VYSOKÁ ŠKOLA BÁŇSKÁ
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Optimalizace protokolu TCP

Michal Roháč, roh035
V Ostravě 26. března 2006

Semestrální projekt:
Technologie počítačových sítí

Obsah

Seznam obrázků

1 Přenos velkých objemů dat v rozlehlých WAN sítích

Kapacita linek tvořících páteřní část Internetu na mezinárodní úrovni, i na úrovni jednotlivých národních sítí v současnosti běžně dosahuje rychlostí v řádu jednotek či desítek gigabitů za sekundu. Koncové uživatelské stanice jsou dnes běžně vybaveny kartami typu Gigabit Ethernet. Stále zvyšující se kapacita přenosových linek je potřebná především pro přenosy velkých objemů dat.

Takto vysoké rychlosti spolu se zpožděním šíření signálu, ke kterému v rozlehlých sítích WAN dochází, přináší pro přenos velkých objemů dat některé principiální a implementační problémy. Výsledná propustnost závisí na správné implementaci a interakci všech komponent síťové komunikace - vlastní sítě, operačních systémů koncových stanic i síťových adaptérů koncových stanic. Problémy tohoto typu označujeme pojmem „end to end performance“.

Příklad

Problém můžeme ukázat např. na tomto příkladu. Vzdálenost Praha Brusel je vzdušnou čarou 700km, po cestě je to 900km, ale protože vzdálenost po páteřních linkách je většinou vyšší počítejme s délkou trasy 1000km. Mějme za úkol přenést 1TB dat z Prahy do Bruselu. Víme, že data putují po linkách různé kvality, přičemž ta „nejhorší“ z nich je řekněme typu OC-48 s rychlosti $2,5 Gb/s$.

Dále řekněme, že podle dostupných statistik ze směrovačů, je vytíženost mezinárodní linky z České Republiky v daném okamžiku $400 Mb/s$ a vytíženost ostatních linek předpokládáme srovnatelnou. Využitelná přenosová kapacita by tedy měla být přibližně kolem $2 Gb/s$.

Pokud bychom dokázali využít kapacitu linek a přenášeli data plnou rychlostí, měly by se data z Prahy do Bruselu dostat za něco málo přes hodinu, jak ukazuje následující výpočet.

$$\frac{data[b]}{rychlost[b/s]} = \frac{1 \cdot 10^{12} \cdot 8}{2 \cdot 10^{9}} : 3600 = 1,\bar{1}\, h \tag{1}$$

Toto ale pro nás zatím platí pouze teoreticky a abychom se tomuto času alespoň přiblížili, musíme optimalizovat transportní protokol, který se nám o přenos dat stará. Pokud bychom to neudělali a poslali data v souladu s implicitním nastavením TCP v našem OS, dostaneme se k podstatně delšímu času. Přenos by trval déle než 7 dnů, jak ukazuje výpočet č. 3 ve čtvrté kapitole (Konfigurační problémy).

2 Transportní protokoly

Transportní protokoly jsou v zásadě dvojího druhu. Dělíme je podle spolehlivosti na spolehlivé a nespolehlivé. V případě nespolehlivého se musí aplikace sama postarat o znovuvyslání dat pro případ jejich ztráty nebo poškození. Ty spolehlivé se naopak automaticky postarají o to, že pakety ztracené nebo poškozené při přenosu budou znovu poslány.

2.1 Transmission Control Protocol (TCP)

TCP poskytuje transportní službu se spojením. Poskytuje logické spojení mezi koncovými aplikacemi, tedy spolehlivý přenos dat a jeho specifikaci lze najít v [rfc-1].

2.2 User Datagram Protocol (UDP)

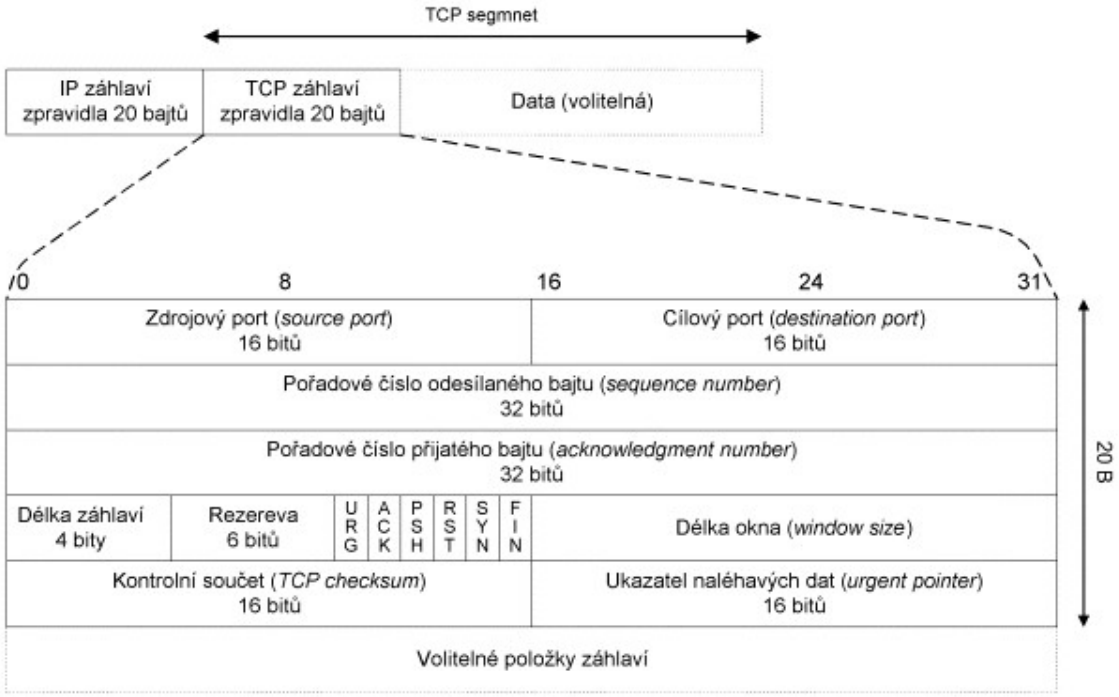
UDP poskytuje transportní službu bez spojení a představuje tak nespolehlivou transportní službu. Jde o velmi jednoduchý transportní protokol, jehož specifikace je uvedena v [rfc-2] z roku 1980.

Aplikace volí protokol podle požadované spolehlivosti. Pokud se dokáže postarat sama o spolehlivý přenos dat mezi koncovými uzly, může vystačit s UDP, ale pokud se sama nepostará o kontrolu doručení a opětovný přenos ztracených nebo zničených dat, pak musí volit spolehlivou transportní službu poskytovanou TCP. Další text bude věnován pouze protokolu TCP a jeho optimalizaci.

3 Řízení toku a řízení zahlcení

Řízení toku se realizuje několika spolupracujícími mechanismy:

- velikostí okna,
- pořadovými čísly bytů proudu dat,
- potvrzováním.



Obrázek 1: TCP segment

Záhlaví TCP segmentu obsahuje pole velikost okna (window size), které specifikuje, kolik bytů dat se může přenést od odesílatele k příjemci bez průběžného potvrzování doručení jednotlivých segmentů. Velikost okna není pevná hodnota, nedohaduje se při navazování spojení a lze ji v průběhu komunikace dynamicky měnit.

Jestliže má OS nastavenu velikost okna (vyrovnávací pamět pro příjem segmentů) na hodnotu větší než lze pojmout 16-ti bitovým číslem, dohaduje se při navazování spojení v TCP Option hodnota window scale. Velikost okna

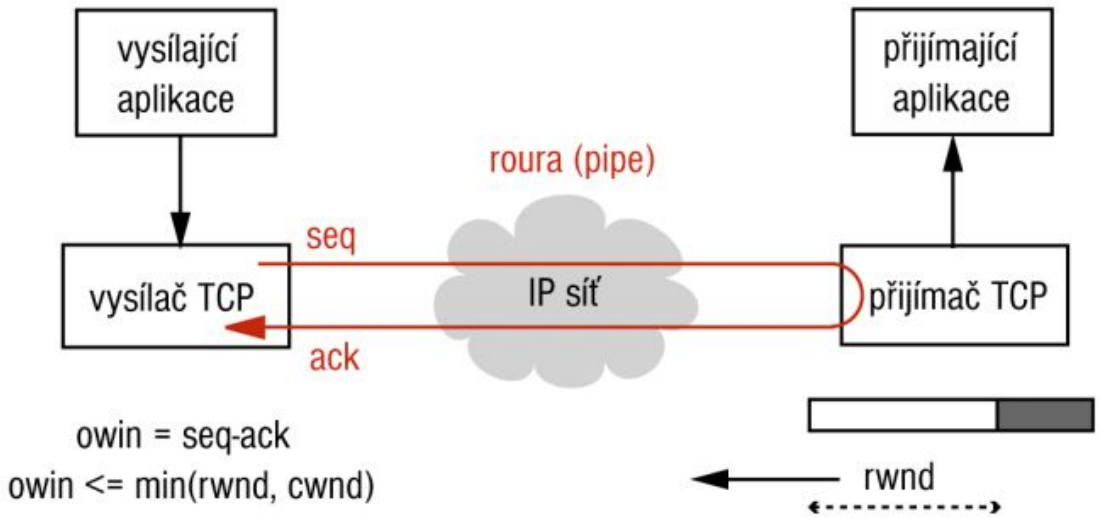
může být na straně příjemce i odesílatele různá. V průběhu navazování spojení (three-way handshaking) dochází k výměně tří segmentů, kdy v prvním (označovaném SYN) oznamuje hodnotu window scale žadatel o spojení a ve druhém (označovaném SYN ACK) může provést obdobnou operaci druhá strana. Samotná hodnota window scale je v průběhu dalších fází spojení považována za konstantu a je užívána oběma stranami k násobení 16-ti bitové hodnoty window size a k získání tak její skutečné hodnoty.

Velikost okna je velmi důležitý parametr. Když bude tato hodnota příliš nízká např. 1, musí se čekat na potvrzení každého odeslaného bytu resp. segmentu. Aby mohl být každý takovýto byte odeslán, musí být součástí TCP segmentu a ten součástí IP datagramu. Po sečtení velikosti hlaviček TCP segmentu a IP datagramu (viz. předchozí obrázek) zjistíme, že na přenos každého bytu dat připadá režie $40B$ dat v hlavičkách IP a TCP. Pokud by velikost okna byla naopak velká, dochází ke zbytečnému plýtvání vyrovnávací paměti. Při větším počtu navázaných spojení by tak mohlo docházet k mrhání systémovými prostředky.

Význam velikosti okna také dramaticky narůstá v souvislosti se zpožděním v rozlehlých WAN sítích. Protokol TCP využívá k zajištění spolehlivého přenosu dat mechanizmu potvrzování. Jinými slovy, odesílatel vyšle určitý objem dat a čeká na potvzení příjemce, že tyto data opravdu obdržel. Pokud odesílatel neobdrží potvrzení o doručení dat příjemci, může to být ze dvou důvodů. Prvním je, že data opravdu k cílové stanici nedorazily a druhým, že data sice k cílové stanici dorazily, ale ztratilo se potvrzení o jejich přijetí. V obou případech se vysílající stanice zachová stejně. Při odeslání segmentu si nastavuje svůj časovač (retransmission timer) pro přijetí potvrzení přijetí segmentu cílovou stanicí. Pokud vyprší časovač dříve než dorazí potvrzení, považuje vysílající stanice segment za ztracený a připraví jej k opětovnému vyslání (retransmisi).

Objem dat, který může odesílatel vyslat bez potvrzení, specifikuje v hlavičce TCP segmentu hodnota window size. Rozlehlé WAN sítě jsou charakteristické velkou latencí (zpožděním), která je dána jednak vzdáleností jednotlivých stanic v ramci sítě (dlouhá doba šíření signálu) a v neposlední řadě dobou potřebnou ke zpracování každého paketu v aktivních prvcích, především směrovačích. Vysílající stanice po vyslání dat o velikosti okna (window size), je nucena před vysláním dat nových čekat na potvrzení o jejich doručení (stačí potvrzení alespoň části dat viz. dále). Kdyby např. chodily potvrzení o přijetí segmentů se zpožděním $1s$ a velikost okna by byla nastavena na maximální velikost (samozřejmě bez využití window scale option), dosáhli

bychom přenosové rychlosti max. $64kB/s$ a to bez ohledu na to, že bychom měli k dispozici mnohonásobně rychlejší linky.



Obrázek 2: Řízení toku a řízení zahlcení (zdroj [ubi-1])

Pod pojmem **řízení toku** (flow control) se rozumí přizpůsobení rychlosti vysílače rychlosti příjímače. Přijímač průběžně informuje vysílač o zbývajícím volném místě ve své vyrovnávací paměti, neboli o tzv. okénku příjímače (recieve window, rwnd). Příjimačem avizovaná hodnota rwnd limituje vysílači objem dat, který může ještě odeslat (kdyby odeslal dat více, bude zahlcovat příjímající stanici). Tento objem dat, které může vysílač v daný okamžik poslat cílové stanici se nazývá aktuální okénko (outstanding window, owin). Pokud bude příjimač avizovat $rwnd = 0$ znamená to, že jeho vyrovnávací paměť je plná a vysílač musí s vysíláním dalších dat počkat, než příjimač potvrdí přijetí alespoň části dat a avizuje opět nenulové rwnd. Snižováním hodnoty rwnd nutí přijímač stanici vysílající data ke zpomalení vysílání.

Pod pojmem **řízení zahlcení** (congestion control) se rozumí přízpůsobení rychlosti vysílače rychlosti sítě, resp. nejmenší volné kapacitě na trase. Okénko owin není limitováno pouze okénkem rwnd, ale také tzv. okénkem zahlcení (congestion window, cwnd), které si vysílač sám průběžně určuje podle volné kapacity trasy. Vysílač se snaží využít volnou kapacitu trasy, proto průběžně zvětšuje okénko cwnd.

Obdrží-li vysílač signál o již nastalém nebo blížícím se zahlcení některé linky, zmenší velikost okénka cwnd na polovinu. Signálem o již nastalém zahlcení může být vypršení limitu pro příjem potvrzení, ale mohou to být také duplicitní potvrzení posledního přijatého segmentu od přijímače. Pro vysílající stranu může duplicitní potvrzení znamenat různé síťové problémy: zahozené segmenty (všechny segmenty došlé po nedoručeném segmentu vyvolají duplicitní potvrzení), přerovnání pořadí datových segmentů nebo duplikace segmentu potvrzení nebo dat v síti.

První fází řízení zahlcení je tzv. slow start, který se používá na začátku nového spojení a v případě obdržení signálu o zahlcení. Cílem je rychle rozběhnout posílání segmentů až do rychlosti přibližně odpovídající kapacitě trasy. Princip spočívá v tom, že každé nové spojení, je zahájeno vysláním jednoho segmentu s nastavením $cwnd = 1$. Po přijetí potvrzení o doručení tohoto segmentu, se okno zahlcení dvojnásobně zvětší, což umožňuje vyslat dvojnásobně větší množství dat (bez čekání na potvrzení o jejich doručení). S každým dalším potvrzením se velikost okna cwnd zvyšuje dvojnásobně (exponenciální růst), maximálně však do kapacity okna příjemce (nemá význam posílat více dat než je schopen příjemce pojmout). Slow startem můžeme vlastně označit fázi spojení, kdy se propustnost zvyšuje nejrychleji.

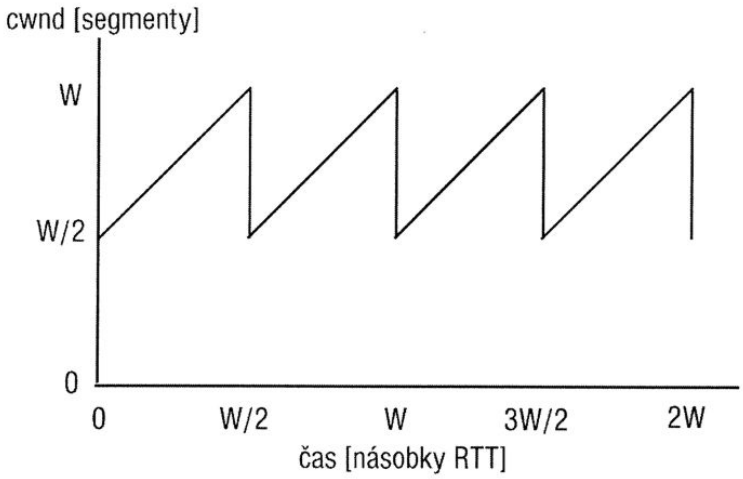
Jakmile vysílač obdrží zprávu o zahlcení, přechází do fáze vyhýbání se zahlcení (congestion avoidance). Jejím cílem je udržovat rychlost odesílání segmentů pod volnou kapacitou trasy a zároveň se snažit tuto kapacitu využít. Okénko zahlcení cwnd se zmenší na polovinu a dále se zvyšuje o jeden segment maximální velikosti tzv. MSS (maximum segment size) během každého intervalu RTT (round-trip time), roste tedy lineárně v závislosti na MSS. RTT je doba, za kterou přijde potvrzení o doručení segmentu. Zahrnuje zpoždění šíření signálu na vedení dané fyzikálními vlastnostmi přenosového média ($2/3c$[1] pro metalická vedení) a dobu zpracování paketů ve směrovačích na cestě a v přijímacím počítači. Jakmile dojde k další ztrátě segmentů v důsledku překročení kapacity trasy nebo kvůli chybě na některé lince trasy, cyklus prevence zahlcení se zopakuje.

Tato metoda nastavování okénka cwnd se nazývá AIMD (additive increase, multiplicative decrease), protože se při detekci zahlcení zmenšuje cwnd na polovinu původní hodnoty (násobek 1/2) a při snaze o znovudosažení volné kapacity trasy dochází pouze k lineárnímu nárustu cwnd (přírustek o

[1] rychlost světla ve vakuu

MSS). V ustáleném stavu by průběh velikosti okénka cwnd měl mít tvar pily, jak je znázorněno na dalším obrázku.

V důsledku kolísání objemu dat v síti a tím potažmo i RTT a samozřejmě také dalších nepravidelných jevů, přichází potvrzení od přijímače i signály o zahlcení též nepravidelně. Průběh cwnd se ve skutečnosti od ideální pily liší.



Obrázek 3: Vývoj okénka cwnd v ustáleném stavu (zdroj [ubi-1])

Pozn.: Hodnota W a W/2 na ose y, označuje velikost (resp. poloviční velikost) okna cwnd.

4 Konfigurační problémy

Přijímač může oznamovat okénko rwnd nejvýše do velikosti své vyrovnávací paměti. Vysílač rovněž může zvětšit okénko owin jen do velikosti své vyrovnávací paměti. Implicitní nastavení velikosti vyrovnávací paměti pro TCP, může být v různých OS různé. Např. v mých WindowsXP byla implicitní hodnota nastavena na -1, což v konečném důsledku znamená, že nejvyšší hodnota TCP window size, kterou bude můj OS avizovat, je limitována 16 bity (tedy může být max. 65535). Pokud na mém systému poběží např. ftp server, ze kterého mohou klienti stahovat data, žádný problém nenastane. Při navazování TCP spojení se serverem klientovi nic nebrání, aby s případným využitím window scale option oznámil „libovolnou“ hodnotu svého přijímajícího okna. Pro server bude tato hodnota horním limitem objemu dat, které může vyslat bez ohledu na to, jestli obdržel potvrzení o doručení těchto dat.

Problém nastane ve chvíli, kdy bude můj OS místo v roli serveru, vystupovat v roli klienta. Implicitní hodnota TCP window size bude v tomto případě (na mých WinXP) znamenat, že při navazování spojení bude klient (já) avizovat nejvyšší možnou velikost přijímajícího okna 65535 a neumožní tak serveru vyslat více dat, dokud od klienta (ode mě) neobdrží potvrzení o přijetí alespoň části dat předešlých. Když si představíme, že potřebujeme přenést velký objem dat na velkou vzdálenost (potvrzení chodí pomalu), tento problém nabývá na rozměru.

Interval RTT se v rozlehlých sítích pohybuje v řádech desítek až stovek milisekund. Nyní je správný čas vrátit se k našemu příkladu. Řekněme, že jsme pomocí příkazu ping zjistili průměrné RTT na trase Praha Brusel ve výši $40ms$. Znamená to, že jsme schopni každých $40ms$ poslat max. $64kB$ dat. Přesto, že je volná kapacita na trase $2Gb/s$, výsledná propustnost je pouze $12.8Mb/s$ jak ukazuje následující výpočet.

$$\frac{data[b]}{RTT[s]} = \frac{64 \cdot 10^3 \cdot 8}{4 \cdot 10^{-2}} = 12,8Mb/s \tag{2}$$

Standardní nastavení protokolu TCP a velká zpoždění, ke kterým dochází v rozlehlých sítích, se stává pro přenos velkých objemů dat po linkách s velkou kapacitou značně neefektivní. Kdybychom byli schopni využít volnou kapacitu linek, přenesli bychom data již za $1,1h$, jak jsme již spočítali (viz. výpočet výše).

Následující výpočet demonstruje jak dramaticky se při stejném zpoždění avšak standardním nastavení velikosti příjimacího okna pro TCP segment zpomalí přenos dat.

$$\frac{data[b]}{rychlost[b/s]} = \frac{1 \cdot 10^{12} \cdot 8}{12,8 \cdot 10^6} : 3600 = 173,6\bar{1}\ h \approx 7,23dne \tag{3}$$

Jak je vidět, nestačí jen mít k dispozici kvalitní linky s velkou přenosovou rychlostí, ale také nakonfigurovat transportní protokol tak, aby se výsledná propustnost rovnala nebo se alespoň blížila využitelné kapacitě nejpomalejší linky na trase. Pokud bude propustnost vyšší než využitelná kapacita, znamená to plýtvání pamětí, v opačném případě pamětí šetříme až příliš na úkor nevyužití dostupné kapacity přenosových linek.

Propustnost zvýšíme zvětšením vyrovnávacích pamětí jak na straně vysílače tak příjímače. Aby nebylo spojení limitováno velikostí vyrovnávacích pamětí (tzn. aby výsledná propustnost nebyla pod úrovní využitelné kapacity linek), musíme okénka na obou stranách zvětšit na objem „roury“ mezi vysílačem a příjmačem daný součinem RTT a volné kapacity trasy. Následující výpočet bere v úvahu údaje z našeho příkladu a ukazuje optimální velikost vyrovnávacích pamětí pro dané spojení.

$$kapacita[B/s] \cdot RTT[s] = (2 \cdot 10^9 : 8) \cdot 4 \cdot 10^{-2} = 10MB \tag{4}$$

Optimální velikost okna je přímo závislá na zpoždění šíření signálu a na kapacitě přenosového kanálu. Když pomineme různé negativními vlivy (např. přetížení linek, zahlcení nebo spadnutí linky a následně nutnost posílání paketů jinou cestou), lze obecně říci, že v ustáleném stavu by mělo být zpoždění mezi 2 stanicemi v rámci sítě víceméně konstantní. Za proměnnou můžeme naopak považovat kapacitu linek na trase, která plyne z existence alternativních cest (s různou kapacitou), navíc se kapacita linek mění podle aktuální zátěže. Stanovení optimální velikosti okna se tedy bude lišit případ od případu.

Klíčem ke stanovení optimální hodnoty je uvědomit si k jakým účelům je síť využívána. Pokud bude uživatelem člověk, který využívá pro svou práci mail, chat a www stránky, můžeme od něj čekat, že bude navazovat větší počet spojení, ale většinou nebude přenášet žádná velká data. Pokud by měl takovýto uživatel (a stává se to často) např. otevřených 20 oken webového prohlížeče, spuštěného emailového klienta a do toho ještě chatoval s několika lidmi najednou, má v jednu chvíli navázaných např. 25 spojení. Pokud by pro každé spojení bylo nutné, aby OS alokoval např. $10MB$ pro rwnd, bude tento uživatel brzy pociťovat nedostatek paměti RAM - což v praxi znamená najít volné prostředky na její rozšíření.

Optimální velikost je tedy dána součinem RTT a volné kapacity trasy. Pro univerzální nastavení lze sice použít místo aktuálně dostupné kapacity trasy rychlost připojení k síti providerem, ale vždy je třeba zvážit jaký to bude mít dopad. U velmi rychlých připojení bych doporučuval nastavovat optimální velikost rwnd jen pro stahování větších objemů dat a po dokončení stahování vrátil jeho hodnotu zpět na nižší úroveň. U pomalejších přípojení (řádově několik jednotek MB/s), které jsou v dnešní době běžně dostupné, bych doporučoval nastavit hodnotu rwnd max. okolo čtyřnásobku základní velikosti okna tedy na $262140B$.

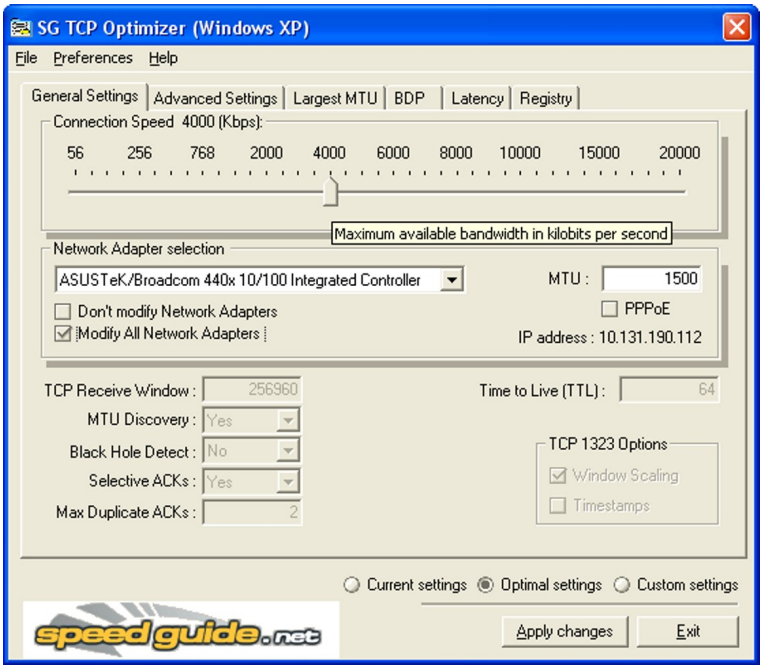
5 Optimalizace TCP na platformě Windows

Na platformě Windows lze ovlivňovat protokol TCP pomocí registrů. Přímý zásah do registru není zrovna bezpečný způsob jak ovlivňovat činnost systému Windows ani pro zkušeného uživatele. Pro tyto účely je lepší využít patche do Windows, které lze najít na internetu volně ke stažení, nebo použít software, který poskytne „user friendly“ rozhraní pro ladění výkonu protokolu TCP. Jedním z nich je [prg-1] SG TCP Optimizer, který je volně šiřitelný.

TCP Optimizer je komplexním nástrojem, který poskytuje uživatelské rozhrání pro výpočet, konfiguraci a nastavení parametrů TCP. Dovoluje vygenerovat „optimální“ hodnoty nebo nakonfigurovat své vlastní. Pro jejich výpočet obsahuje nástroje na měření latence sítě, zjišťování velikosti MTU, kalkulátor BDP (Bandwidth Delay Product) a v neposlední řadě také obsahuje vestavěný editor registru. V další části se budu věnovat popisu pouze některých částí tohoto rozhraní, s ohledem na využití pro tuto práci.

TCP Optimizer umožňuje nastavit parametry TCP spojení, nebo si nechat vygenerovat optimální parametry podle zadané šířky pásma připojení k Internetu. Po aplikaci těchto nastavení přepíše příslušné položky v registrech a po restartu systému, který si vyžádá, je nastavení dokončeno. Důležitou součástí je také možnost ukládat si jednotlivá nastavení a později se k nim vracet.

V hlavní menu v nabídce File najdeme funkce pro ukládání a obnovování nastavení TCP. Nejdůležitější je první záložka General. Nastavení Connection speed je určeno pro automatické generování optimálního nastavení. Rozsah rychlostí je na první pohled nedostatečný, ale v dnešní době by si zde měla být schopna vybrat většina uživatelů internetu, protože využitelná kapacita je dáná nejpomalejší linkou, což je obvykle rychlost připojení poskytovaná vaším internet providerem. V případě, že jste připojení rychlejší linkou než $20Mb/s$ musíte vystačit s ručním nastavením parametrů.



Obrázek 4: SG TCP Optimizer - základní okno

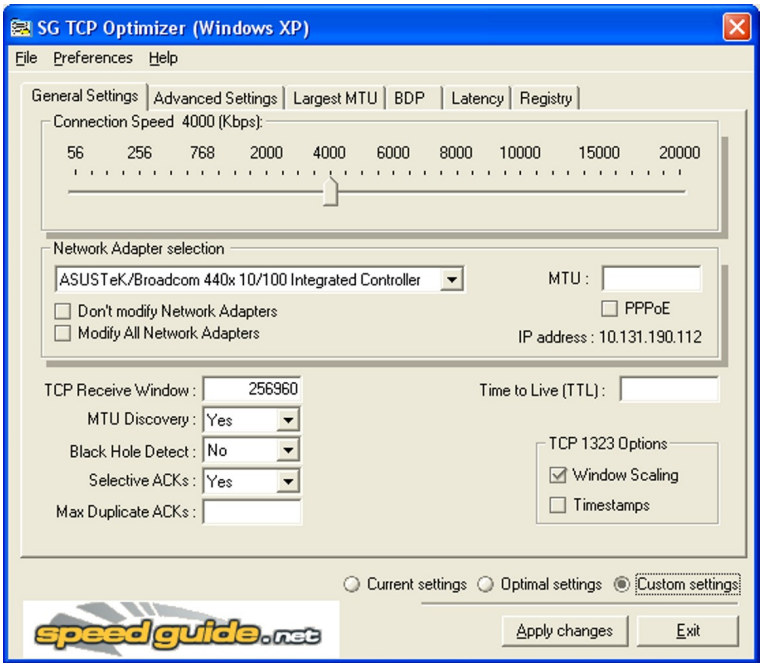
Pro nastavení optimálních parametrů (vygenerovaných programem) stačí vybrat rychlost připojení, vybrat ze seznamu síťových adaptérů ten, pro který chceme změny provést případně zvolit modifikaci parametrů pro všechny adaptéry (Modify all network adapters). Dále ve spodní části vybrat Optimal settings, použít toto nastavení pomocí apply changes viz. úvodní okno a restartovat op. systém (je vyžadováno).

5.1 Vlastní nastavení parametrů

Experimentovat s vlastním nastavením lze zvolením položky Custom Settings. Tato volba dává možnost editace jednotlivých parametrů, jak ukazuje následující obrázek a popis jednotlivých parametrů.

5.1.1 TCP Receive Window (rwnd)

Toto je nejdůležitější parametr pro efektivní využívání TCP jak bylo popsáno výše, který ovlivňuje zásadním způsobem výslednou propustnost TCP spojení. Jeho velikost je dáná 16-ti bitovou hodnotou window size v hlavičce TCP segmentu a případně konstantou (kterou se winow size násobí) window scale option dohodnutou při navazování spojení.



Obrázek 5: SG TCP Optimizer - vlastní nastavení

5.1.2 MTU Discovery

Povoluje nebo zakazuje zjšťování hodnoty MTU (Maximum Transfer Unit). Tato hodnota určuje velikost největšího paketu, který projde sítí bez fragmentace. Fragmentaci paketů má za následek zdržování vysílání, protože fragmentace dat znamená další operaci navíc. Pro IPv4 má každý router na trase schopnost fragmentovat a defragmentovat pakety, což vede k jeho zatěžování a následně zvyšování RTT. Je lepší posílat data o velikosti MTU, nepřispíváme tak k nárůstu RTT, nezatěžujeme routery a přitom vysíláme data pomocí nejmenšího možného množství (nefragmentovaných) paketů, což statisticky přináší nejmenší riziko ztráty paketů a následně jejich opětovného vysílání viz. slow start a congestion avoidance. Bližší informace o path MTU discovery lze najít v [rfc-4].

5.1.3 Black Hole Detect

Povolení této volby má za následek, že se TCP Optimizer snaží detekovat tzv. „black hole“ routery během zjišťování největšího MTU. Při path MTU discovery se nastavuje v hlavičce IP datagramu don‘t fragment bit, pokud router zjistí, že paket nemůže bez fragmentace poslat dál, zahodí jej a informuje o tom odesílatele pomocí zprávy ICMP destination unreachable. V této

zprávě zároveň oznamuje použitelné MTU. Tímto způsobem se při hledání MTU prochází celá trasa, než se najde MTU. Tzv. black hole routery (černé díry) tuto ICMP zprávu nazasílají. TCP je závislé na těchto zprávách během vykonávání path MTU discovery.

Po ztrátě několika segmentů s nastaveným don't fragment bitem (ztráta ACK těchto segmentů) je zde podezření na black hole router. Aktivní volba black hole detect dovoluje TCP pro takovýto případ, aby vyslal několikrát nepotvrzený zkušební segment bez don't fragment bitu. Pokud přijde potvrzení tohoto segmentu, MSS se sníží a vyšle se další segment s don't fragment bitem opět nastaveným. Povolení této volby zvyšuje počet retransmisí segmentů při hledání největšího MTU.

5.1.4 Selective ACKs

TCP/IP používá standardně kumulativní potvrzování paketů, což znamená, že pokud přijde potvrzení paketu s daným pořadovým číslem, předpokládá se automaticky potvrzení i předchozích paketů. Pokud ale vyšleme řadu paketů z nichž se ztratí např. ten první a zbytek dorazí k příjímači v pořádku, ten nemůže poslat jejich potvrzení. Po vypršení timeoutu je tak vysílač nucen vyslat všechny nepotvrzené pakety znovu přesto, že některé z nich dorazily. Selective acknowledgements tento nedostatek odstraňuje tím, že kombinuje kumulativní potvrzování se selektivním (potvrzení každého segmentu). Pokud dojde příjímající strana k závěru, že došlo ke ztrátě segmentu, vyšle vysílači SACK paket, ve kterém ho informuje o datech, které přijala. Vysílající strana poté provádí retransmisi pouze chybějících dat. Bližší vysvětlení celého mechanizmu lze najít v [rfc-5].

5.1.5 Max Duplicate ACKs

Tento parametr udává počet duplicitních potvrzení, která musí být přijata pro stejné sekvenční číslo vyslaných dat před spuštěním tzv. fast retransmit mechanismu. Fast retransmit řeší situaci při doručení segmentu, který je mimo pořadí (s jiným než očekávaným pořadovým číslem). Jakmile příjemce takový segment obdrží, musí vyslat několik duplicitních potvrzení s pořadovým číslem chybějícího segmentu. Pro vysílající stranu však duplicitní potvrzení může znamenat různé síťové problémy (viz. řízení zahlcení) a né vždy je zapotřebí ihned opětovně vyslat chybějící segmenty. Tento parametr by měl být v rozsahu 1 – 3, přičemž standardně se nastavuje 2.

5.1.6 Time to Live (TTL)

Toto nastavení specifikuje implicitní nastavení TTL v hlavičce odchozích IP paketů. TTL udává maximální životnost IP paketu v síti v počtech přeskoků viz. [puz-1, strana 319]. Přeskokem se má na mysli průchod paketu přes směrovač na trase, po průchodu se hodnota TTL sníží o 1. Pokud paket projde sítí přes TTL směrovačů (vykoná TTL přeskoků) a nenajde svůj cíl (hodnota TTL se sníží na 0), je směrovačem zahozen. Malá hodnota TTL má za následek, že paket nemusí dorazit k cíli a velká zase může znamenat potenciálně dlouhou dobu než zjistíme, že se paket ztratil. Doporučená hodnota je 64.

5.1.7 TCP RFC 1323 Options

Tyto parametry vychází z normy [rfc-3]. Pro nás je zajímavá volba zvětšení rwnd okénka nad 65535 pomocí window scale option.

5.2 Detekce parametrů

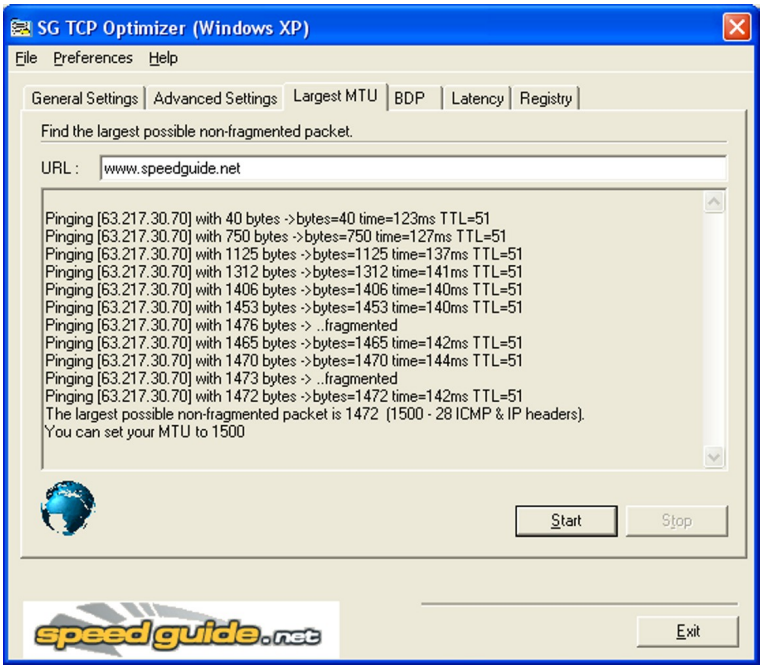
TCP Optimizer má vestavěné nástroje pro snadnější určování některých parametrů. Jedná se především o určování MTU (Maximum Transfer Unit), BDP (Bandwidth Delay Product) a určování latence sítě.

5.2.1 Určování MTU

O způsobu určování MTU pojednává [rfc-4]. TCP optimizer nabízí jednoduchý způsob, jak MTU zjistit. Na záložce „Largest MTU“ se specifikuje ip adresa nebo DNS jméno serveru, čímž definujeme trasu, na které chceme MTU zjistit. Poté už stačí jen kliknout na tlačítko start a počkat si na výsledek jak ukazuje následující obrázek.

5.2.2 BDP kalkulátor

Na záložce BDP se nachází takzvaný BDP kalkulátor. Zkratka BDP (bandwidth delay product) označuje důležitý koncept TCP/IP, který byl v tomto textu již několikrát zmíněn. Jde o souvislost velikosti okna rwnd se zpožděním v síti daným intervalem RTT a těmito parametry limitovanou propustností TCP spojení. Množství dat, které vysílač odešle příjemci závisí (jak už bylo výše popsáno) na velikosti okna rwnd. Další data se mohou vyslat až po obdržení potvrzení předešlých dat, než však toto potvrzení k vysílající stanici dorazí, uplyne doba RTT, což je dáno fyzikálními vlastnosmi média a dobou zpracování paketů ve směrovačích a koncových stanicích.

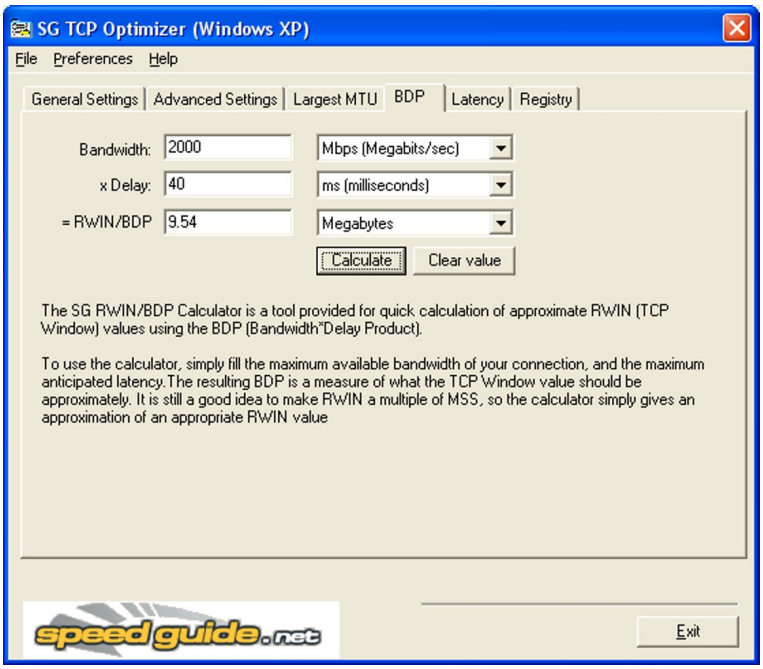


Obrázek 6: SG TCP Optimizer - Largest MTU

Pro využití dostupné šířky pásma, je nutné data vysílat pokud možno nepřetržitě, místo „zdržování“ se čekáním na potvrzení. Na systému potvrzování je však založena spolehlivost protokolu TCP. Způsob jak překonat tento problém je, poslat příjemci takový objem dat, aby nám nejlépe těsně před dokončením takovéhoto přenosu stihlo přijít od příjemce potvrzení o přijetí alespoň části dat a my mohli ve vysílání pokračovat.

Když víme, že nám odpověd přijde za dobu RTT, můžeme příjemci po tuto dobu odesílat data a pokud známe nebo jsme schopni odhadnout šířku pásma kterou máme k dispozici, jsme schopni určit součinem těchto dvou hodnot kolik těchto dat můžeme vyslat a tím potažmo zjistit potřebnou velikost okna rwnd příjemce.

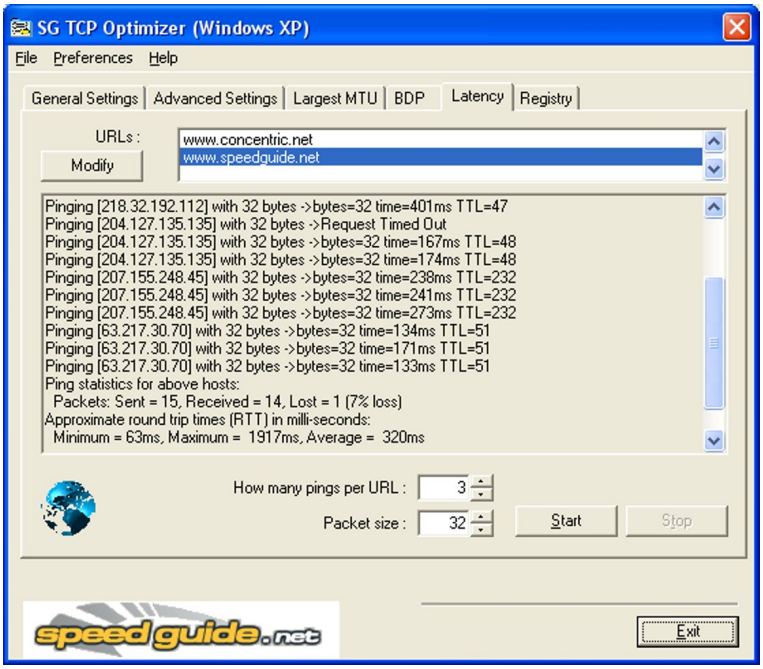
BDP kalkulátor nám práci s násobením a převáděním jednotek ulehčuje tím, že nám dovolí jednoduše zadat hodnoty známých parametrů včetně jednotek a dopočítat neznámou. Protože spolu všechny 3 parametry úzce souvisí a né všechny jsme schopni v praxi lehce určit, dovoluje nám kalkulátor zadat kterékoliv 2 parametry s tím, že třetí dopočte.



Obrázek 7: SG TCP Optimizer - BDP kalkulátor

5.2.3 Určování latence sítě

Pod záložkou Latency se skrývá nástroj pro zjišťování intervalu RTT. Jedná se vlastně o vylepšený ping, kdy si můžeme zvolit adresy jednotlivých serverů u kterých chceme zkoumat dobu odezvy, kolik pingů se má ke každému serveru vyslat a jakou mají mít velikost. Po spuštění potom TCP Optimizer zjistí jednotlivé odezvy a vyhodnotí nejlepší, nejhorší a průměrnou dobu odezvy (RTT).



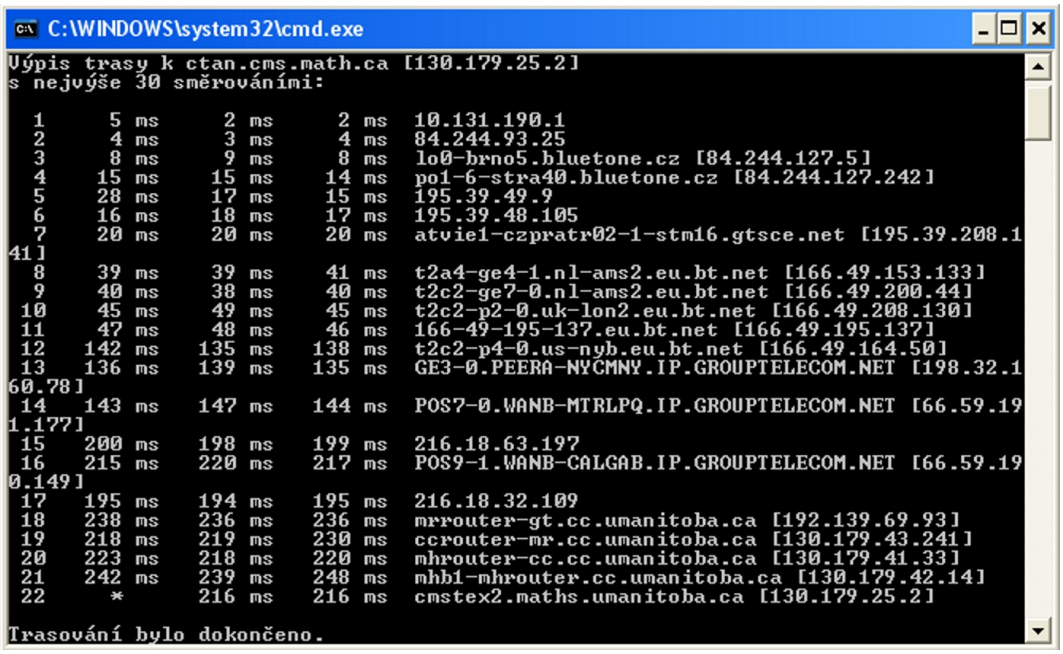
Obrázek 8: SG TCP Optimizer - Latence sítě

5.3 Pokročilé možnosti

TCP Optimizer toho nabízí mnohem více. Mezi zajímavé volby patří např. optimalizace prohlížeče Internet Explorer, která umožňuje lépe využívat protokol HTTP 1.1 tím, že dovolí navázat více spojení s web serverem (implicitně bývají 2). Více spojení samozřejmě znamená rychlejší načítání souborů resp. jejich „paralelní“ stahování. V neposlední řadě zde také patří možnost úpravy registrů. Tyto možnosti jsou přístupné přes záložky „Advanced Settings“ a „Registry“.

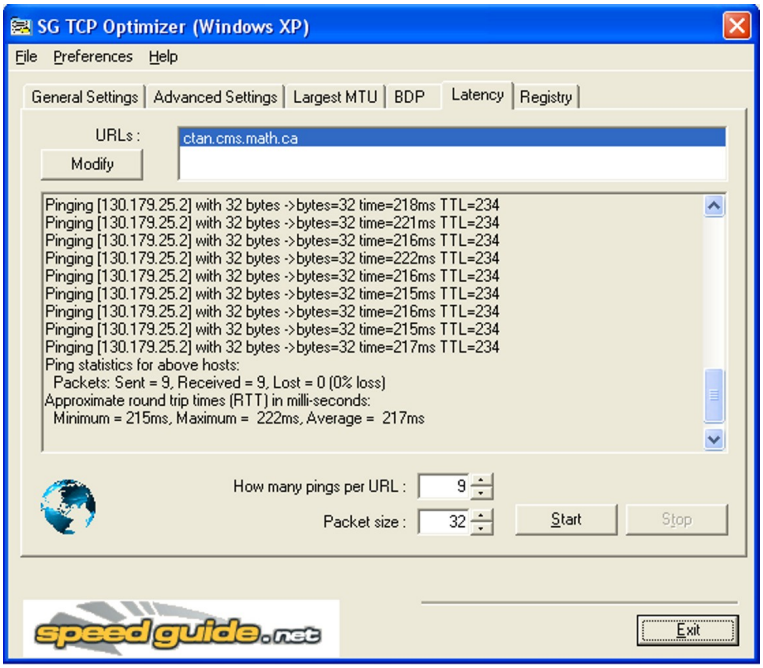
6 Praktický příklad

Výše popsanou teorii jsem se pokusil v rámci dostupných možností ověřit. Měl jsem k tomu k dispozici připojení k internetu s maximální přenosovou rychlostí $4Mb/s$, které je sdílené s dalšími uživateli. Rozhodl jsem se pro stahování většího souboru z umístění, které bude mít pokud možno velkou dobu odezvy. Vhodným souborem se ukázala být distribuce texlive 2005 v podobě iso image souboru zkomprimovaného metodou zip [prg-3] o velikosti $682955936B$.



Obrázek 9: Výpis trasy příkazem tracert

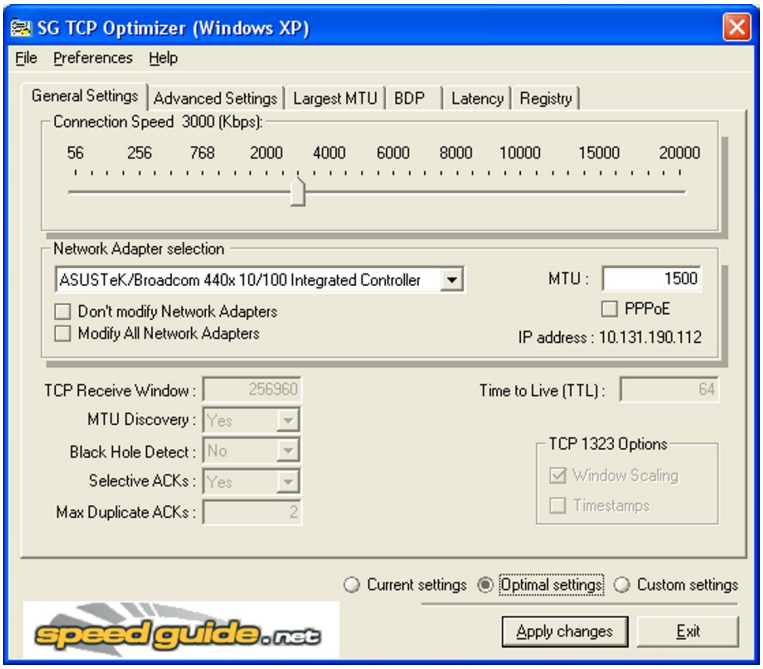
Pro stahování jsem si vybral kanadské zrcadlo kvůli relativně velké vzdálenosti a tím potažmo i velkému zpoždění. Pomocí příkazu tracert jsem zjistil, že pakety budou na cestě procházet 22 směrovači a ověřil zpoždění RTT na této trase, které se pohybuje kolem ($220ms$).



Obrázek 10: Určení průměrného RTT pro ctan.cms.math.ca

Soubor jsem stahoval celkem dvakrát, jednou s implicitním nastavením systému WindowsXP a podruhé s nastavením (viz. obrázek) vygenerovaným TCP Optimizerem. Při stahování jsem odchytával provoz nad protokolem TCP paketovým analyzátorem Ethereal [prg-4].

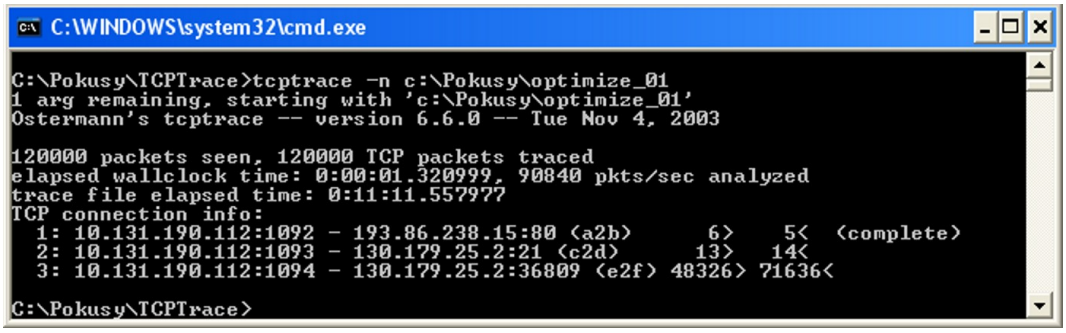
Pro upřesnění dodávám, že soubory s odchycenými pakety byly pro analýzu příliš velké, proto jsem je rozdělil na menší soubory (o velikosti kolem $100MB$), které obsahovaly přibližně kolem 120tis. paketů. Pokud nebude uvedeno explictně jinak, bude se předpokládat analýza pouze první části tohoto souboru, což však jistě nebude na újmu obecnosti, protože výsledky analýzy odchycených paketů uložených v dalších dílčích souborech jsou téměř shodné.



Obrázek 11: Optimální nastavení vygenerované TCP Optimizerem

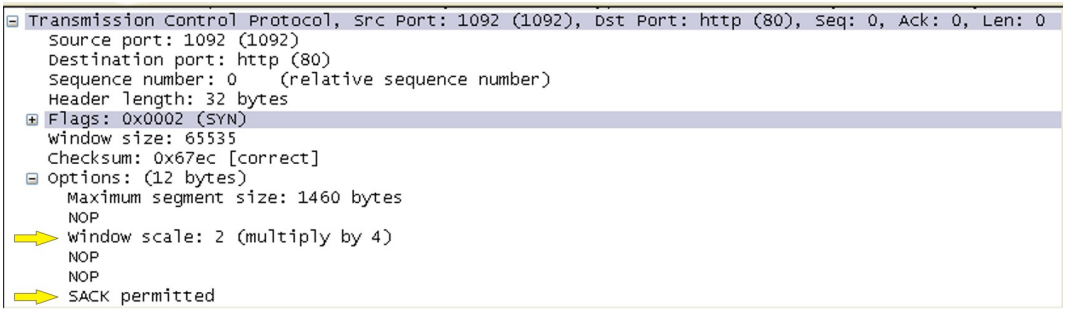
Kliknutím na odkaz pro stahovaný soubor začal proces stahování, který vyžadoval navázání 3 spojení, jak ukazuje výpis komunikace vygenerovaný dalším použitým nástrojem [prg-5]. První spojení, které není z našeho pohledu příliš zajímavé, bylo navázáno na portu 80 (http) a je následkem kliknutí na odkaz s internetovou adresou. O druhé a třetí spojení, které je z našeho pohledu naopak velmi zajímavé, se postaral protokol FTP (protokol pro spolehlivý přenos souborů). Tento protokol využívá dvě spojení, jedno řídící (command chanel) a druhé datové (data chanel). Řídící spojení existuje po celou dobu FTP relace a přenáší se po něm příkazy a odpovědi mezi serverem a klientem. Datové spojení je vždy zřízeno jen pro konkrétní přenos dat. Řídící spojení se navazuje vždy na portu 21 (druhé spojení) a datové spojení (třetí spojení) se v tomto případě navázalo na portu 36809[2].

[2]V případě pasivního režimu (tento případ) otevírá datové spojení klient a server mu přiděluje port nad 1023, v běžném režimu by datové spojení otevíral server na portu 20.

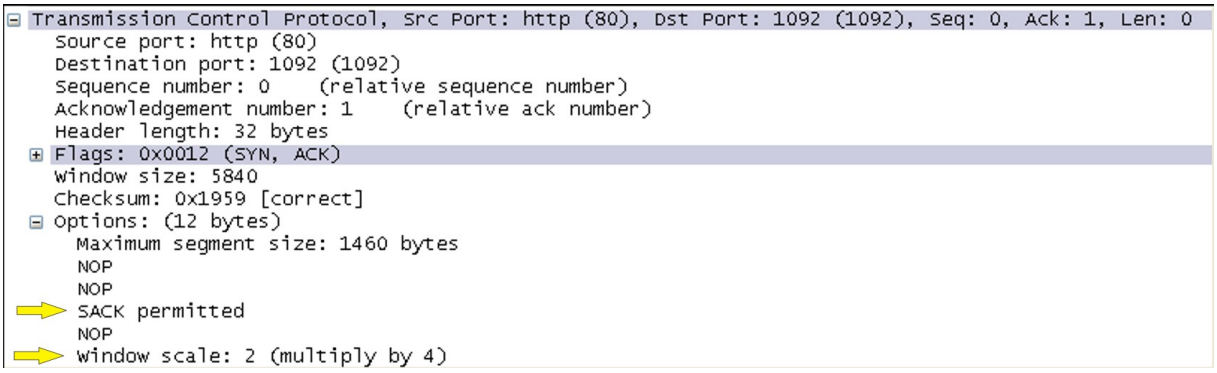


Obrázek 12: Výpis komunikace v odchycených paketech

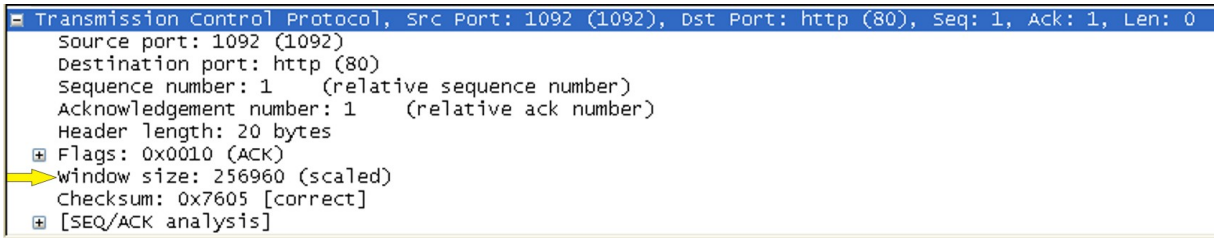
V následujících obrázcích bude ukázán průběh navazování spojení (three-way handshaking) a dohadování parametrů. Pro navázání spojení je nutná výměna 3 segmentů - SYN, SYN ACK a běžného ACK. Během prvních dvou segmentů se domlouvají obě strany na velikosti okna a domluví se na použití scaled window, ve třetím běžném potvrzovacím segmentu jakožto i v dalších, používají dohodnutou konstantu k určení správné velikosti okna.



Obrázek 13: První segment v rámci three-way handshaking

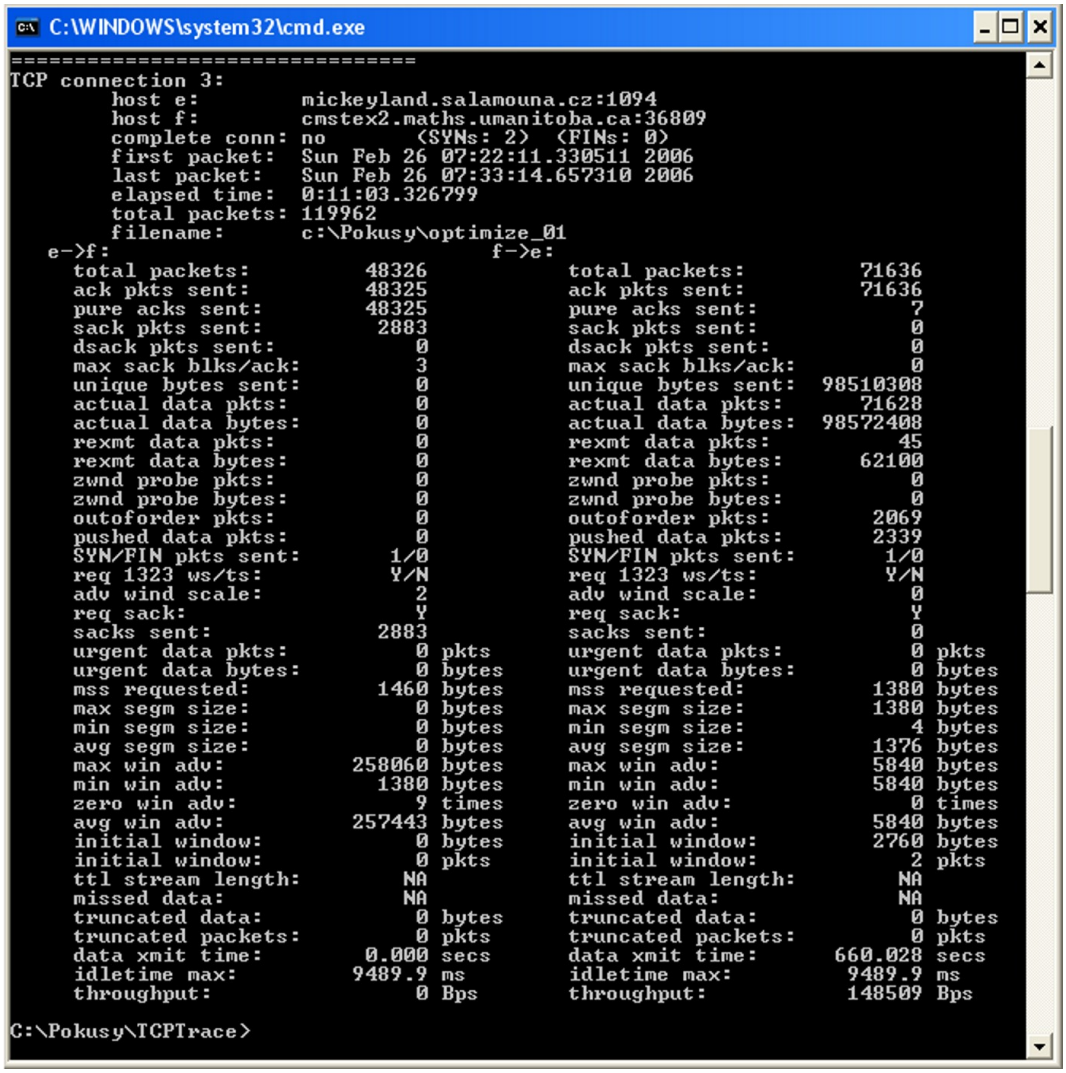


Obrázek 14: Druhý segment v rámci three-way handshaking

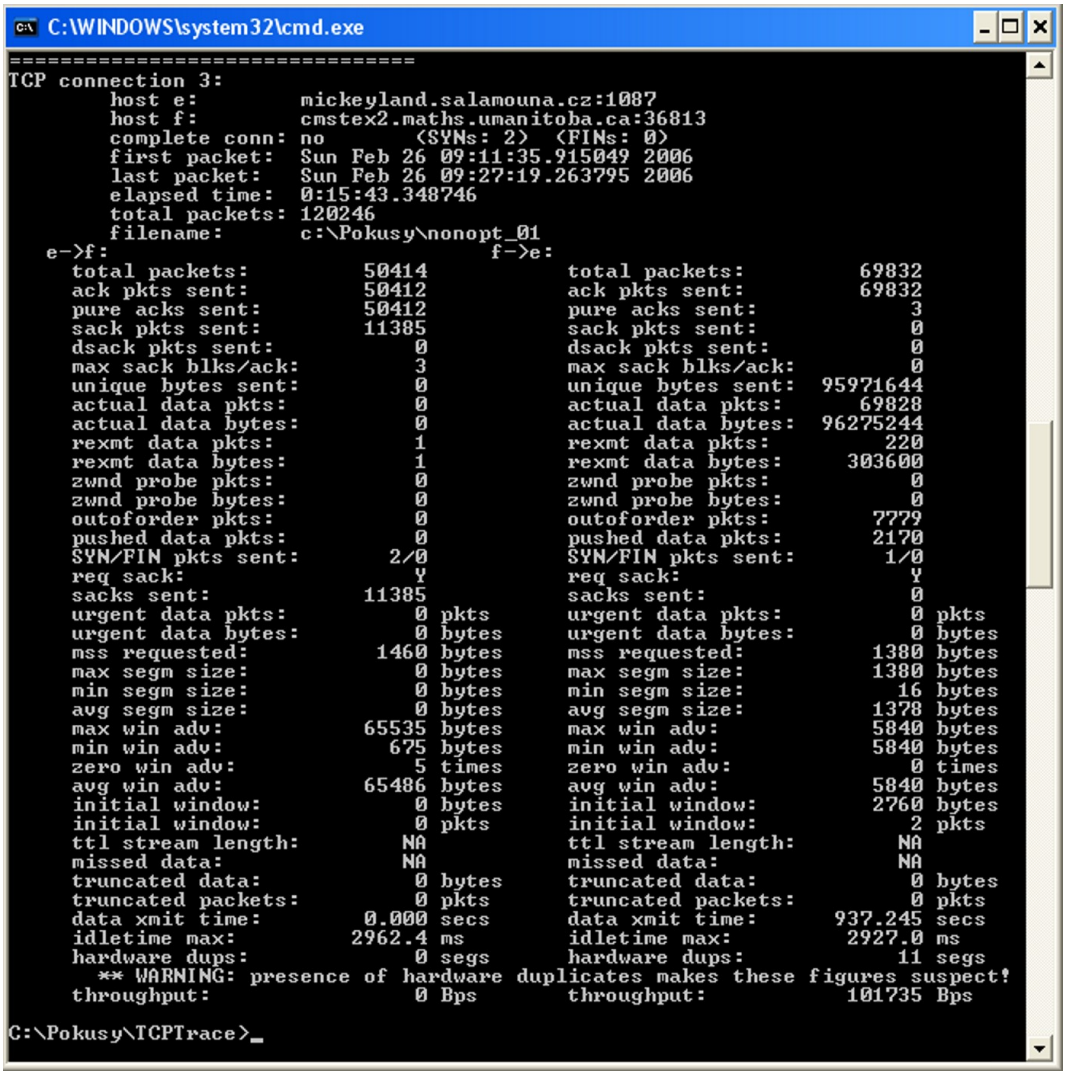


Obrázek 15: Třetí segment v rámci three-way handshaking - běžné potvrzení

Analýzu propustnosti jsem provedl pomocí nástroje tcptrace [prg-5]. Tímto nástrojem ve spolupráci s programem xplot nebo jeho obdobou v jazyce java jPlot [prg-6] se dá vygenerovat jak textový, tak i grafický výstup analýzy TCP. Na následujících obrázcích můžete porovnat textový výstup podrobné analýzy v případě implicitního i optimalizovaného nastavení TCP.



Obrázek 16: Podrobná analýza odchycených paketů pro optimálního nastavení

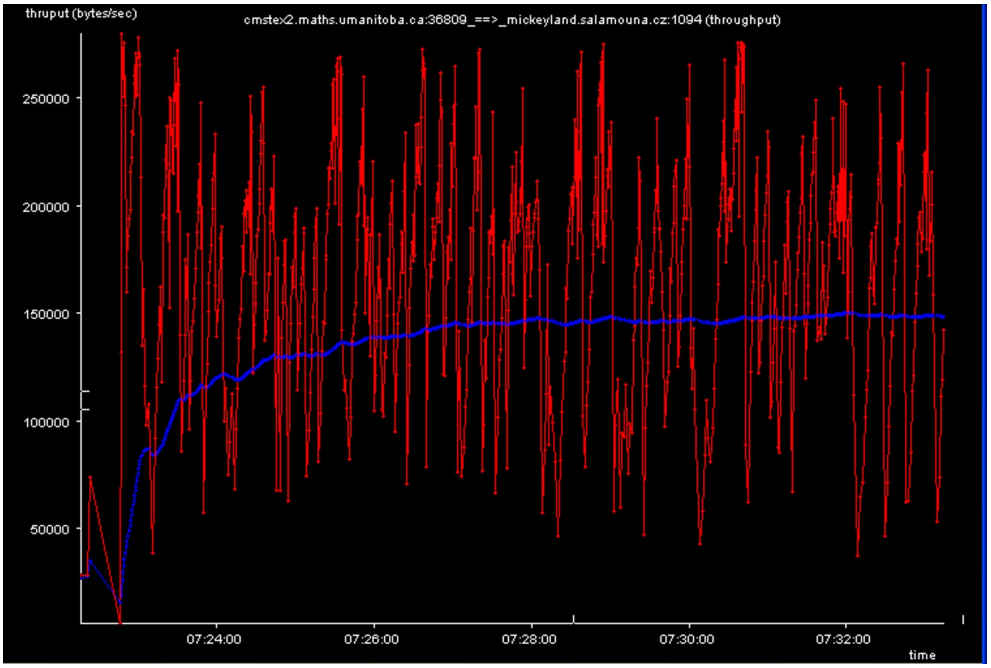


Obrázek 17: Podrobná analýza odchycených paketů pro implicitní nastavení

Poslední položkou ve výpisech je propustnost sítě uváděná v Bps (bytech za sekundu). Porovnáním této hodnoty pro oba případy zjistíme, že při optimálním nastavení byla propustnost sítě o polovinu větší než u implicitního nastavení TCP. Důvodem je právě velikost okna. V položce min win adv (nejmenší nenulová avizovaná velikost okna) si můžeme všimnout nízké hodnoty v obou případech stahování. To znamená, že v obou případech docházelo k využití téměř celé šířky okna rwnd, dokonce byla v obou případech několikrát inzerována nulová velikost okna. Důvodem je latence sítě při doručování potvrzení a také samozřejmě chybovost, protože potvrzení chodila jak s určitým zpožděním (které je v čase proměnlivé), tak se mohla také ztrácet.

Podílem rexmt data bytes a unique bytes sent je možno udělat si představu o procentuálním počtu retrasmisí odesílaných dat. V případě implicitního nastavení byl server přesvědčen o $0,32\%$ dat z celkového přenášeného objemu, že jsou ztracena a byl nucen je vyslat znovu. V případě optimalizovaného nastavení bylo toto číslo pětkrát nižší ($0,06\%$). Důvodem nižší hodnoty v optimalizovaném případě je fakt, že měl server „více času“ k rozhodnutí jestli jsou data ztracena. Předpokládal totiž, že se ztratilo ACK a dostatečná kapacita rwnd na straně klienta mu umožňovala ještě nějakou dobu odesílat další data v pořadí místo okamžité retransmise. Během této doby však mohla přicházet kumulativní potvrzení s vyšším pořadovým číslem, které přijetí segmentu se ztraceným ACK potvrdila.

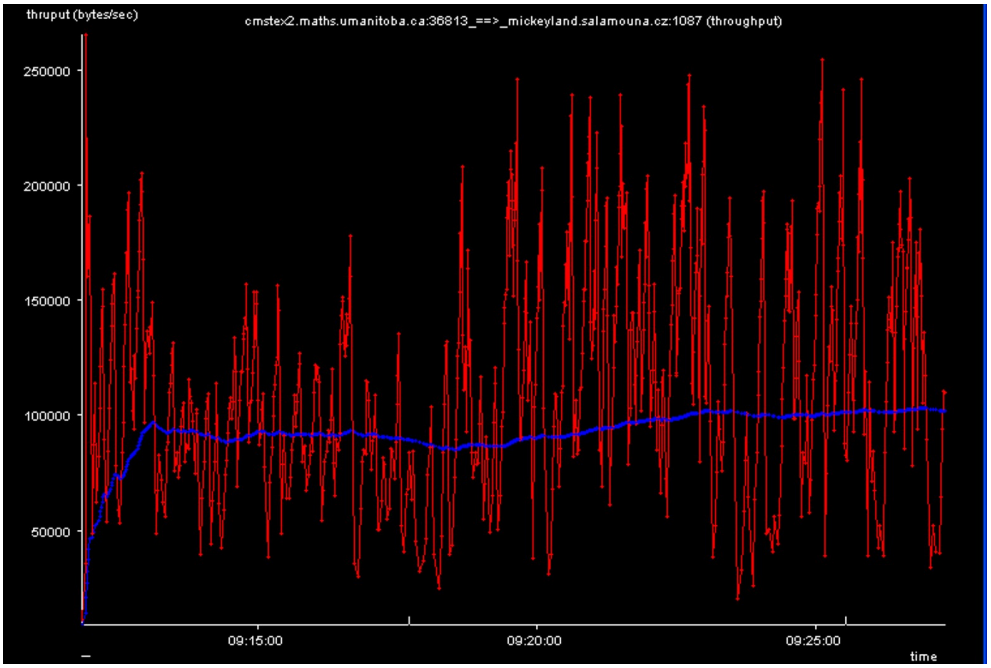
Pokud se ztrácela data, nebo přicházela mimo pořadí, oznamoval toto klient pomocí selektivního potvrzování, což bylo povoleno v obou případech stahování. Pokud k takovéto situaci dojde, klient o tom informuje server pomocí SACK segmentu. Porovnáním polí sack sent u obou analýz zjistíme, že v případě malého okna rwnd k tomu docházelo mnohem častěji. Samozřejmě SACK nutí vysílající stanici vyslat chybějící segment znovu, což opět zvyšuje režii TCP. SACK nemusí být poslán přijímající stanicí ihned (záleží na implementaci), protože ztracený segment může stále dorazit. Jediné, co přijímající stanici k takovému chování opravdu nutí, je velikost svého rwnd rovná nule, čili stav, kdy jakýkoliv další došlý segment musí stejně ignorovat, protože ho jednoduše nemá kam uložit. S větší velikosti okna tedy není vysílání SACK segmentů tak časté, což má samozřejmě pozitivní vliv na propustnost TCP spojení.



Obrázek 18: Graf propustnosti TCP spojení s nastavením optimálních parametrů

Grafy propustnosti zobrazil jPlot [prg-6] na základě dat vygenerovaných pomocí tcptrace [prg-5]. Červená křivka udává pseudo[3] okamžitou hodnotu propustnosti a modrá vývoj propustnosti průměrné během celé doby stahování.

[3]Okamžitá propustnost není přesný výraz. Pro lepší přehlednost grafu, bylo použito ke kalkulaci této „okamžité“ hodnoty vždy 100 vzorků.



Obrázek 19: Graf propustnosti TCP spojení s nastavením implicitních parametrů

Stahování v případě implicitního nastavení parametrů TCP ve WinXP trvalo 2h a propustnost TCP spojení tak dosáhla v průměru $95kB/s$. S optimálním nastavením se soubor stáhl za 1h a 17min a propustnost TCP spojení tak dosáhla v průměru $150kB/s$. S nárůstem šířky pásma přenosových linek a s potřebou přenášet stále větší objem dat na velké vzdálenosti lze očekávat, že v budoucnu bude také vzrůstat potřeba úpravy TCP pro konkrétní (velké) přenosy dat, samozřejmě pokud se neobjeví nová technologie řešící daný problém.

Vzhledem k složitosti řešení výkonnostních problémů, které vyplývají z kombinace chování řady komponentů, vznikla z iniciativy evropských sítí národního výzkumu a vývoje (National Research and Educational Network, NREN) aktivita PERT (Performance Enhancement and Response Team), jejíž cílem je koordinace výzkumu v oblasti výkonnostních problémů ve vysokorychlostních sítích na mezinárodní úrovni.

Reference

[puz-1] Ing. Rita Pužmanová, CSc.: TCP/IP v kostce, KOPP 2004

[ubi-1] Dr. Sven Ubik: Přenosy velkých objemů dat v rozlehlých Gigabitových sítích, 2003, http://staff.cesnet.cz/~ubik/publications/2003/gigaST.doc

[rfc-1] RFC 793 – Transmission Control Protocol, http://www.faqs.org/rfcs/rfc793.html

[rfc-2] RFC 768 – User Datagram Protocol, http://www.faqs.org/rfcs/rfc768.html

[rfc-3] RFC 1323 – TCP Extensions for High Performance, http://www.faqs.org/rfcs/rfc1323.html

[rfc-4] RFC 1191 – Path MTU discovery, http://www.faqs.org/rfcs/rfc1191.html

[rfc-5] RFC 2018 – TCP Selective Acknowledgement Options, http://www.faqs.org/rfcs/rfc2018.html

[prg-1] SG TCP Optimizer version 2.0.3, http://www.speedguide.net

[prg-2] TCP Optimizer Help, http://www.speedguide.net/tcpoptimizer.php

[prg-3] TeXLive 2005 iso image soubor komprimovaný metodou zip (využito k testu), stahováno z kanadského zrcadla na adrese ctan.cms.math.ca

[prg-4] Ethereal – paketový analyzátor, http://www.ethereal.com

[prg-5] TCPTrace - program pro analýzu TCP provozu z odchycených paketů, Shawn Ostermann, Ohio University, http://www.tcptrace.org

[prg-6] jPlot - java verze programu xPlot pro generování grafických výstupů, http://masaka.cs.ohiou.edu/software/tcptrace/jPlot