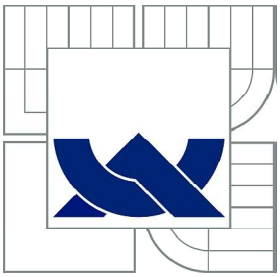
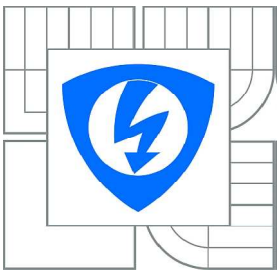


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

TESTOVÁNÍ SÍŤOVÝCH SLUŽEB POMOCÍ EMULÁTORU ROZSÁHLÉ SÍTĚ WAN A SIMULÁTORU ZÁTĚŽE

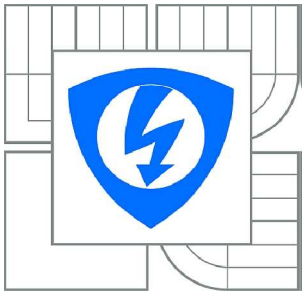
NETWORK SERVICE TESTING WITH WAN NETWORK EMULATOR AND LOAD SIMULATORS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MAREK KOCÁB
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. VÍT NOVOTNÝ, Ph.D.
SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ
TECHNICKÉ V BRNE

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor

Telekomunikační a informační technika

Student:	Bc. Marek Kocáb	***ID:***	111068
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Testování síťových služeb pomocí emulátoru rozsáhlé sítě WAN a simulátoru záteže

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte problematiku požadavku různých telekomunikačních služeb na parametry sítě, jako jsou zpoždění, variabilita zpoždění, propustnost sítě, ztrátovost, aj. Podrobně se seznamte s aplikací emulující činnost reálné sítě typu WAN, jejími možnostmi. Vytvořte řadu scénářů s různými hodnotami transportních parametru sítě, připojte různé druhy provozu a vyhodnoťte vliv těchto parametrů na kvalitu služby. Navrhněte laboratorní úlohu a vytvořte k ní návod.

DOPORUČENÁ LITERATURA:

[1] SZIGETI, T., HATTINGH, CH., End-to-End QoS Network Design. Cisco Press, ISBN 1-58705-176-1, Indianapolis, USA
[2] MARCHESE M. QoS over Heterogeneous Networks. John Wiley & Sons, ISBN 978-0-470-01752-4, 2007

Termín zadání:	6.2.2012	***Termín odevzdání:***	24.5.2012

Vedoucí práce: doc. Ing. Vít Novotný, Ph.D.

Konzultanti diplomové práce:

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

UPOZORNĚNÍ:

ANOTACE

Cílem diplomové práce bylo prostudovat požadavky různých telekomunikačních služeb na parametry sítě, která jsou zpoždění, ztrátovost, propustnost, kolísání zpoždění atd. Provést testování těchto služeb pomocí emulátoru rozsáhlé sítě WAN a simulátoru zátěže. Dále prostudovat jednotlivé TCP algoritmy a provést jejich vyhodnocení ze strany propustnosti.

V první části diplomové práce je popsána kvalita služeb QoS pro hlasovou komunikaci VoIP, videokonverzaci, streamované audio, video a pro data. V další části najdeme, jaké druhy zpoždění vznikají od odesílatele přes mezilehlá zařízení, přenosová média až k příjemci. Třetí kapitola popisuje aplikaci NetDisturb. Je to aplikace, která dokáže emulovat reálné vlastnosti datové sítě a umí přicházející provoz ovlivnit například zpožděním paketů, ztrátovostí, omezit šířku pásma atd. Tématem dalších části je TCP protokol a TCP algoritmy. Jsou zde uvedeny základní vlastnosti TCP protokolu, mechanismy předcházející zahlcení a analýza jednotlivých TCP algoritmů.

V praktické části jsou vytvořeny dvě laboratorní úlohy. První laboratorní úloha se zabýva kvalitou služeb QoS pro různé druhy provozu. Druhá laboratorní úloha provádí analýzu TCP algoritmů z hlediska jejich propustnosti.

KLÍČOVÁ SLOVA

Qos, zpoždění, přenos, TCP protokol, propustnost

ABSTRACT

The main task of this thesis was to explore the requirements of different telecommunications services on the network parameter, that are packet delay, packet loss, throughput, jitter. Make to test network service with WAN network emulator and load simulators. Next focus was to explore TCP congestion avoidance algorithms and focus on their throughput.

The purpose of the first chapter is to introduce Quality of Service (QoS) for VoIP, video conversation, streaming audio, streaming video and data. The second chapter are to explore various types of delay, that are created from the sender through network device, transmission media to the recipient. The third chapter is to introduce NetDisturb application. It is application, that can emulate real properties of the data network and incoming traffic affect example packet delay, packet loss, bandwidth limit and so on. The key objectives of the fourth and fifth part is TCP protocol and TCP congestion avoidance algorithms. Hier are to introduce TCP protocol basic properties, congestion avoidance mechanisms and analysis of TCP congestion avoidance algorithms.

In the practical part are created two laboratory exercises. First laboratory exercise is designed on the implementation QoS for various types of traffic. Second laboratory exercise is focused on the analysis TCP congestion avoidance algorithms and their throughput.

KEYWORDS

QoS, delay, transmission, TCP protocol, throughput

KOCÁB, M. *Testování síťových služeb pomocí emulátoru rozsáhlé sítě WAN a simulátorů zátěže*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 108 s. Vedoucí diplomové práce doc. Ing. Vít Novotný, Ph.D..

Prohlášení

Prohlašuji, že diplomovou práci na téma Testování síťových služeb pomocí emulátoru rozsáhlé sítě WAN a simulátorů zátěže jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestně právních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne …….. .....

podpis autora

Poděkování

Děkuji vedoucímu práce doc. Ing. Vítu Novotnému, Ph.D. za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce. Dále chci poděkovat Ing. Radku Krkošovi za pomoc při problému s hardwarovým zařízením.

V Brně dne …… .....

podpis autora

SEZNAM ZKRATEK

AIAD	Additive Increase Additive Decrease
AIMD	Additive Increase Multiplicative Decrease
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CWND	Congestion Window
DHCP	Dynamic Host Configuration Protocol
DSCP	Differentiated Services Code Point
EIGRP	Enhanced Interior Gateway Routing Protocol
FDDI	Fiber Distributed Data Interface
FF	Fragment Free
FIFO	First In First Out
FTP	Foiled Twisted Pair
FTP	File Transfer Protocol
GEO	Geostationary Earth Orbit
HTTP	Hypertext Transfer Protocol
HSTCP	HighSpeed TCP
IP	Internet Protocol
IPv6	Internet Protocol v6
LEO	Low Earth Orbit
LTE	Long Term Evolution
MAC	Media Access Control
MEO	Medium Earth Orbit
MGCP	Media Gateway Control Protocol
MSL	Maximum Segment Lifetime
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NAT	Network Address Translation
OWIN	Outstanding Window
OSPF	Open Shortest Path First
QoS	Quality of Service
RAM	Random Access Memory
RIPv2	Routing Information Protocol verse 2
RSVP	Resource-Reservation Protocol
RTCP	Real-time Transport Control Protocol
RTO	Retransmission TimeOut
RTP	Real-time Transport Protocol
RTT	Round Trip Time
RTTVAR	Round Trip Time VARiation
RWND	Receiver Windows
S&F	Store & Forward
SAP	Systems - Applications - Products in data processing
SACK	Selective ACKnowledgment
SIP	Session Initiation Protocol
SN	Sequence Number
SNMP	Simple Network Management Protocol

SOAP	Simple Object Access Protocol
SSTHRESH	Slow Start Threshold
STP	Shielded Twisted Pair
TCP	Transmission Control Protocol
TTL	Time To Live
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunication System
UTP	Unshielded Twisted Pair
VAD	Voice Activity Detection
VoIP	Voice over Internet Protocol
WiMax	Worldwide Interoperability for Microwave Access
xDSL	x Digital subscriber line
XML	eXtensible Markup Language

Obsah

ÚVOD 15
1 KVALITA SLUŽEB - QoS 17
1.1 Kvalita služby VoIP 17
1.2 Kvalita služeb pro videopřenosy, streamované video a audio 19
1.3 Kvalita služeb pro data 20
1.4 Třídy QoS 21
1.5 Implementace QoS 22
2 ZPOŽDĚNÍ V SÍTI 23
2.1 Zpoždění v koncových zařízeních 23
2.2 Zpoždění v mezilehlých zařízeních 24
2.2.1 Opakovač (repeater) a rozbočovač (hub) 25
2.2.2 Přepínač (switch), most (bridge) 25
2.2.3 Vícevrstvý směrovací prvek – L3 přepínač (L3 switch) 26
2.2.4 Směrovač (router) 26
2.3 Zpoždění na přenosových médiích 27
2.3.1 Metalický kabel 28
2.3.2 Optický kabel 28
2.3.3 Bezdrátový přenos 29
3 EMULÁTOR SÍTÍ NETDISTURB 30
3.1 Uživatelské rozhraní NetDisturb Server 31
3.2 Uživatelské rozhraní NetDisturb Client 33
4 ZÁKLADNÍ VLASTNOSTI PROTOKOLU TCP 39
4.1 Záhlaví TCP protokolu 40
4.2 Průběh navázání spojení, přenos dat a ukončení spojení 41
4.2.1 Princip potvrzování 43

4.2.2 Nastavení časovače ... 43
4.3 Řízení toku dat protokolem TCP ... 45
4.4 Mechanismy předcházející zahlcení ... 47
4.4.1 Pomalý start (slow start) ... 48
4.4.2 Vyhýbání se zahlcení (congestion avoidance) ... 48
4.4.3 Rychlé přeposílání (fast retransmit) ... 49
4.4.4 Rychlé zotavení (fast recovery) ... 49
4.4.5 Omezené vysílání (limited trasmit) ... 50
5 ALGORITMY TCP PROTOKOLU ... 51
5.1 Propustnost algoritmů TCP ... 57
5.2 Shrnutí ... 73
ZÁVĚR ... 74
SEZNAM LITERATURY ... 76
PŘÍLOHY ... 80
LABORATORNÍ ÚLOHA č.1 - Měření a testování kvality služeb pro ... 80
různé druhy provozu ... 80
Teoretický úvod ... 80
Postup ... 83
Kontrolní otázky ... 91
LABORATORNÍ ÚLOHA č. 2 – Analýza TCP algoritmů ... 93
Teoretický úvod ... 93
Postup ... 100
Kontrolní otázky ... 106

SEZNAM OBRÁZKŮ

Obrázek 1.1: Velikost paketů přenášených u videokonferenční služby 20

Obrázek 2.1: Místa vzniku zpoždění v komunikační síti a jeho zdroje 23

Obrázek 3.1: Princip aplikace NetDisturb 30

Obrázek 3.2: Odškrtnutí protokolu pro operační systém Windows 7 a Windows XP..... 32

Obrázek 3.3: Varovné hlášení při nesplnění podmínek pro spuštění NetDisturb Server 32

Obrázek 3.4: Uživatelské rozhraní NetDisturb Server..... 33

Obrázek 3.5: Připojení klienta na server na lokálním PC..... 34

Obrázek 3.6: Připojení klienta na vzdálený server 34

Obrázek 3.7: Hlášení o chybě spojení mezi klientem a serverem 35

Obrázek 3.8: Uživatelské rozhraní NetDisturb Client 37

Obrázek 3.9: Ukázkový příklad a chyba u vybraného agregátu 37

Obrázek 3.10: Konfigurace síťových rozhraní 38

Obrázek 4.1: TCP záhlaví 40

Obrázek 4.2: Průběh navazování TCP spojení..... 42

Obrázek 4.3: Přenos dat jednosměrná (vlevo) a obousměrná komunikace 42

Obrázek 4.4: Ukončení TCP spojení..... 43

Obrázek 4.5: Výměna zpráv mezi vysílačem a přijímačem..... 46

Obrázek 4.6: Pilovitý průběh okénka zahlcení cwnd..... 47

Obrázek 4.7: Fáze pomalý start a vyhýbání se zahlcení 49

Obrázek 4.8: Fáze rychlé přeposílání a rychlé zotavení 50

Obrázek 5.1: Propustnost TCP algoritmů pro linku 10 Mb/s a ztrátovosti 1 z 20..... 59

Obrázek 5.2: Propustnost TCP algoritmů pro linku 10 Mb/s a ztrátovosti 1 ze 100 61

Obrázek 5.3: Propustnost TCP algoritmů pro linku 10 Mb/s a ztrátovosti 1 ze 150 63

Obrázek 5.4: Propustnost TCP algoritmů pro linku 100 Mb/s a ztrátovosti 1 z 20..... 65

Obrázek 5.5: Propustnost TCP algoritmů pro linku 100 Mb/s a ztrátovosti 1 ze 100 67

Obrázek 5.6: Propustnost TCP algoritmů pro linku 100 Mb/s a ztrátovosti 1 ze 150 69
Obrázek 5.7: Propustnost TCP algoritmů pro linku 10 Mb/s a náhodná ztrátovost do 5 % 71
Obrázek 5.8: Propustnost TCP algoritmů pro linku 100 Mb/s a náhodná ztrátovost do 5 % 73
Obrázek A. 1: Princip aplikace NetDisturb 81
Obrázek A. 2: Připojení NetDisturb klienta na server 83
Obrázek A. 3: Schéma zapojení laboratorní úlohy 84
Obrázek A. 4: Nastavení síťových rozhraní 85
Obrázek A. 5: Pojmenování provozů 85
Obrázek A. 6: Výchozí nastavení filtru RTP Video Filter 86
Obrázek A. 7: Nastavení kodeku pro streamované video 86
Obrázek A. 8: Nastavení filtru pro videokonverzaci 87
Obrázek A. 9: Nastavení kodeku pro streamované audio 87
Obrázek A. 10: Puštění provozu bez použití filtrů 87
Obrázek A. 11: Nastavení ztrátovosti a propustnosti 88
Obrázek A. 12: Nastavení zpoždění 88
Obrázek A. 13: Nastavení protokolu pro streamování 89
Obrázek A. 14: Nastavení IP adresy a kodeku pro streamování 89
Obrázek A. 15: Vytvoření agregátu 91
Obrázek A. 16: a) přiřazení agregátu b) změna tlačítek Run 91
Obrázek A. 17: Varovné hlášení 92
Obrázek B. 1: Fáze pomalý start a vyhýbání se zahlcení 95
Obrázek B. 2: Fáze rychlé přeposílání a rychlé zotavení 96
Obrázek B. 3: Princip aplikace NetDisturb 99
Obrázek B. 4: Schéma zapojení laboratorní úlohy 100
Obrázek B. 5: Nastavení IP adresy pro PC3 100
Obrázek B. 6: Konfigurace FTP serveru a uživatele 102

Obrázek B. 7: Nastavení síťových rozhraní 104
Obrázek B. 8: Provoz bez omezení 104
Obrázek B. 9: Aplikování ovlivnění provozu z A do B 105
Obrázek B. 10: Nastavení propustnosti 105
Obrázek B. 11: Varovné hlášení 108

SEZNAM TABULEK

Tabulka 2.1: Zpoždění pro různé velikosti rámců v přepínačích (převzato z [14]) 26
Tabulka 2.2: Serializační zpoždění pro různé velikosti rámců a různé přenosové rychlosti 28
Tabulka 4.1 Velikost MTU podle použité síťové technologie 39
Tab. 1: Kvalita QoS pro různé druhy provozu služeb 90

ÚVOD

V dnešní multimediální době, kde každý má možnost připojit se k Internetu, dochází k velkému rozvoji síťových služeb. Sortiment služeb nabízených poskytovatelem služeb je závislý na spokojenosti uživatelů s provozem služeb a cílem nabídky je potencionálního zákazníka co nejvíce zaujmout. Výsledkem snažení operátora je zlepšování služeb co do technických parametrů, i do přístupu operátora k zákazníkům (úroveň zákaznické podpory). Co se týče technických parametrů služeb, jedná se především o co nejmenší zpoždění zahájení služby, co nejmenší zpoždění přenosu informace, aby nedocházelo k různým výpadkům a ztrátám či zkreslení informace, apod. Požadavky na hodnoty technických parametrů a úroveň jejich dodržení v síťovém prostředí se označují jako Quality of Service (QoS). V integrovaném síťovém prostředí je pak úkolem přidělit každé službě tolik prostředků, aby svou technickou kvalitou uspokojila potřebu uživatele služby, a současně, aby dostupné síťové prostředky byly využity co nejefektivněji. V praxi to bývá realizováno tak, že systém každé realizované službě přidělí určitou prioritu, podle toho jsou pak datové jednotky systémem obsluhovány, například data, videokonverzační služby musí mít velmi malé a málo proměnlivé zpoždění, protože komunikace probíhá v reálném čase, a tak její data mají přidělenou vysokou prioritu. Naopak u stahování obecných dat zpoždění a jeho proměnlivost tolik nevadí, jde především o to, aby data byla stažená správně, bez chyb a priorita je tedy nízká.

Cílem této práce je prostudovat požadavky různých telekomunikačních služeb na parametry sítě, mezi které patří zpoždění, kolísání zpoždění, propustnost, ztrátovost atd. Dále seznámit se s aplikací umožňující emulaci reálné sítě typu WAN a jejími možnostmi. Připojit do sítě různé druhy provozu vyhodnotit, ověřit vliv těchto parametrů na kvalitu služby a navrhnout laboratorní úlohu s návodem pro studenty. Pomocí aplikace emulující reálné sítě typu WAN vytvořit druhou laboratorní úlohu, která bude sledovat zpomalení TCP vysílání například v době zahlcení sítě, ztrátovosti.

V první kapitole je vysvětlena kvalita služeb QoS a jaké problémy řeší. Potom jsou zde popsány různé druhy služeb a parametry, které je ovlivňují. Na závěr této kapitoly jsou zmíněné služby rozděleny do čtyř tříd QoS a přiřazeny priority, podle kterých budou zpracovávaný v rámci komunikační sítě.

V druhé kapitole jsou popsány jednotlivé zdroje zpoždění od odesílatele přes mezilehlá zařízení, přenosová média až k příjemci. Vysvětluje, která zpoždění během přenosu jsou a nejsou zanedbatelná. Dále jak probíhá zpracování jednotlivých paketů během cesty v komunikační síti.

Třetí kapitola popisuje aplikaci NetDisturb, která dovoluje ovlivňovat příchozí provoz, nastavit jeho zhoršení (jako ztrátovost, zpoždění, jitter atd.) a poslat ho k příjemci. Je zde popsána uživatelská a serverová část. Dále jak danou aplikaci bez problému spustit a začít s ní pracovat.

Ve čtvrté kapitole je proveden rozbor protokolu TCP, který pracuje na transportní vrstvě v modelu OSI. Jeho postup při navazování a ukončování spojení. Dále je zde vysvětlen princip časovače. Patří mezi nejdůležitější parametr TCP protokolu, kde jeho správným nastavením nedochází ke zbytečnému plýtvání šířky pásma sítě. V další části práce se nacházejí mechanismy, jejichž cílem je předcházet zahlcení. Mezi tyto mechanismy patří pomalý start, vyhýbání se zahlcení, rychlé přeposílání, rychlé zotavení a omezené vysílání.

V poslední kapitole jsou popsány jednotlivé TCP algoritmy, které mají v sobě implementované mechanismy pro předcházení zahlcení, potom jejich reakci v případě ztráty

segmentů. Dále je provedena analýza a porovnání jednotlivých TCP algoritmů z hlediska jejich propustnosti, férovosti a chováním k ostatním tcp spojením.

V závěru jsou navrženy dvě laboratorní úlohy. Kde první laboratorní úloha slouží pro ověření úrovně požadavků různých služeb na kvalitativní parametry. Studenti budou mít možnost ovlivňovat provoz a sledovat co se s ním děje. V druhé laboratorní úloze provedou konfiguraci FTP serveru na operačním systému Ubuntu a následně budou provádět analýzu jednotlivých TCP algoritmů z hlediska jejich propustnosti při omezené přenosové rychlosti a ztrátovosti. Součástí obou úloh jsou i kontrolní otázky k prověření znalostí absolventa laboratorní úlohy, jak o kvalitě služeb – QoS tak i o propustnosti jednotlivých TCP algoritmů.

1 KVALITA SLUŽEB - QoS

V dnešní době dochází k rychlému rozvoji služeb a úspěšnost každé služby závisí nejen na uživateli, ale také i na komunikační síti, přes kterou jsou služby provozovány. Můžeme si to představit následovně, například u stahování souborů nebo načítání stránek dochází k problémům, které nejsou hned viditelné. Myslí se tím, různé zpoždění v doručení paketů, příjem paketů v jiném pořadí než byly vysílány, ale i ztrátovost paketů. Samozřejmě protokolová výbava koncových uzlů i komunikačního systému se o tyto problémy postará a my nic nepoznáme. Ale jsou jiné služby, kde musíme mít výsledky „okamžitě“ (v relativně krátké době vzhledem k lidským měřítkům, tj. jednotky až stovky ms). Patří sem hlasová komunikace (VoIP - Voice over Internet Protocol), videokonverzace, různá streamovaná videa a audia. Výsledkem nedodržení požadavků je narušování plynulosti realizované služby, například u videokomunikace dojde k „zaseknutí“ obrazu, u hlasové služby k zadrhávání řeči nebo k opětovnému pokládání dotazu mluvčím, nedočká-li se do určité doby odezvy od protějšího účastníka atd. Tyto problémy se snaží řešit technologie podpory kvality služeb pod názvem QoS.

Hlavním úkolem podpory QoS je, aby se zabránilo zahlcení komunikačních spojů a přepojovacích prvků daty a aby se každé realizované službě zajistil dostatek prostředků podél trasy, kudy probíhá přenos informace. Technologie podpory QoS je schopna rozlišovat jednotlivé přenosy, jejich typy a každému typu nastavit jinou kvalitu. Zjednodušeně lze říct, že jakýkoliv provoz se bude snažit doručit včas a bez žádných problémů. Pomáhá si přidělováním různých priorit ke každému provozu. Nejprve je provoz klasifikován a z klasifikace potom plyne priorita. Platí, že vyšší hodnota priority má přednost před nižší hodnotou priority a podle toho se pak zachází s provozem. Mezi jeho další používané funkce patří omezování přenosového pásma (nastaví se maximální přenosové pásmo pro využití) a vyhrazení přenosového pásma (nastaví se minimální přenosové pásmo, které bude provoz využívat).

U provozování některých služeb může dojít většinou k špatnému přenosu paketů a tyto problémy QoS řeší. Přenos paketů je ovlivňován zpožděním, kolísáním zpožděním tzv. jitter, ztrátovostí paketů, šířkou pásma a doručení paketů mimo dané pořadí. Zpoždění je čas, kdy požadovaná data nejsou doručena okamžitě, ale s menší časovou prodlevou. Na tohle zpoždění má vliv zpracování dat v koncovém zařízení, v odchozích frontách, přes dobu zpracování informace v mezilehlých zařízeních, spolu se serializačním zpožděním a také použité přenosové médium. Více o tomto zpoždění a jeho typech je napsáno v kapitole 2. Kolísání zpoždění je rozdíl v intervalech mezi přijímanými pakety. Tento jitter se odstraňuje za pomoci vyrovnávací paměti tzv. jitter buffer, který vyrovnává tyto rozdíly v komunikaci. Procento paketů, které jsou ztraceny při průchodu v komunikační síti, nazýváme ztrátovost paketu. Šířka pásma definuje, jaký objem dat můžeme odeslat přes síť v daném čase, je udávána v bitech za sekundu (b/s). Poslední co ovlivňuje provozování služeb je doručování paketu mimo dané pořadí. Pakety procházejí v komunikační síti různými cestami nebo při jejich ztrátě jsou vysílány znovu a pořadí je hned změněno, jakmile dojdou k příjemci.[9][10][11]

1.1 Kvalita služby VoIP

Zpracování hlasu přes IP síť zahrnuje digitalizovat hlasový signál, převést ho do paketového formátu IP a přenést po IP síti do cílového koncového uzlu, dekódovat datový tok

a převést ho na akustický signál. Díky tomu, že můžeme posílat hlasový provoz s daty po IP síti, snížily se výdaje na hlasové služby, jak u koncových uživatelů, tak i v podnicích. Mezi důležité požadavky pro správný provoz služby VoIP patří trvalá dostupnost služby, zpoždění v jednom směru pod 150 ms, předat informace jen oprávněným osobám, zajistit vysokou kvalitu a odezvu v reálném čase. Z hlediska šířky pásma potřebuje služba VoIP menší šířku, než je u klasických telefonních sítí, jsou zde totiž metody komprese a potlačení ticha. Metoda komprese aplikuje na různé zvuky různý počet bitů. Dochází tím k šetření šířky pásma a není snížena kvalita hovoru. Během hovoru často dochází k tomu, že v určitých časových úsecích nemluví jedna nebo obě strany, nevyužije se generovaný datový tok, protože nenese téměř žádnou informaci a pomocí metody potlačení ticha je tato šířka pásma ušetřena a ve výsledku dochází k více než 50% úspoře přiděleného pásma. Metoda potlačení ticha používá k této detekci mechanismus VAD (Voice Activity Detection), který neustále monitoruje hluk v pozadí a podle toho je nastavena detekce řečové aktivity. Služba VoIP musí mít tedy zajištěnou potřebnou šířku pásma pro hovor i signalizaci. Jinak dochází ke snižování hlasové kvality u této služby.[15]

Služba VoIP používá transportní protokol UDP (User Datagram Protocol) pro přenos hlasové informace spojený s protokolem RTP (Real-Time Protocol) a dohled nad přenosem (informace o přenosu) provádí protokol RTCP (Real-time Transport Control Protocol). Signalizace je přenášena pomocí signalizačních protokolů například H.323, SIP (Session Initiation Protocol), MGCP (Media Gateway Control Protocol) a obsahuje informaci o navázání, ukončení a případné změny v sestavované relaci.

Cílem podpory QoS jsou pro VoIP řešeny problémy jako celkové zpoždění, kolísání zpoždění, ztrátovost paketů.

Celkové zpoždění, jak bylo zmíněno v předchozí kapitole, definuje dobu zpoždění přenosu informace od okamžiku jejího vstupu do komunikačního systému a okamžiku výstupu z komunikačního systému požadovaným výstupním bodem, tedy od zachycení zvukového signálu mikrofonem telefonu na straně mluvčího a výstupem zvukového signálu ze sluchátka telefonu na straně naslouchajícího. Velké zpoždění má za následek degradaci kvality hovoru. Podle normy ITU-T G.114 jsou definovány časy, které určují kvalitu hovoru. Jestliže celkové zpoždění dosáhne hodnoty pod 150 ms, je tento hovor velmi kvalitní a při zpoždění pod 20 ms nedochází k ozvěně v hovoru. Mezi 150-400 ms považujeme tento hovor za dobrý (přijatelný). Pokud hodnota přesáhne 400 ms, srozumitelnost mezi komunikujícími je špatná a může mezi nimi dojít i ke ztrátě synchronizace.

Kolísání zpoždění je rozdíl v intervalech mezi přicházejícími pakety a eliminuje se na straně příjemce ve vyrovnávací paměti. Každé VoIP zařízení má implementovanou adaptivní vyrovnávací paměť, která se mění podle aktuální hodnoty rozptylu zpoždění. Maximální hodnota bývá už přednastavena. Při statickém nastavení pro vyrovnání rozptylu je nejčastěji hodnota 60 ms. Kolísání zpoždění najdeme v síti, kde dochází k různému mísení toku (vstupů) od jediného výstupu. U koncového zařízení je vyrovnávací paměť implementována a eliminuje nežádoucí rozptyl (více v kapitole 2.1).

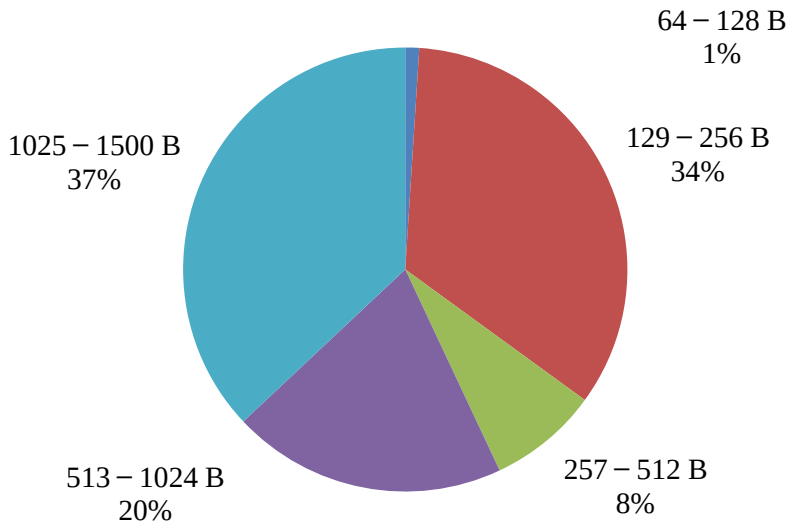
Ztrátovost paketů u VoIP je nežádoucí, tak jako poslání paketu znova nebo snížení rychlosti. Ve výsledku může dojít ke ztrátě (výpadku) hovoru a neuslyšíme zprávu od volaného. Pro názornost jsou zde uvedeny procentuální ztrátovosti, které ovlivňují kvalitu hovoru. Ztráty paketu do 2 % nemají vliv na kvalitu hovoru. Mezi 2-5 % pomalu dochází k nesrozumitelnosti hovoru, které ovlivňuje typ kodeku a jeho implementované vlastnosti pro maskování ztrát. U 5-10 % je srozumitelnost nízká a znehodnocuje se kvalita hlasu. Pokud ztrátovost překročí 10 %, kvalita hovoru bude velice nízká, srozumitelnost minimální a nepomůžou ani kvalitní kodeky proti odolnosti vůči ztrátám.

Na závěr této podkapitoly je důležité vědět, že přenos hlasu potřebuje menší, zato konstantní šířku pásma, ale u kodeku s proměnlivou rychlostí to už neplatí. Reakce na zpoždění je mnohem vyšší než běžného stahování dat, maximální hodnota kolísání zpoždění by se měla pohybovat pod 20 ms a ztrátovost paketů je doporučena do 2 %. [3][11][16]

1.2 Kvalita služeb pro videopřenosy, streamované video a audio

Použitím těchto služeb jsou určeny jiné podmínky na komunikační síť, ale jejich odlišnost není tak velká.

U videokonverzační služby (například videokonferenční služba nebo videotelefon) musí být zajištěna potřebná kvalita, jak přenášeného hovoru, tak i obrazu. Při provozování videokonference je potřeba zajistit, aby přenášené pakety přes komunikační síť měly stejné zpoždění a interval mezi přijímanými pakety by měl být také stejný. Pro tento typ služby jsou definovány následující parametry. Jsou to ztrátovost paketů, neměla by být větší než 1 %, kolísání zpoždění pod 30 ms a jednosměrné zpoždění by se mělo pohybovat pod 150 ms. Velkou odlišnost, ale nalezneme u celkového provozu této služby. Pakety nejsou stejné, ale pokaždé mají různou velikost a přenosová rychlost se také velmi často mění. Ve společnosti Cisco Systems byl proveden výzkum, kde jeho záměrem bylo zjistit, jaké velikosti paketů jsou nejčastěji přenášeny. Výsledky uváděly, že pakety o velikostech 129-256 B a 1025-1500 B byly k provozu této služby používány nejčastěji, viz Obrázek 1.1. Pakety s největší velikostí 1025-1500 B nesly informace o přenášeném videu pomocí I (referenčního) snímku a pakety o velikostech 129-256 B přenášely informace o snímcích P a B, kde snímek B je získán z rozdílu aktuálního snímku I a následujícího snímku I. Snímek P je vektor pohybu, který obsahuje informace o tom, jak se změnil aktuálně kódovaný snímek oproti předcházejícímu snímku. Minimální přenosová rychlost pro provozování videokonferenční služby je 128 kb/s, přičemž u této rychlosti musí být zajištěno kvalitní spojení mezi koncovými zařízeními. Důvodem je, že tato služba používá pro přenos paketů transportní protokol UDP a během přenosu může dojít ke ztrátě paketů. Pokud je během cesty mezi koncovými zařízeními ztraceno pár paketů, naše oko to nepostřehne, ale při větší ztrátovosti už to poznáme například obraz, je zmražen, uvidíme „kostičky“ v obraze atd. Dále k této rychlosti musí být také připočítána šířku pásma pro režijní informace obsahující informace hlavičky protokolů RTP, UDP, IP (Internet Protocol) a i hlavičky linkové vrstvy. Důvodem je, že při velkém objemu dat jednoznačně neurčíme procentuální podíl režijních informací v celkovém toku dat. Proto je připočítávaná hodnota 20 % a bere se v úvahu, že tuto hodnotu režijní informace nepřekročí. Pro dříve zmíněnou minimální rychlost 128 kb/s, je rezervována šířka pásma 154 kb/s.[3][18]



Obrázek 1.1: Velikost paketů přenášených u videokonferenční služby

Streamovaná videa jsou méně náročnější na přenos než videopřenosy, protože se jedná o jednosměrný přenos. Tato služba není tolik ovlivněna zpožděním, a proto může příjemce využít větší vyrovnávací paměť, která dokáže odstranit vyšší hodnotu kolísání zpoždění u přicházejících paketů. Mezi důležité parametry tohoto provozu patří, že ztrátovost paketu je do 5 %, zpoždění se pohybuje mezi 4-5 s a streamování videa je většinou jednosměrné tzn., síťové prostředky jsou využívány asymetricky. Služba je většinou určena pro zábavu a bývá často potlačována různými firemními aplikacemi, protože zaměstnavatel chce, aby zaměstnanci pracovali, a nechce, aby během pracovní doby sledovali videa například na YouTube. [3][18]

Na streamované audio neboli hudbu a mluvené slovo má také vliv zpoždění. Lidé dokážou lépe vnímat chybný přenos zvuku než špatně přenášené video, proto QoS má velmi přísné požadavky na ztrátovost, zpoždění a jitter pro zvuk. Požadavky na šířku pásma jsou malé. Pokud bude ztrátovost do 1 % a nedochází k žádným změnám, do 13 % pořád rozumíme mluvenému slovu, ale už slyšíme „praskání“, do 20 % větám rozumíme, jejich obsah si můžeme částečně domyslet, ale „praskání“ se zvětšuje, pokud je ztrátovost paketů nad 25 % srozumitelnost je na nízké úrovni, věty budou méně nebo vůbec pochopitelné, „praskání“ bude intenzivnější a bude mnohem lepší přenos ukončit. Zpoždění jednotlivých paketů by se mělo pohybovat od 300 do 800 ms a při zpoždění do 20 ms neuslyšíme echo ozvěnu. Kolísání zpoždění je ovlivněno vyrovnávací paměti u každého příjemce. Poznáme ho tak, že příjemce musí počkat určitou dobu, než budou načteny pakety do vyrovnávací paměti. Je to dáno kvůli tomu, že některé pakety dojdou později. U předčasného přehrávání by příjemce slyšel mezery v signálu a nedalo by se to poslouchat. Po načtení všech paketů do vyrovnávací paměti bude signál přehrán bez mezer.[27]

1.3 Kvalita služeb pro data

Datové služby patří do největší skupiny provozovaných služeb, jejichž cílem je provozovat různé síťové aplikace. Při analyzování provozu pro datové služby byly definovány čtyři základní mechanismy služeb a jsou to Best-Effort, Bulk Data, Transakční a interaktivní služby a Locally Defined Mission-Critical Data (Uživatelsky definované kritické služby).

Prvním typem služby je Best-Effort, hlavním jejím znakem je, že nepoužívá žádné priority, snaží se co nejrychleji a nejlépe přenést pakety během cesty v komunikační síti k cíli. Pokud je malá dostupná šířka pásma dochází k zahození paketu a ztráta paketů je řešena opakováním vysílání. Doba doručení datových jednotek je proměnlivá a závisí na vytíženosti sítě. Je spousta odlišných síťových aplikací v rozmezí stovek až tisíce a proto musíme dbát na dostatečnou šířku pásma. Pro plynulý provoz této služby je vhodné vyhradit minimálně 25 % dostupné šířky pásma. Dalším typem je služba Bulk Data, která je vhodná pro přenos objemných dat přes komunikační síť. Uplatnění nachází v neinteraktivních aplikacích, není citlivá na zpoždění, je provozována většinou na pozadí. Hlavně je kladen důraz na to, aby data byly přeneseny správně a bez chyb. Služba je používána u FTP (File Transfer Protocol), e-mailové komunikace, zálohování, synchronizování databáze atd. Dokáže dynamicky využívat šířku pásma během doby, kdy není síť tolik zatížená a u intenzivního provozu nebrání přidělování síťových zdrojů pro jiné aplikace. Transakční a interaktivní služby jsou třetím typem služby pro přenos dat. Jsou kombinací dvou typu aplikací a to transakční aplikace pomocí tenkého klienta (klient-server) například Oracle, SAP (Systems - Applications - Products in data processing) nebo interaktivní komunikační služby jako ICQ, Jabber, Google Talk na gmailu. Posledním typem je uživatelsky definovaná kritická služba, kde její hlavním úkolem je upřednostnit transakční a interaktivní služby před ostatními. Jaké služby budou prioritně využívány, záleží na požadavcích jednotlivých firem (podniků). Aplikováním této služby bude výsledkem pouze provoz od zvolených transakčních nebo interaktivních síťových aplikací.[3][18]

1.4 Třídy QoS

V předchozích podkapitolách byly popsány síťové služby podle jejich nároků na šířku pásma, ztrátovosti paketu, zpoždění a jitter. Pro lepší přehlednost a orientaci jsou jednotlivé služby rozděleny do čtyř odlišných tříd. Patří sem:

- konverzační třída,
- třída proudového přenosu dat (data streaming),
- interaktivní třída,
- třída služeb probíhající na pozadí.

V konverzační třídě jsou provozovány služby, které pracují v reálném čase například telefonie VoIP, videokonverzace atd. Tato třída velmi přísně kontroluje zpoždění a jitter, musí být co nejmenší, a proto je řazena do třídy s nejvyšší prioritou. Komunikace je typu klient-klient.

Třída proudového přenosu dat neboli streaming třída slouží k provozování audiovizuálního materiálu mezi koncovým uživatelem a zdrojem například televizního vysílání, různá internetová rádia nebo stahování videoklipů a hudby. Probíhá zde komunikace typu klient-server. Třída je citlivá na kolísání zpoždění než na zpoždění, protože na straně příjemce se nachází vyrovnávací paměť, která má omezenou velikost a například u videa jde hlavně o plynulost, teprve až potom o kvalitu obrazu.

Interaktivní třída pracuje na principu žádosti a odpovědi, dále musí zajistit během trvání komunikace, že celý obsah dat bude doručen správně a s minimální chybovostí. Zpoždění se pohybuje řádově do jednotek sekund. V této třídě najdeme služby pro prohlížení internetových stránek, stahování e-mailových zpráv, správu nebo přístup k různým serverům a databázím.

Poslední třídou je třída služeb probíhající na pozadí, do které patří například služby pro stahování a přenos dat (FTP), otevírání webových stránek atd. Tato třída není citlivá na zpoždění ani kolísání zpoždění, ale očekává se bezchybné doručení dat. Proto mají tyto služby nejnižší prioritu a nejmenší požadavky na šířku pásma.]

1.5 Implementace QoS

V moderních komunikačních sítích musí být zajištěno, aby provozované služby (jejich datové jednotky) dokázala síť rozeznat a následně jim zajistila požadovaný způsob zacházení. Pro způsoby zacházení existují dva typy podpory QoS:

- Integrované služby (Integrated Services - IntServ),
- Diferencované služby (DiffServ).

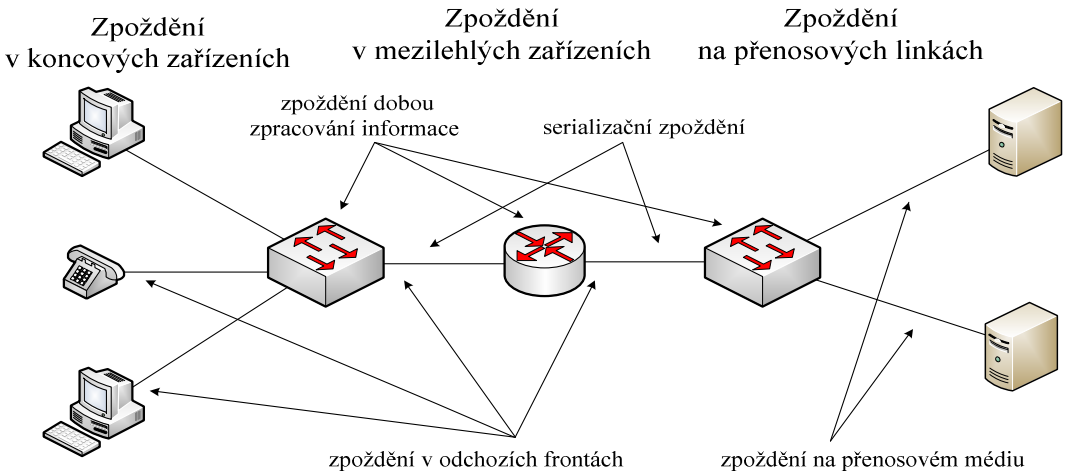
U integrovaných služeb aplikace informují síť o svých požadavcích na kvalitu služby například minimální šířku pásma, maximální zpoždění atd. Síť podle požadavku aplikace ověří, zda může mít k dispozici dostatek prostředků na splnění daného požadavku od aplikace a potom se rozhodne, jestli požadavkům vyhoví nebo ne. Jestli síť nemůže požadavkům vyhovět, spojení nebude vytvořeno a aplikace může zmírnit své požadavky na přenos. Pokud síť může požadavkům vyhovět, informuje všechny síťové prvky o těchto požadavcích a provede se rezervace požadovaných prostředků například šířka pásma spoje mezi směrovači, nastaví se velikost front pro pakety atd. K rezervaci těchto prostředků je používán protokol RSVP (Resource Reservation Protocol), může pracovat i na sítích s protokolem IPv6. Kvalita služby u technologie IntServ je zajištěna během celého spojení od zdroje až k příjemci přes všechny síťové prvky. U všech síťových prvků, přes které spojení prochází, dochází k velkému zatížení procesoru, musí neustále udržovat informaci o jednotlivých spojeních. Je to nevýhoda této technologie.

U diferencovaných služeb aplikace neoznamují síti své požadavky na kvalitu služby. Směrovače neudržují informace o jednotlivých spojeních, rezervační protokoly lze použít, ale nemusíme. Kvalita služby, je zde zajištěna pomocí značkovaní, kde každý paket má přidělenou značku. Podle označkování je určeno, jak s paketem bude v síti zacházeno a určuje třídu QoS. Značkování je u DiffServ prováděno v poli DSCP (DiffServ Code Point) v hlavičce paketu. Označkování paketů probíhá před vstupem do DiffServ domény. Cestou přes síť si směrovače přečtou značku paketu a provedou jeho zpracování. Může se stát, že paket nesplňuje sjednané podmínky přenosu nebo přechází z jedné sítě do druhé (v jiné síti může být jiný způsob značkování), dochází tedy k přeznačkování. Implementace DiffServ nezatěžuje tolik procesor síťových prvků a v dnešní době je nejpoužívanější pro zajišťování kvality služeb.[30][31]

Cílem práce není popsat podrobně princip jednotlivých implementací QoS, ale pro úplnost byly zde uvedeny a stručně vysvětleny.

2 ZPOŽDĚNÍ V SÍTI

Zpoždění můžeme charakterizovat jako dobu, která uplyne od odeslání zprávy od zdroje až po její doručení přes komunikační síť příjemci. Zpoždění v paketových komunikačních sítích má proměnlivý charakter, kde se vlivem proměnlivých poměrů v síti neustále mění její velikost a ovlivňuje tak celkové zpoždění přenášené zprávy. Pro lepší vysvětlení kde ke zpoždění dochází v komunikační síti, lze vidět na Obrázek 2.1. Je potřeba také rozlišovat zpoždění jednosměrné a zpoždění obousměrné tzv. round-trip latency. Jednosměrné zpoždění můžeme chápat jako dobu, kdy je odeslána zpráva od zdroje až po následné přijetí příjemcem. U obousměrného zpoždění je započítávaná doba přenosu zprávy tam i zpět a následné zpracování příjemcem. Tato specifikace zpoždění se v sítích používá nejčastěji, můžeme ho změřit z jedné stanice (uzlu). Celkové zpoždění můžeme charakterizovat jako součet řady parciálních zpoždění. Každý prvek v komunikační síti vnáší do přenosu nějaké zpoždění. Velmi důležitým faktorem je, jaký mají tato zpoždění vliv na výsledné zpoždění. Následující podkapitoly postupně rozebírají parciální zpoždění dle místa svého vzniku. Zpoždění hraje velmi důležitou roli při přenosu audia, videa, videokonference a jiných multimediálních služeb. Aby se zabránilo problémům se zpožděním je používána technologie zajištění kvality služby s názvem QoS.[12]



Obrázek 2.1: Místa vzniku zpoždění v komunikační síti a jeho zdroje

2.1 Zpoždění v koncových zařízeních

Koncová zařízení jsou zdrojem a cílem přenášené informace v komunikační síti, například to mohou být počítače, servery, ale také zdroje zvuku a videa (VoIP telefony, videotelefony, kamery, IP televize atd.). Odesílatel na koncovém zařízení připraví zprávu pro přenos. Zpráva bude obsahovat adresu odesílatele a příjemce. Potom je segmentována a na nižších vrstvách je zapouzdřována, tzn. na každé vrstvě je přidáno záhlaví (hlavička) pro identifikaci jednotlivé vrstvy. Podle použité technologie je na spojové (linkové) vrstvě v OSI modelu paket zapouzdřen do rámce nebo buňky, potom je předán fyzické vrstvě teprve pak vyslán bit po bitu do komunikační sítě. Na druhé straně komunikační sítě provede příjemce rozbalování datových jednotek, předávání dat jednotlivým entitám vyšších vrstev a postupné skládání zprávy, případně i její prezentaci uživateli (odstraňují se postupně hlavičky jednotlivých vrstev, až zůstanou samotná data). Dále bude popsáno zpoždění na 3. vrstvě

(síťové) a nižší vrstvy v modelu OSI. Příprava zprávy k odeslání trvá nějakou dobu, proto dochází ke vzniku zpoždění, které jsou zdrojové kódování, paketizační, v odchozích frontách a vyrovnání jitteru tzv. jitter buffer.

U zdrojového kódování jsou zpracovávaný hovorové, zvukové a obrazové signály, které mají spoustu redundantních (obsahují větší množství dat než je nutné k přenosu, slouží také pro minimalizování ztrát) a irelevantních informací (jsou to nepodstatné složky informace, které se dají potlačit, dál nejsou přenášeny a je to nevratný proces, dojde tedy ke ztrátě informace). Vhodné kodeky dokážou tyto redundantní a irelevantní informace minimalizovat. Paketizační zpoždění je doba pro vytvoření paketu. Ovlivňuje ji hlavně počítač podle toho, jak dokáže informaci rychle zpracovat. Zpoždění v odchozích frontách je uplatňováno v mezilehlých síťových prvcích (směrovače, přepínače). Hodnota tohoto zpoždění se neustále mění podle zatížení prvku. Aplikace pracující v reálném čase potřebují, aby datové jednotky přicházely pravidelně tj. se stejným zpožděním. Proto najdeme v koncovém zařízení zabudovaný jitter neboli jitter buffer, který vyrovnává proměnlivé zpoždění. Tento buffer má omezenou velikost a pakety, které mají velký rozptyl zpoždění, jsou považovány za ztracené. Příkladem mohou být aplikace pracující v reálném čase. Pro přenos dat není jitter až tak moc důležitý, protože o skládání jednotlivých paketů dohromady je postaráno transportní vrstvou v modelu OSI.

2.2 Zpoždění v mezilehlých zařízeních

Pod pojmem mezilehlá zařízení si můžeme představit síťové (aktivní) prvky, které najdeme v komunikační síti mezi koncovými zařízeními. Jednotlivé prvky plní své úkoly podle toho na jaké vrstvě operují v modelu OSI. Začneme-li od nejnižší vrstvy tzv. fyzické, nachází se zde prvky pro obnovu signálu, zesílení. V linkové (spojové) vrstvě je prvek pro propojování různých segmentů sítě a snižuje zatížení sítě. Poslední síťová vrstva provádí směrování paketů do jiných částí sítě popřípadě do stejné podsítě. Snaží se vybrat co nejkratší (nejvýhodnější) cestu do požadovaného cíle. Nesmíme zapomenout také i na zabezpečení pro všechny zde zmíněné vrstvy. Nikdo z nás nechce, aby naše odeslána data byla odposlechnuta třetí osobou. Důležitým úkolem je i správa sítě, kde správce dohlíží na chod své sítě a případné problémy okamžitě řeší.

V následujících podkapitolách budou popsána jednotlivá mezilehlá zařízení, jakým způsobem pracují a jak latence na nich vzniká. Dále je zapotřebí ještě vysvětlit tyto následující latence mezi, které patří zpoždění ve vstupních frontách, doba potřebná pro zpracování informace, zpoždění v odchozích frontách. Každé rozhraní má hardwarovou frontu, která je typu FIFO (First In First Out) tzn., že první příchozí paket vstoupí do fronty, je zpracováván jako první a potom poslán do dalšího obslužného prvku. Před hardwarovou frontou najdeme softwarové fronty na, které můžeme aplikovat jakékoliv jiné podporované fronty.

Zpoždění ve vstupní frontě je doba, kterou paket (rámec) stráví ve vstupní vyrovnávací paměti (buffer), dokud nejsou načteny všechny potřebné bity pro zpracování a dokud se nedostane ke zpracování výkonnou jednotkou. Doba potřebná pro zpracování informace je doba, kterou potřebuje sítové zařízení k přečtení důležitých informace, jako je adresa příjemce, provést vyhledání v tabulce, zjistit výstupní port a následně předat paket do výstupní fronty. Rychlost zpracování je ovlivněna výkonem a typem zařízení. Dále při zvětšujícím provozu se bude zvyšovat i doba zpracování. U zpoždění v odchozí frontě má vliv také počet paketů směrovaných na stejný port. Z hlediska aplikování QoS služeb nelze dobu strávenou ve frontě přesně určit.

2.2.1 Opakovač (repeater) a rozbočovač (hub)

Tyto aktivní prvky jsou řazeny do nejnižší vrstvy modelu OSI tj. fyzické. Jejich úkolem, je přijmout symbolový prvek, zesílit ho či obnovit ho a odeslat na všechny ostatní porty kromě portu, od kterého signál přijaly (obdržely). Rámec, který je zpracováván, není načítán celý, symboly nejsou ukládány do vyrovnávací paměti a jeho zpoždění bereme pro opakovače i huby jako zanedbatelné, hodnota zpoždění je kolem 0,68 μs (pro CentreCom MR820TR) [14]. Opakovače jsou používány k prodloužení dosahu spojení v místech, kde dojde k překročení maximální povolené vzdálenosti pro daný typ kabelu v síti. Rozbočovače jsou využívány dále pro vytvoření mnohabodového fyzického spoje. Velkou nevýhodou u těchto zařízení je společná kolizní doména, kdy všechna připojená zařízení sdílí jeden fyzický komunikační kanál. Kolizní doména je tvořena více než dvěma uzly, jenž jsou propojeny prvky pracujícími pouze na fyzické vrstvě v modelu ISO/OSI. Ke kolizím většinou dochází, když začne více stanic vysílat zároveň nebo s takovým časovým odstupem, kdy před zahájením vlastního vysílání nejsou schopny detekovat cizí provoz na sdíleném komunikačním kanálu. Posílaná informace je znehodnocována a musí být vyslána znova. Pomocí přístupových metod typu CSMA (Carrier Sense Multiple Access), lze těmto kolizím částečně zabránit. S použitím některé CSMA metod, stanice kontroluje obsazenost sdíleného kanálu, jestli je volný, potom může začít vysílat. [11][19][21]

2.2.2 Přepínač (switch), most (bridge)

Přepínač či most jsou prvky pracující na druhé tj. linkové (spojové) vrstvě v modelu OSI. Hlavním jejich úkolem je přepínání rámců podle cílové MAC (Media Access Control) adresy na konkrétní port, který vede k cíli. Most může být buď dvou nebo víceportové zařízení. Přepínač je charakterizován počtem portů pro připojení jednotlivých segmentů a jeho přepínací schopnosti. Při zapojení nového přepínače nebo mostu do sítě nebo vymazáním jejich přepínací tabulky, začnou pracovat jako rozbočovače. Postupným přijímáním rámců se učí, odkud byl rámec přijat, na jakém portu, MAC adresu rámce, a také na jaký cílový port bude rámec odeslán. Pokud jsou ze začátku rozbočovačem, odesílají rámce na všechny porty, kromě portu odkud byl příchozí rámec přijat a čekají na odezvu příjemce, ostatní uživatelé na rámec nereagují. Postupně takhle dochází k naplnění jejich přepínací tabulky a snaží se zajistit pro každý komunikující uzel maximální možnou přenosovou kapacitu. Na každém portu vytvářejí vlastní kolizní doménu (pokud vůbec) a snižuje tak zátěž konkrétního fyzického spoje. Podle rychlosti přepínání, nastavení úrovně kontroly a úrovně zatížení se zpoždění průchodu rámce přepínačem mění. Přepínač nebo most může pracovat v některém ze třech režimů a to Cut-Through, Fragment Free (FF), Store & Forward (S&F).

V režimu Cut-Through není přijatý rámec kontrolován, provede se jen přečtení hlavičky s cílovou MAC adresou a hned je rámec poslán na výstupní port. U režimu Fragment Free přepínač čeká, až přijme prvních 64 B rámce, a pokud nebyla detekována kolize, pak rámec teprve přepošle na výstupní port. V posledním režimu Store & Forward je načten celý rámec. Zkontroluje se CRC kód celého rámce, jestli souhlasí pak přepínač přepošle rámec na výstupní port, jinak ho zahodí. [11][21]

Přepínače společnosti Cisco řada 29XX používají ke zpracování rámců metodu Store & Forward a k výsledné době přepnutí rámce na výstupní port je započítáno i serializační zpoždění vstupní linky pro příchozí rámec. V dokumentu [14] je provedeno testování různých typů přepínačů od společnosti Cisco a výsledky měření jsou uvedeny v tabulce, viz Tabulka 2.1. Celkové zpoždění přepínače (t_s) je definováno zpracováním přijatého rámce a přepnutí na výstupní port (t_p), spolu s hodnotou serializačního zpoždění pro přijatý daný rámec a linky s

přenosovou rychlostí 10 Mb/s. Síťové prvky jiných výrobců například přepínač HP ProCurve 2510 má dobu t_p maximálně 4,9 μs (hodnota je udávána pro rámec velikosti 64 B a linku 100 Mb/s) [20].

Tabulka 2.1: Zpoždění pro různé velikosti rámců v přepínačích (převzato z [14])

Síťový prvek	Režimy přepínání	98 B rámec		546 B rámec		1492 B rámec	
		Δt_s [μs]	Δt_p [μs]	Δt_s [μs]	Δt_p [μs]	Δt_s [μs]	Δt_p [μs]
Catalyst 1900 (switch)	FF	71,6	–	71,6	–	71,6	–
Catalyst 1900 (switch)	S&F	96,0	7,0	456,0	7,0	1210,0	7,4
Catalyst 2900 (switch)	S&F	101,0	14,0	461,0	12,0	1210,0	15,0
Catalyst 3550 (L3 switch)	S&F	118,0	28,0	516,0	70,0	1380,0	168,0

2.2.3 Vícevrstvý směrovací prvek – L3 přepínač (L3 switch)

L3 přepínač je hybridní zařízení, které obsahuje kombinaci směrovače a přepínače. Hardwarová část pracuje na principu přepínače, provádí už i směrování a softwarová část obsahuje méně funkcí, než které jsou nabízeny u klasického směrovače. Pracuje na třetí vrstvě v modelu OSI. Dokáže pracovat rychleji než dnešní klasické směrovače (kdy směrování je na úrovni softwaru) a hardwarová řešení jsou vždy rychlejší, proto je všechno z větší části implementováno už na hardwarové úrovni. S přijatým rámcem pracuje následovně, analyzuje celý obsah rámce, zjistí cílovou IP adresu a podle záznamu ve směrovací tabulce zjistí na jaký výstupní port odeslat rámec. Může dojít k situaci, že přepínač nemá přiřazenou IP adresu ke konkrétnímu portu a potom tento problém vyřeší zabudovaný směrovač. Směrovač určí, co se provede s rámcem a potom danou informaci uloží i do přepínací tabulky. Velikost zpoždění pro rámec o velikosti 98 B, 546 B a 1492 B jsou uvedeny v tabulce, viz Tabulka 2.1, výsledky jsou pro zařízení Cisco 3550. Hodnoty obsahují také serializační zpoždění pro linku 10 Mb/s. [8][14]

L3 přepínač je hlavně používán pro vysokorychlostní přenos dat přes lokální sítě. Existují a používají se přepínače, které pracuji i na vyšších vrstvách (L4-7 Switch). Přepojují data podle znalosti portů na transportní vrstvě nebo typu aplikačních dat. Zpoždění na těchto přepínačích je vyšší a nelze jej jednoznačně definovat, záleží na konkrétním použití. Tyto přepínače jsou hlavně používány pro loadbalancing různých serverů nebo dokážou filtrovat provoz podle pravidel, které nastavil administrátor.[7]

2.2.4 Směrovač (router)

Směrovač je zařízení, které propojuje sítě s různými technologiemi (Ethernet, WiFi, xDSL - x Digital Subscriber Line, ATM -Asynchronous Transfer Mode atd.), snaží se zajistit nejlepší trasu pro směrování paketů, odděluje domény se všesměrovým provozem. V modelu OSI tento síťový prvek pracuje ve třetí vrstvě tj. síťové. Jeho součástí je směrovací tabulka a různé druhy směrovacích protokolů například RIPv2 (Routing Information Protocol), EIGRP (Enhanced Interior Gateway Routing Protocol), OSPF (Open Shortest Path First) atd. Existují i směrovače poskytující další funkce, jako například překlad adres a portů (NAT), bezpečnostní funkce - firewally, DHCP server (Dynamic Host Configuration Protocol), VoIP, bezdrátové směrování atd. V modelu OSI tento síťový prvek pracuje ve třetí vrstvě tj. síťové. S jednotlivými pakety pracuje směrovač následovně, v hlavičce paketu si přečte cílovou adresu a snaží se najít odpovídající záznam v tabulce. Po vyhledání určí, zda bude paket

přesměrován do jiné sítě nebo zůstane v té sítí, odkud byl přijat. Jestli směrovač nenajde záznam o cestě kam paket poslat je buď zahozen, nebo je využita výchozí cesta (ve směrovací tabulce má tvar 0.0.0.0/0) a paket je pak poslán na rozhraní pro výchozí cestu a ven do sítě.

Velikost zpoždění u směrovače je ovlivňováno dobou pro načtení paketů, doby potřebné pro načtení paketu a doby pro nalezení cílové adresy ve směrovací tabulce a posláním na výstupní port (konkrétní rozhraní)[11][21].

Testováním páteřních směrovačů byly naměřeny hodnoty zpoždění mezi 10-20 μs. Směrovač firmy Cisco 12416 dosáhl průměrného zpoždění 17,7 μs, měření bylo provedeno pro paket o velikosti 40 B na lince s přenosovou rychlostí 2,5 Gb/s. U měření smíšeného internetového provozu, kde byla také použitá linka s přenosovou rychlostí 2,5 Gb/s jsou průměrné hodnoty od 65 do 252 μs (Cisco směrovač 12416 dosáhl průměrného zpoždění 252 μs)[23].

V síti s provozovaným protokolem IPv6 (Internet Protocol v6) bylo pro páteřní směrovače od firem Hitachi, NEC, Fujitsu a Juniper naměřeno zpoždění od 9 do 73 μs [22].

2.3 Zpoždění na přenosových médiích

Zpoždění zde ovlivňuje nejen použité přenosové médium, ale i fyzická délka trasy. Na zpoždění se tedy podílejí dvě složky, a to zpoždění vlivem konečné rychlosti šíření elektromagnetického signálu vedením či volným prostorem, a tzv. serializační zpoždění způsobené dobou potřebnou pro přenos fyzického bloku symbolů určité velikosti (obsahující rámec), jenž závisí na fyzické přenosové rychlostí kanálu dané vlastnostmi komunikačního kanálu (šířka pásma, útlumová charakteristika kanálu a úroveň rušení) a použitým typem modulace. Zpoždění při mezikontinentální komunikaci se pohybuje řádově kolem desítek až stovek milisekund. Fyzické vedení nejsou vedeny nejkratší cestou, ale mohou být umístěny podél železnic, silnic nebo na sloupech s elektrickým vedením nebo přes satelit. Lze tedy říct, pokud existuje nejkratší cesta, nemusí být vždy použita. O směru v síti většinou rozhoduje směrovač. Svá rozhodnutí provádí zhodnocením jednotlivých linek podle jejich rychlosti nebo vytíženosti. Dále podle použité technologie je ovlivněna přenosová rychlost a i serializační zpoždění. Serializační zpoždění je doba, kdy rámec prochází přes danou linku. Zpoždění je ovlivňováno velikostí rámce a přenosové rychlosti linky. Čím větší bude rámec nebo menší přenosová rychlost spoje, tím větší bude zpoždění. Každý aktivní síťový prvek, který zpracovává rámec, vnáší do přenosu toto zpoždění. Velikost serializačního zpoždění (t_{sz}), můžeme snadno vypočítat ze vztahu, viz (2.1).

$$t_{sz} = \frac{FS \cdot 8}{R} [\text{ ms}] \tag{2.1}$$

kde *FS* velikost rámce [B],
R přenosová rychlost spoje [kb/s].

Pro ukázku jsou zde vypočítány hodnoty serializačního zpoždění pro různé velikosti rámců a rychlosti přenosových linek viz Tabulka 2.2. Z vypočítaných hodnot serializačního zpoždění vidíme, že zpoždění má velký vliv na velké rámce při nižších přenosových rychlostech, ale u vyšších přenosových rychlostech v řádech gigabitů nemá velikost rámce vliv na zpoždění a je zanedbatelné.

Následující podkapitoly obsahují popis používaných přenosových médií a jejich charakteristické parametry.

Tabulka 2.2: Serializační zpoždění pro různé velikosti rámců a různé přenosové rychlosti

Velikost rámce [B]	Přenosové rychlosti					
	64 kb/s	1 Mb/s	10 Mb/s	100 Mb/s	1 Gb/s	10 Gb/s
64	8 ms	512 µs	51,2µs	5,12 µs	0,512 µs	0,0512 µs
512	64 ms	4,1 ms	0,41 ms	41 µs	4,1 µs	0,41 µs
1024	128 ms	8,2 ms	0,82 ms	82 µs	8,2 µs	0,82 µs
1500	187,5 ms	12 ms	1,2 ms	0,12 ms	12 µs	1,2 µs

2.3.1 Metalický kabel

V dnešních LAN i přístupových sítích můžeme nalézt metalické kabely typu, koaxiální kabel a kabely s kroucenými páry (dvojlinkami).

Koaxiální kabel má dobrou robustnost, pevnost, ale zato je málo ohebný. Jeho přenosová rychlost je 10 Mb/s. Díky své malé neohebnosti a přenosové rychlosti, je postupně nahrazován kroucenou dvojlinkou.

Kroucená dvojlinka je prodávána ve třech typech UTP (Unshielded Twisted Pair), STP (Shielded Twisted Pair) a FTP (Foiled Twisted Pair). UTP, STP a FTP kabel je tvořen čtyřmi páry vodičů, kde každý pár je zkroucený s různou délkou zkrutu. UTP kabel je lehčí, tenčí, levnější než stíněné kabely, instalace je jednodušší a nemá žádné stínění kolem vodičů. Aby dokázal blokovat elektromagnetické rušení, musí být během výroby zajištěna dokonalá symetrie jednotlivých párů v kabelu. Při zajištění dokonalé symetrie páru budou mít vybuzené parazitní proudy opačnou orientaci a dojde k jejich vyrušení. STP kabel má využití tam, kde hrozí velmi vysoká intenzita elektromagnetického rušení, je více obalen ochrannými plasty, každý pár vodičů je stíněn samostatně, je silnější, těžší a instalace je náročnější (musí být zachován minimální poloměr ohybu, aby nedošlo k poškození stínění). FTP kabel je stíněný okolo všech vodičů dohromady, je dražší než UTP kabel a levnější než STP kabel, má větší odolnost proti rušení, instalace je trochu náročnější než u UTP kabelu, ale jednodušší než u STP kabelu a uzemnění je obtížnější. Přenosová rychlost je 100 Mb/s a s použitím všech párů ve vodiči, lze dosáhnout rychlosti v gigabitech za sekundu. [1][29]

Udávané zpoždění pro UTP kabel cat5E, cat6 je 535 ns/100 m, cat7 439 ns/100 m, pro kabel STP je zpoždění u cat6A 500 ns/100 m, cat7 460 ns/100 m a poslední FTP kabel má zpoždění pro cat5E, cat6 535 ns/100 m. Všechny typy kabelů jsou značky Signamax.[4]

2.3.2 Optický kabel

Optický kabel je řazen mezi nejrychlejší a nejmodernější, můžeme ho nalézt na transportních sítích, ale postupně je implementován do přístupových sítí. Hlavní využití má v oblastech, kde je nutná velmi vysoká přenosová rychlost tzn., že většinou propojuje místa na velmi dlouhých vzdálenostech. Dále ho charakterizuje jeho vysoký nosný kmitočet kolem 10^{14} Hz, nízký útlum, nemá elektromagnetické vyzařování, odolný proti elektromagnetickému rušení a používá velkou šířku pásma. Používají se tři typy optických vláken, jsou to jednovidová, mnohovidová a plastová. U jednovidových vláken je signál přenášen jediným videm, jsou používána pro přenos dat na větší vzdálenosti (do 50 km), dosahují vysokých přenosových rychlostí řádově desítky Gb/s, cenově dražší než mnohovidová optická vlákna, křehčí a instalace je náročnější. Mnohovidová optická vlákna přenáší signál více vidy současně. Každý vid se šíří jinou trasou a nabírá tak různého zpoždění, což má za následek

rozptyl signálu a hrozí tak mezisymbolová interference. Proto jsou používány na kratších vzdálenostech (do jednotek km), přenosová rychlost je v jednotkách Gb/s, jsou cenově dostupnější a vyznačují se jednodušší instalací. Oba tyto typy optických vláken používají jádro z křemíku. Plastová optická vlákna mají jádro z plastu, jsou levnější než křemíková, využívají se do vzdálenosti několik metrů (do 50 m) a přenosová rychlost je řádově ve stovkách Mb/s. Velký vliv na přenosovou rychlost v optickém kabelu má index lomu v jádru vlákna, hodnota je přibližně 194 895 km/s, tj. 0,65·c [5]. Při přepočítání zpoždění na 100 m vyjde hodnota 513 ns. Když uvážíme, že některé transportní sítě měří přes několik tisíc kilometrů, potom zmíněné zpoždění může mít velký vliv na celkové zpoždění.[13][28]

2.3.3 Bezdrátový přenos

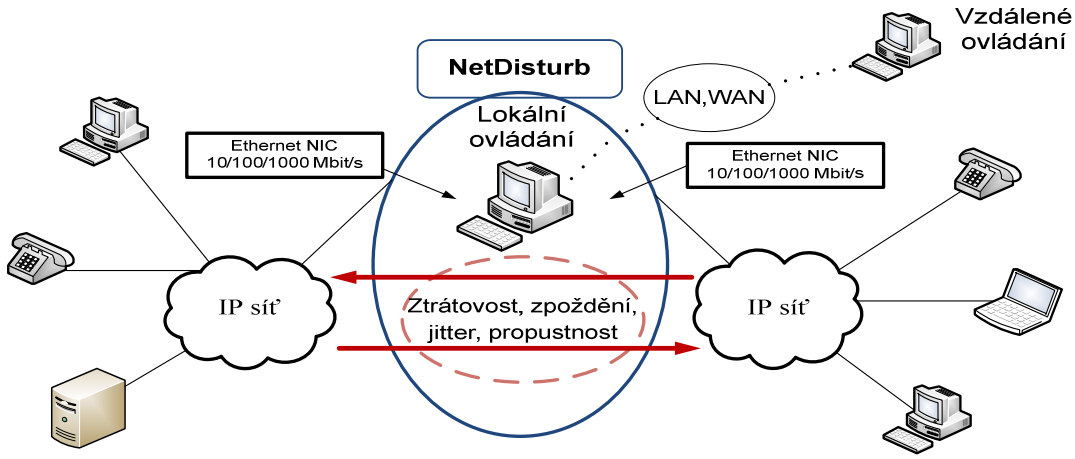
Bezdrátový přenos je v dnešní době hodně používán, zejména v místech kde nesmí být taženy kabely například chráněné historické památky a také je to občas cenově výhodnější. Mezi velkou výhodu tohoto přenosu patří, že pokud je uživatel v dosahu nějaké bezdrátové sítě může se odkudkoli připojit, je tedy zajištěna mobilita účastníka. Každý bezdrátový přenos potřebuje k provozu tři složky, jsou to vysílač, přenosové médium a přijímač. Přenosovým médiem se zde rozumí volný prostor. Mezi používané bezdrátové techniky patří UMTS (Universal Mobile Telecommunications System), Wi-Fi, WiMax (Worldwide Interoperability for Microwave Access), LTE (Long Term Evolution), mikrovlnné a satelitní spoje. Rozdíl mezi těmito technologiemi je v délce cesty šíření signálu. Technologie UMTS, Wi-Fi, WiMax, LTE využívají různé velikosti buněk pro přenos signálu. Maximální poloměr buňky má technologie WiMax (do 50 km) a ostatní technologie se pohybuji kolem 30 km. Zpoždění během přenosu u jednotlivých technologií je malé, je to dáno schopnosti signálu odrážet se a pronikat různými překážkami. Signály na nižších frekvencích mohou pronikat a odrážet se překážkami, signály na vyšších frekvencích jsou šířeny přímočaře, ale jsou ovlivňovány počasím například déšť, mlha, silný vítr. Rychlost šíření signálu je u těchto technologií přibližně stejný, tzn. zpoždění 334 ns/100 m.

Mikrovlnné spoje nabízí velkou šířku pásma při přímé viditelnosti, jsou používány většinou na přenos signálu do velkých vzdáleností (do 50 km, při větších vzdálenostech musí být postaveny retranslační stanice, které zvyšují dosah a přenosovou rychlost) a zpoždění je malé. Tento spoj najdeme na páteřních buňkových sítích nebo pro dálkový přenos televizních, telefonních sítí a satelitních spojů.

Komunikace u satelitního spojení je prováděna přes družici, která je umístěna na oběžné dráze Země a dochází k prodlužování přenosu signálu. Tento typ spoje nachází použití v oblastech nepokrytých pozemními sítěmi (například na moři, ve velmi řídce obydlených oblastech – velehory, pouště, polární oblasti, prales), ale také ve vojenství, u rozhlasového, televizního vysílání, meteorologických systémů, navigace atd. Najdeme zde spoustu komunikačních systémů, mezi nejznámější patří Inmarsat, Iridium, Globalstar, Intelsat, Eutelsat, GPS, GALILEO, VSAT. Pro více informací o těchto komunikačních systémech, naleznete v [24]. Družice mohou obíhat kolem Země ve třech oběžných drahách. První je LEO (Low Earth Orbit), která je ve výšce 700-1500 km nad povrchem Země a zpoždění signálu ke družici je 4-8 ms. Na druhé oběžné dráze MEO (Medium Earth Orbit) je umístění družic ve výšce nad 10 000 km a zpoždění příjmu signálu je do 100 ms. Poslední je GEO (Geostationary Earth Orbit) oběžná dráha a družice najdeme ve výšce 36 000 km, zpoždění signálu ke družici a zpět se pohybuje kolem 270 ms. V této výšce se družice jeví pro obyvatele Země nehybné, jejich doba oběhu je stejná s rychlostí otáčení Země.[21][25]

3 EMULÁTOR SÍTÍ NETDISTURB

NetDisturb je aplikace, která dokáže emulovat reálné vlastnosti datové sítě a umí zhoršit procházející provoz přes IP sítě (IPv4 a IPv6). Zhoršením je zde myšleno nastavením různého zpoždění, latence, jitteru, omezení šířky pásma, ztrátovosti paketu, kopírování a různé změny nastavení v paketech. Hlavním účelem aplikace NetDisturb je testování různých komunikačních služeb a sledovat jejich chování při zhoršených podmínkách v síti. NetDisturb slouží jako most mezi dvěma ethernetovými segmenty a umožňuje buď jednosměrný, nebo obousměrný přenos paketů na síťových kartách typů Ethernet, Fast Ethernet a Gigabit Ethernet. Jeho součástí je také uživatelské rozhraní pro server a klienta. Obě tato rozhraní mohou být spouštěna nejen na jednom počítači (NetDisturb Server a NetDisturb Client), ale i na dvou počítačích, přičemž u druhé varianty na jednou počítači poběží serverová část (NetDisturb Server, dále jen server) a na druhém klientská část (NetDisturb klient, dále jen klient) viz Obrázek 3.1. Uživatel na klientské části pak musí vytvořit spojení se serverem pomocí jeho IP adresy a portu, na kterém budou spolu komunikovat, po ověření těchto údajů, je spuštěno uživatelské rozhraní pro klienta.



Obrázek 3.1: Princip aplikace NetDisturb

Z hlediska softwarového a hardwarového vybavení počítače pracuje NetDisturb, jak na 32- nebo 64- bitových operačních systémech typu Windows XP, Vista, 7, Server 2003 nebo 2008. Pro provoz potřebuje nejméně 1GB paměti RAM (Random Access Memory), 20 MB volného místa na disku a rozlišení monitoru nejméně 1024x768. Další důležitým hardwarem je mít dvě stejné síťové karty a to jak pro Ethernet, Fast Ethernet a GigabitEthernet.

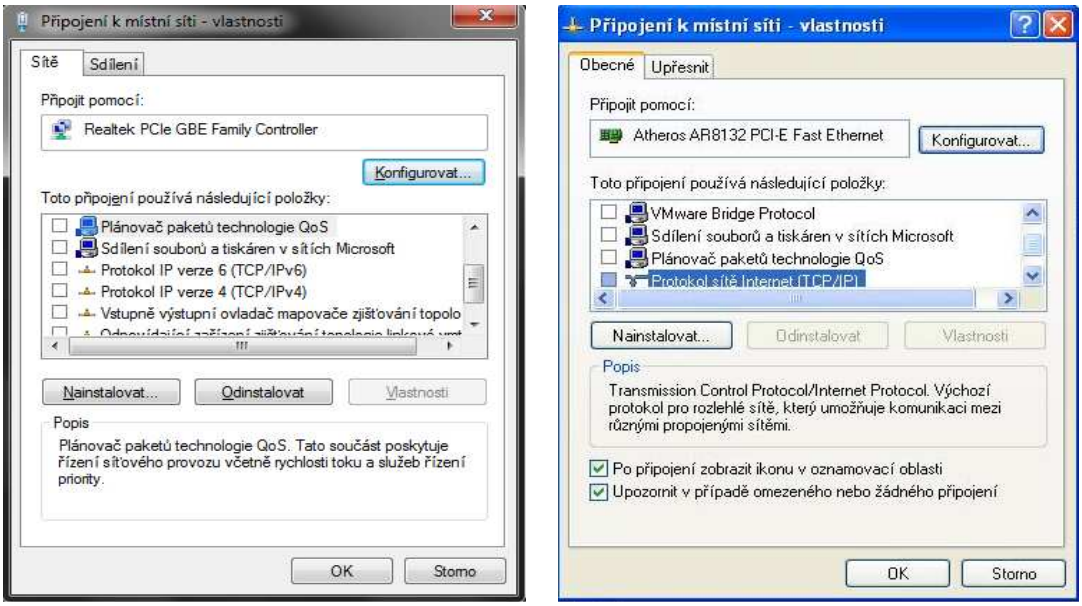
Tento software je k dostání ve dvou verzích a to NetDisturb Standard Edition a NetDisturb Enhanced Edition. Obě tyto edice nabízí například podporu protokolu IPv4 a IPv6, výměnu zpráv mezi serverem a klientem, která probíhá pomocí protokolu SOAP (Simple Object Access Protocol), který používá XML (eXtensible Markup Language) formát a HTTP (HyperText Transfer Protocol) protokol, jednoduché uživatelské rozhraní, ovlivnění jednosměrného nebo obousměrného provozu, umožňuje sloučit ovlivňované provozy do jednoho provozu, kde budou sdílet zpoždění a jitter, jsou zde předefinované filtry na základě hlaviček protokolů (například TCP, UDP, IP atd.), nebo je možnost si vytvořit vlastní filtry, můžeme si zobrazit i statistiky spuštěných provozů a uložit si je do souboru. Toto je jen pár příkladů co edice umožňují nastavit a sledovat. Edice Enhanced je odlišná od edice Standard v tom, že dokáže zhoršit provoz podle konkrétních protokolů například ARP Address

Resolution Protocol), FTP, HTTP, SIP a poskytuje podrobnější výpisy o spuštěných aplikací a jejich zhoršení provozu podle pravidel vytvořených uživatelem. Informace o více možnostech využití edice Enhanced, a jaké další protokoly podporuje, naleznete v [26].

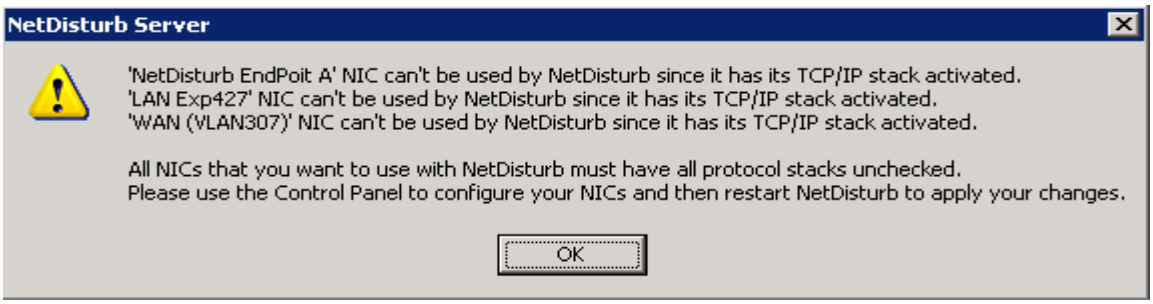
Ze strany praktického využití dokáže NetDisturb testovat aplikace jako VoIP, streamované video, streamované audio, IPTV, služby a aplikace pracující v reálném čase, provádět simulace symetrických nebo asymetrických podmínek v síti (například jitter, latency, šířku pásma atd.), provádět testování odolnosti aplikačních protokolů u služby Triple Play (obsahuje služby pro přenos dat, videa a hlasu) u set-top boxů přes DSL spojení například u VoIP, kde můžeme testovat jestli byly přeneseny zprávy SIP REGISTER (provede se registrace do ústředny) nebo SIP INVITE (slouží k poslání žádosti protější straně o hovor) na které byly aplikovány zpoždění nebo ztrátovost. Je toho hodně, kde se dá NetDisturb použít, zde bylo vypsáno jen pár příkladů, další naleznete v [26].

3.1 Uživatelské rozhraní NetDisturb Server

Předtím, než se nám spustí uživatelské rozhraní serveru, je důležité vypnout všechny síťové protokoly a služby na síťových kartách. Při nevypnutí by docházelo k ovlivňování našeho provozu. Vypnutím je zde myšleno odškrtnutí jednotlivých protokolů a služeb. Tento pokyn můžeme provést pro operační systém Windows XP následovně *Start – Ovládací panely – Připojení k síti a Internetu – Síťová připojení* – vybrat připojení k místní, které se používá pro připojení do Internetu, pak kliknout pravým tlačítkem myši a dát *Vlastnosti* a v kartě *Obecné* odškrtnout všechny síťové protokoly a služby viz Obrázek 3.2 nebo kliknutím pravým tlačítkem myši na hlavním panelu na ikonku problikávajících počítačů a dát *Otevřít síťová připojení*, potom vybrat připojení k místní a pokračovat stejným způsobem, viz výše. U Windows 7 provedeme nastavení následovně *Start – Ovládací panely – Síť a Internet – Centrum síťových připojení*, vybrat *Připojení k místní síti*, kliknout na tento odkaz, na kartě *Obecné* kliknout na položku *Vlastnosti* potom na kartě *Sítě* odškrtnout všechny protokoly a služby viz Obrázek 3.2 nebo kliknutím pravým tlačítkem myši na hlavním panelu na ikonku problikávajících počítačů. Nesplněním těchto požadavků bude NetDisturb zobrazovat varovné hlášení viz Obrázek 3.3. Pokud je vše nastaveno můžeme spustit NetDisturb Server pomocí zástupce umístěného na ploše nebo *Start – Všechny programy – NetDisturb – NetDisturb Server*. Po spuštění, bude zobrazeno uživatelské rozhraní serveru, viz Obrázek 3.4 a je zde vidět pro ukázku spuštěn provoz.



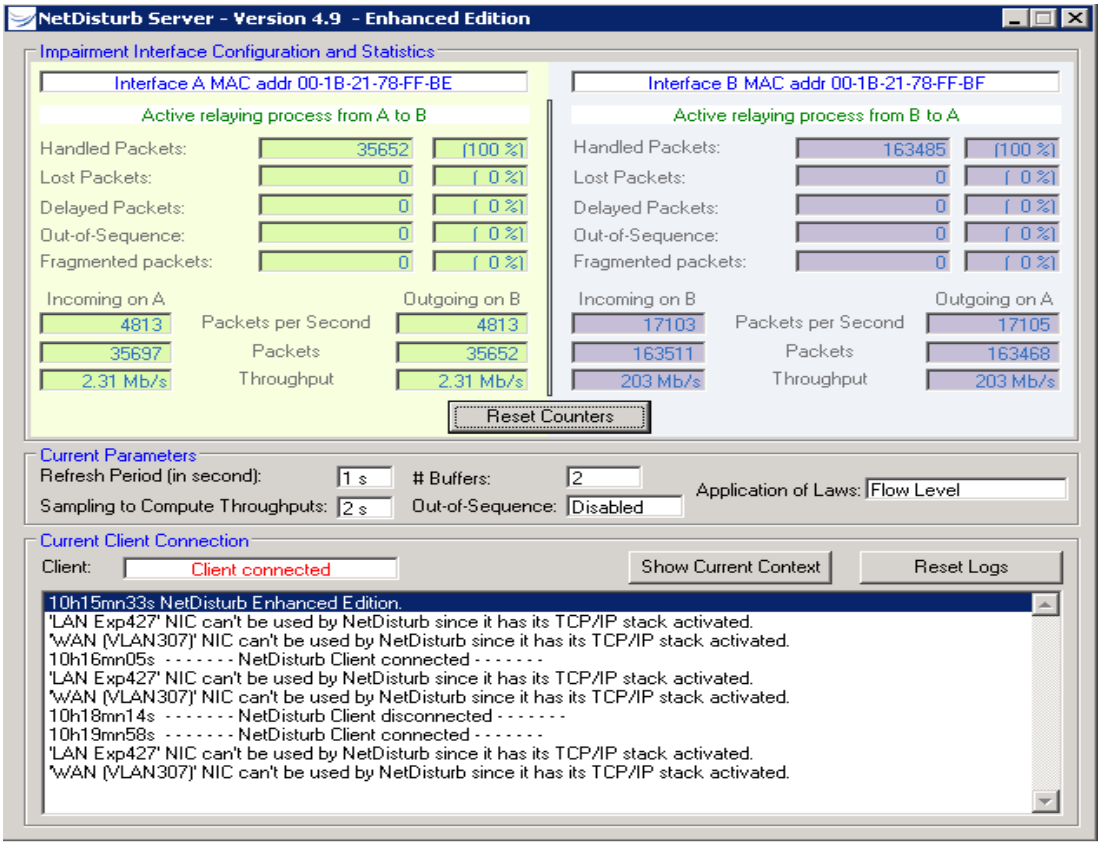
Obrázek 3.2: Odškrtnutí síťových služeb a protokolu pro OS Windows 7 a Windows XP



Obrázek 3.3: Varovné hlášení při nesplnění podmínek pro spuštění NetDisturb Server

Na uživatelském rozhraní serveru viz Obrázek 3.4, můžeme vidět nastavené síťové karty a jejich MAC adresy. Pro každý směr dostaneme informaci, s kolika pakety bylo pracováno (*Handled Packets*), kolik bylo ztraceno paketů (*Lost Packets*), zpožděno paketů (*Delay packets*), kolik paketů došlo v jiném pořadí, než jak byly vyslány (*Out-of-sequence*) a fragmentováno - kolik paketu bylo odmítnuto, protože nebyly označkovány například podle druhu aplikace a čísel portu (*Fragmented Packets*). Dále vidíme, kolik paketů přišlo na příchozí rozhraní A za sekundu a kolik jich bylo odesláno na odchozí rozhraní B (*Packets per Second*), celkový počet přijatých paketů (*Packets*) a přenosovou rychlost (*Throughput*). Tlačítkem *Reset Counter* budou vymazány všechny zobrazené informace, ale klient není tímto pokynem ovlivněn. V části *Current parameters* jsou parametry, které nastavuje klient na uživatelském rozhraní softwaru NetDisturb. Položky *Refresh Period*, *Sampling to Compute Throughputs* a *Buffers* najdeme na klientovi v části 3 s názvem *Configure NIC* a také na, viz Obrázek 3.10, kde jsou vysvětleny. Potom položky *Out-of-Sequence* a *Application of Laws* nastavíme u klienta v části 1 u nabídky *Working Modes*, jsou tam tyto nastavení popsány, co provádějí. Položka *Out-of-Sequence* bude nastavena buď na Enabled nebo Disabled, definuje, jak pakety budou přijímány. Dále položka *Application of Laws* určí, jak budou pakety zpracovány a v jejím okně uvidíme buď *Flow level* pro *Laws apply to the IP Flow* nebo *Per Connections* pro *Laws apply to each TCP/UDP connection of the IP Flow*. Tato nastavení

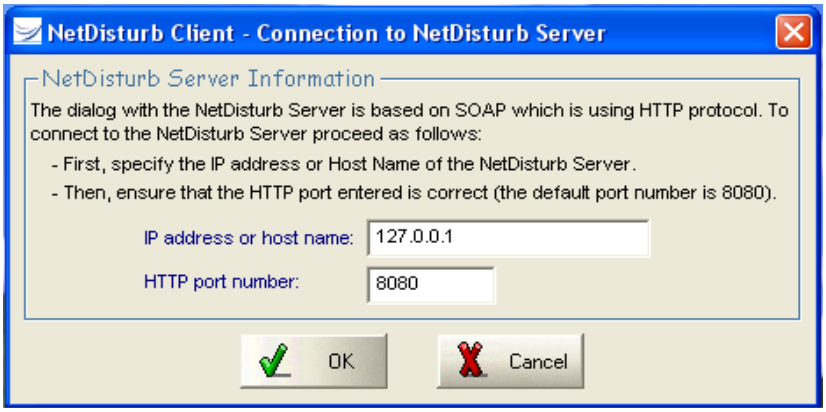
jsou vysvětlena u klienta v části 1 u nabídky *Working Modes*. Jestli je klient spuštěn poznáme informační hlášením zobrazeným červeně *Client connected.* Pod touto informací je prováděn výpis, v kolik hodin byl klient připojen nebo k jeho ukončení komunikace se serverem, potom jaké rozhraní nebude použito aplikací NetDisturb a proč, protože jsou aktivní TCP/IP protokoly. Tyto výpisy můžeme zobrazit kliknutím na tlačítko *Show Current Context* a jejich vymazání tlačítkem *Reset Logs*.[26]



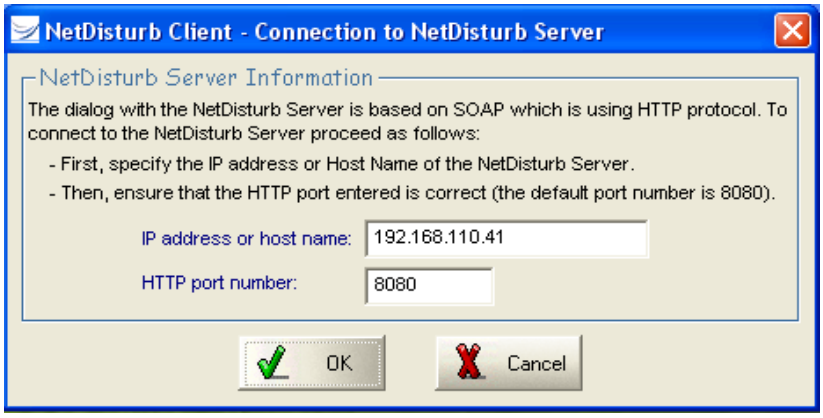
Obrázek 3.4: Uživatelské rozhraní NetDisturb Server

3.2 Uživatelské rozhraní NetDisturb Client

Uživatelské rozhraní klienta spustíme přes zástupce umístěného na ploše nebo kliknutím na *Start – Všechny programy – NetDisturb – NetDisturb Client*. Potom dojde k zobrazení dialogového okna, kde musí být zadána IP adresa a port pro komunikaci se serverem. Jestli je server nainstalován spolu s klientem na jednom PC stačí zadat následující IP adresu a port viz Obrázek 3.5. Pokud je klient nainstalován na jiném PC, musí být zadána IP adresa počítače, na kterém běží server a port zůstává stejný, viz Obrázek 3.6 (Pozn. Musí se zkontrolovat, zda není port 8080 na serveru blokován firewallem).



Obrázek 3.5: Připojení klienta na server na lokálním PC



Obrázek 3.6: Připojení klienta na vzdálený server

Při špatně zadané IP adrese a portu dojde k zobrazení hlášení o chybě spojení mezi klientem a serverem viz Obrázek 3.7. Po správném nastavení je uživateli zobrazeno uživatelské rozhraní klienta (pro ukázku je uživatelské rozhraní částečně naplněné) a pro následující popis bude uživatelské rozhraní klienta rozděleno do pěti částí (číslováno od 1 do 5) viz Obrázek 3.8.

Část 1 obsahuje klasické aplikační menu. Najdeme, zde nabídky *File*, *Edit*, *Actions*, *Working Modes*, *Statistics*, *Help* a *Show (Hide) Aggregates*. Nabídka *File*, obsahuje položky pro vytvoření nového souboru, ukládání, načtení souboru a nedávno otevřené soubory. Soubory obsahující konfiguraci testovacího scénáře jsou ukládány s příponou *.wsx* a jsou uloženy u uživatele ve výchozím adresáři *C:\Program Files\NetDisturb\NetDisturb Client*. Nabídka *Edit* nabízí možnosti kopírování, vkládání, mazání, nastavená pravidla pro filtrování vrátit do výchozího stavu, přesun mezi jednotlivými filtrovanými pravidly nahoru a dolů. Nabídce *Actions* bude věnována pozornost v části 3. V nabídce *Working Modes* jsou na výběr čtyři možnosti jak ovlivňovat provoz, sice jsou po prvním spuštění klienta vybrány, ale bude vhodnější popsat, co každá možnost ovlivňuje. První dvě jsou *Enable Out-of-Sequence Packets (Internet-like)*, který dovoluje přijímat pakety v libovolném pořadí a *Disable Out-of-Sequence Packets (Ethernet-like)*, umožnuje přijímat pakety v tom pořadí jak byly vyslány. Jedna z těchto dvou možnosti musí být vybrána. Zbylé dvě možnosti ovlivňování provozu jsou *Laws to be applied to the Flow*, provádí kontrolu paketů, zda patří do vytvořených filtrů, pokud ano proběhne zpracování, jestli ne, paket bude zahozen a *Laws to be applied to each*

TCP/UDP connection of the Flow, analyzuje každý IP paket podle protokolu, IP adresy, portu a potom je pak zařazen buď do transportního protokolu TCP nebo UDP. Ze zbylých dvou možností musí být také jedna vybrána. Nabídka *Statistics* umožňuje pomocí položky *Start* uložit nastavení jednotlivých filtrů. Uloženou statistiku lze opětovně načíst, ale nelze měnit její obsah, což je nevýhoda. Položkou *Stop* je ukončeno ukládání a položka *Configuration* definuje, jaký filtr má být uložen (číslováno od 1 do 17, kde 17 má označení *Other flows to impair without using filters*), dále jaké pakety mají být uloženy (myslí se tím ztracené, odchozí, příchozí, zpožděné pakety). Nabídku *Help* není důvod vysvětlovat, zná ji asi každý. Poslední nabídka *Show (Hide) Aggregates* umožňuje sdružovat jednotlivá nastavená pravidla filtrování do jednoho provozovaného toku.

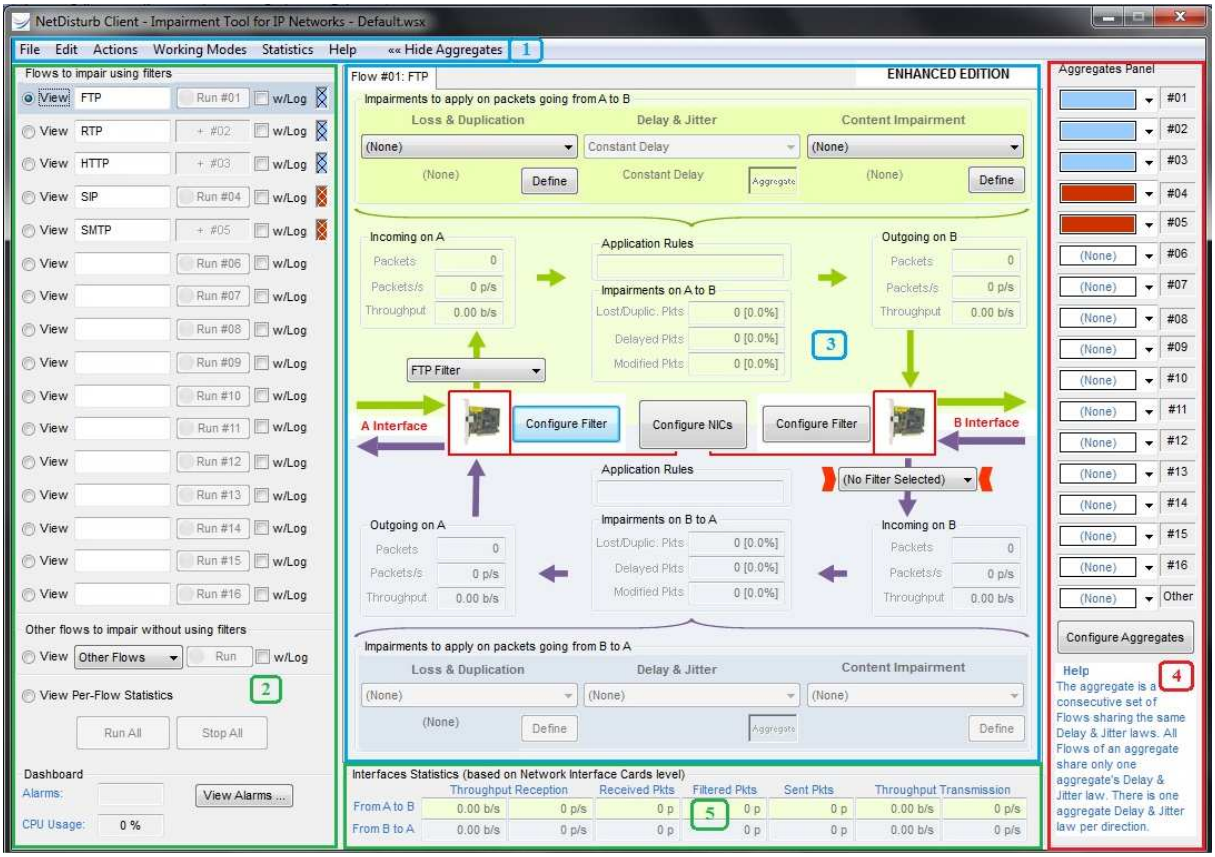
Obrázek 3.7: Hlášení o chybě spojení mezi klientem a serverem

V části 2 můžeme libovolně ovlivňovat vstupní tok pomocí různých protokolů, které používají služby pro jejich provoz. Lze vytvořit až 16 různých konfigurací pro ovlivňování dílčích částí testovaného provozu, dále obsahují editační okno pro pojmenování, tlačítkem *Run* a *Stop* spustíme nebo zastavíme toky. Kliknutím na *View* si zobrazíme charakteristiku jednotlivých toků, zaškrtnutím *w/Log* budou zobrazeny podrobnější informace o nastaveném toku, ale až dojde k jeho spuštění. Získané informace můžeme uložit do souborů s příponou *.pcap* a *.txt*. Soubor s příponou *.pcap* bude obsahovat všechny zachycené pakety a v souboru s příponou *.txt* budou uloženy informace, které ovlivňovaly zhoršení provozu pro každý paket, například zpoždění, ztrátovost atd. Tyto soubory je možnost opětovně otevřít. Podrobnější výpis o každém nastaveném toku je jen u edice Enhanced. Barevné označené okno říká, k jakému sdruženému toku tok patří, vyšrafované okno znamená, že nejsou nastavená síťová připojení, viz Obrázek 3.8 nebo nejsou nastaveny filtry u toků, viz Obrázek 3.9, po těchto nastavení šrafování zmizí. Je zde i 17 s názvem *Other flows to impair without using filters*, který umožňuje spravovat provoz podle rámců (*Other Flows*) nebo paketů (*Other IP Flows*). Kliknutím na *View Per-Flow Statistics* bude zobrazena aktivita všech toků (propustnost v příchozím a odchozím směru, totéž u paketu, zpoždění, ztrátovost, modifikované pakety a v procentech kolik paketů bylo filtrováno) a také zda vytvořenými toky prochází námi zvolený provoz nebo ne. Tlačítkem *Run All* jsou spuštěny všechny toky najednou, kromě těch u kterých není nastaveno nějaké pravidlo pro filtrování a tlačítkem *Stop All* jsou zastaveny všechny toky. Dále *Alarms* informuje uživatele o chybách varovným hlášením *Warning* , které vznikly při spuštění na síťovém připojení například přetečení bufferu, chyby během kontroly CRC (Cyclic Redundancy Check), tlačítkem *View Alarms* tyto chyby zobrazíme. Posledním je *CPU Usage* udává informaci o vytížení procesoru.

Část 3 slouží k nastavování zpoždění, jitteru, ztrátovosti, vše je pro pakety a to jak ve směru od A do B tak i obráceně. Všechna tato nastavení je možnost aplikovat pro námi vytvořené toky zcela nezávisle. Jednotlivá nastavení nemusí být stejná pro směry A do B nebo z B do A, mohou se lišit podle uživatele, jaké informace potřebuje zjistit ze zpracovávaného

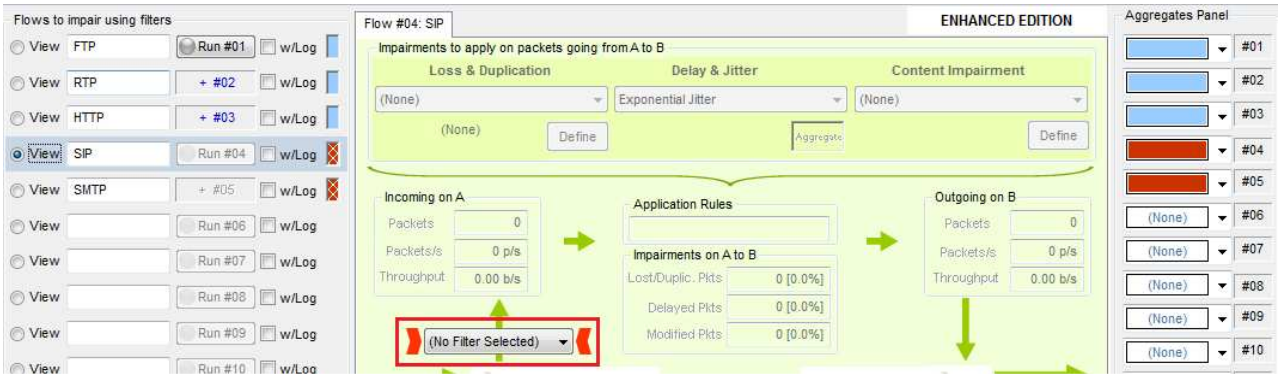
provozu. Při spuštění je uprostřed třetí části tlačítko *Configure NIC*, zde provádíme nastavení jednotlivých síťových karet a to pak v oblasti *Interface Selection*, kde provedeme výběr síťových připojení pro každé rozhraní, tedy A i B, po výběru uvidíme i jejich MAC adresy, které jsou zobrazeny v serveru, viz Obrázek 3.4. Nastavení, lze také provést přes nabídku *Actions* a položku *Configuration*, umístěnou v aplikačním menu. V zobrazeném dialogovém okně viz Obrázek 3.10, kde je prováděna konfigurace síťového připojení můžeme pak ještě nastavit, kdy mají být aktualizované informace na uživatelském rozhraní – *Refresh Period*, dobu odebírání vzorků pro výpočet propustnosti – *Sampling period for the throughput calculation*. Dále je na výběr zda propustnost bude okamžitá (*Instant Throughout*) nebo s odebíráním vzorků (*Average Throughout using Sampling Mechanism*), v jakých jednotkách se budeme pohybovat, jestli v kb/s (1 kb/s = 1000 b/s) nebo Ki/s (1 Ki/s = 1024 b/s). V položce *number of buffers containing the law values* provádíme nastavení počet vyrovnávacích paměti pro hodnoty, které chceme aby NetDisturb sledoval například zpoždění, ztrátovost a používá se pro každé vytvořené spojení. Minimální hodnota vyrovnávací paměti je 2 a maximální hodnota 100. Jedna vyrovnávací paměť může obsahovat až 20 480 hodnot. Tohle nastavení je používáno jen u možnosti *Laws to be applied to each TCP/UDP connection of the Flow* v nabídce *Working Modes* v části 1. Všechny tyto nastavené údaje jsou odeslány do serveru a server podle těchto informací pracuje.

Ve 4. části provádíme sdružování různých toků do jednoho provozovaného toku. Kliknutím na tlačítko *Configure Aggregates*, provádíme konfiguraci jednotlivých agregátů. Je možnost si nadefinovat až 8 různých agregátů, které barevně odlišíme od ostatních. Pro každý agregát si můžeme nadefinovat zpoždění, jitter a to buď ve směru z A do B nebo naopak. Po vytvoření agregátu a přiřazením různých služeb bude každého zajímat, jak nastavit parametry ovlivňující kvalitu služeb QoS. Tento příklad vše objasní. Použijeme například protokoly FTP, RTP, HTTP, které si předem připravíme. Jestli tyto protokoly vytvoříme v pořadí 1. FTP, 2. RTP, 3. HTTP a potom všechny přiřadíme do stejného agregátu (označíme ho například modrou barvou, viz Obrázek 3.9), bude mít přednost protokol FTP před ostatními, je mu totiž udělena nejvyšší priorita. Zjednodušeně priorita je zde v intervalu od 1 do 17, přičemž jednička má nejvyšší prioritu. Ve výsledku pro správný provoz, může být pořadí následující 1. RTP, 2. HTTP, 3. FTP. Přidělením protokolů do agregátu, dochází i ke změně tlačítek Run a Stop. Prvnímu protokolu tlačítko Run a Stop zůstává, ale pro další protokoly jsou tlačítka změněna například na + #2, + #3.[26]

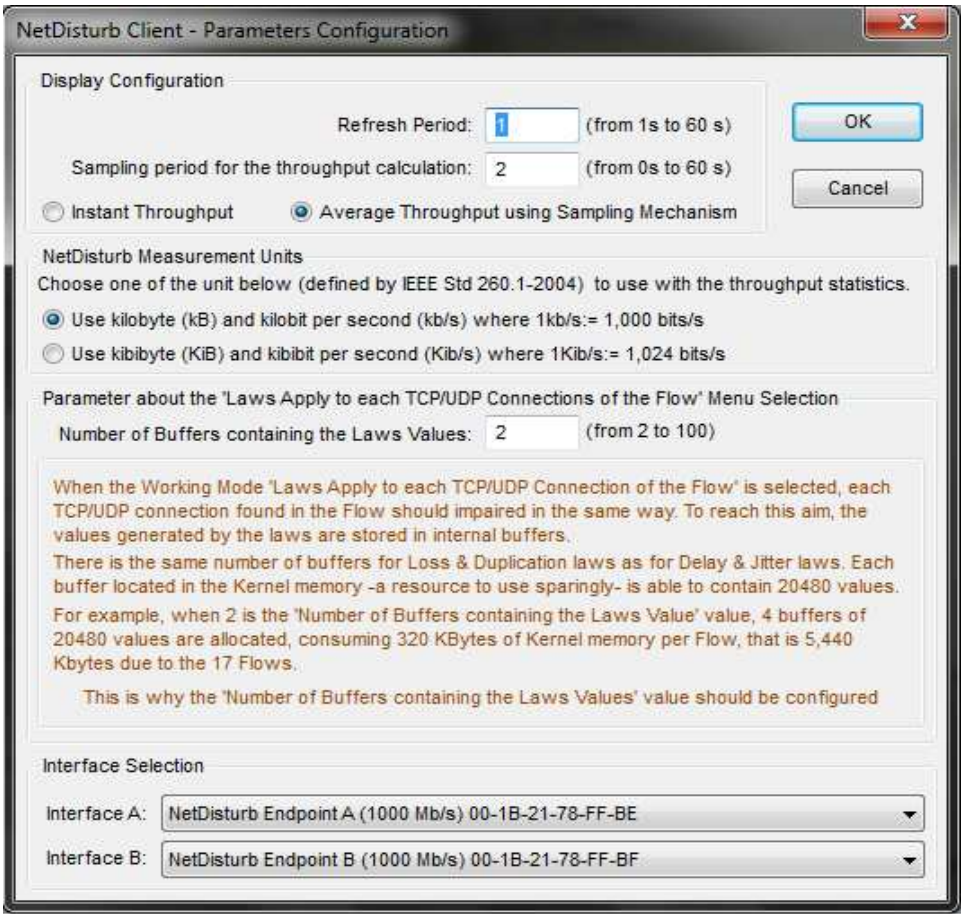


Obrázek 3.8: Uživatelské rozhraní NetDisturb Client

Poslední 5. část informuje uživatele o stavu síťových rozhraní, tedy o propustnosti, přijatých, filtrovaných a poslaných paketech.



Obrázek 3.9: Ukázkový příklad a chyba u vybraného agregátu



Obrázek 3.10: Konfigurace síťových rozhraní

4 ZÁKLADNÍ VLASTNOSTI PROTOKOLU TCP

Protokol TCP najdeme v síťovém modelu TCP/IP a obsahuje pouze 4 vrstvy dle modelu ISO/OSI. Do modelu TCP/IP patří vrstva síťového rozhraní, síťová vrstva, transportní vrstva a aplikační vrstva. Vrstva síťového rozhraní umožňuje přístup k fyzickému médiu a je závislá na použité přenosové technologii například Ethernet, FDDI (Fiber Distributed Data Interface), Token Ring. Síťová vrstva je realizována protokolem IP. Je to nespolehlivý protokol. Pro posílání datových jednotek používá datagramy, které cestují nezávisle na sobě v síti, a není zaručeno, že datagram dojde k příjemci. Během cesty může dojít k jeho ztrátě (vyprší doba jeho života, která je nastavena v jeho záhlaví v poli TTL - Time to Live, aby nedocházelo k zahlcení sítě), nebo k příjemci dorazí několikrát a neručí se také za správné pořadí datagramů. Hlavním úkolem síťové vrstvy je, aby se jednotlivé pakety dostaly od odesílatele k příjemci v co nejkratší možné době. Na transportní vrstvě najdeme protokoly TCP a UDP. Protokol UDP je nespolehlivý protokol a využívají ho hlavně aplikace, které si spolehlivost přenosu nepřejí například DHCP protokol pro přidělování IP adres počítačům, SNMP (Simple Network Management Protocol) protokol pro sledování, vyhodnocování a správu sítě. O protokolu TCP bude zmíněno později. Poslední vrstvou modelu TCP/IP je aplikační vrstva. Entitami této vrstvy jsou aplikační programy, které plní pokyny uživatele a pro svůj přenos dat využívají protokoly TCP nebo UDP.

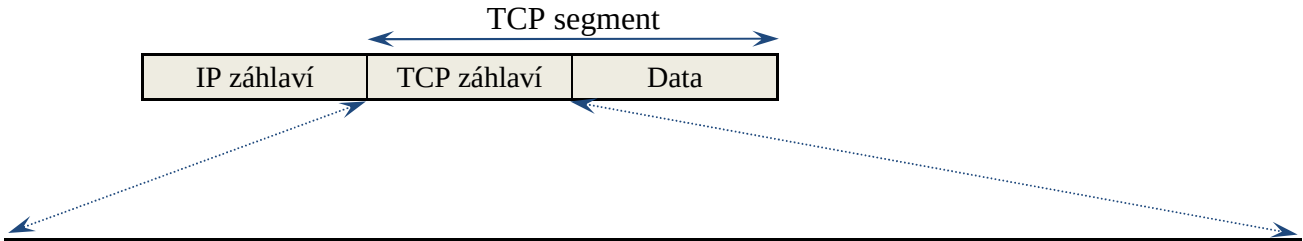
Protokol TCP vytváří virtuální dvoubodové duplexní spojení, kdy se před každým vysíláním datových jednotek sestaví spojení mezi vysílačem a přijímačem vznikne tak tzv. virtuální okruh. Pro vytvoření spojení mezi aplikací odesílatele a aplikací příjemce musí být jednoznačně určena IP adresa, číslo portu a použitý transportní protokol. Číslo portu může nabývat hodnot 0 až 65535. Na transportní vrstvě jsou přijatá data od aplikační vrstvy rozdělena na segmenty. Každý segment obsahuje TCP záhlaví a část rozdělených dat. Množství aplikačních dat, které může každý TCP segment přenést definuje parametr MSS (Maximum Segment Size). TCP segment je potom vložen do IP datagramu, kde velikost maximální velikost IP datagramu je 65535 B (je to maximální hodnota pole „délka IP datagramu“ o 16 bitech), musí být hodnota parametru MSS nastavena na MTU minus velikost TCP a IP záhlaví. Hodnota MTU (Maximum Transmission Unit) označuje maximální velikost paketu a jeho hodnota závisí na použité síťové technologii, viz Tabulka 4.1. Pokud TCP segment, který bude vkládán do IP datagramu je větší než hodnota MTU, musí IP protokol provést fragmentaci (rozdělení na menší části) IP datagramu. Je proto důležité vhodně vybírat hodnoty MSS, protože nízká hodnota způsobí, že využití sítě bude nízké. U větší hodnoty MSS dochází ke snižování výkonnosti vytvářením spousty IP datagramů, které pak nemohou být okamžitě potvrzeny nebo přeposlány.

Tabulka 4.1 Velikost MTU podle použité síťové technologie

Síťová technologie	MTU [B]
Ethernet II	1500
Frame Relay	1600
ATM	48
16 Mb Token Ring	17914
PPP	296
FDDI	4478
IEEE 802.3/802.2	1492

4.1 Záhlaví TCP protokolu

TCP segment je složen z TCP záhlaví a Data. Struktura TCP záhlaví je uvedena na Obrázek 4.1. Záhlaví obsahuje 11 polí, kde 10 polí musí být vždy definováno a 11. pole je volitelné (není vyžadováno).



Bity	0-3	4-7	8-15	16-31
0	Zdrojový port (source port)			Cílový port (destination port)
32	Pořadové číslo odesílaného bajtu (sequence number)			
64	Pořadové číslo potvrzovaného bajtu (acknowledgement number)			
96	Délka záhlaví (data offset)	Rezerva (reserved)	CWR ECN URG ACK PSH RST SYN FIN	Délka okna (window size)
128	Kontrolní součet (TCP checksum)			Ukazatel naléhavých dat (urgent pointer)
160,192	Volitelné položky záhlaví (options - optional)			

Obrázek 4.1: TCP záhlaví

1. **Zdrojový port** (16 b) – identifikuje port odesílatele TCP segmentu
2. **Cílový port** (16 b) – identifikuje port příjemce TCP segmentu, většinou nebývá shodný se zdrojovým
3. **Pořadové číslo odesílaného bajtu** (32 b) – definuje pořadové číslo prvního oktetu v TCP segmentu (pokud je příznak SYN nastaven na „0“). Jestliže je příznak SYN nastaven na „1“, jde o inicializační sekvenční číslo (ISN - initial sequence number) a první datový oktet má hodnotu ISN+1.
4. **Pořadové číslo potvrzovaného bajtu** (32 b) – používá se, když je nastaven příznak ACK. Číslo vyjadřuje hodnotu dalšího očekávaného oktetu, který je příjemce připraven přijmout, tzn., že příjemce přijal správně poslední přijímaný oktet například 200, ale pole bude obsahovat hodnotu 201.
5. **Délka záhlaví** (4 b) – určuje délku TCP záhlaví. Minimální délka záhlaví je 20 B, maximální 60 B a informuje, kde je začátek přenášených dat v segmentu.
6. **Rezerva** (4 b) – pole je vyhrazeno pro budoucí použití a bity musí být nastaveny na nulu.
7. **Příznakové bity** (8 b) – jsou to kontrolní bity a při nastavení na „1“, je jejich význam následující:

- **CWR** (Congestion Window Reduced) – 1 b, je využíván pro zmenšení okna zahlcení a bude aktivní, když odesílatel příjme data s příznakem ECE nastaveným na „1“.
- **ECE** (ECN-Echo) – 1 b, je nastaven u 3 fázové inicializace spojení tzv. Three-Way Handshake a indikuje, že TCP spojení je kompatibilní s ECN (RFC 3168).
- **URG** (urgent) – 1b, TCP segment nese naléhává data.
- **ACK** (acknowledgement) – 1 b, indikuje platnou hodnotu v poli potvrzovaného oktetu, tj., že segment potvrzuje přijetí dat.
- **PSH** (push function) – 1 b, indikuje, že TCP segment nese aplikační data, které budou předány u příjemce aplikace.
- **RST** (reset the connection) – 1 b, odmítnutí spojení.
- **SYN** (synchronize sequence number) – 1 b, odesílatel začíná novou sekvenci číslování, tj. TCP segment nese pořadové číslo prvního odeslaného oktetu.
- **FIN** (no more data from sender) – 1 b, je používáno pro ukončení spojení.

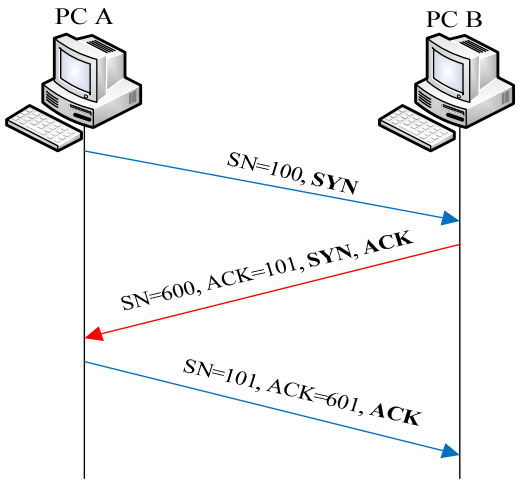
8. **Délka okna** (16 b) – definuje maximální množství oktetů, které může vysílač odeslat k příjemci, aniž by čekal na potvrzení jednotlivých segmentů. Velikost není pevně stanovena, mění se dynamicky během spojení.
9. **Kontrolní součet** (16 b) – slouží k detekci správného doručení TCP segmentu. Před odesláním TCP segmentu, vysílač spočítá kontrolní součet a výslednou hodnotu vloží do odesílaného TCP segmentu. Přijímač provede svůj vlastní výpočet kontrolního součtu a výslednou hodnotu porovná s hodnotou od vysílače. Pokud se shodují, byl TCP segment přijat v pořádku, v opačném případě bude segment zahozen.
10. **Ukazatel naléhavých dat** (16 b) – pole je využito, jen když je nastaven kontrolní bit URG na „1“.
11. **Volitelné položky záhlaví** (volitelné množství bitů) – volitelné položky mohou být použity pro různé doplňkové funkce. V dnešní době, lze využít až 29 doplňkových funkcí. Použitím těchto funkcí bude délka záhlaví větší, jak 20 B. Níže je uvedeno pár příkladů doplňkový funkcí, více těchto funkcí můžete nalézt na [32].
    - **Maximum Segment Size** – umožňuje nastavit maximální velikost TCP segmentu
    - **Selective Acknowledgement –** vysílač je pravidelně informován o segmentech, které byly doručeny v pořádku.
    - **Window Scale –** definuje nastavení většího okna pro řízení toku dat. Volitelnou funkci Windows scale může každá strana použít jen v úvodním SYN segmentu.
    - **User Timeout Option –** definuje, jak dlouho bude vysílač čekat na potvrzení odeslaných segmentu, než dojde k ukončení spojení. Doba, po kterou bude vysílač čekat na potvrzení se ve výchozím nastavení, pohybuje řádově kolem jednotek minut, podle RFC 5482. [33][34]

4.2 Průběh navázání spojení, přenos dat a ukončení spojení

Protokol TCP je spolehlivý a spojově orientovaný protokol, který před přenosem dat naváže spojení s komunikujícími stranami a ručí zato, že data budou přenesena k příjemci. Jestli nebude schopen data přenést, informuje o tom aplikační vrstvu. Dále zajišťuje, že předá přijaté segmenty aplikační vrstvě v tom pořadí, jak byly odeslány vysílačem.

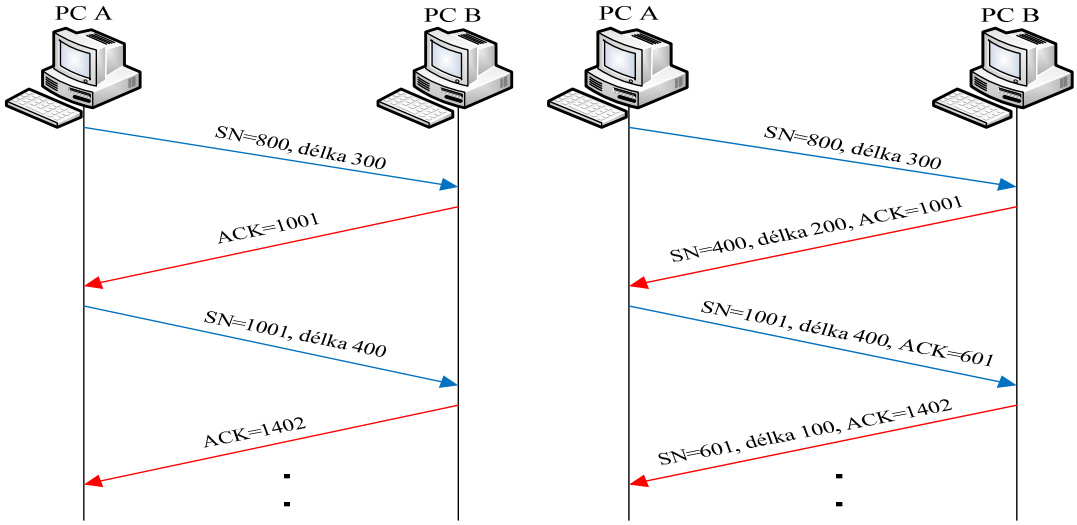
Průběh navazování spojení lze vidět na, viz Obrázek 4.2. Navazování spojení začíná tím, že vysílač A nejprve vygeneruje náhodné číslo, které bude používat jako startovací pořadové číslo odesílaného bajtu a uloží ho do pole SN (Sequence Number) například SN = 100. Vygenerována hodnota SN je společně s příznakem SYN v jednom segmentu odeslána

k příjemci B. Přijímač B přijímá segment a také vygeneruje náhodné pořadové číslo odesílaného bajtu, uloží ho do pole SN například SN= 600. Spolu s hodnotou SN, příznakem SYN potvrdí přijetí bajtu očíslovaného jako 100 nastavením ACK na hodnotu 101 a vše odešle v jednom segmentu k vysílači A. Oznamuje tím vysílači, že segment očíslovaný jako 100 dorazil v pořádku a očekává následující segment s hodnotou SN=101. Vysílač A po přijetí segmentu nastaví příznak ACK to znamená, že synchronizace pro zpětný přenos je v pořádku. Dále potvrdí příjem bajtu s číslem 600 tak, že do pole ACK vloží číslo 601. Vytvořená jednotka je opět odeslána k přijímači a spojení je navázáno. Celý tento proces navazování spojení se jmenuje three-way handshake a zkrácený zápis této komunikace je **SYN > SYN, ACK > ACK**.



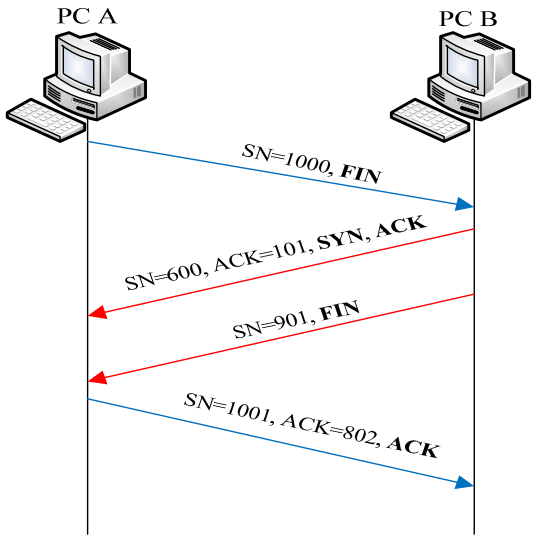
Obrázek 4.2: Průběh navazování TCP spojení

Po navázání spojení může začít samotný přenos dat. Číselná hodnota prvního bajtu je vložena do pole SN, potom je důležitá i délka přijatých dat. Podle této délky přijímací stanice pošle příznak ACK, jejíž hodnota je nastavena v pořadí dalšího očekávaného bajtu. U protokolu TCP může být komunikace buď jednosměrná, nebo obousměrná viz Obrázek 4.3. Během jednosměrné komunikace přijímač pouze data potvrzuje. U přenosu dat obousměrně má každý směr komunikace jiná zcela nezávislá pořadová čísla prvního odesílaného bajtu SN a potvrzovaného bajtu ACK.



Obrázek 4.3: Přenos dat jednosměrná (vlevo) a obousměrná komunikace

Po přenosu dat následuje ukončení TCP spojení. Princip ukončení spojení je podobný jak u navazování spojení s tím rozdílem, že je používán příznak FIN místo příznaku SYN viz Obrázek 4.4. Proces ukončování spojení se jmenuje four-way handshake, zkráceně lze zapsat jako **FIN > ACK, FIN > ACK** a spojení ukončuje vysílač i přijímač v každém směru zvlášť. Většinou je ale používáno ukončení spojení three-way handshake, zkráceně lze zapsat jako **FIN > FIN, ACK > ACK**.[34]



Obrázek 4.4: Ukončení TCP spojení

4.2.1 Princip potvrzování

U potvrzování TCP protokolu je používána metoda zpožděné odpovědi ACK tzv. piggybacking, který zajišťuje snížení velké množství režie spojeného s odesíláním krátkých segmentů, které jsou u interaktivních aplikací například Telnet nebo příkazový kanál FTP. Přijatá data nejsou ihned potvrzována, ale s určitým zpožděním (většinou 200 ms, maximálně 500 ms). K tomuto účelu je zde používán tzv. Nagleův algoritmus, který pozdržuje data k odeslání. Zdržení trvá tak dlouho, dokud nenashromáždí data větší než je velikost záhlaví segmentu nebo až vyprší časovač.[33]

4.2.2 Nastavení časovače

Velmi důležitým parametrem u protokolu TCP je správné nastavení časovače. Pokud je nastavená hodnota časovače krátká, může docházet ke zbytečnému opakování vysílaných segmentů a výsledkem bude plýtvání šířky pásma sítě. Naopak, jestli je nastavená hodnota časovače příliš velká, přijímač musí hodně (zbytečně) dlouho čekat, než dorazí data, která se mohla ztratit během cesty k přijímači a šířka pásma sítě nebude dostatečně využita. TCP protokol používá následující časovače. Časovač pro vytvoření spojení, spustí se pomocí příznaku SYN. Jestli nedostane vysílač ACK potvrzení od příjemce do 75 sekund bude spojení zrušeno. Časovač pro přeposílání je aktivní během odesílání dat. Jestliže vypršela doba časovače a vysílač nedostal ACK potvrzení na odeslaný segment, bude odeslán znova. Hodnota časovače se mění podle aktuální doby odezvy RTT (Round-Trip Time – je to doba od vyslání po přijetí potvrzení ACK na daný segment). Časovač pro opožděné ACK potvrzení je použit tehdy, když přijaté segmenty nemusí být okamžitě potvrzeny. Persist časovač je spuštěn, když přijímač ohlašuje, že má nulovou velikost okénka pro příjem dalších dat.

Okénko je na straně přijímače uzavřeno a nepřijímá další data do té doby, než se uvolní. Keepalive časovač se použije tehdy, když je spojení po dobu dvou hodin v klidovém stavu. Po vypršení časovače je odeslán speciální segment, který zjistí, jestli není protější strana vypnutá, pokud bude, příznakem RST se spojení ukončí. MSL (Maximum Segment Lifetime) časovač je spuštěn, když bylo ukončeno aktivní spojení a definuje dobu, po kterou může existovat segment v síti, než bude zahozen. Přitom časovač nadále očekává potvrzení o ukončení spojení [49]. U většiny implementací TCP protokolu je aktuální hodnota odezvy odhadnuta na základě doby odezvy posledních potvrzených segmentů. K získání požadované hodnoty je používána metoda exponenciálního průměrování doby odezvy, která je vyjádřena vztahem viz (4.1).

$$SRTT = \alpha \cdot SRTT + (1 - \alpha) \cdot RTT \ \text{[ms]} \tag{4.1}$$

kde *SRTT* odhad doby odezvy *RTT* [ms],
α vyhlazovací faktor [-],
RTT doba od vyslání po přijetí potvrzení ACK na daný segment [ms].

Konstanta α nabývá hodnot v intervalu $0 < \alpha < 1$, podle RFC 793 je vhodné volit konstantu α v rozsahu 0,8-0,9. Dosáhne se toho, že odhad závisí z větší části na předcházející hodnotě *SRTT* a jen z malé části na nově naměřené hodnotě *RTT* to znamená, že při pomalejších změnách *RTT* se nebude hodnota *SRTT* příliš rychle měnit, viz (4.1). Podle starších specifikací TCP protokolu můžeme interval časovače *RTO* (Retransmission Time-Out) vypočítat ze vztahu, viz (4.2).

$$RTO = \beta \cdot SRTT \ \text{[ms]} \tag{4.2}$$

kde β predikovaná doba odezvy a je doporučováno ji nastavit na $\beta = 2$,
SRTT odhad doby odezvy *RTT* [ms].

Jestli se budeme nacházet ve stabilním prostředí s pevně nastavenou hodnotou koeficientu β a s mírným kolísáním doby odezvy, je vypočítaná hodnota pro časovač velká. Dochází k tomu, že je zde velmi dlouhý interval čekání před opětovným vysláním segmentu. Naopak u proměnlivého prostředí by pevně nastavená hodnota koeficientu β nedokázala zabránit opětovnému vysílání segmentu. Proto byl algoritmus později pozměněn (jmenuje se Jacobsonův algoritmus) a ke sledování hodnoty *SRTT* je nutné i sledovat změny *RTT*. Změny *RTT* vypočítáme pomocí průměrné absolutní odchylky *RTTVAR* (Round-Trip Time VARiation) podle vztahu viz (4.3).

$$RTTVAR = \alpha \cdot RTTVAR + (1- \alpha) \left| SRTT - RTT \right| \ \text{[ms]} \tag{4.3}$$

kde *RTTVAR* průměrná absolutní odchylka [ms],
α vyhlazovací faktor [-],
SRTT odhad doby odezvy *RTT* [ms],
RTT doba od vyslání po přijetí potvrzení ACK na daný segment [ms].

Výsledný interval časovače potom vypočítáme ze vztahu, viz (4.4).

$$RTO = SRTT + 4 \cdot RTTVAR \ \text{[ms]} \tag{4.4}$$

kde *RTTVAR* průměrná absolutní odchylka [ms],

SRTT odhad doby odezvy *RTT* [ms].

Jacobsonův algoritmus tedy umožňuje zlepšit efektivnost protokolu TCP, ale pro nasazení do reálného prostředí je nutné zamyslet nad tím, jaká musí být nastavená hodnota časovače *RTO* u opakovaného vysílání segmentů a jaké naměřené hodnoty doby odezvy můžeme použít do Jacobsonova algoritmu. Odpovědi na první otázku je algoritmus exponenciálního prodlužování doby časovače opakovaného vysílání (Exponential RTO Backoff) a na druhou otázku Karnův algoritmus.

Algoritmus Exponential RTO Backoff pracuje tak, že při každém opětovném vyslání segmentu zvyšuje hodnotu časovače. Vysílač proto bude čekat delší dobu na potvrzení vyslaného segmentu. Tato doba kdy vysílač čeká na potvrzení, je využíváno sítí pro zotavení se ze stavu zahlcení. Novou hodnotu časovače *RTO* při každém opětovném vyslání segmentu můžeme vypočítat ze vztahu, viz (4.5).

$$RTO^{(m+1)} = \mathrm{q} \cdot RTO^{(m)} \ [\mathrm{ms}] \tag{4.5}$$

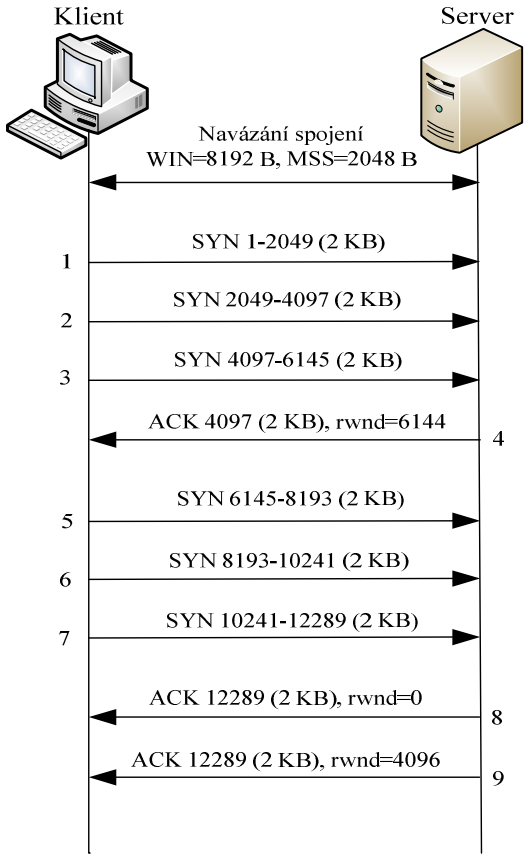
kde konstanta q > 1, běžně je udávaná hodnota konstanty q = 2 a jde o binární exponenciální algoritmus backoff, který najdeme i u síťové technologie Ethernet.

Karnův algoritmus pracuje tak, že jeho cílem není ovlivňovat hodnoty *SRTT* a *RTTVAR* tím způsobem, že nepoužívá naměřené doby odezvy *RTT* pro segmenty, které musely být opětovně vysílány. Pokud musí být segment opětovně vyslán, je hodnota časovače změněna podle vztahu, viz (4.5). Z potvrzení od přijatého segmentu, lze potom odvodit aktuální dobu odezvy *RTT* a vypočítat pomocí Jacobsonova algoritmu hodnotu časovače *RTO*.[36][44]

4.3 Řízení toku dat protokolem TCP

Na rychlost přenosu u protokolu TCP má velký vliv příchod potvrzení u segmentů, které byly předtím už vyslány. Rychlostí jakou budou přicházet potvrzení ACK na odeslané segmenty, je ovlivňováno úzkým místem mezi vysílačem a přijímačem. Úzkým místem většinou bývá přijímač nebo přenosová síť. Výsledná přenosová rychlost odesílaných segmentů potom odpovídá rychlosti nejpomalejšího spojení na síti. Hlavním důvodem je, že příliš rychle odesílání segmentů způsobí u přijímače zvýšenou zátěž na procesoru při jejich zpracování, přeplní se vyrovnávací paměť (každá má omezenou velikost) a dojde k zahazování segmentů. Je proto důležité, aby vysílač omezoval počet odesílaných a nepotvrzených segmentů. Kontrola toku je realizována pomocí mechanismu posuvného okénka. Velikost okénka *owin* (outstanding window) na straně vysílače, definuje množství dat, které mohou být najednou odeslány, aniž by vysílač čekal na potvrzení od přijímače. Další data může vysílač odeslat, až dostane potvrzení o přijetí z některých částí odeslaných dat. Vysílač je průběžně informován přijímačem o jeho volném místě ve vyrovnávací paměti pomocí tzv. okénka *rwnd* (receive window) a říká mu, kolik dat ještě může odeslat, aniž by přijímač byl zahlcen. Jestli hodnota rwnd bude rovna 0, je vyrovnávací paměť přijímače plná a vysílač musí čekat, až dostane nenulové *rwnd*. Tento způsob snižování *rwnd* nutí vysílač, aby snížil rychlost odesílání dat. Musí platit, že *owin* <= *rwnd*. Na Obrázek 4.5 můžeme vidět navazování spojení mezi vysílačem (klientem) a přijímačem (serverem), jejich vzájemné domluvení se na velikosti okénka *WIN*, velikost segmentu *MSS* a princip fungování posuvného okénka. Vysílač a přijímač se dohodly během navazování spojení, že velikost okénka *owin* = 8192 B a *MSS*=2048 B. Vysílač odešle segmenty 1, 2, 3 a obdrží od přijímače potvrzení 4, ve kterém jsou potvrzeny segmenty 1 a 2. Vysílač ihned odešle segmenty 5, 6, 7 a následně obdrží potvrzení 8, které informuje vysílač, že přijímač ještě nezpracoval segmenty

3, 5, 6, 7 a vysílači uzavře okno *rwnd* nastavením na hodnotu 0. Jakmile přijímač zpracuje část dat, ihned pošle vysílači informaci, že může odesílat další data, ale jen do velikosti okna *rwnd*, tedy 4096 B. Zbylá data ve vyrovnávací paměti přijímače ještě nejsou zpracována.



Obrázek 4.5: Výměna zpráv mezi vysílačem a přijímačem

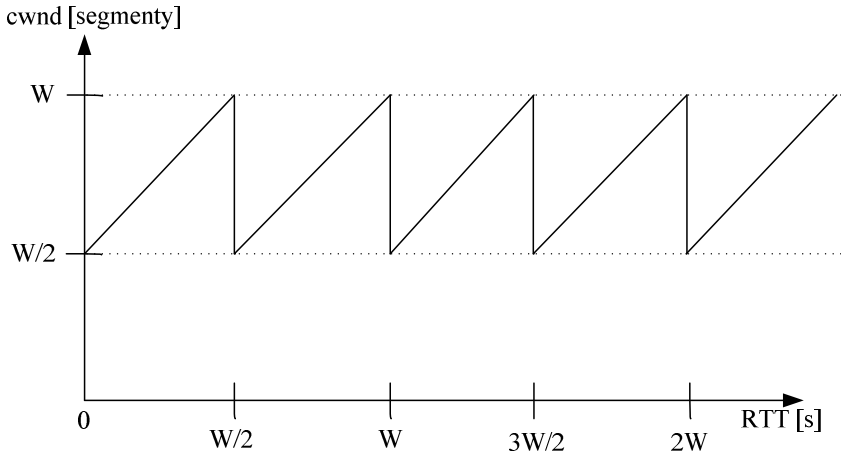
Jestliže známe velikost okna *owin* na straně vysílače a dobu odezvy *RTT*, můžeme vypočítat přenosovou rychlost TCP spojení podle vztahu viz (4.6).

$$R = owin / RTT \ \text{[B/s]} \qquad (4.6)$$

Tato rychlost platí jen tehdy, když linka bude mít dostatečnou propustnost. Důvodem je, aby nedocházelo k rychlému naplnění odchozích front ve směrovačích. Většinou se stává, že linka nemá dostatečnou propustnost (např. má nižší přenosovou rychlost než je u vysílače), dojde tedy k zahlcení a následné ke ztrátě dat. Proto je na straně vysílače okno *cwnd* (Congestion Window), které informuje vysílač, kolik může odeslat nepotvrzených dat, aniž by došlo k zahlcení sítě. Cílem vysílače je, aby využil maximální dostupnou kapacitu linky, proto postupně zvyšuje hodnotu *cwnd*. Hodnota *cwnd* je aditivně (přídavně) zvyšována o konstantu (začíná se s hodnotou 1) u každé hodnoty doby odezvy *RTT*. Jakmile vysílač obdrží informaci o blížícím se zahlcení, sníží hodnotu *cwnd* na polovinu. O blížícím se zahlcení vysílač informuje například vypršení časového limitu pro příjem potvrzení nebo příjem duplicitního potvrzení od posledního odeslaného segmentu. Okénko *cwnd* nemůže být neustále zvyšováno, jeho hodnota by se měla pohybovat pod úrovní prahové hodnoty *SSTHRESH* - Slow Start Treshold (tato hodnota se nastavuje během navazování spojení). Prahová hodnota *SSTHRESH* je hranice, kde její překročení oznamuje, že s velkou pravděpodobností dojde k zahlcení.

SSTHRESH je udržováno jen v násobcích *MSS*, aby vysílač rychle dosáhl dostupné kapacity na lince po detekci zahlcení. Okénko zahlcení *cwnd* má pilovitý průběh viz Obrázek 4.6. Největší velikost okénka, které vysílač použije pro odeslání nepotvrzených dat, bude mít hodnotu, viz (4.7). To znamená, že velikost okénka nepřevýší velikost okénka u přijímače a ani okénka cwnd.

$$owin = \min(cwnd, rwnd)\ [\mathrm{B}] \tag{4.7}$$



Obrázek 4.6: Pilovitý průběh okénka zahlcení cwnd

Mechanismus posuvného okénka *owin*, je používán pro řízení provozu mezi koncovými uzly a také ovlivňuje propustnost, aby nedošlo k zahlcení. Maximální velikost okénka je určena 16 bitovým polem v záhlaví TCP, které se pohybuje v rozsahu od 0 do 65535 B (2^{16}-1). Tento rozsah by měl být pro většinu aplikací dostačující. Může se stát, že tato velikost okénka je malá. Řešením je použití TCP Window Scale (zvětšení okénka), které dovoluje zvětšit výchozí velikost okénka až na 1 GB. Tuto hodnotu získáme vynásobením maximální velikosti výchozího okénka (2^{16}-1) a hodnoty pro zvětšení okénka 2^{14}. Kde pro zvětšení okénka je využíváno rozsahu 0 až 14 bitů. Volbu zvětšení okénka nastavíme jen v úvodních segmentech s příznakem SYN při inicializaci spojení. Metoda zvětšení okénka nachází uplatnění v gigabitových sítích, protože jeho nastavením využijeme dostatečně přidělenou šířku pásma. Naopak použitím výchozí velikosti okénka by přidělena šířka pásma byla využita jen částečně.[33][35][36]

4.4 Mechanismy předcházející zahlcení

Nejprve, než budou popsány jednotlivé algoritmy na kontrolu zahlcení, je důležité vědět co to vlastně zahlcení sítě je. Zahlcení sítě je stav, kdy do sítě vstupuje víc dat, než je schopna samotná síť přenést nebo zpracovat. Jedná se buď o zahlcení vyrovnávacích front směrovače, vstupní linka s větší propustnosti chce přenést dat přes výstupní linku s nižší propustností, nebo pokud více vstupních linek přenáší data na jeden a tentýž port u výstupní linky. Každý směrovač nemá neomezenou velikost vyrovnávacích front a dochází tedy k zahazování paketů. Když dojde k zahazování paketů, můžeme to brát jako znamení, že se síť nachází ve stavu zahlcení. Ale každá ztráta paketu nemusí ihned znamenat stav zahlcení například u TCP SACK (Selective ACKnowledgment), je stav zahlcení teprve až dojde ke ztrátě více paketů během jednoho *RTT*.

Mechanismy předcházející zahlcení pracují na koncových stanicích a způsobí, že je zpomaleno TCP vysílání v době, když se síť nachází ve stavu zahlcení. Pro tento účel byly vytvořeny algoritmy, které dokážou efektivně řídit velikost posuvného okénka vysílače. Cílem jednotlivých algoritmů je, aby byla co nejlépe využita dostupná šířka pásma linky a zajistili spravedlivé sdílení linky s ostatními toky. Mezi tyto algoritmy patří:

- pomalý start (slow start),
- vyhýbání se zahlcení (congestion avoidance),
- rychlé přeposílání (fast retransmit),
- rychlé zotavení (fast recovery),
- omezené vysílání (limited trasmit).

4.4.1 Pomalý start (slow start)

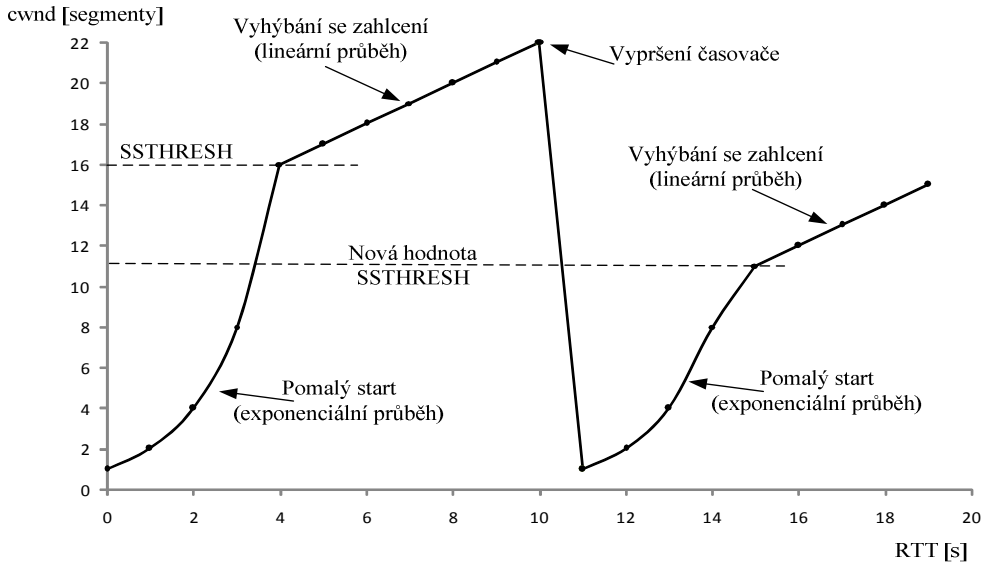
Po navázání TCP spojení nezačne vysílač vysílat plnou rychlostí na lince, protože posíláním velkého počtu segmentů, může způsobit vyčerpání kapacity sítě a také rychle naplnit okno *rwnd* na straně příjemce. Vysílač nejprve nastaví hodnotu *cwnd* na *cwnd* = 1 to znamená, že odešle v daném okamžiku jeden segment. Po odeslání segmentu čeká vysílač na potvrzení, jakmile potvrzení dorazí, je hodnota *cwnd* nastavena na *cwnd* = 2. Vyšlou se tedy 2 segmenty. V závislosti na tom, zda odesílatel přijme jedno nebo 2 potvrzení se *cwnd* zvýší o hodnotu 1 nebo o 2, tedy hodnota *cwnd* bude 3 nebo 4, potom tedy mohou být odeslány 3 nebo 4 segmenty. Rychlost vysílání je tedy exponenciálně zvyšována až na prahovou hodnotu *SSTHRESH*, kde už hrozí větší pravděpodobnost, že už může dojít ke ztrátě segmentu, a další nárůst už probíhá lineárně, viz Obrázek 4.7. [36][38][44]

4.4.2 Vyhýbání se zahlcení (congestion avoidance)

Během vysílání se může stát, že hodnota *cwnd* bude větší než je prahová hodnota *SSTHRESH* a odesláním dalšího dvojnásobku segmentu by způsobilo zahlcení. Jak již bylo zmíněno stav zahlcení je indikován ztrátou segmentu. Cílem tohoto algoritmu je se zahlcení vyhnout a změní exponenciální nárůst hodnoty *cwnd* na lineární. Jestliže vysílač detekuje ztrátu segmentu, je aktuální hodnota *cwnd* snížena na polovinu a stejná hodnota v *cwnd* se nastaví i na prahové hodnotě *SSTHRESH*. Se změněnými hodnotami *cwnd* a *SSTHRESH*, bude vysílač používat algoritmus pomalého startu jen do té doby, dokud se nedostane na prahovou hodnotu *SSTHRESH*. Po překročení prahové hodnoty *SSTHRESH*, bude hodnota *cwnd* zvyšována lineárně, inkrementována o jedničku, viz Obrázek 4.7. Nejen, že se sníží aktuální rychlost vysílání segmentů, ale také dojde ke změně zvyšování rychlosti vysílání, která se bude pomalu blížit ke stavu zahlcení. Celý princip algoritmu vyhýbání se zahlcení lze shrnout do těchto bodů:

1. během inicializace spojení je nastavena hodnota *cwnd* = 1 (vysílač odešle jeden segment) a prahová hodnota *SSTHRESH* = 65535 B (maximální velikost okénka),
2. při detekci zahlcení se nastaví prahová hodnota *SSTHRESH* = *cwnd*/2,
3. hodnota *cwnd* je uvedena do počátečního stavu, *cwnd* = 1 a provede se fáze pomalého startu, dokud hodnota *cwnd* nedosáhne hodnoty *SSTHRESH*, *cwnd=SSTHRESH*,
4. od *cwnd* ≥ *SSTHRESH*, je hodnota cwnd postupně inkrementována o jedničku.

Při každé detekované ztrátě segmentu je hodnota *cwnd* snížena na jedničku a množství posílaných dat v síti také klesá exponenciálně. Tímto snížením dochází k vyprazdňování front ve směrovačích a síť má čas se vzpamatovat ze stavu zahlcení. [36][38][44]



Obrázek 4.7: Fáze pomalý start a vyhýbání se zahlcení

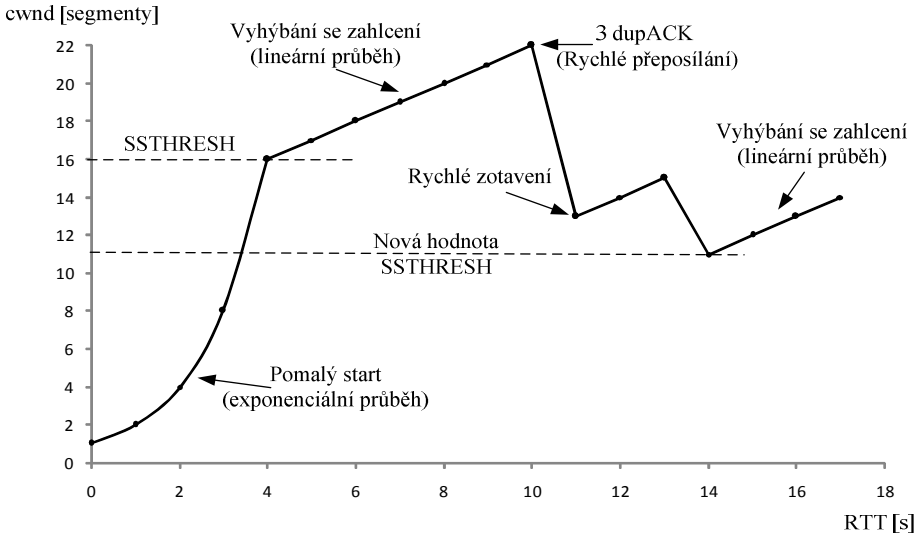
4.4.3 Rychlé přeposílání (fast retransmit)

Při přenosu dat může dojít k tomu, že se během cesty ztratí segment. Příjemce to zjistí až příchodem dalších segmentů mimo pořadí. Příjemce musí po přijetí segmentu mimo pořadí zopakovat (duplikovat) potvrzení posledního správně přijatého segmentu. Po odeslání duplikovaného potvrzení (dupACK) bude přijímač očekávat ztracený segment. Pokud ztracený segment stále nebude přicházet, pošle přijímač druhé dupACK a po chvíli třetí dupACK. Vysílač po příjmu třech dupACK (celkem to jsou čtyři potvrzení na stejný segment), vyhodnotí danou situaci tím, že došlo během přenosu ke ztrátě segmentu. Vysílač okamžitě zopakuje odeslání ztraceného segmentu a potom bude pokračovat v odesílání dalších segmentů. Příjemce příjme ztracený segment, potvrdí ho včetně segmentů, které přijal mimo pořadí již dříve. Algoritmus rychlého přeposílání umožňuje zopakovat pouze ztracený segment a ne všechny segmenty, které nebyly potvrzeny, ještě v době, než dojde k vypršení časovače na straně vysílače, viz Obrázek 4.8. [36][38][44]

4.4.4 Rychlé zotavení (fast recovery)

Algoritmus rychlého zotavení zajišťuje, aby po vykonání mechanismu rychlého přeposílání neklesla hodnota *cwnd* na nulu. Hodnota *cwnd* je snížena na nulu teprve až dojde k vypršení časovače. Algoritmus po příjmu třech dupACK, nastaví hodnotu cwnd na polovinu, provede opětovné odeslání ztraceného segmentu a pokračuje ve vysílání segmentů s lineárním nárůstem *cwnd* o 1. Výsledkem je, že se vynechá počáteční exponenciální náběh (slow start), viz Obrázek 4.8. Princip algoritmu rychlého zotavení, lze shrnout do těchto následujících bodů:

1. Když vysílač příjme třetí dupACK pak:
    a) nastaví prahovou hodnotu *SSHTRESH* na *cwnd*/2,
    b) provede opětovné odeslání TCP segmentu,
    c) nastaví hodnotu *cwnd* na *SSHTRESH* + 3·*MSS* (zvýšením hodnoty *cwnd* o 3 znamená, že už byly doručeny tři další segmenty po ztraceném segmentu, který byl detekován příchodem pomocí třech dupACK).
2. Vysílač po přijetí čtvrtého a dalšího duplikátu provede pokaždé zvětšení okna *cwnd* o velikost segmentu.
3. Jestliže, je ztracený segment potvrzen příjemcem, nastaví vysílač okno *cwnd* na prahovou hodnotu *SSHTRESH*.[36][38][44]



Obrázek 4.8: Fáze rychlé přeposílání a rychlé zotavení

4.4.5 Omezené vysílání (limited trasmit)

Algoritmus omezeného vysílání umožňuje odesílat nové segmenty i v případě, že to není nutné během stavu zahlcení, konkrétně když:

a) vysílač přijal dva dupACK tj. celkem to jsou tři potvrzení pro stejný segment,
b) okno vysílače nastavené od přijímače dovoluje vyslání segmentu,
c) množství nepotvrzených dat po odeslání nového segmentu nepřekročí hodnotu *cwnd*+2, to znamená, že vysílač má možnost odeslat dva segmenty nad rámec velikosti okna zahlcení.

Algoritmus omezeného vysílání dovoluje odeslat omezené množství dat i přes mechanismy předcházející proti zahlcení.[44]

5 ALGORITMY TCP PROTOKOLU

Algoritmy TCP protokolu obsahují mechanismy pro předcházení zahlcení. Jejich cílem je včas reagovat na blížící se zahlcení a zpomalit rychlost vysílání dat. Dále se snaží co nejlépe využívat dostupnou šířku pásma. V práci budou popsány všechny dostupné implementace, které můžeme nalézt na operačním systému Linux. Budou mezi sebou porovnány a to z hlediska jejich propustnosti.

TCP Tahoe

Prvním algoritmem pro řízení zahlcení byl TCP Tahoe, v jeho implementaci najdeme první tři mechanismy pro předcházení zahlcení. Jsou to pomalý start, předcházení zahlcení a rychlé přeposílání. Ztracený segment pozná vysílač teprve, až dojde k vypršení časovače, to znamená, že vysílač během té doby nedostal potvrzení ACK na odeslaný segment. Počet odesílaných segmentů je tak rychle snížen a velikost okénka cwnd je potom nastaveno na 1, neboli přejde se do fáze pomalého startu.[36][38]

TCP Reno

Algoritmus TCP Reno obsahuje mechanismy pomalý start, předcházení zahlcení, rychlé přeposílání a rychlé zotavení. Vysílač detekuje ztrátu segmentu po přijetí třech dupACK. Po přijetích třech dupACK je zmenšeno okénko *cwnd* na polovinu, potom přechází do fáze rychlého přeposílání a po chvíli se dostane opět do fáze rychlého zotavení. Jestliže dojde k vypršení časovače u odeslaného segmentu, přejde se do fáze pomalého startu tak jako u TCP Tahoe. Nevýhodou Rena je, že může pracovat jen v síti, kde je nízká ztrátovost paketů. Jestliže ztrátovost paketů je vyšší, nebude pracovat příliš spolehlivě a jeho výkonnost bude stejná jako u TCP Tahoe. Je to z důvodu, že Reno dokáže detekovat jen jednu ztrátu paketu. Když je ale ztrátovost větší, Reno bude tuto ztrátovost detekovat postupným odesíláním ztracených segmentů v následném potvrzení ACK od přijímače. [36][38]

TCP New Reno

TCP New Reno je modifikovaná verze TCP Reno, která umožňuje detekovat větší ztrátovost paketů. Podobně jako u Rena, po přijetí třech dupACK, přechází do fáze rychlého přeposílání, potom do fáze rychlého zotavení a v této fázi setrvává tak dlouho dokud, nebudou potvrzena všechna data, která byla poslána během fáze rychlého zotavení. Výsledkem je, že se předchází nedostatku u Rena a to opakovanému snižování velikosti okénka *cwnd*. Problémem tohoto algoritmu je, že pro detekování ztráty paketu využívá dobu odezvy *RTT*. Dobu odezvy *RTT* využije u prvního opětovně zaslaného segmentu s přijatým potvrzením ACK. Z toho dále může odvodit segmenty, které byly ztraceny.[36][38]

TCP Westwood

TCP Westwood je další modifikovaná verze TCP Rena, která při ztrátě segmentu tj. přijetí třech dupACK nesníží velikost okénka cwnd a prahové hodnoty *SSTHRESH* na polovinu. Novou hodnotu *cwnd* a *SSTRESH* vypočítá pomocí doby odezvy *RTT* z přijatého třetího dupACK a dostupné šířky pásma, viz (5.1), neboli provádí postupné adaptivní zvyšování a adaptivní snižování (AIAD – Adaptive Increase Adaptive Decrease) velikosti

okénka cwnd a prahové hodnoty *SSTRESH*. Tímto způsobem přechází do fáze rychlého zotavení a efektivně se vyhýbá i zahlcení.[36][38]

$$cwnd = SSTHRESH = BW \cdot RTT \tag{5.1}$$

kde *BW* šířka pásma [b/s]
RTT doba odeslání a příjem potvrzení daného segmentu [s]

TCP SACK

TCP Sack je pokročilejší verze TCP Rena, TCP New Rena a využívá selektivní potvrzování (SACK). U TCP Rena zvýší propustnost, když dochází k větší ztrátovosti segmentů a to tím, že při ztrátě segmentu využívá příjemce selektivní potvrzování, ve kterém oznamuje vysílači, které pakety byly přijaty v pořádku a které nedorazili a chybějí. Pomocí selektivního potvrzení si vysílač udělá obrázek o tom, jaký je průběh přenosu dat. Dále u New Rena bylo provedeno zlepšení jeho problému, které umožňuje přeposílat více paketu během jedné doby odezvy RTT.[36][38]

TCP Vegas

Tato implementace umožňuje snížit ztrátovost paketu a používá k tomu tři mechanismy, jsou to rychlé přeposílání, vyhýbání se zahlcení a pomalý start.

U mechanismu rychlého přeposílání si Vegas zaznamená do paměti dobu, kdy odeslal segment příjemci, tato doba se nazývá časová známka (timestamp). Po přijetí potvrzení ACK vezme vysílač hodnotu časové známky a určí přesnou dobu odezvy *RTT*. Když vysílač příjme dupACK paket, Vegas zkontroluje, jestli je rozdíl uložené časové známky daného segmentu a časové známky dupACK větší jak doba odezvy *RTT*. Pokud je, vysílač okamžitě odešle požadovaný segment a nečeká až příjme 3 dupACK pakety. Dále, když vyprší časovač pro daný segment (vysílač nedostal ACK potvrzení), bude daný segment automaticky přeposlán, aniž by čekal na dupACK paket a vyhne se tím snížení velikosti okénka cwnd.

Vegas používá mechanismus vyhýbání se zahlcení, který nejen reaguje na ztracené segmenty, ale i na dobu odezvy *RTT* pro odeslaný segment. Tento mechanismus provádí výpočet očekávané a aktuální propustnosti. Očekávanou propustnost vypočítá ze vztahu, viz (5.2) a definuje dostupnou šířku pásma bez stavu zahlcení.

$$očekáváná\_propustnost = cwnd \,/\, baseRTT \text{ [b/s]}, \tag{5.2}$$

kde *cwnd* velikost okénka,
baseRTT je minimální doba odezvy RTT na segment, kdy síť není ještě zahlcená [s].

Aktuální propustnost definuje aktuálně používanou šířku pásma během přenosu segmentů, vypočítá se ze vztahu, viz (5.3).

$$aktuální\_propustnost = cwnd \,/\, RTT \text{ [b/s]}, \tag{5.3}$$

kde *cwnd* velikost okénka,
RTT doba odeslání a příjem potvrzení daného segmentu [s].

Z očekávané a aktuální propustnosti vypočítá jejich *rozdíl* podle vztahu, viz (5.4). Dále definuje parametry α a β, podle kterých bude velikost okénka cwnd zvětšována nebo

zmenšována. Výchozí hodnoty parametrů jsou pro $\alpha = 1$, $\beta = 3$. Jestli je výsledná hodnota *rozdíl* < α, velikost okénka cwnd se lineárně zvýší během příští doby odezvy *RTT*. Pokud je *rozdíl* > β, velikost okénka se lineárně sníží během příští doby odezvy *RTT*. Velikost okénka cwnd zůstane nezměněna, když bude platit $\alpha < rozdíl < \beta$.

$$rozdíl = očekávaná\_propustnost - aktuální\_propustnost \ \text{[b/s]} \qquad (5.4)$$

Mechanismus pomalého startu zdvojnásobí velikost okénka cwnd při každé druhé odezvě *RTT*. Tímto způsobem roste okénko cwnd exponenciálně až dosáhne dostupné šířky pásma, kde už hrozí stav zahlcení. Stavu zahlcení předchází Vegas tím, že při každé druhé odezvě *RTT* porovnává očekávanou a aktuální přenosovou rychlost. Jestli je aktuální přenosová rychlost nižší, než očekávaná přenosová rychlost přejde Vegas z fáze pomalého startu do fáze vyhýbání se zahlcení.[37]

TCP Veno

Algoritmus TCP Veno během vysílání segmentu rozlišuje, zda ztráta segmentu byla náhodná nebo vlivem zahlcení sítě. K tomuto rozlišení používá parametr N, který indikuje, jestli vytvořené spojení mezi vysílačem a přijímačem je ve stavu zahlcení. K výpočtu parametru N použije Veno vztahy, viz (5.2), (5.3), (5.4), které nalezneme u algoritmu TCP Vegas a výslednou hodnotu parametru N získáme ze vztahu, viz (5.5).

$$N = rozdíl \cdot baseRTT \ , \qquad (5.5)$$

kde *baseRTT* je minimální doba odezvy RTT na segment, kdy síť není ještě zahlcená [s].

Dále je definována konstanta β s výchozí hodnotou $\beta = 3$, podle konstanty β a vypočtené hodnoty N, určí Veno jestli ztráta segmentů je náhodná nebo způsobena vlivem zahlcení sítě. Při náhodné ztrátě segmentu musí platit $N < \beta$. Jestli je hodnota parametru N větší nebo rovno β ($N \geq \beta$), bude Veno detekovat ztrátu segmentu stavem zahlcení. TCP Veno používá mechanismus pomalý start (více v podkapitole 4.4.1), algoritmus postupného zvyšování velikosti okénka cwnd, to znamená, že jakmile je cwnd větší jak prahová hodnota *SSTHRESH* bude se okénko cwnd zvyšovat o *cwnd*=(*cwnd* + 1) /*cwnd* po každém přijatém ACK potvrzení. U TCP Vena dále najdeme mechanismus rychlého zotavení, který pracuje na stejném principu jako v podkapitole 4.4.4, ale obsahuje jen malou modifikaci při definování prahové hodnoty *SSTHRESH*. Tato malá modifikace se projeví pokud Veno detekuje náhodnou ztrátu segmentu, potom *SSTHRESH* je vypočítáno jako *SSTHRESH*= *cwnd* ·(4/5). Okénko *cwnd* nemusí být nutně vynásobeno hodnotu 4/5, ale je vhodné se pohybovat v rozmezí od 1/2 do 1, protože nebude docházet k rychlému snížení velikosti okénka jako v případě zahlcení. Podle experimentu ve [41] je vhodné vynásobit okénko cwnd hodnotou větší jak 3/4. Naopak při detekování stavu zahlcení zůstane vypočet *SSTHRESH* beze změny, tj. *SSTHRESH*= *cwnd* /2. Díky těmto výše zmíněným vlastnostem je Veno hlavně používán v bezdrátových sítích. [41]

H-TCP

H-TCP je používán hlavně ve vysokorychlostních sítích, kde zmenšením časového intervalu předchází události přetížení. Časový interval se začne zmenšovat, když během vysílání dosáhne linka ke své hranici přidělené šířky pásma. Je to bráno jako informace, která oznamuje, že může dojít k zahlcení a se zvyšující se frekvenci sleduje tuto informaci častěji a přesněji. K tomuto sledování nepoužívá velikost okénka cwnd, ale dobu Δ, která uplynula od poslední události přetížení. Dobu Δ využívá algoritmus adaptivního zvyšování AIMD (Adaptive Increase Multiplicative Decrease) a podle této doby se neustále mění. Dále AIMD obsahuje měřítko *RTT* dané linky, které zajišťuje férovost mezi soutěžícími přenosy s odlišnými časy *RTT*. Velikost okénka cwnd při každém přijatém ACK potvrzení vypočítá ze vztahu, viz (5.6) a při ztrátě segmentu ze vztahu, viz (5.7).[36][42][43]

$$cwnd = cwnd + (2 \cdot (1 - \beta) \cdot a(\Delta)/cwnd) \tag{5.6}$$

$$a(\Delta) = \begin{cases} 1 & \Delta \leq \Delta_L \\ \max\{a'(\Delta)T_{min}; 1\} & \Delta > \Delta_L \end{cases}$$

$$cwnd = g(B) \cdot cwnd \tag{5.7}$$

$$g(B) = \begin{cases} 0{,}5 & \left|\frac{B(k+1) - B(k)}{B(k)}\right| > \Delta_B \\ min\left\{\frac{T_{\min}}{T_{max}}; 0{,}8\right\} & \text{jinak} \end{cases}$$

kde Δ_L je práh, který používá standardní TCP algoritmus při $\Delta \leq \Delta_L$,
$a'(\Delta)$ je kvadratická přírůstková funkce navyšování navrhnutá jako $a'(\Delta) = 1 + 10(\Delta - \Delta_L) + 0{,}25(\Delta - \Delta_L)^2$,
$T_{\min}$, $T_{\max}$ jsou měření minima a maxima doby odezvy RTT při přenosu,
B(k+1) je měření maximální propustnosti během poslední události přetížení,
Δ_B hranice, kde dochází ke změně šířky pásma u které je použít TCP pokles

Highspeed TCP (HS-TCP)

HSTCP je algoritmus, který se dokáže přizpůsobit velkým okénkům zahlcením během TCP spojení. Pokud je velikost okénka zahlcení menší jak *SSTHRESH* nebo ztrátovost paketů je nanejvýš do 10^{-3}, chová se tento algoritmus jako standardní TCP a při opačné velikosti okénka zahlcení nebo ztrátovosti paketů nižší než 10^{-3} je použit algoritmus HSTCP. Při stavu, kdy se chová jako HSTCP je na probíhající přenos mnohem agresivnější. Využívá k tomu funkci odezvy, která má 3 parametry, jsou to *Low_Window*, *High_Window* a *High_P*. Parametr *Low_Window* je používán pro zajištění kompatibility a stanovení bodu přechodu, to znamená, když bude aktuální velikost okénka zahlcení menší nebo rovno hodnotě v parametru *Low_Window*, použije HSTCP stejnou funkci odezvy jakou má standardní TCP hodnotu. Naopak pokud aktuální velikost okénka bude větší jak hodnota v parametru *Low_Window* použije funkci odezvy HSTCP. Parametry *High_Window* a *High_P* definují horní hranici funkci odezvy, kde *High_P* je počet zahozených segmentů a *High_Window* je průměrná velikost okénka zahlcení během přetížení. Odezvy funkce HSTCP jsou reprezentovány

novými hodnotami aditivního zvyšování a vícenásobného snížení, které se mění podle aktuální velikosti okénka cwnd. Ve fázi vyhýbání se zahlcení, lze nové okénko zahlcení pro přijaté potvrzení ACK vypočítat ze vztahu, viz (5.8) a při ztrátě segmentu, viz (5.9). Konstanta *a* je nastavena hodnotu 1, konstanta *b* má hodnotu 0,5.

$$cwnd = cwnd + (a / cwnd) \tag{5.8}$$

$$cwnd = cwnd - (b \cdot cwnd) \tag{5.9}$$

Algoritmus HSTCP je vhodný pro přenos velkého množství dat, používá se ve vysokorychlostních sítích a dokáže udržovat vysokou přenosovou rychlost, při různých stavech sítě. Přizpůsobování se různým podmínkám v síti provádí už při fázi pomalého startu, to znamená, pokud dojde ke ztrátě segmentu, dovolí využívat menší šířku pásma sítě než 1/2 cwnd.[36][45]

Scalable TCP

Úkolem Scalable TCP je zmenšit závislost mezi spravováním okénka TCP a dobou odezvy *RTT*. Standardní TCP ve fázi zahlcení, když dostane ACK potvrzení na daný segment zvýší vysílač velikost okénka zahlcení jedním segmentem během každé doby odezvy *RTT*. Při ztrátě segmentu reaguje standardní TCP snížením aktuální velikosti okénka na polovinu. Pro zlepšení výkonnosti oproti standardnímu TCP používá Scalable pevné hodnoty aditivního zvyšování *a* a vícenásobného snižováním *b*. Aktualizaci okénka zahlcení provádí tento algoritmus dvěma způsoby.

- Po příchodu ACK potvrzení s dobou odezvy *RTT*, kdy nebyl ještě detekován stav zahlcení, vypočítá velikost okénka zahlcení ze vztahu, viz (5.10).

  $$cwnd = cwnd + a \tag{5.10}$$

  kde a konstanta v intervalu $0 < a < 1$.

- při detekování stavu zahlcení během doby odezvy *RTT*, vypočítá velikost okénka zahlcení ze vztahu, viz (5.11).

  $$cwnd = (1 - b) \cdot cwnd \tag{5.11}$$

  kde b konstanta v intervalu $0 < b < 1$.

Kvůli rozdílné přenosové rychlosti, stabilitě a alokované šířce pásma, jsou konstanty nastaveny na hodnotu a=0,01 a b=0,125. Scalable TCP má využití především ve vysokorychlostních sítích, protože dokáže rychle reagovat na ztrátu segmentu a vrátit se na dostupnou šířku pásma.[36][46]

TCP Hybla

Tento algoritmus umožňuje zvýšit propustnost v sítích s dlouho dobou odezvy *RTT* například satelitní a bezdrátové přenosy. Velké zpoždění odezvy *RTT* má za následek velmi rychlé snížení velikosti okénka cwnd a zpomalení rychlosti vysílání. Aby k tomuto snížení nedocházelo, provádí Hybla odstranění závislosti doby odezvy *RTT* během aktualizování

algoritmu. Dosáhne toho tak, že velikost okénka zahlcení nastaví do normalizované velikosti předchozího okénka zahlcení podle vztahu, viz (5.12).

$$\rho = RTT / RTT_0 \text{ [-]}, \tag{5.12}$$

kde ρ normalizovaná velikost okna zahlcení pro aktualizování okna zahlcení
RTT_0 je referenční doba spojení [s],
RTT doba odeslání a příjem potvrzení daného segmentu [s].

Nová velikost okénka zahlcení se vypočítá podle vztahu, viz (5.13)

$$cwnd^H(t) = \begin{cases} \rho 2^{\rho t/RTT} \ , \ 0 \le t < t_{\gamma,0} \\ \rho \left[\rho \frac{t - t_{\gamma,0}}{RTT} + \gamma \right], \ t \ge t_{\gamma,0} \end{cases}, \tag{5.13}$$

kde H identifikuje algoritmus TCP Hybla,
γ prahová hodnota SSTHRESH,
RTT doba odeslání a příjem potvrzení daného segmentu [s],
$t_{\gamma,0}$ doba, kdy okno zahlcení dosáhne hodnoty $\rho\gamma$, je stejné pro každé RTT, počátek času je dán $t_{\gamma,0}=RTT_0 \cdot \log_2 \gamma$ [s],
t doba, kdy se změní velikost okna zahlcení [s].

Hybla používá selektivní potvrzování SACK (příjemce oznamuje vysílači, které segmenty se ztratily) a umožňuje během jedné doby odezvy *RTT* přeposlat více ztracených segmentů. Dále tento algoritmus využívá i časové známky (timestamps) pro kontrolu odeslaných segmentů, popis u algoritmu TCP Vegas.[39][40]

TCP Cubic

TCP Cubic je vylepšená verze algoritmu TCP BIC a dokáže lépe spravovat velikost okénka zahlcení. Hlavní myšlenkou algoritmu je, že zvětšení velikosti okénka závisí jen na dvou po sobě jdoucích událostech přetížení, tím dochází k odstranění závislosti na době odezvy *RTT*. Dovoluje tedy ostatním tokům využívat stejnou velikost okénka nezávisle na jejich době odezvy *RTT*. Jméno Cubic dostal algoritmus podle kubické funkce, kterou používá pro výpočet nové velikost okénka zahlcení, viz (5.14).

$$cwnd_{\text{cubic}} = C \cdot (t - K)^3 + cwnd_{\max} \tag{5.14}$$

kde C váhový faktor a má hodnotu C=0,4 [-],
t doba od poslední změny velikost okénka zahlcení [s],
$cwnd_{\max}$ maximální velikost okénka zahlcení před jeho snížením,
K provádí aktualizaci času od poslední ztráty segmentu, vypočítá se jako $K = \sqrt[3]{cwnd_{\max} \cdot \beta / C}$ [s], kde β je konstanta vícenásob. snížení a má hodnotu β = 0,8 [-].

Algoritmus Cubic je optimalizován pro použití ve vysokorychlostních sítích, dokáže jednoduše kontrolovat velikost okénka a je méně agresivní, než varianta BIC, to znamená, že nezabere moc přenosové šířky pásma pro ostatní TCP toky. Dnes je využívaný jako standardní algoritmus v linuxovém jádře.[47]

TCP BIC

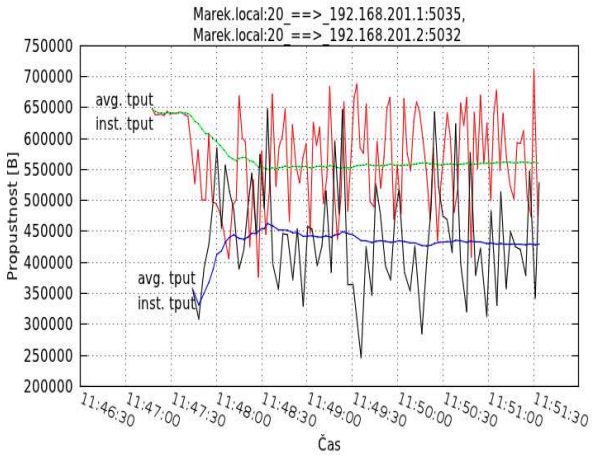
Algoritmus BIC (Binary Increase Control) má více agresivní chování, než Cubic. Jeho cílem je co nejrychleji dosáhnout maximální dostupné šířky pásma, i když svým chováním omezí ostatní TCP toky. Dále má definovanou minimální a maximální velikost okénka zahlcení. U minimální velikosti okénka zahlcení má jistotu, že nedojde ke ztrátovosti segmentu a maximální velikost okénka zahlcení definuje horní hranici přenosové rychlosti, kde už ke ztrátě segmentu může dojít. Když maximální velikost okénka zahlcení ještě navýší a ke ztrátě segmentu nedochází, stanoví tak novou maximální velikost okénka zahlcení a z původní maximální velikosti okénka zahlcení se následně stane minimální velikost okénka zahlcení. Jestliže, ale dojde ke ztrátě segmentu, je velikost okénka snížena na minimální velikost. Tento postup neustále opakuje a pokouší se dostat na maximální přidělenou šířku pásma. Čím více se bude blížit k maximální přidělené šířce pásma, jeho chování bude méně agresivní. BIC byl dříve používaný jako standardní algoritmus v linuxovém jádře, ale byl nahrazen jeho vylepšenou verzí TCP Cubic. Díky své rychlosti dosáhnout maximální přidělené šířky pásma je vhodný pro použití ve vysokorychlostních sítích. [48]

5.1 Propustnost algoritmů TCP

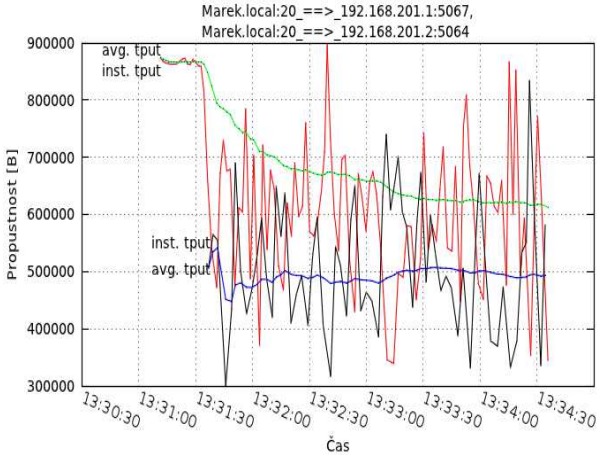
V předchozí kapitole byly popsány jednotlivé tcp algoritmy. Nyní následuje hodnocení tcp algoritmů a to jestli pro každé vytvořené tcp spojení spravedlivě rozdělí dostupnou přidělenou šířku pásma nebo se chovají agresivně a snaží ostatní tcp toky znevýhodnit. Měření bylo prováděno pomocí dvou FTP klientů, kteří vytvořili tcp spojení s FTP serverem. Mezi FTP serverem a klienty byla pro ovlivňování provozu použita aplikace NetDisturb. V aplikaci Netdisturb byla nastaveny postupně rychlosti přenosových linek na 10 Mb/s, 100 Mb/s a spolu s přenosovými rychlostmi se měnila i ztrátovost paketu. Ztrátovost byla nejdříve nastavena tak, že během stahování dat se ztratil jeden paket z definovaného počtu například 1 z 20. Ztrátovost při tomto nastavení byla 5% a během přenosu nebyla měněna. Pro další měření byla použita 5 % náhodná ztrátovost. Ztrátovost do 5 % byla dostačující, protože některé algoritmy měli už s touto ztrátovosti problémy, viz dále. Při náhodné ztrátovosti nad 5 % měli FTP klienti už problém navázat spojení, aniž by došlo k výpadku. Nastavení zpoždění během měření nebylo prováděno, protože ztrátu paketu pozná vysílač až po vypršení časovače nebo po doručení 3dupACK. Pomocí ztrátovosti, lze rychleji ukázat, jak se jednotlivé algoritmy přizpůsobí rychlým změnám podmínek v síti. FTP server byl vytvořen na počítači netbook s operačním systémem Ubuntu verze 11.10 pomocí programu Proftpd. Zachytávání paketů bylo provedeno sítovým zachytávačem tcpdump, programem tcptrace se provedla analýza TCP spojení pro každých 500 přijatých segmentů a pro vykreslování průběhů byl použit gnuplot. Na následujících obrázcích jsou zachyceny závislosti propustnosti jednotlivých algoritmů na čase, kde čas je vyjádřen v hodinách, minutách a sekundách. Ve všech závislostech bylo nejdříve vytvořeno tcp spojení z počítače s IP adresou 192.168.201.1 a po třech procentech stažených dat vytvořeno druhé tcp spojení z počítače s IP adresou 192.168.201.2. Křivky s červenou, černou barvou vyjadřují okamžitou propustnost a zelená s modrou barvou jejich průměrnou propustnost vypočítanou z okamžité propustnosti.

Na obrázku, viz Obrázek 5.1, jsou zobrazeny závislosti pro nastavenou přenosovou rychlost 10 Mb/s a ztrátovost paketu 1 z 20 (5%). Z grafů můžeme vidět, že algoritmy HSTCP, Scalable, HTCP, Vegas a BIC mají agresivní chování to znamená, že při každé ztrátě paketu se dokážou během krátké doby dostat na co největší přenosovou rychlost i přesto, že tím omezí druhé tcp spojení. Méně agresivní jsou Hybla a Cubic, ale i tak dosahují během přenosu vyšší propustnosti. Zbylé algoritmy Westwood, a Veno nejsou tak agresivní, neférové

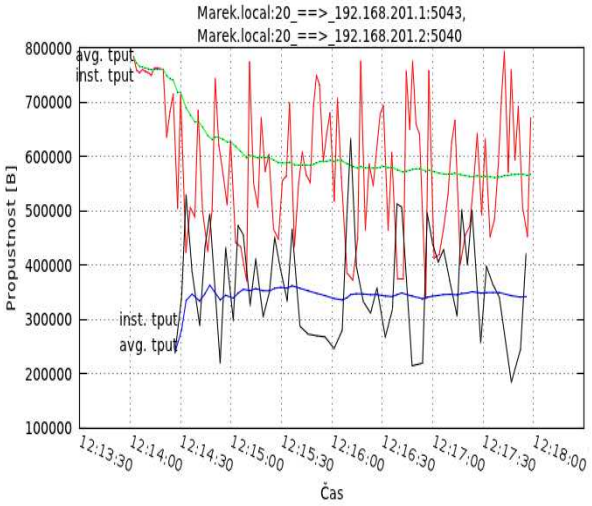
a nesnaží se druhé tcp spojení omezit. Nejlepší propustnosti při těchto nastavených podmínkách dosahují algoritmy Scalable, Cubic, Veno a naopak nejhorší je Vegas.



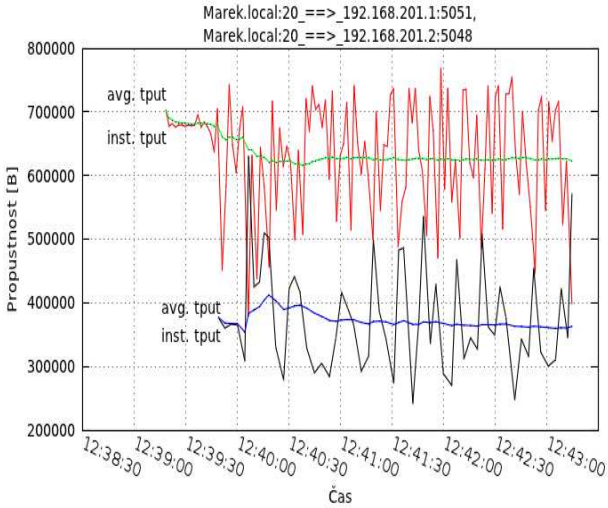
BIC



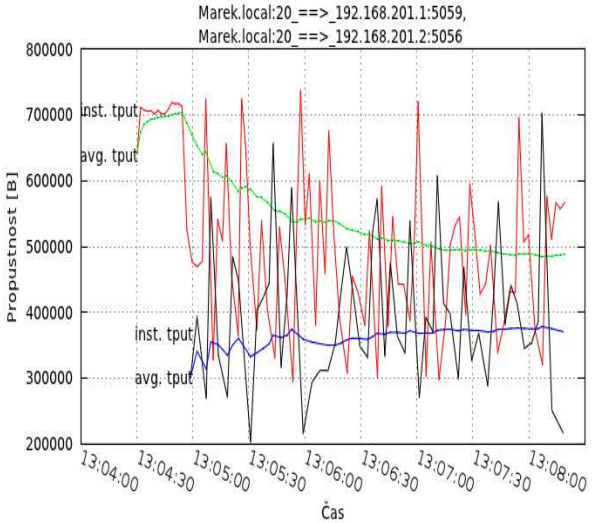
Cubic



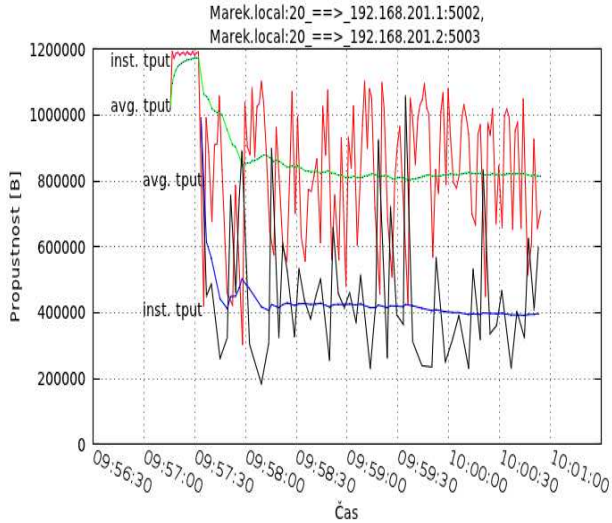
HSTCP



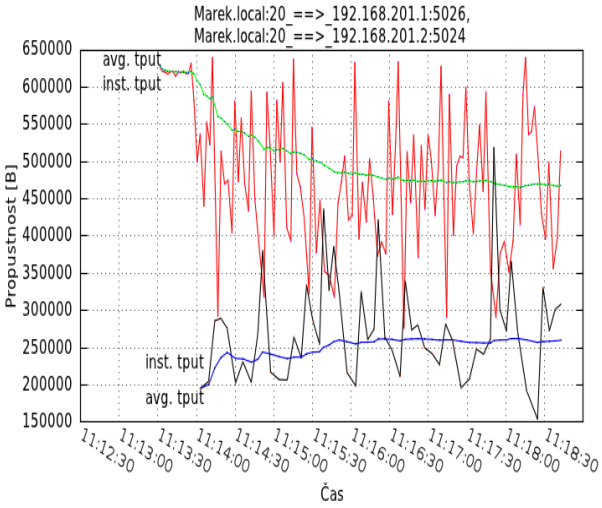
HTCP



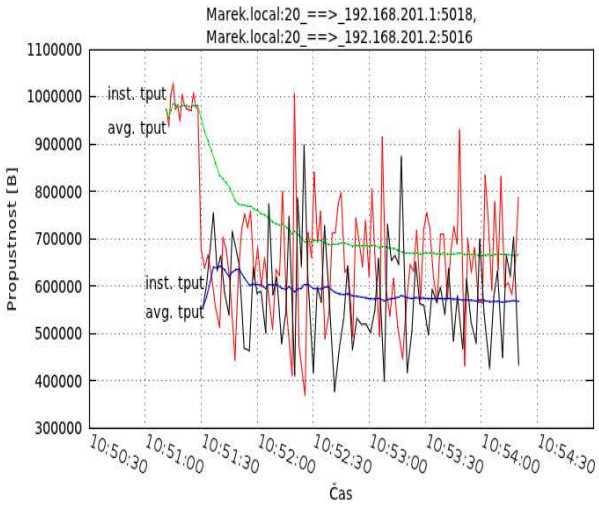
Hybla



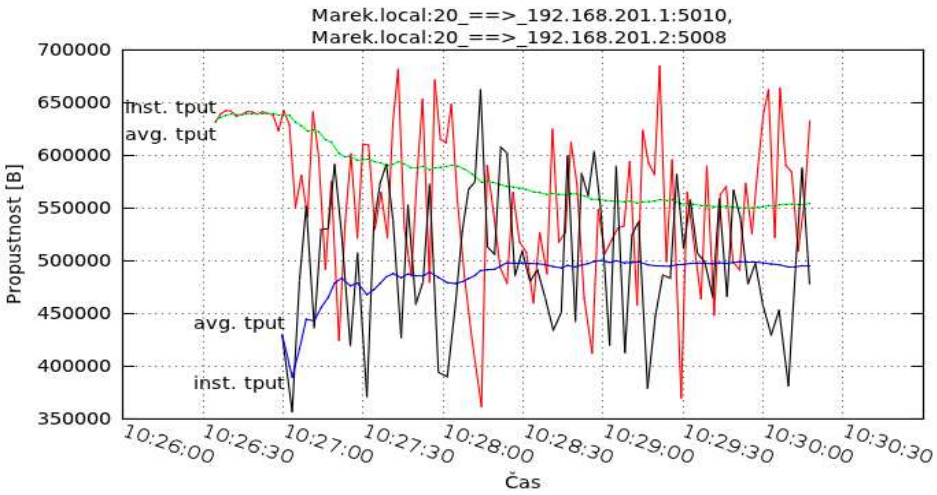
Scalable



Vegas



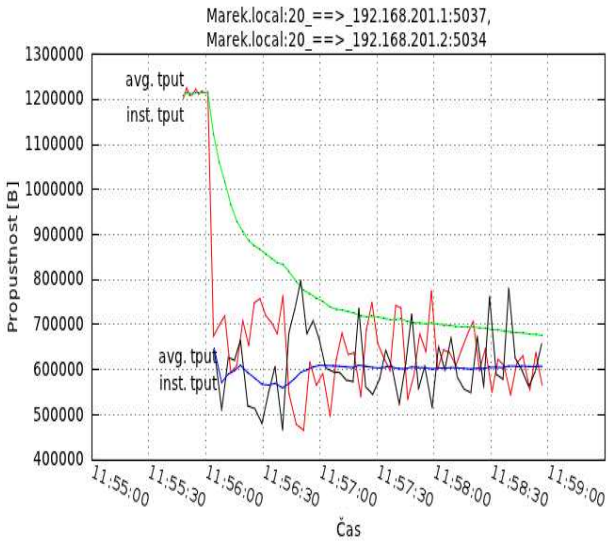
Veno



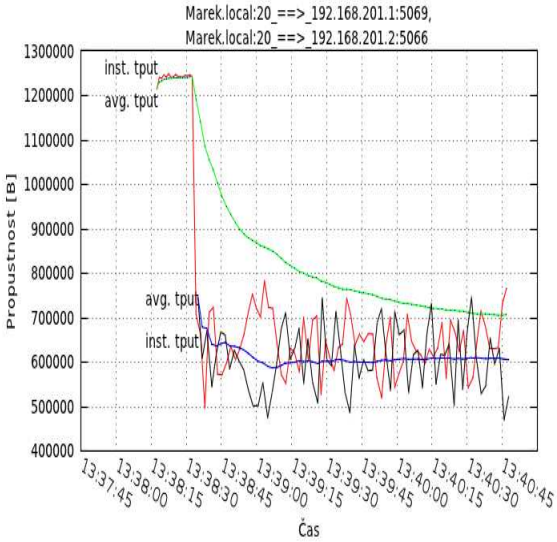
Westwood

Obrázek 5.1: Propustnost TCP algoritmů pro linku 10 Mb/s a ztrátovosti 1 z 20

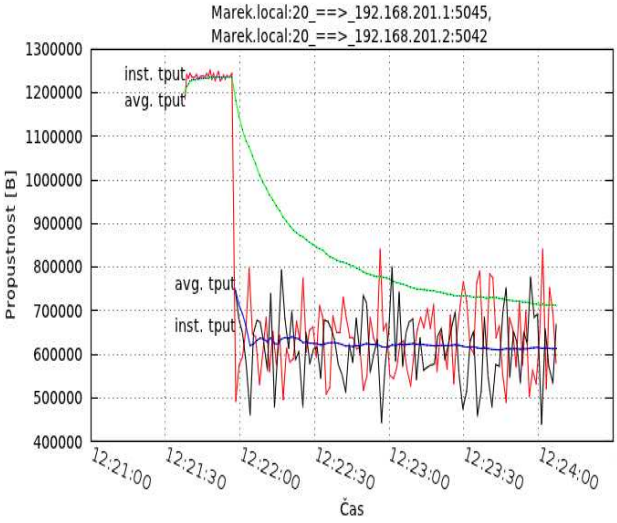
Na obrázku, viz Obrázek 5.2 je nastavená přenosová rychlost na 10 Mb/s a ztrátovost na 1 ze 100 (1 %). Z následujících grafů je vidět, že agresivní chování algoritmů HSTCP, HTCP, Scalable a BIC bylo sníženo, ale pořád se chovají neférově. U algoritmu Cubic, Hybla a Veno je agresivní chování přibližně na stejné úrovni. Zato u Vegas a Westwood dokáže druhé tcp spojení získat větší převahu o získání vyšší přenosové rychlosti. Při této nastavené ztrátovosti a přenosové rychlosti má nejlepší propustnost algoritmus BIC. Zbylé algoritmy jsou na tom přibližně stejně.



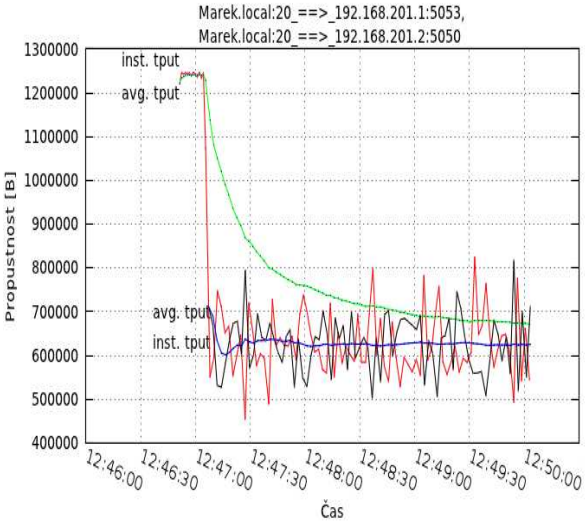
BIC



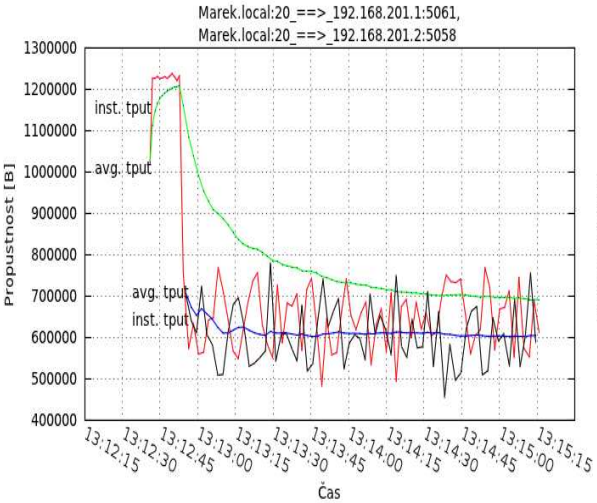
Cubic



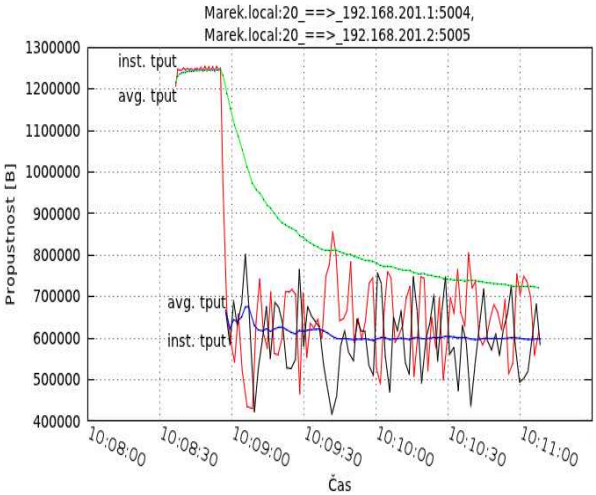
HSTCP



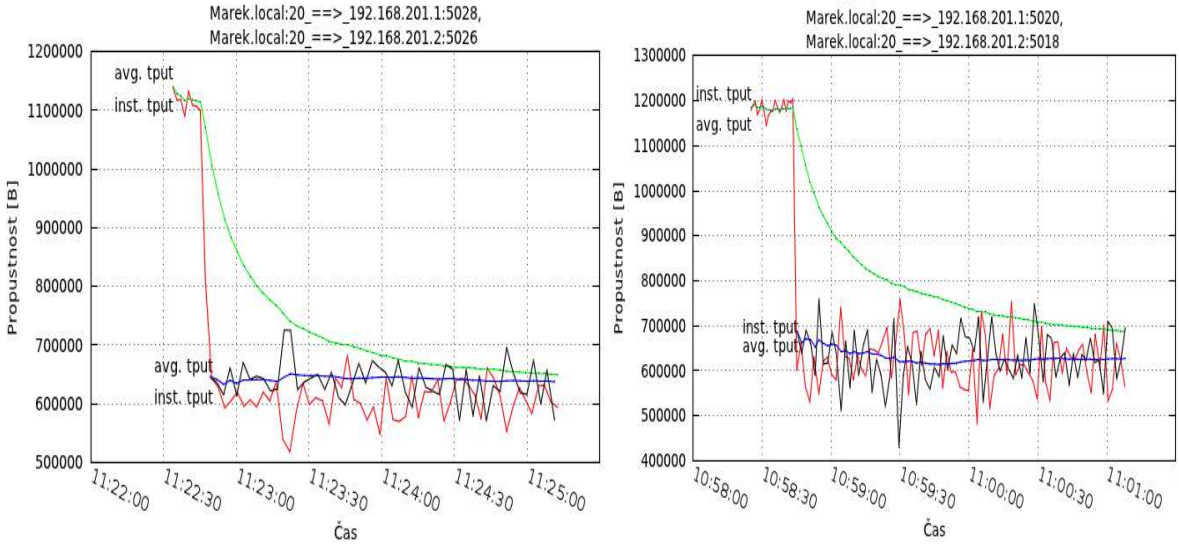
HTCP

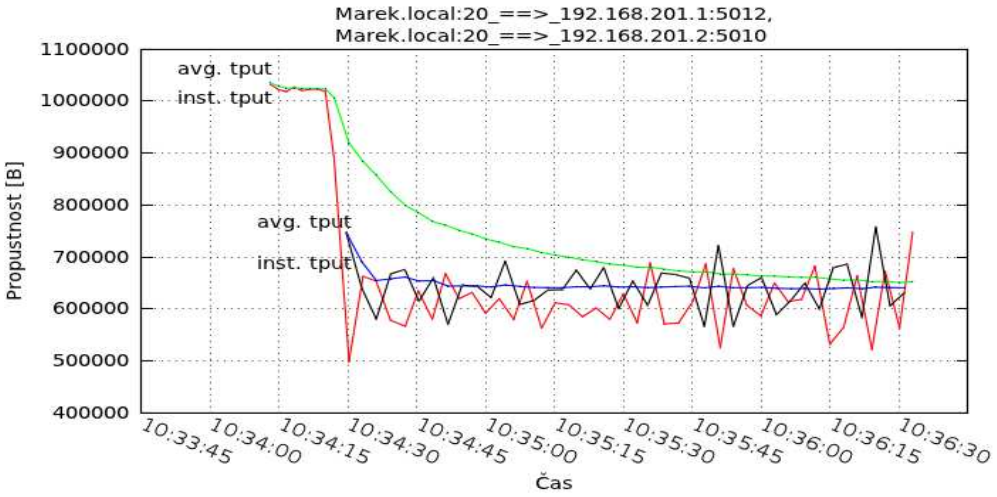


Hybla



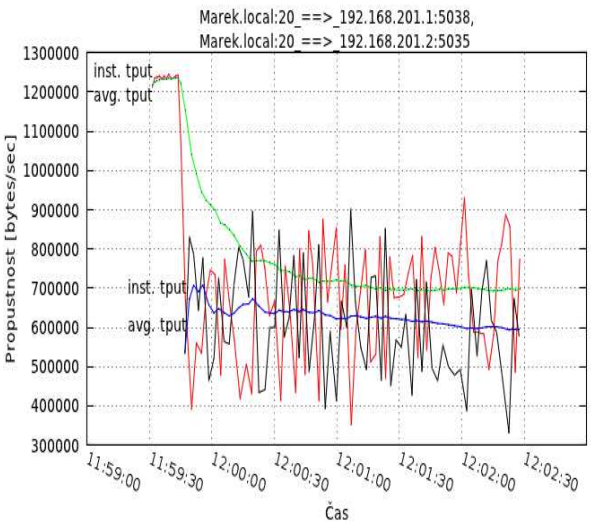
Scalable



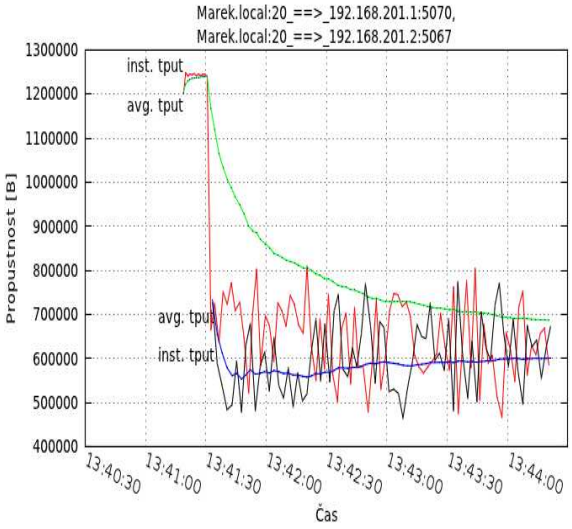


Obrázek 5.2: Propustnost TCP algoritmů pro linku 10 Mb/s a ztrátovosti 1 ze 100

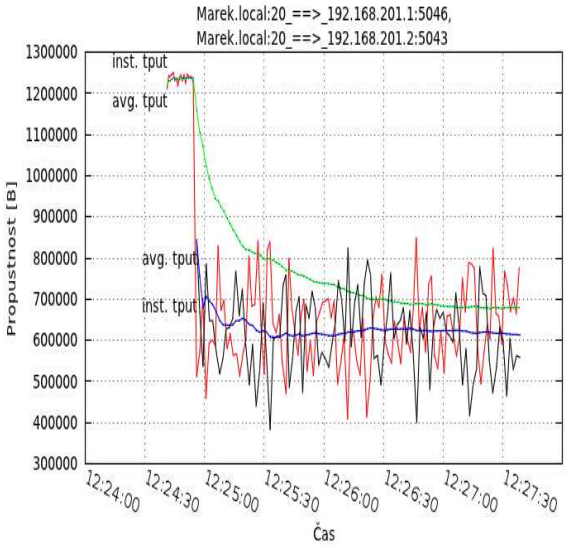
Na obrázku, viz Obrázek 5.3 je měření prováděno pro přenosovou rychlost 10 Mb/s a ztrátovost 1 ze 150 (0,66 %). Při této ztrátovosti je o trochu agresivnější algoritmus BIC a Scalable. Zbylé algoritmy se chovají férově, nesnaží omezit druhé spojení, ale u algoritmů Vegas a Westwood je druhé spojení agresivnější a rychlé využívá pro sebe uvolněnou rychlost při následně ztrátě paketu u prvního spojení. Nastavená nízká ztrátovost nejlépe vyhovuje algoritmu BIC, HSTCP a zbylé algoritmy mají propustnost přibližně stejnou.



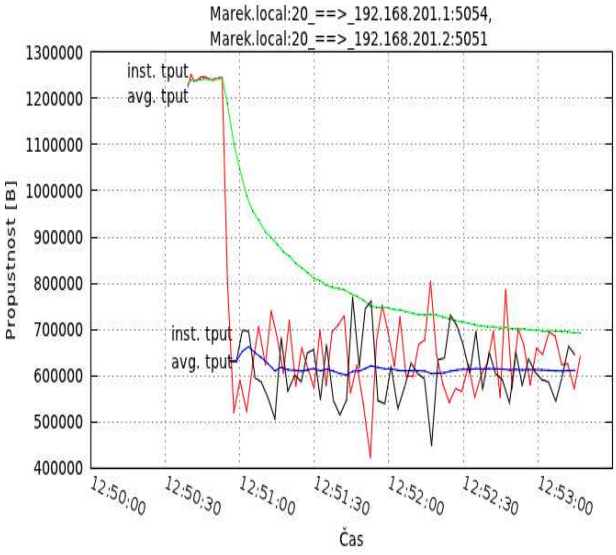
BIC



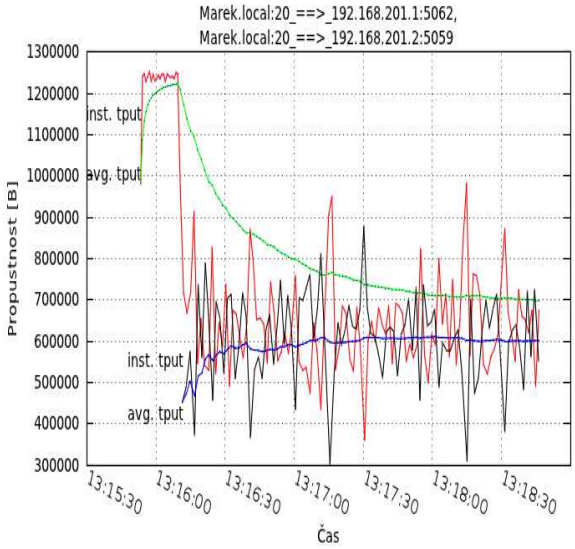
Cubic



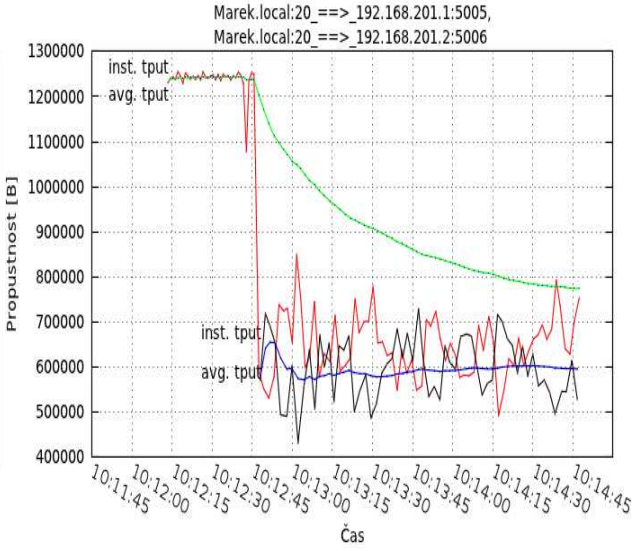
HSTCP



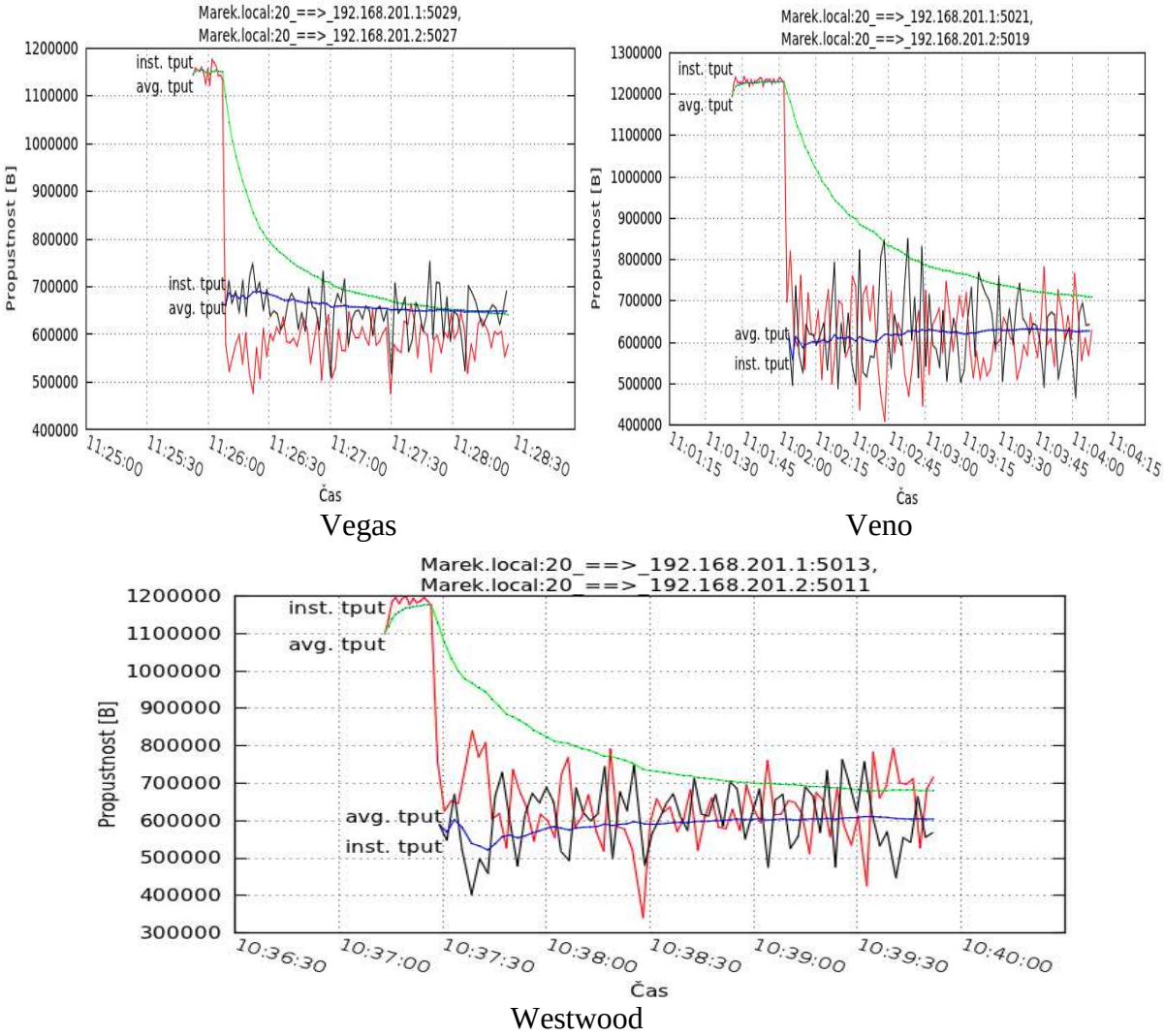
HTCP



Hybla

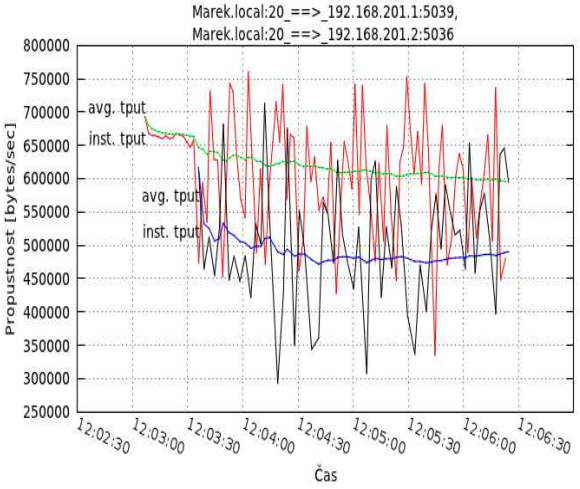


Scalable

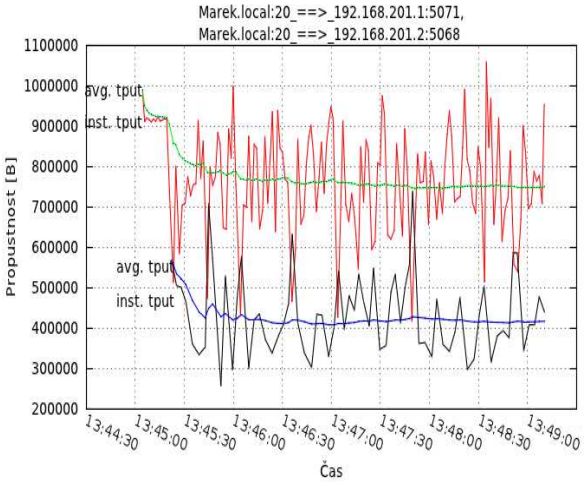


Obrázek 5.3: Propustnost TCP algoritmů pro linku 10 Mb/s a ztrátovosti 1 ze 150

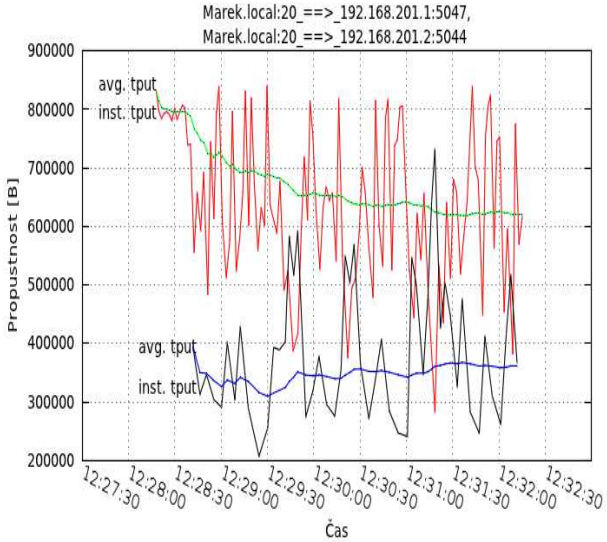
Na obrázku, viz Obrázek 5.4, jsou závislosti průběhu naměřené pro přenosovou rychlost 100Mb/s a ztrátovost 1 z 20 (5%). Největší agresivitu a neférovost mají algoritmy Cubic, Scalable, HSTCP, HTCP, Hybla, Veno, Vegas a BIC. Chování algoritmu Westwood je férovější a méně agresivnější. Nejvyšších přenosových rychlostí dosahují Scalable má kolem 1,1 MB/s, Veno má přibližně 1 MB/s. Nejnižších přenosových rychlostí dosahuje algoritmus Vegas přibližně 480 kB. Nejlepší propustnost má algoritmus Scalable, ale za výrazně snížení přenosové rychlosti druhého spojení. Nastavená přenosová rychlost 100 Mb/s a ztrátovost 5 % vyhovuje algoritmu Veno, nedochází u něho k velkému agresivnímu chování a neférovosti. Přenosová rychlost prvního tcp spojení je kolem 1 MB/s a druhé se pohybuje kolem 700 kB. Nejhorší propustnost má algoritmus Vegas.



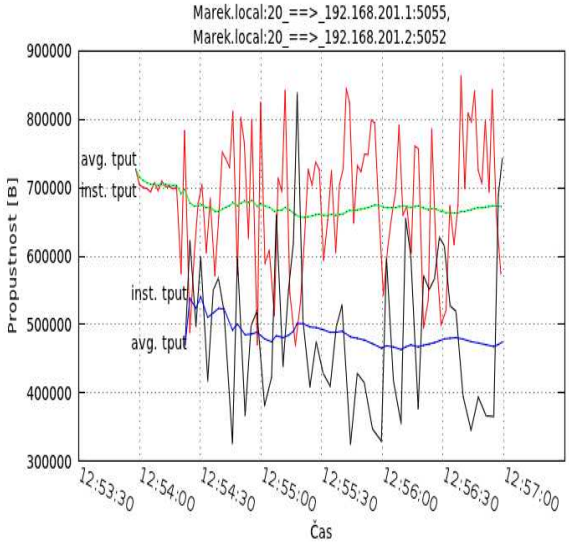
BIC



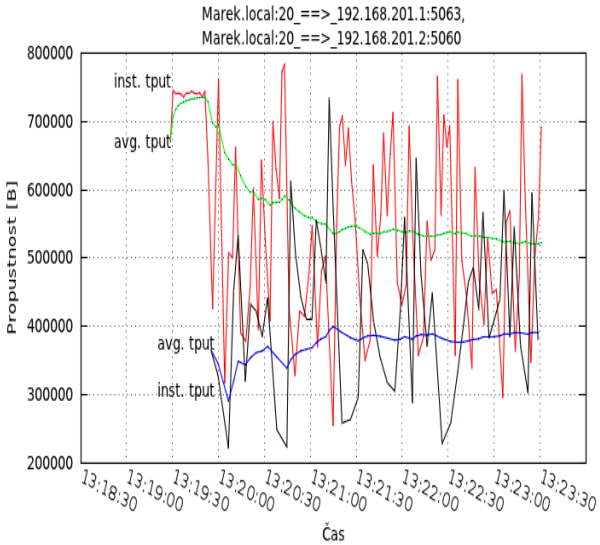
Cubic



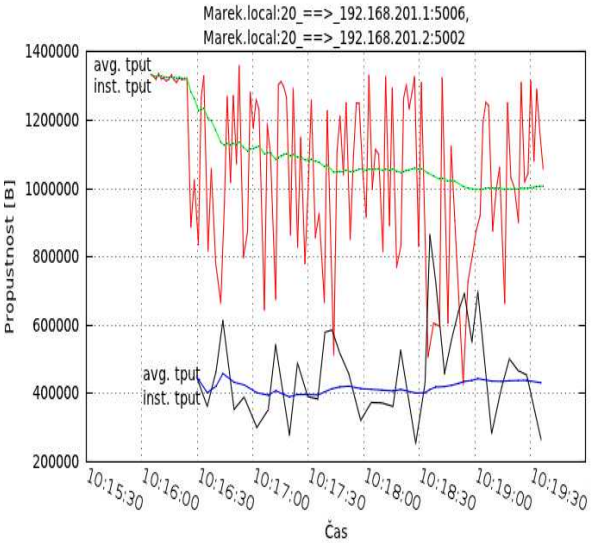
HSTCP



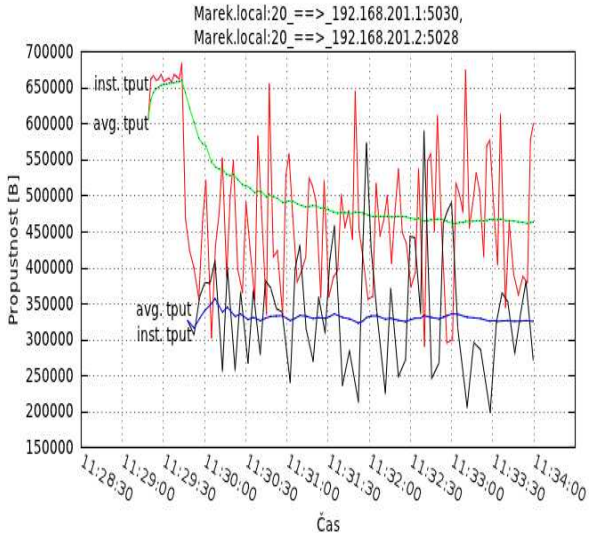
HTCP



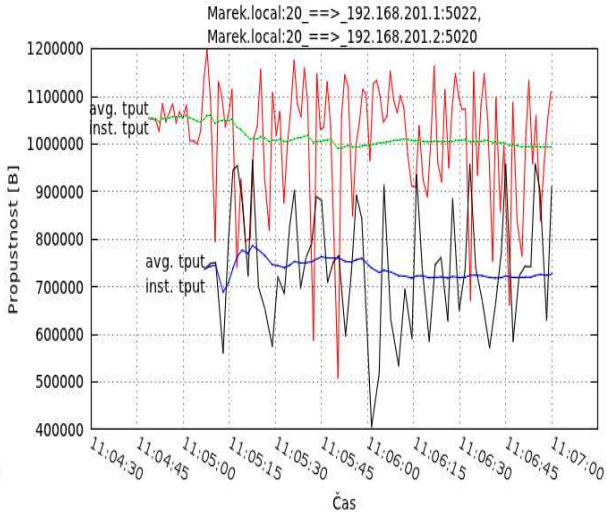
Hybla



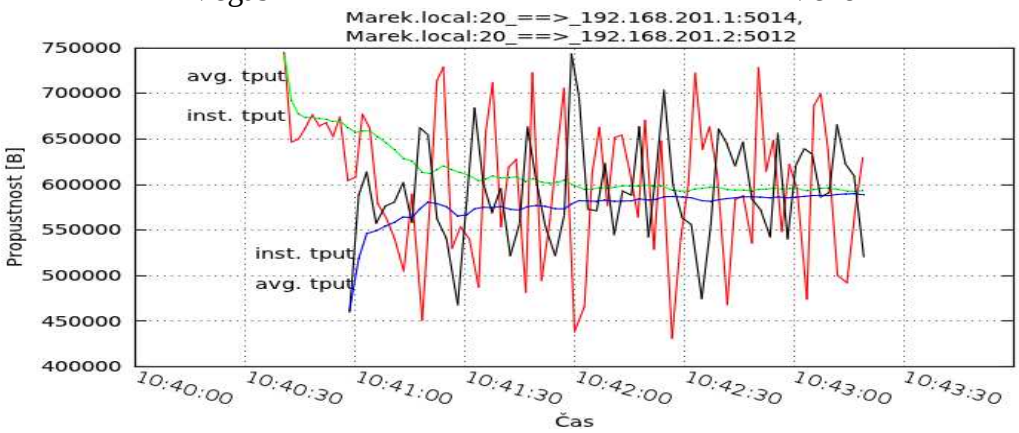
Scalable



Vegas



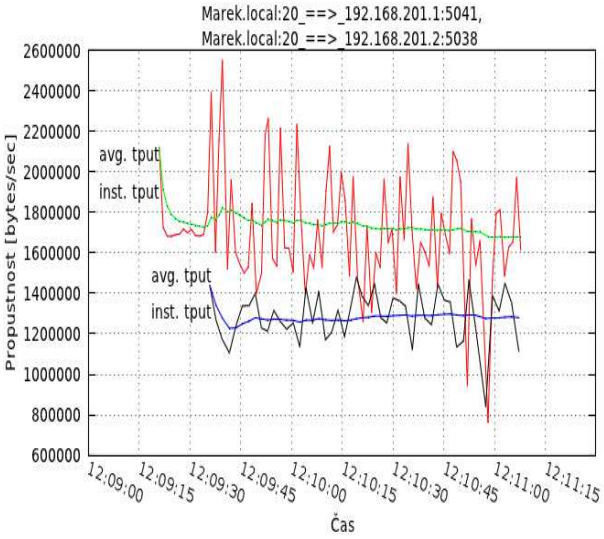
Veno



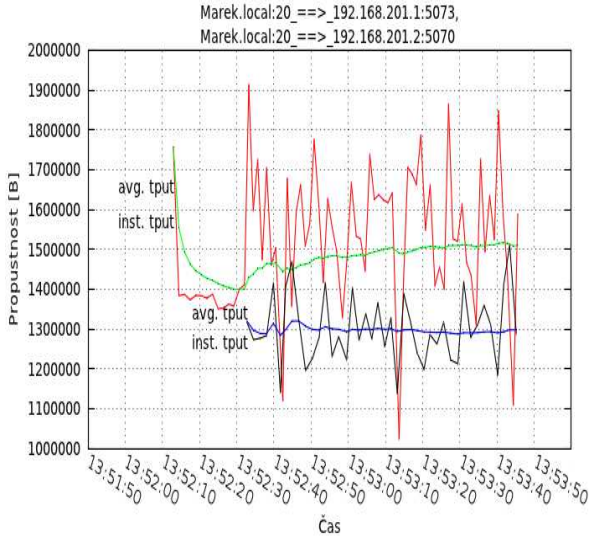
Westwood

Obrázek 5.4: Propustnost TCP algoritmů pro linku 100 Mb/s a ztrátovosti 1 z 20

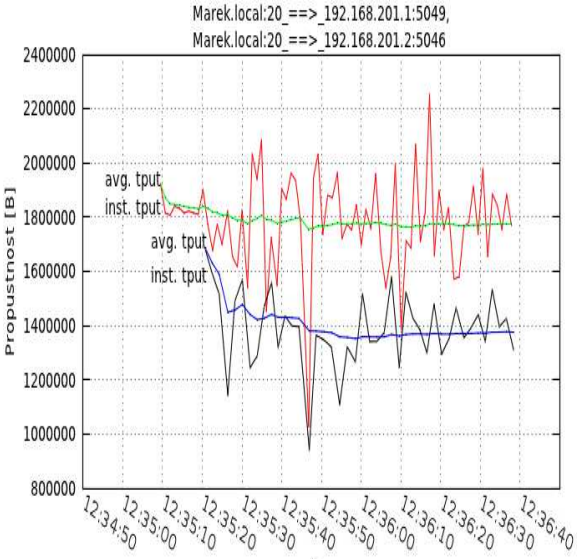
Na obrázku, viz Obrázek 5.5, byla pro měření použita přenosová rychlost 100 Mb/s a ztrátovost 1 ze 100 (1 %). Z grafů je patrné, že při ztrátovosti 1 ze 100 začíná vyhovovat všem algoritmům, chovají se agresivně a neférově kromě Vena a Westwoodu. Tyto výjimky nejsou tak agresivní, jako ostatní a snaží se spravedlivě rozdělit přenosovou rychlost mezi obě spojení. Největší agresivní a neférové chování má opět algoritmus Scalable, jeho propustnost je také nejlepší. Všechny algoritmy mají přenosové rychlosti větší jak 1,2 MB/s.



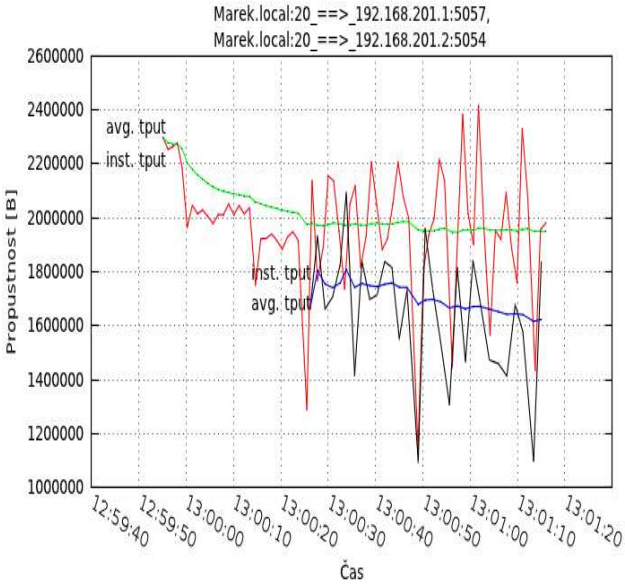
BIC



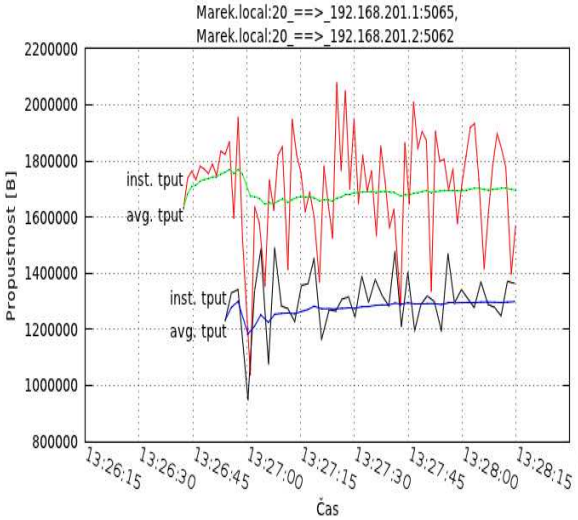
Cubic



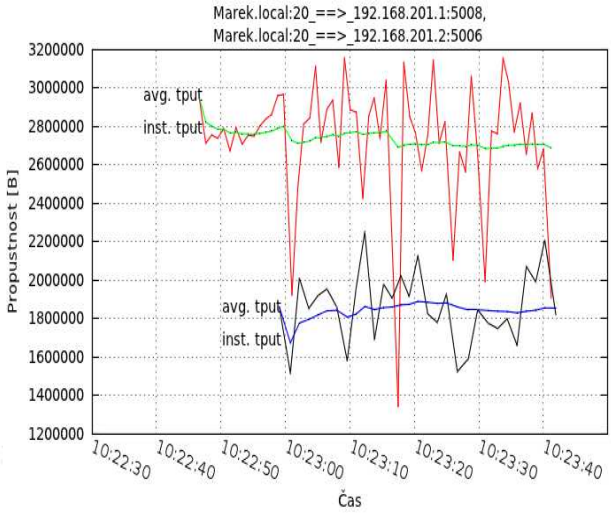
HSTCP



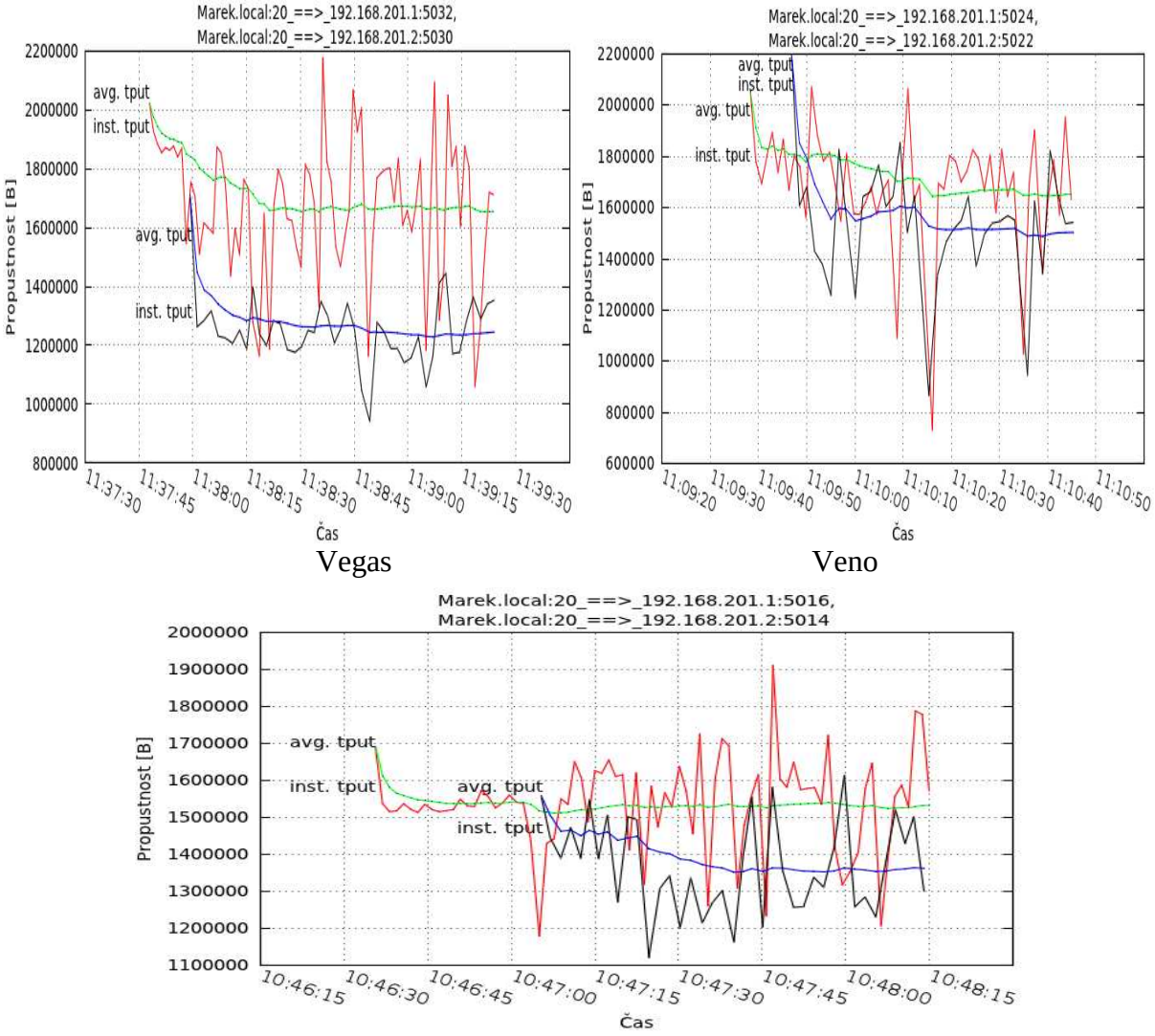
HTCP



Hybla

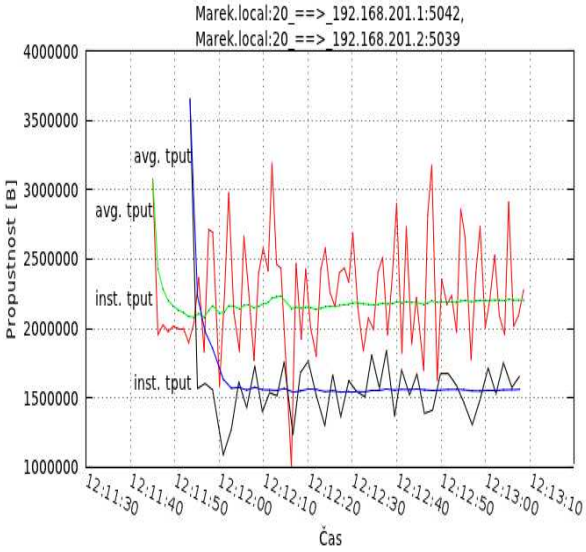


Scalable

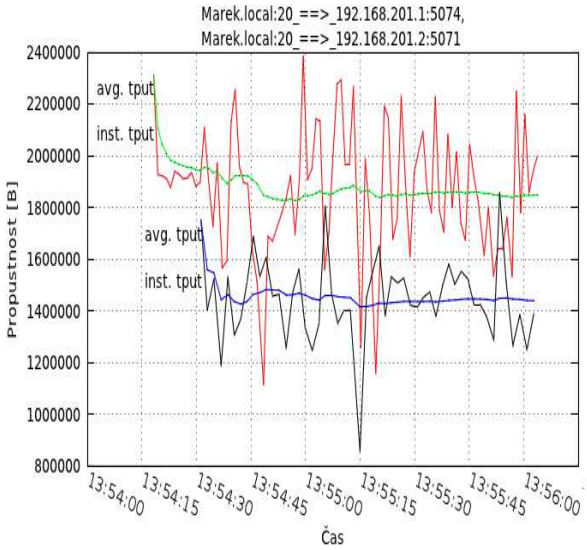


Obrázek 5.5: Propustnost TCP algoritmů pro linku 100 Mb/s a ztrátovosti 1 ze 100

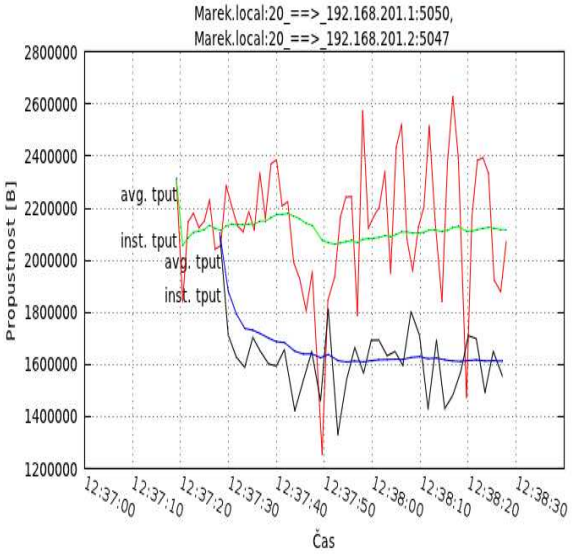
Na obrázku, viz Obrázek 5.6, byla pro měření použita přenosová rychlost 100 Mb/s a ztrátovost 1 ze 150 (0,66 %). Při této nízké ztrátovosti mají všechny algoritmy výbornou propustnost, ale za cenu agresivního a neférového chování. Nejmenší agresivní a férové chování má algoritmus Veno, jeho propustnost pro obě tcp spojení je přibližně 1,8 MB/s. Nejlepší propustnosti dosahuje opět algoritmus Scalable a následně za ním je HTCP.



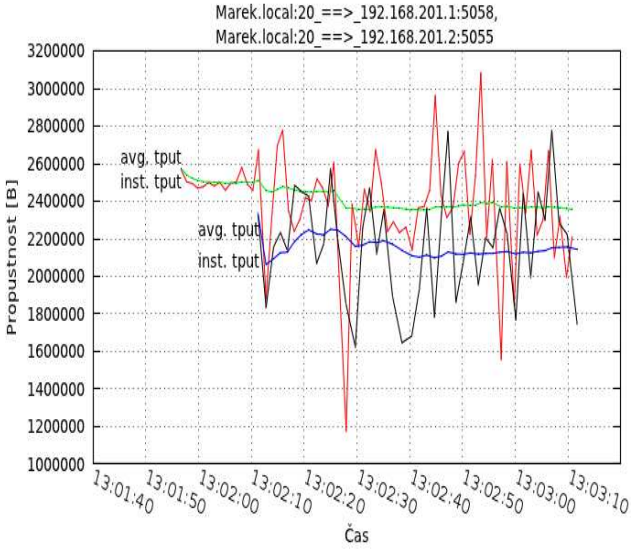
BIC



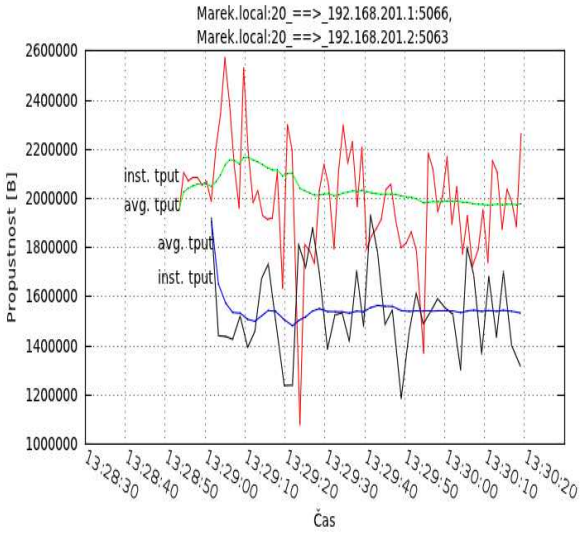
Cubic



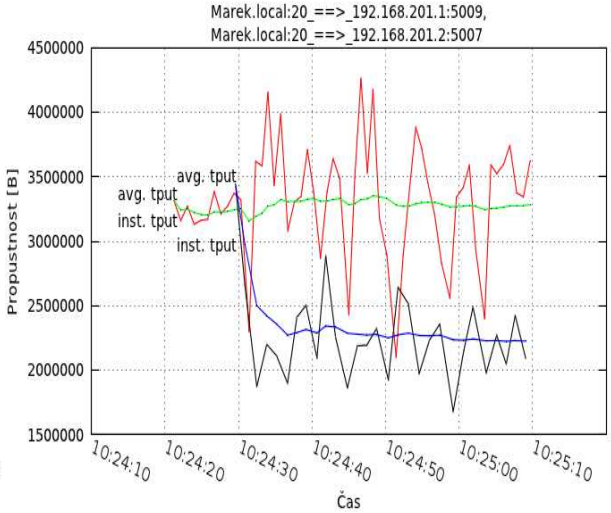
HSTCP



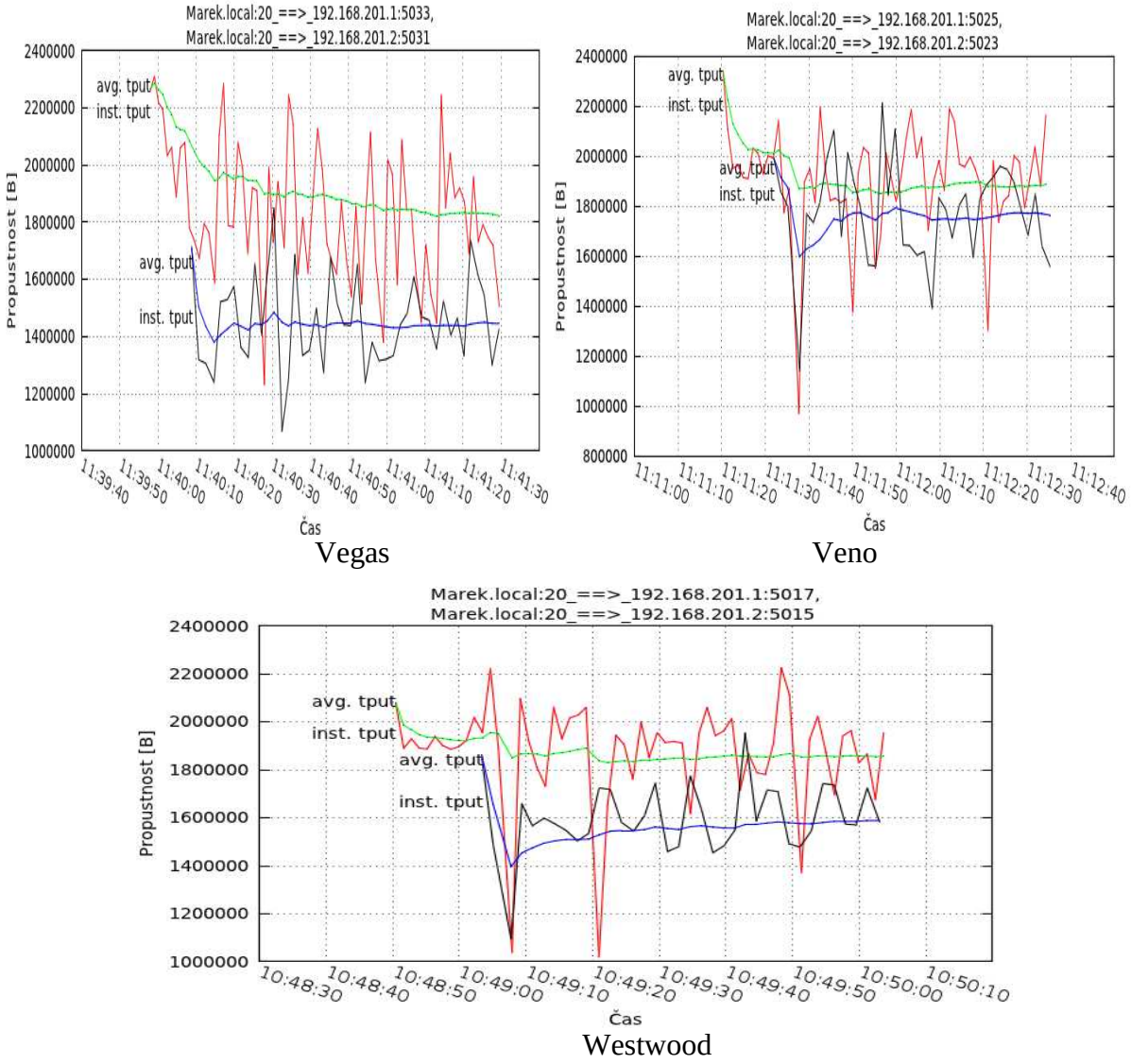
HTCP



Hybla

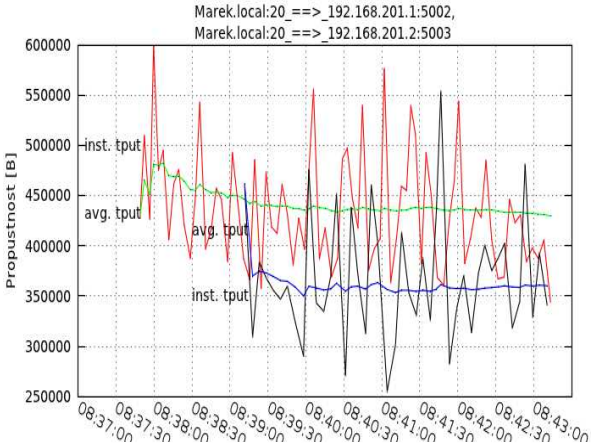


Scalable

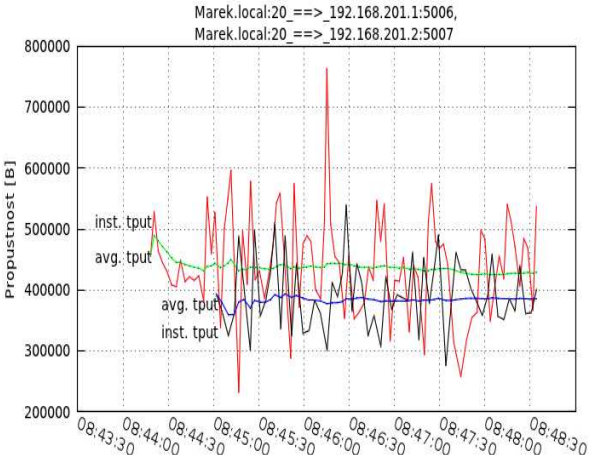


Obrázek 5.6: Propustnost TCP algoritmů pro linku 100 Mb/s a ztrátovosti 1 ze 150

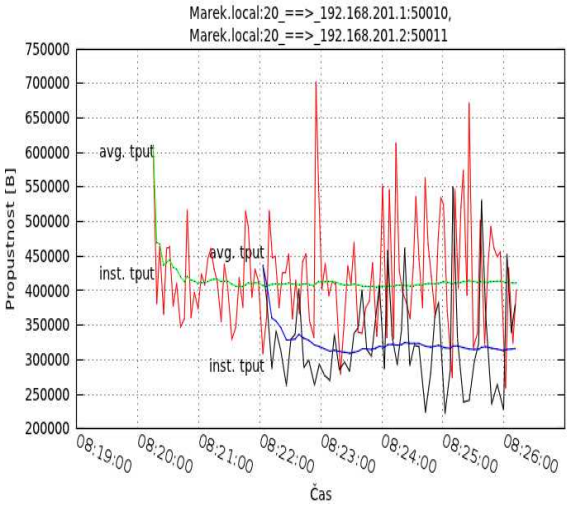
Dále bylo provedeno měření pro náhodnou ztrátovost do 5 % a přenosovou rychlost linky 10 Mb/s. Na obrázku, viz Obrázek 5.7, můžeme z grafů vidět, že nejlepších výsledků dosahují algoritmy Veno, Scalable, HTCP ale i Westwood. Naopak nejhorší je Vegas jeho propustnost se pohybovala pro obě tcp spojení kolem 220 kB/s. Agresivní chování se nejvíce projevuje u HSTCP, BIC a HTCP. Scalable, který měl zatím nejlepší propustnost má při náhodné ztrátovosti problémy. Jeho agresivní chování, není tak znatelné a druhé tcp spojení toho využívá ve svůj prospěch. Algoritmům Westwood a Veno dělá náhodná ztrátovost nejmenší problémy, chování je férové s mírnou agresivitou.



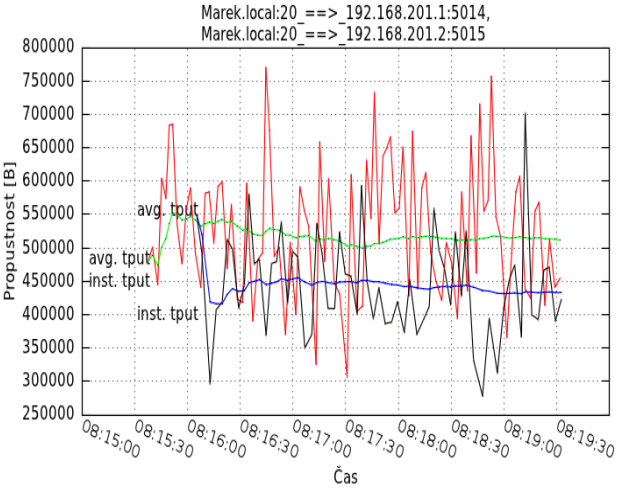
BIC



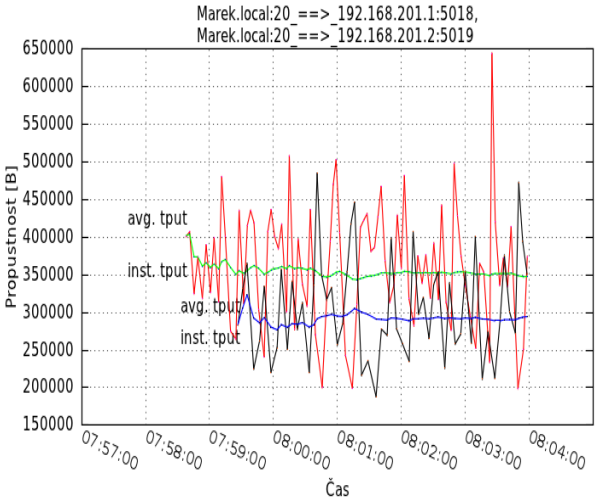
Cubic



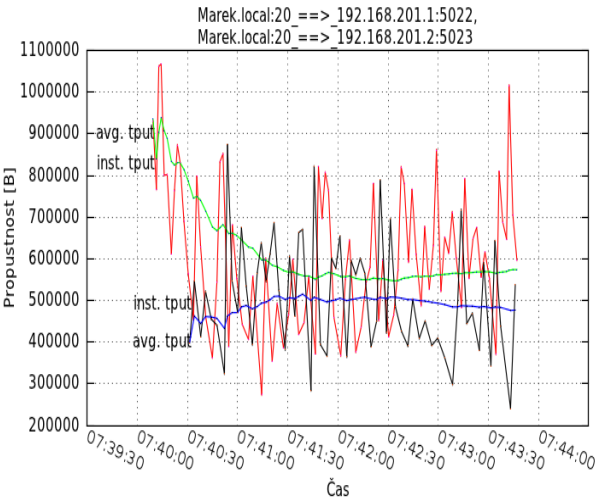
HSTCP



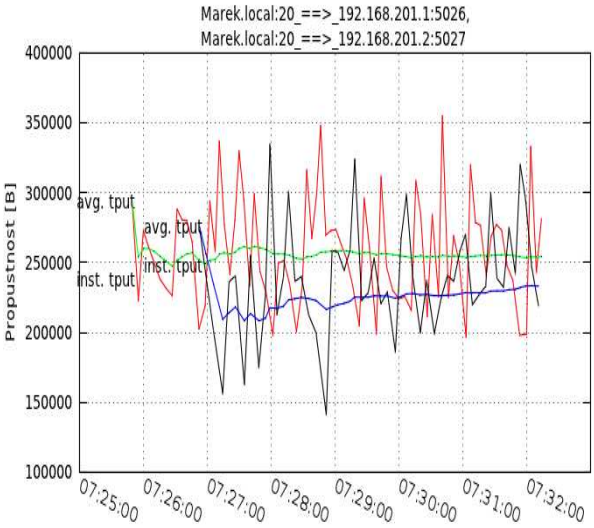
HTCP



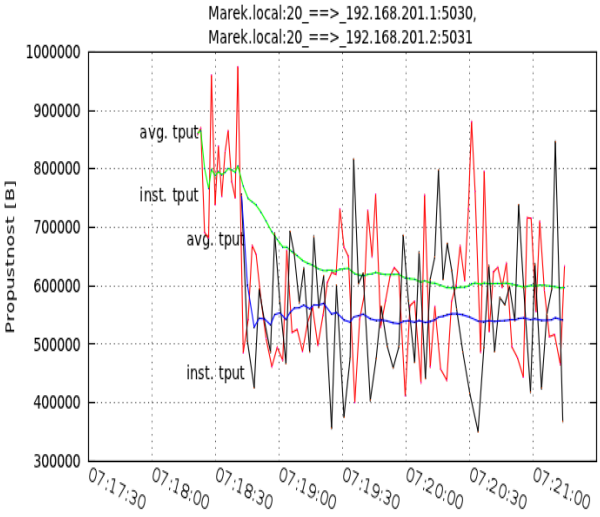
Hybla



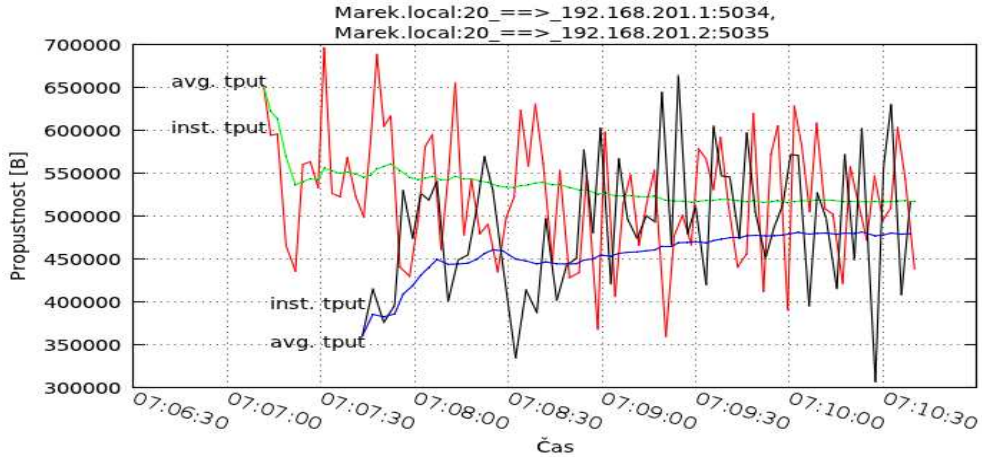
Scalable



Vegas



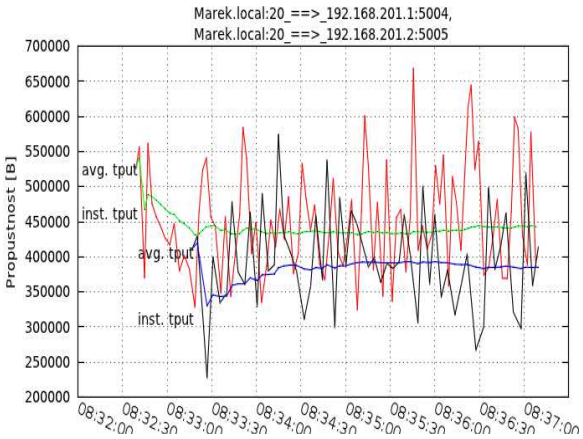
Veno



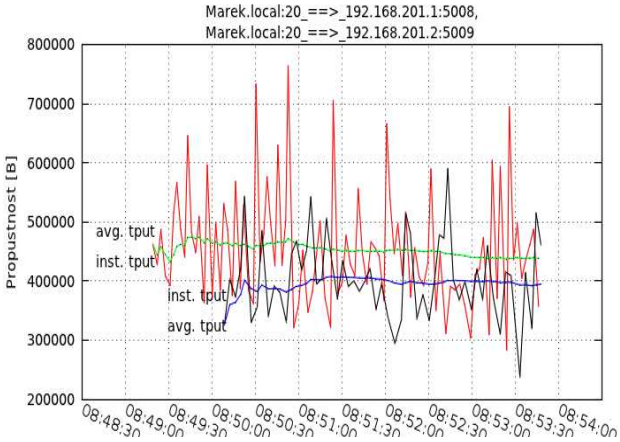
Westwood

Obrázek 5.7: Propustnost TCP algoritmů pro linku 10 Mb/s a náhodná ztrátovost do 5 %

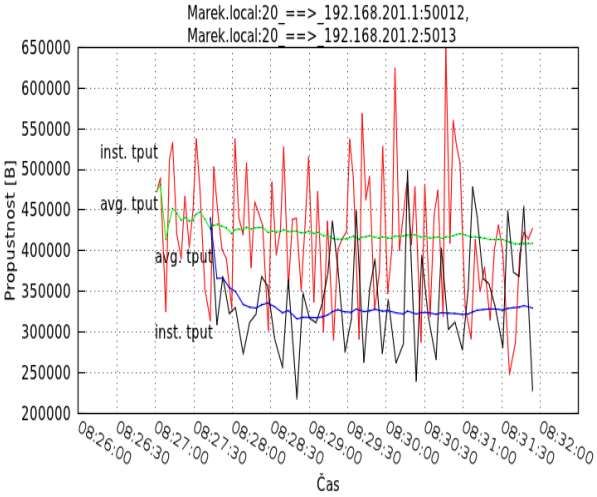
Poslední měření bylo provedeno pro náhodnou ztrátovost do 5 % a přenosovou rychlost 100 Mb/s. Na obrázku, viz Obrázek 5.8 lze grafů vidět, že i když byla zvýšena přenosová rychlost, je propustnost všech algoritmů přibližně stejná kromě Scalable a Veno. Veno nemá s náhodnou ztrátovostí problémy, jeho propustnost byla nejvyšší (950 kB/s) a i přes agresivní chování je propustnost druhého spojení přibližně 650 kB/s. Propustnost druhého spojení bez problému dokázalo překonat i agresivní algoritmy jako BIC, HSTCP, HTCP. Scalable sice dosahuje vyšší propustnosti než u linky s 10 Mb/s, ale omezí tím následně druhé spojení. Nejhorší propustnost má algoritmus Vegas přibližně kolem 250 kB/s.



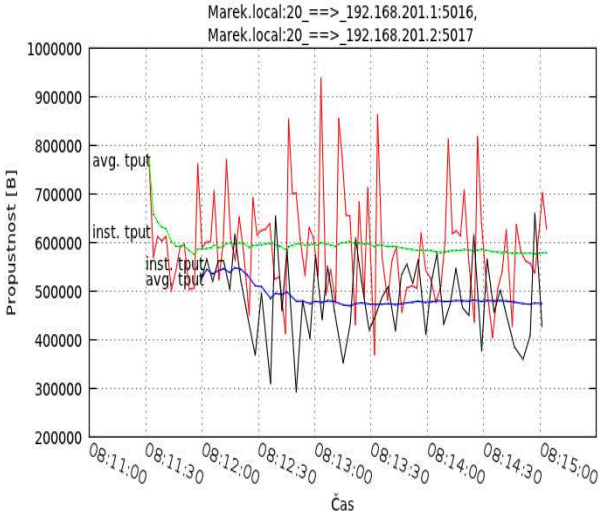
BIC



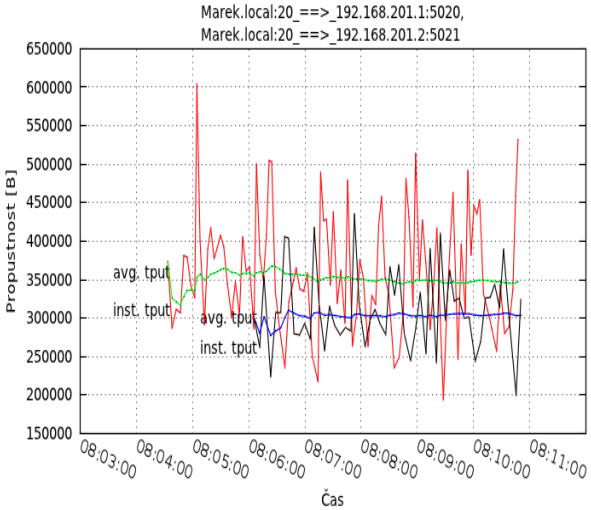
Cubic



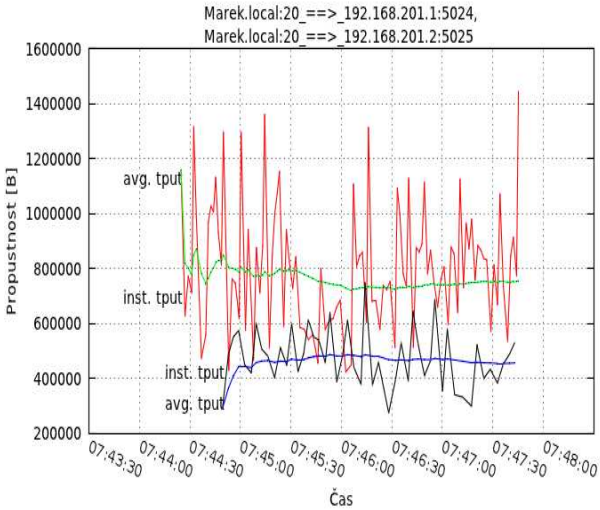
HSTCP



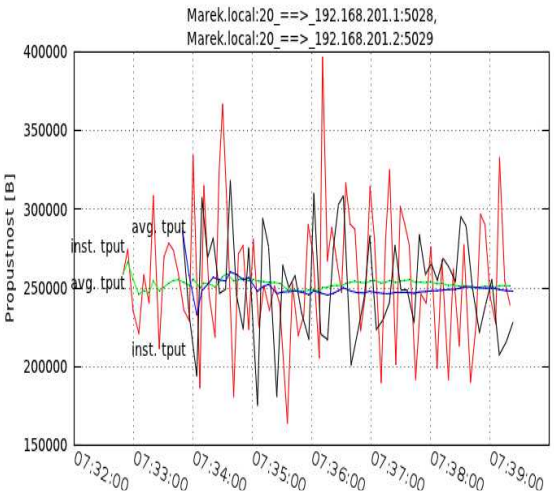
HTCP



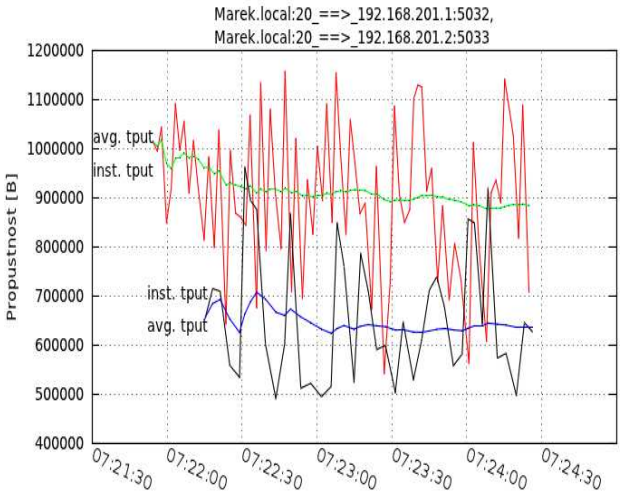
Hybla



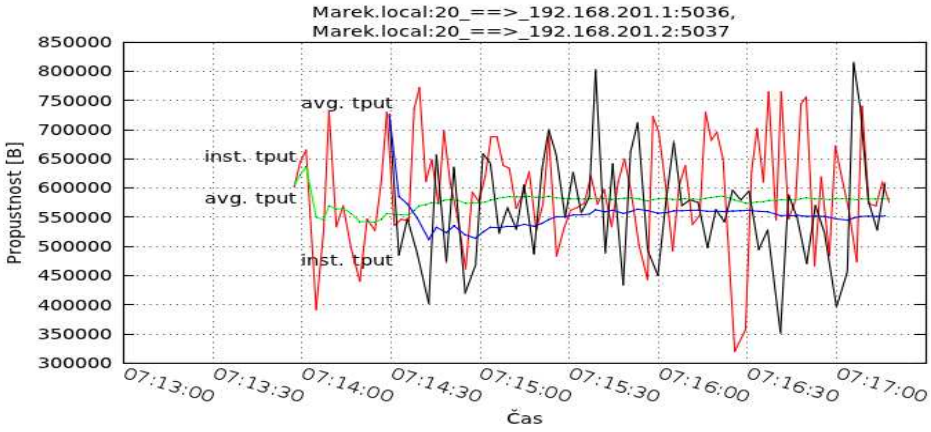
Scalable



Vegas



Veno



Westwood

Obrázek 5.8: Propustnost TCP algoritmů pro linku 100 Mb/s a náhodná ztrátovost do 5 %

5.2 Shrnutí

Algoritmy HSTCP, BIC, Cubic, HTCP, Scalable jsou používány ve vysokorychlostních a měli při náhodně ztrátovosti problémy. Jejich propustnost byla nízká, ale Scalable dokázal dobře pracovat i na přenosové rychlosti 10 Mb/s, kde ostatní vysokorychlostní algoritmy zaostávali. Zato u ztráty jednoho paketu z definovaného počtu vůbec neměli problémy a dokázali rychle včas na ztrátu reagovat. Z náhodné ztrátovosti nejlépe dopadl algoritmus Veno, jeho reakce na ztrátu paketu byla výborná a při přenosové rychlosti 10 Mb/s měl mírnější agresivní chování. Svým chováním a reagováním na ztrátu paketů, je Veno nejčastěji používán v bezdrátových sítích. Největší problémy s náhodnou ztrátovostí měl algoritmus Vegas, jeho propustnost byla nejnižší, ale jeho chování k druhému spojení bylo férové. Pro domácí použití je dostačující algoritmus Westwood, má férové chování s mírnou agresivitou. Jeho propustnost při náhodné ztrátovosti je kolem 530 kB/s pro linku 10 Mb/s a 550 kB/s pro linku 100 Mb/s.

ZÁVĚR

Všechny dostupné síťové služby se vyvíjejí, snaží se mít co nejlepší kvalitu a proto je potřeba zajistit dodržení požadovaných přenosových parametrů v hodnotách, které jsou pro danou službu a úroveň kvality stanovené. Komunikace nikdy nebude dokonalá (bezchybná) a nedodržením požadavků na provozované služby dochází k narušení jejich plynulosti. Všechny problémy se snaží řešit kvalita služeb – QoS, která zabraňuje zahlcení datového spoje a zajistí realizované službě dostatek prostředků během cesty přes komunikační síť.

V práci byly uvedeny, jaké parametry ovlivňují kvalitu síťových služeb, patří mezi ně například zpoždění, kolísání zpoždění, ztrátovost, propustnost atd. Byla popsána kvalita služeb pro hlasovou komunikaci, videokonverzaci, streamovaná audia, streamovaná videa a data. Nakonec jsou tyto služby rozděleny do čtyř tříd QoS.

V další části práce se nacházejí jednotlivá zpoždění, která také napomáhají k nedodržení požadovaných přenosových parametrů, jež jsou pro různé druhy služeb důležité, to znamená vytvoření paketů u odesílatele a poslat ho přes mezilehlá síťová zařízení, použitá přenosová media až k příjemci, kde dojde k rozbalení paketů a získání zprávy, kterou sám příjemce požaduje.

K testování různých komunikačních služeb byla použita aplikace NetDisturb pomocí, které můžeme sledovat jejich chování při zhoršených podmínkách v síti. NetDisturb umožňuje vytvářet takové podmínky, které se mohou vyskytnout v reálném prostředí. Byly zde popsány edice Standard, Enhanced a jejich hlavní rozdíly. Hlavním rozdílem je, že edice Enhanced umožňuje filtrovat provoz podle konkrétních protokolů (RTP, HTTP, SIP atd.).

Dále v práci byl popsán transportní protokol TCP, jeho vlastnosti, postup při navazování a ukončování spojení. Je zde vysvětlen i nejdůležitější parametr TCP protokolu tj. časovač. A nakonec zde byly vysvětleny mechanismy, jejichž cílem je předcházet zahlcení.

V poslední kapitole této práce byl proveden rozbor jednotlivých TCP algoritmů a vysvětleno jak dovedou reagovat na ztrátu segmentů a blížící se zahlcení. Potom byla provedena analýza těchto algoritmů z hlediska jejich propustnosti, férovosti a agresivity. V aplikaci NetDisturb byly nastaveny různé ztrátovosti nejprve, že došlo ke ztrátě jednoho paketu z definovaného počtu například 1 ze 100 (jeden paket ze sta byl ztracen) a potom náhodná ztrátovost. Měření byly provedeny pro přenosové rychlosti 10 Mb/s a 100 Mb/s. Zpoždění nebylo uvažováno, protože vysílač detekuje ztrátu segmentu buď vypršením časovače nebo přijetím 3dupACK. Nastavením různých ztrátovostí se rychleji ukáže, který algoritmus dokáže včas reagovat na ztrátu segmentu a přizpůsobit se. Z algoritmů používaných ve vysokorychlostních sítích měl při náhodné ztrátovosti nejlepší výsledky algoritmus Scalable. Na přenosové rychlosti 10Mb/s byl méně agresivní než na vyšší přenosové rychlosti. Jeho propustnost pro linku 10 Mb/s byla přibližně 520 kB/s a pro linku 100 Mb/s přibližně 800 kB/s. Ze všech algoritmů si dokázal s náhodnou ztrátovostí nejlépe poradit algoritmus Veno. Na lince 10 Mb/s byl méně agresivní a snažil se spravedlivě rozdělit rychlost linky mezi obě tcp spojení, propustnost byla přibližně 580 kB/s. Na lince s vyšší rychlosti jeho agresivita stoupla, ale i přes tuto zvýšenou agresivitu mělo druhé tcp spojení výbornou propustnost. Propustnost prvního tcp spojení byla přibližně 900 kB/s a druhé tcp spojení získalo pro sebe přibližně 610 kB/s. S náhodnou ztrátovostí měl největší problémy algoritmus Vegas. U obou přenosových rychlostí dosahovala jeho propustnost kolem 250 kB/s, ale na druhou stranu se k ostatním tcp spojením snažil chovat férové. Pro domácí použití jsou vhodné algoritmy Westwood nebo HTCP (je používán ve vysokorychlostních sítích). Westwood má mírné agresivní chování, ale snaží se spravedlivě rozdělit dostupnou přenosovou rychlost linky. Propustnost u linky s rychlostí 10 Mb/s byla pro první tcp spojení

přibližně 520 kB/s, druhé tcp spojení mělo 480 kB/s a pro linku s rychlostí 100 Mb/s byla propustnost u obou tcp spojení přibližně 550 kB/s. HTCP je o trochu agresivnější než Westwood, ale i při této agresivitě nedochází ke značnému omezení druhého tcp spojení. Propustnost linky s rychlostí 10Mb/s byla u prvního tcp spojení přibližně 520kB/s, druhé tcp spojení mělo přibližně 450 kB/s a pro linku s rychlostí 100 Mb/s byla propustnost u prvního tcp spojení přibližně 600 kB/s a u druhého tcp spojení přibližně 500 kB/s. Při ztrátě jednoho paketů z definovaného počtu prokázaly nejlepší výsledky algoritmy používané ve vysokorychlostních sítích. Zbylé TCP algoritmy může uživatel použít, pokud mu nevadí, že jeho další vytvořená tcp spojení budou omezována neférovým chováním některých algoritmů.

Součástí práce bylo vytvořit dvě laboratorní úlohy. Kde v první laboratorní úloze by si studenti mohli vyzkoušet sami ovlivňovat provoz a sledovat co se s ním během přenosu přes komunikační síť děje. Vyzkouší si vytvořit streamované video, streamované audio, přenos dat pomocí protokolu FTP, telefonní komunikaci a videokonverzaci. V druhé laboratorní úloze budou provádět analýzu TCP algoritmů. Součástí této úlohy je i vytvoření FTP serveru na operačním systému Ubuntu. Při analýze TCP algoritmů se zaměří na jejich propustnost při omezených podmínkách v síti, dále budou sledovat, férovost jednotlivých algoritmů ale i jejich agresivní chování.

SEZNAM LITERATURY

[1] PUŽMANOVÁ, Rita. *Moderní komunikační sítě od A do Z.* 1. vydání. [s.l.] : Computer Press, 1998. 446 s. ISBN 80-7226-098-7.

[2] BEZPALEC, P. *Analýza zpoždění v IP telefonním systému I.* [online]. 9. 5. 2008 [cit. 2011-10-10]. Dostupný z WWW: <http://access.feld.cvut.cz/view.php?cisloclanku=2008050004/>.

[3] MOLNÁR, Karol. *Teorie front*. c2009. Dostupný z WWW: <www.utko.feec.vutbr.cz/ molnar/>. s. 38.

[4] *Intelek* [online]. 2011 [cit. 2011-10-1]. Dostupný z WWW: <www.intelek.cz>.

[5] PUŽMANOVÁ, Rita. *Moderní komunikační sítě od A do Z.* 2. aktualiz. vyd. [s.l.] : Computer Press, 2006. 432 s. ISBN 80-251-1278-0.

[6] PETERKA, J. *Aktivní síťové prvky* [online]. 1998 [cit. 2011-09-25]. Dostupný z WWW: < http://www.earchiv.cz/a94/a438c500.php3 >.

[7] SATHAYE, S. *What is layer 4 Switching* [online]. 2011 [cit. 2011-09-28]. Dostupný z WWW: <http://www.nanog.org/meetings/nanog15/.../alteon.ppt>.

[8] CIAMPA, R. *What is layer 3 switching* [online]. 2011 [cit. 2011-09-28]. Dostupný z WWW: <http://www.pulsewan.com/data101/layer3_switching_basics.htm >.

[9] *Úvod do Quality of Service a DiffServ* [online]. 2009, [cit. 2011-10-8]. Dostupný z WWW: < http://www.samuraj-cz.com/clanek/cisco-qos-1-uvod-do-quality-of-service-a-diffserv/>.

[10] *Řízení šířky pásma a QoS* [online]. 2009, [cit. 2011-10-9]. Dostupný z WWW: <http://manuals.kerio.com/control/adminguide/cz/chap-bwmgmt.html >.

[11] DONAHUE, Gary A. *Kompletní průvodce síťového experta.* 1. vydání. : Computer Press, 2009. 528 s. ISBN 978-80-251-2247-1.

[12] KLAŠKA, L. *Základní kvalitativní parametry sítě – latency (zpoždění).* [online]. 2006, [cit. 2011-09-20]. Dostupný z WWW: <http://www.svetsiti.cz/clanek.asp?cid=2617>.

[13] PETERKA, J. *Optické kabely.* [online]. 2004 [cit. 2011-09-27]. Dostupný z WWW: <http://www.earchiv.cz/a92/a208c110.php3>.

[14] KRUMNIKL, Michal. *Měření latence sítových prvků.* 2005, [cit. 2011-09-30].Dostupný z WWW: <http://www.cs.vsb.cz/grygarek/SPS/projekty0405/LatenceSW_kru106.pdf>.

[15] HOUSER, P. *Komprese VoIP ušetří pásmo, ale představuje riziko* [online]. 2009 [cit. 2011-10-21]. Dostupný z WWW: <http://securityworld.cz/securityworld/komprese-voip-usetri-pasmo-ale-predstavuje-riziko-511>.

[16] PUŽMANOVÁ, R.; VOZŇÁK, M. *Kvalita služby pro VoIP*. [online]. 2009, [cit. 2011-10-20]. Dostupný z WWW: <http://www.cesnet.cz/sdruzeni/napsali-o-nas/2009/04/20090415_Professional_Computing.html>.

[17] BOVY, C. J.; MERTODIMEDJO, H. T.; HOOGHIEMSTRA, G.; UIJTERWAAL, H.; VAN MIEGHEM, P. *Analysis of End-to-End Delay Measurements in Internet.* 2002, [cit. 2011-10-25]. Dostupný z WWW: <http://www.csse.monash.edu.au/hons/projects/2002/Siuwing.Szeto/end_to_end_delay.pdf>.

[18] SZIGETI, T.; HATTINGH, Ch. *End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs.* 1. s.l. : Cisco Press, 2004. 768s. ISBN 978-1-58705-390-0. Dostupný z WWW: <http://www.scribd.com/doc/53153739/28/QoS-Design-Overview>

[19] *Ethernet - CSMA/CD, kolizní doména, duplex* [online]. 2007 [cit. 2011-10-23]. Dostupný z WWW: <http://www.samuraj-cz.com/clanek/ethernet-csmacd-kolizni-domena-duplex/>

[20] *HP ProCurve switch 2510-48* [online]. 2007 [cit. 2011-10-29]. Dostupný z WWW: <http://www.64bit.cz/hp-procurve-gigabitovy-switch-2510-48-48-portu/>.

[21] SOSINSKY, B. *Počítačové sítě*. 1. vydání, Brno: Computer Press, 2010. 840s. ISBN 978-80-251-3363-7.

[22] NING, Hua. *IPv6 Core Router Test Report.* 2003, [cit. 2011-10-30]. Dostupný z WWW: <http://www.ipv6-tf.com.pt/documentos/geral/bii_v6_interop.pdf>.

[23] *Internet Core Router Test* [online]. 2001 [cit. 2011-10-30]. Dostupný z WWW: <http://www.lightreading.com/document.asp?doc_id=4009&page_number=7>.

[24] RUMÁNEK, Jaroslav. *Družicové komunikační systémy*. 2007, [cit. 2011-11-1]. Dostupný z WWW: <http://www.urel.feec.vutbr.cz/web_pages/projekty/clanky /Rumanek_druzice.pdf>.

[25] RICHTR, T. *Oběžné dráhy družic.* [online]. 2002 [cit. 2011-10-31]. Dostupný z WWW: <http://tomas.richtr.cz/mobil/sateo.htm>.

[26] *NetDisturb User Guide.* 2010, [cit. 2011-11-5]. Dostupný z WWW: <http://www.zti-telecom.com/User_GuidesN/NetDisturb_User_Guide_V4.9.pdf>.

[27] SCHMID, Stefan. *QoS based Real-Time Audio Streaming in the Internet*. 1999, [cit. 2011-11-6]. Dostupný z WWW <http://sjschmid.de/publications/Diplom-Thesis.pdf>.

[28] GOLEMBIOVSKÁ, A. *Přenosové cesty*. 2011, [cit. 2011-11-8]. Dostupný z WWW: <http://spseiostrava.cz/golembiovska/ict/texty/22_prenosova_media.pdf>.

[29] *Srovnání UTP a STP kabeláží.* [online]. 2009 [cit. 2011-12-04]. Dostupný z WWW: <http://www.kassex.cz/knihovna/clanky/srovnani-utp-a-stp-kabelazi>.

[30] UBIK, Sven. *QoS a diffserv - Úvod do problematiky*. 2000, [cit. 2012-02-17]. Dostupný z WWW: <http://www.cesnet.cz/doc/techzpravy/2000-6/diffserv.pdf >.

[31] MOLNÁR, Karol. *Mechanismus diferencovaných služeb*. c2009. Dostupný z WWW: <www.utko.feec.vutbr.cz/ molnar/>. s. 28.

[32] *Transmission Control Protocol (TCP) Parameters*. 2011, [cit. 2012-02-24]. Dostupný z WWW: <http://www.iana.org/assignments/tcp-parameters/tcp-parameters.xml>.

[33] DOSTÁLEK, L.; KABELOVÁ, A. *Velký průvodce protokoly TCP/IP a systémem DNS*. 5. aktualiz. vyd. [s.l.] : Computer Press, 2008. 488 s. ISBN 978-80-251-2236-5.

[34] JEŘÁBEK, Jan. *Pokročilé komunikační techniky*. 2011, [cit. 2012-02-26], 244s.

[35] ROHÁČ, M. *Optimalizace protokolu TCP*. 2006, [cit. 2012-03-04]. Dostupný z WWW: <http://www.cs.vsb.cz/grygarek/TPS/projekty/0506Z/roh035-TCP-Optimization.pdf>.

[36] MODLITBA, Jan. *Vývojové trendy protokolu TCP pro vysokorychlostní sítě*. [online]. Brno, 2008 [cit. 2012-04-04]. Diplomová práce. Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Josef Vyoral.

[37] BRAKMO, Lawrence; O'MALLEY, Sean; PETERSON, Larry. *TCP Vegas: New Techniques for Congestion Detection and Avoidance.* [cit. 2012-04-06]. Dostupný z WWW: <http://www.cs.umd.edu/class/spring2010/cmsc711/vegas.pdf >**.**

[38] QURESHI, B.; OTHMAN, M.; HAMID, N. A. W. *Progress in Various TCP Variants*. 2009, [cit. 2012-03-31]. Dostupný z WWW: <http://www.ieeexplore.ieee.org>.

[39] CAINI, C.; FIRRINCELI, R. *TCP Hybla: a TCP enhancement for heterogeneous network.* 2004, [cit. 2012-04-07]. Dostupný z WWW: <http://web.eecs.utk.edu/~dunigan/ipp05/hybla.pdf>.

[40] MARCONDES, C.; MATTHEWS, J.; CHEN, R.; SANADIDI, M. Y.; GERLA, M. *A Cross-Comparison of Advanced TCP Protocols in High Speed and Satellite Environments.* [cit. 2012-04-07]. Dostupný z WWW: <http://www.ieeexplore.ieee.org>.

[41] CHENG, P. F. *TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks*. 2003, [cit. 2012-04-09]. Dostupný z WWW: <http://wwwbk.ie.cuhk.edu.hk/fileadmin/staff_upload/soung/Journal/J3.pdf>.

[42] LI, Y.; LEITH, D; SHORTEN, R. N. *Experimental Evaluation of TCP Protocols for High-Speed Networks* 2007, [cit. 2012-04-15]. Dostupný z WWW: <http://www.ieeexplore.ieee.org>.

[43] LEITH, D.;SHORTEN, R. *H -TCP: TCP for high-speed and long-distance networks.* [cit. 2012-04-15]. Dostupný z WWW: <http://jauu.net/var/print-queue/htcp3.pdf>.

[44] MOLNÁR, Karol. *TCP bez korekce*. c2009. Dostupný z WWW: <www.utko.feec.vutbr.cz/ molnar/>. s. 27.

[45] SOUZA, E; AGARWAL, D. *A HighSpeed TCP Study: Characteristics and Deployment Issues.* [cit. 2012-05-02]. Dostupný z WWW: < http://www.citeseerx.ist.psu.edu>.

[46] ANGAJALA, R. S. *Different Protocols for High Speed Networks.* 2012, [cit. 2012-05-02]. Dostupný z WWW: <http://www.ieeexplore.ieee.org>.

[47] RHEE, I.; XU, L. *CUBIC: A New TCP-Friendly High-Speed TCP Variant.* [cit. 2012-05-04]. Dostupný z WWW: < http://www4.ncsu.edu/~rhee/export/bitcp/cubic-paper.pdf>.

[48] SCHURMAN, K. *BIC-TCP* [online]. 2008 [cit. 2012-05-05]. Dostupný z WWW: <http://www.computerpoweruser.com/articles/archive/c0408/26c08/26c08.pdf?guid=>.

[49] CHEN, D. M. *Overview of TCP structure and operations*. [online]. 2009 [cit. 2012-05-16]. Dostupný z WWW: <https://dbsgrad.sfsu.edu/~mmurphy/dachen/documents/Overview.htm>.

PŘÍLOHY

LABORATORNÍ ÚLOHA č.1 - Měření a testování kvality služeb pro různé druhy provozu

Cíl

Cílem laboratorní úlohy je proměřit, ověřit si, jak parametry typu zpoždění, jitter a ztrátovost ovlivňují kvalitu služeb pro různé druhy provozu. Mezi sledované provozy budou patřit streamované video, streamované audio, data, telefonní hovor mezi studenty a videkonverzace. Jednotlivé parametry budou nastavovány v aplikaci NetDisturb. Absolvováním této laboratorní úlohy by měli studenti získat znalosti, při jakém zpoždění, jitteru a ztrátovosti dochází k ovlivňování kvality různých síťových služeb.

Vybavení pracoviště

HW vybavení: PC 1, CPU 2 GHz, 1 GB RAM, 1 GB volného místa na disku
PC 2, CPU 2 GHz, 1 GB RAM, 1 GB volného místa na disku
SW vybavení: PC1, PC2 oba operační systém Windows XP, FTP server FileZilla, FTP klient Total Commander, VLC media player, NetMeeting.

Úkoly

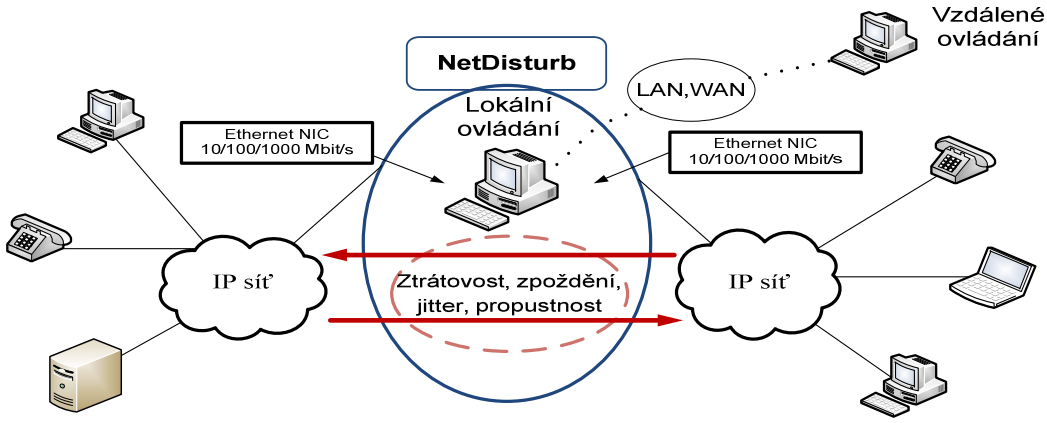
1. Seznamte se s pracovištěm a zkontrolujte, zda obsahuje výše uvedenou HW i SW výbavu.
2. Pomocí zástupce na ploše spustíte software NetDisturb Client.
3. Vytvořte účet na FTP serveru FileZilla a FTP klientu Total Commander.
4. Vytvořte filtry pro streamované video, audio, data a telefonní hovor.
5. Vytvořte streamování pomocí VLC přehrávače, nastavte zpoždění, ztrátovost uvedená v teoretickém úvodu a ovlivňujte provoz.
6. Vytvořte telefonní hovor po IP síti, nastavte zpoždění, jitter, ztrátovost a ověřte si uvedené hodnoty v teoretickém úvodu, jestli ovlivňují telefonní hovor.
7. Po provedení všech úkolu smažte vytvořený FTP účet klienta a účet na FTP serveru, ukončete vzdálené připojení na jednotlivých počítačích a vypněte Netdisturb, vypněte počítače a odejděte.

Teoretický úvod

Emulátor sítí NetDisturb

NetDisturb je aplikace, která dokáže emulovat reálné vlastnosti datové sítě, tedy umí zhoršit procházející provoz přes IP sítě (IPv4 a IPv6). Zhoršením je zde myšleno nastavením různého zpoždění, latence, jitteru, omezení šířky pásma, ztrátovosti paketu, kopírování a různé změny nastavení v paketech. Hlavním účelem aplikace NetDisturb je testování různých komunikačních služeb a sledovat jejich chování při zhoršených podmínkách v síti. NetDisturb slouží jako most mezi dvěma ethernetovými segmenty a umožňuje buď jednosměrný, nebo obousměrný přenos paketů na síťových kartách typů Ethernet, Fast Ethernet a Gigabit Ethernet. Jeho součástí je také uživatelské rozhraní pro server a klienta. Obě tato rozhraní mohou být spouštěna nejen na jednom počítači (NetDisturb Server a NetDisturb Client), ale i na dvou počítačích, přičemž u druhé varianty na jednou počítači poběží serverová část

(NetDisturb Server) a na druhém klientská část (NetDisturb klient) viz Obrázek A. 1. Uživatel na klientské části pak musí vytvořit spojení se serverem pomocí jeho IP adresy a portu, na kterém budou spolu komunikovat, po ověření těchto údajů, je spuštěno uživatelské rozhraní pro klienta.



Obrázek A. 1: Princip aplikace NetDisturb

Kvalita služeb – QoS

Hlavním úkolem podpory QoS je, aby se zabránilo zahlcení komunikačních spojů a přepojovacích prvků daty a aby se každé realizované službě zajistil dostatek prostředků podél trasy, kudy probíhá přenos informace. Technologie podpory QoS je schopna rozlišovat jednotlivé přenosy, jejich typy a každému typu nastavit jinou kvalitu. Zjednodušeně lze říct, že jakýkoliv provoz se bude snažit doručit včas a bez žádných problémů. Pomáhá si přidělováním různých priorit ke každému provozu. Nejprve je provoz klasifikován a z klasifikace potom plyne priorita. Platí, že vyšší hodnota priority má přednost před nižší hodnotou priority a podle toho se pak zachází s provozem. Mezi jeho další používané funkce patří omezování přenosového pásma (nastaví se maximální přenosové pásmo pro využití) a vyhrazení přenosového pásma (nastaví se minimální přenosové pásmo, které bude provoz využívat).

Přenos paketů je ovlivňován zpožděním, kolísáním zpožděním tzv. jitter, ztrátovostí paketů, šířkou pásma a doručení paketů mimo dané pořadí. Zpoždění je čas, kdy požadovaná data nejsou doručena okamžitě, ale s menší časovou prodlevou. Na tohle zpoždění má vliv zpracování dat v koncovém zařízení, v odchozích frontách, přes dobu zpracování informace v mezilehlých zařízeních, spolu se serializačním zpožděním a také použité přenosové médium. Kolísání zpoždění je rozdíl v intervalech mezi přijímanými pakety. Tento jitter používá vyrovnávací pamět tzv. jitter buffer, který vyrovnává tyto rozdíly v komunikaci. Procento paketů, které jsou ztraceny při průchodu v komunikační síti, nazýváme ztrátovost paketu. Šířka pásma definuje, jaký objem dat můžeme odeslat přes síť v daném čase, je udávána v bitech za sekundu (b/s).

Hlavním úkolem v laboratorní úloze bude ovlivňovat (zhoršovat) kvalitu služeb pro streamované video, streamované audio, data, telefonní hovor po IP síti.

Streamované video

Streamovaná videa jsou méně náročnější na přenos než videopřenosy, protože se jedná o

jednosměrný přenos. Tato služba není tolik ovlivněna zpožděním, a proto může příjemce využít větší vyrovnávací paměť, která dokáže odstranit vyšší hodnotu kolísání zpoždění u přicházejících paketů. Mezi důležité parametry tohoto provozu patří, že ztrátovost paketu je do 5 %, zpoždění se pohybuje mezi 4-5 s a streamování videa je většinou jednosměrné tzn., síťové prostředky jsou využívány asymetricky.

Streamované audio

Na streamované audio neboli hudba a mluvené slovo má také vliv zpoždění. Lidé dokážou lépe vnímat chybný přenos zvuku než špatně přenášené video, proto QoS má velmi přísné požadavky na ztrátovost, zpoždění a jitter pro zvuk. Požadavky na šířku pásma jsou malé. Pokud, bude ztrátovost do 1 % a nedochází k žádným změnám, do 13 % pořád rozumíme mluvenému slovu, ale už slyšíme i „praskání“, do 20 % větám rozumíme, jejich obsah si můžeme částečně domyslet, ale „praskaní“ se zvětšuje, pokud je ztrátovost paketů nad 25 % srozumitelnost je na nízké úrovni, věty budou méně nebo vůbec pochopitelné, „praskání“ bude intenzivnější a bude mnohem lepší přenos ukončit. Zpoždění jednotlivých paketů by se mělo pohybovat od 300 do 800 ms. Kolísání zpoždění je ovlivněno vyrovnávací paměti u každého příjemce. Poznáme ho tak, že příjemce musí počkat určitou dobu, než budou načteny pakety do vyrovnávací paměti. Je to dáno kvůli tomu, že některé pakety dojdou později. U předčasného přehrávání by příjemce slyšel mezery v signálu a nedalo by se to poslouchat. Po načtení všech paketů do vyrovnávací paměti bude signál přehrán bez mezer.

Data

U přenosu dat je kladen hlavně důraz na to, aby dat byla stažená správně a bez chyb, pokud dojde k chybě je přenos opakován.

Služba VoIP

Služba VoIP (telefonní hovor) používá transportní protokol UDP (User Datagram Protocol) pro přenos hlasové informace spojený s protokolem RTP (Real-Time Protocol) a dohled nad přenosem (informace o přenosu) provádí protokol RTCP (Real-Time Transport Control Protocol). Signalizace je přenášena pomocí signalizačních protokolů například H.323, SIP (Session Initiation Protocol), MGCP (Media Gateway Control Protocol) a obsahuje informaci o navázání, ukončení a případné změny v sestavované relaci. Podle normy ITU-T G.114 jsou definovány časy, které určují kvalitu hovoru. Jestliže celkové zpoždění dosáhne hodnoty pod 150 ms, je tento hovor velmi kvalitní a při zpoždění pod 20 ms nedochází k ozvěně v hovoru. Mezi 150-400 ms považujeme tento hovor za dobrý (přijatelný). Pokud hodnota přesáhne 400 ms, srozumitelnost mezi komunikujícími je špatná a může mezi nimi dojít i ke ztrátě synchronizace.

Ztrátovost paketů u VoIP je nežádoucí, tak jako poslání paketu znova nebo snížení rychlosti. Ve výsledku může dojít ke ztrátě (výpadku) hovoru a neuslyšíme zprávu od volaného. Pro názornost jsou zde uvedeny procentuální ztrátovosti, které ovlivňují kvalitu hovoru. Ztráty paketu do 2 % nemají vliv na kvalitu hovoru. Mezi 2-5 % pomalu dochází k nesrozumitelnosti hovoru, které ovlivňuje typ kodeku a jeho implementované vlastnosti pro maskování ztrát. U 5-10 % je srozumitelnost nízká a znehodnocuje se kvalita hlasu. Pokud ztrátovost překročí 10 %, kvalita hovoru bude velice nízká, srozumitelnost minimální a nepomůžou ani kvalitní kodeky proti odolnosti vůči ztrátám.

Třídy QoS

Výše byly popsány síťové služby podle jejich nároků na šířku pásma, ztrátovosti paketu, zpoždění a jitter. Pro lepší přehlednost a orientaci jsou jednotlivé služby rozděleny do čtyř odlišných tříd. Patří sem konverzační třída, třída proudového přenosu dat, interaktivní třída a třída služeb na pozadí.

V **konverzační třídě** jsou provozovány služby, které pracují v reálném čase například telefonie VoIP, videokonverzace atd. Tato třída velmi přísně kontroluje zpoždění a jitter, musí být co nejmenší, a proto je řazena do třídy s nejvyšší prioritou. Komunikace je typu klient-klient.

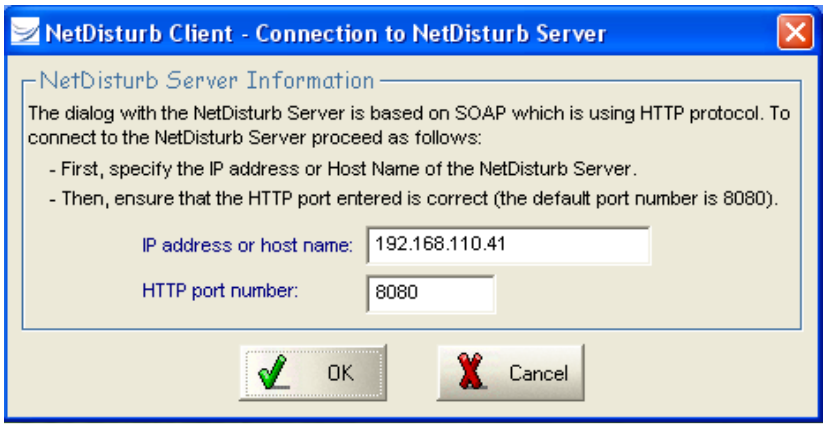
Třída proudového přenosu dat neboli streaming třída slouží k provozování audiovizuálního materiálu mezi koncovým uživatelem a zdrojem například televizního vysílání, různá internetová rádia nebo stahování videoklipů a hudby. Probíhá zde komunikace typu klient-server. Třída je citlivá na kolísání zpoždění než na zpoždění, protože na straně příjemce se nachází vyrovnávací paměť, která má omezenou velikost a například u videa jde hlavně o plynulost, teprve až potom o kvalitu obrazu.

Interaktivní třída pracuje na principu žádosti a odpovědi, dále musí zajistit během trvání komunikace, že celý obsah dat bude doručen správně a s minimální chybovostí. Zpoždění se pohybuje řádově v jednotkách sekund. V této třídě najdeme služby pro prohlížení internetových stránek, stahování e-mailových zpráv, správu nebo přístup k různým serverům a databázím.

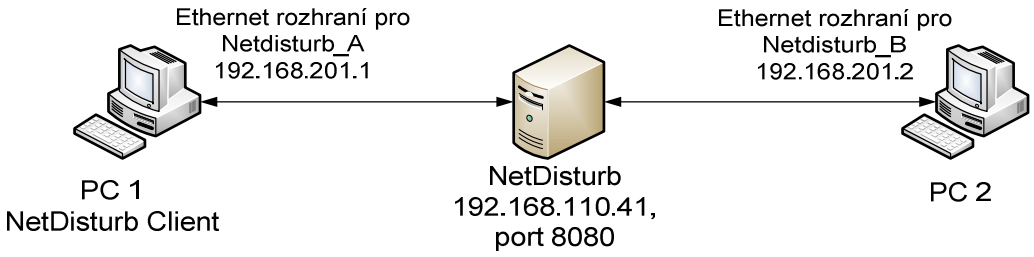
Poslední třídou je **třída služeb probíhající na pozadí**, do které patří například služby pro stahování a přenos dat (FTP), otevírání webových stránek atd. Tato třída není citlivá na zpoždění ani kolísání zpoždění, ale očekává se bezchybné doručení dat. Proto mají tyto služby nejnižší prioritu a nejmenší požadavky na šířku pásma.

Postup

Po spuštění obou počítačů na pracovišti, zjistěte na kterém je umístěna aplikace NetDisturb Client (dále jen klient). Spusťte klienta a do zobrazeného dialogového okna *NetDisturb Client – Connection to NetDisturb Server* zadejte IP adresu *192.168.110.41* a port *8080* pro komunikaci se serverem viz Obrázek A. 2, zobrazí se vám uživatelské rozhraní klienta, zatím na klientu nic nenastavujte. Dále podle schématu zapojení přidělte stanicím IP adresy, viz Obrázek A. 3. Přihlašte se na PC jako uživatel **Student** a heslo **.Student427-**.



Obrázek A. 2: Připojení NetDisturb klienta na server



Obrázek A. 3: Schéma zapojení laboratorní úlohy

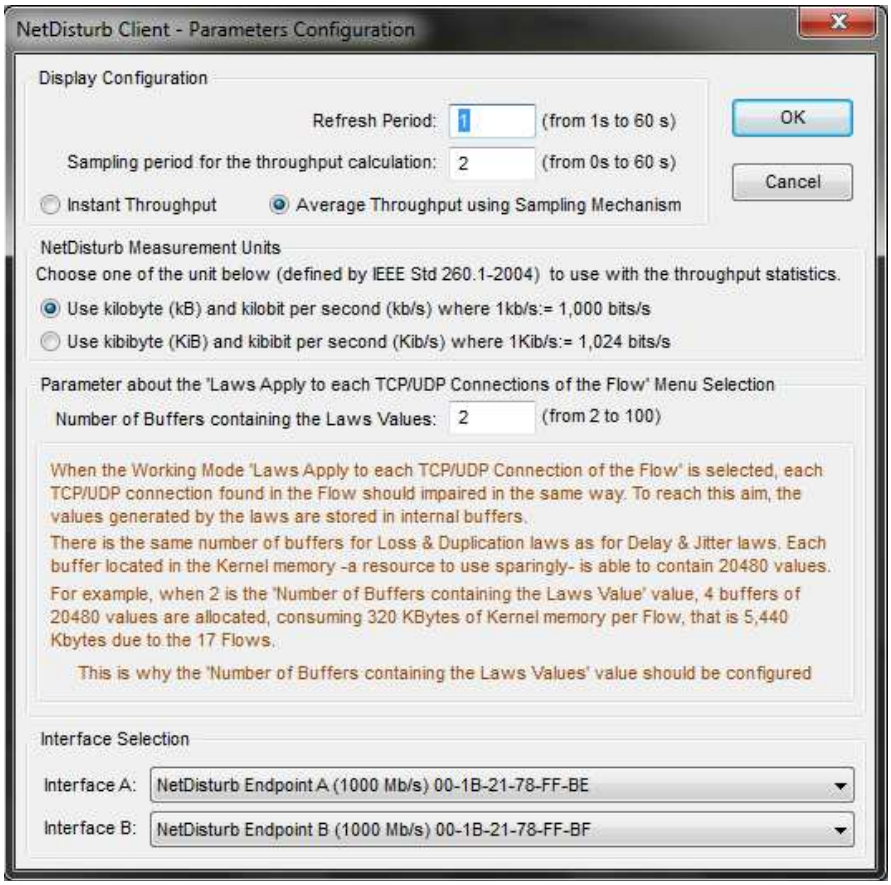
Nastavení FTP serveru a klienta

Pomocí FileZilla serveru na PC2 vytvořte účet pro připojení FTP klienta. Spusťte zástupce *FileZilla Server Interface* umístěného na ploše, ponechejte IP adresu 127.0.0.1 a číslo portu 14147. Klikněte na třetí ikonku zleva  a zobrazí se dialogové okno *FileZilla Server Options*. Dále v nabídce **General Settings** proveďte nastavení portu na kterém má čekat server příchozí spojení, port bude *21* v položce *Listen on these* ports. **Welcome message** nastavte na **%v**. V poli **IP bindings** smažte hvězdičku * a nastavte IP adresu *192.168.201.2/24*, na této IP adrese bude FTP server naslouchat. Zbylé položky si prohlédněte a nechte je ve výchozím nastavení a potom klikněte na *OK*. Nyní proveďte nastavení uživatelského účtu pomocí ikonky  . Nejprve zkontrolujte a vymažte předchozí vytvořený účet, provedete to tak, v položce **General** v menu napravo kliknete na vytvořený účet a stiskněte na tlačítko **Remove**. Vytvoření uživatelského účtu provedete klinutím na tlačítko **Add**. Do prvního pole vložíte název účtu *PC1* a u druhého pole můžete přiřadit vytvořený účet do určité skupiny, pole nechte ve stavu *none*. Potom zaškrtněte pole **Password** a napište heslo *qos*. V položce **Shared folders** nastavte cestu do složky *FTP_server* umístěnou na ploše a zaškrtněte práva Read, Write a Delete. Zkontrolujte obsah adresáře, pokud je prázdný informujte o tom učitele. Zbylé položky ponechejte beze změny a stiskněte tlačítko OK.

Na PC1 pomocí Total Commanderu vytvořte uživatelský účet pro FTP klienta. Otevře se dialogové okno *Připojení k serveru FTP*, potom kliknutím na **Nové připojení** vytvořte uživatelský účet pro připojení k serveru FTP. Parametry jsou následující: **Název relace** pojmenujte například ftp, **Hostitel** zadejte IP adresu FTP serveru tedy *192.168.201.2*, **Jméno uživatele:** *PC1*, **Heslo:** *qos*, klikněte na OK. Připojení vytvořeného účtu provedete kliknutím na tlačítko *Připojit*.

Nastavení klienta NetDisturb pro různé druhy provozu

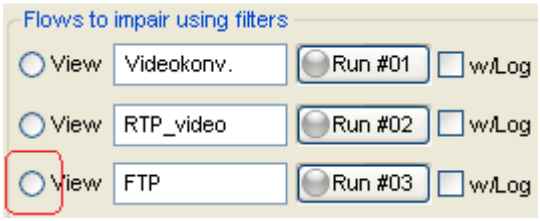
Zobrazte si uživatelské rozhraní klienta, odklikněte informační hlášení *There is no interface selected*, potom otevřete soubor Default.WSX pomocí *File – Open – Default.WSX*, kde jsou předvytvořené konfigurace filtrů a uložte si ho například pod názvem *provozy* (pojmenování souboru je zcela libovolné na funkčnost nemá vliv). Uprostřed klienta je tlačítko *Configure NIC*, klikněte na něj a zobrazí se okno *NetDisturb Client – Parameters Configuration* a proveďte nastavení podle viz Obrázek A. 4.



Obrázek A. 4: Nastavení síťových rozhraní

Po nastavení síťových rozhraní, proveďte konfiguraci filtrů pro FTP provoz, streamované video, audio a telefonní hovor.

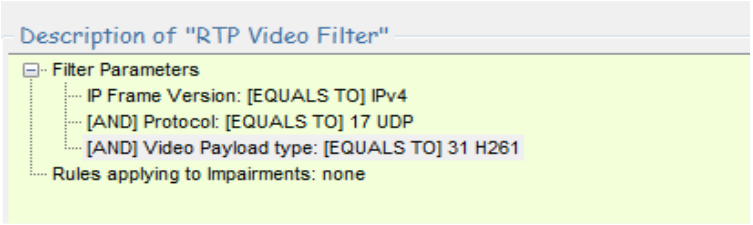
Nejprve si pojmenujte jednotlivé provozy, provedete to vepsáním do editačních oken, viz Obrázek A. 5. Vedle každého pojmenovaného provozu jsou umístěny přepínací kolečka s názvem *View* viz Obrázek A. 5 a kliknutím na jednotlivá *View* budete provádět nastavení parametrů pro jednotlivé provozy.



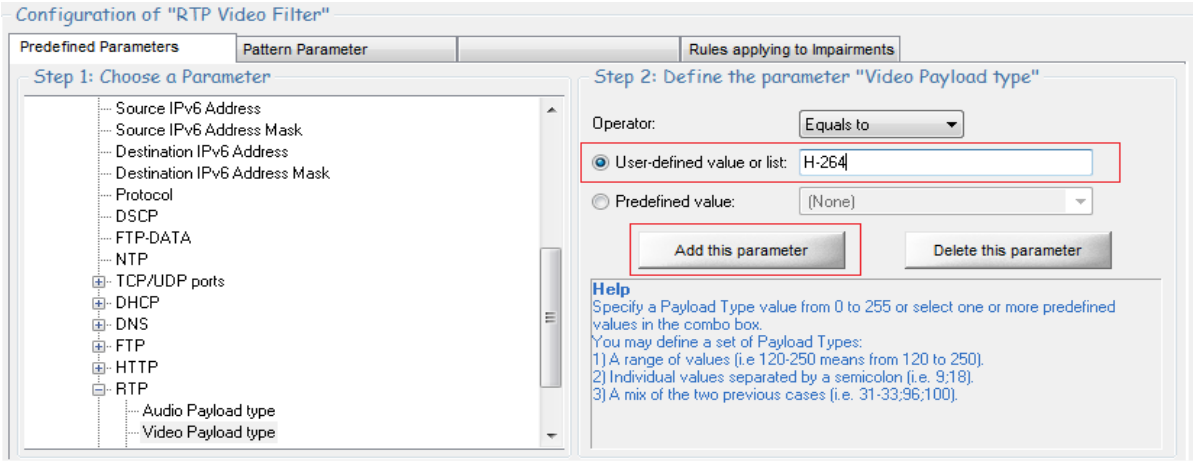
Obrázek A. 5: Pojmenování provozů

Nejdříve klikněte na *View* u editačního okna s názvem FTP a proveďte nastavení tohoto provozu, viz Obrázek A. 5. U **A interface** klikněte na *Configure Filter* a z nabídek filtrů v levé části otevřeného okna vyberte *FTP Filter* nebo nad **A interface** je napsáno *No Filter Selected*, je to rozevírací nabídka a pomocí ní taky můžete vybrat *FTP Filter*. U **B *interface***, vyberte také *FTP Filter*, tím je konfigurace FTP provozu ukončena, konfigurace ostatního provozu je podobná, ale s odlišnými filtry.

Dále nastavte provoz pro streamované video. Vyberte filtr *RTP Video Filter*, po vybrání filtru bude zobrazeno varovné hlášení, že musí být filtr nastaven i u **B *interface***, musí pracovat v páru, stiskněte OK a filtr bude nastaven i u druhého rozhraní. Dále stiskněte *Configure Filter* a zobrazí se nastavení filtru v okně s názvem *Description of RTP Video Filter* viz Obrázek A. 6. Proveďte menší změnu nastavení v kodeku u *Video Payload Type*, aby NetDisturb streamované video pak propustil. Kodek nastavíte kliknutím na řádek *Video Payload Type* (musíte kliknout dvakrát), v záložce *Predefined Parameters* je automaticky zobrazeno co potřebujete změnit, najdete to u *Step 2: Define the parametr Video Payload Type* v části *User-defined value or list*, kde do editačního okna napište **H-264** a stiskněte *Add this parameter*, kodek bude změněn, nastavení proveďte podle viz Obrázek A. 7. Kodek H-264 podporuje VLC přehrávač. Konfigurace streamovaného videa je hotová a stiskněte OK.



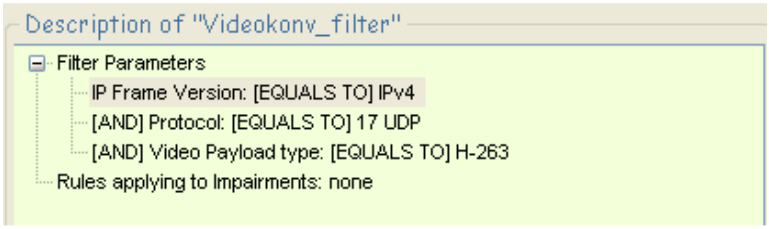
Obrázek A. 6: Výchozí nastavení filtru RTP Video Filter



Obrázek A. 7: Nastavení kodeku pro streamované video

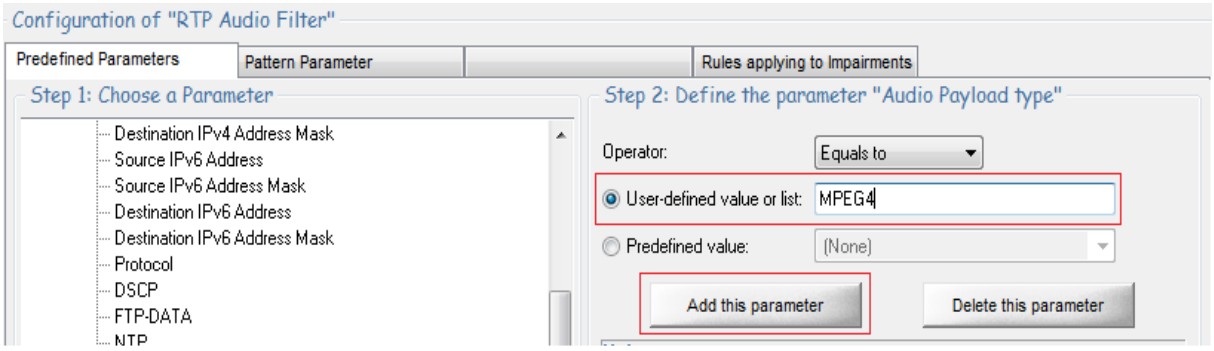
Streamované audio nastavíte tak, že vyberete filtr *RTP Audio Filter* a potom nastavíte kodek u *Audio Payload Type* na **MPEG4** viz Obrázek A. 9. Kodek nastavíte podobně jak u streamovaného videa.

Pro filtrování videokonverzace vytvořte nový filtr s názvem *Videokonv_filter*. Provedete to následovně. Klikněte na *View* u okna *Videokonv*. a potom na *Configure filter*. V otevřeném okně klikněte na tlačítko *New Filter* (nachází se v horní části okna na levé straně) a v otevřeném dialogovém okně *NetDisturb Client – New Filter* napište *Videokon_filter*. Nyní do vytvořeného filtru nastavte následující parametry, viz Obrázek A. 8.



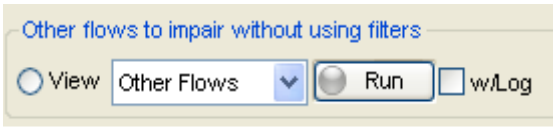
Obrázek A. 8: Nastavení filtru pro videokonverzaci

Filtrování hovorové komunikace vytvoříte podobně jako u videkonverzace, viz Obrázek A. 8. Vytvořte nový filtr s názvem *Hovor_filter*. Do vytvořeného filtru nastavíte tyto položky a to *IP Frame Version: IPv4*, *Protocol: 17 UDP* a *Audio Payload Type: G.711; G.723.1*.



Obrázek A. 9: Nastavení kodeku pro streamované audio

Pokud máte vytvořené filtry pro různé druhy provozu, spustíte je stisknutím tlačítka *Run* a tlačítkem *Stop* zastavíte. Než budete jednotlivé toky ovlivňovat, vyzkoušejte, jestli počítače a jejich síťová rozhraní spolu komunikují, využijte příkazovou řádku a funkci **ping** (použijte IP adresy 192.168.201.1 nebo 192.168.201.2). Odezva by neměla být úspěšná. Pro úspěšnou odezvu spusťte provoz, který nebude ovlivňován žádným filtrem (bude propuštěno všechno) viz Obrázek A. 10 a znovu vyzkoušejte komunikaci mezi počítači, teď musí být úspěšná. Ponechejte provoz, který není ovlivňován filtrem spuštěn během měření.

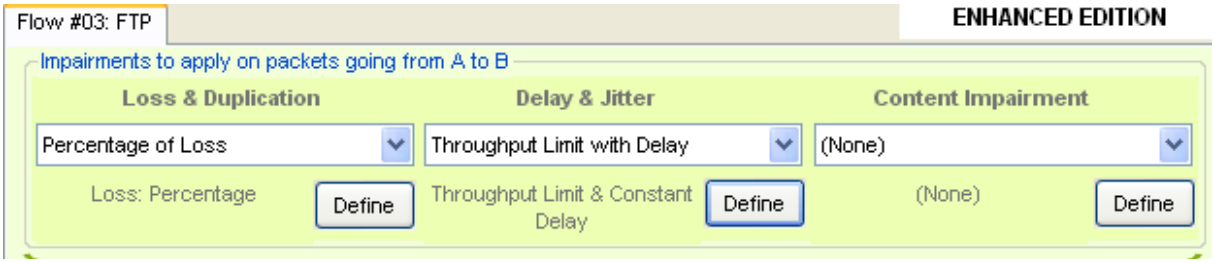


Obrázek A. 10: Puštění provozu bez použití filtrů

Spuštění FTP provozu pro přenos video a audio souboru

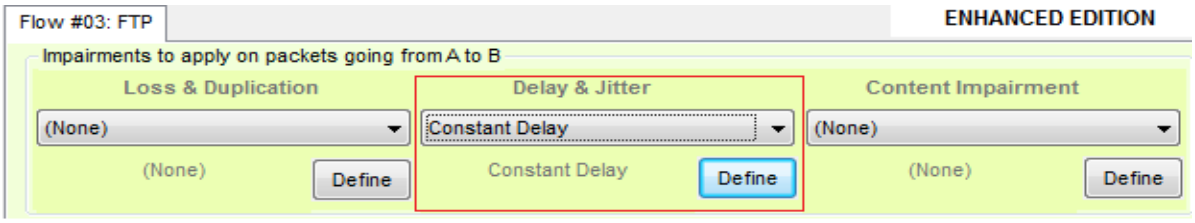
Na klientu NetDisturb spusťte filtr pro FTP přenos. Proveďte přenos video souboru z PC1 do PC2 pomocí FTP přenosu. Aby přenos nebyl ideální (žádná ztrátovost a zpoždění), nastavte nejprve ztrátovost, potom zpoždění a nakonec kombinaci obou. Během nastavování různých parametrů pro ovlivňování provozu musíte všechny spuštěné provozy zastavit, jinak změny neprovedete a potom opětovně spustit. Tento postup budete provádět i u jiných provozů. Nezapomeňte po každé změně parametrů u každého provozu spustit i provoz, který není ovlivňován filtrem, viz Obrázek A. 10. Nastavení pro ztrátovost provedete následovně, kliknutím na rozevírací nabídku s názvem *Loss & Duplication*, vyberte *Percentage of Loss*

viz Obrázek A. 11, tlačítkem *Define* provedete nastavení ztrátovosti a to v záložce *Law Parameters* v okně s názvem **Percentage**, hodnotu si zvolte libovolně například 10 %, zbylé položky neměňte. Potom omezte šířku pásma pro přenos, provedete to kliknutím na rozevírací nabídku *Delay & Jitter*,vyberte *Throughput Limit with Delay* a tlačítkem *Define* nastavte propustnost na 1 Mb/s a zbylé položky neměňte. Stiskněte *OK*, spusťte FTP provoz a přeneste znovu video soubor. Sledujte změny během přenosu, jak v Total Comanderu tak i v aplikaci NetDisturb.



Obrázek A. 11: Nastavení ztrátovosti a propustnosti

Kliknutím na *View Per-Flow Statistics*, zobrazíte statistiku kolik paketů, bylo přijato, nastavenou ztrátovost, propustnost, kolik paketů bylo odesláno, zpoždění a také jestli filtrování pracuje nebo ne, sledujte cestu z **A do B**. Pro zrušení parametrů, který ovlivňuje provoz, stačí z rozevírací nabídky nastavit položku **None**, udělejte. Dále proveďte nastavení i pro zpoždění, které nastavíte z rozevírací nabídky *Delay & Jitter*, vyberte *Constant Delay* viz Obrázek A. 12, klikněte na *Define* a v záložce *Law Parameters*, nastavte zpoždění například na 30 ms, potom si zvolte libovolně jinou hodnotu a zjistěte, zda došlo ke zlepšení nebo ke zhoršení přenosu. Nakonec proveďte ovlivnění provozu, jak se ztrátovostí tak i zpožděním dohromady a sledujte výsledek. Po vyzkoušení si přeneste soubory ještě jednou bez ovlivněného provozu a FTP filtr v klientu NetDisturb zastavte. Po skončení přenosu určete jaký vliv má ztrátovost a zpoždění na přenos těchto souboru.

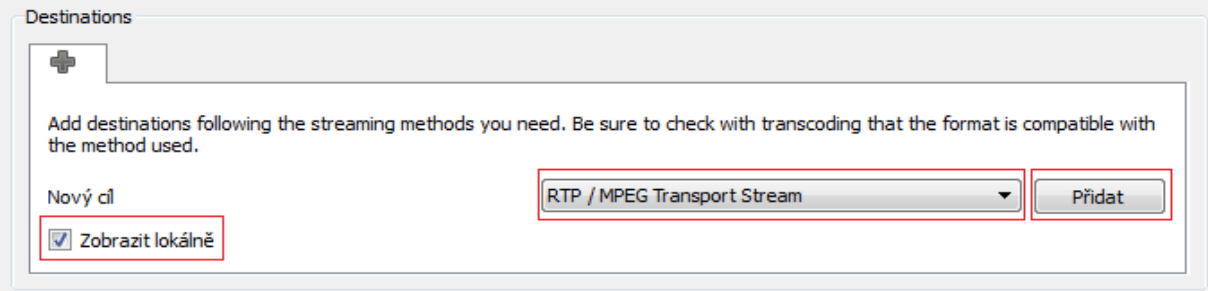


Obrázek A. 12: Nastavení zpoždění

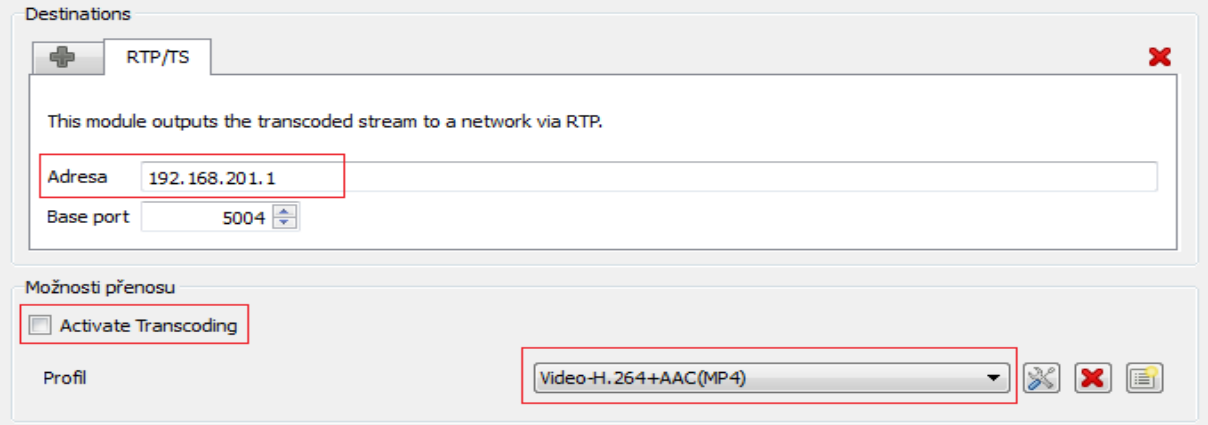
Nastavení VLC přehrávače pro streamování (proudové vysílání) videa a audia

Na PC2 klikněte na zástupce *VLC media player* umístěného na ploše a vytvořte proudové vysílání. Provedete to následovně, stiskněte nabídku *Média* a vyberete *Proudové vysílání* nebo použijte zkratku Ctrl+S. Na záložce **Soubor**, pomocí tlačítka *Přidat* vyberete film, který budete streamovat. Film je umístěn ve složce *FTP_přenos* na ploše, který jste stáhli z FTP serveru. Po přidání filmu klikněte na tlačítko *Pustit proudem*. V záložce *Zdroj* uvidíte, jaký film jste vybrali a jeho cestu k němu. Potom klikněte na *Další*, přejdete do záložky **Destinations**,vyberete, pomocí jakého protokolu bude probíhat streamování. Postup je následující zaškrtněte *Zobrazit lokálně*, napravo je rozevírací nabídka a z ní vyberte *RTP/MPEG Transport Stream* pak stiskněte *Přidat*, viz Obrázek A. 13, potom zadejte IP

adresu 192.168.201.1 (PC1) a port nechte beze změny, deaktivujte *Activate Transcoding* a **Profil** nastavte na *Video-H.264+AAC(MP4)* viz Obrázek A. 14. Nastavení pro streamování videa je ukončeno a stiskněte na tlačítko *Pustit proudem*. Potom na klientu NetDisturb spusťte vytvořených filtr s názvem **RTP_video**.



Obrázek A. 13: Nastavení protokolu pro streamování



Obrázek A. 14: Nastavení IP adresy a kodeku pro streamování

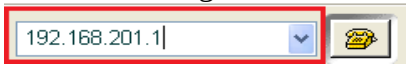
Na počítači PC1 spusťte VLC media player, je umístěn na ploše. V nabídce *Média* vyberte položku *Otevřít síťový proud*, do editačního okna zadejte **rtp://192.168.201.1:5004** a potom klikněte na *Přehrát*. Streamování během chvíle začne, pokud nezačne a zobrazí se VLC se nepodařilo otevřít MRL 'rtp://192.168.201.1:5004'. Zkontrolujte logy pro detaily. Vytvořte streamování znova a v části, kde zadáváte IP adresu, kam chcete vysílat, změňte port (*Base port*) například na 6000 viz Obrázek A. 14, totéž proveďte v případě neúspěchu i u streamovaného audia. Pokud streamování videa je funkční nastavte jednotlivá zpoždění, ztrátovosti a jitter podle tabulky viz **Chyba! Nenalezen zdroj odkazů.**. Pokud po změněném nastavení u ztrátovosti, zpoždění nebo jitteru nedojde ke změně u streamovaného videa nebo audia na PC1, klikněte ve VLC přehrávači na nabídku *Média*, potom položku *Otevřít síťový proud* a klikněte na *Přehrát*. Kolísání zpoždění (jitter) nastavíte ve stejné rozevírací nabídce jako propustnost u FTP filtru. Z nabídky vyberte *Uniform Jitter*, v záložce *Law Parameters* nastavte parametr **Alpha** na 1 ms, **Beta** na 20 ms a zpoždění na 20 ms. V intervalu od 1 do 20 budou náhodně vybírány hodnoty jitteru a použijte tento rozsah jak pro VoIP komunikaci i pro videokonverzaci. Dále proveďte nastavení **Alpha** na 20 ms a **Beta** na 30 ms, zpoždění změňte také na jinou hodnotu a sledujte co se děje s provozem. Nakonec nastavte **Alpha** na 30 ms a **Beta** na 40 ms.

Tab. 1: Kvalita QoS pro různé druhy provozu služeb

	Zpoždění [ms]			Ztrátovost [%]				Jitter [ms]
VoIP	0-150	150-400	> 400	< 2	2-5	5-10	> 10	< 20
Videopřenosy	< 150			< 1				< 30
Streamované video	4-5 s			< 5				ovlivněno vyrovnávací paměti
Streamované audio	300-800			< 1	1-13	13-25	> 25	ovlivněno vyrovnávací pamětí
Data	různé			různé				-

Podobně postupujte u streamování audia, ale v záložce *Destinations* vyberte z rozevírací nabídky *RTP Audio/Video Profile*, kodek bude stejný a u PC1 stačí znovu *Otevřít sítový proud*, ponechat **rtp://192.168.201.1:5004** a kliknout *Přehrát*. Jednotlivé ztrátovosti a zpoždění nastavte podle tabulky, viz Tab. 1.

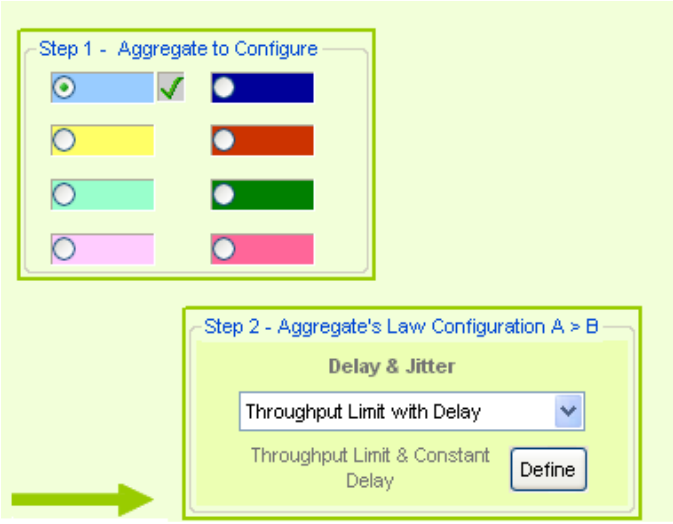
Videokonverzace a telefonní hovor

Telefonní komunikaci spolu s videokonverzací vytvoříte pomocí aplikace NetMeeting umístěnou na ploše. Na obou počítačích spusťte aplikaci NetMeeting a z PC2 zavolejte na PC1. Provedete to napsáním do editačního okna 

a pro zavolání klikněte na ikonku telefonu  . Na PC1 v pravém dolním rohu dojde k zobrazení okna s přijmutím nebo odmítnutím hovoru. Ikonkou  hovor ukončíte. Videokonverzaci spustíte kliknutím na ikonku  a opětovným kliknutím ji zase vypnete. Podle tabulky, viz Tab. 1 nastavte jednotlivé parametry ztrátovosti, zpoždění a jitteru.

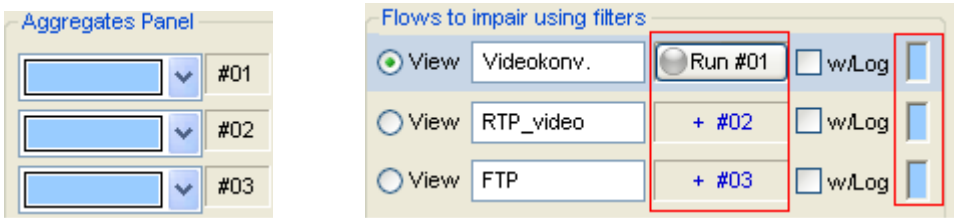
Sloučení provozů do jednoho toku (agregace)

Posledním úkolem bude sloučit vytvořené provozy do jednoho a ověřit si kvalitu služby QoS. Zastavte všechny spuštěné provozy a klikněte na *Configure Aggregates*. Z nabídky barev, pro odlišení jednotlivých agregátů pro větší počet provozů vyberte například modrou. Dále z rozevírací nabídky vyberte *Throughput Limit with Delay* viz Obrázek A. 15 a nastavte šířku pásma na 1 Mb/s, zbylé položky neměnte.



Obrázek A. 15: Vytvoření agregátu

Po nastavení klikněte na *OK* a proveďte přiřazení vytvořeného agregátu k jednotlivým provozům, stačí pro první tři. V pravé části aplikace NetDisturb je sloupec s rozevíracími nabídkami s nápisem *(None)*, ty umožňují přiřadit vytvořený agregát k vytvořeným provozům. Proveďte přirazení, viz Obrázek A. 16, přiřazení poznáte změnou tlačítek Run na jedno tlačítko Run a modrým označením (tvar obdélníka) vedle pojmenovaných provozů, viz Obrázek A. 16. Spusťte sloučený provoz, a sledujte, co se s ním stane. Nejvyšší prioritu bude mít videokonverzace, potom RTP_video a nakonec FTP provoz. Výsledek uvidíte kliknutím na *View Per Flow Statistic*. Libovolně si změňte hodnoty u agregátu a přidejte k němu ještě například ztrátovost 10 %, pozorujte změny.



a) b)

Obrázek A. 16: a) přiřazení agregátu b) změna tlačítek Run

Po sestavení laboratorní úlohy a zodpovězení kontrolních otázek vyučujícímu, vraťte pracoviště do původního stavu.

Kontrolní otázky

1. Jaký je rozdíl při přenosu služeb po sítí s podporou QoS a bez podpory QoS?
2. Do jakých tříd rozdělujeme jednotlivé služby?
3. Proveďte vyhodnocení uvedených hodnot v Tab. 1 to znamená do, jaké míry ovlivňují kvalitu jednotlivých služeb.

Literatura

[1] MOLNÁR, Karol. *Teorie front*. Dostupný z WWW: <www.utko.feec.vutbr.cz/molnar/>.

[2] PUŽMANOVÁ, R.; VOZŇÁK, M. *Kvalita služby pro VoIP*. Dostupný z WWW: <http://www.cesnet.cz/sdruzeni/napsali-onas/2009/04/20090415_Professional_Computing.html>.

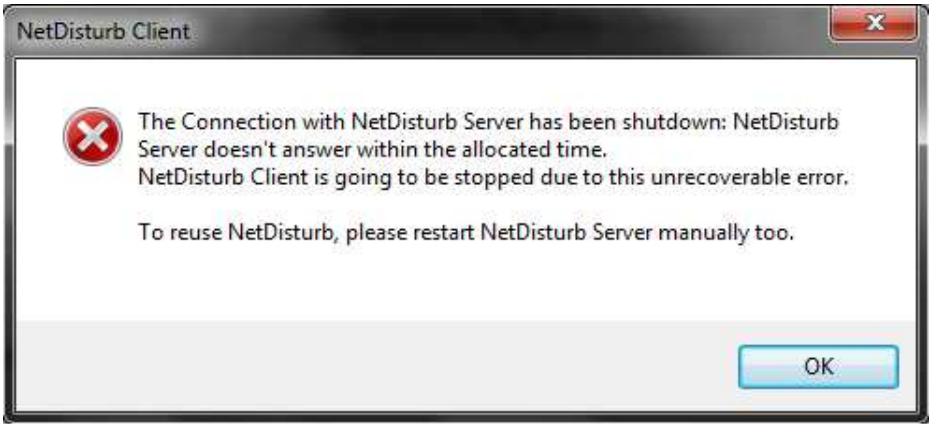
[3] SCHMID, Stefan. *QoS based Real-Time Audio Streaming in the Internet*. Dostupný z WWW <http://sjschmid.de/publications/Diplom-Thesis.pdf>.

[4] *NetDisturb User Guide*. 2010, 221s. Dostupný z WWW: <http://www.zti-telecom.com/User_GuidesN/NetDisturb_User_Guide_V4.9.pdf>.

Poznámky pro učitele

Pokud došlo na fakultě k výpadku proudu, neuvidí studenti v klientu NetDisturb síťová rozhraní pojmenované Netdisturb_A a NetDisturb_B (záložka *Actions*, položka *Configuration* a *Interface Selection*), je zapotřebí se připojit na server a opětovně je spustit.

Když studenti spustí klienta Netdisturb a po chvíli se jim zobrazí varovné hlášení, viz Obrázek A. 17, aplikace „shodila“ celý server a je zapotřebí ho celý restartovat a opětovně na serveru spustit serverovou část aplikace NetDisturb.



Obrázek A. 17: Varovné hlášení

LABORATORNÍ ÚLOHA č. 2 – Analýza TCP algoritmů

Cíl

Cílem laboratorní úlohy je seznámit se TCP algoritmy a provést jejich analýzu zhlediska propustnosti.

Vybavení pracoviště

HW požadavky: PC 1, CPU 1 GHz, 512 MB RAM, 1 GB volného místa na disku.
PC 2, CPU 1 GHz, 512 MB RAM, 1 GB volného místa na disku.
PC 3, CPU 1 GHz, 512 MB RAM, 2 GB volného místa na disku.
Přepínač NetGear ProSafe 8 Port Gigabit Smart Switch, model GS108T

SW požadavky: Operační systém Ubuntu, Proftpd (FTP server), FTP klient (Total Commander), NetDisturb, xplot.org, xpl2gpl, tcptrace, tcpdump, gnuplot.

Úkoly

1. Zapojte laboratorní úlohu podle schématu zapojení.
2. Vytvořte na počítači s operačním systémem Ubuntu FTP server.
3. Spusťte aplikaci klientskou aplikaci NetDisturb, ověřte funkčnost FTP serveru a nastavte podmínky přenosu.
4. Proveďte zachytávání provozů na PC s operačním systémem Ubuntu.
5. Změňte TCP algoritmus.
6. Zobrazte průběhy a proveďte vyhodnocení jednotlivých TCP algoritmů ze strany propustnosti.
7. Naměřené výsledky smažte, dejte pracoviště do původního stavu a odejděte.

Teoretický úvod

TCP protokol

Protokol TCP najdeme v síťovém modelu TCP/IP a vytváří virtuální dvoubodové duplexní spojení, kdy se před každým vysíláním datových jednotek sestaví spojení mezi vysílačem a přijímačem vznikne tak tzv. virtuální okruh. Pro vytvoření spojení mezi aplikací odesílatele a aplikací příjemce musí být jednoznačně určena IP adresa, číslo portu a použitý transportní protokol. Číslo portu může nabývat hodnot 0 až 65535. Na transportní vrstvě jsou přijatá data od aplikační vrstvy rozdělena na segmenty. Každý segment obsahuje TCP záhlaví a část rozdělených dat. Množství aplikačních dat, které může každý TCP segment přenést definuje parametr MSS (Maximum Segment Size).

Řízení toku dat protokolem TCP

Kontrola toku je realizována pomocí mechanismu posuvného okénka. Velikost okénka *owin* (outstanding window) na straně vysílače, definuje množství dat, které mohou být najednou odeslány, aniž by vysílač čekal na potvrzení od přijímače. Další data může vysílač odeslat, až dostane potvrzení o přijetí z některých částí odeslaných dat. Vysílač je průběžně informován přijímačem o jeho volném místě ve vyrovnávací paměti pomocí tzv. okénka *rwnd* (receive window) a říká mu, kolik dat ještě může odeslat, aniž by přijímač byl zahlcen. Jestli

hodnota rwnd bude rovna 0, je vyrovnávací paměť přijímače plná a vysílač musí čekat, až dostane nenulové *rwnd*. Tento způsob snižování *rwnd* nutí vysílač, aby snížil rychlost odesílání dat. Musí platit, že *owin* <= *rwnd*. Mechanismus posuvného okénka *owin*, je používán pro řízení provozu mezi koncovými uzly a také ovlivňuje propustnost, aby nedošlo k zahlcení. Maximální velikost okénka je určena 16 bitovým polem v záhlaví TCP, které se pohybuje v rozsahu od 0 do 65535 B (2^{16}-1). Tento rozsah by měl být pro většinu aplikací dostačující. Může se stát, že tato velikost okénka je malá. Řešením je použití TCP Window Scale (zvětšení okénka), které dovoluje zvětšit výchozí velikost okénka až na 1 GB. Tuto hodnotu získáme vynásobením maximální velikosti výchozího okénka (2^{16}-1) a hodnoty pro zvětšení okénka 2^{14}. Kde pro zvětšení okénka je využíváno rozsahu 0 až 14 bitů. Volbu zvětšení okénka nastavíme jen v úvodních segmentech s příznakem SYN při inicializaci spojení. Metoda zvětšení okénka nachází uplatnění v gigabitových sítích, protože jeho nastavením využijeme dostatečně přidělenou šířku pásma. Naopak použitím výchozí velikosti okénka by přidělena šířka pásma byla využita jen částečně.

Mechanismy předcházející zahlcení

Mechanismy předcházející zahlcení pracují na koncových stanicích a způsobí, že je zpomaleno TCP vysílání v době, když se síť nachází ve stavu zahlcení. Pro tento účel byly vytvořeny algoritmy, které dokážou efektivně řídit velikost posuvného okénka vysílače. Cílem jednotlivých algoritmů je, aby byla co nejlépe využita dostupná šířka pásma linky a zajistili spravedlivé sdílení linky s ostatními toky. Mezi tyto algoritmy patří:

- pomalý start (slow start),
- vyhýbání se zahlcení (congestion avoidance),
- rychlé přeposílání (fast retransmit),
- rychlé zotavení (fast recovery),
- omezené vysílání (limited trasmit).

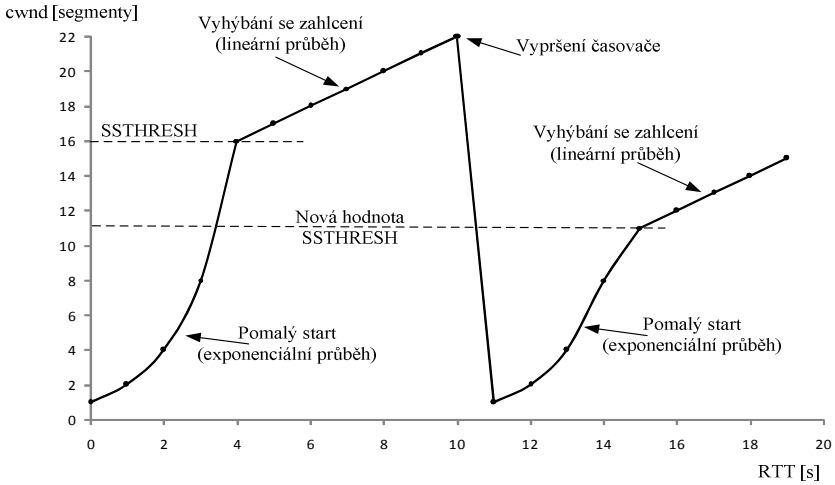
Pomalý start (slow start)

Po navázání TCP spojení nezačne vysílač vysílat plnou rychlostí na lince, protože posíláním velkého počtu segmentů, může způsobit vyčerpání kapacity sítě a také rychle naplnit okno *rwnd* na straně příjemce. Vysílač nejprve nastaví hodnotu *cwnd* na *cwnd* = 1 to znamená, že odešle v daném okamžiku jeden segment. Po odeslání segmentu čeká vysílač na potvrzení, jakmile potvrzení dorazí, je hodnota *cwnd* nastavena na *cwnd* = 2. Vyšlou se tedy 2 segmenty. V závislosti na tom, zda odesílatel přijme jedno nebo 2 potvrzení se *cwnd* zvýší o hodnotu 1 nebo o 2, tedy hodnota *cwnd* bude 3 nebo 4, potom tedy mohou být odeslány 3 nebo 4 segmenty. Rychlost vysílání je tedy exponenciálně zvyšována až na prahovou hodnotu *SSTHRESH* (Slow Start Treshold), kde už hrozí větší pravděpodobnost, že už může dojít ke ztrátě segmentu, a další nárůst už probíhá lineárně, viz Obrázek B. 1.

Vyhýbání se zahlcení (congestion avoidance)

Během vysílání se může stát, že hodnota *cwnd* bude větší než je prahová hodnota *SSTHRESH* a odesláním dalšího dvojnásobku segmentu by způsobilo zahlcení. Jak již bylo zmíněno stav zahlcení je indikován ztrátou segmentu. Cílem tohoto algoritmu je se zahlcení vyhnout a změní exponenciální nárůst hodnoty *cwnd* na lineární. Jestliže vysílač detekuje ztrátu segmentu, je aktuální hodnota *cwnd* snížena na polovinu a stejná hodnota v *cwnd* se nastaví i na prahové hodnotě *SSTHRESH*. Se změněnými hodnotami *cwnd* a *SSTHRESH*,

bude vysílač používat algoritmus pomalého startu jen do té doby, dokud se nedostane na prahovou hodnotu *SSTHRESH*. Po překročení prahové hodnoty *SSTHRESH*, bude hodnota *cwnd* zvyšována lineárně, inkrementována o jedničku, viz Obrázek B. 1.



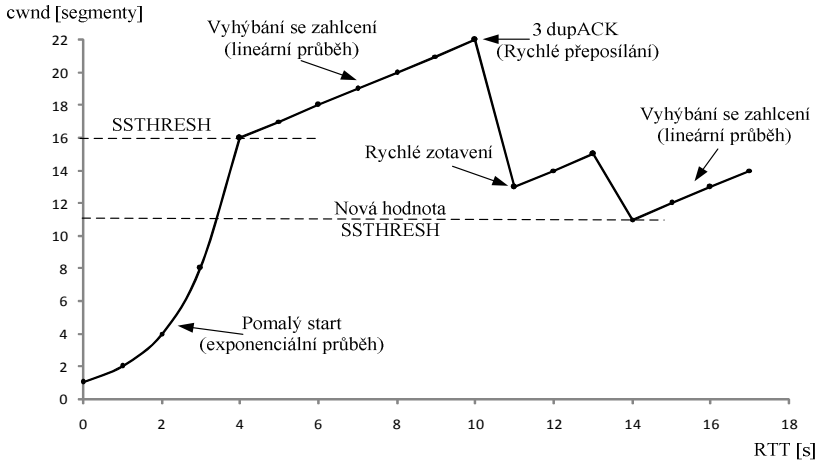
Obrázek B. 1: Fáze pomalý start a vyhýbání se zahlcení

Rychlé přeposílání (fast retransmit)

Při přenosu dat může dojít k tomu, že se během cesty ztratí segment. Příjemce to zjistí až příchodem dalších segmentů mimo pořadí. Příjemce musí po přijetí segmentu mimo pořadí zopakovat (duplikovat) potvrzení posledního správně přijatého segmentu. Po odeslání duplikovaného potvrzení (dupACK) bude přijímač očekávat ztracený segment. Pokud ztracený segment stále nebude přicházet, pošle přijímač druhé dupACK a po chvíli třetí dupACK. Vysílač po příjmu třech dupACK (celkem to jsou čtyři potvrzení na stejný segment), vyhodnotí danou situaci tím, že došlo během přenosu ke ztrátě segmentu. Vysílač okamžitě zopakuje odeslání ztraceného segmentu a potom bude pokračovat v odesílání dalších segmentů. Příjemce příjme ztracený segment, potvrdí ho včetně segmentů, které přijal mimo pořadí již dříve. Algoritmus rychlého přeposílání umožňuje zopakovat pouze ztracený segment a ne všechny segmenty, které nebyly potvrzeny, ještě v době, než dojde k vypršení časovače na straně vysílače, viz Obrázek B. 2.

Rychlé zotavení (fast recovery)

Algoritmus rychlého zotavení zajišťuje, aby po vykonání mechanismu rychlého přeposílání neklesla hodnota *cwnd* na nulu. Hodnota *cwnd* je snížena na nulu teprve až dojde k vypršení časovače. Algoritmus po příjmu třech dupACK, nastaví hodnotu cwnd na polovinu, provede opětovné odeslání ztraceného segmentu a pokračuje ve vysílání segmentů s lineárním nárůstem *cwnd* o 1. Výsledkem je, že se vynechá počáteční exponenciální náběh (slow start), viz Obrázek B. 2.



Obrázek B. 2: Fáze rychlé přeposílání a rychlé zotavení

Omezené vysílání (limited trasmit)

Algoritmus omezeného vysílání umožňuje odesílat nové segmenty i v případě, že to není nutné během stavu zahlcení, konkrétně když:

d) vysílač přijal dva dupACK tj. celkem to jsou tři potvrzení pro stejný segment,
e) okno vysílače nastavené od přijímače dovoluje vyslání segmentu,
f) množství nepotvrzených dat po odeslání nového segmentu nepřekročí hodnotu *cwnd*+2, to znamená, že vysílač má možnost odeslat dva segmenty nad rámec velikosti okna zahlcení.

Algoritmus omezeného vysílání dovoluje odeslat omezené množství dat i přes mechanismy předcházející proti zahlcení.

ALGORITMY TCP PROTOKOLU

Algoritmy TCP protokolu obsahují mechanismy pro předcházení zahlcení. Jejich cílem je včas reagovat na blížící se zahlcení a zpomalit rychlost vysílání dat. Dále se snaží co nejlépe využívat dostupnou šířku pásma.

TCP Tahoe

Prvním algoritmem pro řízení zahlcení byl TCP Tahoe, v jeho implementaci najdeme první tři mechanismy pro předcházení zahlcení. Jsou to pomalý start, předcházení zahlcení a rychlé přeposílání. Ztracený segment pozná vysílač teprve, až dojde k vypršení časovače, to znamená, že vysílač během té doby nedostal potvrzení ACK na odeslaný segment. Počet odesílaných segmentů je tak rychle snížen a velikost okénka cwnd je potom nastaveno na 1, neboli přejde se do fáze pomalého startu.

TCP Reno

Algoritmus TCP Reno obsahuje mechanismy pomalý start, předcházení zahlcení, rychlé přeposílání a rychlé zotavení. Vysílač detekuje ztrátu segmentu po přijetí třech dupACK. Po

přijetích třech dupACK je zmenšeno okénko *cwnd* na polovinu, potom přechází do fáze rychlého přeposílání a po chvíli se dostane opět do fáze rychlého zotavení. Jestliže dojde k vypršení časovače u odeslaného segmentu, přejde se do fáze pomalého startu tak jako u TCP Tahoe. Nevýhodou Rena je, že může pracovat jen v síti, kde je nízká ztrátovost paketů. Jestliže ztrátovost paketů je vyšší, nebude pracovat příliš spolehlivě a jeho výkonnost bude stejná jako u TCP Tahoe. Je to z důvodu, že Reno dokáže detekovat jen jednu ztrátu paketu. Když je ale ztrátovost větší, Reno bude tuto ztrátovost detekovat postupným odesíláním ztracených segmentů v následném potvrzení ACK od přijímače.

TCP Westwood

TCP Westwood je další modifikovaná verze TCP Rena, která při ztrátě segmentu tj. přijetí třech dupACK nesníží velikost okénka cwnd a prahové hodnoty *SSTHRESH* na polovinu. Novou hodnotu *cwnd* a *SSTRESH* vypočítá pomocí doby odezvy RTT z přijatého třetího dupACK a dostupné šířky pásma, neboli provádí postupné adaptivní zvyšování a adaptivní snižování (AIAD – Adaptive Increase Adaptive Decrease) velikosti okénka *cwnd* a prahové hodnoty *SSTRESH*. Tímto způsobem přechází do fáze rychlého zotavení a efektivně se vyhýbá i zahlcení.

TCP Vegas

Tato implementace umožňuje snížit ztrátovost paketu a používá k tomu tři mechanismy, jsou to rychlé přeposílání, vyhýbání se zahlcení a pomalý start. U mechanismu rychlého přeposílání si Vegas zaznamená do paměti dobu, kdy odeslal segment příjemci, tato doba se nazývá časová známka (timestamp). Po přijetí potvrzení ACK vezme vysílač hodnotu časové známky a určí přesnou dobu odezvy *RTT*. Když vysílač příjme dupACK paket, Vegas zkontroluje, jestli je rozdíl uložené časové známky daného segmentu a časové známky dupACK větší jak doba odezvy *RTT*. Pokud je, vysílač okamžitě odešle požadovaný segment a nečeká až příjme 3 dupACK pakety. Dále, když vyprší časovač pro daný segment (vysílač nedostal ACK potvrzení), bude daný segment automaticky přeposlán, aniž by čekal na dupACK paket a vyhne se tím snížení velikosti okénka *cwnd*.

H-TCP

H-TCP je používán hlavně ve vysokorychlostních sítích, kde zmenšením časového intervalu předchází události přetížení. Časový interval se začne zmenšovat, když během vysílání dosáhne linka ke své hranici přidělené šířky pásma. Je to bráno jako informace, která oznamuje, že může dojít k zahlcení a se zvyšující se frekvenci sleduje tuto informaci častěji a přesněji. K tomuto sledování nepoužívá velikost okénka cwnd, ale dobu Δ, která uplynula od poslední události přetížení. Dobu Δ využívá algoritmus adaptivního zvyšování AIMD (Adaptive Increase Multiplicative Decrease) a podle této doby se neustále mění. Dále AIMD obsahuje měřítko *RTT* dané linky, které zajišťuje férovost mezi soutěžícími přenosy s odlišnými časy *RTT*.

Highspeed TCP (HS-TCP)

HSTCP je algoritmus, který se dokáže přizpůsobit velkým okénkům zahlcením během TCP spojení. Pokud je velikost okénka zahlcení menší jak *SSTHRESH* nebo ztrátovost paketů je nanejvýš do 10^{-3}, chová se tento algoritmus jako standardní TCP a při opačné velikosti okénka zahlcení nebo ztrátovosti paketů nižší než 10^{-3} je použit algoritmus HSTCP. Při stavu,

kdy se chová jako HSTCP je na probíhající přenos mnohem agresivnější. Využívá k tomu funkci odezvy, která má 3 parametry, jsou to *Low_Window*, *High_Window* a *High_P*. Parametr *Low_Window* je používán pro zajištění kompatibility a stanovení bodu přechodu, to znamená, když bude aktuální velikost okénka zahlcení menší nebo rovno hodnotě v parametru *Low_Window*, použije HSTCP stejnou funkci odezvy jakou má standardní TCP hodnotu. Naopak pokud aktuální velikost okénka bude větší jak hodnota v parametru *Low_Window* použije funkci odezvy HSTCP. Parametry *High_Window* a *High_P* definují horní hranici funkci odezvy, kde *High_P* je počet zahozených segmentů a *High_Window* je průměrná velikost okénka zahlcení během přetížení. Odezvy funkce HSTCP jsou reprezentovány novými hodnotami aditivního zvyšování a vícenásobného snížení, které se mění podle aktuální velikosti okénka *cwnd*. Algoritmus HSTCP je vhodný pro přenos velkého množství dat, používá se ve vysokorychlostních sítích a dokáže udržovat vysokou přenosovou rychlost, při různých stavech sítě. Přizpůsobování se různým podmínkám v síti provádí už při fázi pomalého startu, to znamená, pokud dojde ke ztrátě segmentu, dovolí využívat menší šířku pásma sítě než 1/2 *cwnd*.

TCP Hybla

Tento algoritmus umožňuje zvýšit propustnost v sítích s dlouho dobou odezvy *RTT* například satelitní a bezdrátové přenosy. Velké zpoždění odezvy *RTT* má za následek velmi rychlé snížení velikosti okénka cwnd a zpomalení rychlosti vysílání. Aby k tomuto snížení nedocházelo, provádí Hybla odstranění závislosti doby odezvy *RTT* během aktualizování algoritmu. Hybla používá selektivní potvrzování SACK (příjemce oznamuje vysílači, které segmenty se ztratily) a umožňuje během jedné doby odezvy *RTT* přeposlat více ztracených segmentů. Dále tento algoritmus využívá i časové známky (timestamps) pro kontrolu odeslaných segmentů, popis u algoritmu TCP Vegas.

TCP Cubic

TCP Cubic je vylepšená verze algoritmu TCP BIC a dokáže lépe spravovat velikost okénka zahlcení. Hlavní myšlenkou algoritmu je, že zvětšení velikosti okénka závisí jen na dvou po sobě jdoucích událostech přetížení, tím dochází k odstranění závislosti na době odezvy *RTT*. Dovoluje tedy ostatním tokům využívat stejnou velikost okénka nezávisle na jejich době odezvy *RTT*. Algoritmus Cubic je optimalizován pro použití ve vysokorychlostních sítích, dokáže jednoduše kontrolovat velikost okénka a je méně agresivní, než varianta BIC, to znamená, že nezabere moc přenosové šířky pásma pro ostatní TCP toky. Dnes je využívaný jako standardní algoritmus v linuxovém jádře.

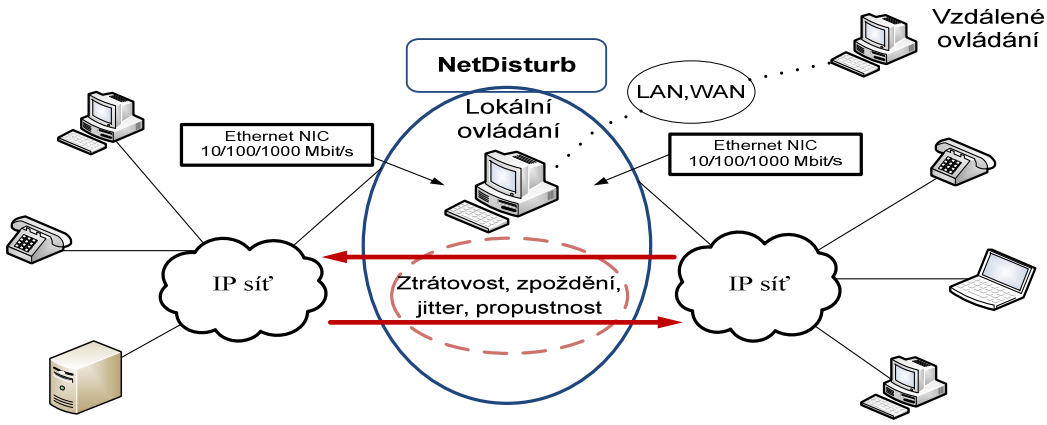
TCP BIC

Algoritmus BIC (Binary Increase Control) má více agresivní chování, než Cubic. Jeho cílem je co nejrychleji dosáhnout maximální dostupné šířky pásma, i když svým chováním omezí ostatní TCP toky. Dále má definovanou minimální a maximální velikost okénka zahlcení. U minimální velikosti okénka zahlcení má jistotu, že nedojde ke ztrátovosti segmentu a maximální velikost okénka zahlcení definuje horní hranici přenosové rychlosti, kde už ke ztrátě segmentu může dojít. Když maximální velikost okénka zahlcení ještě navýší a ke ztrátě segmentu nedochází, stanoví tak novou maximální velikost okénka zahlcení a z původní maximální velikosti okénka zahlcení se následně stane minimální velikost okénka zahlcení. Jestliže, ale dojde ke ztrátě segmentu, je velikost okénka snížena na minimální

velikost. Tento postup neustále opakuje a pokouší se dostat na maximální přidělenou šířku pásma. Čím více se bude blížit k maximální přidělené šířce pásma, jeho chování bude méně agresivní. BIC byl dříve používaný jako standardní algoritmus v linuxovém jádře, ale byl nahrazen jeho vylepšenou verzí TCP Cubic. Díky své rychlosti dosáhnout maximální přidělené šířky pásma je vhodný pro použití ve vysokorychlostních sítích.

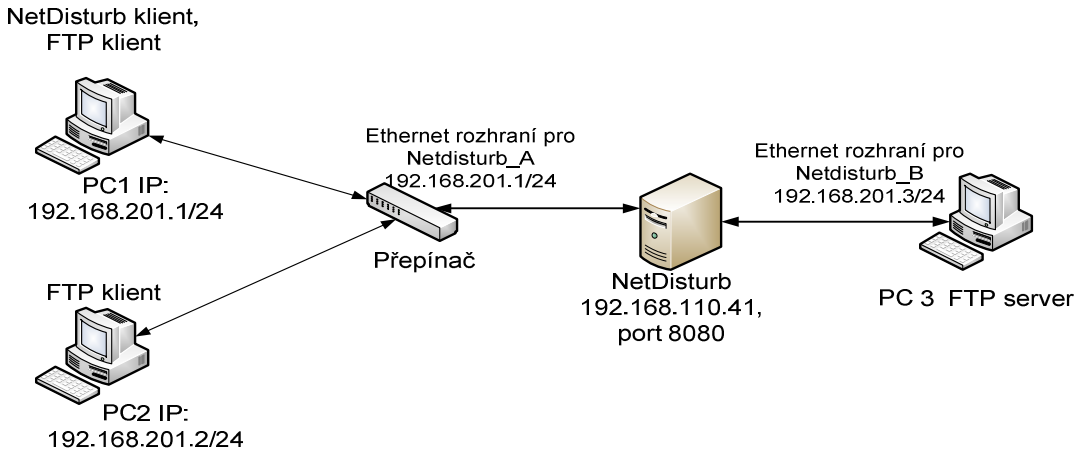
Emulátor sítí NetDisturb

NetDisturb je aplikace, která dokáže emulovat reálné vlastnosti datové sítě, tedy umí zhoršit procházející provoz přes IP sítě (IPv4 a IPv6). Zhoršením je zde myšleno nastavením různého zpoždění, latence, jitteru, omezení šířky pásma, ztrátovosti paketu, kopírování a různé změny nastavení v paketech. Hlavním účelem aplikace NetDisturb je testování různých komunikačních služeb a sledovat jejich chování při zhoršených podmínkách v síti. NetDisturb slouží jako most mezi dvěma ethernetovými segmenty a umožňuje buď jednosměrný, nebo obousměrný přenos paketů na síťových kartách typů Ethernet, Fast Ethernet a Gigabit Ethernet. Jeho součástí je také uživatelské rozhraní pro server a klienta. Obě tato rozhraní mohou být spouštěna nejen na jednom počítači (NetDisturb Server a NetDisturb Client), ale i na dvou počítačích, přičemž u druhé varianty na jednou počítači poběží serverová část NetDisturb Server a na druhém klientská část NetDisturb klient viz Obrázek B. 3. Uživatel na klientské části pak musí vytvořit spojení se serverem pomocí jeho IP adresy a portu, na kterém budou spolu komunikovat, po ověření těchto údajů, je spuštěno uživatelské rozhraní pro klienta.



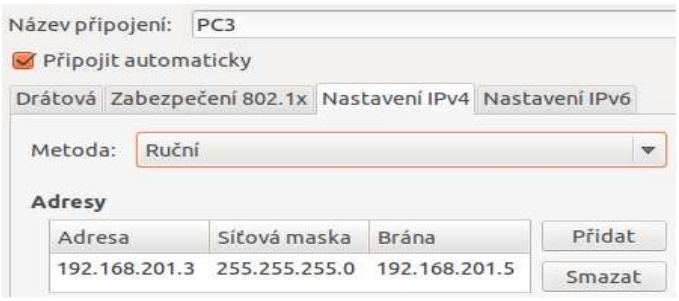
Obrázek B. 3: Princip aplikace NetDisturb

Postup



Obrázek B. 4: Schéma zapojení laboratorní úlohy

Zapojte laboratorní úlohu podle schématu zapojení, viz Obrázek B. 4. Přihlaste se na účet **Student**, heslo **.Student427-** a přidělte jednotlivým stanicím IP adresy. Na PC3 přidělíte IP adresu následovně. Klikněte v pravém horním rohu na ikonku  a z nabídky vyberte *Upravit připojení*. V záložce **Drátová** klikněte na *Přidat* a proveďte nastavení IP adresy pro PC3, viz Obrázek B. 5.



Obrázek B. 5: Nastavení IP adresy pro PC3

FTP server

Na PC 3 s operačním systémem Ubuntu vytvořte FTP server, spusťte terminál a přihlaste se jako root (superuživatel), použijte příkaz *sudo –i*. Pro ulehčení vypisování používejte klávesu Tabulátor (Tab), bude doplňovat názvy složek a souborů. Nainstalujte si program **Proftpd** příkazem *apt-get install proftpd*, při následné instalaci dejte **standalone** (server bude spuštěn jako samostatný proces). Dále vytvořte uživatele *ftpuser* s heslem a přiřaďte ho do skupiny *ftpgroup*. Nejprve vytvořte skupinu *ftgroup* příkazem *addgroup ftpgroup*, dále proveďte příkazem *adduser --ingroup ftpgroup ftpuser* přidání uživatele *ftpuser* do skupiny *ftpgroup*. Po potvrzení budete vyzvání k zadání Unix hesla, dejte **ftp**, potom ho zadejte znovu, zbylé položky ponechejte prázdné a potvrďte klávesou Enter. Kontrolu uživatele a přidělení uživatele do skupiny provedete příkazem *cat /etc/group* a *cat /etc/passwd*. V uživateli *ftpuser* vytvořte složku s názvem *ftp_server*. Nejprve se musíte přepnout do domovské složky uživatele *ftpuser* příkazem *cd /home/ftpuser*, příkazem *mkdir ftp_server* vytvoříte složku. Dále

provedete zkopírování souborů z předem vytvořené složky, její název a umístění vám řekne učitel. Přesunete se do složky, kterou vám sdělil učitel a kopírování souborů provedete příkazem *cp „název souboru“ /home/ftpuser/ftp_server*. Nezapomeňte po zkopírování zkontrolovat zda vlastníkem daného souboru je uživatel *ftpuser* a ne root, jinak nebudete moci stahovat tento soubor stahovat z FTP serveru (*ls -l*), změnte vlastníka souboru a skupiny! Po úspěšné instalaci Proftpd musíte v konfiguračním souboru *proftpd.conf* vytvořit uživatele na FTP serveru a zprovoznit ho. Do terminálu napište příkaz *vim /etc/proftpd/proftpd.conf*, zobrazí se konfigurační soubor FTP serveru. Podle obrázku, viz Obrázek B. 6, proveďte nastavení konfiguračního souboru a vytvořte uživatele na FTP serveru. Stiskněte na PC klávesu **Insert**, budete moci upravovat konfigurační soubor. Pro uložení stiskněte klávesu **ESC** a napište v terminálu **:w** nebo pro celé ukončení a uložení napište **:wq**. Po každé změně v konfiguračním souboru *proftpd.conf* je nutné restartovat FTP server příkazem */etc/init.d/proftpd restart*, jinak se provedené změny neprojeví. V domovské složce uživatele Student, vytvořte složku s názvem například *analyzatcp* (*cd /home/Student* a *mkdir analyzatcp*).

```
ServerName                      "FTP_server Debian"
ServerType                      standalone
DeferWelcome                    off

MultilineRFC2228                on
DefaultServer                   on
ShowSymlinks                    on

TimeoutNoTransfer               600
TimeoutStalled                  600
TimeoutIdle                     1200

DisplayLogin                    TCP_algoritmy.msg
DisplayChdir                    .message true
ListOptions                     "-l"

DenyFilter                      \*.*/

# Use this to jail all users in their homes
 DefaultRoot                    ~!ftp_server

 DefaultRoot                    ~!ftp_server

# Users require a valid shell listed in /etc/shells to login.
# Use this directive to release that constrain.
# RequireValidShell             off

# Port 21 is the standard FTP port.
 Port                           21


# In some cases you have to specify passive ports range to by-pass
# firewall limitations. Ephemeral ports can be used for that, but
# feel free to use a more narrow range.
 PassivePorts                   49152 65534

# If your host was NATted, this option is useful in order to
# allow passive tranfers to work. You have to use your public
# address and opening the passive ports used on your firewall as well
# MasqueradeAddress             1.2.3.4
```

```
    # Umask 022 is a good standard umask to prevent new files and dirs
    # (second parm) from being group and world writable.
    Umask                           022  022
    # Normally, we want files to be overwriteable.
    AllowOverwrite                  on #
    AllowStoreRestart               on #dovoli klientovi znovu zopakovat přerušené stahování
                                       #souboru
    # Uncomment this if you are using NIS or LDAP via NSS to retrieve passwords:
    # PersistentPasswd              off

    # This is required to use both PAM-based authentication and local passwords
    # AuthOrder                     mod_auth_pam.c* mod_auth_unix.c

    # Be warned: use of this directive impacts CPU average load!
    # Uncomment this if you like to see progress and transfer rate with ftpwho
    # in downloads. That is not needed for uploads rates.
    #

# </Anonymous>

  <Anonymous ~/ftp_server> #konfigurace anonymního účtu
   User                    ftpuser
   Group                   ftpgroup

   UserAlias               anonymous ftpuser
   AnonRequirePassword     on  #povolí požadování hesla i pro anonymního uživatele
   RequireValidShell       off #propojí ftp server s uživateli přiřazené k přiděleným
                               #skupinám
   MaxClients              10
 <Directory *ftp_server> #nastavení omezení nebo povolení pro práci ve složce
  AllowOverwrite yes
  <LIMIT WRITE > #povoleni zapisu do složky
   AllowAll
  </LIMIT>
  <LIMIT READ STORE> #povoleni ukládání a čtení
  AllowAll
  </LIMIT>
 </Directory>
</Anonymous>
```

Obrázek B. 6: Konfigurace FTP serveru a uživatele

TCPDUMP, TCPTRACE, XPLOT.ORG, XPL2GPL a GNUPLOT

Nainstalujte si tcpdump (je to síťový zachytávač paketů, jako Wireshark), tcptrace (provádí analýzu TCP spojení ze zachycených paketů), xplot.org (vykresli grafy analyzovaných TCP spojení), xpl2gpl (provádí převod souborů s příponou **.xpl** na příponu **.gpl**) a gnuplot (umožňuje vykreslit více analyzovaných TCP spojení do jednoho grafu). S instalací programu xplot.org by se měl nainstalovat i program xpl2gpl, pokud ne, nainstalujte. Pro zopakování, instalaci provedete příkazem *apt-get install tcpdump*, stačí za **install** napsat názvy programů, které chcete nainstalovat.

Před samotným zachytáváním paketů pomocí tcpdump, zjistěte na kterém síťovém rozhraní je IP adresa 192.168.201.3 (*ifconfig*). Spusťte si další dva terminály (celkem budou tři), v třetím terminálu bude provádět přepínání mezi TCP algoritmy.

Ve dvou terminálech se přesuňte do vytvořené složky analyzatcp v uživateli Student. Vytvořte v ní složky s názvy tcp algoritmů jako cubic, hybla, vegas atd., které Ubuntu podporuje (viz Přepínání mezi algoritmy). Dále ve složce například cubic, vytvořte složky s názvy **10mbitloss1z20** a **10mbitloss1z100**, znamená to, že měření provádíte pro přenosovou rychlost do 10 Mb/s a ztrátovost paketů například 1z 20 (tzn., že 1 z 20 paketů se během cesty ztratí). Podobně vytvořte složky pro přenosové rychlosti 100 Mb/s a 1 Gb/s.

Dále provedete analýzu jednotlivých TCP algoritmů, kde bude ztrátovost náhodná. Vytvořte u každého TCP algoritmu složky s názvy **5loss10mbit**, **5loss100mbit** a **5loss1gbit**. Kde 5loss10mbit znamená, že náhodná ztrátovost bude do 5 % pro přenosovou rychlost do 10 Mb/s.

V obou terminálech se přesuňte do složky například 10mbitloss1z20 a napište příkaz *tcpdump –i eth0 (eth1) –w tcpdump.out port 20* tzn., že síťový analyzátor tcpdump naslouchá na rozhraní eth0 nebo eth1 podle výpisu u příkazu *ifconfig* a ukládá jen zachycené pakety s portem 20 do souboru tcpdump.out.

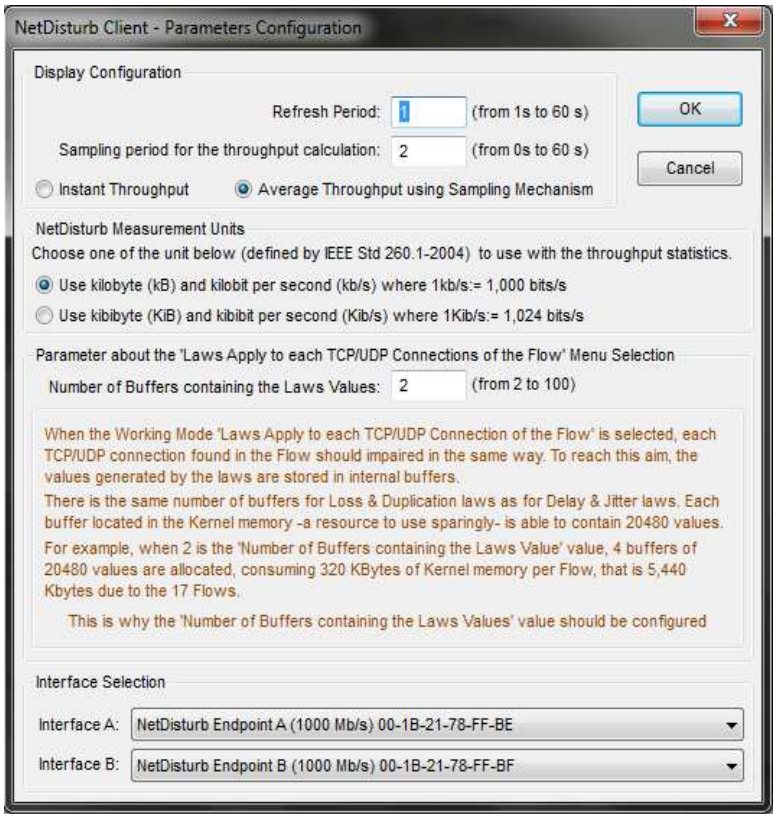
V druhém terminálu budete zachycené pakety analyzovat pomocí tcptrace, napište příkaz *tcptrace –yT –A500 tcpdump.out*, kde příkaz znamená, že tcptrace vytvoří soubory (s příponou .xpl) pro grafickou analýzu propustnosti (přepínač T), bez šumu v pozadí (-y) a bude zpracovávat segmenty v souboru tcpdump.out po 500 (-A500). Ve složce se potom objeví soubory s názvy a2b_tput.xpl, b2a_tput.xpl atd.

Přepínání mezi TCP algoritmy

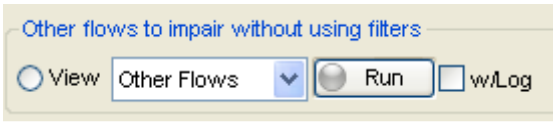
V třetím terminálu pomocí příkazu *ls /lib/modules/"číslo linuxového jádra"/kernel/net/ipv4* si zobrazíte všechny algoritmy, které Ubuntu umožňuje použít. Číslo linuxového jádra může být například *3.0.0-12-generic*. Příkazem *cat /proc/sys/net/ipv4/tcp_congestion_control* si zobrazíte právě používaný TCP algoritmus, ve výchozím nastavení je tcp algoritmus Cubic. Přepínání mezi TCP algoritmy budete provádět příkazem *echo "westwood" > /proc/sys/net/ipv4/tcp_congestion_control*, kde v uvozovkách je zvolený tcp algoritmus a proveďte kontrolu, zda k přepnutí na zvolený algoritmus došlo.

FTP klient a NetDisturb klient

Na PC1 spusťte aplikaci NetDisturb, která je umístěná na ploše. Do zobrazeného dialogového okna *NetDisturb Client – Connection to NetDisturb Server* zadejte IP adresu *192.168.110.41* a port *8080* pro komunikaci se serverem. Po úspěšném navázání spojení se zobrazí uživatelské rozhraní klienta. Uprostřed bude tlačítko s názvem *Configure NIC*, pokud není tak v hlavním menu klikněte na *Actions* pak *Configuration* a proveďte nastavení sítových rozhraní podle obrázku, viz Obrázek B. 7. Pomocí příkazu *ping*, otestujte, zda můžete komunikovat s FTP serverem (PC3) z PC1 nebo PC2. Odezva nebude žádná. Pro komunikaci s PC3 spusťte v aplikaci NetDisturb tlačítkem *Run* provoz bez omezení, viz Obrázek B. 8. Nyní zkuste ping znova, odezva bude úspěšná. Ponechte provoz bez omezení zapnut.



Obrázek B. 7: Nastavení síťových rozhraní



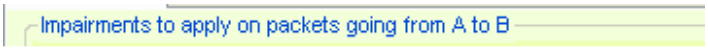
Obrázek B. 8: Provoz bez omezení

Na PC1 a PC 2 pomocí Total Commanderu, vytvořte FTP účet pro spojení s FTP serverem. V dialogovém okně *Připojení k serveru FTP*, klikněte na **Nové připojení,** vytvoříte uživatelský účet pro připojení k serveru FTP. Parametry jsou následující: **Název relace** pojmenujte například ftp-ubuntu, **Hostitel** zadejte IP adresu FTP serveru tedy *192.168.201.3*, **Jméno uživatele:** *ftpuser*, **Heslo:** *ftp*, klikněte na OK. Připojení vytvořeného účtu provedete kliknutím na tlačítko *Připojit*. Pokud jste se úspěšně připojili na FTP server, pokračujte nastavením omezení pro jednotlivé tcp algoritmy.

Nastavení omezení

V uživatelské rozhraní NetDisturb, zastavte provoz bez omezení, klikněte na *View* u provozu bez omezení viz Obrázek B. 8 a všechny nastavení provádějte ve směru od A do B, viz Obrázek B. 9. Dále omezení přenosové rychlosti na 10 Mb/s provedete kliknutím na rozevírací záložku s názvem Delay & Jitter a vyberete *Throughput Limit with Delay*, viz Obrázek B. 10. Tlačítkem *Define* nastavíte potřebné parametry, zpoždění nastavte na 5 ms a velikost paměti ponechte jak je. Nyní klikněte na rozevírací nabídku vedle Delay & Jitter s názvem Loss & Duplication a vyberte z nabídky *1 Packet out of N*, kde nastavte rozmezí 20

a docílíte, že se ztratí 1 paket z 20 během přenosu. Klikněte na OK a potom na tlačítko Run. Pro každou změnu nastavení jako šířky pásma a ztrátovosti je nutné tlačítkem Stop provoz zastavit a potom opětovně spustit.



Obrázek B. 9: Aplikování ovlivnění provozu z A do B

Obrázek B. 10: Nastavení propustnosti

Nyní spusťte na FTP serveru zachytávání paketů pomocí *tcpdump* a do PC1 stáhněte soubor z FTP serveru. Nechte stahování chvíli běžet a sledujte, jakou rychlostí se soubor stahuje. Jakmile je staženo 2 – 3 %, vytvořte nové TCP spojení a to z PC2. Do PC2 stáhněte jiný soubor a sledujte průběh stahování. Až budete mít staženo mezi 10-15 %, klávesou kombinací CTRL+C ukončete zachytávání paketů a zrušte také stahování souborů na obou PC. V druhém terminálu můžete spustit analýzu vytvořeného souboru *tcpdump.out* pomocí *tcptrace*.

Vytvořené soubory od tcptrace, zobrazíte pomocí programu xplot.org (*xplot.org a2b_tput.xpl*). Na FTP serveru budete analyzovat dvě TCP spojení z PC1, PC2. Nejvíce budete používat vygenerované soubory **a2b_tput.xpl** a **c2d_tput.xpl**. Oba si vykreslíte zapsáním příkazu *xplot.org a2b_tput.xpl c2d_tput.xpl*. Převeďte soubory s příponou **.xpl** na **.gpl**, použijte příkaz *xpl2gpl a2b_tput.xpl* a *xpl2gpl c2d_tput.xpl*. Příkazem *ls* si vypište obsah adresáře. Nyní úpravou souboru *a2b_tput.gpl* si zobrazíte naměřená tcp spojení v jednom grafu (*vim a2b_tput.gpl*). Do souboru *a2b_tput.gpl* zkopírujte na první řádek zdrojový kód umístěný na konci laboratorní úlohy s názvem *Zdrojový kód pro vykreslení průběhů v gnuplotu***.** Uložte provedené změny a graf vykreslíte příkazem *gnuplot a2b_tput.gpl*. Tento postup opakujte pro všechny provedená měření u každého TCP algoritmu.

Analýzu souboru tcpdump.out pomocí tcptrace a vykreslování grafu provádějte během stahování souborů ze serveru až pro další nastavené parametry.

Po dokončení předchozího měření, nyní proveďte měření pro náhodnou ztrátovost. Použijte z rozevírací nabídky názvem Loss & Duplication položku *Percentage of Loss*. Do editačního okna s názvem **Percentage** nastavte 5 %, **Minimum Burst** na 1 a **Maximum Burst** na 2. Toto nastavení během měření neměňte, ale šířku pásma ano.

Zpracujte naměřené hodnoty, graficky znázorněte a porovnejte s grafy v předchozím úkolu. Je mezi zobrazenými průběhy rozdíl, vysvětlete, proč a který byste použili TCP algoritmus na vašem PC?

Smažte vytvořeného uživatele *ftpuser*, jak na FTP serveru tak i jeho účet spolu s jeho přidělenou skupinou (použijte příkazy dostupné na konci laboratorní úlohy) a veškeré naměřené hodnoty i s průběhy. Vypněte počítače a odejděte.

Kontrolní otázky

1. Jaké porty používá FTP protokol pro komunikaci?
2. Je komunikace pomocí FTP protokolu bezpečná, jestli ne tak proč a jak to můžete ověřit?
3. Jakým příkazem zabráníte, aby se uživatel nedostal do kořenového adresáře FTP serveru?
4. Jaké TCP algoritmy mohou být použity pro vysokorychlostní sítě?
5. Které TCP algoritmy se snaží zabrat celou šířku pásma pro sebe a znevýhodnit tím ostatní TCP provozy?

Literatura

[1] *NetDisturb User Guide.* 2010, 221s. Dostupný z WWW: <http://www.zti-telecom.com/User_GuidesN/NetDisturb_User_Guide_V4.9.pdf>.

[2] MOLNÁR, Karol. *TCP bez korekce*. c2009. Dostupný z WWW: <www.utko.feec.vutbr.cz/ molnar/>. s. 27.

[3] http://www.proftpd.org/

Zdrojový kód pro vykreslení průběhů v programu gnuplot

```
set title "Student.local:20_==>_192.168.201.1:5014,\rStudent.local:20_==>_192.168.201.2:5012" # vloží název grafu
set xlabel "Čas" #popisek osy x
set ylabel "Propustnost [B]" #popisek osy y
set format x "%H:%M:%S" # nastavení formátu času H-hodiny, M-minuty, S – sekundy
set format y "%.0f" #zobrazí čísla bez desetinných míst
set xdata time #povolení zobrazení času
set grid #zobrazení mřížky
set offset 18,15 #provede posunutí grafu na ose x
set xtic rotate by -20 scale 0 #popisek osy x bude zobrazen šikmo
set nokey #vypne legendu
set style data lines # povolí změnit barvy původních čar na jiné barvy
set style line 1 lt 1 lc rgb "black" lw 1 # nastavení čáry pro černou barvu s tloušťkou 1 (lw 1)
set style line 2 lt 1 lc rgb "blue" lw 1
load "a2b_tput.labels"; # načte hodnoty z a2b_tput.labels
load "c2d_tput.labels";
#vykreslení více průběhů do jednoho grafu, vygenerováno programem xpl2gpl, jsou zde hodnoty pro vykreslení #jednotlivých křivek
plot "a2b_tput.datasets" index 0 using ($1-946684800.0):2 with lines, "a2b_tput.datasets" index 1 using ($1-946684800.0):2 with lines, "a2b_tput.datasets" index 2 using ($1-946684800.0):2 with dots, "a2b_tput.datasets" index 3 using ($1-946684800.0):2 with dots, "c2d_tput.datasets" index 0 using ($1-946684800.0):2 with lines ls 1, "c2d_tput.datasets" index 1 using ($1-946684800.0):2 with lines ls 2, "c2d_tput.datasets" index 2 using ($1-946684800.0):2 with dots, "c2d_tput.datasets" index 3 using ($1-946684800.0):2 with dots;
set term postscript # nastaví terminál na postscript
set output "a2b_tput.ps" #vytvoří výstup ve formátu a2b_tput.ps
set output "c2d_tput.ps"
replot #opětovné vykreslení grafu
pause -1
```

Doporučené příkazy pro práci v terminálu

ls	vypíše obsah adresáře

ls –l	vypíše obsah adresáře s právy vlastníka
chmod	změní práva uživatele
chgrp	změní skupinu vlastníka daného souboru
chown	změní vlastníka souboru
cd	přesunutí o adresář výše
cd ..	přesunutí se o jeden adresář níže
cat	vypíše obsah souboru
less	vypíše obsah souboru po stránkách
mkdir	vytvoří adresář
rm –rf	vymaže adresář i s jeho obsahem
userdel	smaže uživatele
groupdel	smaže skupinu
poweroff	vypnutí operačního systému Ubuntu
ifconfig	zobrazí nebo nakonfiguruje síťová rozhraní
man	zobrazí manuálové stránky programů, například man tcptrace

Příkazy pro práci s textovým režimem vim

vim *proftpd.conf*	spustí konfigurační soubor *proftpd.conf* pro FTP server
:q	ukončení textového režimu vim bez uložení a beze změn
:wq	ukončení textového režimu vim s uložením a provedenými změnami
:q!	ukončení textového režimu vim bez provedených změn i přes provedené změny

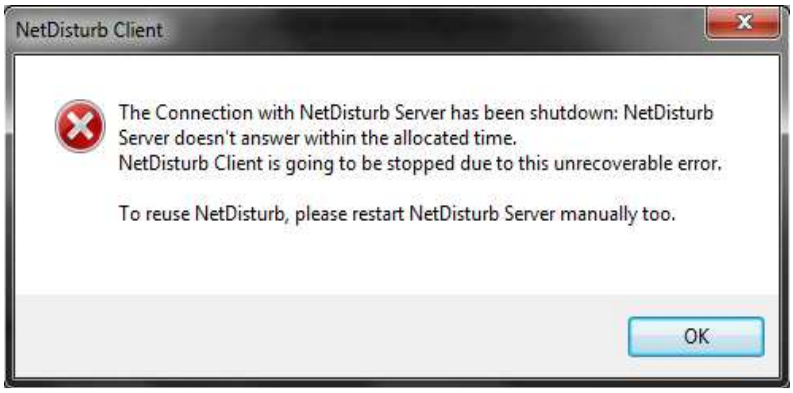
Poznámky pro učitele

Laboratorní úloha Analýza TCP algoritmů je časově náročná, doporučuji, ať si studenti sami vyberou, jaké TCP algoritmy budou analyzovat nebo ať vybere sám učitel.

Vytvořte na operačním systému Ubuntu v uživateli Student složku a vložte do ní soubory pro stahování (minimální velikost souboru 700 MB), potom studentům sdělte její název a umístění. Studenti provedou zkopírování jejího obsahu do své vlastní vytvořené složky.

Pokud došlo na fakultě k výpadku proudu, neuvidí studenti v klientu NetDisturb síťová rozhraní pojmenované Netdisturb_A a NetDisturb_B (záložka *Actions*, položka *Configuration* a *Interface Selection*), je zapotřebí se připojit na server a opětovně je spustit.

Když studenti spustí klienta Netdisturb a po chvíli se jim zobrazí varovné hlášení, viz Obrázek B. 11, aplikace „shodila“ celý server a je zapotřebí ho celý restartovat a opětovně spustit serverovou část aplikace NetDisturb.



Obrázek B. 11: Varovné hlášení

CD

- Diplomová práce formát pdf.