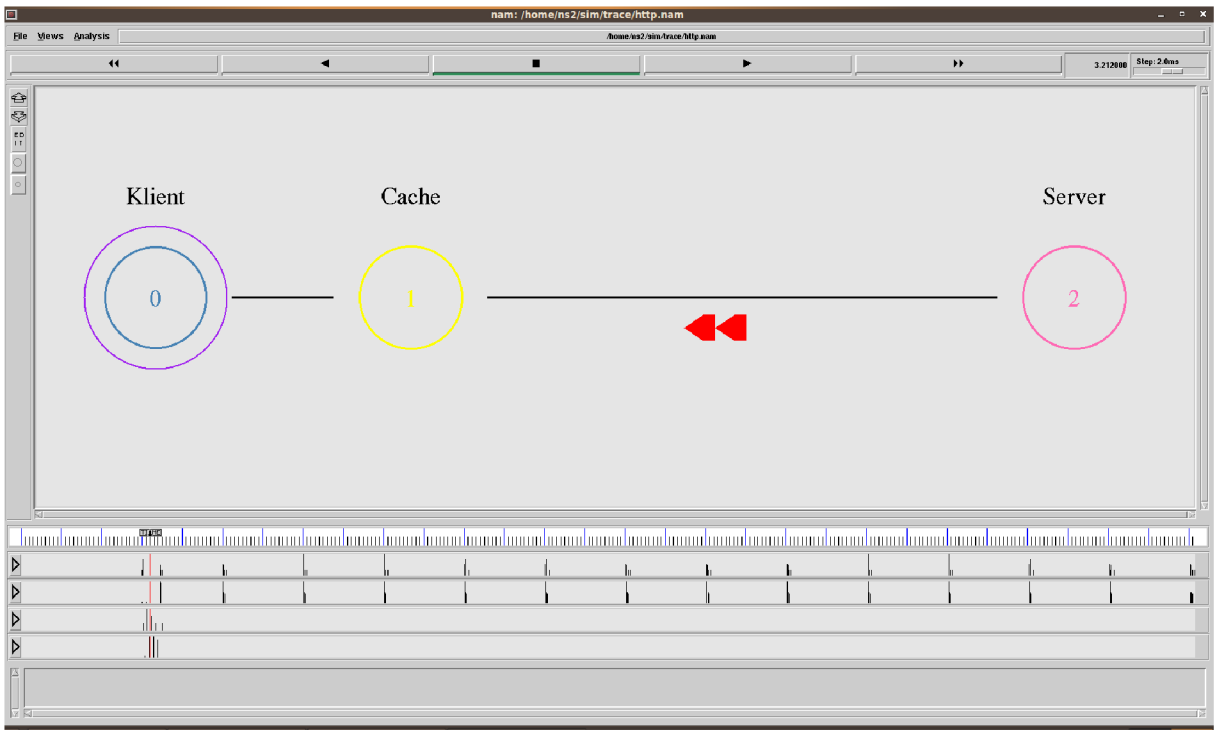
V této kapitole budou předvedeny koncepty simulačních úloh, které jsem v rámci bakalářské práce vytvořil. Nejsložitější a nejpropracovanější simulační úloha je pak v příloze B detailně zpracována formou návodu, který mohou studenti využít přímo ve výuce. Všechny programy budou popsány s pomocí obrázků, které zachycují okno aplikace NAM nebo XGRAPH. Kompletní zdrojové kódy napsané v jazyku Tcl jsou uvedeny v přílohách a bude na ně jednotlivě odkazováno. Všechny soubory potřebné ke spuštění jednotlivých simulací budou k dipozici na přiloženém CD.

V první úloze (PagePool/Math) si mohou studenti vyzkoušet nejjednodušší typ komunikace mezi klientem, cache a webovým serverem, kde dochází k posílání jednoduché webové stránky. Komunikace v druhé úloze (PagePool/CompMath) je o trochu složitější z důvodu zakomponování FTP serveru do simulace, který posílá data FTP klientovi, a tím je simulován datový provoz podobný provozu v reálných sítích. Webová stránka posílaná serverem se skládá z vložených objektů, což mohou být například obrázky. Výstupy těchto dvou úloh jsou grafy a logovací soubory, které slouží k lepšímu pochopení těchto úloh. Ve třetí úloze (PagePool/WebTraf) je zobrazena současná komunikace několika klientů s několika servery. V důsledku zapojení FTP serveru do komunikace dochází na jednom uzlu k zahazování paketů z fronty. Ve třech úlohách vytvořených za použití třídy PagePool/ProxyTrace, si studenti mohou velmi detailně nakonfigurovat celou komunikaci a následně výsledky ověřit z logovacího souboru nebo grafů. Tyto tři úlohy se od sebe liší počtem a rozestavením uzlů. K poslední z těchto úloh byl vypracován velmi podrobný návod.

5.1 Komunikace klienta se serverem přes cache (s PagePool/Math)

V prvním vytvořeném programu je simulována komunikace mezi webovým prohlížečem, což je klient, a webovým serverem. Komunikace jde přes paměť cache, kde se ukládají údaje o webových stránkách pro případ, že by tato data chtěl klient ještě do budoucna použít, tak aby nemusel posílat požadavek až na server. Na obr. 5.1 je zobrazena komunikace, kde klient požádá prostřednictvím cache webový server o webovou stránku a server ji začne po částech posílat do cache. Když je celá webová stránka uložená v cache, tak se odešle klientovi. Čtyři zobrazené grafy, které jsou součástí obr. 5.1, určují datové toky paketů na jednotlivých linkách. První a druhý graf (bráno od shora dolu) zobrazuje datovou komunikaci mezi klientem a cache a třetí a čtvrtý mezi serverem a cache. Z grafů je patrné, že poté co

server odešle webovou stránku do cache (zobrazují to červené pakety), tak už se dále komunikace neúčastní. Komunikace probíhá pouze mezi klientem a cache a to v okamžiku, kdy klient požádá o webovou stránku, která je v této simulaci pouze jedna a je uložená cache. Z prvního a druhého grafu je patrné, že klient posílá požadavek o tu samou stránku ve stejných časových intervalech. Mezi další výstupy tohoto programu patří logovací soubor, kde je zaznamenáno, jaké HTTP události v určitý čas nastaly. Úplný zdrojový kód tohoto programu je uveden v příloze A. 1.

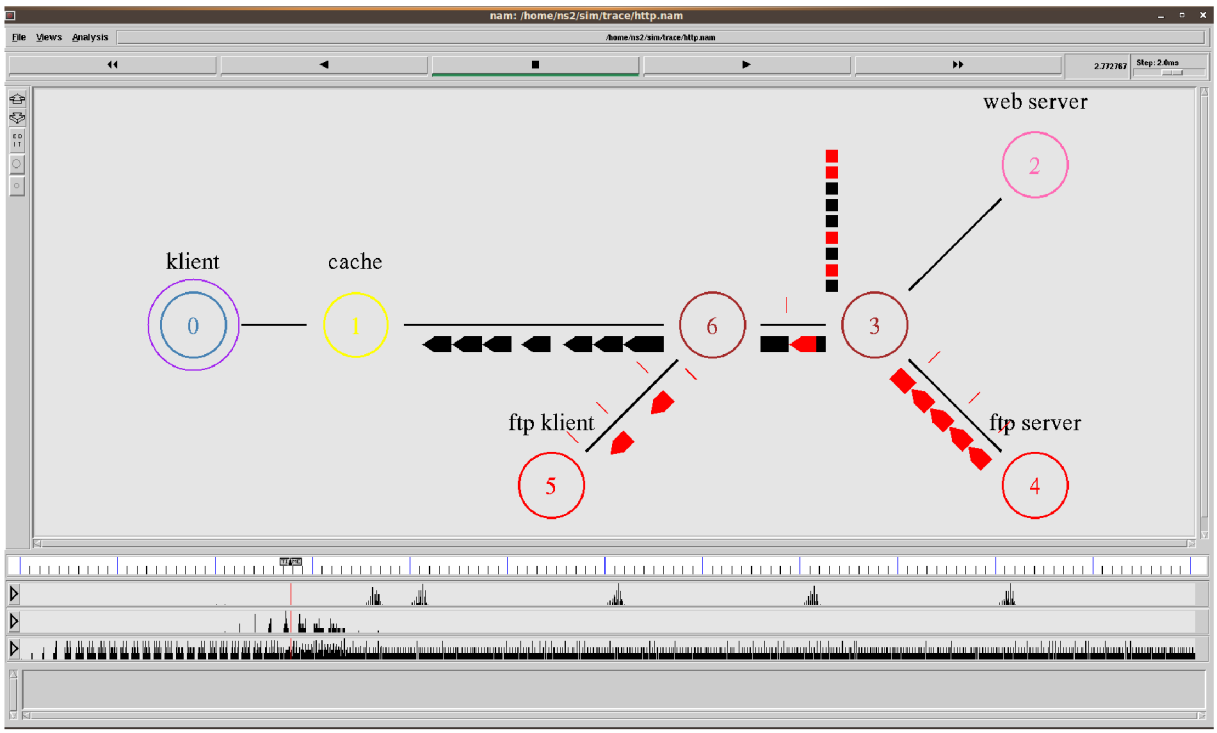


Obr. 5.1: Komunikace klienta se serverem přes cache (s PagePool/Math).

5.2 Komunikace klienta se serverem přes cache (s PagePool/CompMath)

V druhé navržené simulaci je zobrazena opět komunikace mezi klientem a webovým serverem za použití cache, ale je zde použit jiný generátor webových stránek a to PagePool/CompMath, který pracuje s objekty, které jsou součástí webové stránky. Součástí simulace, jak je patrné z obr. 5.2, je i FTP klient a FTP server. Tyto uzly jsou obsaženy v simulaci proto, aby byla simulována reálná situace v síti, kde běží více přenosů paralelně a to může mít za následek zpomalení přenosu z důvodů nárustu front v některých uzlech sítě. Komunikace mezi klientem a webovým serverem přes cache je obdobná, jako již byla popsána v úloze 5.1. Klient pošle požadavek webovému serveru o stránku, ten ji pošle

klientovi a stránka se při tom uloží v cache, odkud je v případě požadavku poslána klientovi. To je opět patrné z prvního grafu na obr. 5.2. Druhý graf zobrazuje komunikaci mezi web serverem a uzlem 3 a graf třetí mezi uzlem 3 a uzlem 6. Dalším výstupem kromě souboru typu *.nam je opět i logovací soubor. Úplný zdrojový kód tohoto programu je uveden v příloze A. 2.

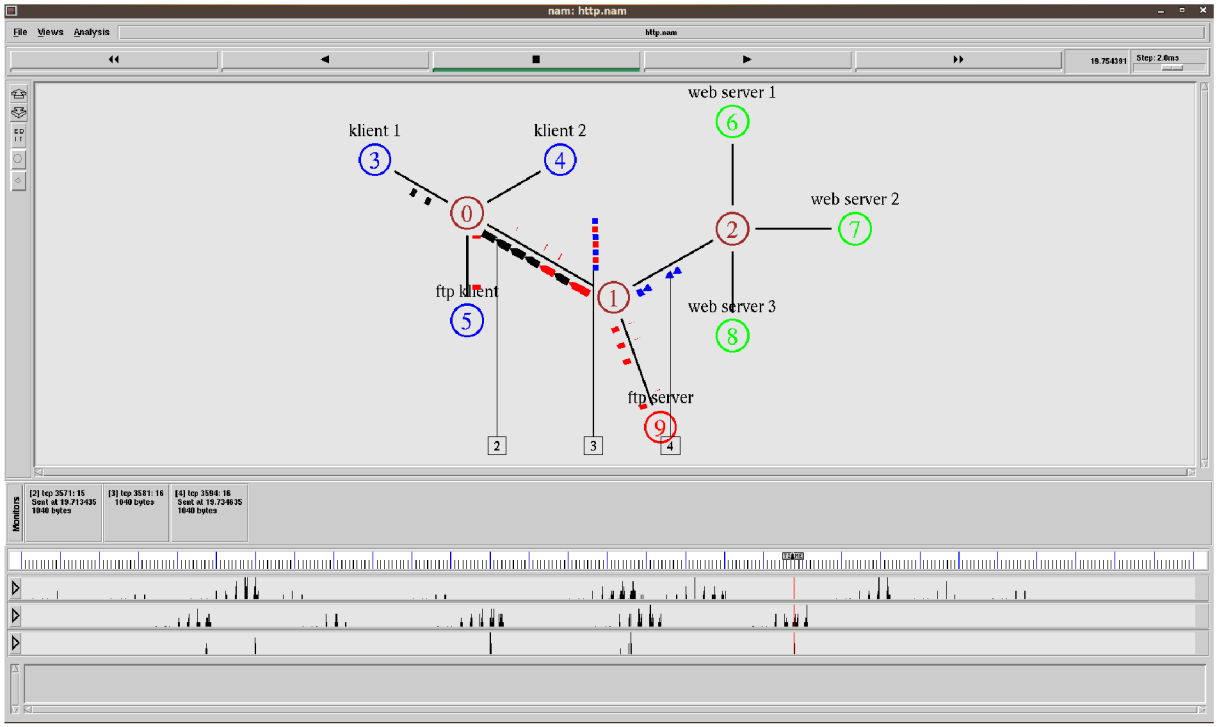


Obr. 5.2: Komunikace klienta se serverem přes cache (s PagePool/CompMath).

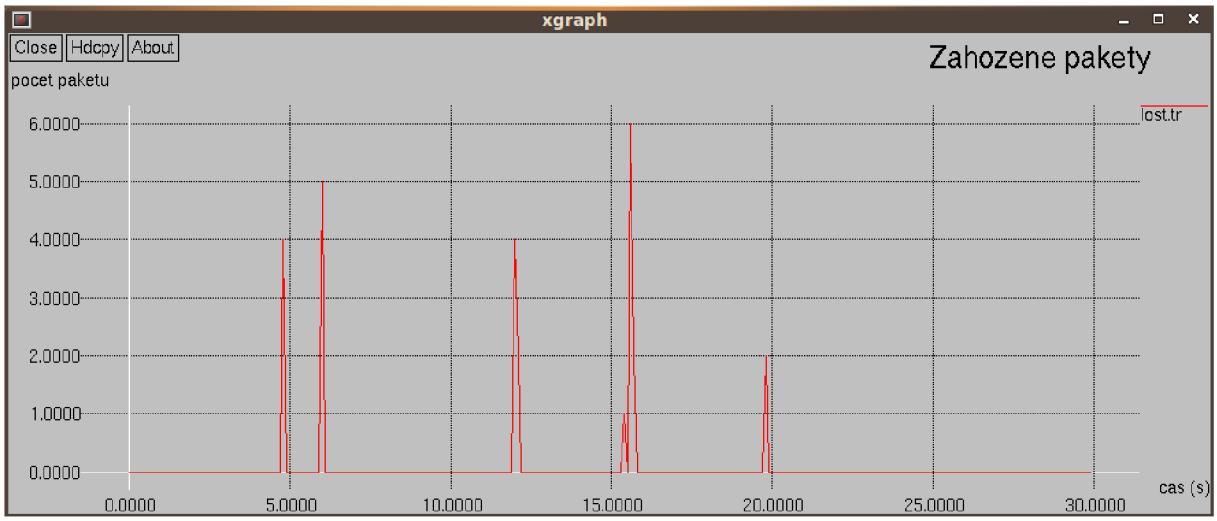
5.3 Komunikace skupiny klientů a serverů (s PagePool/WebTraf)

Tato úloha zobrazuje komunikaci mezi dvěma klienty a třemi webovými servery s využitím třídy PagePool/WebTraf. Součástí simulace je opět FTP server posílající data FTP klientovi z důvodu simulace reálné sítě, kde dochází ke zpomalení přenosu z důvodu tvoření front na uzlu číslo 3. Vytvořená topologie je patrná z obr. 5.3. Komunikace probíhá tak, že klient vyšle požadavek na server o poslání webové stránky a server stránku pošle klientovi. Každý klient při tom komunikuje postupně s různými servery nebo mohou i oba klienti komunikovat se stejným serverem ve stejný okamžik. Každá webová stránka se skládá z definovatelného počtu objektů, které jsou před odesláním rozkouskovány na pakety, které jsou značeny u každého objektu jinou barvou. Na obr. 5.3 jsou dole tři grafy, kde první graf zobrazuje komunikace na lince mezi uzlem 0 a klientem 1, druhý graf zobrazuje komunikaci

mezi uzlem 0 a klientem 2 a třetí graf zobrazuje zahozené pakety na lince mezi uzly 1 a 0. Jednotlivé pakety lze také monitorovat, jak zobrazují čísla 2, 3, 4 v čtvercovém rámečku. Údaje o jednotlivých paketech jsou pak zobrazeny v liště monitors, která se nachází nad lištou s časovým posuvníkem. Na obr. 5.4 je v aplikaci XGRAPH zobrazena grafická závislost paketů, zahozených na uzlu číslo 1, na čase. Důvodem zahazování paketů z fronty vytvořené na uzlu je překročení kapacity vyrovnávací paměti uzlu, která je v této simulaci nastavena na uložení deseti paketů. Z důvodů paralelní komunikace přes tento uzel dochází v určitých okamžicích k úplnému zaplnění této paměti a pakety, které se nevejdou do paměti, jsou zahozeny. Výstupními soubory této simulace jsou soubory typu *.tr a soubor typu *.nam. Úplný zdrojový kód tohoto programu je uveden v příloze A. 3.



Obr. 5.3: Komunikace skupiny klientů a serverů (s PagePool/WebTraf).

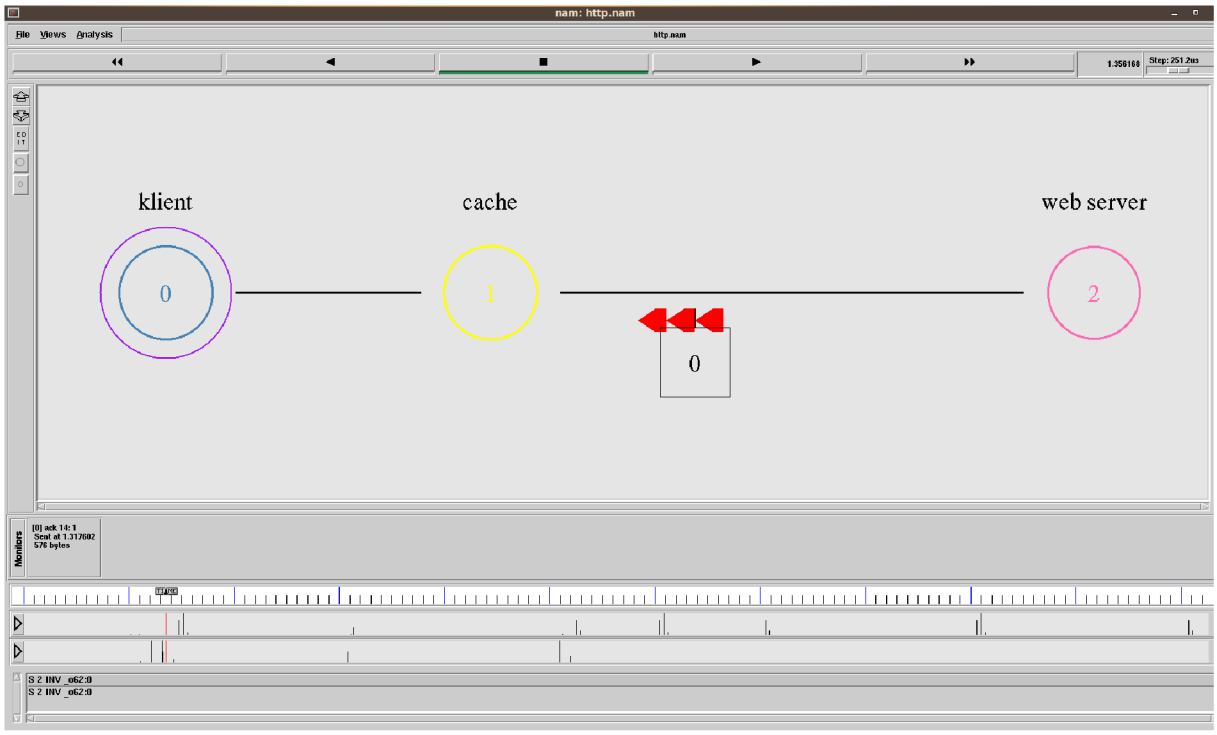


Obr. 5.4: Pakety zahozené na uzlu číslo 1 (s PagePool/WebTraf).

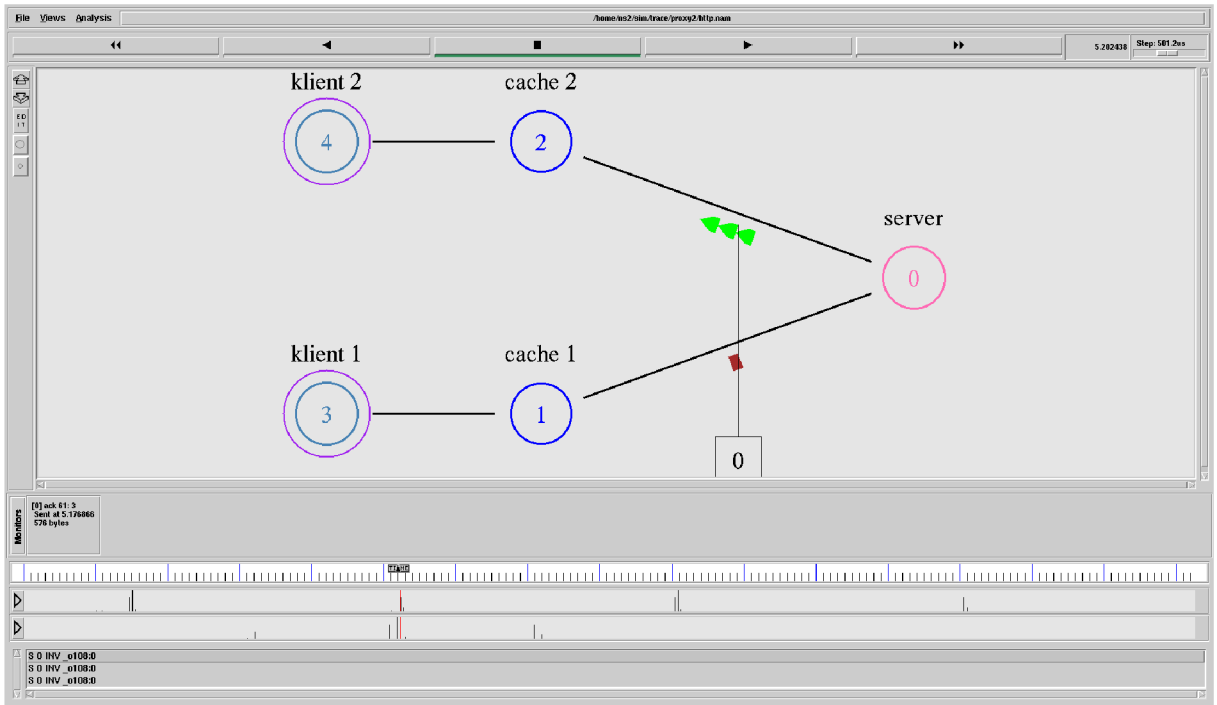
5.4 Komunikace klientů se serverem přes cache (s PagePool/ProxyTrace)

S využitím třídy PagePool/ProxyTrace jsem vytvořil tři simulace. První nejjednodušší simulace představuje komunikaci klienta s webovým serverem za použití paměti cache. Soubor pglog obsahuje tři nadefinované webové stránky a soubor reqlog obsahuje sedm aktivních požadavků o stáhnutí různých webových stránek ze serveru. Na malých grafech z obr. 5.5 je vidět celá komunikace. První graf od shora zobrazuje datové vytížení linky mezi cache a klientem. Zde je vidět sedm přenesených stránek ke klientovi. A druhý graf zobrazuje vytížení linky mezi serverem a cache. Z tohoto grafu je patrné, že ze serveru byly odeslány tři webové stránky. Podrobný souhrn všech HTTP událostí je obsažen v logovacím souboru, který bude spolu se zdrojovými kódy a vstupními soubory k dispozici na přiloženém CD.

Druhá složitější simulace se zabývá komunikací dvou klientů s jedním webovým serverem s tím, že každý klient používá vlastní cache pro ukládání webových stránek. Komunikace je opět řízena vstupními soubory pglog a reqlog. Z malých grafů na obr. 5.6 je patrný počet odeslaných stránek z cache 1 ke klientovi 1 (horní graf) a počet odeslaných stránek serverem do cache 2 (dolní graf). Všechny ostatní možné podrobnosti k těmto dvěma simulacím, jako například popis souborů pglog, reqlog, logovacího souboru, budou rozebrány v podrobném návodu pro poslední simulaci v příloze B.



Obr. 5.5: Komunikace klienta se serverem přes cache (s PagePool/ProxyTrace).



Obr. 5.6: Komunikace dvou klientů se serverem přes dvě cache (s PagePool/ProxyTrace).

Poslední úloha se zabývá komunikací dvou klientů s jedním serverem. Zde je jako úložiště webových stránek požita proxy cache, která propojuje oba klienty se serverem. Vše, co se týká této simulace, je podrobně vysvětleno v návodu v příloze B. Úplný zdrojový kód simulace je uveden v příloze A.4.

Závěr

V bakalářské práci jsem se v teoretické části zaměřil na popis, používání a instalaci programu Network Simulator verze 2 (NS2). Dále jsem uvedl základní syntaxe jazyka Tcl a nakonec jsem rozebral některé možnosti simulace protokolu HTTP s využitím OTcl tříd PagePool.

V praktické části bakalářské práce jsem se zaměřil na návrh simulačních úloh s protokolem HTTP pro předmět Pokročilé komunikační techniky. Vytvořil jsem šest úloh za použití čtyř typů tříd PagePool. V prvních dvou simulacích, kde první simulace využívá třídu PagePool/Math a druhá simulace třídu PagePool/CompMath, je zobrazena komunikace klienta s webovým serverem s využitím paměti cache, do které se ukládají webové stránky poslané serverem. Výstupním souborem těchto simulací je logovací soubor, který zaznamenává všechny HTTP události, které během simulací nastanou. Třetí simulační úloha, která využívá třídu PagePool/WebTraf, zobrazuje komunikaci mezi dvěma HTTP klienty a třemi webovými servery, kde každý klient posílá postupně požadavky o stáhnutí různých druhů webových stránek ze serverů. S využitím třídy PagePool/ProxyTrace jsem vytvořil tři úlohy, které využívají k řízení celé komunikace dva vstupní soubory. Úlohy se od sebe liší rozdílnou topologií sítě a průběhem komunikace. K poslední úloze, která simuluje komunikaci mezi dvěma klienty propojenými s jednou cache a dále webovým serverem, jsem vypracoval podrobný návod s vysvětlením obsahu logovacího souboru a vstupních souborů řídících komunikaci. Součástí některých již zmiňovaných úloh jsou grafy zobrazené aplikací XGRAPH.

Seznam použitých zdrojů

[1] The VINT Project: *The Network Simulator - ns-2* [online]. [cit. 25. 11. 2009]. Dostupný z URL: < http://www.isi.edu/nsnam/ns/>

[2] CHUNG, Jae, CLAYPOOL, Mark: *NS by example* [online]. [cit. 27. 11. 2009]. Dostupný z URL: <http://nile.wpi.edu/NS/>.

[3] Jeřábek, J.: Skripta k předmětu Pokročilé komunikační techniky, Vysoké učení technické v Brně, 230 s., Brno 2009.

[4] The VINT Project*: Network Simulator ns-2: Validation Tests* [online]. [cit. 27. 11. 2009]. Dostupný z URL: <http://www.isi.edu/nsnam/ns/ns-tests.html>

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

[6] The VINT Project*: Nam: Network Animator* [online]. 3. 7. 2002, [cit. 27. 11. 2009]. Dostupný z URL: < http://www.isi.edu/nsnam/nam/>

[7] TIŠNOVSKÝ, Pavel*: Programovací jazyk TCL* [online]. 2005, [cit. 30. 11. 2009]. Dostupný z URL: <http://www.root.cz/serialy/programovaci-jazyk-tcl/>

[8] *NOTEPAD++* [online]. 12.10.2009, [cit. 10. 12. 2009]. Dostupný z URL: <http://notepad-plus.sourceforge.net>

[9] *Editor PSPad - freeware editor* [online]. 12.7.2009, [cit. 10. 12. 2009]. Dostupný z URL: < http://www.pspad.com >

[10] FALL, Kevin, VARADHAN, Kannan: *The ns Manual* [online]. 2009, 429 s, [cit. 15. 2. 2010]. Dostupný z URL: <http://isi.edu/nsnam/ns/doc/ns_doc.pdf>.

Seznam použitých zkratek

ADU - application-level data unit

AODV - Ad hoc On-Demand Distance Vector

CBQ - Class Based Queuing

CBR - Constant Bit Rate

CONSER - Collaborative Simulation for Education and Research

CSMA/CD - Carrier Sense Multiple Access with Collision Detection

DARPA - Defense Advanced Research Projects Agency

DRR - Deficit Round Robin

DSR - Dynamic Source Routing Protocol

FIFO - First In First Out

FQ - Fair Queueing

FTP - File Transfer Protocol

HTTP - Hypertext Transfer Protocol

Lisp - List processing

MAC - Medium Access Control protocol

NAM - Network Animator

NS2 - Network Simulator 2

NS3 - Network Simulator 3

NSF - The National Science Foundation

OSI - Open Systems Interconnection

OTCL - Object Tool Command Language

PIM-SM - Protocol Independent Multicast – Sparse Mode

RED - Random Early Discard

RTP - Real-time Transport Protocol

SFQ - Stachastic Fair Queueing

TCL - Tool Command Language

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

VINT - Virtual InterNetwork Testbed

Seznam příloh

A Zdrojové kódy programů

A.1 Zdrojový kód programu, který využívá třídu PagePool/CompMath

```
#Schema topologie
#
# klient-----cache--------------server
#

#Vytvoreni instance simulatoru
set ns [new Simulator]

#Vytvoreni souboru pro naslednou simulaci v NAM
set myNAM [open http.nam w]
$ns namtrace-all $myNAM

# Vytvoreni trasovaciho souboru
set myTRACE [open http.tr w]
$ns trace-all $myTRACE

#Vytvoreni log souboru
set log [open "http.log" w]

#-----------------------------------------------------------------

#Vytvoreni topologie
set n0 [$ns node] ;#klient
set n1 [$ns node] ;#cache
set n2 [$ns node] ;#server


#Nastaveni parametru linek
$ns duplex-link $n2 $n1 1.5Mb 50ms DropTail
$ns duplex-link $n1 $n0 10Mb 10ms DropTail


#Nastaveni polohy jednotlivych uzlu v NAM
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right


#Nastaveni barev paketu
$ns color 0 green
$ns color 1 red


#Popisek uzlu v definovanem case
$ns at 0.0 "$n0 label \"Klient\""
$ns at 0.0 "$n1 label \"Cache\""
$ns at 0.0 "$n2 label \"Server\""
```

```
#Nastaveni barev popisku
$n0 label-color black
$n1 label-color black
$n2 label-color black

#-------------------------------------------------------------

#Vytvoreni promenne generatoru stranek PagePool/Math
set pool [new PagePool/Math]

#Vytvoreni nahodne promenne pro nasledne ulozeni hodnoty
set tmp [new RandomVariable/Constant]

#Prirazeni hodnoty k promenne
$tmp set val_ 2048

#Nastaveni velikosti stranky v bytech
$pool ranvar-size $tmp

#Vytvoreni nahodne promenne pro nasledne ulozeni hodnoty
set tmp [new RandomVariable/Constant]

#Prirazeni hodnoty k promenne
$tmp set val_ 5

#Nastaveni prumerneho casu zivotnosti stranky
$pool ranvar-age $tmp

#Vytvoreni HTTP serveru a prirazeni k uzlu
set server [new Http/Server $ns $n2]

#Pripoji generator stranek k HTTP serveru
$server set-page-generator $pool

#Zaznamena HTTP udalosti, ktere nastanou na serveru, do log souboru
$server log $log

#Vytvoreni cache a prirazeni k uzlu
set cache [new Http/Cache $ns $n1]

#Zaznamena HTTP udalosti, ktere nastanou v cache, do log souboru
$cache log $log

#Vytvoreni klienta a prirazeni k uzlu
set client [new Http/Client $ns $n0]

#Vytvoreni nahodne promenne pro nasledne ulozeni hodnoty
set tmp [new RandomVariable/Constant]

#Prirazeni hodnoty k promenne
$tmp set val_ 2

#Nastaveni prumerneho intervalu mezi pozadavky o stahnuti stranky
(v sekundach)
$client set-interval-generator $tmp
```

```
#Pripojeni generatoru stranek ke klientovi
$client set-page-generator $pool

#Zaznamena HTTP udalosti, ktere nastanou u klienta, do log souboru
$client log $log


#Vytvoreni funkce, jejiz prikazy spusti komunikaci
proc start {} {
        #Deklarovani pouzitych promennych
        global ns server cache client

        #Klient se spoji s cache
        $client connect $cache


        #Cache se spoji se serverem
        $cache connect $server

        #Klient vysle pozadavek o webovou stranku na serveru,
        kde komunikace jde od klienta pres cache
        $client start-session $cache $server
}


#Funkce ukonceni
proc finish {} {
        global ns log myNAM myTRACE
        $ns flush-trace
        flush $log
        close $log
        close $myNAM
        close $myTRACE
        puts "Vytvarim soubory: http.nam, http.tr, http.log"
        puts "Startuji NAM"
        exec nam http.nam &
        exit 0
}

#Cas spousteni jednotlivych uloh
$ns at 1.0 "start"
$ns at 30.0 "finish"
$ns run
```

A.2 Zdrojový kód programu, který využívá třídu PagePool/Math

```
#Schema topologie
#
#                            n2
#                           /
#                          /
# n0-----n1-------n6----n3
#                  /      \
#                 /        \
#              n5           n4

#Vytvoreni instance simulatoru
set ns [new Simulator]

#Vytvoreni souboru pro naslednou simulaci v NAM
set myNAM [open http.nam w]
$ns namtrace-all $myNAM

# Vytvoreni trasovaciho souboru
set myTRACE [open http.tr w]
$ns trace-all $myTRACE

#Vytvoreni log souboru
set log [open "http.log" w]

#-----------------------------------------------------------------

#Vytvoreni topologie
set n0 [$ns node] ;#klient
set n1 [$ns node] ;#cache
set n2 [$ns node] ;#server
set n3 [$ns node] ;#uzel
set n4 [$ns node] ;#ftp server
set n5 [$ns node] ;#ftp klient
set n6 [$ns node] ;#uzel


#Nastaveni parametru linek
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n6 1Mb 40ms DropTail
$ns duplex-link $n6 $n3 2Mb 10ms DropTail
$ns duplex-link $n3 $n2 2Mb 20ms DropTail
$ns duplex-link $n3 $n4 2Mb 20ms DropTail
$ns duplex-link $n6 $n5 2Mb 20ms DropTail

#Nastavi velikost bufferu pro linku mezi uzly n3 a n6
$ns queue-limit $n3 $n6 50


#Nastavi pozici pro zobrazeni fronty v NAM
$ns duplex-link-op $n3 $n6 queuePos 1.5
```

```
#Nastaveni barev paketu
$ns color 0 green
$ns color 1 black

#Nastavi barvu paketu pro FTP
$ns color 30 red


#Popisek uzlu v definovanem case
$ns at 0.0 "$n0 label \"klient\""
$ns at 0.0 "$n1 label \"cache\""
$ns at 0.0 "$n2 label \"web server\""
$ns at 0.0 "$n4 label \"ftp server\""
$ns at 0.0 "$n5 label \"ftp klient\""


#Nastaveni polohy jednotlivych uzlu v NAM
$ns duplex-link-op $n1 $n0 orient left
$ns duplex-link-op $n1 $n6 orient right
$ns duplex-link-op $n3 $n2 orient right-up
$ns duplex-link-op $n3 $n4 orient right-down
$ns duplex-link-op $n6 $n5 orient left-down
$ns duplex-link-op $n6 $n3 orient right


#Nastaveni barvy uzlu
$n3 color brown
$n4 color red
$n5 color red
$n6 color brown


#Nastaveni barev popisku
$n0 label-color black
$n1 label-color black
$n2 label-color black
$n4 label-color black
$n5 label-color black

#-------------------------------------------------------------

#Vytvoreni agenta TCP
set tcp [new Agent/TCP]

#Prirazeni agenta TCP k uzlu
$ns attach-agent $n4 $tcp

#Vytvoreni agenta TCPSink
set sink [new Agent/TCPSink]

#Prirazeni agenta TCPSink k uzlu
$ns attach-agent $n5 $sink

#Propojeni obou agentu
$ns connect $tcp $sink
```

```
#Prirazeni barvy paketu agentovi TCP (cervena)
$tcp set fid_ 30

#Vytvoreni aplikace FTP
set ftp [new Application/FTP]

#Aplikace FTP bude vyuzivat TCP spojeni
$ftp attach-agent $tcp

#---------------------------------------------------------------

#Vytvoreni promenne generatoru stranek PagePool/CompMath
set pool [new PagePool/CompMath]

#Nastavi velikost hlavni stranky v bytech
$pool set main_size_ 50000

#Nastavi velikost vlozenych objektu v bytech
$pool set comp_size_ 30000

#Nastavi pocet objektu, ktere bude hlavni stranka obsahovat
$pool set num_pages_ 10

#Vytvoreni nahodne promenne pro nasledne ulozeni hodnoty
set tmp [new RandomVariable/Constant]
#Prirazeni hodnoty k promenne
$tmp set val_ 100

#Nastavi zivotnost objektu (v sekundach)
$pool ranvar-obj-age $tmp

#Vytvoreni nahodne promenne pro nasledne ulozeni hodnoty
set tmp [new RandomVariable/Constant]

#Prirazeni hodnoty k promenne
$tmp set val_ 50

#Nastavi zivotnost stranky (v sekundach)
$pool ranvar-main-age $tmp

#Vytvoreni HTTP serveru a prirazeni k uzlu
set server [new Http/Server $ns $n2]

#Prirazeni generatoru stranek k serveru
$server set-page-generator $pool

#Zaznamena HTTP udalosti, ktere nastanou na serveru, do log souboru
$server log $log

#Vytvoreni cache a prirazeni k uzlu
set cache [new Http/Cache $ns $n1]

#Zaznamena HTTP udalosti, ktere nastanou v cache, do log souboru
$cache log $log
```

```
#Vytvoreni klienta a prirazeni k serveru
set client [new Http/Client $ns $n0]

#Vytvoreni nahodne promenne pro nasledne ulozeni hodnoty
set tmp [new RandomVariable/Constant]

#Prirazeni hodnoty k promenne
$tmp set val_ 2

#Priradi ke klientovi generator intervalu pozadavku o stranky
$client set-interval-generator $tmp

#Priradi klientovi generator stranek
$client set-page-generator $pool

#Zaznamena HTTP udalosti, ktere nastanou u klienta, do log souboru
$client log $log

#---------------------------------------------------------------

#Funkce startu
proc start {} {
        #Deklarovani pouzitych promennych
        global ns server client cache

        #Klient se spoji s cache
        $client connect $cache

        #Cache se spoji se serverem
        $cache connect $server

        #Klient vysle pozadavek o webovou stranku na serveru,
        kde komunikace jde od klienta pres cache
        $client start-session $cache $server
}

#Funkce ukonceni
proc finish {} {
        global ns log myNAM myTRACE
        $ns flush-trace
        flush $log
        close $log
        close $myNAM
        puts "Vytvarim soubory: http.nam, http.tr, http.log"
        puts "Startuji NAM"
        exec nam http.nam &
        exit 0
}

#Cas spousteni jednotlivych uloh
$ns at 0.0 "$ftp start"
$ns at 0.0 "start"
$ns at 12.0 "finish"
$ns run
```

A.3 Zdrojový kód programu, který využívá třídu PagePool/WebTraf

```
#Schema topologie
#
# n3     n4               n6
#   \    /                |
#    \  /                 |
#     n0-------n1--------n2----n7
#     |         \         |
#     |          \        |
#     n5         n9       n8

#Vytvoreni instance simulatoru
set ns [new Simulator]

#Vytvoreni souboru pro naslednou simulaci v NAM
set myNAM [open http.nam w]
$ns namtrace-all $myNAM

#Vytvoreni souboru grafu
set queue [open queue.tr w]
set bw [open bw.tr w]
set lost [open lost.tr w]
set bw1 [open bw1.tr w]
set bw2 [open bw2.tr w]

#---------------------------------------------------------------

#Vytvoreni topologie
set n0 [$ns node] ;#uzel 1
set n1 [$ns node] ;#uzel 2
set n2 [$ns node] ;#uzel 3
set n3 [$ns node] ;#klient 1
set n4 [$ns node] ;#klient 2
set n5 [$ns node] ;#FTP klient
set n6 [$ns node] ;#web server 1
set n7 [$ns node] ;#web server 2
set n8 [$ns node] ;#web server 3
set n9 [$ns node] ;#FTP server


#Nastaveni parametru linek
$ns duplex-link $n1 $n0 1.5Mb 40ms DropTail
$ns duplex-link $n1 $n2 3Mb 30ms DropTail
$ns duplex-link $n0 $n3 5Mb 20ms DropTail
$ns duplex-link $n0 $n4 5Mb 20ms DropTail
$ns duplex-link $n0 $n5 5Mb 20ms DropTail
$ns duplex-link $n1 $n9 5Mb 30ms DropTail
$ns duplex-link $n2 $n6 5Mb 20ms DropTail
$ns duplex-link $n2 $n7 5Mb 25ms DropTail
$ns duplex-link $n2 $n8 5Mb 20ms DropTail
```

```
#Nastaveni polohy jednotlivych uzlu v NAM
$ns duplex-link-op $n1 $n2 orient 30deg
$ns duplex-link-op $n1 $n0 orient 150deg
$ns duplex-link-op $n1 $n9 orient 290deg
$ns duplex-link-op $n0 $n3 orient 150deg
$ns duplex-link-op $n0 $n4 orient 30deg
$ns duplex-link-op $n0 $n5 orient 270deg
$ns duplex-link-op $n2 $n6 orient up
$ns duplex-link-op $n2 $n7 orient right
$ns duplex-link-op $n2 $n8 orient down


#Popisek uzlu v definovanem case
$ns at 0.0 "$n3 label \"klient 1\""
$ns at 0.0 "$n4 label \"klient 2\""
$ns at 0.0 "$n5 label \"ftp klient\""
$ns at 0.0 "$n6 label \"web server 1\""
$ns at 0.0 "$n7 label \"web server 2\""
$ns at 0.0 "$n8 label \"web server 3\""
$ns at 0.0 "$n9 label \"ftp server\""


#Nastaveni barev uzlu
$n0 color brown
$n1 color brown
$n2 color brown
$n3 color blue
$n4 color blue
$n5 color blue
$n6 color green
$n7 color green
$n8 color green
$n9 color red


#Nastaveni barev popisku
$n3 label-color black
$n4 label-color black
$n5 label-color black
$n6 label-color black
$n7 label-color black
$n8 label-color black
$n9 label-color black


#Nastaveni barev paketu
$ns color 1 black
$ns color 2 blue
$ns color 3 brown
$ns color 4 green
$ns color 5 gray
$ns color 6 yellow
$ns color 7 chocolate
$ns color 8 black
$ns color 9 blue
$ns color 10 brown
```

```
$ns color 11 green
$ns color 12 gray
$ns color 13 yellow
$ns color 14 chocolate
$ns color 15 black
$ns color 16 blue
$ns color 17 brown
$ns color 18 green
$ns color 19 gray
$ns color 20 yellow

#Nastavi barvu paketu pro FTP
$ns color 30 red


#Nastavi velikost bufferu pro linku mezi uzly n1 a n0
$ns queue-limit $n1 $n0 10


#Nastavi pozici pro zobrazeni fronty v NAM
$ns duplex-link-op $n1 $n0 queuePos 1.67

#-----------------------------------------------------------------
#Vytvoreni agenta TCP
set tcp [new Agent/TCP]

#Prirazeni agenta TCP k uzlu
$ns attach-agent $n9 $tcp

#Vytvoreni agenta TCPSink
set sink [new Agent/TCPSink]

#Prirazeni agenta TCPSink k uzlu
$ns attach-agent $n5 $sink

#Propojeni obou agentu
$ns connect $tcp $sink

#Prirazeni barvy paketu agentovi TCP (cervena)
$tcp set fid_ 30

#Vytvoreni aplikace FTP
set ftp [new Application/FTP]

#Aplikace FTP bude vyuzivat TCP spojeni
$ftp attach-agent $tcp

#Vytvoreni agenta UDP
set udp [new Agent/UDP]

#Prirazeni agenta UDP k uzlu
$ns attach-agent $n9 $udp

#Vytvoreni agenta Null
set null [new Agent/Null]
```

```
#Prirazeni agenta Null k uzlu
$ns attach-agent $n5 $null

#Vzajemne propojeni agentu
$ns connect $udp $null

#Prirazeni barvy paketu agentovi UDP (cervena)
$udp set fid_ 30

#Vytvoreni aplikace s konstantni bitovou rychlosti (CBR)
set cbr [new Application/Traffic/CBR]
#Nastavi se velikost paketu
$cbr set packetSize_ 1000

#Nastavi se rychlost odesilani paketu (v Mb/s)
$cbr set rate_ 1Mb

#Aplikace CBR bude vyuzivat k prenosu paketu UDP spojeni
$cbr attach-agent $udp

#-------------------------------------------------------------

#Definuje se promenna pro generator stranek PagePool/WebTraf
set pool [new PagePool/WebTraf]

#Nastaveni poctu klientu
$pool set-num-client 2

#Nastaveni poctu serveru
$pool set-num-server 3

#Priradi kliety k uzlum a nastavi jejich jednotliva ID
$pool set-client 0 $n3
$pool set-client 1 $n4

#Priradi servery k uzlum a nastavi jejich jednotliva ID
$pool set-server 0 $n6
$pool set-server 1 $n7
$pool set-server 2 $n8

#Nastavi pocet relaci
$pool set-num-session 2

#=============================================================

#Nastavi pocet stranek behem jedne relace
set numPage 3

#Nastavi se hodnota id relace
set id 0

#Nastavi se cas startu relace
set time 0.1
```

```
#Nastavi se casovy interval mezi dvemi strankami (v sekundach)
set interPage [new RandomVariable/Constant]
$interPage set val_ 2.5

#Nastavi se pocet objektu, ktere bude obsahovat jedna stranka
set pageSize [new RandomVariable/Constant]
$pageSize set val_ 3

#Nastavi se casovy interval mezi dvema po sobe jdoucimi objekty (v
sekundach)
set interObj [new RandomVariable/Constant]
$interObj set val_ 2

#Nastavi se velikost objektu
set objSize [new RandomVariable/ParetoII]
$objSize set avg_ 40
$objSize set shape_ 1.2

#set objSize [new RandomVariable/Constant]
#$objSize set val_ 20

#Vytvoreni relace
$pool create-session $id $numPage $time $interPage $pageSize
$interObj $objSize
#============================================================

#Nastavi pocet stranek behem jedne relace
set numPage 5

#Nastavi se hodnota id relace
set id 1

#Nastavi se cas startu relace
set time 3.25

#Nastavi se casovy interval mezi dvemi strankami (v sekundach)
set interPage [new RandomVariable/Constant]
$interPage set val_ 2

#Nastavi se pocet objektu, ktere bude obsahovat jedna stranka
set pageSize [new RandomVariable/Constant]
$pageSize set val_ 2


#Nastavi se casovy interval mezi dvema po sobe jdoucimi objekty
set interObj [new RandomVariable/Constant]
$interObj set val_ 0.01

#Nastavi se velikost objektu
set objSize [new RandomVariable/ParetoII]
$objSize set avg_ 40
$objSize set shape_ 1.2

#set objSize [new RandomVariable/Constant]
#$objSize set val_ 20
```

```
#Vytvoreni relace
$pool create-session $id $numPage $time $interPage $pageSize
$interObj $objSize

#Pri nastavene hodnote "1" se bere hodnota promene $interPage jako
casovy usek mezi poslednim prijatym
paketem prvni stranky a prvni prijatym paketem nasledujici stranky
#Pri nastavene hodnote "0" se bere hodnota promene $interPage jako
casovy usek mezi prvnim prijatym
paketem prvni stranky a prvnim prijatym paketem nasledujici stranky
$pool set-interPageOption 1

#-------------------------------------------------------------

#Nastavi promenne pro monitorovani front mezi definovanymi uzly
set qmon [$ns monitor-queue $n1 $n0 ""];
set qmon1 [$ns monitor-queue $n0 $n3 ""];
set qmon2 [$ns monitor-queue $n0 $n4 ""];

#Funkce pro vytvoreni grafu
proc record {} {
          #Deklarovani pouzitych promennych
          global ns qmon qmon1 qmon2 bw bw1 bw2 lost queue

          #Vytvoreni instance simulatoru
          set ns [Simulator instance]

          #Nastavi cas, ktery urcuje v jakych intervalech se
          budou delat zaznamy
          set time 0.05

          #Nastavi se promenna, ktera ma hodnotu casu
          startu funkce record (aktualni cas simulace)
          set now [$ns now]

          #Promenna urcuje velikost fronty v bytech
          set size [$qmon set size_]

          #Promenna urcuje velikost fronty v paketech
          set pkts [$qmon set pkts_]

          #Promenna urcuje pocet paketu, ktere dorazily do fronty
          set parrivals [$qmon set parrivals_]

          #Promenna urcuje pocet bytu, ktere dorazily do fronty
          set barrivals [$qmon set barrivals_]

          #Promenna urcuje pocet paketu, ktere byly zahozeny
          set pdrops [$qmon set pdrops_]

          #Promenna urcuje pocet bytu, ktere byly zahozeny
          set bdrops [$qmon set bdrops_]

          #Promenna urcuje pocet bytu, ktere odesly z fronty
          set bdepartures [$qmon set bdepartures_]
```

```
            #Promenna urcuje pocet paketu, ktere odesly z fronty
            set pdepartures [$qmon set pdepartures_]

            #Promenna urcuje pocet bytu, ktere odesly z fronty
            (pro monitorovani dalsi urcene fronty)
            set bdepartures1 [$qmon1 set bdepartures_]

            #Promenna urcuje pocet bytu, ktere odesly z fronty
            (pro monitorovani dalsi urcene fronty)
            set bdepartures2 [$qmon2 set bdepartures_]



            #Zapisovani dat do souboru pro XGRAPH
            #Na ose x bude cas a na ose y bude pocet paketu ve fronte
            puts $queue "$now $pkts"

            #Vypocet vyuziti sirky pasma v Mb/s
            (pro linku mezi n1 a n0)
            puts $bw "$now [expr $bdepartures/$time*8/1000000]"

            #Vynuluje promennou
            $qmon set bdepartures_ 0

            #Celkovy soucet zahozenych paketu
            puts $lost "$now $pdrops"

            #Vynuluje promennou
            $qmon set pdrops_ 0

            #Vypocet vyuziti sirky pasma v Kb/s
            (pro linku mezi n0 a n3)
            puts $bw1 "$now [expr $bdepartures1/$time*8/1000]"

            #Vynuluje promennou
            $qmon1 set bdepartures_ 0

            #Vypocet vyuziti sirky pasma v Kb/s
            (pro linku mezi n0 a n4)
            puts $bw2 "$now [expr $bdepartures2/$time*8/1000]"

            #Vynuluje promennou
            $qmon2 set bdepartures_ 0

            #Pripocte k aktualnimu casu dalsi casovy krok a znovu
            se zavola funkce "record"
            $ns at [expr $now+$time] "record"
}

#---------------------------------------------------------------
```

```
#Funkce ukonceni
proc finish {} {
        global ns myNAM queue bw bw1 bw2 lost
        $ns flush-trace
        close $myNAM
        close $queue
        close $bw
        close $bw1
        close $bw2
        close $lost
        puts "Vytvarim soubory: http.nam, queue.tr, bw.tr,
        bw1.tr, bw2.tr, lost.tr"

        puts "Startuji NAM a XGRAPH"

        exec xgraph queue.tr -x "cas (s)" -y "pocet paketu"
        -t "Velikost fronty" -geometry 1000x400 &

        exec xgraph bw.tr -x "cas (s)" -y "Mb/s" -t "Vyuziti
        sirky pasma mezi uzly n1 a n0" -geometry 1000x400 &

        exec xgraph lost.tr -x "cas (s)" -y "pocet paketu"
        -t "Zahozene pakety" -geometry 1000x400 &

        exec xgraph bw1.tr -x "cas (s)" -y "Kb/s" -t "Vyuziti
        sirky pasma mezi uzly n0 a n3" -geometry 1000x400 &

        exec xgraph bw2.tr -x "cas (s)" -y "Kb/s" -t "Vyuziti
        sirky pasma mezi uzly n0 a n4" -geometry 1000x400 &

        exec nam http.nam &
        exit 0
}

#Cas spousteni jednotlivych uloh
$ns at 0.0 "record"
$ns at 0.0 "$ftp start"
#$ns at 0.0 "$cbr start"
$ns at 30.0 "$ftp stop"
#$ns at 30.0 "$cbr stop"
$ns at 30.0 "finish"
$ns run
```

A.4 Zdrojový kód programu, který využívá třídu PagePool/ProxyTrace

```
#Schema topologie
#
#  n0
#    \
#     \
#     n2-------------n3
#     /
#    /
#  n1

#Vytvoreni instance simulatoru
set ns [new Simulator]

#Vytvoreni souboru pro naslednou simulaci v NAM
set myNAM [open http.nam w]
$ns namtrace-all $myNAM

#Vytvoreni souboru grafu
set gr1 [open gr1.tr w]
set gr2 [open gr2.tr w]

#Vytvoreni log souboru
set log [open "http.log" w]

#-------------------------------------------------------------

#Vytvoreni topologie
set n0 [$ns node] ;#klient 0
set n1 [$ns node] ;#klient 1
set n2 [$ns node] ;#cache
set n3 [$ns node] ;#server

#Nastaveni parametru linek
$ns duplex-link $n2 $n0 10Mb 20ms DropTail
$ns duplex-link $n2 $n1 10Mb 20ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 50ms DropTail

#Popisek uzlu v definovanem case
$ns at 0.0 "$n0 label \"klient 0\""
$ns at 0.0 "$n1 label \"klient 1\""
$ns at 0.0 "$n2 label \"proxy cache\""
$ns at 0.0 "$n3 label \"web server\""

#Nastaveni polohy jednotlivych uzlu v NAM
$ns duplex-link-op $n2 $n0 orient 145deg
$ns duplex-link-op $n2 $n1 orient 215deg
$ns duplex-link-op $n2 $n3 orient 0deg

#Nastavení barev paketů
$ns color 0 blue
$ns color 1 red
$ns color 2 brown
```

```
#Nastaveni barev popisku
$n0 label-color black
$n1 label-color black
$n2 label-color black
$n3 label-color black
#---------------------------------------------------------------

#Vytvoreni promenne generatoru stranek PagePool/ProxyTrace
set pool [new PagePool/ProxyTrace]

#Pripojeni vstupniho souboru, ktery definuje, jak bude probihat
komunikace. Soubor "reqlog" je využíván HTTP klientem
$pool set-reqfile "reqlog"

#Pripojeni vstupniho souboru, ktery obsahuje informace o HTTP
strankach. Soubor "pglog" je využíván HTTP serverem
$pool set-pagefile "pglog"

#Nastaveni poctu HTTP klientu
$pool set-client-num 2

#Nastaveni hodnoty dvourezimoveho modelu (v rozmezi 0 - 1).
Hodnota (po vynasobeni 100) urcuje kolik procent webovych stranek
bude dynamickych
$pool bimodal-ratio 0.1

#Vytvoreni nahodne promenne pro nasledne ulozeni hodnoty
set tmp [new RandomVariable/Constant]

#Přiřazení hodnoty k proměnné
$tmp set val_ 5

#Nastaveni generatoru modifikacnich intervalu pro dynamicke stranky
$pool ranvar-dp $tmp

#Vytvoreni nahodne promenne pro nasledne ulozeni hodnoty
set tmp [new RandomVariable/Constant]

#Přiřazení hodnoty k proměnné
$tmp set val_ 10000

#Nastaveni generatoru modifikacnich intervalu pro staticke stranky
$pool ranvar-sp $tmp

#Vytvoreni HTTP serveru a prirazeni k uzlu
set server [new Http/Server $ns $n3]

#Pripojeni generatoru stranek k HTTP serveru
$server set-page-generator $pool

#Zaznamena HTTP udalosti, ktere nastanou na serveru, do log souboru
$server log $log

#Vytvoreni cache a prirazeni k uzlu
set cache [new Http/Cache $ns $n2]
```

```
#Zaznamena HTTP udalosti, ktere nastanou v cache, do log souboru
$cache log $log

#Vytvoreni klientu a prirazeni k uzlum
set client0 [new Http/Client $ns $n0]
set client1 [new Http/Client $ns $n1]

#Pripojeni generatoru stranek ke klientum
$client0 set-page-generator $pool
$client1 set-page-generator $pool

#Zaznamena HTTP udalosti, ktere nastanou u klientu, do log souboru
$client0 log $log
$client1 log $log

#-------------------------------------------------------------

#Vytvoření funkce, jejiz prikazy spusti komunikaci
proc start {} {
        #Deklarovani pouzitych promennych
        global ns server cache client0 client1

        #Klienti se spoji s cache
        $client0 connect $cache
        $client1 connect $cache

        #Cache se spoji se serverem
        $cache connect $server

        #Klienti vyslou pozadavky o poslani webovych stranek
        ze serveru. Komunikace jde od klientu pres cache do
        serveru
        $client0 start-session $cache $server
        $client1 start-session $cache $server
}

#-----------------------------------------------------------------

#Vytvoreni promene pro monitorovani dat mezi definovanymi uzly
(mezi serverem a cache)
set qmon [$ns monitor-queue $n3 $n2 ""];

#Vytvoreni funkce pro tvorbu grafu
proc record {} {
        #Deklarovani pouzitych promennych
        global ns qmon gr1 gr2

        #Nastavi cas, ktery urcuje v jakych intervalech se budou
        delat zaznamy
        set time 0.005

        #Nastavi se promenna, ktera ma hodnotu casu startu funkce
        record (aktualni cas simulace)
        set now [$ns now]
```

```
            #Vytvoreni promene, ktera urcuje kolik bytu odeslo z uzlu
            $n3 (serveru) do uzlu $n2 (cache)
            set bdepartures [$qmon set bdepartures_]

            #Vytvoreni promene, ktera urcuje kolik paketu odeslo
            z uzlu $n3 (serveru) do uzlu $n2 (cache)
            set pdepartures [$qmon set pdepartures_]

            #Zapsani hodnot do souboru grafu. Na ose x jsou hodnoty
            casu a na ose y je pocet odeslanych bytu
            puts $gr1 "$now $bdepartures"

            #Zapsani hodnot do souboru grafu. Na ose x jsou hodnoty
            casu a na ose y je pocet odeslanych paketu
            puts $gr2 "$now $pdepartures"

            #Pripocte k aktualnimu casu dalsi casovy krok a znovu se
            zavola funkce "record"
            $ns at [expr $now+$time] "record"
}

#Funkce ukončení
proc finish {} {
            #Deklarovani pouzitych promennych
            global ns log myNAM gr1 gr2

            #Ukonceni zaznamenavani do logovaciho souboru
            flush $log

            #Zavreni vystupnich souboru
            close $log
            close $gr1
            close $gr2
            close $myNAM

            #Spusteni souboru http.nam v programu NAM
            exec nam http.nam &

            #Spusteni souboru grafu aplikaci XGRAPH s definovanymi
            parametry
            exec xgraph gr1.tr -x "cas (s)" -y "pocet bytu"
            -t "Mnozstvi prenesenych dat ze serveru do cache"
            -geometry 1000x400 &

            exec xgraph gr2.tr -x "cas (s)" -y "pocet paketu"
            -t "Pocet paketu odeslanych serverem do cache"
            -geometry 1000x400 &

            exit 0
}

#Planovac uloh
$ns at 0.0 "start"
$ns at 0.0 "record"
$ns at 13.0 "finish"
$ns run
```

B Vypracovaný návod

Úvod:

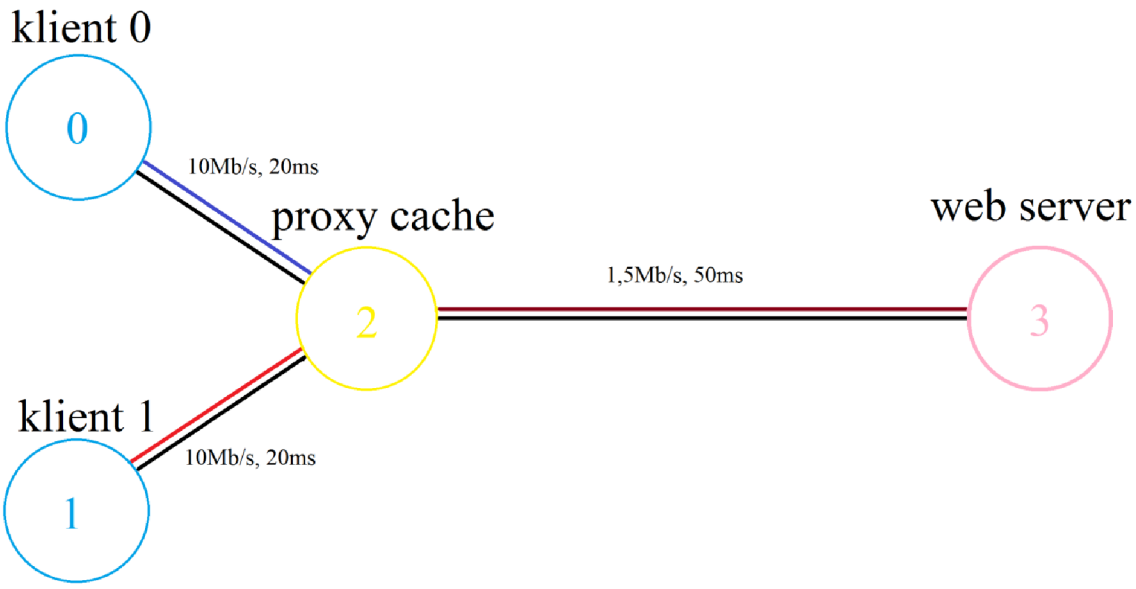
Simulační úloha je zaměřena na vytvoření komunikace mezi webovým serverem a dvěma webovými klienty, kteří se serverem komunikují prostřednictvím proxy cache serveru, který ukládá webové stránky, které si klienti stáhnou z webového serveru. Proxy cache server tedy posílá klientům webové stránky, o které žádali webový server. Pokud je webová stránka už jednou uložena v cache, tak při požadavku některého z klientů o stažení této stránky probíhá komunikace už jen mezi klientem a cache. V případě, že vyprší platnost webové stránky v cache, tak ji server musí nahradit aktuálnější verzí. Platnosti stránek mohou být různé. Jedna stránka je aktuální třeba jen několik minut, zato jiná stránka může být aktuální i měsíce. Mezi hlavní výhody proxy cache patří: snížení objemu přenesených dat, zkrácení doby přenosu stránek a snížení zátěže WWW serveru.

K simulaci této úlohy budeme používat program Network Simulator verze 2 (NS2). NS2 je objektově orientovaný volně šiřitelný simulační nástroj řízený diskrétními událostmi, který se používá k simulaci síťového provozu. Mezi hlavní funkce NS2 patří: budování modelu sítě, vytváření spojení mezi uzly, generování síťového provozu, modelování poruch a monitorování síťové komunikace. O grafické zobrazení výstupních souborů simulace se starají programy NAM a XGRAPH.

Teoretický rozbor:

Na začátku si vytvoříme soubor typu *.tcl, který se může otevřít v libovolném textovém editoru. Pro lepší orientaci v kódu je vhodné použít volně dostupné editory jako Notepad++, PSPad nebo gedit. Pro vytvoření instance simulátoru se používá příkaz **set ns [new Simulator]**. Vytvoření a spuštění výstupního souboru pro následné spuštění v programu NAM se provede příkazem **set myNAM [open http.nam w]**. Soubor uložený na disku se tedy jmenuje http.nam. Zaznamenání všech událostí, které během simulace nastanou, se do tohoto trasovacího souboru provede příkazem **$ns namtrace-all $myNAM**. Dalším výstupním souborem, který se musí vytvořit, je soubor pro logování HTTP událostí. Vytvoří se příkazem **set log [open "http.log" w]**.

Dále se musí nadefinovat uzly, které se budou v simulaci vyskytovat. To se provádí příkazem **set n0 [$ns node]**, kde proměnná **$n0** je název uzlu. Schéma simulace je patrné z obr. 1. Nastavení parametrů linky mezi cache a serverem se provádí zadáním příkazu **$ns duplex-link $n2 $n3 1.5Mb 50ms DropTail**, kde parametr **duplex-link** určuje použití plně duplexní linky mezi uzly **$n2** a **$n3** s šířkou pásma 1,5Mbit/s a zpožděním 50 ms, které je způsobeno především délkou linky. Parametr DropTail určuje typ fronty na lince a odpovídá frontě typu FIFO. Parametry mezi klienty a cache jsou: šířka pásma 10Mb a zpoždění 20ms. Pro přehledné uspořádání uzlů a linek se využívá příkaz **$ns duplex-link-op $n2 $n3 orient 0deg**, který nastaví umístění jednotlivých uzlů a to tak, že uzel **$n2** bude od uzlu **$n3** vlevo. Místo zadávání polohy ve stupních lze použít jako parametr **left**. Další možnosti parametrů jsou: **right, down, up** nebo jejich kombinace: například **right-up**.



Obr. 1: Schéma simulace.

Kvůli lepší orientaci ve výsledné simulaci je vhodné si k jednotlivým uzlům nastavit popisky. To se provede příkazem **$ns at 0.0 "$n0 label \"klient 0\""**, kde **$ns at 0.0** určuje čas v sekundách, kdy se má daný popisek zobrazit a **$n0** specifikuje uzel, u kterého se popisek **„klient 0"** zobrazí. Barva popisku u definovaného uzlu se nastaví příkazem **$n0 label-color black.** Barvy, které se mohou použít, jsou: **black, red, green, blue, qray, yellow, chocolate, brown.** Když simulace obsahuje více zdrojů, které vysílají pakety, tak je vhodné si tyto jednotlivé druhy paketů barevně odlišit. Použije se příkaz **$ns color 0 blue**, který určuje barvu prvního druhu vysílaných paketů. Následujícím druhům paketů se nastaví jiná barva

zvyšováním pořadového čísla za parametrem **color** a zvolením barvy pro toto pořadové číslo. V naší simulaci bude vhodné barevně oddělit pakety jdoucí mezi jednotlivými klienty a cache a pakety jdoucí mezi cache a serverem.

Komunikace mezi serverem cache a klienty bude definována za použití třídy PagePool/ProxyTrace. Tato třída používá pro nadefinování celé komunikace dva vstupní soubory, které jsou uloženy ve stejné složce jako hlavní soubor http_proxytrace.tcl a upravují se pomocí textového editoru. První vstupní soubor nese název „reqlog" a je využíván HTTP klientem. Soubor obsahuje data, která definují, jak bude komunikace probíhat. Druhý soubor se nazývá „pglog" a je využíván serverem. Obsahuje informace o webových stránkách. Oba vstupní soubory budou podrobně rozepsány dále.

Vytvoření proměnné **$pool** generátoru stránek PagePool/ProxyTrace se provede příkazem **set pool [new PagePool/ProxyTrace]**. Následuje připojení již zmiňovaných vstupních souborů příkazy **$pool set-reqfile "reqlog"** a **$pool set-pagefile "pglog"**. Dále se nastaví počet HTTP klientů v simulaci příkazem **$pool set-client-num 2**. Webové stránky vyskytující se v simulaci se dají rozdělit na dva typy: na stránky dynamické a stránky statické. Dynamické stránky se modifikují (aktualizují) často a statické stránky se nemodifikují. Nejdříve je nutné zvolit, kolik procent z celkového počtu webových stránek budou stránky dynamické. Nastavení se provede příkazem **$pool bimodal-ratio 0.1**, kde hodnota 0.1 značí, že 10 procent stránek bude dynamických a tím pádem 90 procent stránek statických. Zvolené číslo může nabývat pouze hodnoty v násobcích 10-ti procent (10%, 20%, 30%,…).

Nyní se musí nastavit generátory modifikačních intervalů webových stránek. Nastavení dynamického generátoru modifikačních intervalů začíná definováním náhodné proměnné. Vytvoří se příkazem **set tmp [new RandomVariable/Constant]**. Dále se k vytvořené náhodné proměnné přiřadí hodnota, která určuje časový interval modifikace dynamických stránek (v sekundách) a to příkazem **$tmp set val_ 5**. A nakonec se přiřadí tato hodnota pomocí náhodné proměnné **$tmp** ke generátoru modifikačních intervalů pro dynamické stránky a to příkazem **$pool ranvar-dp $tmp**. Nastavení statického generátoru modifikačních intervalů stránek se provádí obdobně jako u dynamického generátoru. Znovu se vytvoří náhodná proměnná konstantního typu, ale k proměnné se přiřadí hodnota například 10000 sekund a to znamená, že interval mezi modifikacemi stránky bude větší, než je celkový čas simulace, takže modifikace u statické stránky neproběhne. Přiřazení náhodné proměnné ke statickému generátoru modifikačních intervalů se provede příkazem **$pool ranvar-sp $tmp**.

Nyní je třeba vytvořit HTTP server a přiřadit ho k uzlu. Použije se příkaz **set server [new Http/Server $ns $n3]**. Připojení generátoru webových stránek k serveru se provede příkazem **$server set-page-generator $pool**. Ještě se musí u klientů, cache a serveru provést nastavení, aby se veškerá komunikace protokolu HTTP, která během simulace proběhne, zaznamenala do souboru typu *.log. U serveru se to provede příkazem **$server log $log**. Informace obsažené v logovacím souboru budou podrobně rozebrány dále.

Dalším prvkem, který se vytvoří a přiřadí k uzlu, je cache. Použije se příkaz **set cache [new Http/Cache $ns $n2]**. Opět se musí nastavit zaznamenávání do logovacího souboru obdobným způsobem, jako tomu bylo u serveru (**$cache log $log**).

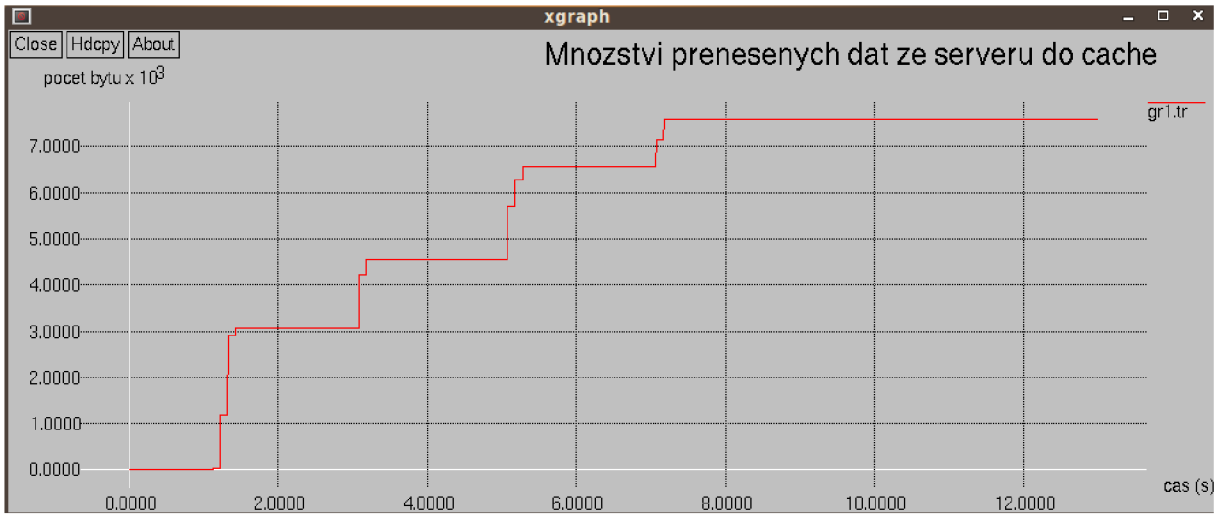
Nyní se musejí vytvořit dva klienti. Vytvoření a přiřazení k uzlu prvního klienta se provede příkazem **set client0 [new Http/Client $ns $n0]**. Stejným způsobem se zpracuje i druhý klient, který se přiřadí k uzlu **$n1**. Připojení generátoru webových stránek se k prvnímu klientovi provede příkazem **$client0 set-page-generator $pool**. Obdobně se zapíše příkaz i pro druhého klienta. K oběma klientům se ještě nastaví zaznamenávání HTTP komunikace do logovacího souboru, který je společný pro všechny uzly.

Na závěr zdrojového kódu se musí vytvořit tři funkce. Funkce startu, funkce pro vytváření grafů a funkce ukončení. Funkce startu se vytvoří příkazem **proc start {} {}**, kde se ve druhých složených závorkách zapíše vlastní kód funkce. Deklarování všech použitých proměnných funkce se zapíše příkazem **global ns server cache client0 client1**. Nyní se musí klienti spojit s cache pomocí příkazu **$client0 connect $cache** a následně se cache spojí se serverem příkazem **$cache connect $server**. Start relace, kdy klienti začnou vysílat požadavky o poslání webových stránek ze serveru, se provede příkazem **$client0 start-session $cache $server** pro prvního klienta a obdobným příkazem i pro druhého klienta.

Ještě před vytvořením funkce „record" je nutné vytvořit proměnnou pro monitorování dat a definovat uzly, mezi kterými monitorování bude probíhat. Použije se příkaz **set qmon [$ns monitor-queue $n3 $n2 ""]**. Deklarují se použité proměnné **ns**, **qmon**, **gr1**, **gr2**. Dále se provede příkazem **set time 0.005** nastavení hodnoty proměnné **$time**, která určuje, v jakých intervalech se budou dělat záznamy hodnot. Dále je nutné vytvořit proměnnou, která nese aktuální hodnotu času startu funkce record. Použije se příkaz **set now [$ns now]**. Proměná, která bude sčítat byty jdoucí ze serveru do cache, se vytvoří příkazem

set bdepartures [$qmon set bdepartures_]. Nyní se hodnoty času a počtu bytů zapíší do výstupního souboru příkazem **puts $gr1 "$now $bdepartures"**. Ve výsledném grafu se na ose y nachází počet bytů a na ose x je čas. Druhý soubor grafu zobrazuje počet paketů poslaných serverem do cache v čase, a proměnná sčítající pakety se vytvoří příkazem **set pdepartures [$qmon set pdepartures_]**. Soubor druhého grafu se vytvoří příkazem **puts $gr2 "$now $pdepartures"**. Aby se do výstupních souborů zapsaly všechny hodnoty, musí se funkce record spuštět znovu v časových intervalech, které jsou definovány proměnnou **$time**. Použije se příkaz **$ns at [expr $now+$time] "record"**.

Funkce ukončení se vytvoří příkazem **proc finish {} {}**. Opět se deklarují použité proměnné příkazem **global ns log myNAM**. Příkaz close zavře otevřené výstupní soubory, do kterých se zapisovaly informace (**close $log**). Spuštění souboru programem NAM se provede příkazem **exec nam http.nam**. Úplně na konec zdrojového kódu se musí naplánovat spuštění jednotlivých funkcí. Funkce startu se spustí v čase 0.0 sekund od začátku simulace (takže vlastně v začátku) a je definována příkazem **$ns at 0.0 "start"**. Funkce ukončení proběhne v čase 20 sekund a použije se příkaz **$ns at 20.0 "finish"**. Na obr. 2 je znázorněn přenos webových stránek ze serveru do cache. Každý schod v grafu znázorňuje jedno webovou stránku poslanou do cache. Z grafu se dá určit i přesná velikost jednotlivých stránek, která odpovídá po odečtení velikosti hlavičky každého paketu, která je 40 bytů, velikosti uvedené v logovacím souboru.



Obr. 2: Grafické zobrazení množství přenesených dat ze serveru do cache v čase.

Popis vstupních souborů:

Vstupní soubory „pglog" a „reqlog", pomocí kterých se nastaví celá vlastní komunikace, mají každý svůj definovaný formát a upravují se v textovém editoru. Soubor „pglog" je využíván serverem a obsahuje informace o webových stránkách. Formát každého řádku v souboru je: <*číslo uzlu serveru*> <*ID webové stránky*> <*velikost webové stránky v bytech*> <*počet zobrazení stránky během simulace*>. Číslo uzlu serveru je v této simulaci vždy 3. ID webové stránky je vlastně pořadové číslo stránky (čísluje se postupně od 0). Hodnoty velikosti webové stránky se mohou libovolně zvolit a hodnota počet zobrazení stránky je závislá na druhém vstupním souboru „reqlog", jenž obsahuje informace o tom, kdy který klient vysílá požadavek na server o stáhnutí určité webové stránky. Formát všech řádků (kromě posledního) v souboru je: <*čas vyslání požadavku klientem* > <*číslo uzlu klienta*> <*číslo uzlu serveru*> <*ID webové stránky*>. Čísla uzlů klientů jsou 0 a 1. Poslední řádek souboru obsahuje informace o celkovém trvání a o počtu webových stránek, o které klienti požádali (číslo je shodné s počtem stránek v souboru pglog). Obsah vstupních souborů je znázorněn v tab. 1.

Tab. 1: Obsah vstupních souborů.

obsah souboru pglog	obsah souboru reqlog
3 0 2788 2	1.000000 0 3 0
3 1 924 2	3.000000 1 3 2
3 2 1356 2	3.000000 0 3 2
3 3 1862 2	5.000000 1 3 3
	7.000000 0 3 1
	7.000000 1 3 0
	9.000000 1 3 1
	11.000000 0 3 3
	i 13.000000 4

Popis výstupního logovacího souboru:

Pro zaznamenání HTTP událostí se používá výstupní logovací soubor, který je uložen na stejném místě na disku s ostatními zmiňovanými soubory a nese název http.log. Formát každého řádku logovacího souboru vypadá následovně:

- čas, kdy se událost stala,
- číslo uzlu, na kterém se událost stala,
- typ objektu (server, client, cache),
- typ události,
- hodnoty.

Typy událostí a jejich hodnoty jsou rozepsány v tab. 2 a vysvětleny v tab. 3. Jako typ objektu značí písmeno C klienta, písmeno E cache a písmeno S značí server.

Tab. 2: Formát logovacího souboru.

typ objektu	typ události	hodnoty
C	GET	p <ID stránky> s <číslo uzlu cache> z <velikost požadavku>
E	MISS	p <ID stránky> c <číslo uzlu klienta > s <číslo uzlu serveru > z <velikost požadavku>
S	MOD	p <ID stránky> m <poslední čas modifikace stránky> n <následující čas modifikace stránky>
E	ENT	p <ID stránky> m <poslední čas modifikace stránky (pouze dynamické stránky, u statických = 0)> z <velikost webové stránky v bytech> z <číslo uzlu serveru >
E	SND	c <číslo uzlu klienta > p <ID stránky> z <velikost stránky v bytech>
C	RCV	p <ID stránky> s <číslo uzlu cache> l <časový interval mezi posláním požadavku GET a přijetím stránky klientem> z <velikost stránky >
E	HIT	p <ID stránky> c <číslo uzlu klienta > s <číslo uzlu serveru >
C	STA	p <ID stránky> s <číslo uzlu serveru > l <časový úsek uplynutý od poslední modifikace stránky>
S	INV	p <ID stránky> s <číslo uzlu serveru >

Tab. 3: Vysvětlení jednotlivých typů událostí.

typ objektu	typ události	vysvětlení
C	GET	Klient vyšle tento požadavek jako žádost o webovou stránku.
E	MISS	Pokud není požadovaná stránka nalezena v cache, je vyslán tento požadavek serveru o poslání požadované stránky.
S	MOD	Server modifikuje webovou stránku a nastaví i čas příští modifikace. Stránka se začíná odesílat ze serveru.
E	ENT	Celá stránka je vložena do cache.
E	SND	Požadovaná stránka se začíná odesílat klientovi.
C	RCV	Klientovi je doručena celá stránka.
E	HIT	Požadovaná stránka je nalezena v cache a je poslána klientovi.
C	STA	Událost nastává při doručení stránky klientovi, pokud je tato stránka dynamická a byla uložena v cache.
S	INV	Server pošle požadavek o zrušení platnosti dynamické stránky. Požadavek není zaznamenán v logovacím souboru, protože využívá pro přenos svůj vlastní protokol.

Nyní následuje ukázka části obsahu výstupního logovacího souboru (tab. 4). Vybrané řádky jsou očíslovány (číslo události) a poté podle čísla vysvětleny v následující tab. 5.

Tab. 4: Ukázka části logovacího souboru.

číslo události	Výpis událostí z logovacího souboru v čase od 5 sekund do 10 sekund
1	`5 i 1 C GET p _o85:3 s 2 z 43`
2	`5.020066 i 2 E MISS p _o85:3 c 1 s 3 z 43`
3	`5.070509 i 3 S MOD p _o85:3 m 5.0705090666666672 n 10005.070509066667`
4	`5.328648 i 2 E ENT p _o85:3 m 0 z 1862 s 3`
5	`5.328648 i 2 E SND c 1 p _o85:3 z 1862`
6	`5.429869 i 1 C RCV p _o85:3 s 2 l 0.42986853333333475 z 1862`

7	6.211245 i 3 S MOD p _o85:0 m 6.2112450666666668 n 11.211245066666667
8	7 i 0 C GET p _o85:1 s 2 z 43
9	7 i 1 C GET p _o85:0 s 2 z 43
10	7.020066 i 2 E MISS p _o85:1 c 0 s 3 z 43
11	7.020066 i 2 E HIT p _o85:0 c 1 s 3
12	7.020066 i 2 E SND c 1 p _o85:0 z 2788
13	7.070509 i 3 S MOD p _o85:1 m 7.0705090666666672 n 10007.070509066667
14	7.122092 i 1 C STA p _o85:0 s 3 l 0.91084693333333355
15	7.122092 i 1 C RCV p _o85:0 s 2 l 0.12209200000000031 z 2788
16	7.226077 i 2 E ENT p _o85:1 m 0 z 924 s 3
17	7.226077 i 2 E SND c 0 p _o85:1 z 924
18	7.286912 i 0 C RCV p _o85:1 s 2 l 0.28691226666666747 z 924
19	9 i 1 C GET p _o85:1 s 2 z 43
20	9.020066 i 2 E HIT p _o85:1 c 1 s 3
21	9.020066 i 2 E SND c 1 p _o85:1 z 924
22	9.080902 i 1 C RCV p _o85:1 s 2 l 0.080901600000000684 z 924

Tab. 5: Vysvětlení číslovaných událostí logovacího souboru.

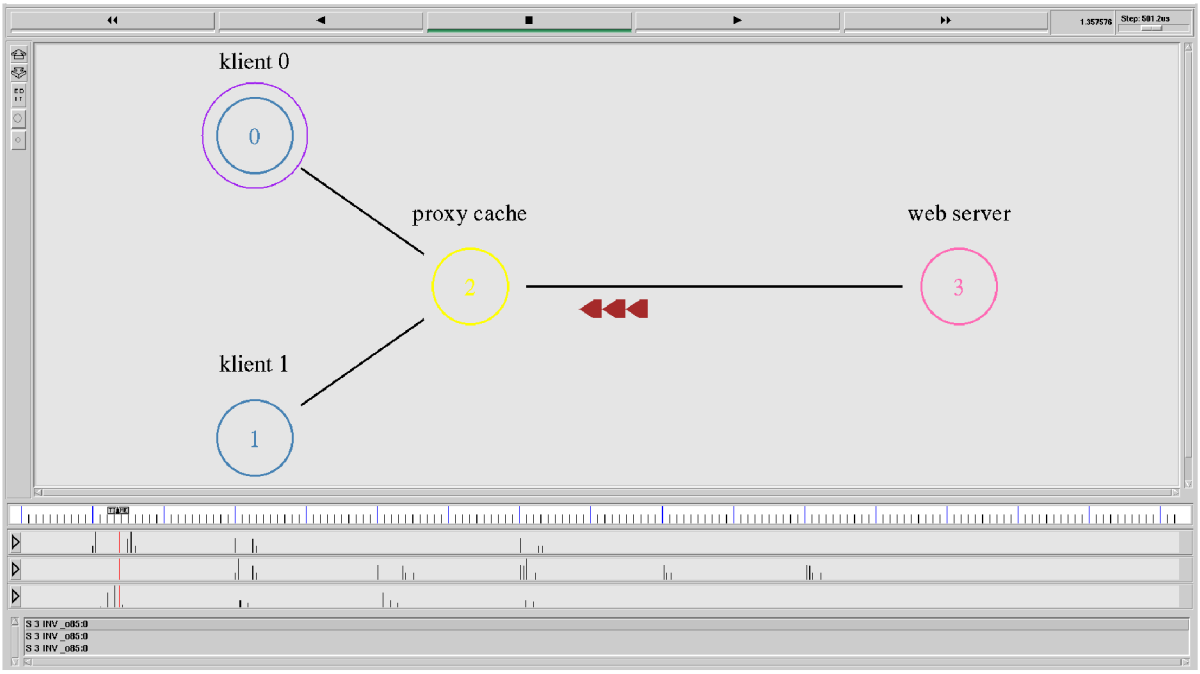
číslo události	**Popis jednotlivých událostí logovacího souboru**
1	V čase pěti vteřin od startu simulace klient 1 odešle požadavek GET o stáhnutí webové stránky číslo 3 (ID stránky je _o85:3). Požadavek má velikost 43 bytů a je doručen do cache, kde čeká na další vyhodnocení.
2	Poté, co požadovaná stránka číslo 3 není v cache nalezena (událost MISS), je vyslán požadavek, který obsahuje číslo požadované stránky, číslo uzlu klienta, velikost požadavku, směrem k serveru.

3	Server nastaví u stránky číslo 3 aktuální čas modifikace a následující čas modifikace. Podle hodnoty 10005.07 za parametrem „n" je zřejmé, že se jedná o statickou webovou stránku, protože interval 10000 sekund nastavuje statický generátor modifikačních intervalů. Stránka se začíná formou paketů odesílat do cache.
4	V tomto okamžiku je celá stránka vložena do cache (dorazily všechny pakety stránky). Událost ENT obsahuje informace o čísle stránky i o velikosti.
5	Událost SND proběhne ve stejném okamžiku jako událost ENT. To znamená, že v momentě vložení stránky do cache začne odesílání této stránky klientovi číslo 1. Událost SND obsahuje informace o číslu uzlu klienta, kterému je stránka posílána, číslu stránky a velikosti stránky.
6	Klient obdrží všechny pakety stránky (celou stránku). Událost RCV informace o číslu stránky, velikosti stránky a za parametrem „l" je doba, která uplynula od vyslání požadavku klienta o stáhnutí stránky po přijmutí stránky klientem.
7	Dochází k modifikaci stránky číslo 0. Tato stránka je dynamická, protože její následující čas modifikace je 11.211 sekund. Zvolení dynamických a statických stránek probíhá při startu simulace náhodně. Stránka 0 bude každých 5 sekund modifikována. Modifikace stránky je vykonána na základě přijmutí události INV(zruší platnost stránky a událost není zobrazena v logovacím souboru) a nastaví stránce další čas modifikace. Událost INV je zobrazena při simulaci v programu NAM hned pod časovou osou a možnými grafy.
8	Tentokrát o webovou stránku číslo 1 požádal klient číslo 0.
9	Ve stejný okamžik je odeslán i požadavek od klienta číslo 1, který žádá o stránku číslo 0.
10	Požadavek klienta 0 je doručen do cache a po nenalezení požadované stránky (událost MISS) je vyslán požadavek serveru o stáhnutí požadované stránky číslo 1.
11	Stránka číslo 0, o kterou požádal klient 1, je nalezena v cache.
12	Stránka 0 se po nalezení v cache začíná odesílat klientovi 1.
13	Stránka 1se modifikuje a je nastaven i čas příští modifikace. Webová stránka číslo 1 je statická.

14	Událost STA signalizuje, že stránka 0, kterou přijal klient 1, je dynamická, a že od poslední modifikace uplynul již čas 0.9108 sekund.
15	Událost RCV nastane společně s událostí STA a signalizuje přijmutí požadované stránky číslo 0.
16	Stránka číslo 1 je vložena do cache. Protože je stránka statická, tak za parametrem „m" je vždy hodnota 0.
17	Ihned po vložení stránky číslo 1 do cache se začíná stránka odesílat klientovi číslo 0.
18	Stránka číslo 1 je doručena klientovi číslo 0.
19	Klient 1 vyšle požadavek o stáhnutí stránky číslo 1.
20	Statická stránka číslo 1, o kterou požádal klient 1, je nalezena v cache.
21	Začíná odesílání stránky 1 ke klientovi číslo 1.
22	Stránka číslo 1 je doručena klientovi číslo 1.

Popis simulace:

Program se spustí zapsáním příkazu ns http_proxytrace.tcl do okna terminálu. Poté nastartuje program NAM a zobrazí simulaci. Po spuštění přehrávání simulace se nejprve klient 0 synchronizuje s cache pomocí three-way handshake a vyšle požadavek o stáhnutí webové stránky do cache. Tento stav zobrazuje fialový kruh kolem klienta 0, který zmizí až po doručení celé stránky klientovi 0. Po nenalezení požadované stránky v cache probíhá nejprve synchronizace cache se serverem a poté dochází k odesílání stránky ze serveru do cache. Když cache obdrží první svoji stránku, tak se změní barva uzlu cache ze žluté na tmavě modrou. Při startu komunikace klienta 1 je opět nutná synchronizace s cache. Z obr. 3 je na třech grafech, které jsou pod časovou osou, patrné, že mezi posílanými daty jsou časové mezery a to znamená, že je chvíli vidět pohyb paketů a další chvíli se neděje nic. K přeskočení těchto časových mezer se používá klávesa „n". Když během simulace vyprší platnost některé stránky v cache je to indikováno událostí INV, která se zobrazí pod grafy v okně aplikace NAM. V ten stejný okamžik, kdy je stránka zbavena platnosti, musí server tuto stránku v cache modifikovat. Tato modifikace je zaznamenána do logovacího souboru, ale v simulaci to není nijak patrné, protože při modifikování stránek se využívá jiný protokol pro přenos informací.



Obr. 3: Komunikace dvou klientů se serverem přes proxy cache (s PagePool/ProxyTrace).

C Obsah CD

```
-- bakalarska_prace.pdf
-- PagePools
   |-- CompMath
   |   |-- http.log
   |   |-- http.nam
   |   |-- http.tr
   |   `-- http_compmath.tcl
   |
   |-- Math
   |   |-- http.log
   |   |-- http.nam
   |   |-- http.tr
   |   `-- http_math.tcl
   |
   |-- ProxyTrace
   |   |-- ProxyTrace1
   |   |   |-- http.log
   |   |   |-- http.nam
   |   |   |-- http_proxytrace.tcl
   |   |   |-- pglog
   |   |   `-- reqlog
   |   |
   |   |-- ProxyTrace2
   |   |   |-- http.log
   |   |   |-- http.nam
   |   |   |-- http_proxytrace.tcl
   |   |   |-- pglog
   |   |   `-- reqlog
   |   |
   |   `-- ProxyTrace3
   |       |-- gr1.tr
   |       |-- gr2.tr
   |       |-- http.log
   |       |-- http.nam
   |       |-- http_proxytrace.tcl
   |       |-- pglog
   |       `-- reqlog
   |
   `-- WebTraf
       |-- bw.tr
       |-- bw1.tr
       |-- bw2.tr
       |-- http.nam
       |-- http_webtraf.tcl
       |-- lost.tr
       `-- queue.tr
```