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

MATEMATICKÉ MODELOVÁNÍ SÍŤOVÉHO PROVOZU SLUŽBY WWW

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

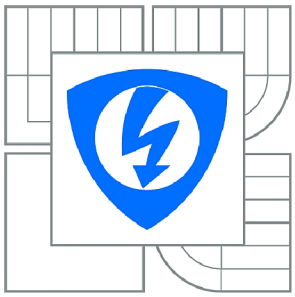
AUTOR PRÁCE
AUTHOR

LUKÁŠ PAPŠÍK

BRNO 2012

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

MATEMATICKÉ MODELOVÁNÍ SÍŤOVÉHO PROVOZU SLUŽBY WWW

MATHEMATICAL MODELLING OF WWW NETWORK TRAFFIC

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

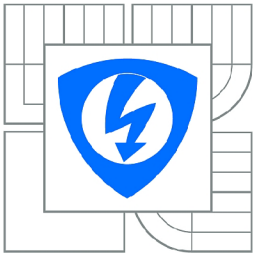
AUTOR PRÁCE — LUKÁŠ PAPŠÍK

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. KAROL MOLNÁR, Ph.D.

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Lukáš Papšík	***ID:***	125171
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Matematické modelování síťového provozu služby WWW

POKYNY PRO VYPRACOVÁNÍ:

Proveďte dlouhodobé zachycení provozu protokolu HTTP s minimálním trváním dvou dnů. Na základě vlastností zachyceného síťového provozu navrhněte metodu pro odvození parametrů pro klasický model síťového provozu (např. Markovský modulovaný, ON/OFF, IPP či Autoregresivní model). Porovnejte výsledné chování modelu se zachyceným provozem.
Prostudujte popis stochastického modelu síťového provozu a navrhněte postup pro odvození parametrů modelu. Odvoďte či odhadněte na základě navržených metod co nejpřesněji parametry pro stochastický model. Porovnejte chování stochastického modelu se zachyceným reálným provozem. Zdokumentujte nově získané poznatky.

DOPORUČENÁ LITERATURA:

[1] FROST, V., MELAMED, B. Traffic Modeling for Telecommunication Networks, IEEE Communications Magazine, 32(3), pp. 70-80, March, 1994
[2] FAPOJUWO, A., LEE, I.: Mathematical Modeling and Characterization of Wireless Network Traffic. Hauppauge: Nova Science Publishers, 2008, ISBN: 978-1604568691.
[3] PAPOULIS, A., PILLAJ, S.U.: Probabilities, Random Variables and Stochastic Process. New York: McGraw-Hill, 2002, ISBN:978-0071226615.

Termín zadání:	6.2.2012	***Termín odevzdání:***	31.5.2012

Vedoucí práce: doc. Ing. Karol Molnár, Ph.D.
Konzultanti bakalářské práce:

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

ABSTRAKT

Cílem bakalářské práce bylo prakticky využít získané teoretické znalosti ze semestrální práce. Náplní práce bylo analyzovat dlouhodobý provoz z důrazem na získání potřebných parametrů pro vybrané modely. Pro analýzu sítě bylo použito dvou programu, které získaly potřebné parametry. Jak zachycená data upravit, aby bylo možné je použít v programech, je popsáno v práci. V práci se zaměřuji na dva vybrané modely, na model Stochastický a Markovsky-modulovaný Poissonovský. Každý z těchto modelů potřebuje jiné parametry. Ty byly získané ze zachyceného provozu. Dále byl modelovaný provoz porovnán s reálným. Bylo zjištěno, že ani jeden z modelů není schopen dostatečně přesně modelovat chování sítě, pokud přijde shluk paketů. Pokud však provoz vykazoval lineární charakter, modely byly schopné tento provoz modelovat.

KLÍČOVÁ SLOVA

Hurst, predikce, intenzita, sobě-podobnost

ABSTRACT

The aim of this thesis was to practically use theoretical knowledge of the semestral work. The scope of work was to analyze long term operation of an emphasis on obtaining the necessary parameters for selected models. Network analyzer was used for two programs that have received the required parameters. How do I edit the captured data that can be used in programs described in the work. In this work I focus on two models, the Stochastic model and Markov-modulated Poisson's. Each of these models requires different parameters. These were obtained from the captured traffic. Furthermore, the modeled traffic compared with the real. It was found that none of the models is not sufficiently capable of accurately modeling the behavior of the network, if it comes to cluster of packets. However, if traffic shown linear character models were able to model this traffic.

KEYWORDS

Hurst, prediction, intensity, self-similary

PAPŠÍK, Lukáš *Matematické modelování síťového provozu služby WWW*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 50 s. Vedoucí práce byl doc. Ing. Karol Molnár, Ph.D

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Matematické modelování síťového provozu služby WWW“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a 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.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Karlu Molnárovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Žijeme v době, kdy si už neumíme představit život bez internetu a dalších počítačových vymožeností. Počítačová síť jak „drátová“, tak „bezdrátová“ nás obklopuje na každém kroku. Proto je v dnešní době modelování jejich provozu velice důležité. Nejde samozřejmě jenom o modelování obrovských MAN či WAN sítí, ale do těchto modelu se musí zahrnout i malé a střední sítě PAN , LAN či WLAN . Modelování v těchto a dalších sítí je důležitě z hlediska zajištění přenosové rychlosti, ztrátovosti, zpoždění a QoS . Je také důležité pro výrobce těchto zařízení, vývojáře softwaru, který se stará o přenos dat. Sebelepší technologie přenosu dat neuspěje, pokud ji třeba zaruší stávající „starší“ technologie. Výrobci jednotlivých síťových prvků, směrovačů a přepínačů si dobře uvědomují, že tyto síťové prvky mají svá omezení. Ty spočívají v kapacitě připojených zařízení, uživatelů atd. Modelováním se výrobci nebo zřizovatelé sítí mohou vypořádat s rozdílnými rychlostmi, mohou predikovat uživatelské chování na rozdílných datových tarifech a reagovat na asymetrii rychlosti uplinku a downlinku. Dále můžeme predikovat „budoucí“ chování sítě a tak tuto síť připravit tak, aby byla schopna reagovat na „budoucí“ změny. Tato „budoucnost“ se projevuje jako náhlý příchod většího množství dat. Na tuto změnu musí síťový prvek nebo síť okamžitě zareagovat tak, aby nebyl ohrožen uživatelův komfort. K modelování se přistupovalo již v počátcích telefonie, kde sice predikce nemusela být tak sofistikovaná, ale již uměla předvídat chování sítě a zřizovatelům pomáhala při budování této sítě.

Obsahem této práce je využít teoreticky získané poznatky v praxi na vybraných dvou modelech. Bude zachycen běžný dlouhodobý domácí provoz. Tento provoz bude analyzován, budou získány základní parametry tohoto provozu a chování uživatele. Následně budou vybrány dva modely. Budou zjištěny jednotlivé parametry potřebné pro dané modely. Nakonec bude modelovaný provoz porovnán se skutečným a budou vyvozeny závěry.

1 ŘEŠENÍ STUDENTSKÉ PRÁCE

1.1 Protokol HTTP

Jelikož je má bakalářská práce o matematickém modelovaní síťového provozu HTTP, musím se o tomto protokolu zmínit. Protokol HTTP je protokolem aplikační vrstvy[1] a pro internet jako takový se používá od roku 1990. Vlastní přenos dat v protokolu HTTP zajišťuje nižší vrstva, kterou je vrstva transportní a protokolem této vrstvy pro přenos dat u HTTP je protokol TCP. HTTP protokol dnes pracuje s verzí 1.0 a 1.1. HTTP protokol verze 1.0 pracuje na principu dotaz-odpověď. Klient pošle požadavek na server, server odpoví a komunikace se rozpojí. Pokud klient odešle znovu požadavek na server, navazuje se nové spojení a opět se uzavírá. Tento děj se opakuje, dokud klient odesílá požadavky. Informace o tom, která data klient již obdržel (informace o komunikaci), jsou čistě na uživateli. Protokol si informace o spojeních neudržuje, a proto je protokolem bezstavovým. Verze protokolu 1.1 již umožňuje trvalé spojení a to vede ke snížení zátěže, která vzniká při neustálém sestavovaní spojení. Trvalé spojení tedy funguje tak, že klient odesílá požadavky na server a ten mu odesílá odpovědí, dokud neukončí komunikaci a proto musí být definován mechanizmus ukončení. Tento protokol také podporuje zřetězení požadavků u klienta, který podporuje trvalé spojení. Další novým prvkem v HTTP protokolu je snaha o poslání co nejlepší varianty dokumentu, pokud samozřejmě existuje několik různých variant (např. různý jazyk, formát, komprese apod.). HTTP 1.1 poskytuje dva dohadovací mechanismy, které je možné kombinovat nebo použít odděleně:

- serverem řízené dohadování,
- klientem řízené dohadování.

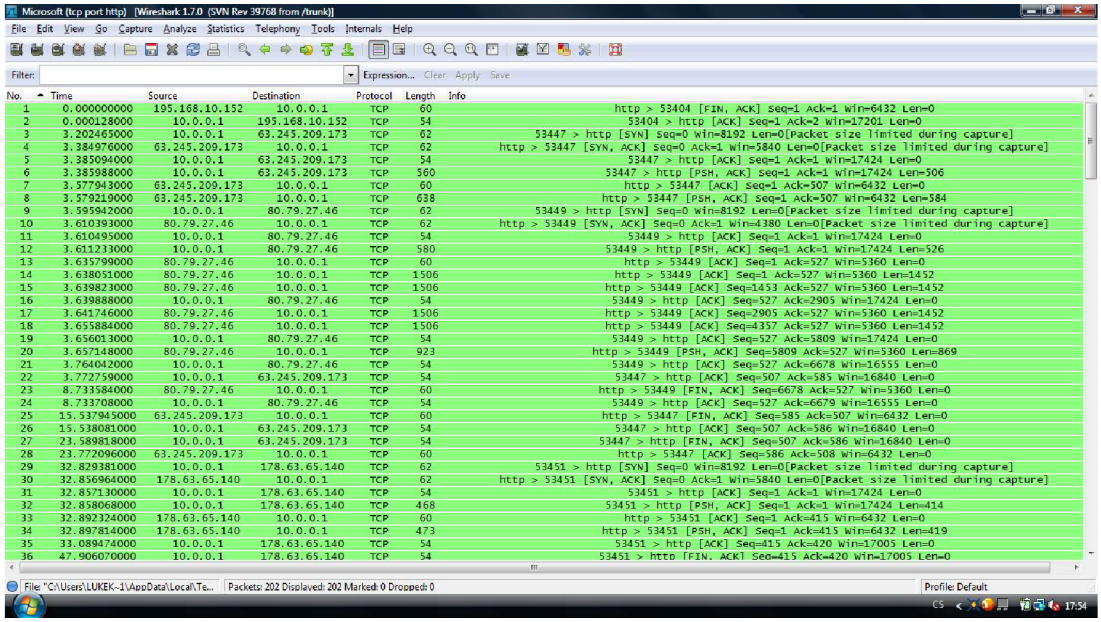
Podle toho co klient od serveru požaduje, rozlišujeme následující metody [14]:

- **OPTIONS** - Metoda OPTIONS představuje dotaz na možnosti komunikace spojené s uvedeným URL. Metoda umožňuje klientovi určit možnosti a omezení spojené se zdrojem nebo schopnostmi serveru. Pokud je URL v dotazu ve tvaru „*“, pak se jedná o dotaz na možnosti serveru jako celku.
- **GET** - Metoda GET představuje požadavek na poslání dokumentu určeného pomocí URL. V souvislosti s proxy se může metoda GET změnit na „podmíněný GET“, která požaduje poslat dokument pouze za určitých podmínek definovaných v hlavičce dotazu.
- **HEAD** - HEAD metoda je identická s metodou GET, server však nemusí posílat tělo odpovědi. Metodu je možné použít k získání doplňkových informací o dokumentu, často se používá k testování hypertextových linek, jejich

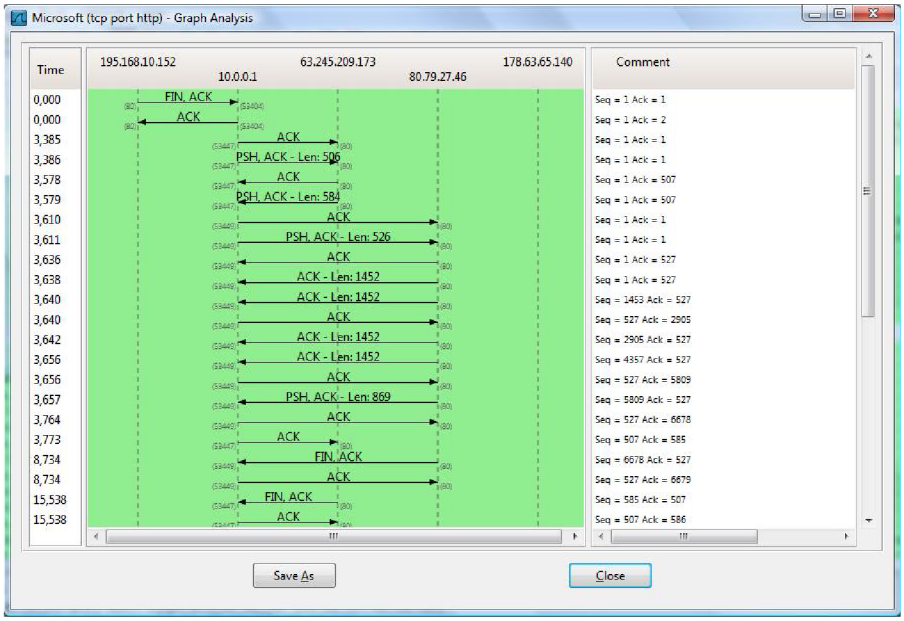
[1] 7 vrstva ISO/OSI modelu

dostupnosti a poslední modifikace. Klient může získané hlavičky analyzovat a případně požádat o data novým dotazem GET (např. test zda dokument není příliš dlouhý).

- **POST** - POST metoda se používá v případě, kdy má cílový server přijmout data z požadavku. Skutečná funkce metody závisí na URL s ní spojené. Výsledkem POST metody může být poslání mailu, předání dat do procesu, který data zpracuje, rozšíření databáze. Posílaná data nejsou nijak omezená a je možné v hlavičkách tělo zprávy popsat.
- **PUT** - PUT metoda představuje požadavek na uložení posílaných dat pod specifikované URL na server. Takto uložená data budou dostupná např. následnými dotazy GET. Tato metoda se již v praxi příliš nevyužívá.
- **DELETE** - Požadavek na zrušení dokumentu na serveru. Rušený dokument je specifikován v URL. Stejně jako metoda PUT se v praxi již příliš nevyužívá.
- **TRACE** - Metoda použitá k testování originálního serveru. Originální server má vrátit klientovi kladnou odpověď bez dat.



Obr. 1.1: Ukázka síťové komunikace HTTP zachycené programem Wireshark 1.7.0.



Obr. 1.2: Ukázka výměny zpráv

2 MODELY SÍŤOVÉHO PROVOZU

2.1 Sobě-nepodobnost

První modely vycházely z telekomunikačního modelu[12] a zaměřily se především na jednoduchost analýzy. Tyto jednoduché modely většinou pracují tak, že agregace provozu z více zdrojů inklinuje k vyhlazení shluků tzn., že čím více zdrojů je připojeno/provozováno, tím více mizí shlukový charakter provozu. A zvláště první modely ignorují shluky úplně. S postupem času se sice zkoušely přidávat shluky do modelů, o čem svědčí například Coumpand Poisson nebo MMPM . V této době byla sobě-podobnost zcela neznámá a matematická definice sobě-podobnosti byla považována za zvláštnost. Pro modelování síťového provozu HTTP nás ovšem zajímá především sobě-podobný provoz.

2.1.1 Poissonův distribuční model

Nejstarší a jeden z nejrozšířenějších modelů provozu. Využíval se již v telefonii a patří mezi sobě-nepodobné modely. Jeho jediným parametrem je rychlost příchodu λ . Má dva jednoduché předpoklady:

- počet zdrojů je nekonečný,
- příchozí vzorek je náhodný.

Rozdělení pravděpodobnosti funkce je 2.1. Poissonovy distribuce jsou tedy vhodné při přenosu z velkého množství nezávislých zdrojů. Proto má Poissonův model jednoduchá pravidla. Příchody paketu jsou nezávislé a interval mezi pakety je rozdělen exponenciálně 2.3. Je také důležité podotknou, že Poissonovy modely dokáží přesně predikovat chování sítě na krátkou dobu. Jsou hodně populární v teorii front, protože z hlediska analytických vlastnosti patří mezi sobě-nepodobné a agregace dalšími Poissonovými proudy vytvoří nový proud tzv: „Poissonův proud“2.2. Poisson se pro tuto vlastnost hojně využívá v modelu front.

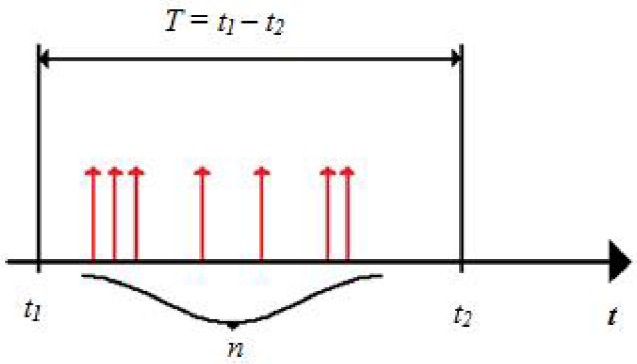
$$F(t) = 1 - \mathrm{e}^{-\lambda \cdot t} \tag{2.1}$$

$$\lambda^t = \sum \lambda \tag{2.2}$$

$$\lambda : P\{A_n\} = 1 - \exp(-\lambda \cdot t) \tag{2.3}$$

2.1.2 Markovské modely

Markovské modely jsou založeny na Markovově teorii řízení front. Požadavkům přicházejícím do systému front odpovídá vstupní tok, který můžeme vidět na obrázku 2.1 náhodných příchodu.



Obr. 2.1: Markův systém front

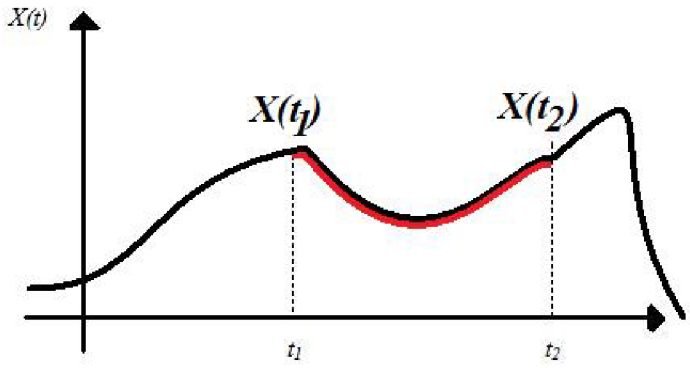
Charakteristikou tohoto vstupu je jeho intenzita. Ta je dána počtem příchozích požadavků, které vstoupí v daném časovém úseku do systému. Tato intenzita se označuje symbolem λ a můžeme ji zjistit takto:

$$\lambda = \frac{n}{T} \tag{2.4}$$

Kde n značí počet příchozích požadavků za dobu T. Dalším důležitým aspektem je stacionarita neboli stálost v čase. Pokud chceme analyzovat nebo simulovat jakýkoli stochastický proces, musí být tento proces stacionární. Avšak téměř každý stochastický proces je nestacionární. Ukázku nestacionárního průběhu můžeme vidět na obr: 2.2.

Pokud tedy chceme stochastický průběh analyzovat, můžeme tento průběh rozdělit na dílčí úseky, na obrázku vyznačen červenou čarou, a o těchto dílčích úsecích můžeme tvrdit, že jsou stacionární. Tím jsme charakterizovaly intenzitu vstupu. Analogicky k intenzitě vstupu můžeme definovat intenzitu obsluhy. Ta se značí jako μ a má stejný rozměr jako intenzita a tím je čas. Intenzita obsluhy vyjadřuje průměrný počet obsloužených jednotek za jednotku času a lze ji vyjádřit vztahem:

$$\mu = \frac{m}{T} \tag{2.5}$$



Obr. 2.2: Stochastický proces

Kde m je počet obsloužených požadavků a T je interval sledování. A pokud známe intenzitu obsluhy, můžeme odvodit střední dobu obsluhy, kterou lze vyjádřit vztahem:

$$\frac{1}{\mu} \tag{2.6}$$

Potom střední doba obsluhy odpovídá době, za kterou je obsloužen jeden požadavek. Markovské modely tedy předpokládají konečný počet stavů. Víme, že patří mezi sobě-nepodobné modely a proto můžeme definovat, že budoucí stav X_{n+1}, záleží jen na aktuální stavu X_n a ne na jiných stavech X_i, kde $i<n$. Tato schopnost, tedy odkazování se na posloupnost náhodných proměnných X, se nazývá diskrétní Markovský řetězec.

2.1.3 Markovsky-modulovaný Poissonovský model

Tento model se snaží odstranit nedostatky Poissonova modelu a to zejména provoz shlukového charakteru, který je v telekomunikační síti běžný. Zejména když paket přijde ve shlucích. A je proto široce používaným nástrojem pro analýzu telekomunikačních modelů. Důležité bylo v souladu s hlasovým modelem měnit příchozí rychlost skokově tzv.: kvantovat. Markovský řetěz je dvoustavový. Každý stav má přiřazenou intenzitu λ a průměrnou dobu pobytu požadavku v systému r tzn. že model je definován jako 4-ntice (λ_1 ; λ_2 ; R_1 ; R_2).Pro zjištění těchto čtyř parametrů je nutný vzorek reálného síťového provozu. Parametry jsou vybrány tak, aby odpovídaly skutečnému provozu a modelu provozu pro následující charakteristiky:

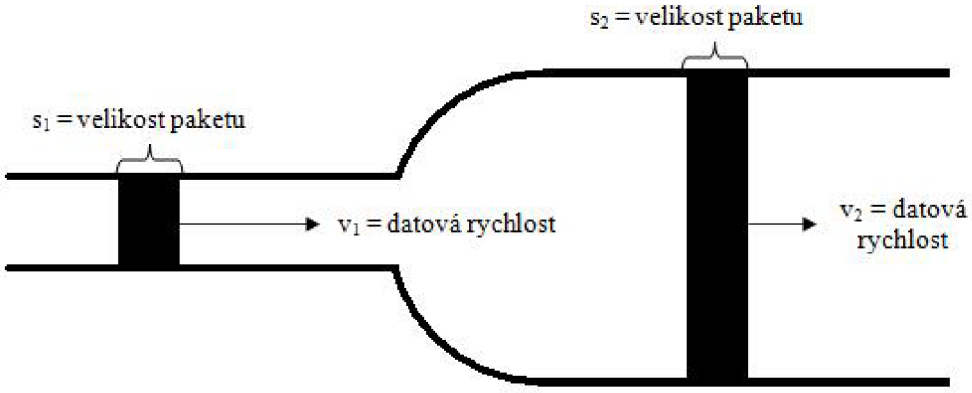
- průměrná příchozí rychlost,
- krátkodobý rozptyl od střední hodnoty počtu příchodů,
- dlouhodobý rozptyl od střední hodnoty počtu příchodů,

- počet příchodů v krátkém časovém úseku.[13]

Tento model také využívá některé užitečné vlastnosti spjaté s modely Poisson. Například přidáním dalšího proudu s hodnotou λ, je tento proud přidán do n-tice $(\lambda + \lambda_1 \ ; \ \lambda + \lambda_2 \ ; \ R_1 \ ; \ R_2)$. Jak vidíme, není těžké tento model rozšířit na více než dva stavy. V tomto případě musíme použít řešení matice řádu *2 · (N+1)*, kde N je velikost vyrovnávací paměti. Proto tento model mohou využít „návrháři“, kteří mohou stanovit dostatečnou velikost vyrovnávací paměti.

2.1.4 Markovsky-modulovaný model proudění

Modely založené na proudění kapalin jsou koncepčně velice jednoduché. Tyto modely jsou využívány v mnoha oblastech výzkumu. Oproti klasickým, které berou příchod paketu jako samostatnou událost a tím musí mít velkou kapacitu paměti, aby byli schopny proces simulovat. Modely proudění zaznamenávají změny akorát při změně „průtoku“, tedy pokud přijde shluk paketů.



Obr. 2.3: Bernoulliho rovnice a jeho srovnání s datovým provozem

Tento přístup je z hlediska výpočetního výkonu méně náročný, protože změny „průtoku“ ve srovnání s příchody jednotlivých paketů, jsou méně časté. Základním rysem modelu proudění 2.3 je charakterizovat provoz na síti jako nepřetržitý proud vstupu s konečnou rychlostí proudění [1]. Model tedy zachytí změny na vstupu pro různé události vyskytující se v síti. Těmito změnami jednoduše charakterizuje provoz, který lze lehce simulovat. Jako každý Markovský model i MMFM[1] používá MC[2]. Z tohoto modelu je zřejmé, že zde budou platit stejná pravidla jako pro Bernoulliho

[1]MMFM = Markov-Modulated Fluid Model
[2]Markov Chain = Markovský řetěz

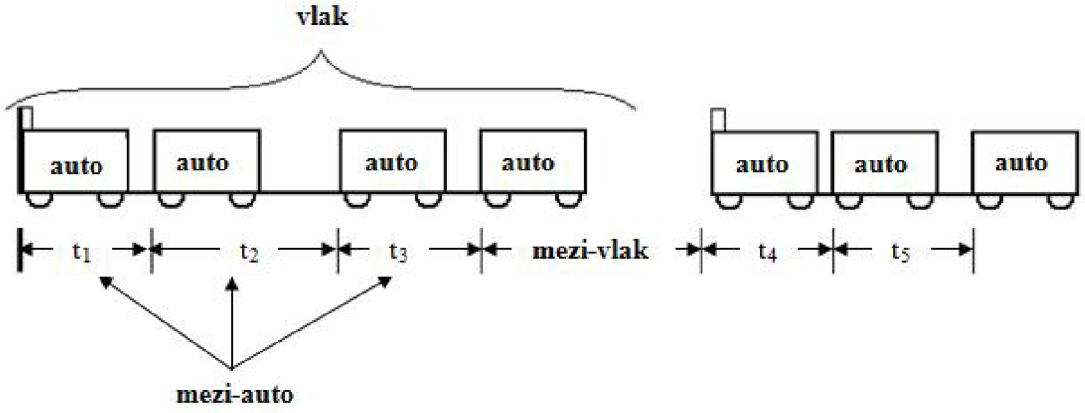
rovnici, akorát místo rychlosti proudění zde máme datovou rychlost a místo průřezu, kterou prochází kapalina zde máme velikost paketu a musí zde platit:

$$s_1 \cdot v_1 = s_2 \cdot v_2 \tag{2.7}$$

2.1.5 Model vlaku

Tento model vznikl jako další ukázka pokusu o dokázaní shlukového provozu. Je založen na předpokladu, že pokud pakety přicházejí za sebou v krátkém čase, směrují se do stejného cílového místa. Proto pro generaci provozu vymysleli autoři [6] tzv. model vlaku. Model vlaku je tedy sekvence po sobě jdoucích paketů z jednoho zdroje mířícího do stejného cíle s odpověďmi v opačném směru. Tato série je poté označována jako vlak a v rámci vlaku můžeme volitelně dělit tento vlak na tandemové přívěsy. Tandemové přívěsy můžeme charakterizovat také jako sérii po sobě jdoucích paketů, avšak v rámci jedné zprávy. V tomto modelu můžeme charakterizovat samotný vlak 2.4, ale i jeho tandemový přívěs. Parametry, kterými se vyznačuje model vlaku jsou:

- průměr mezi-vlakového příchozího času,
- průměr mezi-auto příchozího času,
- průměrná délka auta,
- průměrná velikost vlaku.



Obr. 2.4: Model vlaku

Vlakový model je navržen pro analýzu skutečného provozu, není navržen pro tvorbu umělého zatížení simulace. Nespornou výhodou tohoto modelu je, že nepřekračuje meze systému a to zároveň dokazuje, že různé druhy provozu mají různé

vlastnosti podle typu protokolu [6]. Bohužel jeho velká výhoda je zároveň jeho nevýhodou, protože analyzovat systém pomoci modelu vlaku přináší obrovskou matematickou náročnost. Je to způsobeno velkou podrobností tohoto modelu. Například pro návrháře síťového uzlu je důležité znát jak velká vyrovnávací paměť je nutná. Avšak návrhář nemůže znát dokonale všechny parametry sítě, před nasazením onoho síťového uzlu, protože kompletní popis provozu by musel být podporován před vlastní analýzou.

2.2 Sobě-podobnost

Sobě-podobnost odstraňuje, respektive se snaží odstranit problém předchozích modelů a to nedostatečná predikce na velkém časovém úseku. V předchozích modelech platilo, že čím je agregace větší, tím budou výsledky v průměru pravidelnější. To v modelech založených na sobě-podobnosti neplatí. Sobě-podobnost platí v modelech, které vykazují stejné vlastnosti ve všech měřítkách. Takže například trasování v takovémto modelu bude vypadat stejně agregovaně pro *10 ms* úsek jako pro *10 s* úsek. Toto neplatí například v modelu Poisson, kde platí, jak je uvedeno výše 2.1.1, že čím větší časový úsek máme, tím bude průběh hladší a limitně se bude blížit přímce při dostatečně dlouhém časovém úseku. Naproti tomu sobě-podobný provoz bude vykazovat shluky v jakémkoli měřítku. Avšak ne vždy jsou modely založené na sobě-podobnosti vhodné.

2.2.1 Definice sobě-podobnosti

Jak jsme si již řekly v předchozím odstavci, sobě-nepodobné modely nedokáží dostatečně přesně predikovat chování na velkém časovém úseku. Je to dáno historickým vývojem. První sobě-nepodobné modely se zabývaly převážně predikcí telefonního provozu. Datový provoz, naproti telefonnímu vykazuje velkou časovou variabilitu. Proto musí být modely schopné tuto variabilitu zahrnout. To v případě sobě-nepodobných modelů nepřipadalo v úvahu. Sobě-podobné modely musí být schopné zahrnout nejen velkou časovou variabilitu vyskytující se v datovém provozu, ale také schopnost reagovat na shluky příchozích dat. Tyto vlastnosti jsou typickými projevy provozu nejen služby www, ale i dalších.

Sobě-podobný provoz je charakterizován parametrem Hurst tzn. mírou sobě-podobnosti. Míra sobě-podobného provozu je charakterizována na intervalu:

$$0 \leq H \leq 1. \tag{2.8}$$

Tento interval rozdělujeme do dvou oblastí. Do oblasti SRD neboli krátkodobé závislosti a na oblast LRD neboli dlouhodobou závislost. SRD spadá do intervalu:

$$0 \leq H < \frac{1}{2}, \tag{2.9}$$

a LRD do:

$$\frac{1}{2} < H \leq 1. \tag{2.10}$$

Z hlediska analytických problému znamená oblast SRD daleko menší problém, protože autokorelace se rozkládají dostatečně rychle s agregací. Nás proto zajímá oblast LRD a zejména její matematické vyjádření. Vyjádření autokovarianční [11] funkce pro $\frac{1}{2} < H \leq 1$:

$$\gamma(k) = \frac{\sigma^2}{2}\left[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}\right] \; pro\; všechny\; k \geq 1. \tag{2.11}$$

Pomocí této funkce můžeme definovat autokorelační funkci:

$$r(k) = \frac{\gamma(k)}{\sigma^2}, \tag{2.12}$$

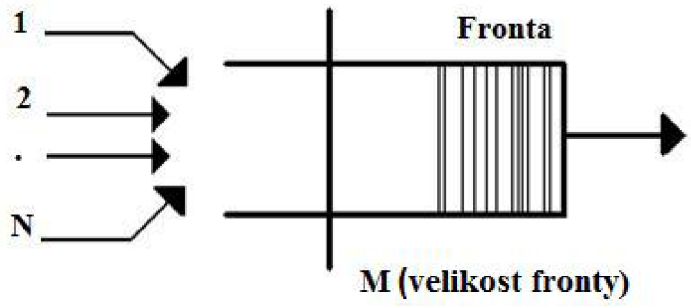
potom pro $\frac{1}{2} < H \leq 1$ platí:

$$\sum_{k=-\infty}^{\infty} r(k) = \infty [13]. \tag{2.13}$$

2.2.2 ON-OFF model

Model popisuje komunikaci mezi linkovou úrovní a aplikační úrovní. Tento model se užívá, především pokud je nutné zachytit měřítko chování síťového provozu. Například pro analýzu struktury IP provozu používáme především ON-OFF model. ON-OFF model používá pouze dva stavy, pojmenované příznačně ON a OFF. Čas strávený mezi ON a OFF stavem se nazývá přechodový čas (transition time) a je většinou popisován exponenciální distribucí [1]. K pochopení ON-OFF modelu uvažujeme fronty v síti sdílené N různými ON-OFF zdroji, ukázané na obr: 2.5. Pro použití ON-OFF modelu v tomto scénáři, je nutné, aby zdroje byli statisticky identické a nezávislé.

Fronta o velikost M je naplňován konstantní rychlostí C od zdroje. Zdroj ON je charakterizován parametrem L. Tento parametr „značí“ průměrný počet paketů generovaných během ON stavu, maximální rychlostí S když zdroj je ve stavu ON a r značí průměrnou rychlost zdroje. Tyto faktory určují průměrné trvání ON a OFF period zdroje. Rovnováha pravděpodobnosti ON fáze zdroje se vypočítá jako:



Obr. 2.5: Analýza fronty v modelu ON-OFF

$$\gamma = \frac{r}{S} \tag{2.14}$$

S exponenciálně distribuovanýma periodami může být zdroj modelován dvoustavovým Markovovým řetězem. Průměrný počet generovaných paketů je považována striktně za větší než <1,L>. Přechod zdroje ze stavu OFF do stavu ON a naopak se vypočítá jako:

- přechod ze stavu $OFF \rightarrow ON$ jako:

$$\gamma \cdot S/(L \cdot (1-\gamma)), \tag{2.15}$$

- a přechod ze stavu $ON \rightarrow OFF$ jako:

$$S/L \tag{2.16}$$

2.2.3 Dopad TCP

První modely provozu byly vyvinuty před tím, než bylo nasazeno řízení přetížení na protokolu TCP, které situaci dramaticky změnilo. Bylo to způsobeno tím, že toky začali být zcela nezávislé a začali reagovat stejnými mechanismy a to způsobilo přetížení sítě. Tato síťová dynamika zapůsobila na sobě-podobné procesy, u kterých výrazně ovlivnila výsledné modely. Sobě-podobné modely pracují jako otevřená smyčka (open loop) to má za následek, že zde není žádná zpětná vazba. U TCP je předpokládáno řízení zahlcení a proto potřebuje zpětnou vazbu, aby protokol TCP dokázal tuto vlastnost ovládat. Definice sobě-podobnosti je asymptotická[3] . Takže

[3]Operační náročnost algoritmu, zjišťuje jakým způsobem se bude chovaní algoritmu měnit v závislostí na změně velikosti (počtu) vstupních dat

zatímco toto platí ve velkých časových měřítkách, v menších měřítkách může být korelace odlišná. Toto tvrzení dokazuje, že v menším měřítku může být viděn dopad TCP [5]. Studie dokazuje, že řízení přetížení TCP a sobě-podobnost mohou koexistovat společně, takže TCP řízení nezpůsobuje sobě-podobnost v síti, ale ani ji nemůže odstranit nebo zmírnit [13].Proto je důležité řídit se při výběru modelu těmito podmínkami:

- řízení sítě by nemělo mít významný dopad na sobě-podobné toky na měřítku oblasti, kde vyšetřujeme důkaz sobě-podobnosti,
- měly by být ve velkém měřítku k agregaci síťového provozu, jinak marginální rozdělené počtů nebude Gaussian,
- dlouhodobá závislosti by měla mít vliv na změnu velikosti oblasti, který je předmětem vyšetřování i pro extrémně krátké vzorky, kde jsou Poisson modely stejně přesné.

2.2.4 Frakční brownův pohyb

Při představení prvních sobě-podobných modelů, nebyly tyto modely příliš efektivní a analyticky špatně ovladatelné pro generování provozu sobě-podobných modelů. Ilka Norros proto vymyslel stochastický proces pro uchování modelu se sobě-podobným vstupem a konstantní bitovou rychlostí na výstupu [10]. Avšak tento první model byl spíše spojitý než diskrétní. Na druhou stranu, byl efektivní a jednoduchý. Než se, ale začneme zabývat Norrosovým stochastickým procesem musíme definovat samotný frakční Brownův pohyb (fBp). FBp je časově spojitý Gaussovský proces, který je definován jako:

$$f(x) = ae^{\frac{(x-b)^2}{2c^2}} \tag{2.17}$$

Kde čísla a a c jsou kladná reálná čísla, μ je libovolné reálné číslo a e je Eulerovo číslo (2,71828...). Graf takovéto funkce má potom v bodě x vrchol μ o výšce a, který dělí graf na dvě souměrné části. Parametr c určuje šířku kopce, ve výšce:

$$ae^{-1/8} \approx 0,8825a. \tag{2.18}$$

Tento proces je platný pro všechny kladné časové hodnoty, s průměrem centrovaným k 0 a autokorelací definovanou na Hurstově parametru jako:

$$\gamma(t,s) = \frac{1}{2}\left(|t|^H + |s|^{2H} - |t-s|^H\right) \tag{2.19}$$

Matematicky je Norros proces zastoupen takto:

$$V(t) = \sup_{s \leq t}(A(t) - A(s) - C(t-s)), t \in (-\infty, \infty) \tag{2.20}$$

Kde funkce supremum (sup) je matematický pojem z oboru teorie uspořádání a nejčastěji se používá ke zkoumání vlastností reálných čísel. Je to alternativa k pojmu největší prvek neboli maximum. Avšak oproti maximu je supremum dohledatelné u více množin. Supremum má tedy každá shora omezená množina reálných čísel. V našem případě supremum s je menší nebo rovno t, protože pro t podle definice suprema neexistuje. *A(t)* je proces definován jako:

$$A(t) = mt + \sqrt{amZ(t)}, \tag{2.21}$$

kde *Z(t)* je normalizovaný fBp s parametrem Hurst $\left(\frac{1}{2}, 1\right]$. Parametry procesu jsou m tedy průměrná vstupní rychlost, a je rozdíl koeficientů, H je parametr Hurst a C je rychlost služby. Tímto Norros demonstroval analytické řešení pro řadu dříve složitých problémů. Tímto způsobem dokonce analyticky vyřešil problém s plánováním kapacity, tedy jak velký prostor musí být přidělen k dosažení dané kvality služeb. Aby tento a další modely fungovaly v sobě-podobném toku, je nutné vhodně určit parametr Hurst. To se v praxi ovšem ukázalo jako velký problém. A proto výrazně omezuje využití Norrosovy práce.

2.2.5 Chaotické mapy

Dopad síťové dynamiky na sobě - podobné modely je popsán v [5]. Poukazují na to, že řízení přetížení TCP má vliv pouze na menším časovém měřítku respektive maximálně na úrovni RTT . V tomto měřítku je nicméně dopad obrovský. Aby se toto upravilo, byl navržen model chaotických map zpětně vazby na TCP. Chaotickou mapu si lze představit jako formu kontinuálního stavu Markova řetězu. Stavovou proměnnou x lze definovat vztahem:

$$f(x) = \begin{cases} f_1(x_n), & x_n > d \\ f_2(x_n), & x_n < d \end{cases} \tag{2.22}$$

Kde x je [0,1]a d představuje stavové hranice. Pro účely modelování ON/OFF zdrojů sítě je návrh, že když x bude větší než d, je zdroj ve stavu ON a když bude x menší než d, je zdroj ve stavu OFF. K použití chaotických map zpětných mechanizmu TCP, použijeme dvojici chaotických map, jednu pro velikost okna TCP a druhou pro stav ON/OFF. Musíme stanovit délku výchozího stavu ON a až bude

stav ON, zdroj odešle celou velikost TCP okna. Za těchto předpokladů můžeme definovat sadu pevně spojených funkcí, které popisují vývoj TCP okna a zdroje stavů ON/OFF. Tento model má ale svá omezení, nezpracovává přenos ztracených paktů a nedokáže odesílat zprávu o vypršení časového limitu, aby byl vykonán opakovaný přenos (Timeout).

2.2.6 Autoregresní model

Tento model patří do skupiny lineárních modelů, používajících se k specifikaci stacionárních stochastických procesů. Autoregresní model se snaží předpovědět aktuální výstup systému označený jako y_n, který je závislý na předchozích výstupech systému označené jako y_k, kde $k < n$ a na vstupech označených jako x_n a x_k, kde $k < n$. Potom můžeme označit autoregresní model řádu p, který označujeme jako AR (p), který má následující tvar:

$$X_t = R_1 \cdot X_{t-1} + R_2 \cdot X_{t-2} + \ldots + R_p \cdot X_{t-p} + W_p \tag{2.23}$$

, kde W_t je bílý šum, R_i jsou reálná čísla a X_t jsou předepsané korelace náhodných čísel. Autokorelací funkce *AR(p)* procesu se skládá z tlumených sinusových vln v závislosti na tom, zda kořeny jsou reálné nebo imaginární. Autoregresní modely patří do velké skupiny modelů a jen pro doplnění, například vektorový autoregresní model (VAR (p)), se používá ke statistické analýze (finanční modely, chováni na akciových trzích, apod.).

2.2.7 SWING

Model používaný pro analýzu síťového provozu, ale také pro jeho generaci. Patří také mezi nejnovější modely. Tento model oproti jiným, které musely mít pro analýzu mnoho někdy až tisíc pozorování, používá tento model velmi jednoduchou analýzu. Zkoumá charakteristiky uživatelů, výměny typu požadavek/odpověď (RREs), spojení, jednotlivé pakety a síť jako celek. Sobě-podobnost je zde vytvářena přirozeně s agregací z mnoha ON/OFF zdrojů. Model SWING můžeme charakterizovat pomocí čtyř parametrů:

- Uživatel - určuje charakter komunikace aplikací, dobu aktivity (požadavky).
- Relace - vyhledávaní webových stránek čí stahování souborů.
- Spojení - vlastnosti připojení v rámci jedné relace jako cíl, počet požadavků a výměn a další.
- Síťová charakteristika - obsahuje ztrátovost, kapacitu, latenci a další.

Vhodnost tohoto modelu ještě podtrhuje, že je schopen generovat shlukový provoz v širokém časovém měřítku.

2.2.8 Stochastický model

Stochastický model vyjadřuje provoz novým způsobem. Oproti sobě-nepodobným modelům, kdy nám stačilo určit pouze čas příchodu nebo velikost paketu, nebo naopak u sobě-podobných modelu, kde jsem musely určit parametr Hurst a mnoho dalších z nichž, některé jsou typické pouze pro daný model. Stochastický model vyjadřuje provoz na síti jako soubor nezávislých TCP spojení každé toto spojení je charakterizováno proměnnýma jako jsou: příchozí čas požadavku, RTT klienta, RTT serveru, počet výměn požadavek/odpověď (request/response), časové prodlevy mezi výměnami, velikost jednotlivých požadavků, velikost jednotlivých odpovědí a zpoždění serveru. Pro pochopení stochastického modelu musíme pochopit nejprve, jak funguje vrstva TCP, na které je model modelován. Předpokládá se, že provoz na síti používá kombinaci protokolu HTTP 1.0 a 1.1. Přesněji řečeno se předpokládá, že některé spojení bude použito pouze pro jednu výměnu typu požadavek/odpověď („krátkodobé“ připojení HTTP) a některé TCP spojení bude použito pro více výměn typu požadavek/odpověď („trvalé“ připojení HTTP). Toto vše vychází z principu funkčnosti protokolu HTTP 1.0 a 1.1, které je uvedeno výše 1.1. Na obr.1[doplním obrázek] vidíme ukázku jednoduché architektury modelu, kde je modelována síť s jedním spojením. Tento příklad nám ukazuje realistické připojení například sítě k internetu. Základním parametrem takové sítě je její rychlost, s jakou jsou sestavována nová TCP spojení od klientů.

Zdrojové proměnné

Proměnné ve stochastickém modelu popisují informace o HTTP přenosech. Každé TCP spojení obsahuje jednu nebo více výměn typu požadavek/odpověď mezi klientem a serverem. Každá taková výměna obsahuje požadavky odeslané od klienta serveru a odpovědi odeslané od serveru klientovi. K řízení HTTP proudu a tím i jeho modelování potřebujeme tyto proměnné:

- t_i - čas mezi spojeními respektive čas mezi spojením i a $i+1$
- R_i - RTT mezi serverem a bodem začátku/konce modelování sítě
- r_i - RTT mezi klientem a bodem začátku/konce modelování sítě
- B_i - rychlost připojení serveru
- b_i - rychlost připojení klienta
- L_i - pravděpodobnost ztráty paketu na straně serveru
- l_i - pravděpodobnost ztráty paketu na straně klienta
- p_i - počet požadovaných stránek
- $m_{i,j}$ => j=1,...,p_i - počet výměn (požadavek/odpověď) pro stránku j s připojení i
- n_i - konečný počet výměn (požadavek/odpověď) $n_i = m_{i,j} + \cdots + m_i, p_i$

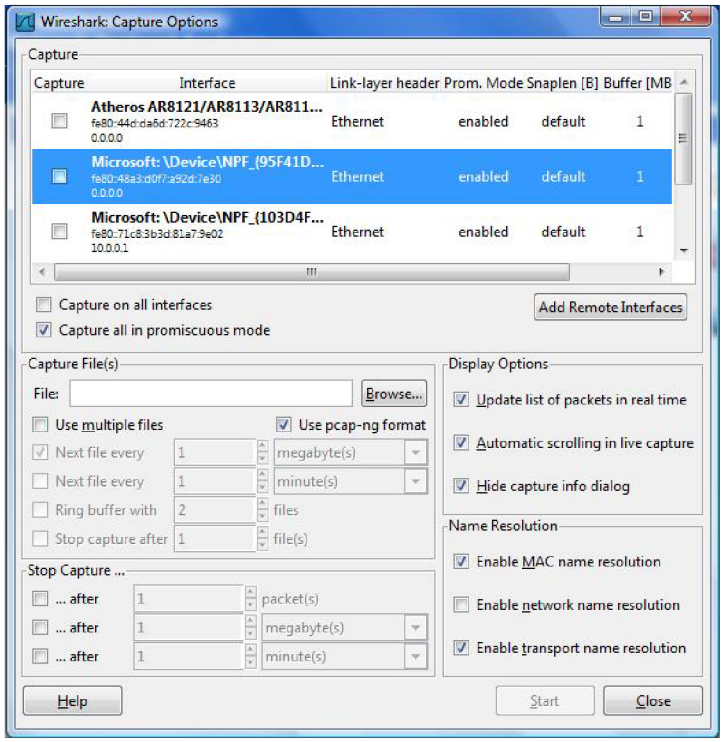
- $F_{i,l}$ => l=1,. . . ,n_i - velikost souboru požadovaného serverem
- $f_{i,l}$ => l=1,. . . ,n_i - velikost souboru požadovaného klientem
- $g_{i,j}$ => j=1,. . . ,$p_i - 1$:pro $p_i > 1$ - prodleva mezi stránkami. Čas mezi tím, kdy dojde poslední paket odpovědí ke klientovy stránky j a první paket žádosti stránky *j+1*
- $g_{i,j,k}$ => k=1,. . . ,$m_{i,j} - 1$:pro $m_{i,j} > 1$ - prodleva mezi výměnami. Čas mezi tím, kdy ke klientovy příjde poslední paket s odpovědí souboru k a pohotovostí klienta na požadavek prvního paketu souboru *k+1*.

3 VÝSLEDKY BAKALÁŘSKÉ PRÁCE

3.1 Výsledky

3.1.1 Nastavení programu Wireshark

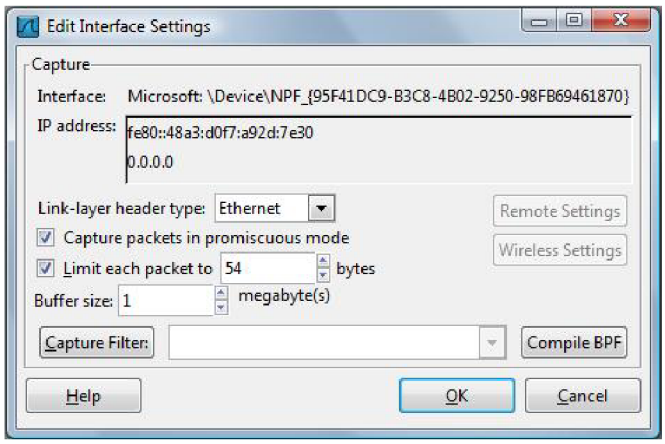
Pro řešení zachytávání provozu na protokolu HTTP jsem využil program Wireshark verze 1.7.0. Program byl instalován na přenosný počítač a k síti se připojoval bezdrátově přes směrovač Zyxel P-600. Tento program obsahuje mnoho výtečných funkcí k analýze sítě. Nejprve musíme provést nastavení programu pro náš konkrétní případ. Jelikož modelujeme protokol HTTP a službu WWW potřebujeme nastavit příslušný filtr. Tento filtr nastavíme tak, že v liště programu vybereme: *Capture>Options* 3.1 a otevře se nám okno,



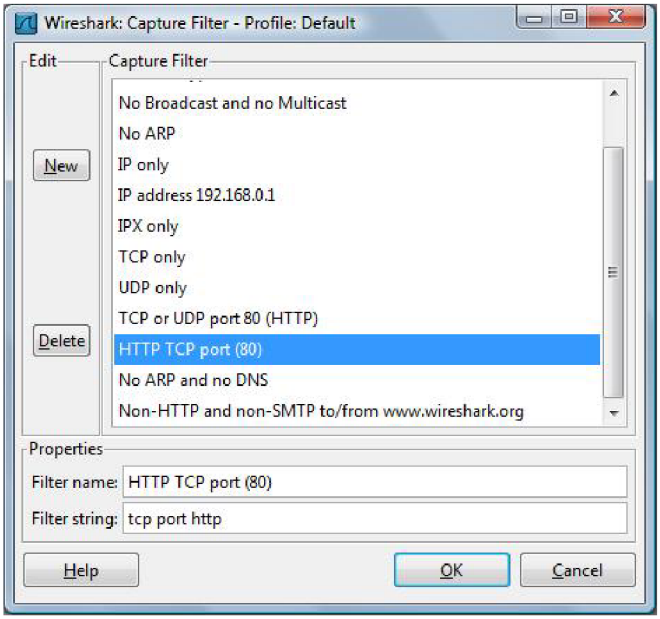
Obr. 3.1: Okno výběru rozhraní

kde dvojklikem vybereme příslušné rozhraní, pokud máme více možností připojení. Otevře se okno *Edit Interface Settings* 3.2, kde v položce *Capture Filter* 3.3, vybereme vhodný filtr.

V našem případě se jedná o filtr: *tcp port http*. Tento filtr potvrdíme a vrátíme se zpět do okna *Edit Interface Settings* 3.2, kde nastavíme položku *Limit each paket to* na hodnotu *54 bytes*. Tato položka se nastavuje, aby byly zachytávány pouze hlavičky



Obr. 3.2: Okno nastavení rozhraní



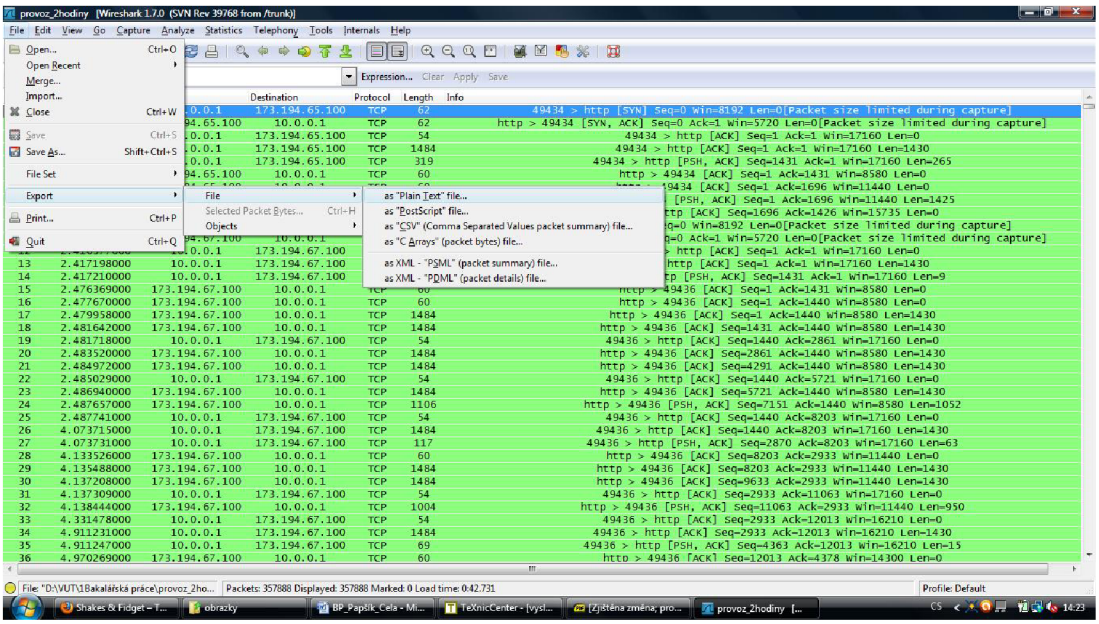
Obr. 3.3: Okno s přednastavenými filtry

paketů, které jsou pro nás důležité a nezaplňovaly místo na disku „zbytečnými“ daty. Pro práci se zachycenými daty, můžeme tyto data exportovat 3.4,

jako textový soubor *„CSV“* (*Coma Separated Value*) a tento soubor můžeme otevřít například programem *Microsoft Excel*.

3.1.2 Zpracování provozu

Program Wireshark zachycuje „7“ základních parametrů. Tyto parametry jsou zobrazeny ve sloupcích. Na řádcích níže bude vysvětleno, co jednotlivé parametry zna-



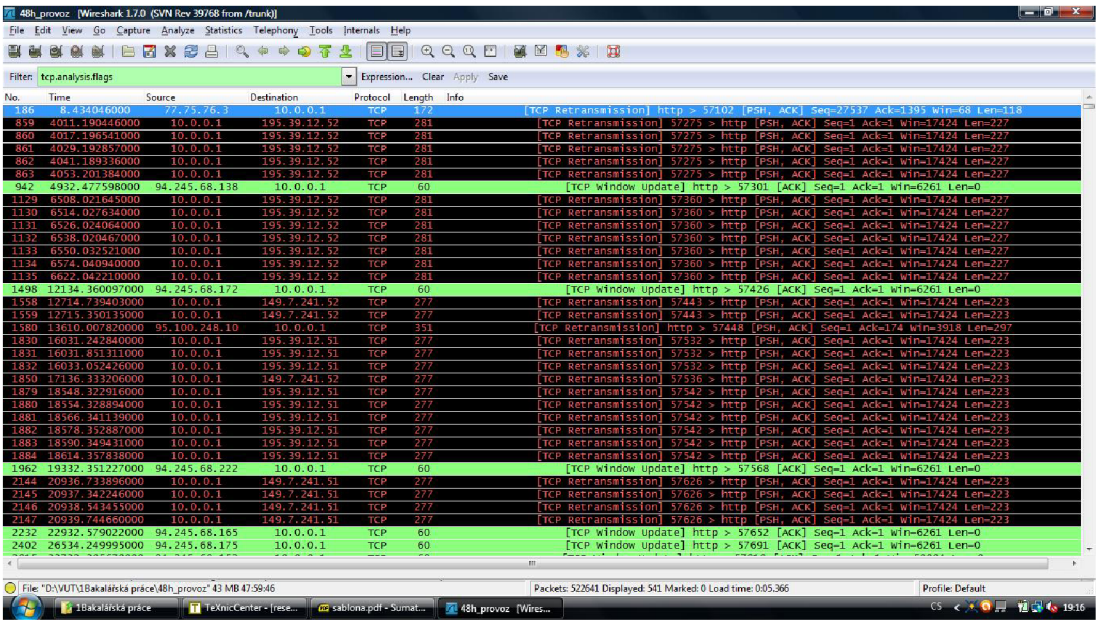
Obr. 3.4: Ukázka exportu dat

menají:

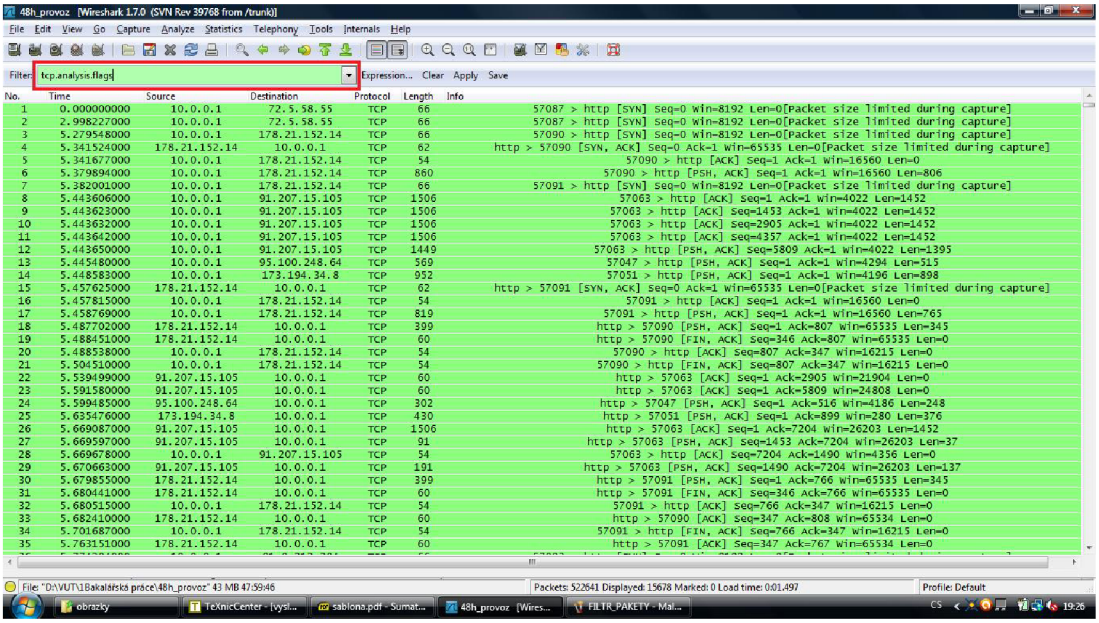
- *No* - (číslo paketu) značí pořadí zachyceného paketu,
- *Time* - (čas) značí v jakém čase od začátku, byl paket zachycen,
- *Source* - (zdroj) udává zdrojovou IP adresu paketu,
- *Destination* - (cíl) udává cílovou IP adresu paketu,
- *Protocol* - (protokol) udává, na jakém protokolu paket pracuje,
- *Lenght* - (délka) udává velikost paketu v bytech,
- *Info* - (informace) tento údaj obsahuje další informace o paketu jako port, Seq, Ack a další informace, které se liší v závislosti na paketu.

V zachyceném provozu, se mohou objevovat nežádoucí pakety, které negativně ovlivňují analýzu. Tyto pakety mohou vzniknout duplikací ACK, špatným kontrolním součtem, chybou spojení a dalšími vlivy. Takové pakety jsou označeny ve Wiresharku černou barvou 3.5. Jak je uvedeno výše je důležité tyto pakety odfiltrovat. To se provede následujícím způsobem: na obrazovce programu Wireshark je okno, kde můžeme zadávat filtry 3.6, vyznačeno červeným rámečkem, do tohoto okna vložíme filtr *tcp.analysis.flags*, po zadáni tohoto filtru se zobrazí pouze chybové (nežádoucí) pakety.

Poté v okně klikneme na *Edit > Ignor all displayed packet*. Po zkušenostech je nutné tuto proceduru několikrát opakovat, aby byly všechny nežádoucí pakety odstraněny. Po odstraněný těchto paketů zrušíme zadaný filtr *tcp.analysis.flags* a zůstanou nám jen „potřebné“ pakety. Po odstranění paketů jsem rozdělil pro snadnější



Obr. 3.5: Ukázka „nežádoucích“ paketů



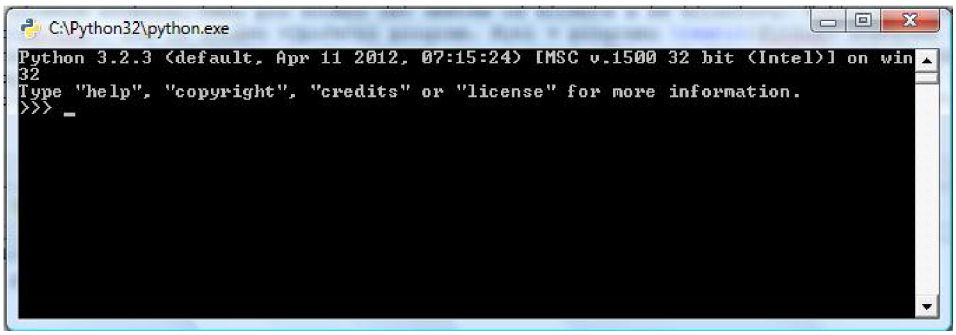
Obr. 3.6: Ukázka vložení filtru

analýzu provoz do dvou směrů. Do směru, ve kterém jdou všechny pakety od klienta (*10.0.0.1*) na různé cílové IP adresy a na pakety, které přicházejí z různých IP adres ke klientovy (*10.0.0.1*). Z tohoto popisu je zřejmé, že IP adresa klienta je *10.0.0.1*. Pro rozdělení aplikujeme v okně, kde jsem již zadaly filtr na zobrazení „nežádoucích“ paketů, filtr *ip.src==10.0.0.1*, tento filtr způsobí, že budeme mít provoz jen jedním

směrem a to od klienta na servery. Aplikováním filtru *ip.dst==10.0.0.1* v tom samém okně, dostaneme provoz ze serveru na klienta. Nyní můžeme jednotlivé směry uložit jak je uvedené výše jako *csv*. Pro základní analýzu musíme takto rozdělené směry opět rozdělit,ale již na 30-ti sekundové intervaly v jednotlivých směrech. V každém tomto intervalu musíme spočítat tyto parametry:

- počet paketů směrem od klienta i ke klientovy,
- počet bajtů směrem od klienta i ke klientovy,
- počet spojení během intervalu,
- průměrná doba trvání spojení během intervalu.

Provedeme to tak, že soubor uložený jako *.csv* otevřeme v programu *Microsoft Excel*,kde se nám 7 základních parametrů zobrazí odděleně čárkou. Poté je nezbytně nutné uložit takto vytvořený „excelovský“ soubor znovu jako *.csv*. Tuto proceduru musíme provést pro oba uložené soubory, tedy pro soubor dat směrem od klienta a ke klientovy. Takto upravené soubory jsou přeneseny do složky programu *Python*, kde je uložen výpočetní program. Nyní v programu *Python* otevřeme aplikaci *python* 3.7.



Obr. 3.7: Spuštěná aplikace Python

Po kliknutí se nám zobrazí okno podobné příkazové řádce, do kterého zadáme příkaz *python název_výpočetního_programu.py název_upraveneho_souboru.csv > výstupní_soubor.csv*. V našem případě je název výpočetního programu *markov* [9]. Výstupní soubor se uloží jako *.csv*, ten opět otevřeme v programu *Microsoft Excel*. V tomto souboru jsou již data rozdělena do 30-ti sekundových intervalů. Obsahuje 5 sloupců:

- interval,
- počet paketů,
- velikost paketů,
- počet spojení,
- průměrná délka spojení.

Program Python pracuje se sedmi parametry Wiresharku. První je rozdělení na intervaly. Je využit parametr čas, kdy po překročení časového intervalu 30 sekund se v souboru vytvoří další řádek. Počet paketů je dán jako součet řádků patřících do intervalu. Každý řádek představuje jeden paket. Velikost paketu zjistíme opět jednoduše a to součtem řádku *Lenght* patřících do intervalu. Tyto kroky jsou stejné pro oba směry jak pro směr od klienta, tak ke klientovy. Počet spojení se vypočítá tak, že pokud se u paketu objeví příznak [SYN] nebo [SYN,ACK] (tento příznak se objeví ve Wiresharku ve sloupci *Info*), začne se počítat nové spojení. Ze směru na klienta se počítá příznak [SYN,ACK] a pro směr od klienta příznak [SYN]. Zároveň je nutné zaznamenat číslo portu, které opět zjistíme ve sloupci *Info* a dále je nutné zaznamenat čas. Pro průměrnou délku trvání spojení program Python vyhledává ve sloupci informace údaj [FIN, ACK] a pomocí čísla portů přiřadí spojení k údaji [SYN, ACK] či [SYN] se shodným číslem portu. Díky rozdílu času u údaje [SYN, ACK] či [SYN] a u údaje času [FIN, ACK] je zjištěna délka trvání spojení. Poté je vypočítán průměr. Během výpočtu se běžně stává, že spojení přesáhne hned několik intervalů. Pokud nastane tento případ, zaznamená se spojení i jeho délka do toho intervalu, ve kterém bylo spojení ukončeno. Výpočetní program respektive jeho kód se nachází v příloze.

3.1.3 Zachycený provoz

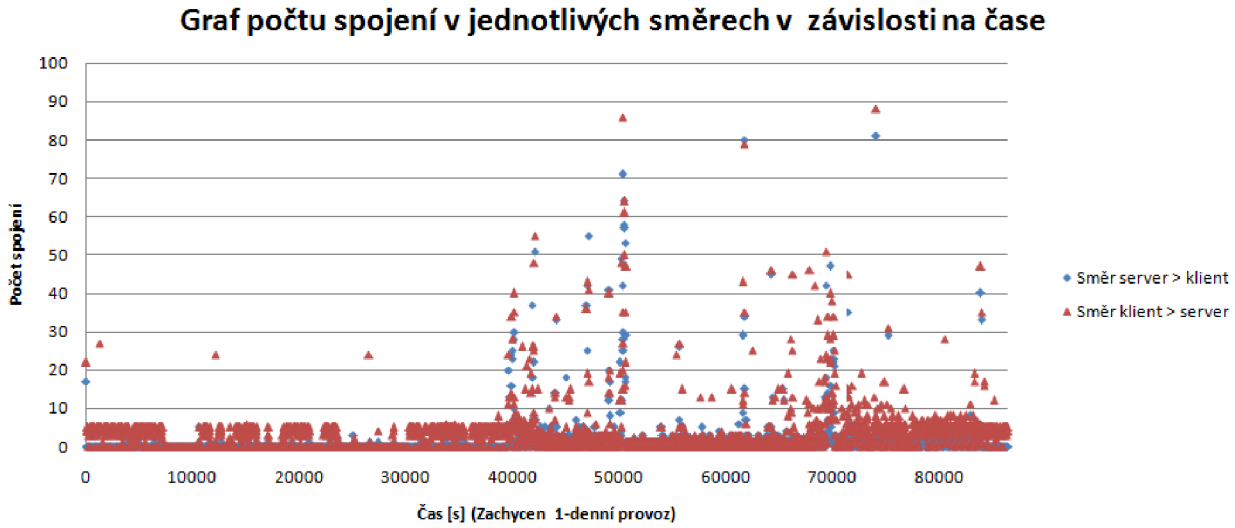
Úkolem bylo zachytit dlouhodobý provoz, v mém případě byl zachycen 48hodinový provoz a zpracován byl provoz 24hodinový. Jedná se o střední až mírný provoz během celého dne. V tomto zpracovaném provozu bylo zachyceno:

- 104405 paketů z toho ve směru klient → server 54904 paketů a ze směru server → klient 49501 paketů,
- 13122 spojení z toho ve směru klient → server 3699 spojení a ze směru server → klient 9423 spojení,
- 57,210 *MiB*[1] dat z toho ve směru klient→ server 45,470 *Mib* dat a ze směru server → klient 11,740 *MiB* dat,
- průměrná doba spojení ve směru klient→ server byla 7,73 *s* a ve směru server → klient 5,83 *s*.

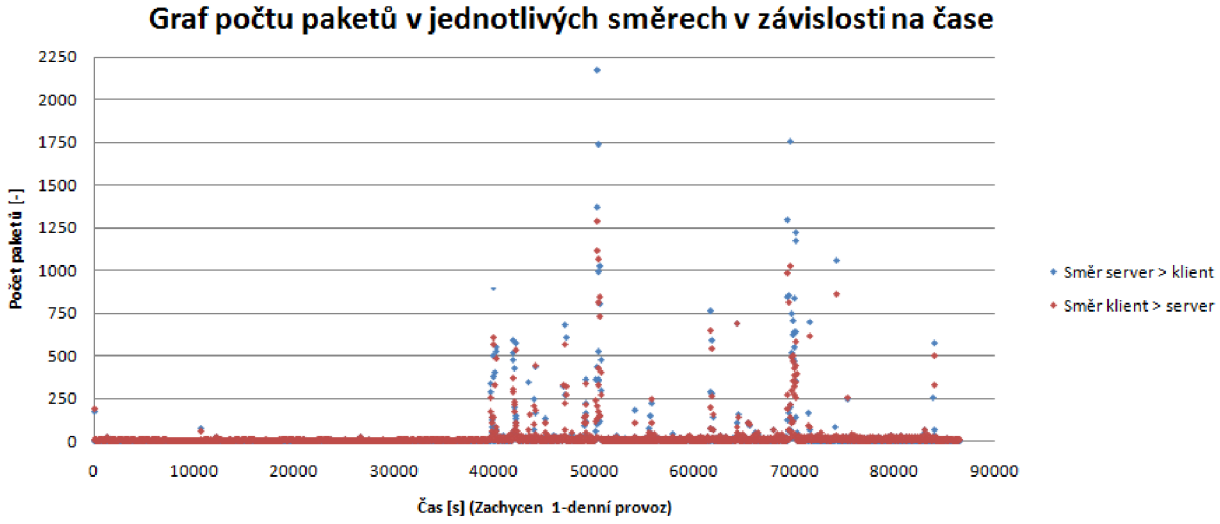
Na následujících obrázcích můžeme vidět počet spojení v jednotlivých směrech v závislosti na čase 3.8 a na počtu paketu přenesených v jednotlivých směrech v závislosti na čase 3.9.

Z vyobrazených grafu je patrné, že většina spojení nepřesáhla *10 s* hranici a počet paketů v obou směrech se pohyboval kolem 50 paketů v každém 30-ti sekundovém intervalu. Celkově je vidět, že 1-denní provoz vykazuje nárazovou charakteristiku. Je

[1] Mebibajt



Obr. 3.8: Graf počtu spojení



Obr. 3.9: Graf počtu paketů

to dáno obecnou charakteristikou domácího uživatele, který je přes den mimo svůj domov a když se vrací zpět, kontroluje e-maily, brouzdá na internetu či stahuje. Avšak i když není doma komunikace mezi serverem a klientem stále pokračuje. Vysílají se pakety a znovu navázání spojení či restart spojení.

3.1.4 Modelování Markovsky-modulovaným Poissonovským modelem

Pro modelování provozu Markovským modelem jsem si vybral závislost počtu paketů na čase. U této metody výpočtu je jedno jestli komunikace probíhá ze směru → server klient nebo naopak. Na předchozím obrázku 3.9, můžeme tuto závislost vidět. Nyní tedy zbývá vytvořit model, který bude umět tuto závislost předpovídat. Předpověď počtu paketů v závislosti na čase získáme pomocí vzorce:

$$p_n(t) = \frac{(\lambda \cdot t)^n}{n!} \cdot e^{-\lambda \cdot t}. \tag{3.1}$$

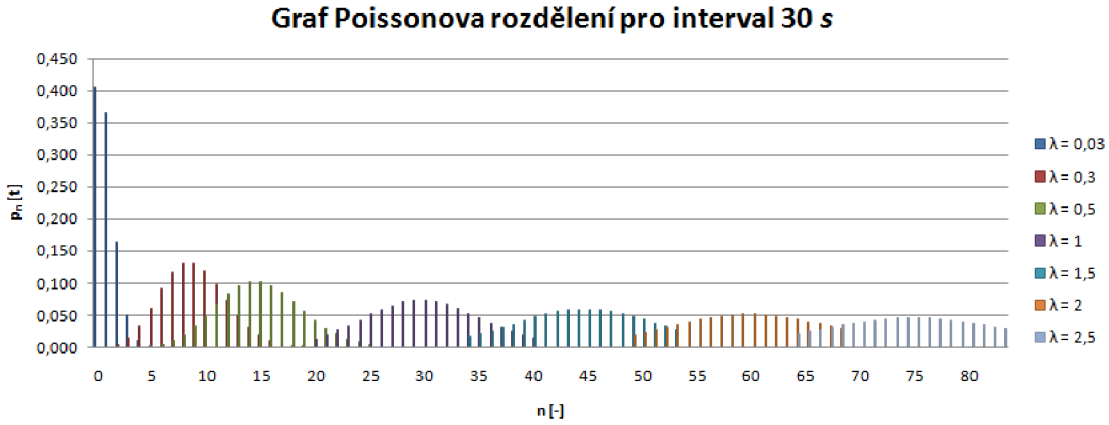
Kde $n = 0, 1, 2 \ldots n$, což značí vztah pro Poissonovo rozdělení. Ve vztahu vyjadřuje n počet paketů, t časový interval, ve kterém byly pakety zachyceny, λ vyjadřuje intenzitu provozu získanou ze vztahu $\frac{n}{t}$ a $p_n(t)$ vyjadřuje pravděpodobnost počtu paketů v daném časovém intervalu [8]. Ze získaného provozu jsme vypočetli λ neboli intenzitu pomocí vzorce:

$$\lambda = \frac{n}{t}, \tag{3.2}$$

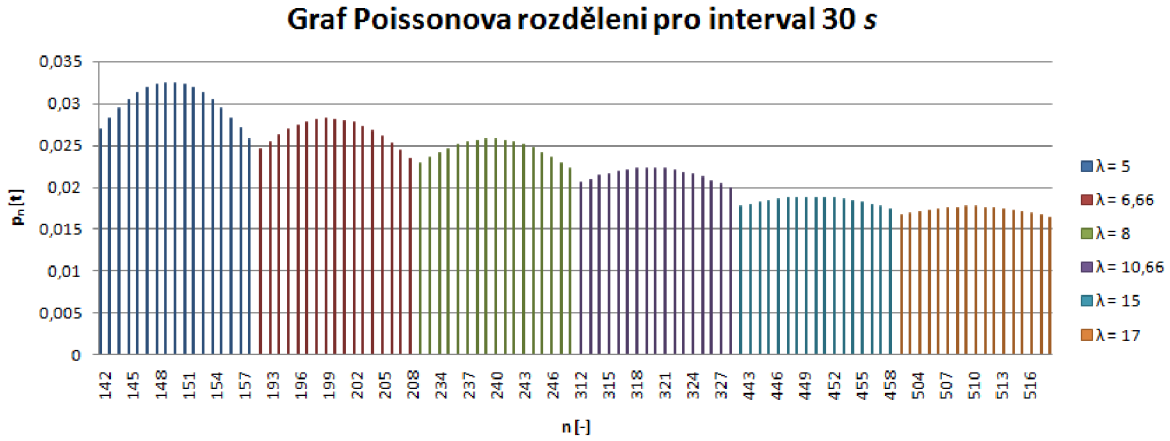
poté jsme z této intenzity vypočetli předpověď intenzity jako průměr pěti předchozích hodnot. Tyto předpovědi byly dosazeny do 3.1 zároveň k tomu byl dosažen skutečný počet paketů n, který odpovídal danému časovému intervalu. Takto vypočítaná pravděpodobnost $p_n(t)$ byla porovnána s modelovanými hodnotami $p_n(t)$ a tak byla zjištěna modelovaná hodnota n. Modelovanou hodnotu $p_n(t)$ zjistíme pomocí Poissonova rozdělení pro jednotlivé λ. Pro dostatečné přesný model musíme vytvořit několik funkcí λ tak abychom dostatečně pokryly definiční obor, který zároveň vyjadřuje počet paketů. Pro každou takto zvolenou intenzitu λ musíme zvolit minimálně 15 hodnot pro n, které pro danou intenzitu vyjadřují největší pravděpodobnost. Na následujících grafech 3.10, 3.11 a 3.12 můžeme vidět Poissonova rozdělení pro různé λ.

Když jsme obdrželi všechny proměnné mohli jsme uskutečnit samotné porovnání reálného provozu s provozem modelovaným. Na násleujících grafech vidíme porovnání reálné a modelovaného provozu jak ve směru klient → server 3.13, tak ve směru server → klient 3.15.

Pro směr klient → server, byl model schopny modelovat okolo 35% hodnot. Je zajímávé, že těchto 35% hodnot modeluje ve 2804 intervalech, zbylích 65% hodnot, které model nedokáže modelovat je skryto v 76 intervalech. Je to způsobeno tím, že model je schopen modelovat kontinuální provoz do cca 50-ti paketů, to lze vidět na grafu 3.14. Pokud příjde shluk paketů, model není schopen reagovat a selhává.



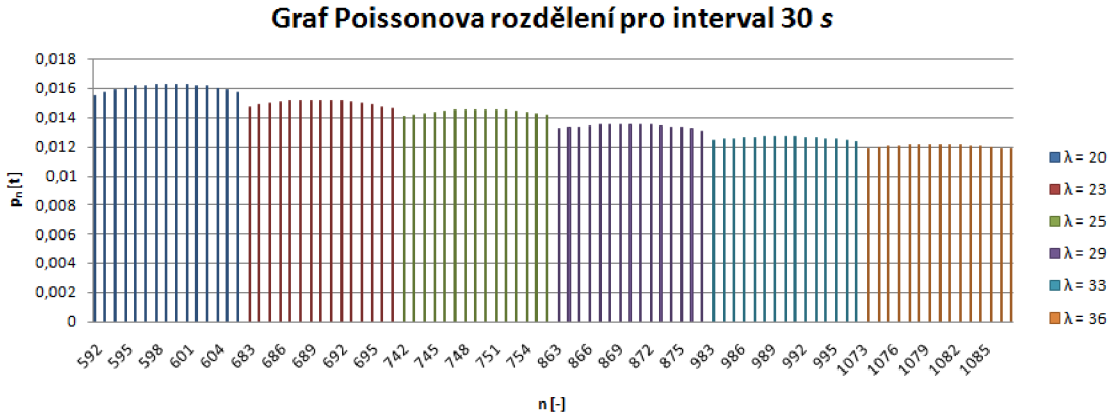
Obr. 3.10: Poissonovo rozdělení pro $t = 30$ s [8]



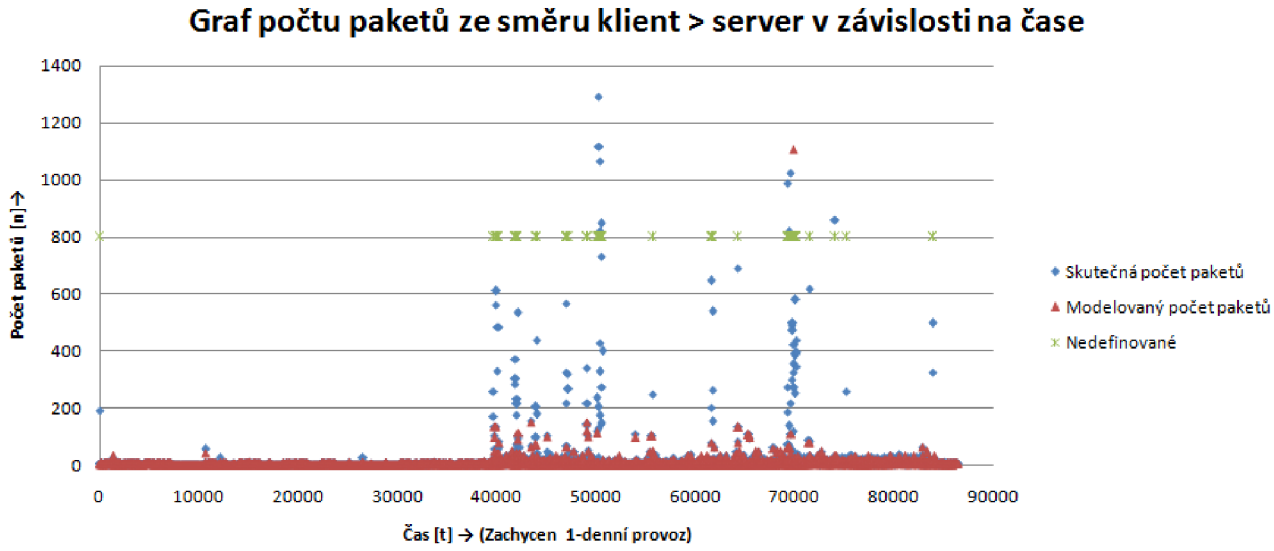
Obr. 3.11: Poissonovo rozdělení pro $t = 30$ s [8]

Za nedefinovanými hodnotami paketu, se skrývá příliš malá hodnota $p_n(t)$. Tato hodnota je tak malá, že nelze určit počet modelovaných paketu n.

Pro směr server → klient, byl model schopny modelovat okolo 20% hodnot. Je zajímávé, že těchto 20% hodnot modeluje ve 2800 intervalech, zbylích 80% hodnot, které model nedokáže modelovat, je skryto v 80 intervalech. Stejně jako pro směr klient → server, je model schopný modelovat provoz do cca 50-ti paketů, což lze vidět na grafu 3.16.



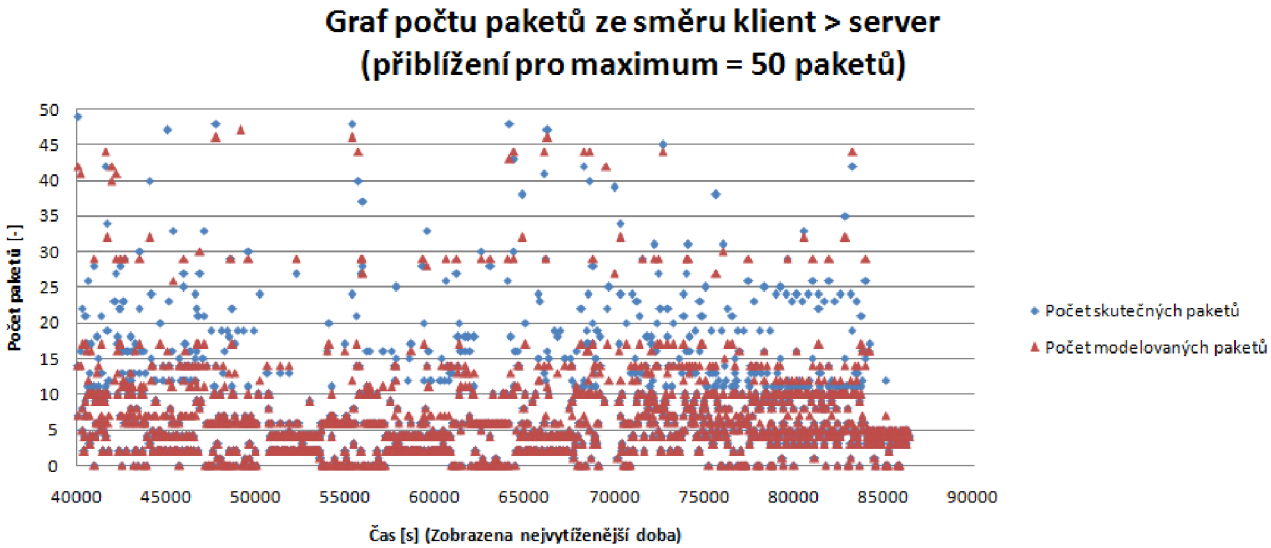
Obr. 3.12: Poissonovo rozdělení pro $t = 30\ s$ [8]



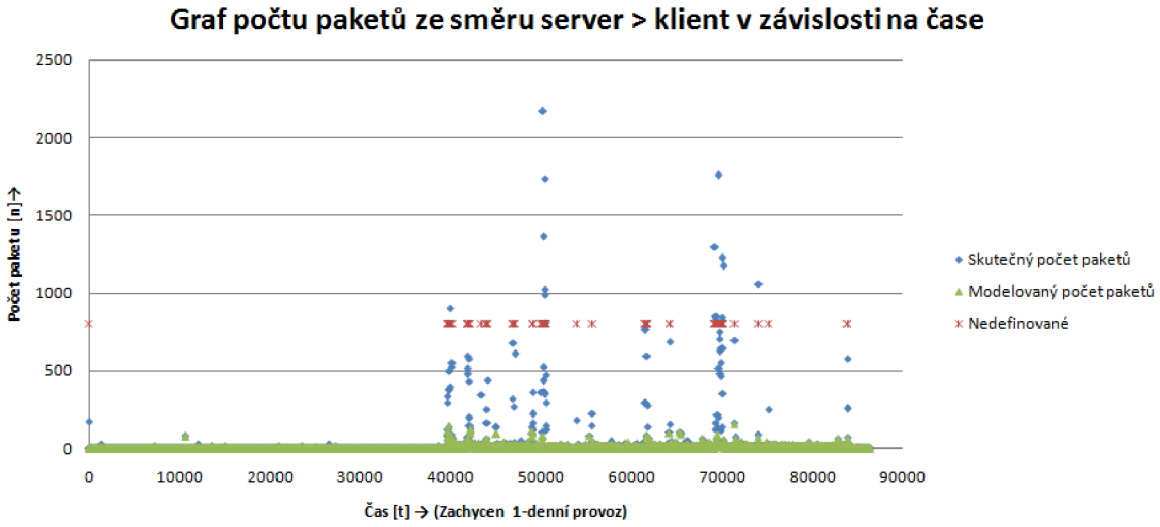
Obr. 3.13: Srovnání reálných a modelovaných hodnot pro směr klient → server

3.1.5 Modelování Stochastickým modelem

Jak je popsáno výše 2.2.8 v kapitole Stochastický model je schopen modelovat několik parametrů. V mé práci se zaměřím na parametr r_i. Tento parametr vyjadřuje čas mezi [SYN,ACK] na straně serveru a [ACK] na straně klienta. To tedy znamená dokončení tzv.: 3-way handshake. Pro zjištění hodnoty r_i využijeme podobný postup jako u základní analýzy. Tentokrát, ale nerozdělíme provoz na jednotlivé směry, ale necháme oba směry v jednom souboru. Postup jak správně uložit zachycený provoz najdeme v kapitole 3.1.2. Pro zjištění potřebných hodnot použijeme opět výpočetní



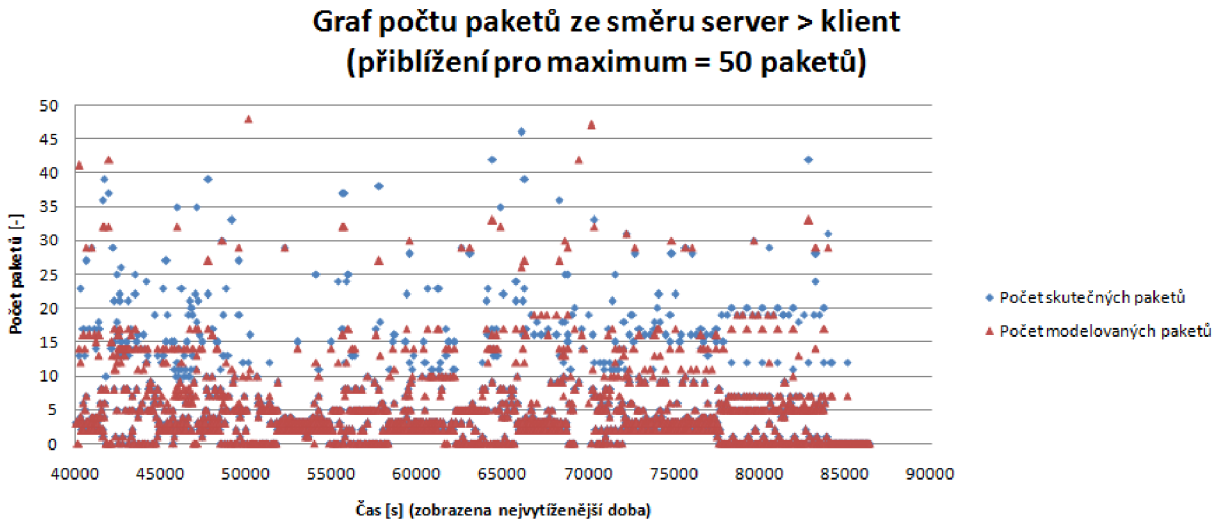
Obr. 3.14: Graf počtů paketu v nejvytíženější době pro maximum 50-ti paketů



Obr. 3.15: Srovnání reálných a modelovaných hodnot pro směr server → klient

program Python, v našem případě je název *stoch* [9], jehož zdrojový kód je v příloze. Návod jak program použít najdeme v 3.1.2.Tento program opět vymezí 30-ti sekundové intervaly, ve kterých budou zjištěny časy r_i a jejich průměry v jednotlivých intervalech. Modelovaná hodnota r_i se vypočítá pomocí Gaussovských časových řad takto:

$$z_i = \sqrt{1-\theta} \cdot s_i + \sqrt{\theta} \cdot n_i[4], \tag{3.3}$$



Obr. 3.16: Graf počtů paketu v nejvytíženější době pro maximum 50-ti paketů

kde s_i je frakční ARIMA proměnlivá podle parametru d, n_i je Gaussovský bílý šum a parametr θ je pomocný proměnlivý parametr podle parametru p. Parametr $d = 0{,}31$ je zvolen podle práce [4] a dopočítává s_i ze vztahu:

$$s_i = n_i + n_i + 1[2], \tag{3.4}$$

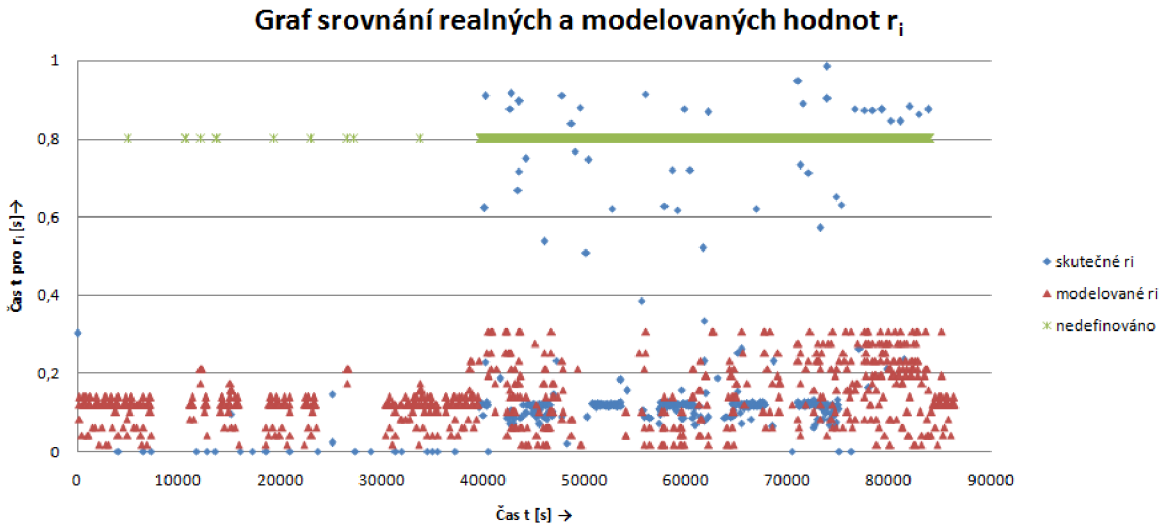
kde n_i je Gaussovský bílý šum, který se vypočítá ze vztahu:

$$n_i = \frac{1-d}{2}[2]. \tag{3.5}$$

Parametr θ se vypočítá ze vztahu:

$$logit2(\theta(\rho)) = -0,445 + 0,554 log_2(\rho)[2]. \tag{3.6}$$

Parametr θ naleží do oboru hodnot $0 \leq \theta \leq 1$. Za ρ se dosazuje hodnota predikce pěti předchozích spojení ze směru server klient [4]. Stochastický model byl schopen modelovat 38% provozu. 70% všech hodnot nesplnilo podmínku $0 \leq \theta \leq 1$ a jsou v grafu značeny jako nedefinované. Z těcho 70% nedefinovaných hodnot bylo 20% hodnot záporných. Záporné hodnoty jsou dány jednoduchým matematickým vyjádřením bílého šumu n_i a z toho plynoucího vyjádření s_i Za neúspěšnost modelu lze přisuzovat příliš mohno spojení ρ. Optimální počet spojení pro modelování se pohybuje v rozmezí 2 až 6. Parametry s_i a n_i mají několik matematických vyjádření a v těchto pracech [4] a [2] se těžko dohledáva přesné dokumenty potřebné pro naše modelování.



Obr. 3.17: Graf srovnání reálného provozu a modelovaného provozu hodnotou r_i

4 ZÁVĚR

V práci jsem zhodnotil získané praktické poznatky ze semestrální práce. Tyto poznatky jsme aplikoval na dva vybrané modely. Na model Stochastický a Markovsky-modulovaný Poissonovský model. Každý z těchto modelů potřeboval pro svoje modelování jiné parametry. Tyto parametry byly získány pomocí dlouhodobého zachycení provozu. Takto získaný provoz bylo nutné upravit, aby ho bylo možné analyzovat pomocí příslušného programu. V zachyceném provozu se projevila typická vlastnost protokolu HTTP a to jeho nárazovost. Jelikož jsem zpracoval jednodenní provoz je nárazovost jasně patrná a to v době, kdy jsem začal aktivně využívat internet. Počet paketů a i spojení prudce vzrostl. Z provozu je vidět, že pokud byl protokol HTTP zatížen mírně počet paketů byl kolem 50 za 30-ti sekundový interval. Avšak pokud byl protokol zatížen více, nebyl problém dosáhnout i 500 paketů a více v jednom intervalu. Přímou úměrou je s tímto faktem spjat i počet spojení.

Při modelovaní Markovsky-modulovaným Poissonvským modelem, bylo zjištěno, že tento model je schopen predikovat 35% provozu ve směru klient → server, respektive 20% v opačné směru. Takto malá procentuální úspěšnost je zapříčiněna tím, že model není schopen reagovat na náhlý příchod většího množství paketů. Číselně model selhává při počtu paketů 150 a více. Pokud je však provoz lineárního charakteru, model je schopen toto chovaní predikovat. Hlavní nevýhodou tohoto modelu je, že patří mezi sobě-nepodobné a tak nedokáže predikovat chování sítě na velkém časovém úseku.

Tento problém, ale již nemá druhý model a to model Stochastický. Model patří mezi sobě-podobné. Model byl schopen modelovat 38% provozu. Takto malá úspěšnost je zapříčiněná tím, že tento model má mnoho parametrů. Některé, jak je uvedeno v textu, je velmi težké přesně dohledat, respektive dohledat přesně jejich matematické vyjádření. Mnoho hodnot také nesplnilo základní podmínku a tak nemohly být použity k modelování.

Celkově jsem si ověřil znalosti, že provoz na protokolu HTTP má nárazový charakter. Tato vlastnost se velmi těžko modeluje. Dle mého názoru pokud by provoz nevykazoval takové nárazy, oba modely by měli daleko větší úspěšnost. V práci jsem si však chtěl vyzkoušet oba extrémy. Takže jak lineární provoz, který byly oba modely schopné modelovat, tak velice nárazový provoz, u kterého Markovsky-modulovaný Poissonovský model selhal a Stochastický, který byl schopen zachytit výkyvy provozu.

LITERATURA

[1] Adas Abdelnaser *Traffic Models in Broadband Networks* IEEE Communications Magazine, Jul. 1997 Dostupné z URL: <http://ieeexplore.ieee.org/ie15/35/13111/00601746.pdf?isnumber=&arnumber=601746

[2] Anderson David, Bowei Xi, Cleveland S. Wiliam *Multifractal and Gaussina Fractional Sum-Difference Models for Internet Traffic* [online] 2011 [cit. 29.4.2012] Dostupné z URL: <http://fodava.gatech.edu/files/reports/FODAVA-10-25.pdf

[3] BOLDIŠ, P. *Bibliografické citace dokumentů podle ČSN ISO 690 a ČSN ISO 690-2* [online]. 2001, poslední aktualizace 11. 11. 2004 [cit. 17. 2. 2005]. Dostupné z URL: <http://www.boldis.cz/citace/citace.html>.

[4] Cao Jin, Cleveland S. Wiliam, Gao Yuan, Jeffay Kevin, Smith F. Donelson, Weigle Michele *Stochastic Models for Generating Synthetic HTTP Source Traffic* [online] 2011 [cit. 2011-11-26] Dostupné z URL: <http://www.stat.purdue.edu/ wsc/papers/packmime.http.pdf

[5] Erramilli Ashok, Roughan Matthew, Veitch Darryl, Willinger Walter *Self-Similar Traffic and Network Dynamics* In proc. of the IEEE., Vol 90, no. 5, 2002 Dostupné z URL: <http://citeseer.ist.pdu.edu/700718.html

[6] R.Jain and S.Routhier *Packet Trains–Measurements and a New Model for Computer Network Traffic* IEEE Journal on Selected Areas in Communications, vol.4, Issue 6, pp.986-995, September 1986 Dostupné z URL: <http://www.cs.wustl.edu/˜jain/papers/train.htm

[7]

[8] Molnár Karol *Moderní síťové technologie* [online] 2011 [cit. 2011-10-29] Dostupné z URL: <http://www.utko.feec.vutbr.cz/ molnar/mmos/fronty.pdf

[9] Miklica Jan *Matematické modelování síťového provozu služby WWW* Bakalářská práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012

[10] Norros Ilkka *A storage model with self-similar input* Queueing Systems, 16:387–396, 1994 Dostupné z URL: <http://citeser.ist.pdu.edu/norros94storage.html

[11] Neznámý autor *Charakteristika náhodné veličiny* [online] poslední aktualizace: 30. 11. 2011 [cit. 10.12.2011] Pod licencí:

<http://creativecommons.org/licenses/by-sa/3.0/ Dostupné z URL: <http://cs.wikipedia.org/wiki/Charakteristika_n%C3%A1hodn%C3%A9_veli%C4%8Diny

[12] Willinger, W *The Discovery of Self-Similar Traffic* In Performance Evalution: Origins and Direction G. Haring, C. Lindermann, and M. Reiser, Eds. Lecture Notes In Computer Science, vol. 1769, Springer-Verlag, London, 513-527 Dostupné z URL: <http://portal.acm.org/citation.cfm?id=647347.724329

[13] Wilson Michael *A historical view of network traffic models* [online] 2006 [cit. 2011-10-20] Dostupné z URL: < http://www.cse.wustl.edu/ jain/cse567-06/ftp/traffic$_m$*odels*2/*index.html*

[14] Zapletal Lukáš *Protokol HTTP 1.1 pod lupou* [online] poslední aktualizace 27. 3. 2001 [cit. 17. 11. 2011] Dostupné z URL: <http://www.root.cz/clanky/protokol-http-1-1-pod-lupou/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

PAN Personal Area Network – Osobní síť

MAN Metropolitan Area Network – Metropolitní síť

WAN Wide Area Network – Rozlehlejší lokální sítě

WLAN Wireless Local Area Network – Bezdrátová lokální síť

LAN Local Area Network – Lokální síť

QoS Quality of Service – Kvalita služeb

TCP Transport Control Protokol – Transportní spojovaný protokol

UDP User Datagram Protokol – Transportní nespojovaný protokol

HTTP Hypertext Transfer Protokol – Protokol určený pro výměnu hypertextových dokumentů

LRD Long-Range Dependence – Dlouhodobá závislost

SRD Short-Range Dependence – Krátkodobá závislost

MiB Mebibajt – tzv.: „Velké kilo“ = 1 048 576 B

SEZNAM PŘÍLOH

A ZDROJOVÝ KÓDY

A.1 Použití kódů

Programy mohou být zapsaný jakýmkoli programovacím interpretem, který podporuje programovací jazyk *Python*. V mém případě bylo použito programu *PSPad* a kód uložen jako soubor s příponou **.py*. Jako interpret byl použit Python v. 3.2, stáhnutelný volně na stránkách www.python.org.

A.2 Zdrojový kód pro základní analýzu

Zdrojový kód programu Python

```
    import sys
import string
from sets import Set
INTERVAL = 30.0
f = open(sys.argv[1], 'r')
#radky souboru
lines = f.read().split( ' n ' )
sections = []
section = []
limit = INTERVAL
#odstraneni prvniho radku s popisky sloupcu
del lines[0]
#rozcleneni na skupiny
for line in lines:
if line == „ “:
break
items = line.split( “;”)
if float(items[1]) < limit:
section.append(items)
else:
sections.append(section)
section = []
limit = limit + INTERVAL
while limit < float(items[1]):
sections.append(section)
section = []
```

limit = limit + INTERVAL
section.append(items)
#spocteni poctu paketu
statistics = []
def is_number(s):
try:
float(s)
return True
except ValueError:
return False
counter = 0
prevsection = []
conndb =
for section in sections:
counter = counter + 1
#soucet delky paketu v bytech
bytesum = 0
for line in section:
bytesum = bytesum + int(line[5])
#pocet paketu ve skupine
packets_cnt = len(section)
#prumerna delka spojeni
#nejdrive zjistime, jestli nektere spojeni bylo uzavreno v teto skupine,
#pokud ano, zjistime, jeslti bylo otevreno v predchozich skupine, pokud ano, zjistime jeho delku
#pro vsechny tyto spojeni spocitame prumer
#hledane otevreni spojeni [SYN, ACK]
#toto je globalni
number_of_connections = 0
for line in section:
if string.find(line[6], „[SYN, ACK]“) != -1 or string.find(line[6], „[SYN]“) != -1:
cols = line[6].split(„ “)
port = „ “
if is_number(cols[0]):
port = cols[0]
elif is_number(cols[2]):
port = cols[2]
else:
port = „ 0“

```
key = line[2] + „<->“ + line[3] + „<->“ + port
#zaznamename cas vytvoreni spojeni
conndb[key] = line[1]
number_of_connections = number_of_connections + 1
number = 0
sum = 0.0
#zjisteni vsechny ukoncenych spojeni ve skupine
for line in section:
if string.find(line[6], „[FIN, ACK]“) != -1:
cols = line[6].split(„ “)
port = „ “
if is_number(cols[0]):
port = cols[0]
elif is_number(cols[2]):
port = cols[2]
else:
port = „ 0 “
key = line[2] + „<->“ + line[3] + „<->“ + port
#podivame, jestli jsou nektera spojeni otevrena v predchozich
if conndb.has_key(key):
number = number + 1
sum = sum + (float(line[1]) - float(conndb[key]))
#odstaneni z db
del conndb[key]
avg_closed_conn = 0
if number > 0:
avg_closed_conn = sum/float(number)
avg_closed_conn = (int(avg_closed_conn*100))/float(100)
statistics.append( [ packets_cnt, bytesum, number_of_connections, avg_closed_conn
] )
print „pocet pkt; pocet bytu, pocet spojeni, prumerne trvani spojeni“
for stat in statistics:
print str(stat[0]) + „;“ + str(stat[1]) + „;“ + str(stat[2]) + „;“ + str(stat[3])
```

A.3 Zdrojový kód pro Stochastický model

Zdrojový kód programu Python

```
    import sys
import string
from sets import Set
INTERVAL = 30.0
f = open(sys.argv[1], 'r')
#radky souboru lines = f.read().split( 'n' )
sections = []
section = []
limit = INTERVAL
#odstraneni prvni radku s popisky sloupcu
del lines[0]
#rozcleneni na skupiny
for line in lines:
if line == „“:
break
items = line.split( ';' )
if float(items[1]) < limit:
section.append(items)
else:
sections.append(section)
section = []
limit = limit + INTERVAL
while limit < float(items[1]):
sections.append(section)
section = []
limit = limit + INTERVAL
section.append(items)
def is_number(s):
try:
float(s)
return True
except ValueError:
return False
counter = 0
prevsection = []
conndb =
statistics = []
for section in sections:
counter = counter + 1
```

```
RTT_counter = 0
time_sum = 0.0
#hledane pakety [SYN,ACK]
for line in section:
if string.find(line[6], „[SYN, ACK]“) != -1:
cols = line[6].split(„ “)
port = „“
if is_number(cols[0]):
port = cols[0]
elif is_number(cols[2]):
port = cols[2]
else:
port = „0“
key = line[2] + „<->“ + line[3] + „<->“ + port
#zaznamename cas SYN,ACK paketu
conndb[key] = line[1]
hledane pakety [ACK]
for line in section:
if string.find(line[6], „[ACK]“) != -1:
cols = line[6].split(„ “)
port = „“
if is_number(cols[0]):
port = cols[0]
elif is_number(cols[2]):
port = cols[2]
else:= „0“
key = line[3] + „<->“ + line[2] + „<->“ + port
if conndb.has_key(key):
RTT_counter = RTT_counter + 1
time_sum = time_sum + (float( line[1] ) - float(conndb[key]))
if (RTT_counter > 0):
statistics.append(time_sum/RTT_counter)
else:
statistics.append(0)
for st in statistics:
print st
```

B OSTATNÍ

Přílohou této semestrální práce je CD-ROM, na kterém je uloženo:

- 48hodinový provoz,
- zdrojový kód pro základní analýzu,
- zdrojový kód pro Stochastický model,
- poissonovo rozděleni ve formátu excel,
- stochastický model ve formátu excel,
- data pro markovský model ve formátu excel.