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

ANALÝZA VLIVU VELIKOSTI OKNA A ZPOŽDĚNÍ NA EFEKTIVITU TCP SPOJENÍ

ANALYSIS OF THE EFFECT OF DELAY AND WINDOW SIZE ON TCP CONNECTION EFFICIENCY

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE
AUTHOR
BC. MARTIN KAVICKÝ

VEDOUCÍ PRÁCE
SUPERVISOR
DOC. ING. KAROL MOLNÁR, PH.D.

BRNO 2010

VYSOKÉ UCENÍ TECHNICKÉ V BRNE

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Martin Kavický ***ID:*** 78051
Ročník: 2 ***Akademický rok:*** 2009/2010

NÁZEV TÉMATU:

Analýza vlivu velikosti okna a zpoždění na efektivitu TCP spojení

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s mechanismy řízení toku dat u transportního protokolu TCP. Vytvořte model v simulačním prostředí OPNET Modeler, který názorně ukazuje chování mechanismu řízení toku dat protokolu TCP a jejich reakce na různé události v síti (např. ztráta, zpoždění či změna poradí TCP segmentu, změna velikosti okna, atd.). Proveďte podrobnou analýzu vlivu uvedených událostí na krátkodobou i dlouhodobou průměrnou rychlost TCP přenosu. V případě analýzy vlivu okna berte do úvahy jak vliv staticky nastavené maximální velikosti, tak i vliv dynamického omezení velikosti okna během přenosu. Navrhněte metodu, která umožňuje dynamické řízení rychlosti TCP spojení umělým vyvoláním zmíněných událostí. Vytvořený simulační model podrobně zdokumentujte a výsledky simulací a provedené analýzy řádně zdůvodněte.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, W., High-Speed Networks and Internets: Performance and Quality of Service, 2002, Prentice Hall, 2002
[2] Defense Advanced Research Projects Agency, Transmission Control Protocol, Protocol Specification, RFC 793, IETF, 1981
[3] BRADEN, R., Requirements for Internet Hosts -- Communication Layers, RFC 1122, IETF, 1989

Termín zadání: 29.1.2010 ***Termín odevzdání:*** 26.5.2010

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

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

Anotace

Obsahem diplomové práce je popis pole Klouzavého okna a jeho rozšiřujících algoritmů, jimiž jsou algoritmy Pomalého startu, Dynamické nastavení okna v případě zahlcení, Rychlé opakování přenosu a Rychlé zotavení. Dále je popsáno vytvoření modelu v simulačním prostředí programu Opnet Modeler. V tomto simulačním prostředí byly provedeny analýzy rozšiřujících algoritmů Klouzavého okna a dále byly analyzovány reakce průměrné přenosové rychlosti na změnu velikosti objemu přenášených dat, velikost ztrátovosti, velikost zpoždění v krátkém a dlouhém časovém úseku a změnu velikosti paměti přijímací strany. V poslední části tohoto dokumentu je navržena metoda umožňující řízení přenosové rychlosti pomocí dynamického nastavení velikosti paměti přijímací strany.

Klíčová slova

Klouzavé okno, Pomalý start, Dynamické nastavení okna, Rychlé opakování, Rychlé zotavení, protokol TCP, FTP přenos, přenosová rychlost, ztrátovost, zpoždění, velikost paměti

Abstract

Content of master's thesis is description field of Sliding window and it's expansion algorithms, witch are Slow start, Congestion avoidance, Fast Retransmit and Fast Recovery algorithm. Thereinafter is described creation of model in Opnet Modeler's simulation area. In this simulation area was analyzed reactions of average transfer speed onto variance of data size, lost ratio, latency in short and long time slot and variance of receiver's buffer size. In last section of this document is method design witch makes it possible of transfer speed control through the use of receiver's buffer size dynamic setting.

Key words

Sliding window, Slow start, Congestion avoidance, Fast Retransmit, Fast recovery, TCP protocol, FTP transfer, transfer speed, lost ratio, latency, buffer size

Citace práce

KAVICKÝ, M. Analýza vlivu velikosti okna a zpoždění na efektivitu TCP spojení. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 53 s. Vedoucí diplomové práce doc. Ing. Karol Molnár, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma **Analýza vlivu velikosti okna a zpoždění na efektivitu TCP spojení** 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 jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne …… ……..

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce doc. Ing. Karolu Molnárovi, Ph.D., za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne …… ……..

(podpis autora)

Obsah

Seznam obrázků

Seznam Tabulek

Seznam Grafů

Úvod

V dnešní době každý z nás takřka denně využívá protokol TCP ať už při prohlížení internetových stránek, čtení emailů, či stahování dat z FTP serverů. Každý z nás podvědomě očekává při poskytování těchto služeb jistou kvalitu, očekáváme rychlý přenos námi požadovaných dat. To však není snadné zajistit, protože mnohdy nevědomky provádíme přenos dat přes celé kontinenty, a tedy přes velký počet uzlů v síti ležící mezi zdrojem požadovaných dat a námi.

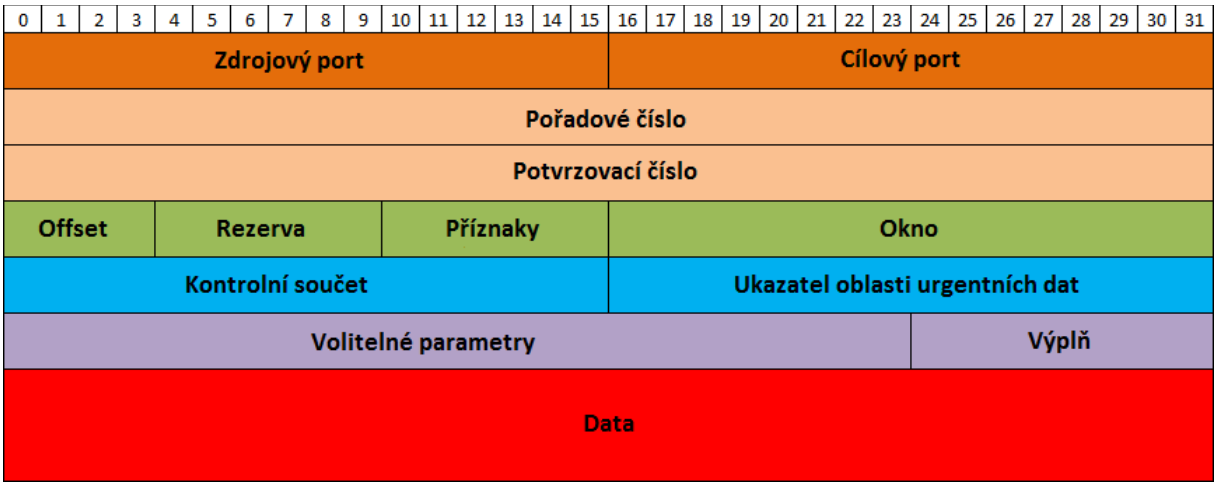
Na přenášená data pak čekají v síti různé nástrahy znemožňující jejich včasné a rychlé doručení. Mohou to být různé výpadky linek, zahlcení uzlů, pomalé zpracování dat uzly atd. V důsledku zahlcení některého z mezilehlých uzlů jsou data zpožděna, nebo dokonce zahozena. Vysílací stanici pak nezbude nic jiného než tato data znovu odeslat. Tyto jevy výrazně ovlivňují rychlost přenosu.

Tento dokument se zabývá popisem mechanismů, které se snaží řídit tok dat mezi dvěma entitami na úrovni transportní vrstvy, konkrétně u protokolu TCP. Popisuje mechanismy, které mají vliv na přenosovou rychlost mezi vysílací a přijímací stanicí.

V první části dokumentu jsou popsány mechanismy rozšiřující klasické řízení toku dat pomocí Klouzavého okna. Tyto mechanismy se snaží předcházet zahlcení sítě. Dále v případě zahlcení uzlu a ztráty dat definuje postupy, jak řídit odesílání dat, aby se síťový uzel dostal ze stavu zahlcení. Další část dokumentu popisuje simulaci chování protokolu TCP v modelové síti. Zde bude ověřeno, jak mechanismy Klouzavého okna pracují. V další části dokumentu bude zkoumáno, jaký vliv má na průměrnou přenosovou rychlost velikost přenášených dat, ztrátovost, zpoždění, velikost paměti přijímací stanice a maximální velikost segmentu. Vliv zpoždění bude zkoumán z hlediska krátkého a dlouhého časového úseku. V poslední části tohoto dokumentu bude řešena implementace dynamického nastavení velikosti okna do programového kódu modelu jedné z koncových stanic.

1 Klouzavé okno a jeho rozšíření

TCP využívá k řízení toku dat Klouzavé okno (viz Obr. 1). Od ostatních protokolů pro řízení se však liší oddělením funkce potvrzování přijatých dat od přidělování kreditů pro vysílání. V hlavičce TCP segmentu jsou důležitá pole, která slouží pro řízení toku dat. Jsou to pole Sekvenčního čísla SN, pole Potvrzovacího čísla AN, a pole Klouzavého okna W.



Obr. 1: Hlavička TCP segmentu.

Pole pro Klouzavé okno má 16 bitů. Skutečnou velikost Klouzavého okna, tzv. kredity, určuje přijímací stanice. Počáteční nastavení Klouzavého okna se provádí při sestavování spojení. Kredity představují počet oktetů, které má vysílací stanice povoleno odeslat bez čekání na potvrzení.

Každý oktet má přiděleno sekvenční číslo SN. TCP segment nese v hlavičce SN toho oktetu, který je na první pozici v datové části. Po odeslání TCP segmentů se velikost okna W zmenší o velikosti datových částí odeslaných TCP segmentů.

Přijímací stanice přijaté TCP segmenty potvrdí jednotlivě či kumulativně vysláním potvrzovacího segmentu, případně využitím datového segmentu zasílaného v opačném směru. Vysílací stanice obdrží tento potvrzovací segment a z pole AN přečte SN očekávaného TCP segmentu. V potvrzovacím segmentu se také nastavuje velikost Klouzavého okna W, ve kterém přijímací stanice přiděluje vysílací stanici další kredity. Přijímací stanice má možnost potvrdit přijaté segmenty, ale vysílací stanici nové kredity nepřidělit.

Tento mechanismus se však nijak nestará o to, jestli nedochází k zahlcení. Proto se pro mechanismus Klouzavého okna zavádí rozšíření, které tyto problémy pomáhají řešit a snaží se jim předcházet. [3][4]

1.1 Rozšíření mechanizmu Klouzavého okna

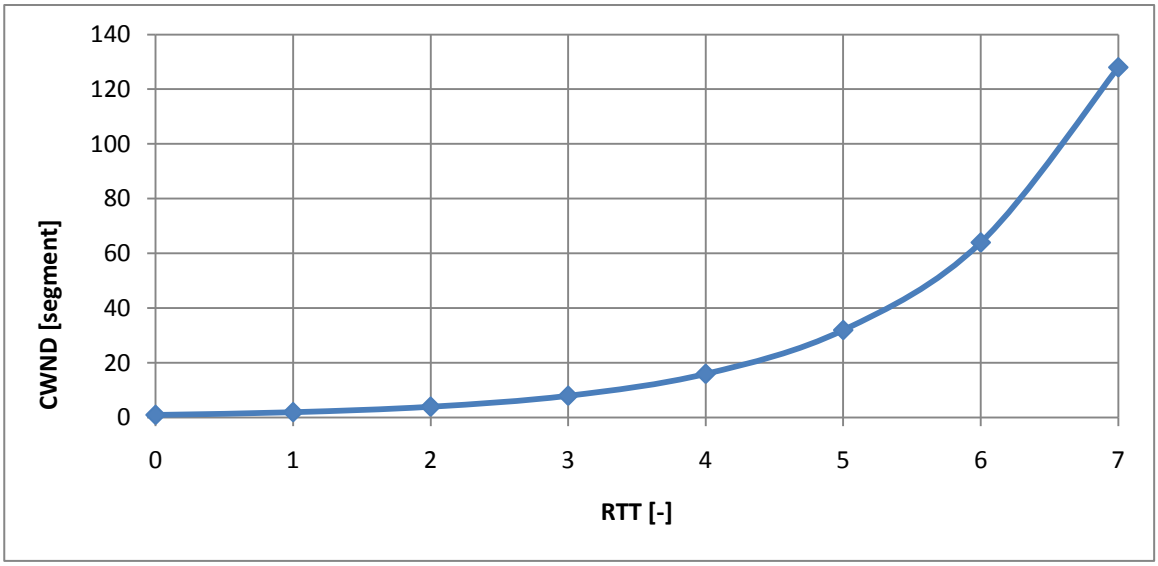
Možnosti rozšíření mechanismu Klouzavého okna jsou:

1. Pomalý start (Slow start)
2. Dynamické nastavení okna v případě zahlcení (Congestion Avoidance)
3. Rychlé zotavení (Fast Recovery)
4. Rychlé opakování přenosu (Fast Retransmit)

1.1.1 Pomalý start (Slow start)

Tento mechanizmus se používá při zahájení přenosu mezi stanicemi nebo když dojde k zahlcení sítě. Po sestavení spojení může vysílací stanice zvolit poměrně velkou počáteční velikost Klouzavého okna. Správnou velikost pak odvodí samočasovací algoritmus na základě doby návratu potvrzení. Je zde však velké riziko zahlcení sítě příliš velkým počtem segmentů. Nebudou se vracet potvrzení a vysílač včas nezaregistruje, že překročil kapacitu sítě. Z toho důvodu se použije metoda pomalého startu. Princip spočívá v přidání dalšího okna, tzv. okna zahlcení (CWND - congestion window).

CWND se nastaví na velikost jednoho segmentu. Vysílací stanice odešle jeden segment a pak čeká na potvrzení zprávou ACK. Jakmile přijde potvrzovací zpráva ACK jednoho segmentu, vysílací stanice zvětší hodnotu CWND o jedna, tedy na 2 segmenty, a tyto segmenty odešle. Po dalším přijetí ACK na oba odeslané segmenty se velikost CWND opět zvýší pro každé přijaté ACK o jedna atd. Tento růst je exponenciální (viz Obr. 2).



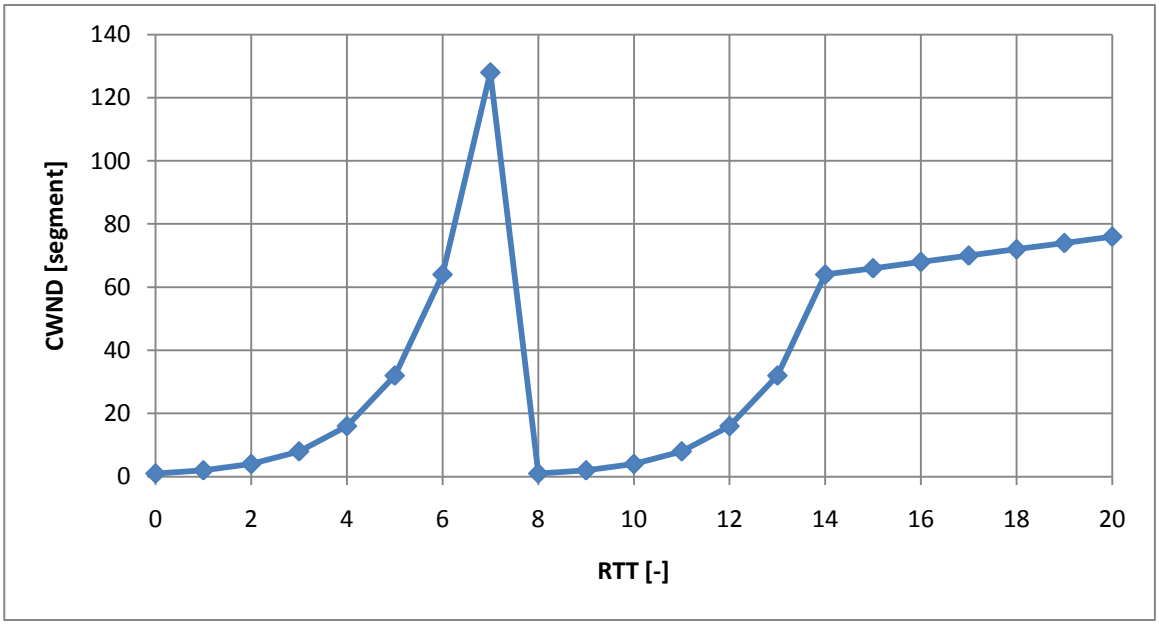
Obr. 2: Velikost CWND u algoritmu Slow Start.

Doba RTT je rovna době uplynulé od odeslání segmentu po přijetí ACK. V grafu uvažujeme tuto dobu jako konstantní. Vysílací stanice může odeslat množství dat rovnající se minimální velikosti z CWND a počtu kreditů. Velikost CWND se zvětšuje až do hodnoty velikosti okna přijímače, nebo o něco málo větší. Tento algoritmus se také používá u metody dynamického nastavení okna, která bude popsána níže. Algoritmus pomalého startu se používá po zahájení spojení a pro omezení datového toku během zahlcení. [3][4]

1.1.2 Dynamické nastavení okna v případě zahlcení (Congestion Avoidance)

Tento algoritmus spolupracuje s algoritmem pomalého startu. Jeho čas nastává v okamžiku, kdy vysílač neobdrží ACK v určené době. Tento jev signalizuje vysílací stanici ztrátu segmentu z důvodu zahlcení některého z mezilehlých uzlů, či samotného přijímače. Vysílací stanice tedy musí zpomalit své vysílání.

Tato metoda má navíc novou proměnnou, tzv. ssthresh, jejíž hodnota je po navázaném spojení 65 535. V okamžiku ztráty segmentu se aktuální hodnota okna (okno = CWND + Klouzavé okno) zmenší na polovinu a její hodnota se uloží do ssthresh. Následně je CWND nastaveno na 1. Vysílací stanice je nyní opět ve stavu pomalého startu s tím rozdílem, že exponenciální růst počtu segmentů skončí v okamžiku, kdy hodnota CWND bude rovna hodnotě ssthresh. Pak se vysílač přepne do stavu zabránění zahlcení a bude zvyšovat CWND lineárně, o 1 (viz Obr. 3). [3][4]



Obr. 3: Algoritmus dynamického nastavení okna v případě zahlcení.

1.1.3 Rychlé opakování přenosu (Fast Retransmit)

V přenosové síti může nastat okamžik, kdy přijímací stanice obdrží segment mimo pořadí, tedy čísla segmentů nepůjdou v řadě za sebou. V tom případě přijímací stanice okamžitě bez jakéhokoliv zpoždění vyšle duplikované ACK s pořadovým číslem očekávaného segmentu. Vysílací stanice však neví, jestli duplikované ACK značí ztrátu segmentu či přijetí mimo pořadí. Proto čeká na přijetí dalších duplikovaných ACK.

U přijetí mimo pořadí se obecně předpokládá, že segment přijatý mimo pořadí nebyl zpožděn více než o dva následující segmenty. To by značilo přijetí dvou duplikovaných ACK. Pokud však jde o ztrátu segmentu, je tato ztráta značena odesláním většího počtu duplikovaných ACK z přijímací stanice v řadě za sebou. Vysílač pak okamžitě znovu odešle chybějící segment, který byl uveden v duplikovaném ACK, a nečeká na vypršení časovače znovu odeslání.

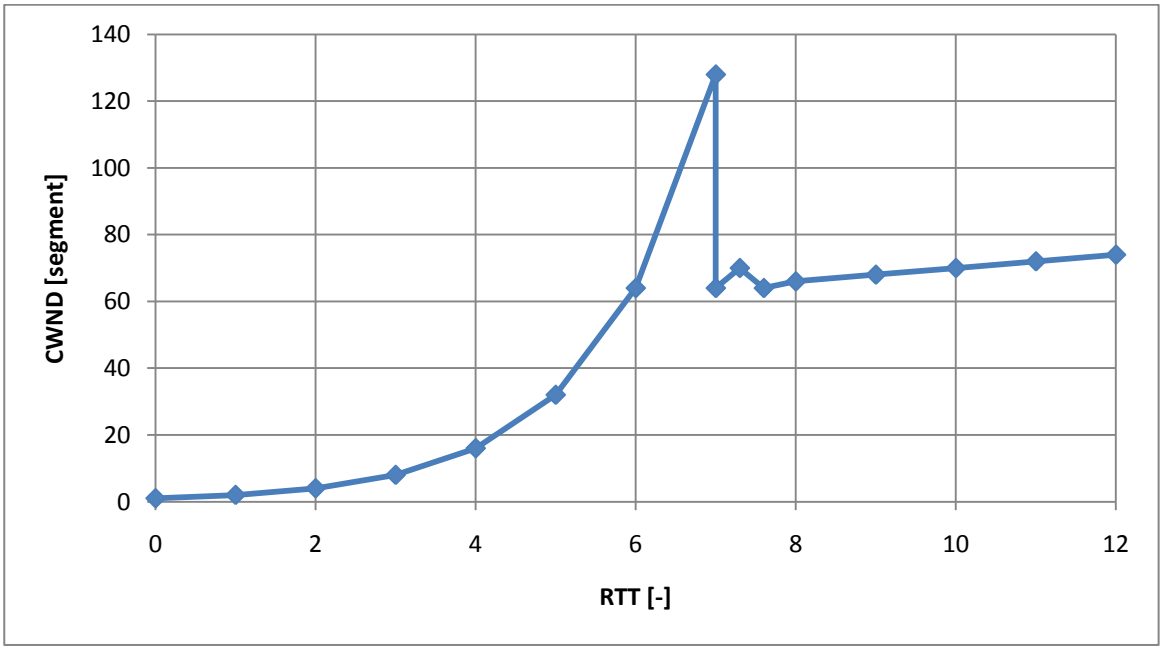
Pokud přijímač přebírá segmenty od IP vrstvy pouze v pořadí, má tato metoda jednu nevýhodu. Nevýhodou metody je opakování vysílání segmentů následujících po segmentu přijatém mimo pořadí, či ztraceném, i když přišly v pořadí, a neztratily se. Důvodem je to, že tato metoda neumí seřadit segmenty přijaté mimo pořadí do správného pořadí. Ve většině případů však IP vrstva přebírá segmenty i mimo pořadí. [3][4]

1.1.4 Rychlé zotavení (Fast Recovery)

Tento algoritmus jde ruku v ruce s algoritmem rychlého odeslání. Je velmi podobný algoritmu dynamického nastavení okna v případě zahlcení. V tomto případě však v síti k úplnému zahlcení nedošlo. Pouze se ztratil jeden nebo několik málo segmentů. Proto zde není efektivní spustit mechanismus dynamického nastavení okna v případě zahlcení.

Přijme-li vysílací stanice třetí duplikované ACK v pořadí, znamená to, že segment byl ztracen, ale nejméně další tři segmenty byly doručeny. Tyto segmenty budou uloženy v bufferu přijímací stanice. Hodnota CWND se zmenší na polovinu, ne však na méně než dva segmenty, a je uložena do proměnné ssthresh. Pak se k CWND připočte hodnota velikosti segmentu za každé přijaté duplikované ACK. Poté se odešle ztracený segment.

ACK, které pak vysílací stanice přijme, potvrzuje znovu odeslaný segment a také segmenty uložené v bufferu přijímače. CWND se nastaví na hodnotu uloženou v ssthresh a následně se hodnota CWND zvyšuje lineárně (viz Obr. 4). [3][4]



Obr. 4: Algoritmus rychlého zotavení.

2 Simulace

Pro provádění simulací byl použit program OPNET IT Guru Academic edition 9.1. Tento program umožňuje sledovat jevy, které v síti mohou nastat v závislosti na nastavovaných parametrech jednotlivých prvků sítě. Díky jeho grafickému prostředí je jeho ovládání velmi jednoduché. Ze simulovaných modelů je pak možno vytvářet různé statistiky a zobrazovat grafy.

Byl tedy simulován FTP přenos z FTP serveru k uživatelské stanici. Byl vytvořen nový projekt a v něm nový scénář. Pro vytvoření architektury scénáře byly nutné tyto objekty:

- IP cloud.
- Dva směrovače.
- Ethernet server.
- Ethernet klient.
- Dvě duplexní sériové linky PPP DS3.
- Dvě Ethernetové linky 100BaseT.

2.1 Popis použitých prvků sítě

Linka 100BaseT

Linka 100BaseT je duplexní linka představující Ethernetové spojení s přenosovou rychlostí 100Mb/s. Může spojit jakoukoliv kombinaci uzlů, jako je stanice, hub, most, přepínač, a uzly LAN s výjimkou kombinace hub – hub.

Linka PPP DS3

Linka PPP DS3 spojuje dva uzly, které mají implementovaný IP protokol. Přenáší IP diagramy rychlostí 44,736 Mb/s. Jedná se o standard používaný především v USA.

ip32 cloud

IP cloud je model s větším množstvím sériových rozhraní s možností výběru přenosové rychlosti. IP pakety, které dorazí na kterýkoliv vstup, jsou směrovány na příslušné výstupní rozhraní podle IP adresy paketu. Je zde možné konfigurovat směrovací protokoly RIP a OSPF pro automatické vytváření dynamických cest.

IP cloud vyžaduje určité množství času pro směrování paketů, které je možno definovat v položce **Zpoždění paketu (Packet Latency)**. Pakety jsou směrovány pravidlem, kdy první příchozí je jako první obsloužen. Pracuje s protokoly TCP, UDP, IP, RIP, OSPF, BGP, IGRP.

Směrovač

Popisem je velmi podobný objektu IP cloud. Má však jen malé množství sériových rozhraní a několik Ethernetových rozhraní. Pracuje s protokoly IP, UDP, Ethernet, RIP, OSPF a SLIP.

Ethernet server

Ethernet server představuje uzel se serverovými aplikacemi běžících na protokolech TCP/IP a UDP/IP. Podporuje Ethernetové spojení na rychlostech 10 Mb/s, 100 Mb/s a 1 Gb/s. Rychlost je definována typem připojené linky. Server může pracovat jak v režimu full-duplex, tak i v režimu half-duplex. Podporuje protokoly IP, TCP, UDP, Ethernet, Fast Ethernet, Gigabit Ethernet, RIP a OSPF.

Ethernetová pracovní stanice

Ethernetová pracovní stanice je nakonfigurována jako klient určité síťové služby či služeb. Klient běží na protokolech TCP/IP a UDP/IP. Dále podporuje 10Mb/s, 100Mb/s a 1Gb/s Ethernetové spojení. Podporuje protokoly TCP, UDP, IP, IEEE 802.3 (Ethernet, Fast Ethernet, Gigabit Ethernet), RIP a OSPF .

Objekt definice aplikace

Slouží k výběru a nastavení aplikací, které budou v modelu sítě provozovány. Je možné vybrat z 16 předdefinovaných aplikací, jako je FTP, Email, přístup do databáze, video-hovor, tisk souboru, Telnet, VoIP a prohlížení webu. U každé předdefinované aplikace je možné editovat její vlastnosti.

Objekt definice profilu

Pomocí profilu se určuje, kdy přesně se má jaká aplikace spustit, ukončit a kolikrát se bude opakovat. Jeden profil může spouštět více aplikací a tyto aplikace se mohou spouštět zároveň nebo v různém čase. Zároveň je možné nastavit spuštění a konec běhu profilu, stejně jako i jeho opakované spouštění.

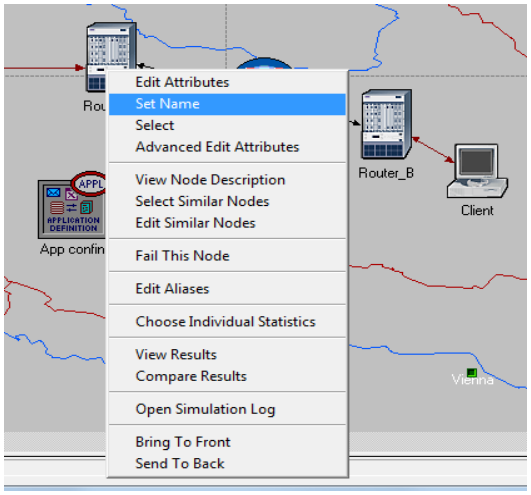
2.2 Vytvoření simulačního modelu sítě

V základním okně programu IT Guru byla otevřena položka **File → New → Project**. Do kolonky **Project name** v nově otevřeném okénku průvodce nastavení bylo napsáno jméno projektu, například **TCP_CWND**, a do kolonky **Scenario name** jméno scénáře. Zde bylo napsáno **No-Drop** a potvrzeno stiskem **OK**. Jak napovídá název, bude se simulovat přenos dat z FTP serveru bez jakékoliv ztráty dat v síti IP cloud. Dále bylo zvoleno vytvoření prázdného scénáře vybráním **Create Empty Scenario** a stiskem **Next** se pokračovalo k volbě přidání rodiny modelů objektů na kartě **Select Technologies**. Zde bylo zvoleno přidání rodiny modelů **Internet Toolbox**. Pak byl dokončen průvodce stisknutím tlačítka **Finish**. Objevila se pracovní plocha nově vytvořeného scénáře s mapou Evropy. Pomocí tlačítka **Zoom** byla přiblížena oblast s mapou České Republiky.

Na plochu scénáře byly vloženy z palety objektů tyto objekty:

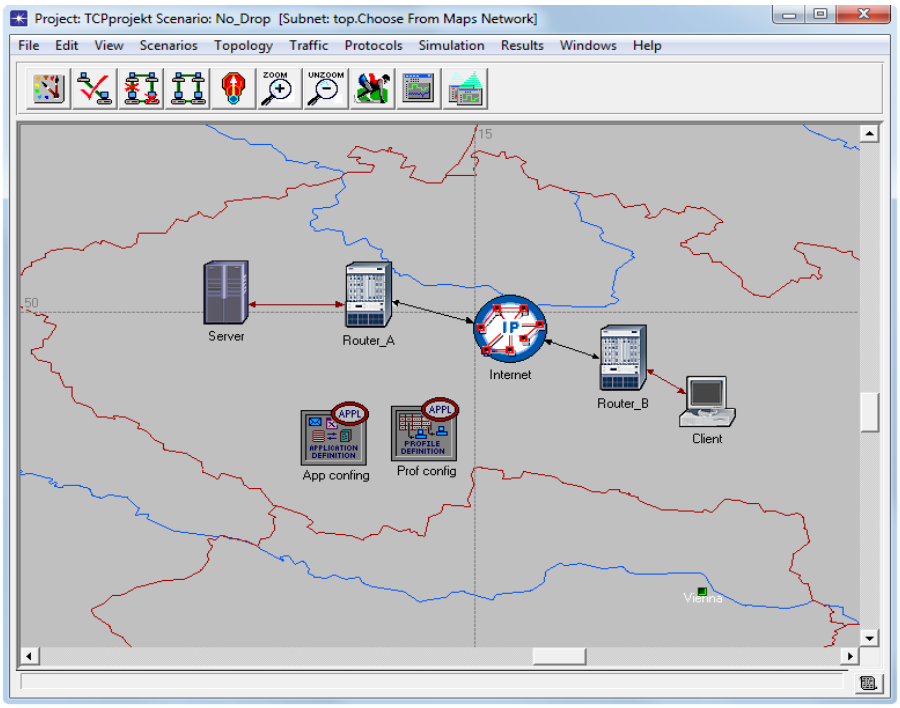
- ip32_cloud,
- dva směrovače ethernet4_slip8_gtwy,
- ethernet_server,
- ethernet_wkstn.

Nakonec byl vložen objekt **Application Config** a **Profile Config**. Objekt ip32_cloud byl propojen sériovými duplexními linkami **PPP_DS3** se směrovači. Kliknutím pravého tlačítka myší na jakýkoliv objekt a plochu se zobrazí kontextové menu (viz Obr. 5). Zde byla zvolena položka **Set Name**. Jednotlivé objekty byly pojmenovány tak, jak je znázorněno na Obr. 6.



Obr. 5: Kontextové menu.

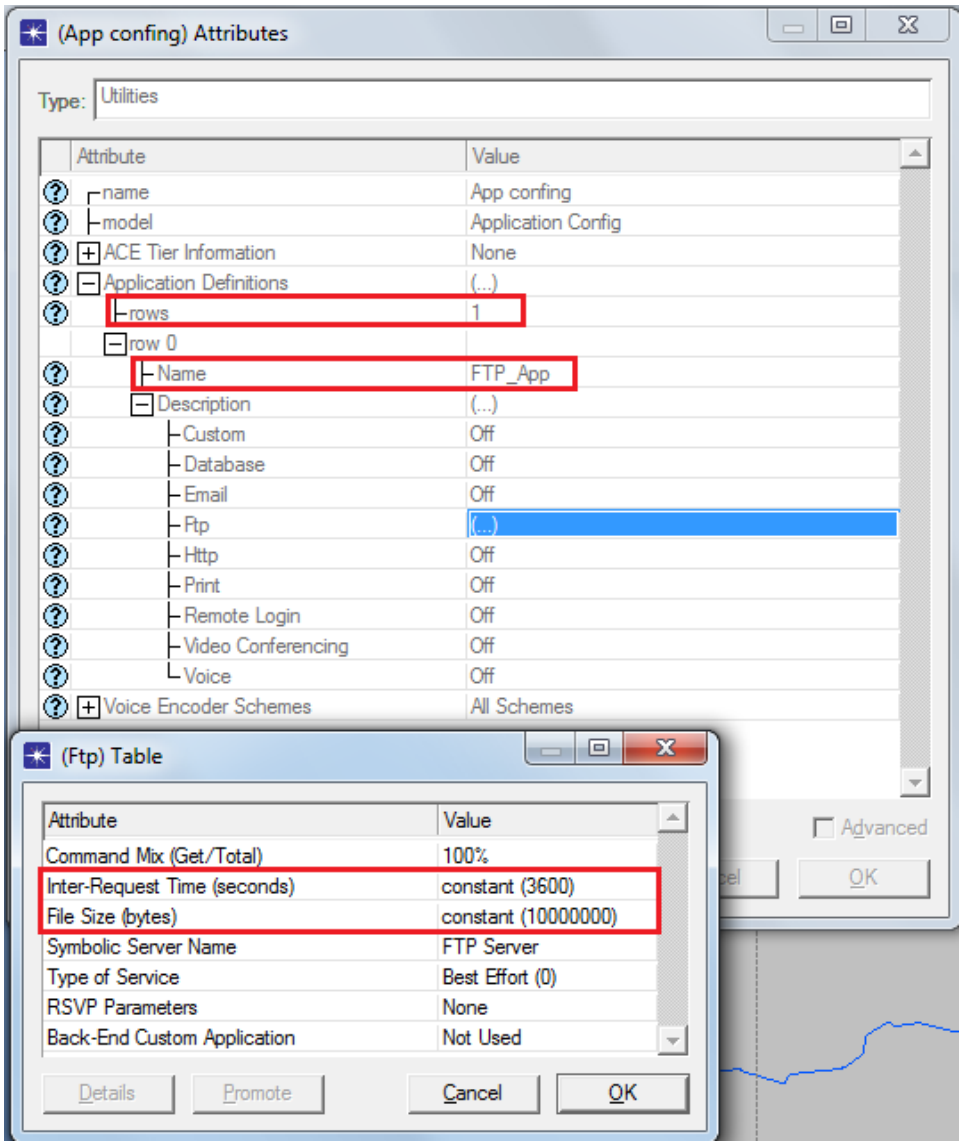
Ke směrovači Router_A byl připojen linkou **100BaseT** Server. Ke směrovači Router_B byla připojena stejnou linkou uživatelská stanice Client.



Obr. 6: Model sítě s FTP přenosem v programu IT Guru.

2.2.1 Nastavení aplikace FTP přenosu

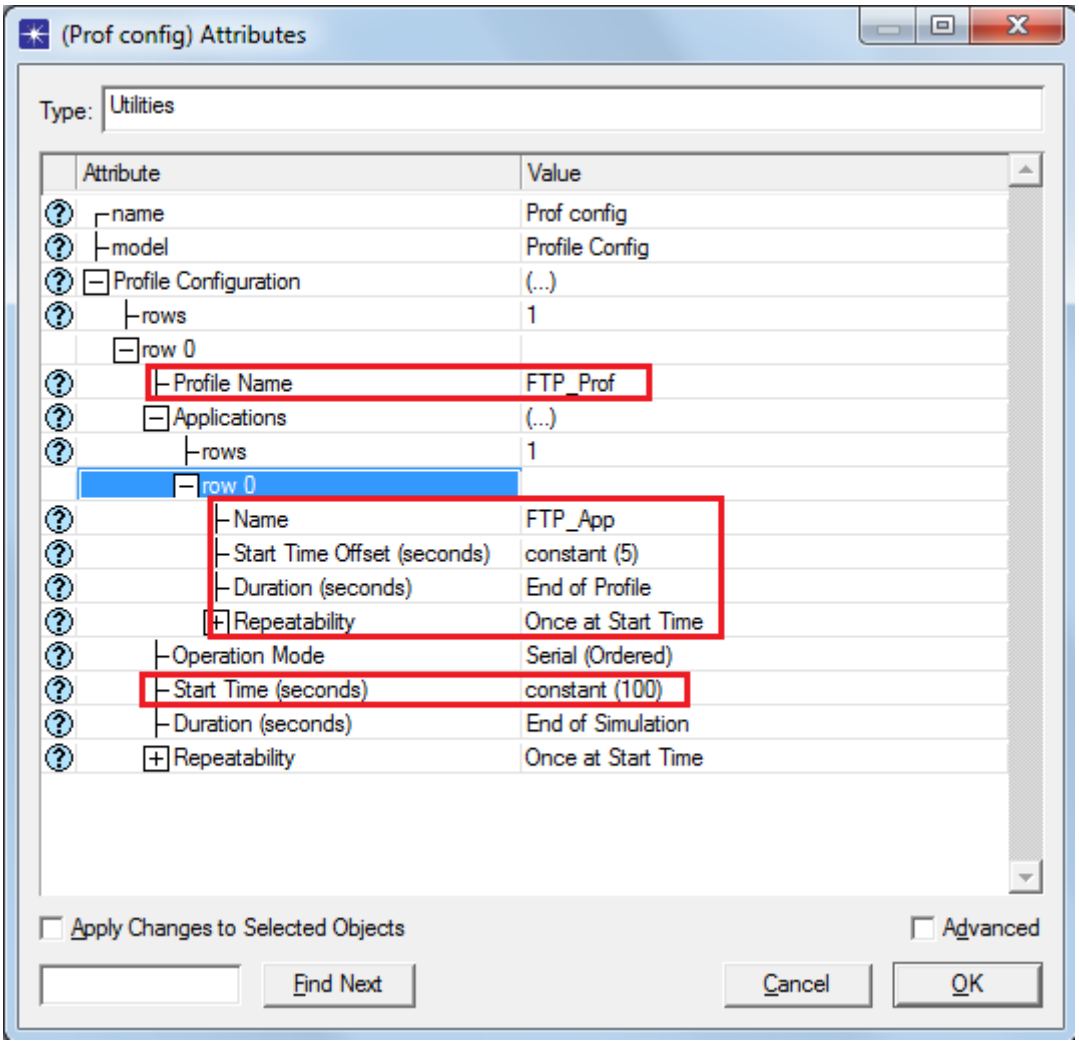
Po propojení všech prvků sítě bylo přistoupeno k nastavení provozované aplikace a profilu. Bylo vyvoláno kontextové menu objektu **Application Config** a zvoleno **Edit Attributes**. V nastavení modelu aplikace byla v položce **Application Definitions** nastavena hodnota **row** na **1**, čímž byla přidána nová aplikace. Ta byla pojmenována v položce **Name** jako **FTP_App**, aby bylo zjevné, že jde o aplikaci pro FTP přenos. Pak u položky **Descripton**, u řádku **FTP**, bylo zadáno **Edit**. V otevřeném okně pro editaci FTP byly nastaveny parametry FTP přenosu. Doba mezi jednotlivými požadavky **Inter-Request Time** byla nastavena na **3600 s**, což zajistí uskutečnění pouze jednoho požadavku během simulace a velikost přenášeného souboru dat **File Size** na **10 000 000 B** (viz Obr. 7). Další nastavitelné položky nebyly upravovány.



Obr. 7: Nastavení parametrů aplikace FTP přenosu.

2.2.2 Nastavení profilu FTP přenosu

Následně bylo přistoupeno k vytvoření profilu pro již vytvořenou aplikaci FTP_App. Bylo otevřeno kontextové menu objektu **Profile Config** a zvoleno **Edit Attributes → Profile Configuration**. Hodnota **rows** byla nastavena na **1** a do pole **Name** profilu bylo napsáno **FTP_Prof**. U pole **Application** byla opět nastavena hodnota pole **rows** na **1** a u pole **Name** této aplikace byla nastavena vytvořená aplikace **FTP_App**. **Start Time Offset** byl nastaven na **constant (5) s**, **Duration** na **End of Profile** a **Repeatabillity** na **Once at Start Time.** Čas spuštění profilu byl nastaven v položce **Start Time** nastavením hodnoty **constant (100) s** (viz Obr. 8). Ostatní položky byly ponechány nezměněné.

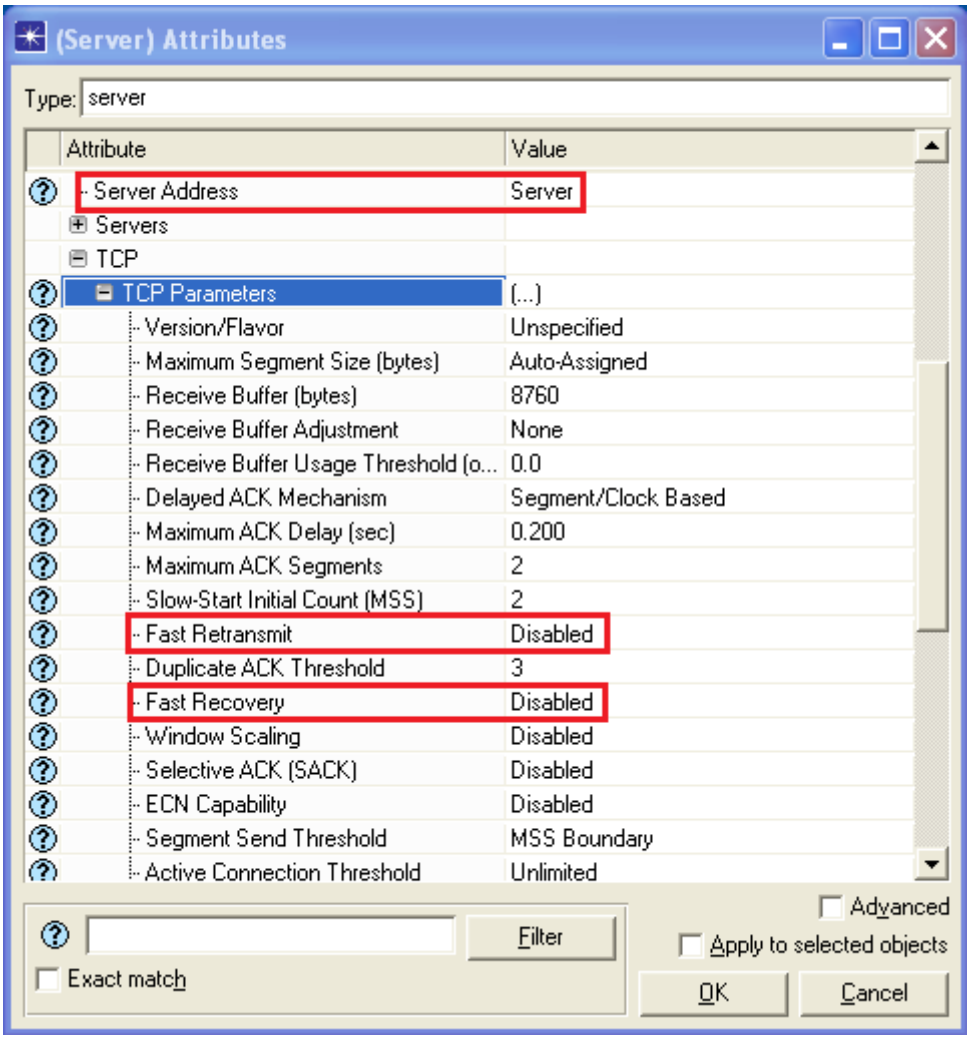


Obr. 8: Nastavení parametrů profilu FTP přenosu.

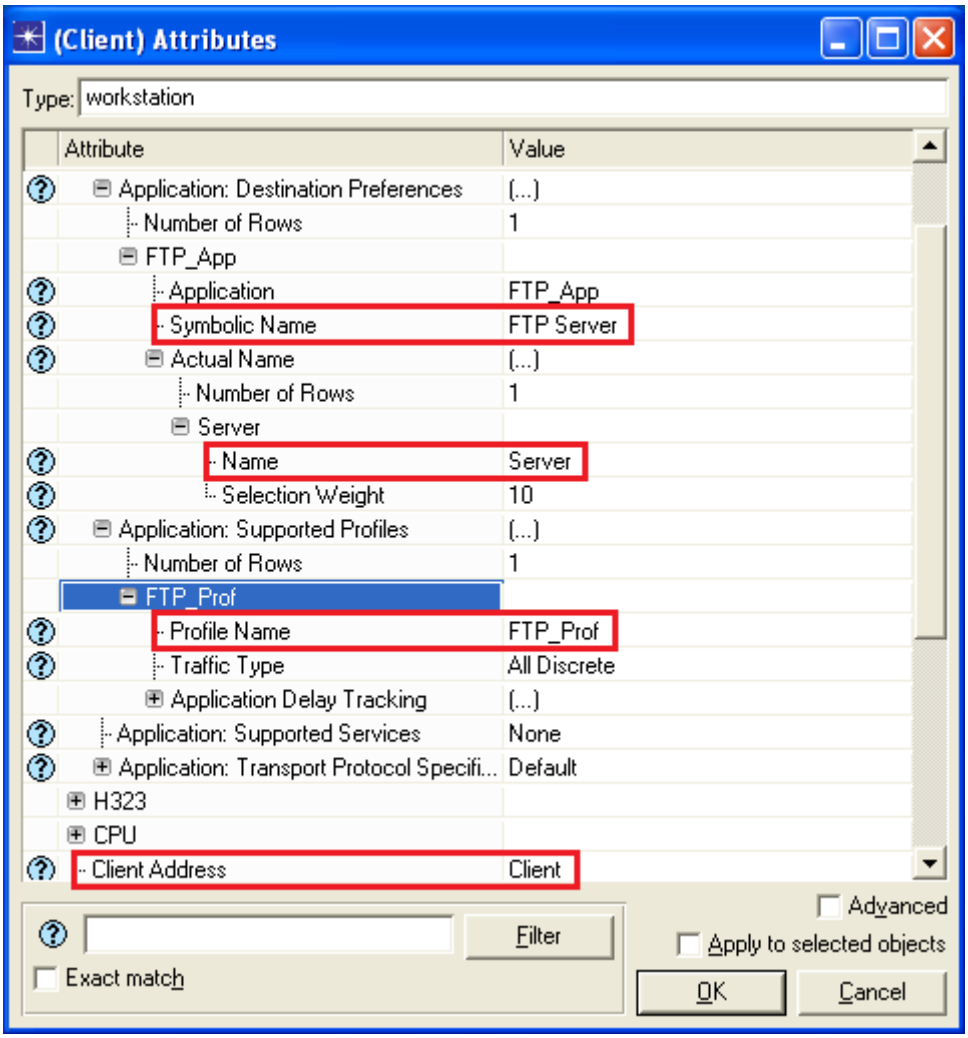
2.2.3 Nastavení serveru a klienta

Bylo otevřeno kontextové menu serveru a zvoleno **Edit Attributes → Application → Supported services → Edit…** a v nově otevřeném okně byla nastavena hodnota **row** na **1**. V kolonce pod **Name** byla zvolena služba **FTP_App**, která byla před tím vytvořena, a nastavení bylo potvrzeno stiskem **OK**. Tím byla na serveru zapnuta podpora služby pro FTP přenos. Dále bylo třeba nastavit položku **Server Addres**. Opět bylo zvoleno **Edit…** a do kolonky napsáno **Server**. Protože bude v tomto scénáři sledován přenos beze ztráty dat, mohly být vypnuty algoritmy **Fast retransmit** a **Fast Recovery**. To bylo provedeno v kolonce **TCP Parameters**. Kliknutím na křížek byla tato sekce otevřena a u výše zmíněných algoritmů zvoleno **Disabled**. Celé nastavení bylo potvrzeno stisknutím tlačítka **OK** (viz Obr. 9).

Nyní bylo přistoupeno k nastavení klienta. Opět bylo vyvoláno kontextové menu a zvoleno **Edit Attributes → Application → Supported Profiles → Edit…** a v nově otevřeném okně byla nastavena hodnota **row** na **1**. U položky **Profile name** bylo zvoleno **FTP_Prof** a nastavení potvrzeno stiskem **OK**. Adresa klienta byla nastavena podobně jako u serveru tak, že do kolonky **Client Addres** bylo zadáno **Client**. Ještě zbývalo nastavit cíl aplikace na FTP server. To bylo provedeno v kolonce **Application → Destination Preferences → Edit…** a hodnota **row** byla nastavena na **1**. V položce **Symbolik Name** bylo zvoleno **FTP Server → Actual Name**. V tomto okně bylo v položce **Name** zvoleno **Server**. Nastavení bylo potvrzeno kliknutím na **OK** u všech otevřených oken nastavení (viz Obr. 10). [2]



Obr. 9: Nastavení modelu Serveru.

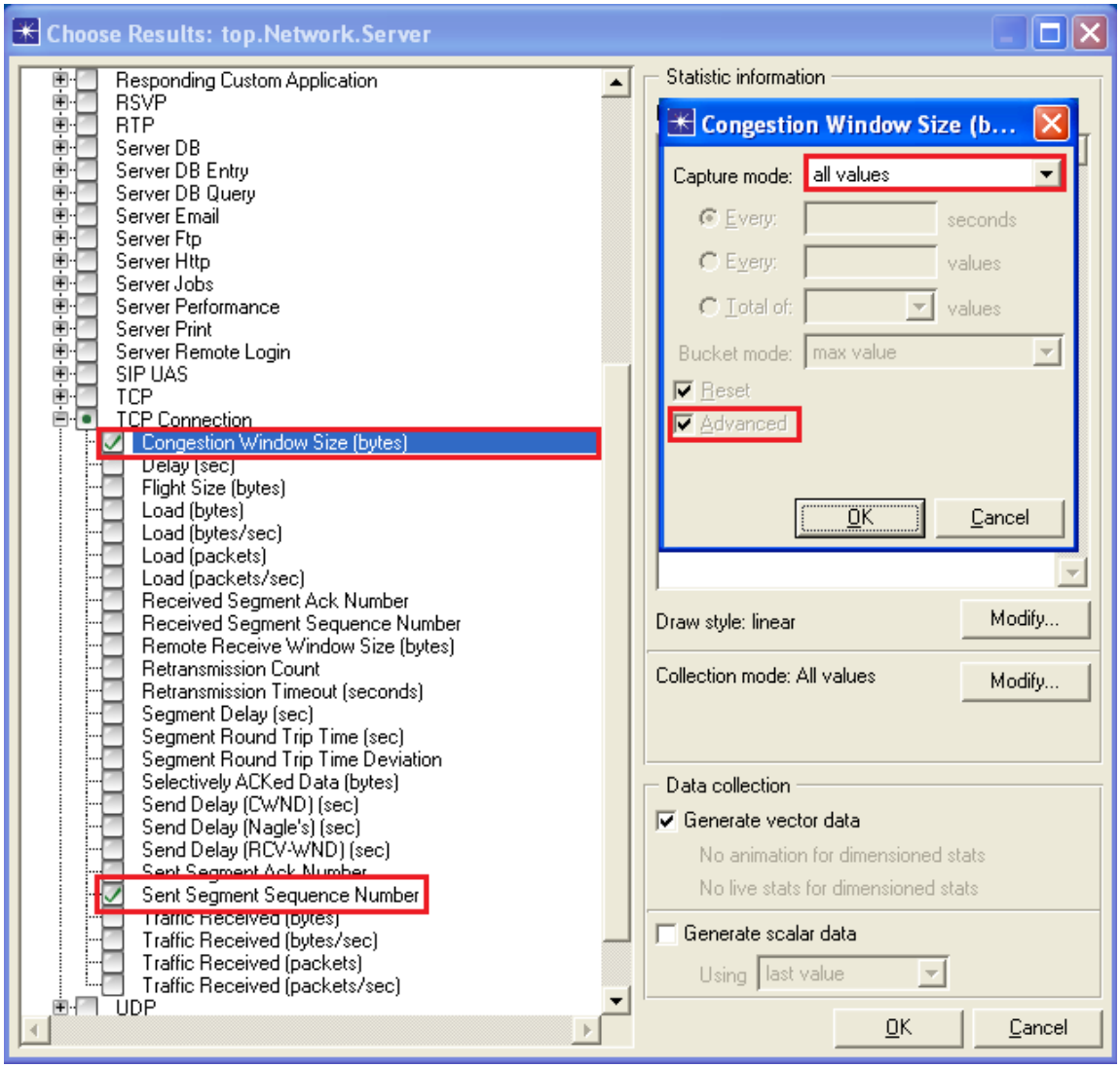


Obr. 10: Nastavení modelu Clienta.

3 Simulace algoritmů rozšíření Klouzavého okna

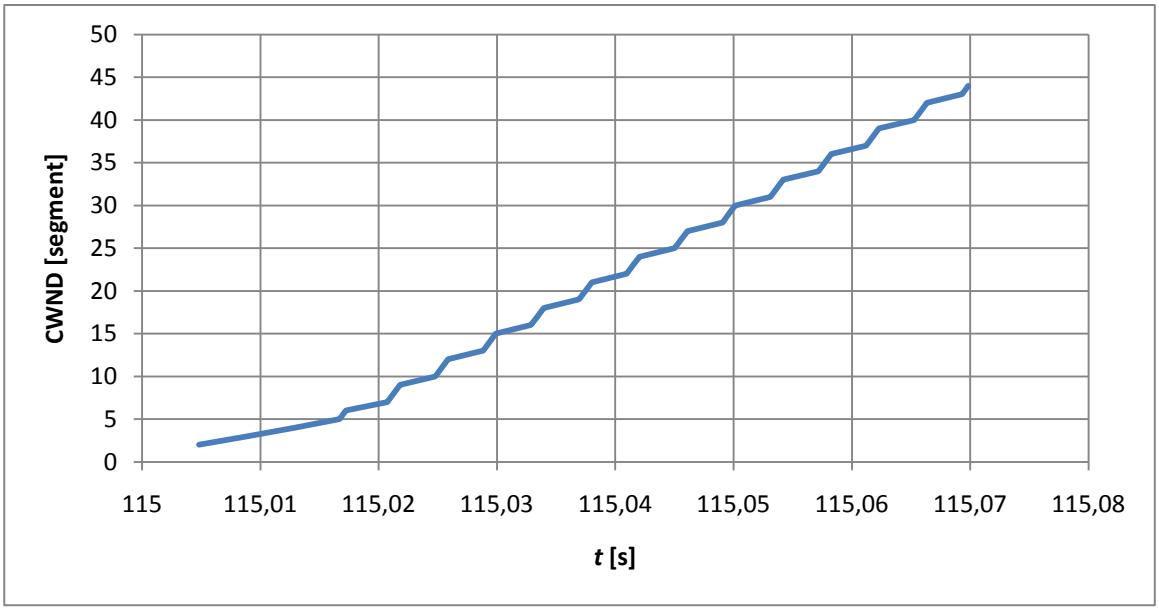
3.1 Simulace pomalého startu

Pro simulaci bylo nutné nastavit, jaké parametry modelu budou sledovány. V kontextovém menu Serveru bylo zvoleno **Choose Individual Statistics** a v otevřeném okně byla otevřena položka **TCP Conection**. Z podnabídky bylo vybráno sledování velikosti okna zahlcení CWND zatrhnutím položky **Congestion Window Size (bytes)** a sledování pořadového čísla odeslaných segmentů zatrhnutím položky **Sent Segment Sequence Number**. Ještě bylo nutné upravit styl sběru informací kliknutím pravým tlačítkem myši na tyto dvě položky a následně poklepáním na **Change colection Mode**. Pak bylo zatrženo **Advanced** a v položce **Capture Mode** zvoleno **all values** (viz Obr. 11). Nastavení bylo potvrzeno stiskem **OK**.



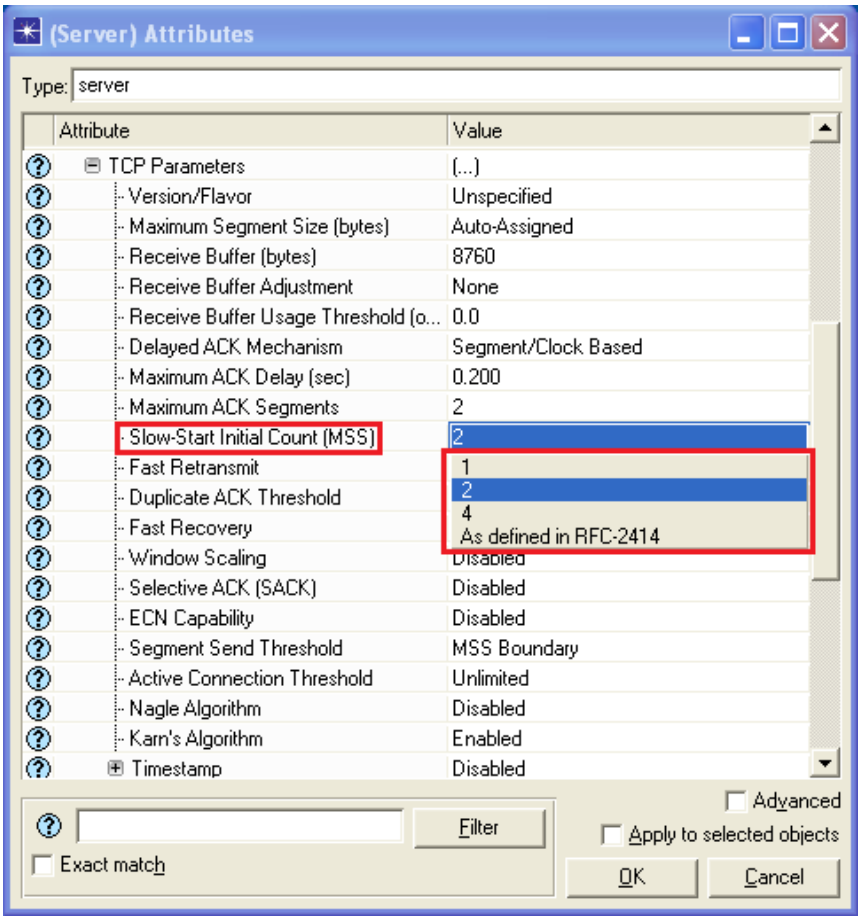
Obr. 11: Nastavení sledování zvolených parametrů.

Projekt byl uložen a v panelu nástrojů bylo kliknuto na tlačítko **Configure/run simulation**. V okně, které se otevřelo, byla nastavena položka délky simulace **Duration** na **10 min** a simulace byla spuštěna kliknutím na **Run**. Po skončení simulace bylo toto okno zavřeno a pokračovalo se zobrazením výsledků simulace. Na ploše scénáře bylo vyvoláno kontextové okno a zvoleno **View results**. V otevřeném okně bylo postupně rozkliknuto **Object Statistics → Network → Server → TCP Conection → Congestion Window Size (bytes)**. Výsledkem je níže zobrazený graf (viz Graf 1).

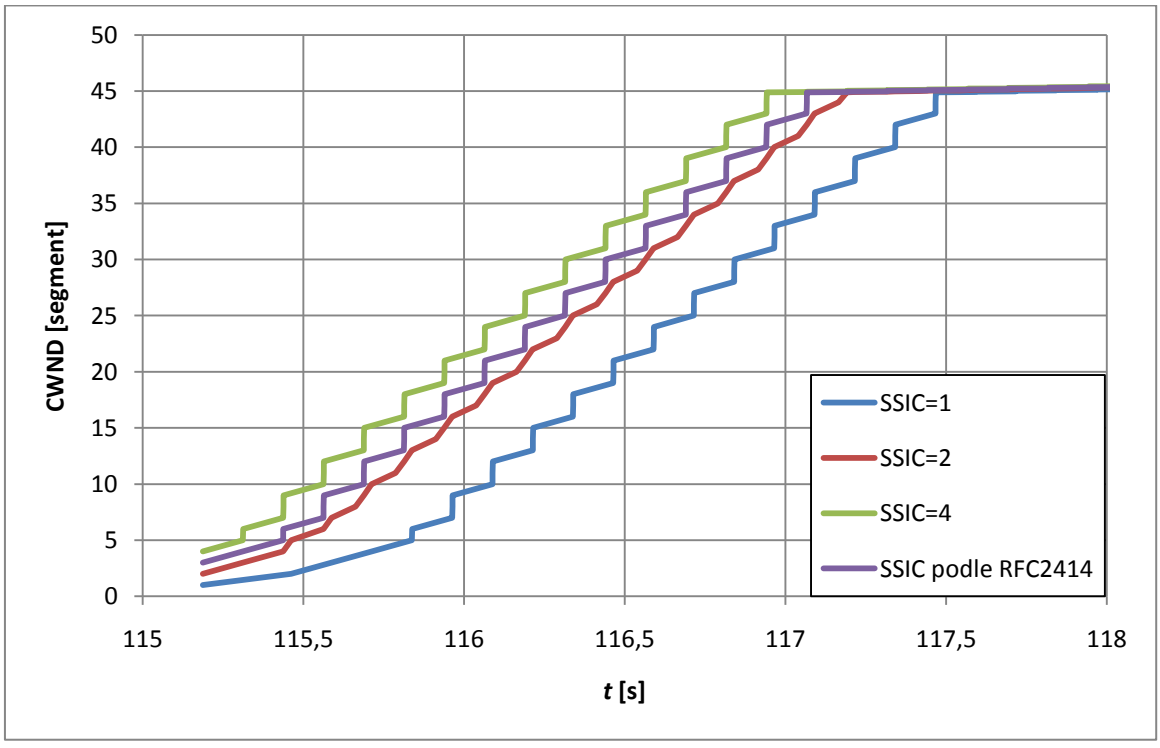


Graf 1: Ukázka algoritmu pomalého startu u simulovaného FTP přenosu.

Z grafu je možné pozorovat exponenciální nárůst velikosti CWND. Průběh je však zkreslen určitým zpožděním v síti. Také je možno vidět, že průběh začíná na hodnotě maximální velikosti jednoho segmentu. Když se zobrazí kontextové menu Serveru, **Edit Attributes → TCP parameters → Maximum Segment Size (bytes)**, dále jen **MSS**, zjistí se, že tato hodnota je nastavitelná. Nastavitelná je i počáteční hodnota CWND v položce **Slow-Start Initial Count** (SSIC) (viz Obr. 12). V již simulované síti byla tato hodnota nastavena na 1. Bylo zjištěno, že tuto hodnotu je možné nastavit na **1**, **2**, **4**, a podle standardu **RFC 2414**. Algoritmus standardu RFC 2414 nastavuje počáteční hodnotu CWND podle zvolené velikosti MSS mezi hodnotami 1 až 4. Tento algoritmus je blíže popsán v [1]. Pro zjištění vlivu tohoto nastavení, byl původní scénář 3× zkopírován pomocí **Scenario→Duplicate Scenario…**, a v každém scénáři byla nastavena jiná počáteční hodnota CWND. Nové scénáře byly uloženy, simulace provedeny a výsledky zobrazeny do grafu (viz Graf 2).



Obr. 12: Nastavení počáteční hodnoty algoritmu Slow-Start.

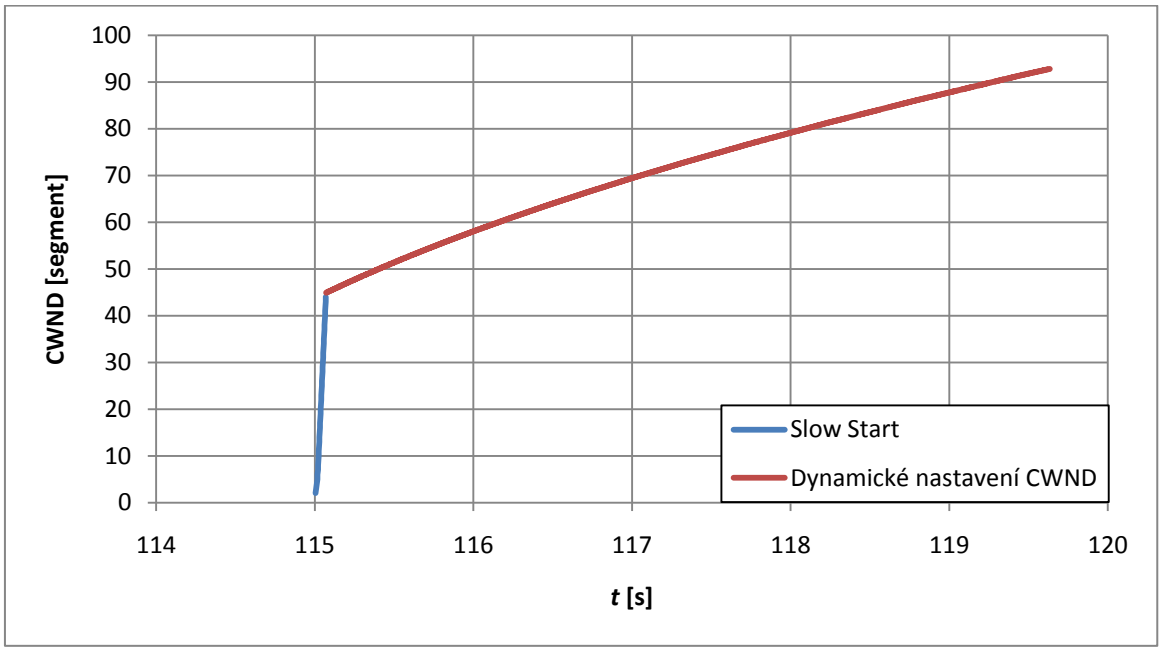


Graf 2: Porovnání různých počátečních velikostí CWND.

Z grafu je patrné různé nastavení počáteční velikosti CWND. Pokud bude graf přiblížen a zobrazen jen počáteční úsek, zjistí se, že odstupy jednotlivých průběhů jsou stejné a rovnají se maximální velikosti jednoho segmentu. Je možné si také povšimnout průběhu velikosti CWND s počátečním nastavením podle standardu RFC 2414. Je umístěn mezi průběhy pro počáteční velikost CWND 2 a 4. Algoritmus standardu RFC 2414 ji tedy nastavil na hodnotu 3. Je možné tedy usoudit, že počáteční velikost CWND celkově nemá na přenos zásadní vliv, protože odstupy jsou o velikosti maximální velikosti segmentu, tedy zanedbatelné.[1]

3.2 Simulace dynamického nastavení okna v případě zahlcení

Jelikož se v nastavení serveru nedá vypnout algoritmus dynamického nastavení okna, byla simulace provedena zároveň se simulací pomalého startu. Je tedy možné rovnou zobrazit výsledek (viz Graf 3).

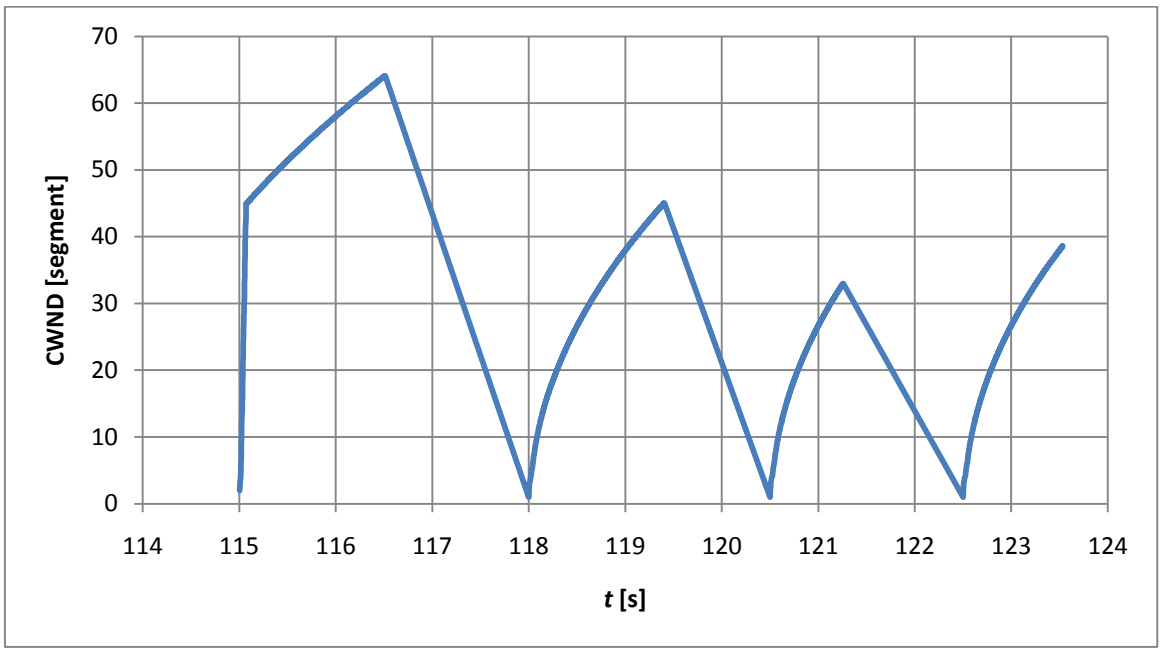


Graf 3: Dynamické nastavení CWND bez zahlcení.

Graf je možné rozdělit na dvě části, z nichž první část představuje algoritmus pomalého startu se svým exponenciálním vzrůstem a druhá představuje přechod na dynamické nastavení. V grafu je patrný jasný zlom mezi těmito dvěma algoritmy. Ke zlomu došlo ve chvíli, kdy hodnota CWND dosáhla 65 535 B, což je hodnota ssthresh po navázaném spojení. Aby bylo jasné, jak se tento algoritmus bude chovat v případě ztráty nějakého segmentu, je třeba v síti nastavit určitou hodnotu ztrátovosti.

Přidání ztrátovosti do sítě

Aby bylo možné později obě dvě situace porovnávat, bylo nutné nejprve původní scénář zkopírovat. To se provedlo v menu **Scenarios → Duplicate Scenario…** a nový scénář byl pojmenován **Drop_NoFast**. U objektu **Internet** bylo zvoleno **Edit Attributes → Packet Discard Ratio** a tato hodnota byla nastavena na **0,04 %** (viz Obr. 13). Nastavení bylo potvrzeno stiskem **OK** a scénář uložen. Pak byla znovu provedena simulace. Po skončení simulace byl opět zobrazen graf velikosti CWND (viz Graf 4).

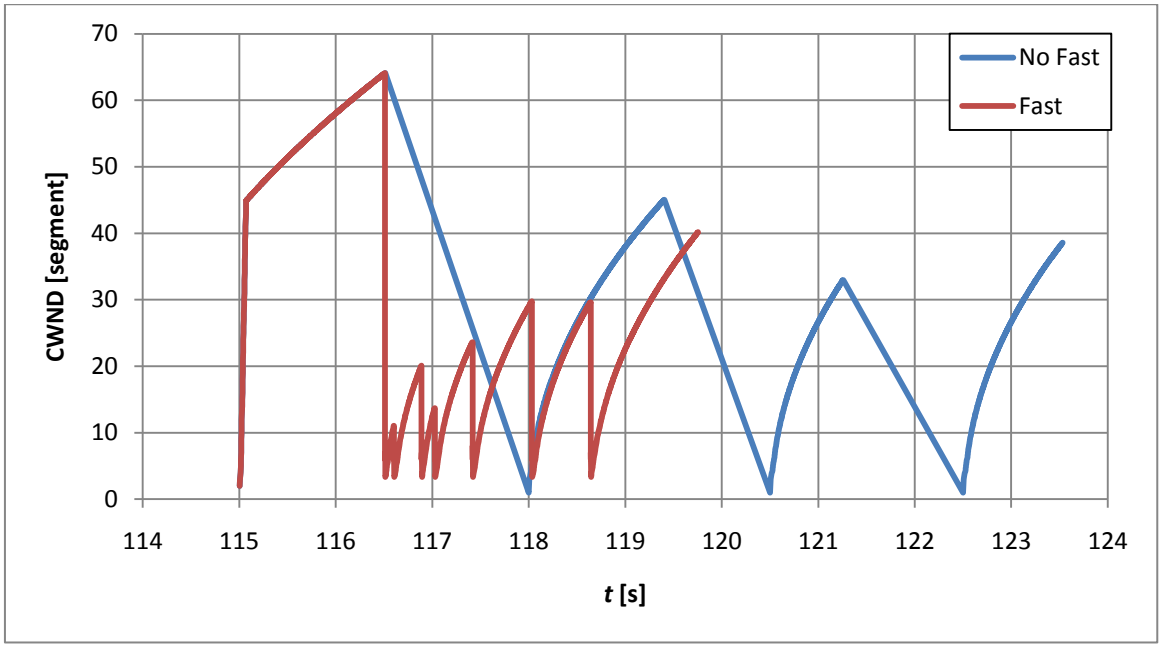


Graf 4: Dynamické nastavení CWND v případě ztráty.

V grafu je možné pozorovat střídající se stoupání a klesání velikosti CWND. Klesání je způsobeno ztrátou segmentu. Velikost CWND tedy klesne až na hodnotu jednoho segmentu a znovu se začíná pomalým startem.

3.3 Simulace algoritmů Rychlého opakování přenosu a rychlého zotavení

Pro simulaci těchto dvou algoritmů se vycházelo ze simulace scénáře Drop_NoFast. Tento scénář byl opět duplikován a nový scénář pojmenován **Drop_Fast**. Bylo otevřeno kontextové menu serveru, zvoleno **Edit Attributes → TCP Parameters**. U položky **Fast Recovery** bylo zvoleno **Reno** a u položky **Fast Retransmit** nastaveno **Enabled** (podobně jako u Obr. 9).



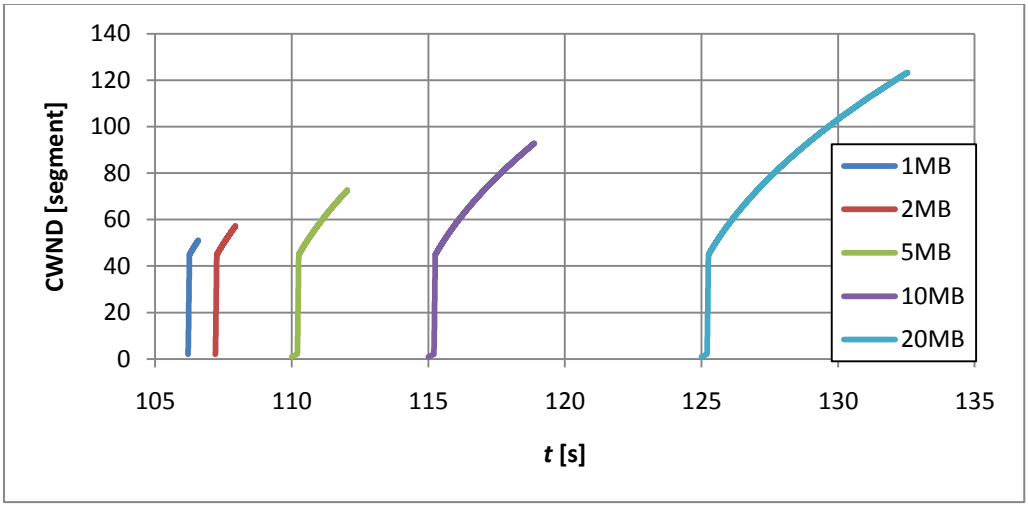
Graf 5: Velikost CWND u algoritmu rychlého znovu-odeslání a rychlého zotavení.

Pak byla již známým způsobem spuštěna simulace nového scénáře a po ukončení simulace byl zobrazen výsledek. Do grafu byl zobrazen i předchozí výsledek simulace, aby byl jasně vidět rozdíl, mezi těmito algoritmy (viz Graf 5). U průběhů pro rychlé zotavení je možné pozorovat okamžitý pokles. Hodnota CWND však neklesla na poloviční hodnotu, jak bylo předpokládáno. Pokud se průběh grafu přiblíží, je možné zjistit jednotlivé hodnoty špiček. Před ztrátou dat byla hodnota CWND (u průběhu s počáteční velikostí CWND = 1) rovna 99 090 B. Po ztrátě klesla na hodnotu 8 760 B a pak vzrostla na 11 680 B. Toto zvětšení odpovídá dvojnásobku MSS, tedy po ztrátě byly doručeny dva segmenty. Po znovuodeslání ztraceného segmentu se hodnota CWND nastavila na 4 866 B a odtud relativně lineárně rostla až do okamžiku další ztráty, kdy se celý proces opakoval naprosto stejně. Je zde tedy patrný určitý rozpor mezi teorií a praxí, který je možná způsoben určitým nastavením a naprogramováním modulu serveru.

3.4 Analýzy vlivu velikosti přenášených dat, zpoždění, a ztrátovosti na přenosovou rychlost

3.4.1 Analýza vlivu velikosti přenášených dat

Pro zkoumání vlivu velikosti přenášených dat se vycházelo ze scénáře bez ztráty dat s vypnutými algoritmy Fast Retransmit a Fast Recovery. Byly zvoleny tyto velikosti dat: **1 000 000 B**, **2 000 000 B**, **5 000 000 B**, **10 000 000 B**, a **20 000 000 B**. Scénář No_Drop byl **5**× zkopírován a nové scénáře pojmenovány podle velikostí dat, například No_Drop_1MB. V nových scénářích bylo vyvoláno kontextové menu objektu **Application Config → Edit Attributes → Application Definitions → FTP_App → FTP → Edit… → File Size (bytes)** a nastaveny zvolené velikosti. Všechna nastavení byla potvrzena stiskem **OK** všech otevřených editačních oken a scénáře byly uloženy. Doposud byly jednotlivé scénáře simulovány samostatně. Byla však možnost všechny scénáře simulovat současně. V menu aktuálního scénáře bylo zvoleno **Scenarios → Manage scenarios…**. Otevřelo se okno, ve kterém byly zobrazeny názvy všech scénářů v projektu. U scénářů No_Drop, Drop_NoFast, a Drop_Fast bylo zvoleno v kolonce **Results – Discard** a u nových scénářů zvoleno **Collect**. Stiskem tlačítka **Run** byla zahájena simulace pěti nových scénářů. Po skončení simulace byly zobrazeny její výsledky. Protože bylo simulováno více scénářů najednou, byla položka pod zobrazením grafu změněna z **This Scenario** na **All Scenarios**. Pak bylo zvoleno zobrazení velikosti CWND a v grafu se zobrazilo 5 průběhů. Nakonec bylo kliknuto na tlačítko Show pro zobrazení grafu v samostatném okně (viz Graf 6).



Graf 6: Vliv změny velikosti přenášených dat na CWND.

Pro získání přesných dat bylo vyvoláno kontextové menu grafu a z nabídky zvoleno **Export Graph Data to Spreadsheet**. Otevřel se nový sešit programu Excel. Z něj byly získány přesné doby začátků a konců jednotlivých přenosů. Z nich pak byly vypočítány délky přenosů a těmito délkami poděleny příslušné velikosti přenášených dat v bitech. Výsledky jsou uvedeny v tabulce (viz Tab. 1).

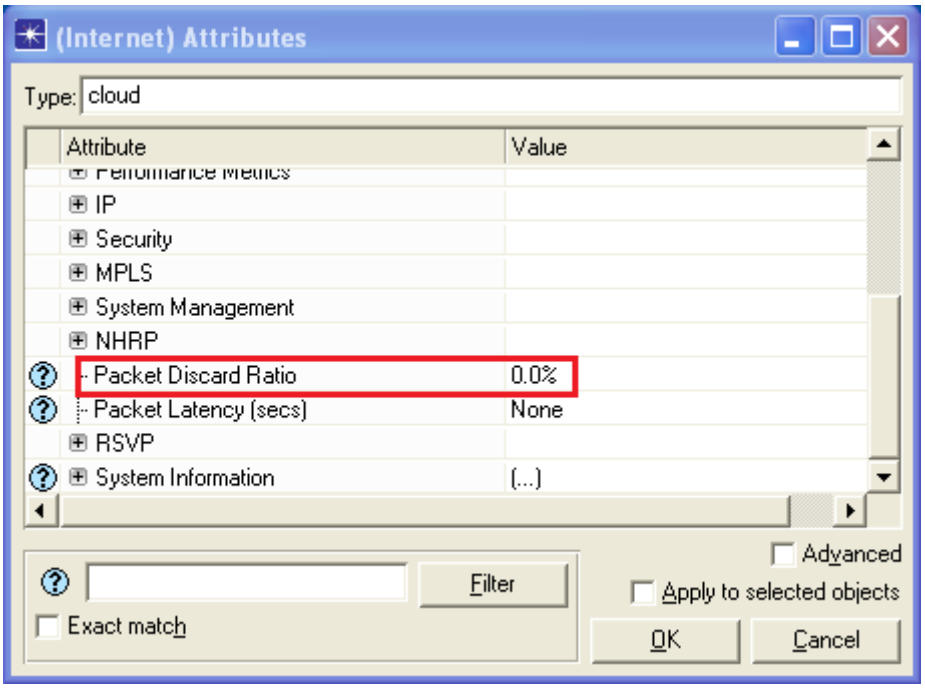
Tab. 1: Tabulka naměřených hodnot vlivu velikosti dat na přenosovou rychlost.

Vel. Dat	Čas	Prům. rychl.
B	s	Mb/s
1 000 000	0,446 8	17,075 6
2 000 000	0,889 4	17,156 3
5 000 000	2,223 4	17,157 0
10 000 000	4,438 7	17,188 4
20 000 000	8,867 1	17,208 3

Z výsledků tabulky je možné usoudit, že velikost přenášených dat nemá vliv na průměrnou přenosovou rychlost.

3.4.2 Analýza vlivu velikosti ztrátovosti

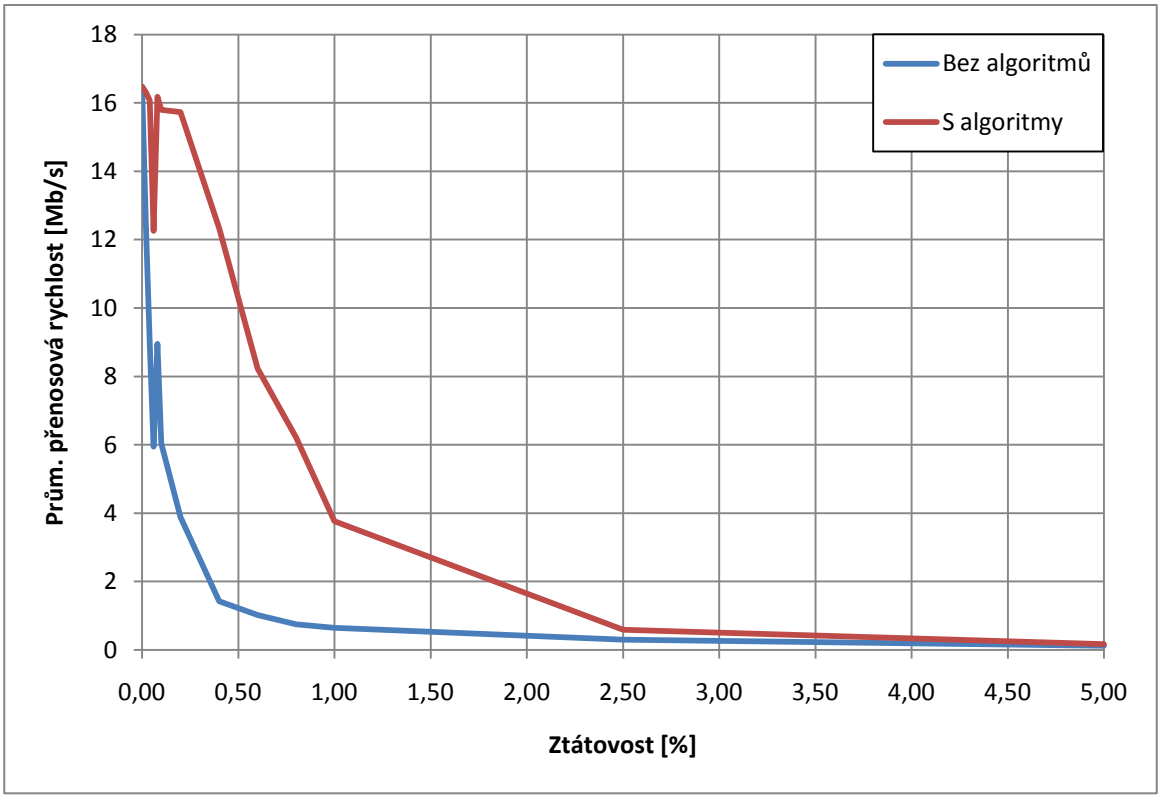
Pro zkoumání tohoto vlivu se vycházelo ze dvou scénářů a to ze scénáře Drop_NoFast, a z Drop_Fast. Účelem bylo zjistit, jaký vliv na přenosovou rychlost bude mít algoritmus Fast Retransmit a Fast Recovery. Byly voleny tyto hodnoty ztrátovosti: **0 %**; **0,02 %**; **0,04 %**; **0,06 %**; **0,08 %**; **0,1 %**; **0,2 %**; **0,4 %**; **0,6 %**; **0,8 %**; **1 %**; **2,5 %**; **5 %**. Oba dva scénáře byly **13×** zkopírovány, tedy pro každou hodnotu ztrátovosti, a pojmenovány například **NoFast_Lost1**. Ztrátovost byla nastavena stejně jako v prvním případě pro scénář Drop_NoFast u položky **Packet Discard Ratio** v editačním menu objektu Internet (viz Obr. 13). Opět byl každý scénář uložen a již známým způsobem bylo vyvoláno okno spuštění simulace z **Manage Scenarios…**. U starších scénářů bylo zvoleno **Discard,** u nových **Collect** v položce **Results** a byla spuštěna simulace. Po skončení simulace byly zobrazeny výsledky a opět získána naměřená data, ze kterých byla stejným způsobem jako u simulace různých velikostí přenášených dat vypočítána průměrná přenosová rychlost a hodnoty byly vyneseny do grafu (viz Tab. 2 a Graf 7).



Obr. 13: Nastavení ztrátovosti.

Tab. 2: Tabulka naměřených hodnot vlivu velikosti ztrátovosti na průměrnou přenosovou rychlost.

Bez algoritmů				S algoritmy			
Ztrátovost	**Čas**	**Vel. dat**	**Prům. rych.**	**Ztrátovost**	**Čas**	**Vel. dat**	**Prům. rych.**
%	**s**	**B**	**Mb/s**	**%**	**s**	**B**	**Mb/s**
0,00	4,630 4	10E+6	16,476 8	0,00	4,630 4	10E+6	16,476 8
0,02	6,141 6	10E+6	12,422 5	0,02	4,677 6	10E+6	16,310 5
0,04	8,526 2	10E+6	8,948 2	0,04	4,749 1	10E+6	16,064 9
0,06	12,828 8	10E+6	5,947 1	0,06	6,226 3	10E+6	12,253 5
0,08	8,525 5	10E+6	8,948 9	0,08	4,715 4	10E+6	16,179 7
0,10	12,637 7	10E+6	6,037 0	0,10	4,829 0	10E+6	15,799 1
0,20	19,643 7	10E+6	3,883 9	0,20	4,849 3	10E+6	15,733 0
0,40	53,522 0	10E+6	1,425 5	0,40	6,185 5	10E+6	12,334 3
0,60	74,594 0	10E+6	1,022 8	0,60	9,268 6	10E+6	8,231 4
0,80	101,534 4	10E+6	0,751 4	0,80	12,275 2	10E+6	6,215 3
1,00	118,053 6	10E+6	0,646 3	1,00	20,268 0	10E+6	3,764 3
2,50	253,545 0	10E+6	0,300 9	2,50	129,600 3	10E+6	0,588 7
5,00	617,999 0	10E+6	0,123 5	5,00	462,029 8	10E+6	0,165 1

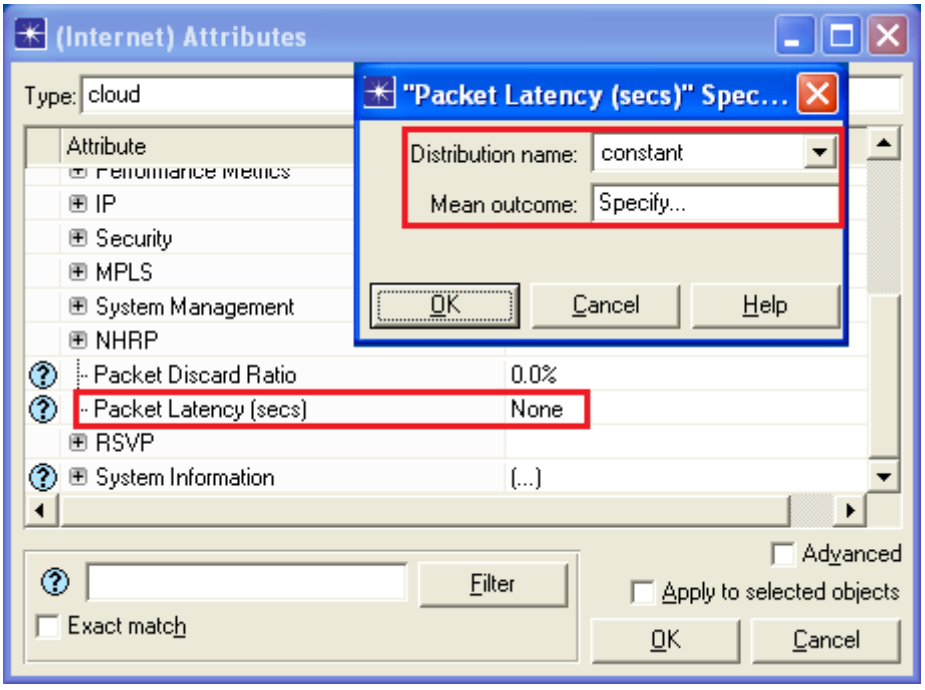


Graf 7: Vliv velikosti ztrát na přenosovou rychlost s a bez algoritmů Fast Retransmit a Fast recovery.

Z výsledného grafu bylo zjištěno, že na průměrnou přenosovou rychlost má vliv jak velikost ztrátovosti, tak i to, zda jsou použity algoritmy Rychlého odeslání a Rychlého opakování přenosu. Bez těchto algoritmů průměrná přenosová rychlost velmi strmě klesala v závislosti na velikosti ztrát. V druhém případě však docházelo nejprve k pomalejšímu poklesu a pak se průměrná přenosová rychlost začala zmenšovat rychlejším tempem. Použití algoritmů tedy poskytuje určitý prostor pro případné omezování přenosové rychlosti pomocí nastavování ztrátovosti na síťových uzlech v síti. Toto je možné i s vypnutými algoritmy, ovšem velmi strmý pokles, viditelný v grafu, nedává mnoho prostoru pro jemnější nastavení.

3.4.3 Analýza vlivu velikosti zpoždění v krátkém časovém úseku

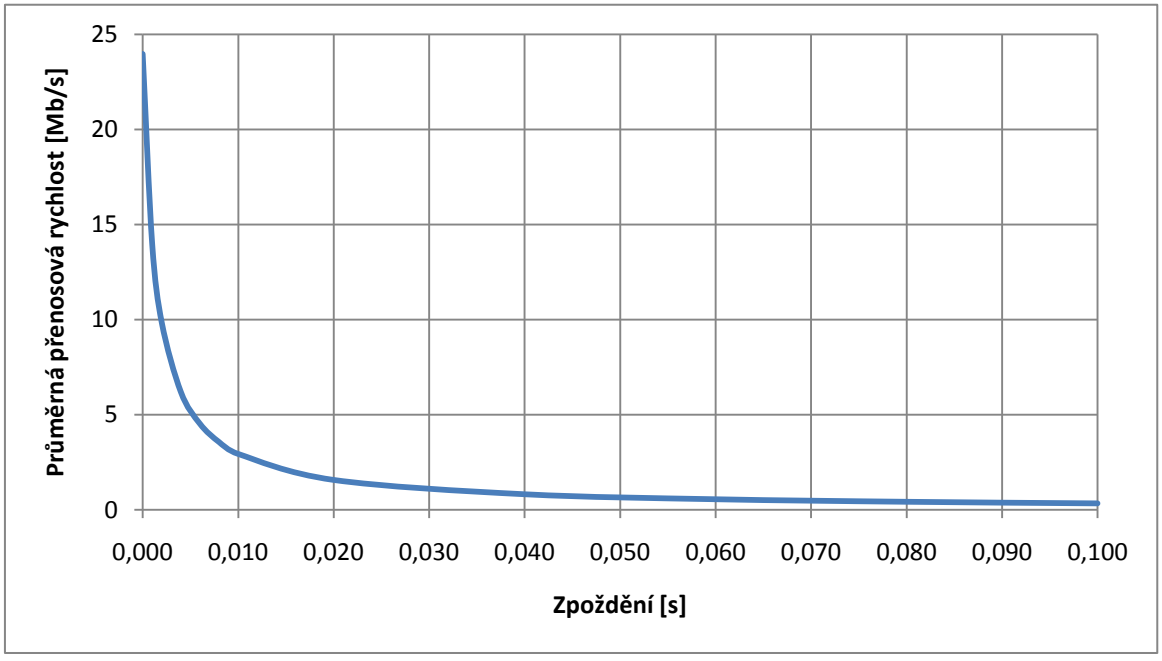
V tomto případě se opět vycházelo ze scénářů Drop_NoFast a Drop_Fast. Opět byl zájem sledovat vliv obou algoritmů Fast Retransmit a Fast Recovery na průměrnou přenosovou rychlost. Bylo zvoleno nastavení těchto hodnot zpoždění: **0 s**; **0,001 s**; **0,002 s**; **0,004 s; 0,006 s**; **0,008 s**; **0,01 s; 0,02 s**; **0,04 s**; **0,06 s**; **0,08 s**; **0,1 s**. Oba výchozí scénáře byly tedy **12×** zkopírovány a jednotlivé scénáře pojmenovány dle hodnoty zpoždění například Fast_Latency1. Zpoždění bylo nastaveno v kontextovém menu objektu Internet **Edit Attributes → Packet Latency (sec)**, kde bylo pole **Distribution name** nastaveno na **constant** a pole **Mean outcome** na **hodnotu zpoždění** (viz Obr. 14). Nastavení bylo potvrzeno odklepnutím **OK** a scénáře byly uloženy. Již známým způsobem byly vybrány požadované scénáře a byla spuštěna simulace. Stejným způsobem jako v minulém případě byly zobrazeny výsledky simulací a z dat vypočítány průměrné přenosové rychlosti (viz Tab. 3 a Graf 8).



Obr. 14: Nastavení zpoždění.

Tab. 3: Tabulka naměřených hodnot vlivu zpoždění na přenosovou rychlost.

Bez algoritmů				S agoritmy			
Zpoždění	Čas	Vel. dat	Prům. rych.	Zpoždění	Čas	Vel. dat	Prům. rych.
s	s	B	Mb/s	s	s	B	Mb/s
0,000	3,183 8	10E+6	23,963 5	0,000	3,183 8	10E+6	23,963 5
0,001	5,471 8	10E+6	13,943 2	0,001	5,471 8	10E+6	13,943 2
0,002	7,759 8	10E+6	9,832 0	0,002	7,759 8	10E+6	9,832 0
0,004	12,335 8	10E+6	6,184 8	0,004	12,335 8	10E+6	6,184 8
0,006	16,911 8	10E+6	4,511 3	0,006	16,911 8	10E+6	4,511 3
0,008	21,487 8	10E+6	3,550 6	0,008	21,487 8	10E+6	3,550 6
0,010	26,063 8	10E+6	2,927 2	0,010	26,063 8	10E+6	2,927 2
0,020	48,943 8	10E+6	1,558 8	0,020	48,943 8	10E+6	1,558 8
0,040	94,703 8	10E+6	0,805 6	0,040	94,703 8	10E+6	0,805 6
0,060	140,450 6	10E+6	0,543 2	0,060	140,450 6	10E+6	0,543 2
0,080	186,210 6	10E+6	0,409 7	0,080	186,210 6	10E+6	0,409 7
0,100	234,397 5	10E+6	0,325 5	0,100	234,397 5	10E+6	0,325 5



Graf 8: Zobrazení průměrné přenosové rychlosti v závislosti na hodnotě zpoždění.

Z této simulace bylo zjištěno, že algoritmy zde nemají žádný vliv. Během přenosu se totiž žádné segmenty neztratily a nebyl tedy důvod k jejich spuštění. V grafu je patrné, jak ovlivňuje velikost zpoždění průměrnou přenosovou rychlost. Nejdříve lineárně klesala, dále přešla do exponenciálního průběhu klesání a pak opět přešla do lineárního klesání. Úsek od 0–0,01 s zpoždění je vhodný pro ovlivňování přenosové rychlosti na síťových uzlech v síti.

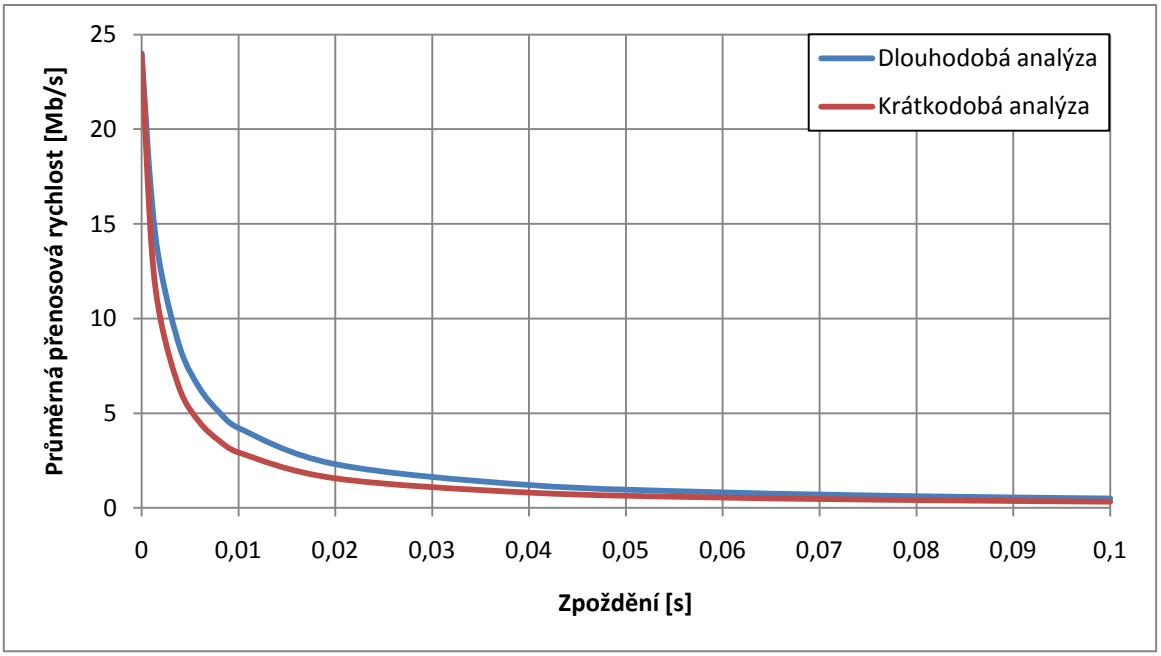
3.4.4 Analýza vlivu změny velikosti zpoždění na dlouhodobou průměrnou přenosovou rychlost

Ke zkoumání vlivu změny zpoždění v delším časovém úseku bylo použito již vytvořených scénářů pro zkoumání zpoždění v krátkém časovém úseku. Nastavení těchto scénářů bylo nutné poněkud upravit. Z důvodu delší doby přenosu byla známým způsobem změněna velikost přenášených dat z původních 10 000 000 B na **500 000 000 B** (viz Obr. 7). Dále bylo změněno nastavení zpoždění objektu Internet. Nastavení hodnoty **Distribution name** bylo změněno z constant na **uniform**. Hodnota položky **Minimum outcome** byla nastavena pro všechny scénáře na **0 s** a hodnota **Maximum outcome** na **hodnotu zpoždění** určenou pro daný scénář (podobně jako na Obr. 14). Nastavení scénářů bylo uloženo a jako poslední věc byla změněna délka simulací všech scénářů z 10 min na **1 hod**. Simulace byla spuštěna známým způsobem v kolonce **Manage Scenarios…**. Zde u upravených scénářů bylo v položce **Results** zvoleno **recolect** a byla změněna délka simulace. U ostatních scénářů v seznamu bylo zvoleno **discard**. Simulace byla spuštěna stiskem tlačítka **Run**. Po skončení simulace byly známým způsobem zobrazeny výsledky.

Pro určení velikosti průměrné přenosové rychlosti byly zobrazeny výsledky pro hodnotu odeslaného pořadového čísla – **Sent Segment Sequence Number**. Tato data byla známým způsobem vyexportována do programu Excel a z těchto dat byla vypočítána průměrná přenosová rychlost pro všechny scénáře (viz Tab. 4). Data byla vynesena do grafu (viz Graf 9).

Tab. 4: Vypočítané hodnoty průměrné přenosové rychlosti v delším časovém úseku.

Zpoždění	Čas	Vel. dat	Prům. rych.
s	s	B	Mb/s
0-0,000	158,877 1	500E+6	24,010 4
0-0,001	230,694 9	500E+6	16,535 7
0-0,002	304,948 8	500E+6	12,509 3
0-0,004	454,493 2	500E+6	8,393 3
0-0,006	604,332 8	500E+6	6,312 2
0-0,008	753,790 7	500E+6	5,060 7
0-0,010	903,679 1	500E+6	4,221 3
0-0,020	1 653,006 6	500E+6	2,307 7
0-0,040	3 078,935 0	500E+6	1,207 5
0-0,060	3 078,897 6	500E+6	0,819 6
0-0,080	3 078,868 6	500E+6	0,617 2
0-0,100	3 078,815 8	500E+6	0,495 7

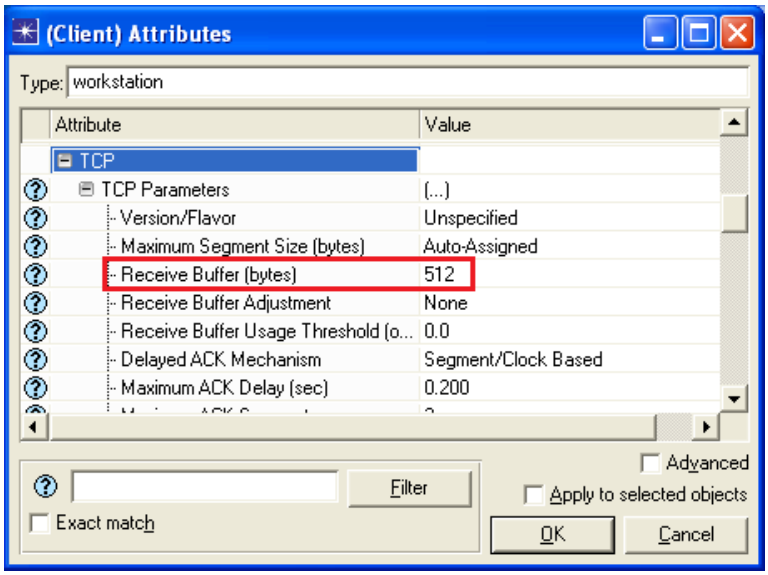


Graf 9: Vliv zpoždění na průměrnou přenosovou rychlost v delším a kratším časovém úseku.

Výsledná data byla velmi podobná datům, která byla naměřena pro zpoždění v krátkodobém úseku času. Z toho je možné usoudit, že zpoždění nemá na průměrnou přenosovou rychlost v delším časovém úseku výrazně jiný vliv než pro kratší časový úsek. Průběh poklesu průměrné přenosové rychlosti u delšího časového úseku byl poněkud pozvolnější. To je zřejmě způsobeno pohybem hodnoty zpoždění mezi minimální a maximální nastavenou hodnotou. Zde byly segmenty pozdrženy méně než u analýzy krátkodobého času. Stejně jako u předchozí analýzy i zde je možné regulovat průměrnou přenosovou rychlost v rozmezí 0–0,01 s.

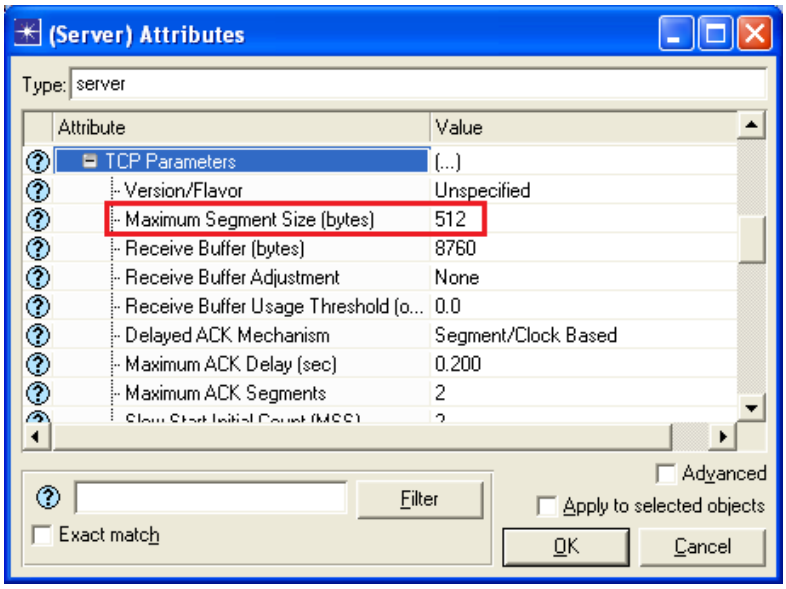
3.4.5 Analýza vlivu změny velikosti přijímací paměti a maximální velikosti segmentu

Pro tuto simulaci se vycházelo ze základního scénáře No_Drop. Bylo zkoumáno, jaký vliv na průměrnou přenosovou rychlost bude mít změna velikosti paměti přijímací stanice. Tento parametr byl nastavován v kontextovém menu objektu Client **Edit Attributes → TCP → TCP Parameters → Receive Buffer (bytes)** (viz Obr. 15). Byly voleny tyto hodnoty velikosti paměti: **512 B**, **1 024B**, **1 536 B**, **2 048 B**, **2 560 B**, **3 072 B**, **3 584 B**, **4 096 B**, **4 608 B**, **5 120 B**, **5 632B**, **6 144 B**, **6 656 B**, **7 168 B**, **7 680 B**, **8 192 B**, **9 216 B**, **10 240 B**, **11 264 B**, **12 288 B**, **13 312 B**, **14 336 B**, **15 360 B**, **16 384 B**, **18 432 B**, **20 480 B**, **22 528 B**, **24 576 B**, **26 624 B**, **28 672 B**, **30 720 B**, **32 768 B**, **36 864 B**, **40 960 B**, **45 056 B**, **49 152 B**, **53 248 B**, **57 344 B**, **61 440 B**, **65 536 B**.



Obr. 15: Nastavení velikosti přijímací paměti přijímače.

Byl vytvořen nový scénář duplikováním scénáře No_Drop a byl pojmenován Buffer_512. Byla nastavena velikost přijímací paměti a nastavení potvrzeno stiskem **Ok**. Další scénáře pro zbylé hodnoty byly vytvářeny z tohoto nového scénáře. Než tak bylo učiněno, musela se nastavit maximální velikost segmentu MSS na straně Serveru. Bylo to z toho důvodu, že program neumožní provést simulaci, pokud bude MSS odesílací stanice větší než velikost paměti přijímací stanice. V kontextovém menu Serveru **Edit Attributes → TCP → TCP Parameters →Maximum Segment Size (bytes)** byla nastavena hodnota **512** (viz Obr. 16). Bylo stisknuto **Ok** a scénář byl uložen. Pak byly vytvořeny zbylé scénáře a také byly uloženy.



Obr. 16: Nastavení MSS vysílací stanice.

Simulaci s velikostí přijímací paměti 512 B bylo nutné provést samostatně. Přenos trval déle než jednu hodinu. U této simulace se tedy musela nastavit délka simulace na **2** hodiny. U ostatních simulací byla nastavena hodnota délky simulace na **1** hodinu. Pak byla simulace spuštěna a výsledky zobrazeny. Opět byla vyexportována data, z nichž byly získány přesné hodnoty počátku a konce přenosu dat. Výsledky simulací byly uvedeny v Tab. 5–8 a zobrazeny v grafu (viz Graf 10).

Tab. 5: Naměřené hodnoty vlivu velikosti paměti na přenosovou rychlost, MSS = 512 B.

Vel. Paměti	Čas	Vel. Dat	Prům. rych.	Vel. Paměti	Čas	Vel. Dat	Prům. rych.
B	s	B	Mb/s	B	S	B	Mb/s
512	3 906,199 8	10E+6	0,019 5	13 312	2,290 3	10E+6	33,312 4
1 024	25,141 5	10E+6	3,034 6	14 336	2,149 4	10E+6	35,495 4
1 536	24,098 8	10E+6	3,165 9	15 360	2,140 1	10E+6	35,649 2
2 048	12,672 2	10E+6	6,020 6	16 384	2,140 1	10E+6	35,649 2
2 560	12,174 0	10E+6	6,267 0	18 432	2,140 1	10E+6	35,649 2
3 072	8,516 7	10E+6	8,958 2	20 480	2,140 1	10E+6	35,649 2
3 584	8,191 8	10E+6	9,313 4	22 528	2,140 1	10E+6	35,649 2
4 096	6,440 1	10E+6	11,846 7	24 576	2,140 1	10E+6	35,649 2
4 608	6,197 7	10E+6	12,310 0	26 624	2,140 1	10E+6	35,649 2
5 120	5,194 0	10E+6	14,688 9	28 672	2,140 1	10E+6	35,649 2
5 632	5,001 5	10E+6	15,254 2	30 720	2,140 1	10E+6	35,649 2
6 144	4,363 9	10E+6	17,483 1	32 768	2,140 1	10E+6	35,649 2
6 656	4,203 4	10E+6	18,150 3	36 864	2,140 1	10E+6	35,649 2
7 168	3,771 2	10E+6	20,230 8	40 960	2,140 1	10E+6	35,649 2
7 680	3,633 8	10E+6	20,995 7	45 056	2,140 1	10E+6	35,649 2
8 192	3,326 8	10E+6	22,933 4	49 152	2,140 1	10E+6	35,649 2
9 216	3,633 8	10E+6	20,995 7	53 248	2,140 1	10E+6	35,649 2
10 240	2,981 3	10E+6	25,590 4	57 344	2,140 1	10E+6	35,649 2
11 264	2,704 4	10E+6	28,210 7	61 440	2,140 1	10E+6	35,649 2
12 288	2,478 7	10E+6	30,780 3	65 536	2,140 1	10E+6	35,649 2

Dále bylo zkoumáno, jak ovlivní simulaci změna velikosti parametru maximální velikosti paměti (Maximum Segment Size - MSS). Jak bylo uvedeno v popisu nastavení předchozí simulace, hodnota MSS se nastavovala v kontextovém menu Serveru. Byly zvoleny další tři hodnoty MSS: **1 024 B**, **2 048 B**, a **4 096 B**. Tyto hodnoty byly postupně pro jednotlivé simulace nastaveny ve všech scénářích kromě scénářů, kde hodnota velikosti paměti přijímače byla menší, a zároveň byly tyto scénáře vyloučeny ze souboru scénářů pro simulaci zvolením **Discard** položky **Results** v nastavení **Scenarios → Manage Scenarios…**. Po dokončení všech nastavení byly provedeny simulace a výsledky zobrazeny.

Tab. 6: Naměřené hodnoty vlivu velikosti paměti na přenosovou rychlost, MSS = 1 024 B.

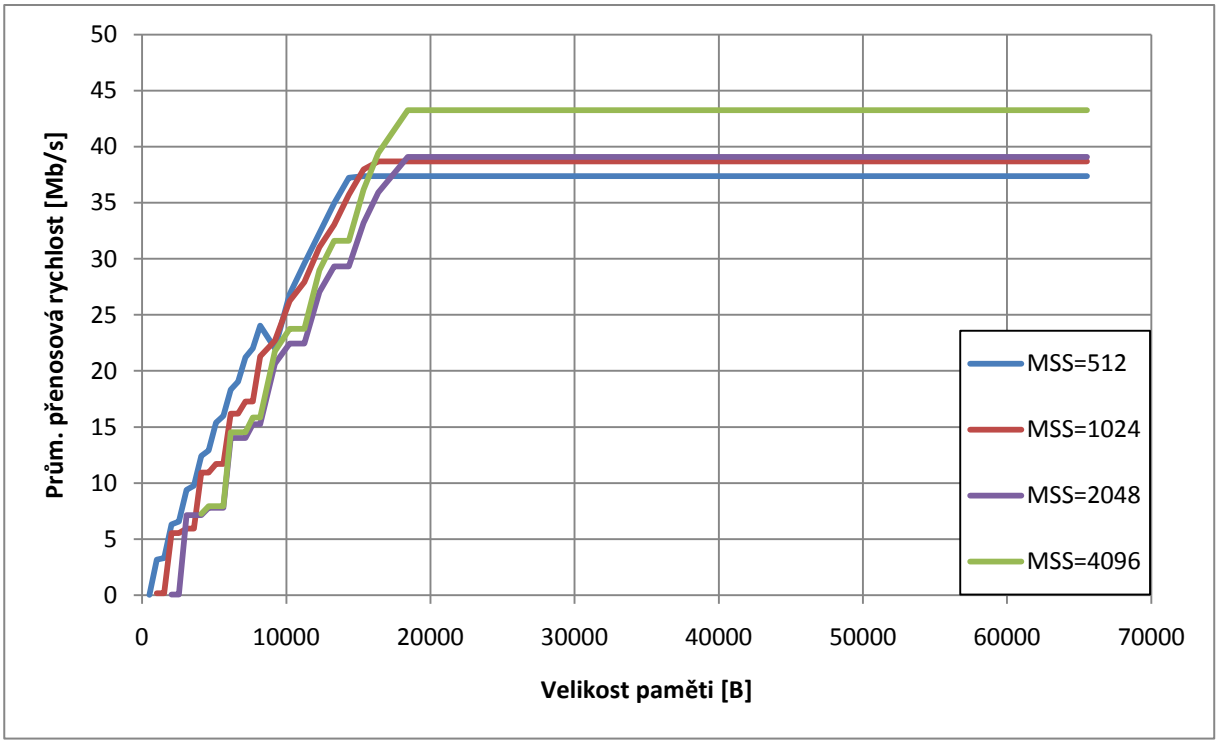
Vel. Paměti	Čas	Vel. Dat	Prům. rych.	Vel. Paměti	Čas	Vel. Dat	Prům. rych.
B	S	B	Mb/s	B	s	B	Mb/s
1 024	484,797 5	10E+6	0,157 4	14 336	2,240 0	10E+6	34,060 2
1 536	484,797 5	10E+6	0,157 4	15 360	2,107 8	10E+6	36,196 1
2 048	14,410 6	10E+6	5,294 3	16 384	2,068 4	10E+6	36,885 3
2 560	14,410 6	10E+6	5,294 3	18 432	2,068 4	10E+6	36,885 3
3 072	13,439 9	10E+6	5,676 7	20 480	2,068 4	10E+6	36,885 3
3 584	13,439 9	10E+6	5,676 7	22 528	2,068 4	10E+6	36,885 3
4 096	7,308 0	10E+6	10,439 7	24 576	2,068 4	10E+6	36,885 3
4 608	7,308 0	10E+6	10,439 7	26 624	2,068 4	10E+6	36,885 3
5 120	6,838 1	10E+6	11,157 2	28 672	2,068 4	10E+6	36,885 3
5 632	6,838 1	10E+6	11,157 2	30 720	2,068 4	10E+6	36,885 3
6 144	4,941 6	10E+6	15,439 2	32 768	2,068 4	10E+6	36,885 3
6 656	4,941 6	10E+6	15,439 2	36 864	2,068 4	10E+6	36,885 3
7 168	4,630 8	10E+6	16,475 3	40 960	2,068 4	10E+6	36,885 3
7 680	4,630 8	10E+6	16,475 3	45 056	2,068 4	10E+6	36,885 3
8 192	3,758 1	10E+6	20,301 1	49 152	2,068 4	10E+6	36,885 3
9 216	3,526 4	10E+6	21,635 3	53 248	2,068 4	10E+6	36,885 3
10 240	3,049 8	10E+6	25,015 7	57 344	2,068 4	10E+6	36,885 3
11 264	2,864 5	10E+6	26,634 2	61 440	2,068 4	10E+6	36,885 3
12 288	2,576 9	10E+6	29,606 8	65 536	2,068 4	10E+6	36,885 3
13 312	2,422 6	10E+6	31,492 7				

Tab. 7: Naměřené hodnoty vlivu velikosti paměti na přenosovou rychlost, MSS = 2 048 B.

Vel. Paměti	Čas	Vel. Dat	Prům. rych.	Vel. Paměti	Čas	Vel. Dat	Prům. rych.
B	s	B	Mb/s	B	s	B	Mb/s
2 048	1 369,600 6	10E+6	0,055 7	15 360	2,409 7	10E+6	31,661 5
2 560	1 369,600 6	10E+6	0,055 7	16 384	2,226 7	10E+6	34,262 6
3 072	11,206 3	10E+6	6,808 1	18 432	2,046 8	10E+6	37,274 4
3 584	11,203 9	10E+6	6,809 6	20 480	2,046 8	10E+6	37,274 4
4 096	11,203 9	10E+6	6,809 6	22 528	2,046 8	10E+6	37,274 4
4 608	10,257 1	10E+6	7,438 2	24 576	2,046 8	10E+6	37,274 4
5 120	10,257 1	10E+6	7,438 2	26 624	2,046 8	10E+6	37,274 4
5 632	10,257 1	10E+6	7,438 2	28 672	2,046 8	10E+6	37,274 4
6 144	5,706 1	10E+6	13,370 6	30 720	2,046 8	10E+6	37,274 4
6 656	5,704 2	10E+6	13,375 0	32 768	2,046 8	10E+6	37,274 4
7 168	5,704 2	10E+6	13,375 0	36 864	2,046 8	10E+6	37,274 4
7 680	5,243 9	10E+6	14,549 0	40 960	2,046 8	10E+6	37,274 4
8 192	5,243 9	10E+6	14,549 0	45 056	2,046 8	10E+6	37,274 4
9 216	3,873 8	10E+6	19,694 6	49 152	2,046 8	10E+6	37,274 4
10 240	3,567 3	10E+6	21,387 2	53 248	2,046 8	10E+6	37,274 4
11 264	3,567 3	10E+6	21,387 2	57 344	2,046 8	10E+6	37,274 4
12 288	2,958 5	10E+6	25,787 6	61 440	2,046 8	10E+6	37,274 4
13 312	2,729 5	10E+6	27,951 9	65 536	2,046 8	10E+6	37,274 4
14 336	2,729 5	10E+6	27,951 9				

Tab. 8: Naměřené hodnoty vlivu velikosti paměti na přenosovou rychlost, MSS = 4 096 B.

Vel. Paměti	Čas	Vel. Dat	Prům. rych.	Vel. Paměti	Čas	Vel. Dat	Prům. rych.
B	s	Mb	Mb/s	B	s	Mb	Mb/s
4 096	11,009 3	10E+6	6,929 9	18 432	1,849 6	10E+6	41,249 2
4 608	10,059 8	10E+6	7,584 0	20 480	1,849 6	10E+6	41,249 2
5 120	10,059 8	10E+6	7,584 0	22 528	1,849 6	10E+6	41,249 2
5 632	10,059 8	10E+6	7,584 0	24 576	1,849 6	10E+6	41,249 2
6 144	5,509 1	10E+6	13,848 6	26 624	1,849 6	10E+6	41,249 2
6 656	5,509 1	10E+6	13,848 6	28 672	1,849 6	10E+6	41,249 2
7 168	5,509 1	10E+6	13,848 6	30 720	1,849 6	10E+6	41,249 2
7 680	5,046 7	10E+6	15,117 6	32 768	1,849 6	10E+6	41,249 2
8 192	5,046 7	10E+6	15,117 6	36 864	1,849 6	10E+6	41,249 2
9 216	3,676 9	10E+6	20,749 5	40 960	1,849 6	10E+6	41,249 2
10 240	3,370 0	10E+6	22,638 9	45 056	1,849 6	10E+6	41,249 2
11 264	3,370 0	10E+6	22,638 9	49 152	1,849 6	10E+6	41,249 2
12 288	2,761 3	10E+6	27,629 6	53 248	1,849 6	10E+6	41,249 2
13 312	2,532 2	10E+6	30,129 3	57 344	1,849 6	10E+6	41,249 2
14 336	2,532 2	10E+6	30,129 3	61 440	1,849 6	10E+6	41,249 2
15 360	2,212 4	10E+6	34,484 0	65 536	1,849 6	10E+6	41,249 2
16 384	2,029 5	10E+6	37,592 4				



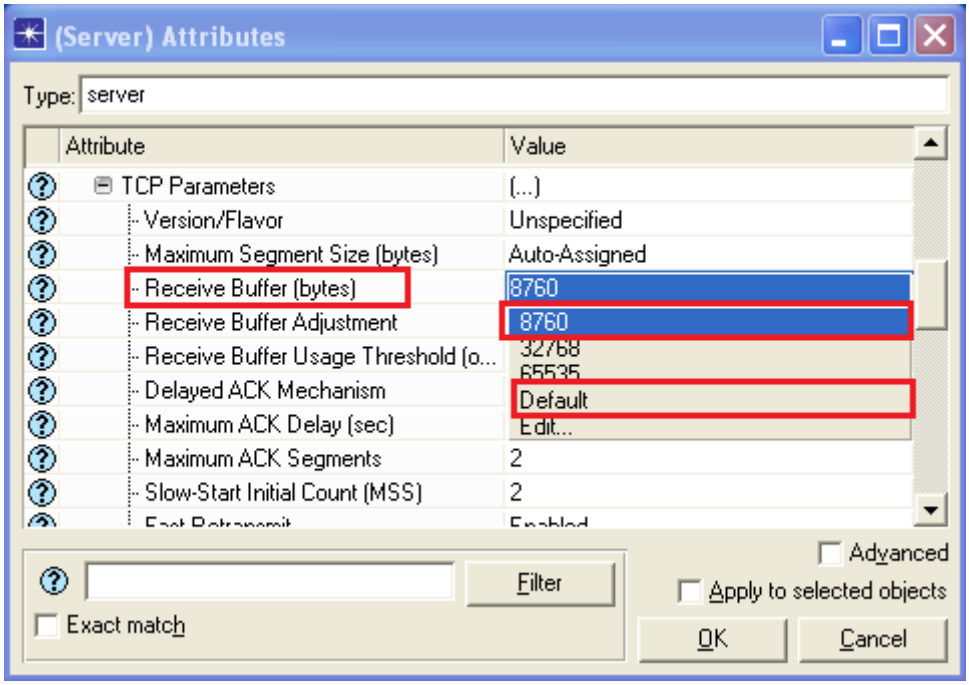
Graf 10: Vliv velikosti vyrovnávací paměti přijímače na průměrnou přenosovou rychlost.

Z naměřených dat a zobrazených výsledů lze vyvodit, že růst průměrné přenosové rychlosti připomíná lineární nárůst. Nejvíce hladký průběh měl soubor simulací, kde byla nastavena hodnota MSS na 512 B. Pro řízení přenosové rychlosti parametrem velikosti přijímací paměti na straně přijímače je nejvíce použitelný tento průběh. Také bylo možné pozorovat větší průměrné přenosové rychlosti pro nastavené velikosti paměti do 14 336 B, než jaké mají ostatní soubory simulací s nastavenými většími hodnotami MSS. Zřejmě je to způsobeno mechanismem odesílání segmentů, respektive dobou mezi odesláním dvou segmentů. Bylo zjištěno, že s rostoucí velikostí paměti přijímače se doba mezi odesláním dvou segmentů zmenšuje. Při porovnání různých velikostí MSS bylo možné vidět, že se hodnota průměrné přenosové rychlosti od určité velikosti paměti přijímače nemění. Zároveň bylo pozorováno zvyšování konečné velikosti této rychlosti zvýšením velikosti MSS.

4 Dynamická změna velikosti okna

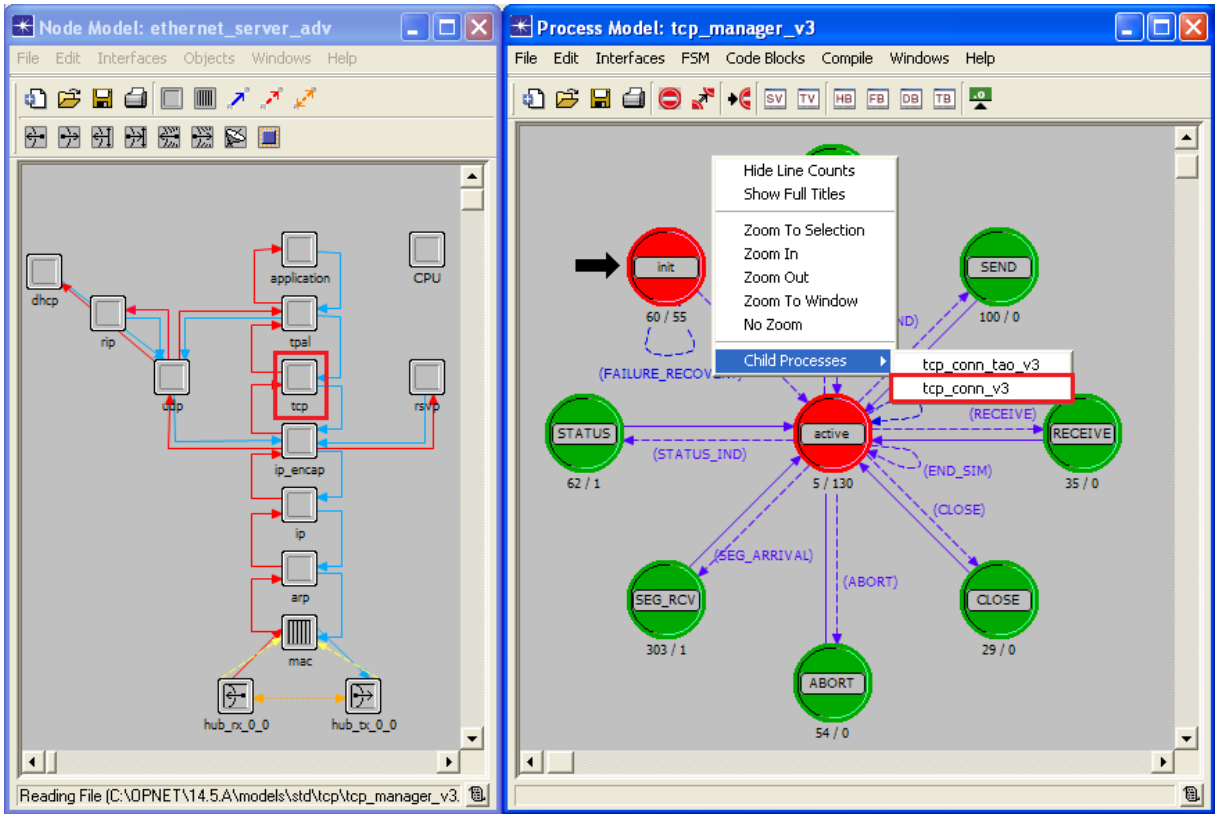
Dynamická změna znamená nastavení atributů přenosu tak, aby bylo možné reagovat na různé události v síti, a to hlavně na zahlcení uzlů. Cílem, kterým se bude zabývat tato část dokumentu, je tedy ovládání velikosti okna. Jak již bylo zmíněno v teoretické části, velikost okna vyjadřuje aktuální hodnotu volné paměti na přijímací straně. Jelikož program Opnet Modeler neumožňuje jakkoliv dynamicky nastavovat hodnoty oken či velikost volné paměti modelu v nástrojích pro nastavení, bude nutno provést zásah do kódu procesů, které mají na starost běh TCP přenosu. Na druhou stranu je však možno tyto hodnoty nastavovat staticky.

Na nastavení hodnot přijímací paměti je možno se podívat do kontextového menu Serveru, či Clienta. Zde v položce **Edit Attributes → TCP → TCP Parameters → Receive Buffer (Bytes)** je buď ručně nastavena konstantní hodnota velikosti paměti, nebo je nastavena na hodnotu **Default** (viz Obr. 17). Při nastavení Default se pak pomocí níže popsaného algoritmu vypočítá velikost automaticky.



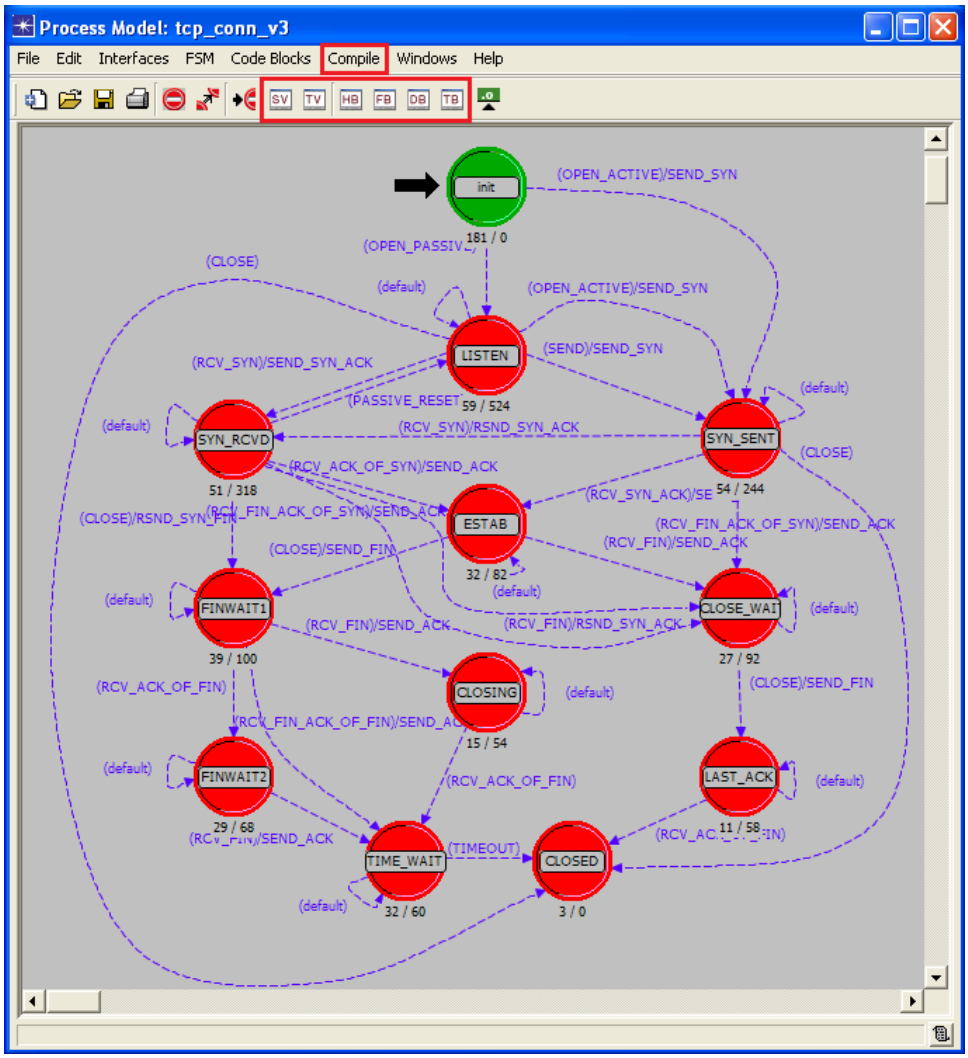
Obr. 17: Statické nastavení velikosti přijímací paměti.

Jelikož přenos dat probíhá od Serveru ke Clientovi, s velikostí paměti přijímací strany musí být seznámen Server, aby podle toho mohl měnit objem odesílaných dat. Dvojklikem tedy bylo vstoupeno do modelu uzlu (**Node model**) Serveru. Byl zobrazen model, jehož bloky znázorňují jednotlivé vrstvy modelu OSI. Opět dvojklikem na blok TCP bylo vstoupeno do procesního modelu vrstvy TCP, který je modelován stavovým automatem (viz Obr. 18). Tento automat se stará například o navázání a ukončení spojení, přijmutí a odeslání segmentu. Tento procesní model vlastní ještě tzv. Child processes. Do těchto procesů se lze dostat pravým klikem kamkoliv v okně procesního modelu a v otevřené nabídce zvolením Child Processes a vybráním konkrétního procesu. Ze zkoumání těchto procesů bylo zjištěno, že je nutné zvolit proces **tcp_conn_v3**. Bylo otevřeno okno zvoleného procesního modelu, ve kterém byl zobrazen další stavový automat (viz Obr. 19). Tento automat obsahuje například stavy pro naslouchání, synchronizaci při navazování spojení, ustanovení spojení a ukončení spojení. Dvojklikem na každý stav lze nahlédnout do kódu, který definuje jednotlivé úkony, které řídí komunikaci.



Obr. 18: Model Serveru a Procesní model vrstvy TCP.

Aby hledání místa, kde se definuje velikost paměti přijímací strany, nebylo ztíženo prohledáváním jednotlivých stavů, program Opnet Modeler poskytoval náhled do tzv. List Code. Tento soubor obsahuje veškerý kód, který je použit ve všech stavech a lze jej nalézt v menu **Compile → List Code**. Do tohoto souboru však nelze přímo zapisovat. Bylo tedy nutné pátrat dále. V nástrojové liště je několik tlačítek určených k editaci procesního modelu. Konkrétně tlačítko **Header Block (HB)**, **Function Block (FB)**, **Diagnostic Block (DB)**, a **Termination Block (TB)** (viz Obr. 19). Po kliknutí na tato tlačítka byla zobrazena editační okna s kódem procesu. Po bedlivějším prozkoumání bylo zjištěno, že se jedná o jednotlivé části kódu, který byl obsažený v List Code. V těchto blocích bylo možné provádět editaci. Editace se pak projevila i v List Code. V nástrojové liště dále zbyla dvě tlačítka. Bylo to tlačítko **State Variables (SV)** a **Temporary Variables (TV)**. Zde byly zapsány a popsány trvalé a dočasné proměnné, které proces používá.



Obr. 19: Potomek Procesního modelu TCP vrstvy tcp_conn_v3.

Nyní bylo potřeba identifikovat proměnnou, do které byla zapisována aktuální velikost paměti přijímací strany. Po prozkoumání **SV** a **TV** byla nalezena proměnná **rcv_buff**, která podle popisu definovala velikost paměti přijímací strany. Pak jen zbývalo tuto proměnnou najít v některém ze zbylých čtyř bloků. Byl použit nástroj pro vyhledávání textu **Ctrl+f** a do textového pole byl zadán název proměnné. Tato proměnná byla nalezena pouze v bloku FB. Jak bylo zjitěno z hledání, proměnná rcv_buff se v tomto bloku vyskytovala na mnoha místech. Ve většině případů byla použita jako zdroj hodnoty pro jinou proměnnou, nebo jako podmínka v cyklu IF. Bylo však nalezeno i několik míst, kde bylo do rcv_buff zapisováno. Šlo o procesy tcp_conn_sv_init, tcp_conn_mss_auto_assign, a tcp_syn_process. Z komentáře u těchto částí kódu bylo vyrozuměno, že se vskutku jednalo o stanovení velikosti paměti.

```
/* Set the receive window size -- it is the amount of          */
/* received data (in bytes) that can be buffered at one        */
/* time on a connection. The sending host can send only        */
/* that amount of data before waiting for an ACK and           */
/* window update from the receiving host. Check if the         */
/* receive buffer is set to be computed dynamically.           */
if (tcp_parameter_ptr->rcv_buff_size == -1)
        {
        /* When set to "Default", this parameter is set to at          */
        /* least four times the negotiated MSS, with a maximum  size of 64 KB.   */
        rcv_buff = (4 * snd_mss <= TCPC_MAX_WND_SIZE) ? 4 * snd_mss :
        TCPC_MAX_WND_SIZE;
        }
else
        {
        rcv_buff = tcp_parameter_ptr->rcv_buff_size;
        }
```

Tento kód byl použit při inicializaci TCP spojení. Ze zápisu kódu je vidět, že je zde zkoumáno, zda je v kontextovém menu zvolena specifická hodnota či nastavena hodnota Default. Hodnota Default je definována hodnotou **-1** a tato hodnota byla také použita v podmínce cyklu IF. Pro hodnotu Default je tu kód pro výpočet velikosti paměti, který vypočítá velikost paměti v rozsahu čtyřnásobku sjednané hodnoty MSS do maximální velikosti 64 kB. Pokud byla v kontextovém menu zadaná konkrétní hodnota, byla do proměnné rcv_buff pomocí ukazatele tato hodnota zapsána.

U zbylých dvou procesů byl zkoumán jen případ Default. Z toho plyne, že jediné místo, kde se velikost paměti nastavuje, je v procesu inicializace TCP spojení. Proces inicializace však proběhne pouze jednou za spojení a v dalších fázích přenosu se pracuje jen s vyjednanou hodnotou. Inicializační proces tedy nebyl vhodný pro umístění kódu, který by hodnotu v proměnné rcv_buff měnil během průběhu simulace.

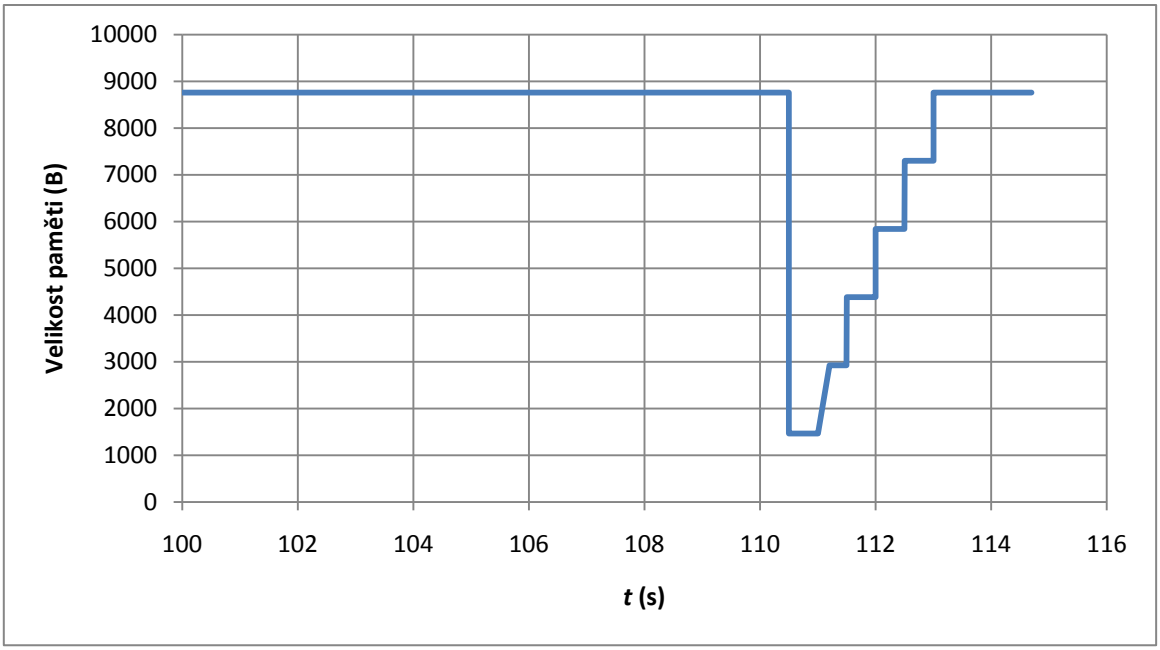
Z úvahy, které procesy se opakují během přenosu, bylo dosaženo názoru, že by vhodným procesem mohl být proces **tcp_seg_receive**. Tento proces byl volán pokaždé, když entita obdrží od protější strany segment s potvrzením ACK, ve kterém přiděluje vysílací straně kredity pro odeslání dat. V tomto procesu proměnná rcv_buff figurovala jen v několika podmínkách nebo jako zdroj hodnoty pro zápis do jiné proměnné. Byl tedy vhodný pro vložení kódu definující dynamicky velikost paměti přijímače.

Kód byl vložen téměř na začátek po deklaraci pomocných proměnných. Kód sestával z cyklu IF. Podmínkou byl dotaz na aktuální čas běžící simulace. Hodnota aktuálního času byla získána z procesu **op_sim_time()**. Pokud byla podmínka splněna, byl proveden příkaz v těle cyklu, který do proměnné rcv_buff zapsal novou hodnotu. Časy porovnávané s časem simulace bylo nutné volit uváženě a musely korespondovat s průběhem simulace bez zásahu do kódu procesu. V nastavení profilu FTP byl definován začátek přenosu dat v čase 100 sekund. Samotný přenos začal o 10 vteřin později a délka přenosu se pohybovala kolem 4 vteřin. Proto byl nastaven první čas na **111 s** a další hodnoty s odstupem **0,5 s**. Tento kód byl přiložen níže.

Po skončení úprav byl soubor uložen a v nástrojové liště procesního modelu tcp_conn_v3 bylo kliknuto na ikonku **kompilace**. Pokud vše proběhlo v pořádku a nebyla nalezena žádná chyba, mohla být spuštěna simulace. Před tím však ještě bylo známým způsobem zadáno sledování statistiky další proměnné a to proměnné **Remote Receive Window Size (Bytes)** ve statistikách Serveru a **Received Segment Sequence Number** ve statistikách Clienta.

Tyto statistiky se nachází v sekci **TCP Conection** a také bylo u nich nastaveno zaznamenávání všech hodnot. Po proběhnutí simulace byl zobrazen výsledek v grafu (viz Graf 11). V grafu je možné sledovat jasný pokles velikosti paměti přijímače ve stanovený čas a dále zvyšování této hodnoty v další stanovené časy.

```
static void
tcp_seg_receive (Packet* seg_ptr, TcpT_Flag flags, TcpT_Seg_Fields*pk_fd_ptr, TcpT_Seq
seg_seq, TcpT_Size seg_len)
	{
	Packet*				data_ptr;
	.
	TcpT_Size			rcv_buf_free;
	/** Process the newly received segment.  Put as much of the	**/
	/** data as is possible into the received data buffer.		**/
	FIN (tcp_seg_receive (seg_ptr, flags, pk_fd_ptr, seg_seq, data_len));
	if (op_sim_time () >= 115.5 )
		{
		rcv_buff = 1460;
		}
	if (op_sim_time () >= 116)
		{
		rcv_buff = 2920;
		}
	if (op_sim_time () >= 116.5)
		{
		rcv_buff = 4830;
		}
	if (op_sim_time () >= 117)
		{
		rcv_buff = 5840;
		}
	if (op_sim_time () >= 117.5)
		{
		rcv_buff = 7300;
		}
	if (op_sim_time () >= 118)
		{
		rcv_buff = tcp_parameter_ptr->rcv_buff_size;
		}
```

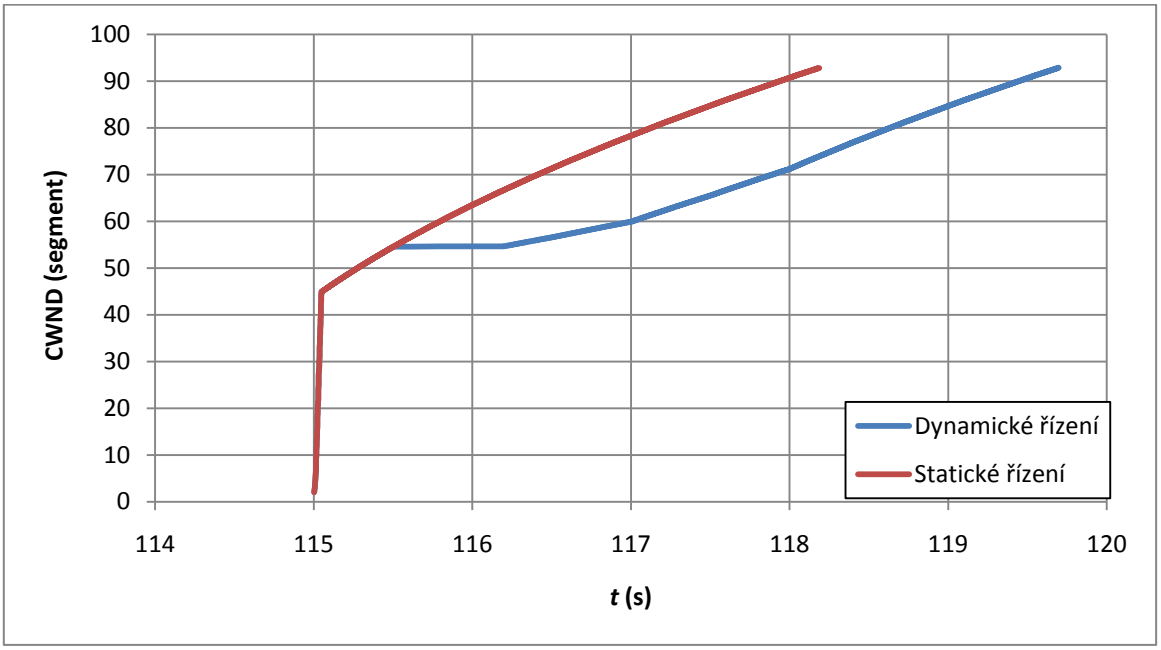


Graf 11: Průběh velikost paměti přijímací strany.

Statistika Received Segment Sequence Number byla nastavena z důvodu sledování aktuální přenosové rychlosti. Byly zobrazeny grafy pro odeslané číslo segmentu na straně Serveru a přijaté číslo segmentu na přijímací straně. Hodnoty byly vyexportovány do programu Excel. V každém časovém úseku, kdy je nastavena určitá hodnota volné paměti, byl zvolen určitý objem dat. Téměř ve všech případech byl volen objem 8 760 B. Pouze u nastavení velikosti volné paměti 1 460 B byl zvolen objem 5 840 B, protože se s tímto nastavením volné paměti víc dat nepřeneslo. U těchto zvolených dat byl odečten čas odeslání prvního segmentu a čas přijetí posledního segmentu. Vzorec pro výpočet přenosové rychlosti je: $R = \frac{X}{t_T - t_R}\ [Mb/s]$, kde X je vybraný objem dat převedený na Mb, t_T [s] je čas odeslání prvního segmentu zvoleného objemu dat ze Serveru a t_R [s] je čas přijmutí posledního segmentu na straně přijímače. Vypočítané hodnoty přenosové rychloti byly pro každou nastavenou hodnotu paměti zobrazeny v tabulce (viz Tab. 9).

Tab. 9: Velikost přenosové rychlosti.

Vel. Paměti	t	Vel. Dat	Přenos. Rychlost
B	s	B	Mb/s
8 760	0,004 1	8 760	16,111 1
1 460	0,498 9	5 840	0,089 3
2 920	0,007 5	8 760	8,946 1
4 380	0,009 2	8 760	7,283 4
5 840	0,006 7	8 760	10,031 7
7 300	0,004 7	8 760	14,258 5
8 760	0,004 4	8 760	15,128 3



Graf 12: Srovnání průběhu velikosti CWND dynamického a statického řízení paměti přijímače.

Jako poslední byl zobrazen průběh CWND. Je zde vidět jasná změna průběhu. Do zvolené hodnoty času simulace 110,5 s vše probíhá standardně. Jakmile však dojde ke změně velikosti okna, dojde k výraznému zpomalení odesílání. Je to způsobeno tím, že Server nyní odešle menší objem dat, než který očekává Client. Client čeká na data od Serveru určitou dobu. Poté přijatá data potvrdí a přidělí Serveru další kredity pro odesílání. S postupným zvětšováním nastavené hodnoty paměti přijímače se také zmenšuje doba potvrzování přijatých dat a data se začnou odesílat rychleji (viz Graf 12).

5 Závěr

V první části tohoto dokumentu byl popsán algoritmus Klouzavého okna. Následně byly popsány rozšiřující algoritmy Klouzavého okna a to algoritmus Pomalého startu, Dynamického nastavení v případě zahlcení, Rychlého zotavení a Rychlého opakování přenosu. Tyto algoritmy řeší nevýhodu Klouzavého okna, což je možnost rychlého zahlcení síťového uzlu posílanými segmenty.

V druhé části tohoto dokumentu byl popsán postup vytvoření simulace přenosu souboru z FTP serveru ke klientovi. To bylo prováděno v programu OPNET IT Guru Academic edition 9.1. Grafy, získané exportem naměřených dat do programu Excel, byly z důvodu lepší přehlednosti vytvořeny jako spojnicové, i když vycházejí z diskrétních hodnot, z důvodu lepší přehlednosti. V další části dokumentu bylo v simulacích ověřováno chování rozšiřujících algoritmů Klouzavého okna. Přitom bylo zjištěno, že u algoritmu Rychlého opakování přenosu nedochází při ztrátě segmentu ke snížení aktuální velikosti CWND na polovinu. Všechny ostatní algoritmy se chovaly dle očekávání, jak bylo uvedeno v první části dokumentu. Pro ověření byly simulace provedeny i v programu Opnet Modeler, při čemž se objevila drobná chyba v programu IT Guru, při simulaci nastavování různých velikostí SSIC u algoritmu Pomalého startu.

V další části tohoto dokumentu byly provedeny analýzy vlivu nastavení různých velikostí dat, ztrátovosti a zpoždění na průměrnou přenosovou rychlost. Vliv zpoždění byl zkoumán v krátkém i delším časovém úseku. Dále byl analyzován vliv změny velikosti přijímací paměti a maximální velikosti segmentu. Ze simulací různých velikostí přenášených dat bylo zjištěno, že na průměrnou přenosovou rychlost nemají žádný vliv. U simulace ztrátovosti bylo zjištěno, že na průměrnou přenosovou rychlost má vliv. Dále byly zjištěny dopady zapnutí algoritmů Rychlého odeslání a Rychlého opakování přenosu. Bez zapnutých algoritmů měla průměrná přenosová rychlost v závislosti na velikosti zpoždění strmý klesající charakter, který se pak zpomalí a dále klesal velmi pozvolně. Se zapnutými algoritmy průměrná přenosová rychlost nejprve mírně klesala a pak přešla do téměř lineárního poklesu, který se pak změnil v jemný pokles. Byl tedy vyvozen závěr, že změna velikosti ztrátovosti se zapnutými algoritmy Fast Retransmit a Fast Recovery je vhodná pro ovlivnění průměrné přenosové rychlosti.

U analýz vlivu zpoždění v krátkém a dlouhém časovém úseku bylo zjištěno, že algoritmy Rychlého odeslání a Rychlého opakování přenosu neměly žádný vliv na průměrnou přenosovou rychlost a že umělým zpožděním lze také ovlivňovat přenosovou rychlost. Také zde byl zaznamenán další rozdíl mezi programem IT Guru a Opnet Modelerem, kdy u IT Guru se zpoždění projevilo až po zadání celých sekund, kdežto v Opnet Modeleru se zpoždění projevilo už při zadávání setin sekundy. Dále byla zjištěna velká podobnost změny průměrné přenosové rychlosti v krátkém a dlouhém časovém úseku. Změna měla u obou analýz klesající exponenciální průběh. U dlouhého časového úseku byl pokles rychlosti méně strmý. Důvodem bylo měnící se zpoždění v určeném intervalu. U analýzy změny velikosti přijímací paměti bylo zjištěno, že pro stejné hodnoty paměti měly scénáře s nejmenší hodnotou maximální velikosti segmentu nejvyšší průměrné přenosové rychlosti. To bylo způsobeno mechanismem při odesílání, kdy se postupně s rostoucí velikostí paměti zmenšovala prodleva mezi jednotlivými segmenty. Průběhy průměrné přenosové rychlosti všech křivek v grafu zdánlivě připomínaly lineární nárůst a od určité hodnoty velikosti paměti byly konstantní. Dále bylo zjištěno, že se zvětšující se hodnotou maximální velikosti segmentu se konečná průměrná přenosová rychlost zvětšuje. Také tento způsob může být vhodný pro manipulaci s přenosovou rychlostí.

V poslední části tohoto dokumentu bylo zkoumáno, jak lze zasáhnout do programového kódu modelu a tím ovlivnit přenosovou rychlost. Postupně byla nalezena proměnná, která v sobě udržuje nastavenou hodnotu velikosti volné paměti přijímací strany. Touto proměnnou byla rcv_buff. Dalším krokem bylo nalezení vhodného místa v kódu programu. Místo pro vložení bylo zvoleno v procesu tcp_seg_receive. Poté bylo jednoduchým způsobem naimplementováno přepsání hodnoty v proměnné rcv_buff v požadovaném čase. Těchto implementací bylo do kódu vloženo několik. Pak byla tato úprava simulována a z výsledků bylo možno konstatovat úspěšnou implementaci kódu. Následně byly vypočítány přenosové rychlosti pro jednotlivé úseky nastavených hodnot velikosti paměti přijímací stanice. Výsledky ukazují pokles a následně nárůst přenosové rychlosti tak, jak byl nastaven pokles a následně nárůst velikosti paměti přijímací stanice.

Literatura

[1] ALLMAN, M., FLOYD, S., PARTRIDGE, C. *RFC2414 - Increasing TCP\'s Initial Window* [online].c2009[cit.2009-11-20].Dostupný z WWW: <http://www.faqs.org/rfcs/rfc2414.html>.

[2] Larry L. Peterson, Bruce S. Davie, Computer Networks: A Systems Approach, Morgan Kaufmann Publishers, San Francisco, California, October 1999

[3] STALLINGS, W., High-Speed Networks and Internets: Performance and Quality of Service, 2002, Prentice Hall, 2002

[4] STEVENS., W. *RFC2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit* [online].c20 [cit. 2009-11-13]. Dostupný z WWW: <http://www.faqs.org/rfcs/rfc2001.html>.

Seznam zkratek

ACK - Acknowledge

AN - Acknowledge Number

BGP - Border Gateway Protocol

CWND - Congestion Window

DS3 - Digital Signal 3

FTP - File Transfer Protocol

IGRP - Interior Gateway Routing protocol

IP - Internet Protocol

MSS - Maximum Segment Size

OSPF - Open Shortest Path First

PPP - Point to Point Protocol

RIP - Routing Information Protocol

RTT - Round Trip Time

SLIP - Serial Line Internet Protocol

SN - Sequence Number

SSIC - Slow Start Initial Count

SSTRESH - Slow Start Treshold

TCP - Trasmission Control Protocol

UDP - User Datagram Protocol

W - Window