FAKUTLA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
BRATISLAVA

Rozšírenia protokolu TCP so zameraním na bezdrôtové a vysoko rýchlostné siete

DIPLOMOVÁ PRÁCA

AUTOR: ANDREJ HOLLÝ
VEDÚCI: RNDR. RICHARD OSTERTÁG

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry.

V Bratislave

_____

Obsah

Kapitola 1

Úvod

V dnešnej dobe si už asi nik nedokáže predstaviť život bez Internetu. Každý deň využíva jeho služby pre rôzne účely veľké množstvo ľudí, každý deň zapĺňame linky tejto globálnej siete veľkým množstvom dát. Každým okamihom do infosféry Internetu pribúdajú nové informácie a jeho veľkosť sa stále zväčšuje. Mnoho ľudí používa Internet intuitívne bez najmenšej vedomosti ako tento systém pracuje. Proste pracuje spoľahlivo a to je jeden z dôvodov jeho nárastu.

No podobne ako sa vyvíja ľudstvo, aj Internet podlieha zmenám. Internet úzko súvisí s pojmom sieťový protokol. Sieťovým protokolom (v ďalšom len protokol) nazývame systém pravidiel, algoritmov, správ a ďalších mechanizmov umožňujúcich komunikáciu medzi software-om a hardware-om v sieťových zariadeniach a medzi samotnými uzlami v sieti. Samotný vznik Internetu a jeho vývoj až po súčasnosť je spätý hlavne s jedným protokolom (rodinou protokolov) - TCP/IP.

Snaha spojiť viaceré počítače do jednej siete sa prvýkrát úspešne podarila v roku 1973. Projekt sa volal ARPAnet a bol vyvinutý americkou armádou (United States Defense Advanced Research Projects Agency). Na komunikáciu sa použil protokol TCP, predchodca terajšej rodiny protokolov TCP/IP, a kompletne bol zdokumentovaný v roku 1973 v RFC 675. V roku 1977 bola vytvorená druhá verzia TCP. Prevratné zmeny však začala až tretia verzia v roku 1978, ktorá stanovila problém TCP v jeho komplexnosti (protokol sa snaží robiť priveľa, preto treba prerozdeliť jeho funkcionalitu na viac častí). Navrhovala rozdelenie architektúry do dvoch protokolov. Táto idea bola nakoniec úspešne zapracovaná vo verzii 4 v roku 1980. Priniesla rozdelenie funkcionality protokolu na dve časti - sieťovú a transportnú. Tomu aj zodpovedá názov TCP/IP - TCP pre transportnú vrstvu a IP pre sieťovú vrstvu. Čoskoro sa stala štandardom pre ARPAnet a s ňou sa zrodil Internet taký, ako ho poznáme dnes.

V súčasnosti je stále väčšina dát v Internete prenášaná rodinou protokolov TCP/IP. Kým protokol sieťovej vrstvy IP sa stará o adresáciu a

doručovanie dát po sieti, protokol transportnej vrstvy má za úlohu poskytovať určité služby užívateľovi nad sieťovou vrstvou. Transportná vrstva TCP/IP obsahuje dva protokoly - TCP a UDP. Tieto dva protokoly nám poskytujú rozdielne služby a záruky. UDP je veľmi jednoduchý protokol, ktorý poskytuje minimum potrebné pre komunikáciu s IP protokolom. Naopak TCP je protokol zaručujúci celú škálu služieb pre komunikáciu, napr. vytvorenie spojenia, zaručenie spoľahlivosti prenosu, atď. Ďalej sa budeme zaoberať len protokolom TCP, ako dominantným transportným protokolom v dnešnom Internete (80-95% celkovej premávky).

Samotný protokol TCP prešiel od svojho vzniku mnohými zmenami. Ako sa vyvíjali prenosové média, zväčšovala sa kapacita a rýchlosť liniek a pribúdali stále nové pripojené počítače do Internetu, menil sa aj TCP. Táto práca sa zaoberá potrebou prispôsobovania sa protokolu TCP novým podmienkam. V druhej kapitole si v skratke pozrieme základné techniky štandardného protokolu TCP, stanovíme teoretický model a zadefinujeme niektoré pojmy potrebné v ďalšom texte.

Cieľom tretej kapitoly je ukázať problémy nasadenia štandardného TCP v špecifických situáciách. V dnešnej dobe zaznamenávame nárast bezdrôtových sietí, čoraz väčšie množstvo dát prenášaných Internetom si vyžaduje linky s väčšou kapacitou. Ako ukážeme, na takýchto sieťach sa TCP správa neefektívne. Zároveň si ukážeme prečo je tomu tak a prečo je potreba vhodného rozšírenia TCP protokolu.

Štvrtá kapitola sa zaoberá rozšíreniami TCP pre vysokorýchlostné siete. Tu si dané rozšírenia kategorizujeme a popíšeme. Cieľom je podať ucelený prehľad techník používaných pre vysokorýchlostné siete a ich charakteristiky.

Posledná, piata kapitola sa zaoberá prehľadom techník rozšírení TCP pre bezdrôtové siete, ich popisom a kategorizáciou z rôznych pohľadov. Z dôvodu veľkého množstva rozšírení sme vybrali len niektoré a ich funkcionalitu popísali.

Kapitola 2

TCP - Základné pojmy a techniky

Transmission Control Protocol (TCP) je protokolom transportnej vrstvy. Nerieši otázky adresácie uzlov v sieti, ani samotné doručovanie dát. Na to využíva služby poskytované protokolom IP. TCP komunikácia prebieha vždy medzi dvoma účastníkmi, *odosielateľom* (sender) a *prijímateľom* (receiver). Ostatné uzly v sieti (smerovače, brány, proxy, atď...), cez ktoré komunikácia fyzicky prechádza, sú pre samotné TCP transparentné. Tieto uzly nazveme spojovacie uzly (intermediate nodes). Takúto komunikáciu tiež nazývame *koncovou* (end-to-end).

TCP medzi odosielateľom a prijímateľom vytvára *spojenie*. Tento fakt nám zaručuje mnohé výhody, napr. dohodnutie rôznych parametrov komunikácie, udržiavanie komunikácie v rámci jedného spojenia, atď. TCP je *spoľahlivý* (teda garantuje doručenie správ resp. nemožnosť prenosu ohlási vyššej vrstve), *potvrdzovaný* (pre každú správu poslanú do siete a doručenú prijímateľovi sa generuje potvrdzovacia správa *ACK*) a *zachováva poradie prenášaných dát* (v akom poradí sme dáta dostali z vyššej vrstvy u odosielateľa, v takom ich vrátime prijímateľovi). Tieto vlastnosti zaručujú maximálnu snahu protokolu o doručenie poslaných dát (TCP robí všetko preto, aby doručil dáta).

Komunikácia TCP je obojsmerná (*full duplexná*), teda odosielateľ aj prijímateľ môžu vysielať súčasne. Pre jednoduchosť ale predpokladajme *half duplexné* spojenie, teda odosielateľ vysiela a prijímateľ prijíma (a posiela ACK). V ďalšom texte predpokladáme aspoň základné znalosti fungovania TCP, poprípade si ich môže čitateľ naštudovať v nasledujúcich zdrojoch [1, 2, 3, 4].

Protokol TCP je navrhnutý tak, že z vyššej vrstvy prijíma prúd dát (stream of bytes) a balí ich do ucelených častí - *správ*. Tieto správy sa v rôznych literatúrach nazývajú rôzne, napr. správa, segment, paket. Veľkosť správ je kontrolovaná dvoma faktormi. Prvým je limit na veľkosť správy

maximum segment size (MSS), ktorý sa dohoduje na začiatku spojenia. Druhým je systém kontroly tokov, ktorý určuje množstvo správ, ktoré je možno spracovať zdrojom. Tým sa budeme zaoberať neskôr.

Veľká časť informácií v tejto kapitole pochádza hlavne z nasledujúcich zdrojov [1, 2, 3, 4], RFC 2001 a RFC 2581.

2.1 *Spoľahlivosť*

Pre zaručenie spoľahlivej komunikácie, protokol TCP používa systém potvrdzovania správ. Pre každú správu doručenú prijímateľovi sa generuje potvrdzovacia správa *ACK*. Tá je následne poslaná späť odosielateľovi a signalizuje správne doručenie. Odosielateľ si musí udržiavať informáciu o stave komunikácie a o tom, ktoré správy boli už potvrdené. Poslané správy si ukladá do špeciálnej fronty a vyhadzuje ich až po potvrdení. Každá správa je identifikovaná *poradovým číslom* (sequence number), ktoré určuje poradie, v akom bola poslaná. Zároveň pre každú poslanú správu má nastavený časovač.

Ak dôjde k strate správy A alebo jej potvrdenia A_Ack, odosielateľ sa to môže dozvedieť nasledovnými spôsobmi. Buď mu prídu potvrdenia k správam poslaným neskôr ako správa A (teda potvrdenia k správam s väčším poradovým číslom ako má A, tzv. duplicitné správy - DupACK alebo DACK), alebo nám vyprší príslušný časovač. V prvom prípade dôjde k porušeniu vlastnosti TCP o zachovávaní poradia dát, v druhom prípade po vypršaní časovača predpokladáme stratu správy A alebo jej potvrdenia.

Potvrdenia mimo poradia a časovače nám signalizujú možnú stratu správy. Samotné zaručenie spoľahlivosti je však riešené preposielaním nepotvrdených správ. Tie máme v špeciálnej fronte, teda nie je problém s ich opätovným preposielaním.

Otázkou je, ako nastaviť interval časovača? V [4] je ukázané, ako sa interval počíta. Závisí od RTT (Round Trip Time), čo je čas od poslania správy do siete až po doručenie jej potvrdenia. Výpočet uvažuje pôvodnú hodnotu RTT (hodnotu už vypočítanú) a upravuje ju podľa novo nameranej hodnoty RTT. Pre nás nie je tento výpočet veľmi dôležitý. Snáď je potrebné podotknúť, že po zistenej strate, TCP zväčší interval časovača pre preposielanú správu.

V praxi poznáme viacero riešení potvrdzovania. Najjednoduchším a všade podporovaným je kumulatívne potvrdzovanie. Idea spočíva v tom, že viaceré po sebe idúce správy (teda súvislá postupnosť správ od začiatku spojenia zoradená podľa poradového čísla) stačí potvrdiť jednou potvrdzovacou s najväčším poradovým číslom (tu predpokladáme, že si prijímateľ pamätá najväčšie poradové číslo pre súvislú postupnosť správ od začiatku spojenia). U odosielateľa po prijatí takéhoto potvrdenia vieme, že všetky správy s menším alebo rovným poradovým číslom boli úspešne doručené.

Tu nastáva problém, napríklad keď pošleme správy 1, 2, 3, 4 a 3 sa stratí, prijímateľ nám pošle potvrdenie pre správu 2. Odosielateľ potom už vie, že správy 1 a 2 boli úspešne doručené. O úspešnom poslaní správy 4 však nič netuší a musí čakať, kým mu nevyprší časovač pre správu 3 (častokrát sa stane, že počas preposielania správy 3 vyprší časovač pre 4, teda budeme nútení preposlať už úspešne doručenú správu 4).

Riešením problému kumulatívneho potvrdzovania je selektívne potvrdzovanie tzv. SACK (selective ACK opísaný v RFC 2018). Tu sa v potvrdzovacej správe posiela dodatočná informácia o doručených správach mimo poradia (teda správach, ktoré boli úspešne doručené, ale nepatria do súvislej postupnosti doručených správ od začiatku spojenia, lebo niektoré správy s menším poradovým číslom ešte nedorazili prijímateľovi). Takéto potvrdzovacie správy nazývame duplicitné potvrdenia *dupACK* s dodatočnou informáciou SACK alebo len SACK. Po obdržaní takejto informácie máme lepšiu predstavu o doručených správach u prijímateľa a nemusíme preposielať správy, ktoré už boli doručené aj keď mimo poradia.

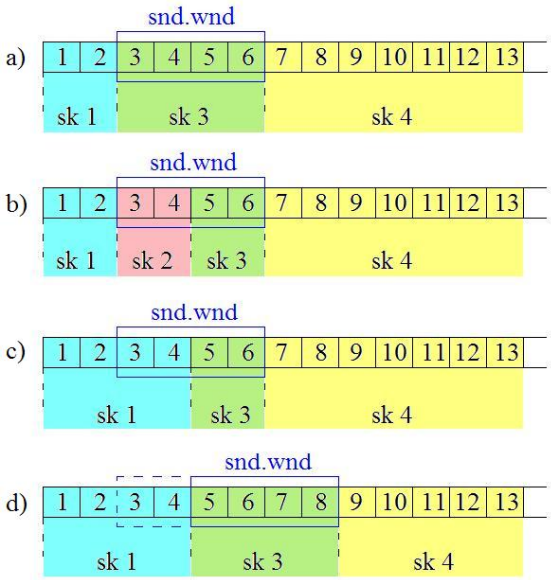
2.2 *Kontrola toku (flow control)*

Základným predpokladom úspešného fungovania TCP bola implementácia kontroly tokov. Bola zahrnutá v pôvodnej špecifikácii RFC 793. Ide o kontrolu rýchlosti posielania dát medzi prijímateľom a vysielateľom, je to ochrana prijímateľa pred preplnením jeho vyrovnávacej pamäte (buffer). Základom je algoritmus *sliding window*, ktorý kontroluje posielanie dát do siete. Na začiatku sa obe strany dohodnú, aké bude maximálne množstvo poslaných dát, počas komunikácie si potom posielajú informáciu o tom, aké množstvo dát sú schopné prijať/spracovať v danom okamihu - okno *snd.wnd*. Algoritmus *sliding window* potom riadi samotný proces posielania. Predpokladáme už znalosť tejto techniky, pre bližšie informácie doporučujeme literatúru [1, 2, 3]. V skratke len spomenieme hlavnú ideu algoritmu.

Sliding window mechanizmus si môžeme predstaviť ako posúvanie posielacieho okna *wnd*. Toto okno má veľkosť *snd.wnd*. Dáta, ktoré sa majú prenášať sieťou, si môžeme predstaviť ako postupnosť správ. Každej správe priradíme poradové číslo. Správy si zaradíme do štyroch skupín:

1. poslané a potvrdené,
2. poslané a nepotvrdené,
3. neposlané a prijímateľ je pripravený prijímať ďalšie,
4. neposlané a prijímateľ nie je pripravený prijímať ďalšie.

Vďaka vlastnosti TCP o zachovávaní poradia dát, postupnosť správ (usporiadaná podľa poradového čísla) bude rozdelená nasledovne. Prvé budú



Obrázok 2.1: Mechanizmus Sliding window: (veľkosť posielacieho okna sú 4 správy) a) správy 1 a 2 sú potvrdené, 3, 4, 5, 6 sú pripravené na poslanie b) k správam 3 a 4 došli potvrdenia c) preklasifikovanie správ 3 a 4 do skupiny 1 d) posun okna.

správy prvej skupiny, potom budú nasledovať správy druhej, potom tretej, a nakoniec štvrtej skupiny. Tu sa ešte môže stať, že správy prvej a druhej skupiny budú navzájom pomiešané v prípade preusporiadania alebo straty správ a ich potvrdení.

Posielacie okno *wnd* si predstavme ako okno nad postupnosťou správ. Začína nad prvou správou druhej skupiny a má dĺžku *snd.wnd* správ. Správy pred oknom patria do prvej, správy za oknom patria do štvrtej skupiny. Ak v okne máme nejaké správy tretej skupiny, tak ich pošleme prijímateľovi a prekategorizujeme ich do druhej skupiny. Po obdržaní potvrdení, prekategorizujeme potvrdené správy z druhej do prvej skupiny. Vtedy posúvame posielacie okno na prvú správu druhej skupiny a proces opakujeme. Tento mechanizmus je zobrazení na obrázku 2.1.

V tejto časti by sme chceli ešte spomenúť pojem *self clocking* mechanizmus. V TCP je posielanie nových správ do siete vyvolané obdržaním potvrdzovacej správy. Inak povedané každá doručená potvrdzovacia správa znamená úspešné doručenie a umožní poslanie ďalšej správy (poprípade ďalších správ). Tým si je TCP sám schopný kontrolovať posielanie dát, čo sa v praxi označuje ako *self clocking* mechanizmus.

2.3 *Kontrola zahltenia (congestion control)*

Kontrolou zahltenia (congestion control) chápeme distribuovaný algoritmus na zdieľanie sieťových prostriedkov medzi susediacimi uzlami. Jej implementáciu si vyžiadali problémy spôsobené preplnením vyrovnávacích pamätí spojovacích uzlov v sieti. Tento problém nazvaný *congestion collapse* (kolaps v dôsledku zahltenia) sa prvýkrát vyskytol v októbri 1986. Kontrola toku TCP nedokázala zabrániť a zvládnuť takú situáciu lebo rieši rýchlosť posielania na úrovni odosielateľ-prijímateľ. Tu však bol problém v zahltení spojovacích uzlov po ceste, ktoré sú pre samotné TCP transparentné. Odosielateľ si danú situáciu vysvetlil tak, že prijímateľ má dostatok prostriedkov na spracovanie dát, preto nie je potrebné znížiť vysielaciu rýchlosť a stačí preposlať všetky stratené správy. Tým sa však celá sieť plnila preposlanými správami a spojovacie uzly boli nútené zahadzovať veľké množstvo dát.

Spojovacie uzly majú rôzne techniky, ako riešiť prepĺňanie svojich vyrovnávacích pamätí. Základné sú QSD (queue scheduling disciplines) a QMM (queue memory management). QSD sa zaoberá klasifikáciou tokov do tried a kontrolou ich prístupu k sieťovým prostriedkom. QSD algoritmami sú napríklad FIFO, Priority queueing, Fair queueing, Weighted Round Robin a Token Bucket. QMM sa zaoberá samotným plnením vyrovnávacích pamätí, radením paketov do front, resp. ich zahadzovaniu v prípade zahltenia (alebo jeho hrozby). Do QMM môžeme zaradiť algoritmy Tail drop a Random early detection. Samotnými technikami v spojovacích uzloch sa nebudeme bližšie zaoberať. Pre podrobnejší prehľad techník odporúčame literatúru [4].

Zahltením nazveme stav, kedy do siete vstupuje viac dát ako je v danom momente sieť schopná preniesť (podľa [4]). Inak povedané zahltenie je stav v sieti, kedy dochádza k plneniu vyrovnávacích pamätí spojovacích uzlov. Tu nastáva problém, ako môže TCP riešiace koncovú komunikáciu predvídať, predchádzať a riešiť problém plnenia spojovacích uzlov.

Predpokladáme, že straty paketu sú vyvolané zahltením a následným zahadzovaním paketov v sieti. Pre klasické drôtové siete, pre ktoré bol TCP stvorený, je tento fakt pravdivý na 99%. Pravdepodobnosť straty z dôvodu nespoľahlivosti linky je taká malá, že sa ňou ďalej nemusíme zaoberať. V nasledujúcej kapitole však ukážeme, že tento predpoklad je kritickým miestom v niektorých špecifických situáciách. Zatiaľ však uvažujme o stratách spôsobených len zahltením.

Riešením problému zahltenia sa zaoberal pán van Jacobsonom, ktorý v roku 1986 vytvoril algoritmy kontroly zahltenia. V súčasnosti sú vyžadované v každej implementácii TCP. My ich popíšeme a v ďalších kapitolách sa pozrieme na ich problémy v rôznych situáciách a na možnosť ich vylepšenia pre dané situácie. Pre podrobné informácie odporúčame preštudovať špecifikácie RFC 2001 a RFC 2581. Sú to nasledovné algoritmy:

- Pomalý štart (Slow start),

- Predchádzanie zahlteniu (Congestion avoidance),
- Rýchle preposielanie (Fast retransmit),
- Rýchle zotavenie (Fast recovery).

Ich cieľom je efektívne využívanie kapacity linky, predchádzanie a riešenie zahltenia, férovosť medzi tokmi.

2.3.1 *Pomalý štart (Slow start)*

Jednou z príčin kolapsu v dôsledku zahltenia bol fakt, že novovytvorené spojenia sa hneď od začiatku snažili využiť dostupnú kapacitu siete a začali posielať maximálnou možnou rýchlosťou dohodnutou pri nadviazaní spojenia. Pomalý štart rieši tento problém. Na začiatku pošle len jeden paket a čaká na jeho potvrdenie *ACK* (teda posielacie okno *wnd* má veľkosť 1). Ďalej pracuje nasledovne: pre každú doručenú *ACK* vyšle do siete dva nové pakety. Ako si môžeme všimnúť, nárast počtu paketov sa zvyšuje exponenciálne. Toto sa však deje len do určitej hranice, kedy sa dosiahne kapacita siete. Potom nastáva zahadzovanie paketov a je rozumné znížiť posielaciu rýchlosť (pôvodne na hodnotu *wnd* 1, neskôr sa ukázalo, že je rozumné uvažovať zníženie len o nejakú časť).

Pomalý štart je svojím spôsobom mechanizmus na prvotné zistenie voľnej kapacity linky.

2.3.2 *Predchádzanie zahlteniu (Congestion avoidance)*

Algoritmus Predchádzania zahlteniu má za úlohu kontrolu toku tak, aby nedošlo k zahlteniu. Úzko spolupracuje s algoritmom Pomalého štartu. Cieľom je zníženie rýchlosti posielania paketov do siete z exponenciálneho na lineárne. Toto sa deje po prekročení určitej hranice *sstreshd* na začiatku pevne stanovenej a počas behu algoritmu počítanej po každej strate ako polovica maximálneho posielacieho okna *wnd* pred stratou.

Nasledujúce dva algoritmy vylepšujú funkcionalitu algoritmov Pomalého štartu a Predchádzania zahlteniu. Navrhol ich pán van Jacobson v roku 1990. Využívajú doručenie *dupACK* správ a ich informáciu o preusporiadaní alebo strate nejakého paketu.

2.3.3 *Rýchle preposielanie (Fast retransmit)*

Predchádzajúce algoritmy považujú doručenie potvrdenia mimo poradia *dupACK* za signál straty paketu. Algoritmus Rýchleho preposlania si naopak *dupACK* môže vyložiť aj ako signál preusporiadania paketov v sieti. Algoritmus uvažuje stratu paketu až po doručení troch *dupACK*. Až vtedy dôjde

k preposlaniu strateného paketu. Týmto opatrením sa vyhneme zbytočnému preposielaniu paketov, ktoré by inak došli do cieľa, ale v inom poradí.

2.3.4 *Rýchle zotavenie (Fast recovery)*

Algoritmus rýchleho zotavenia zvažuje mieru zmenšovania posielacej rýchlosti po strate paketu. Podľa pôvodnej špecifikácie, po zistení straty znížime veľkosť posielacieho okna *wnd* na 1 paket. Strata paketu zistená doručením troch *dupACK* nám poukazuje na plnenie siete, ale samotné ich doručenie dokazuje, že stále ešte prúdia dáta sieťou. Preto je rozumné uvažovať len zníženie *wnd* na nejakú hranicu (tu sa uvažuje *wnd/2*). Po takomto znížení sa pokračuje fázou Predchádzanie zahlteniu s lineárnym nárastom. Pre prípad vypršania časovača sa nič nemení, teda nastáva pokles na minimálnu hodnotu.

2.3.5 *Implementácia Kontroly zahltenia*

Kontrola tokov je štandardnou súčasťou TCP. Jej funkcionalita je riadená premennou *snd.wnd*, ktorá znamená veľkosť okna, ktoré je schopné prijímateľ spracovať. Táto premenná riadi kontrolu rýchlosti posielania medzi prijímateľom a odosielateľom.

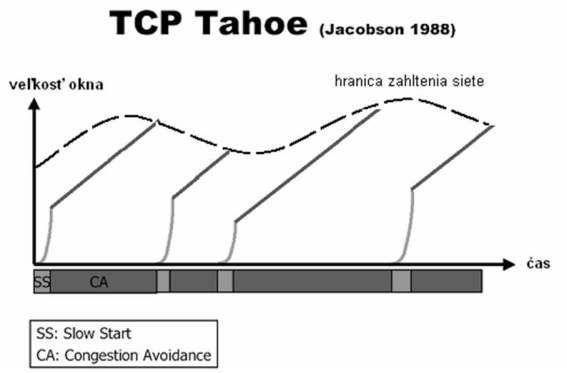
Kontrola zahltenia je implementovaná oknom zahltenia *cwnd* (congestion window). Na rozdiel od *snd.wnd*, *cwnd* sa zaoberá mierou zahltenia siete. Veľkosť *cwnd* by mala odzrkadľovať voľnú kapacitu siete a mala by sa dynamicky meniť podľa miery zahltenia.

Efektívne posielacie okno resp. len Posielacie okno *wnd*, definované ako $wnd = \min(snd.wnd, cwnd)$, nám určuje maximálnu rýchlosť posielania (maximálny počet prenášaných paketov). Takto definované *wnd* nám zaručuje dodržanie obmedzení kladených sieťou aj príjemcom.

Ako sme už spomenuli pri algoritme Predchádzanie zahlteniu, je potrebné definovanie *sstreshd* (slow start threshold - medzná hranica pomalého štartu), ako hranice zmeny z exponenciálneho nárastu na lineárny.

Algoritmus Kontroly zahltenia využívajúci Pomalý štart a Predchádzanie zahlteniu (vychádzame z [1, 4]):

- Inicializácia
  1.) $cwnd = 1$ (paket)
  2.) $sstreshd = 65535/\text{MSS}$ (paketov, kde MSS je maximum segment size)
- telo
  1.) $wnd = \min(snd.wnd, cwnd)$ (zaručíme sa, že nikdy nepošleme viac ako *wnd* dát)



Obrázok 2.2: Kontrola premávky u TCP Tahoe (1988).

2.) po zistení straty paketu (vypršanie časovača alebo $dupACK$)
  2.1.) $sstresh = wnd/2$ (minimálne 2 pakety)
  2.2.) $cwnd = 1$ (paket)
3.) pri obdržaní potvrdenia ACK
  3.1.) ak $cwnd < sshtresh$
    3.1.1.) nachádzame sa vo fáze pomalého štartu
    3.1.2.) nárast o $cwnd = cwnd + 1$
  3.2.) ak $cwnd \geq sshtresh$
    3.2.1.) nachádzame sa vo fáze predchádzania zahlteniu
    3.2.2.) nárast o $cwnd = cwnd + 1/cwnd$

Kontrolu zahltenia využívajúcu Pomalý štart a Predchádzanie zahlteniu má v sebe implementovaný napr. TCP Tahoe. Priebeh kontroly zahltenia je znázornený na obrázku 2.2.

Algoritmus Kontroly zahltenia využívajúci Rýchle preposlanie a Rýchle zotavenie (vychádzame z [1]):

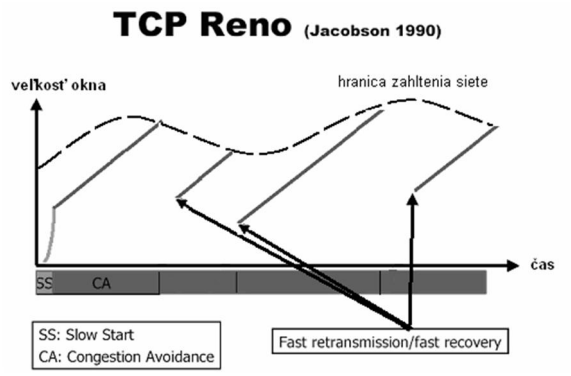
Tu uvažujeme premennú *isSS*, ktorá je pravdivá, ak máme ísť do fázy Pomalého štartu (po obdržaní troch a viacej $dupACK$ alebo vypršaní časovača - štandardná implementácia) a nepravdivá, ak máme pokračovať fázou Predchádzanie zahlteniu (po obdržaní jednej alebo dvoch $dupACK$).

- Inicializácia

  1.) $cwnd = 1$ (paket)

  2.) $sstreshd = 65535/\text{MSS}$ (paketov, kde MSS je maximum segment size)

  3.) $isSS = \text{true}$



Obrázok 2.3: Kontrola premávky u TCP Reno (1990).

- telo

  1.) $wnd = \min(snd.wnd, cwnd)$ (zaručíme sa, že nikdy nepošleme viac ako wnd dát)

  2.) po zistení straty paketu

    2.1.) po doručení jednej alebo dvoch $dupACK$

      2.1.1.) $cwnd = cwnd/2$

      2.1.2.) $isSS =$ false

    2.2.) po doručení troch a viac $dupACK$ alebo vypršaní časovača

      2.1.) $sstresh = wnd/2$ (minimálne 2 pakety)

      2.2.) $cwnd = 1$ (paket)

      2.3.) $isSS =$ true

  3.) pri obdržaní potvrdenia ACK

    3.1.) ak $cwnd < sshtresh$ a $isSS$ je pravdivé

      3.1.1.) nachádzame sa vo fáze pomalého štartu

      3.1.2.) nárast o $cwnd = cwnd + 1$

    3.2.) ak $cwnd \geq sshtresh$ alebo $isSS$ je nepravdivé

      3.2.1.) nachádzame sa vo fáze predchádzania zahlteniu

      3.2.2.) nárast o $cwnd = cwnd + 1/cwnd$

Kontrolu zahltenia využívajúcu Rýchle preposlanie a Rýchle zotavenie ako pozmenené algoritmy Pomalý štart a Predchádzanie zahlteniu má v sebe implementovaný napr. TCP Reno. Priebeh kontroly zahltenia je znázornený na obrázku 2.3.

2.4 *Model TCP*

V nasledujúcej časti si bližšie rozoberieme fungovanie štandardného TCP s implementovanou kontrolou zahltenia. Poznatky odvodené v tejto časti

budú využívané aj v ďalších kapitolách na poukázanie problémov protokolu v špecifických situáciách.

Najskôr si povieme niečo o spôsoboch zisťovania zahltenia v štandardných implementáciách TCP (implementáciách nad štandardnými sieťami ako sú Tahoe, Reno, Vegas atď.), potom definujeme potrebné pojmy. Nakoniec sa pozrieme na rovnicu stabilného stavu TCP a Dynamiku.

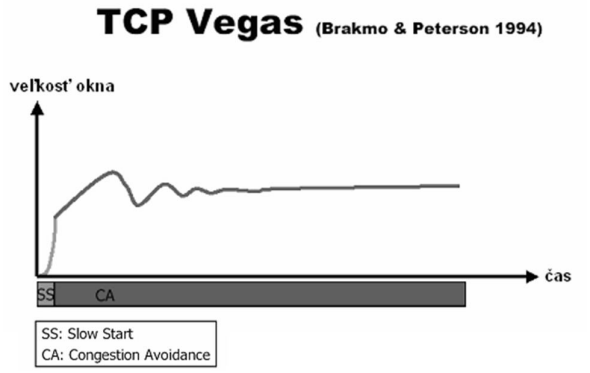
2.4.1 *Spôsoby zisťovania zahltenia*

Pôvodné TCP (za predpokladu spoľahlivej linky), ako je napríklad TCP Reno [28], stratu paketu používa ako signál zahltenia niekde v sieti. Každá strata paketu zistená prijatím viacerých *dupACK* alebo vypršaním časovača, znamená, že do siete vstupuje viac dát, ako je sieť schopná preniesť. Vtedy je rozumné zníženie posielacej rýchlosti a TCP použije Kontrolu zahltenia. Tento spôsob zistenia zahltenia nazývame *Loss based* (*Založený na strate* - ale v ďalšom budeme používať anglický termín).

Čo je paradoxné, Loss based algoritmy samy zvyšujú rýchlosť posielania paketov, kým nevyvolajú zahltenie. Vtedy dôjde k strate paketu, ktorú si TCP vysvetlí ako signál zahltenia a zníži rýchlosť.

TCP Vegas [22] prišlo s myšlienkou, ako rozšíriť loss based systém. Nazvané je *Delay based* alebo *Queueing delay based* (*Založený na zdržaní* - ale v ďalšom budeme používať anglický termín). Cieľom je znížiť, ak nie úplne odstrániť umelé vyvolávanie zahltení zvyšovaním rýchlosti. Tu sa snažíme stratám predísť tým, že monitorujeme a porovnávame vysielacie rýchlosti a čas potrebný na prenos paketov - RTT. Podľa toho zisťujeme, či vchádzame do zahltenia alebo nie. Toto riešenie je na rozdiel od predchádzajúceho proaktívne (preventívne). Nevyvoláva stratu aby vedel, že už presiahol kapacitu siete a má spomaliť, ale naopak, keď zisťuje, že sieť sa začína plniť, postupne spomaľuje k hranici pod úplným využitím siete (hodnota menšia ako maximálna kapacita siete). Tu je problém s určením hraničnej hodnoty, do ktorej sa zväčšuje, aby nebola príliš malá (v porovnaní s vypočítanou maximálnou kapacitou), čo by viedlo k nevyužitiu a neflexibilnosti, a ani príliš veľká, čím by sa zmenšil voľný priestor medzi hranicou a maximálnou veľkosťou najužšej linky a tým by bola pravdepodobnosť zahltenia väčšia. Zároveň je aj rozdielne narábanie s časovačmi a *dupAck*-ami.

Delay based systém vychádza z porovnávania RTT. Na nezahltenej sieti sa predpokladá rovnaký čas RTT pre pakety vyslané do siete. Hneď ako sa sieť začne plniť, posielané pakety začnú čoraz viac obsadzovať vyrovnávacie pamäte spojovacích uzlov. Čakanie vo vyrovnávacích pamätiach stojí nejaký čas, ktorý sa potom odzrkadlí do konečnej hodnoty RTT. Čím je sieť viac zahltená, tým viac paketov čaká vo vyrovnávacích pamätiach na vybavenie a tým sa zväčšujú RTT pre dané pakety. No keďže nedošlo ešte k zahodeniu paketu z vyrovnávacej pamäte spojovacieho uzla, loss based systém sa o to nestará. Delay based však z meraní vie určiť blížiace sa zahltenie a zníži



Obrázok 2.4: Kontrola premávky u TCP Vegas (1994).

vysielaciu rýchlosť. Keď sa naopak ukáže z meraní RTT, že zahltenie ustupuje, zvýši posielaciu rýchlosť. Vďaka tejto plnohodnotnejšej informácii sme schopní udržiavať linky takmer využité. Tu musíme nájsť vhodnú posielaciu rýchlosť, okolo ktorej budeme oscilovať tak, aby sme čo najviac využívali kapacitu linky. Zároveň musíme mať dostatočnú rezervu na to, aby nedošlo k zbytočnej strate paketov, keď presiahneme vhodnú posielaciu rýchlosť, a kým ju naspäť neznížime.

Princíp fungovania delay-based systému u TCP Vegas je zobrazený na obrázku 2.4. Môžeme si všimnúť, že rozšírenie namiesto umelého vyvolávania strát osciluje okolo zvolenej veľkosti okna. Tú si v každom RTT prepočítava, čím dynamicky reaguje na zmeny v sieti.

Síce sa v Delay based systémoch snažíme predísť stratám paketov, niekedy sa im nevyhneme. Preto musia mať tieto systémy v sebe zahrnuté aj Loss based zisťovanie zahltenia.

Aj keď sa môže zdať, že zisťovanie zahltení delay-based princípom, ako je u Vegas rozšírení, je lepšie z pohľadu využívania siete ako u loss based, aj tu sú dôvody, prečo sa používajú oba. S nasadením Vegas rozšírení je potrebné brať do úvahy prostredia, ktoré pôsobia rušivo na delay-based princíp. Tu sa predpokladá, že pozdržania sú z dôsledku plnenia vyrovnávacích pamätí. Vtedy dochádza k stavu, kedy sa sieť stáva zaplnenou. No zdržania môžu nastať aj z iných dôvodov, akými sú bufferovanie (radenie do vyrovnávacích pamätí) na heterogénnych schémach, plánovanie operačnými systémami, spracovávanie prichádzajúcich dát firewallmi a súčasná premávka iných protokolov, ktorá nekontroluje zdržanie vo vyrovnávacích pamätiach alebo nedodržuje zásady predchádzania zahlteniu. Tým si môže algoritmus zle vysvetliť prijaté informácie a zahájiť kroky, ktoré sú neefektívne.

Podrobnejšie si rozoberieme TCP Vegas a niektoré ďalšie Delay based protokoly v kapitole **4. Rozšírenia TCP pre vysokorýchlostné siete**, lebo ako sa ukázalo, Delay based systémy sa lepšie správajú na sieťach s veľkou kapacitou (resp. sieťach, kde je veľká kapacita linky a neúmerne

menšie vyrovnávacie pamäte v spojovacích uzloch).

2.4.2 *Premenné a parametre*

V tejto časti si zadefinujeme niektoré pojmy, ktoré budeme v ďalšom texte používať. V rôznych literatúrach sa rovnaké veci označujú rôzne. Preto sme zvolili najčastejšie spôsoby zápisu.

- $w_i(t)$ je veľkosť okna i-teho toku v čase t, w_i je priemerná veľkosť okna i-teho toku (uvažovaná od začiatku spojenia)
- $q_i(t)$ je miera zahltenia. Môže to byť stratovosť (u Loss based) pre daný i-ty tok v čase t alebo pozdržanie paketu na ceste (u Delay based) pre i-ty tok v čase t. Zápisom q_i stanovíme priemernú mieru zahltenia (berieme od začiatku spojenia).
- $T_i(t)$ je RTT (Round Trip Time) teda čas od poslania paketu po prijatie jeho ACK (pre i-ty tok), podobne T_i je priemerná dĺžka RTT (uvažovaná od začiatku spojenia)
- α alebo niekedy aj a je prírastkový faktor protokolu - riadi rast rýchlosti posielania. V niektorých prípadoch, kedy je α funkciou, použijeme zápis $\alpha(x)$, kde x je parametrom funkcie α (hoci x môže byť ľubovoľná premenná alebo parameter, častokrát je to veľkosť okna *cwnd*).
- β alebo niekedy aj b je faktor poklesu protokolu - definuje zníženie rýchlosti posielania. V niektorých prípadoch, kedy je β funkciou, použijeme zápis $\beta(x)$, kde x je parametrom funkcie β (hoci x môže byť ľubovoľná premenná alebo parameter, častokrát je to veľkosť okna *cwnd*).
- $x_i(t)$ je rýchlosť posielania paketov za čas (pakets per second) definované ako $x_i(t) = \frac{w_i(t)}{T_i(t)}$. Pre priemernú rýchlosť posielania x_i platí $x_i = \frac{w_i}{T_i}$.
- W je maximálna veľkosť okna dosiahnuteľná na linke/ceste. Určuje hornú hranicu priepustnosti linky/cesty.
- p je miera zahltenia prislúchajúca linke (a teda aj W). Táto veličina určuje pre dané rozšírenie (v stabilnom stave) pravdepodobnosť zahodenia paketu v rámci jedného cyklu. Pre prehľadnosť si ju nazveme *vynútená stratovosť* linky/siete.

Ak to bude jasné z kontextu, značenie indexu i z predchádzajúcich premenných môžeme vynechať.

Ďalej ešte spomenieme niektoré funkcie, ktoré si v ďalšom texte zadefinujeme.

- $D_i(t)$ je Dynamika i-teho toku v čase t. V rôznych literatúrach je Dynamika značená rôzne, napr. $w_i(t)$, $\dot{w}_i(t)$. My sme pre jednoduchosť zvolili značenie $D_i(t)$.
- $k_i(t)$ je nárastová funkcia (gain function)
- $u_i(t)$ je prírastková užitočnosť (marginal utility function)

Zadefinujme si pojem *cyklus*. *Cyklom* nazveme obdobie medzi dvoma stratami. Teda je to obdobie, ktoré v sebe zahŕňa stratu paketu a následný pokles veľkosti posielacieho okna *wnd*, jeho postupný nárast, až po nasledujúcu stratu.

Miera zahltenia $q_i(t)$ nám určuje, ako je sieť zaplnená v čase t, teda aká je pravdepodobnosť, že pakety budú v čase t niekde po ceste zahodené. Keďže stále predpokladáme spoľahlivú linku, tak jediné straty sú z dôvodu zahodenia paketov v spojovacích uzloch. Pre loss based systémy, celkovú mieru zahltenia (end-to-end marking probability) $q_i(t)$ dostávame ako súčet čiastkových pravdepodobností zahodenia paketu v uzloch, cez ktoré komunikácia prechádza. Pre delay based systémy je to súčet zdržaní v jednotlivých spojovacích uzloch.

Vynútená stratovosť p nám vyjadruje pravdepodobnosť straty paketu v cykle (pomer stratených paketov ku všetkým poslaným) pre dané rozšírenie a teda závisí od W.

2.4.3 *AIMD a MIMD*

V predchádzajúcich kapitolách sme spomínali, že štandardné TCP (s implementovanou Kontrolou zahltenia, napr. TCP Reno) po každom úspešnom potvrdení paketu zvýši veľkosť posielacieho okna *wnd* o nejaký konštantný faktor (pre TCP Reno je to nárast o $\frac{1}{cwnd}$ pre každý paket okna veľkosti *cwnd* paketov) a po strate paketu ho zníži o nejaký násobok (uvažujeme priebeh po Pomalom štarte, samotný Pomalý štart nám len zaručuje rýchle dosiahnutie maximálnej kapacity siete). Teda nárast je lineárny a pokles multiplikatívny (pokles o nejaký násobok okna *wnd*). Tento systém nazývame *AIMD* (aditívny nárast, multiplikatívny pokles - additive increase, multiplicative decrease).

Niektoré rozšírenia TCP uvažujú pozmenené algoritmy Kontroly zahltenia. Druhou rozšírenou skupinou popri AIMD sú *MIMD* algoritmy. MIMD (multiplikatívny nárast, multiplikatívny pokles - multiplicative increase, multiplicative decrease) na rozdiel od AIMD má nárast o nejaký násobok okna *wnd*, teda samotný nárast nezáleží len od doručenia *ACK*, ale aj od veľkosti aktuálneho *wnd*. Pokles ako u AIMD je tiež multiplikatívny.

V praxi sa môžu vyskytnúť mnohé iné spôsoby riešenia Kontroly zahltenia. V ďalšej kapitole si spomenieme niektoré z nich ako riešenie problémov štandardného TCP na vysokorýchlostných sieťach.

2.4.4 *Stabilný stav*

Stabilným stavom protokolu nazývame stav, kedy protokol maximálne využíva prostriedky siete, pričom podmienky v sieti sa nemenia. Uvažujeme teda prostredie, nad ktorým prebieha komunikácia jedného alebo viacerých tokov, žiadne nové toky nepribúdajú, žiadne existujúce nekončia, a predpokladáme „stabilné“ rozdelenie kapacity siete (liniek) medzi existujúce toky. Každý tok má svoju časť kapacity siete, nad ktorou vysiela.

Tento stav je dosiahnuteľný, vyplýva zo správania sa algoritmov Kontroly zahltenia. Takéto toky majú tendenciu po určitom čase dosiahnuť posielaciu rýchlosť, okolo ktorej oscilujú. Pre jednoduchosť budeme uvažovať jeden tok v sieti. Ten bude po určitom čase (pokiaľ dosiahne maximálnu kapacitu siete) oscilovať medzi maximálnym oknom W a oknom, ktoré dosiahne po strate znížením posielacieho okna *wnd*. Ďalej budeme predpokladať, že prijímateľ je schopný spracovať viacej dát ako sieť, teda $snd.wnd > cwnd$ a budeme uvažovať len posielacie okno veľkosti *cwnd*.

Pre stabilný stav uvažujeme takzvanú *Reakčnú funkciu* (Response function). Podľa [30], je to veľkosť okna zahltenia *cwnd* daného rozšírenia (určeného α a β) v stabilnom stave určená vynútenou stratovosťou p. Pre TCP Reno a jemu podobné protokoly má tvar

$$cwnd = \frac{\sqrt{\alpha \frac{2-\beta}{2\beta}}}{\sqrt{p}} \tag{2.1}$$

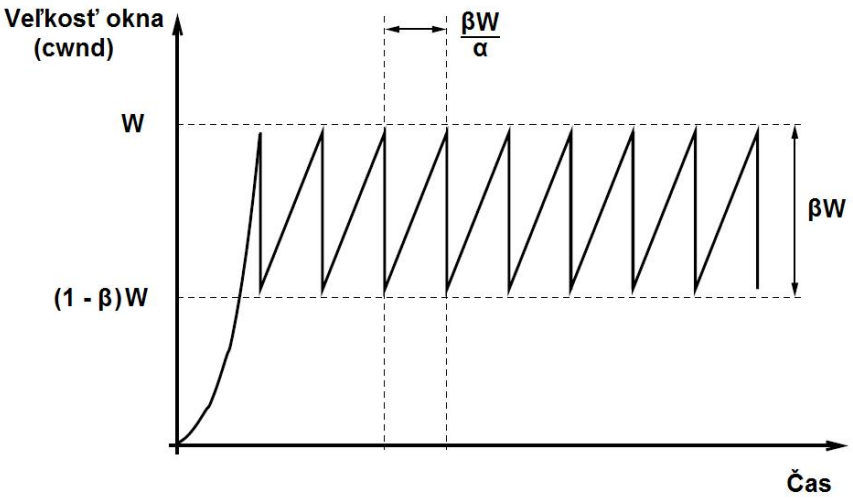
kde p je vynútená stratovosť, α a β sú parametre rozšírenia. Niekedy sa Reakčná funkcia uvádza aj ako posielacia rýchlosť (v paketoch za RTT).

Ak uvažujeme len jednu linku, ktorej kapacitu poznáme (W) a jeden tok, má zmysel vyjadriť Reakčnú funkciu ako funkciu maximálneho okna W. Pre AIMD, toto vyjadrenie vychádza zo závislosti *cwnd* od maximálneho okna W (v nasledujúcej časti si ukážeme vyjadrenie tohoto vzťahu - (2.3)) a má tvar

$$cwnd = \frac{2-\beta}{2} W \tag{2.2}$$

Tento vzťah vyjadruje priemernú veľkosť *cwnd* v stabilnom stave na linke s maximálnou kapacitou W paketov. Problém s takýmto vyjadrením je na sieti ako je napr. Internet, kde nepoznáme akými linkami bude naša komunikácia prebiehať, teda nepoznáme W. Vieme ale jednoducho vypočítať vynútenú stratovosť p ako pomer stratených paketov ku všetkým poslaným (napr. v rámci jedného cyklu). Ďalej budeme používať tvar Reakčnej funkcie ako je uvedený v (2.1).

Uvažujeme vynútenú stratovosť p v jednom cykle stabilného stavu, potom okno zahltenia *cwnd* počítané Reakčnou funkciou je priemerným v rámci jedného cyklu v stabilnom stave. Keďže predpokladáme stabilný stav, cykly sú v ňom rovnaké, a *cwnd* je priemerným oknom počas celého stabilného



Obrázok 2.5: Priebeh AIMD modelu a jeho správanie sa v stabilnom stave.

stavu. Inak povedané Reakčná funkcia nám dáva informáciu o priemernej veľkosti *cwnd* v stabilnom stave.

Reakčná funkcia vo všeobecnosti určuje závislosť veľkosti okna zahltenia *cwnd* od vynútenej stratovosti p vzhľadom na parametre α a β. V praxi sa používa aj pri návrhu rozšírení na stanovenie závislosti parametrov α a β. Jej tvar nám môže prezradiť o protokole veľa vlastností akými sú TCP priateľskosť, férovosť (z pohľadu RTT), konvergenciu k stabilnému stavu na linke a prispôsobivosť.

Odvodenie stabilného stavu pre AIMD

AIMD je charakterizované multiplikatívnym zmenšením posielacieho okna o nejaký konštantný násobok a pri pozitívnom potvrdení zväčšením okna o nejakú konštantnú časť. Tento mechanizmus je zobrazený na obrázku 2.5. Pripomeňme, že W určuje maximálnu veľkosť okna (dosiahnuteľnú v sieti) a parametre α a β závisia od zvoleného rozšírenia (pre náš výpočet sú fixné).

Na začiatku cyklu začíname s veľkosťou okna $(1-\beta)W$ a na konci cyklu máme okno veľkosti W, zisťujeme stratu paketu a klesneme opäť na $(1-\beta)W$. Každý cyklus obsahuje $\frac{\beta}{\alpha}W + 1$ RTT. Toto je znázornené na obrázku 2.5.

Stanovme priemernú veľkosť okna zahltenia (v paketoch) *cwnd*

$$cwnd = \frac{(1-\beta)W + W}{2} = \frac{2-\beta}{2}W \tag{2.3}$$

Zo vzťahu (2.3) a počtu RTT v cykle, môžeme vypočítať počet paketov poslaných v jednom cykle a to nasledovne

$$\left(\frac{\beta}{\alpha}W+1\right)cwnd = \left(\frac{\beta}{\alpha}W+1\right)\frac{2-\beta}{2}W$$
$$\approx \frac{\beta(2-\beta)}{2\alpha}W^2 \tag{2.4}$$

Toto v sebe zahŕňa aj jeden stratený paket. Pravdepodobnosť straty (vynútená stratovosť p) je (použitím (2.4))

$$p \approx \frac{1}{\frac{\beta(2-\beta)}{2\alpha}W^2} = \frac{2\alpha}{\beta(2-\beta)W^2} \tag{2.5}$$

Vyjadrením W z (2.5) dostávame závislosť veľkosti maximálneho okna W od pravdepodobnosti straty paketu

$$W \approx \sqrt{\frac{2\alpha}{\beta(2-\beta)p}} \tag{2.6}$$

Dosadením rovnice (2.6) do (2.3) dostávame našu rovnicu Reakčnej funkcie (2.1)

$$T = \frac{2-\beta}{2}W = \frac{2-\beta}{2}\sqrt{\frac{2\alpha}{\beta(2-\beta)p}}$$
$$= \frac{\sqrt{\alpha\frac{2-\beta}{2\beta}}}{\sqrt{p}}$$

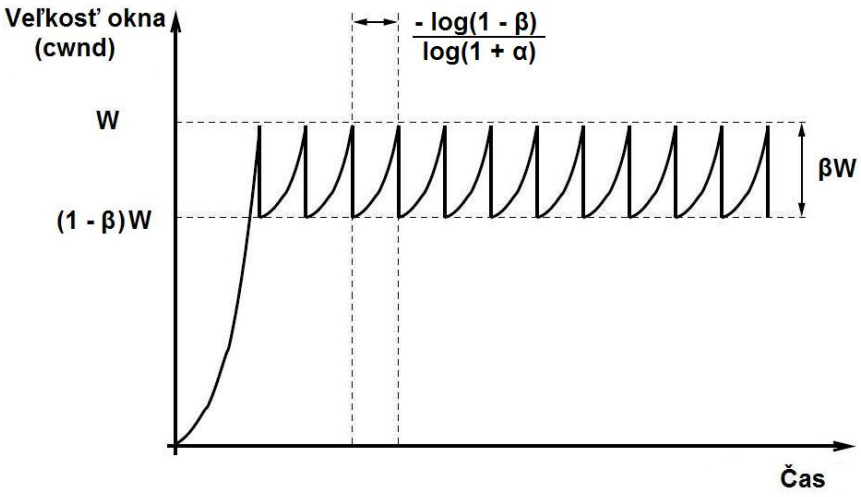
Takto upravená Reakčná funkcia už nie je závislá od znalosti maximálnej kapacity W najužšej linky na ceste, cez ktorú prebieha komunikácia, ale od vynútenej stratovosti p, ktorú vieme vypočítať u odosielateľa.

Odvodenie stabilného stavu pre MIMD

Výpočet rovnice Reakčnej funkcie u MIMD je rozdielny. Vyplýva to z faktu, že pri pozitívnom potvrdení zvyšujeme multiplikatívne. Priebeh MIMD je zobrazený na obrázku 2.6.

Podobne ako pri AIMD aj tu každý cyklus začíname s veľkosťou okna $(1-\beta)W$, na konci cyklu dosiahneme veľkosť W a končíme po strate s veľkosťou $(1-\beta)W$ (závisle od dosiahnutého W, ale predpokladáme, že v stabilnom stave bude rovnaké). Je to znázornené na obrázku 2.6.

Najskôr si vypočítame počet RTT v jednom cykle. Keďže pri MIMD je nárast multiplikatívny, na tento výpočet použijeme vzorec pre výpočet n-tého člena geometrickej postupnosti $a_n = a_1 k^{n-1}$. Po dosadení W za a_n-tý



Obrázok 2.6: Priebeh MIMD modelu a jeho správanie sa v stabilnom stave.

člen, $(1-\beta)W$ za a_1-tý člen a $1+\alpha$ za k (keďže ide o nárast a α je zadávané ako časť okna teda číslo menšie ako 1, musíme k nemu pripočítať 1, aby sme dostali koeficient nárastu)

$$\begin{aligned} W &= (1-\beta)W(1+\alpha)^{n-1} \\ \frac{1}{1-\beta} &= (1+\alpha)^{n-1} \\ \log\left(\frac{1}{1-\beta}\right) &= (n-1)\log(1+\alpha) \end{aligned}$$

Z toho dostávame počet RTT v cykle

$$n = \frac{\log(\frac{1}{1-\beta})}{\log(1+\alpha)} + 1 \approx \frac{-\log(1-\beta)}{\log(1+\alpha)} \tag{2.7}$$

Počet paketov v cykle vypočítame zo vzťahu pre súčet geometrického radu $S_n = a_0\frac{k^n-1}{k-1}$, kde za a_0 dosadíme okno po strate $(1-\beta)W$, za k dáme $1+\alpha$ a n vyjadríme podľa vzťahu (2.7). Dostávame

$$S_n = (1-\beta)W\frac{(1+\alpha)^{-\frac{\log(1-\beta)}{\log(1+\alpha)}} - 1}{(1+\alpha)-1} = (1-\beta)W\frac{(1+\alpha)^{-\frac{\log(1-\beta)}{\log(1+\alpha)}} - 1}{\alpha} \tag{2.8}$$

paketov, z toho sa jeden stratí a tak zisťuje zahltenie. Strata paketu má pravdepodobnosť (vynútená stratovosť) jedna k počtu prenesených paketov

$$p = \frac{1}{(1-\beta)W\frac{(1+\alpha)^{-\frac{\log(1-\beta)}{\log(1+\alpha)}}-1}{\alpha}} = \frac{\alpha}{(1-\beta)W\left((1+\alpha)^{-\frac{\log(1-\beta)}{\log(1+\alpha)}}-1\right)} \quad (2.9)$$

Vyjadrením W zo vzťahu (2.9) dostávame závislosť veľkosti maximálneho okna W od pravdepodobnosti straty paketu

$$W = \frac{\alpha}{(1-\beta)p\left((1+\alpha)^{-\frac{\log(1-\beta)}{\log(1+\alpha)}}-1\right)} \quad (2.10)$$

Priemernú veľkosť okna zahltenia *cwnd* v cykle vypočítame, ako počet paketov v cykle zo vzťahu (2.8) delený počtom RTT v cykle zo vzťahu (2.7) a dostávame

$$cwnd = \frac{S_n}{n} = \frac{(1-\beta)W\frac{(1+\alpha)^{-\frac{\log(1-\beta)}{\log(1+\alpha)}}-1}{\alpha}}{\frac{-\log(1-\beta)}{\log(1+\alpha)}} \quad (2.11)$$

Dosadením (2.10) do vzťahu (2.11) dostávame vyjadrenie *cwnd* v závislosti od vynútenej stratovosti p

$$\begin{aligned} cwnd &= \frac{(1-\beta)\frac{\alpha}{(1-\beta)p\left((1+\alpha)^{-\frac{\log(1-\beta)}{\log(1+\alpha)}}-1\right)}\frac{(1+\alpha)^{-\frac{\log(1-\beta)}{\log(1+\alpha)}}-1}{\alpha}}{\frac{-\log(1-\beta)}{\log(1+\alpha)}} \\ &= -\frac{\log(1+\alpha)}{\log(1-\beta)p} \end{aligned} \quad (2.12)$$

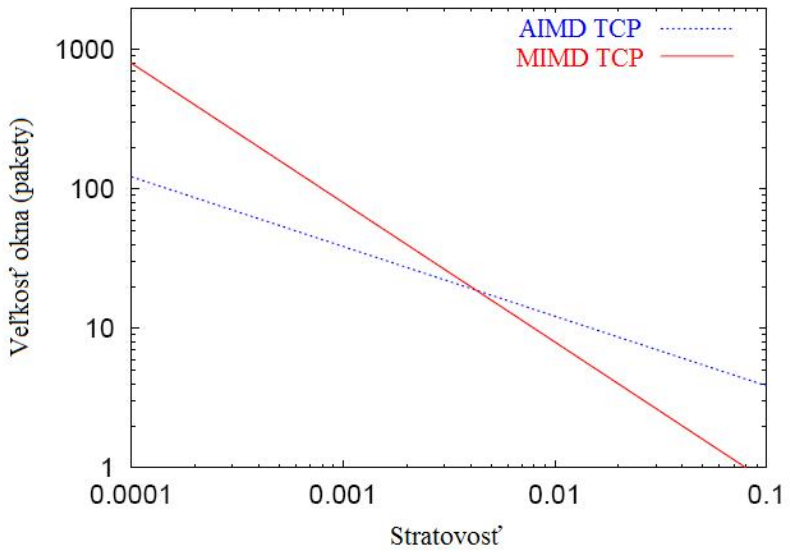
Podobne ako u AIMD, aj tu uprednostňujeme vypočítanie vynútenej stratovosti p pred znalosťou veľkosti W pre najužšiu linku, cez ktorú ide komunikácia.

Pri snahe zjednodušiť vzťah (2.12) a výber parametrov α a β sme použili substitúciu (napr. ako v [21])

$$cwnd \approx \frac{\alpha'}{\beta' p} \quad (2.13)$$

kde $\alpha' = \log(1+\alpha)$ a $\beta' = -\log(1-\beta)$.

Samotný vzťah (2.13) nám reprezentuje Reakčnú funkciu MIMD v stabilnom stave.



Obrázok 2.7: Porovnanie stabilných stavov AIMD (TCP Reno) a MIMD (Scalable TCP).

Porovnanie rovníc stabilného stavu pre AIMD a MIMD

Rovnice (2.1) a (2.13) nám ukazujú rozdielnosť Reakčných funkcií pre modely AIMD a MIMD. Parametre α a β sú určené danými rozšíreniami a ovplyvňujú nárast a pokles. Problémom je však rozdielna pravdepodobnosť straty (vynútená stratovosť p) a jej vplyv na správanie sa protokolu. To spôsobuje rozličné správanie sa týchto systémov v rovnakých podmienkach (tým sa myslí pri rovnakej vynútenej stratovosti p alebo z nej vyplívajúcej maximálnej kapacity najužšej linky W).

Obrázok 2.7 nám zobrazuje závislosť veľkosti okna *cwnd* a vynútenej stratovosti p v stabilnom stave. Pre veľké okná (a z toho vyplývajúcu menšiu vynútenú stratovosť) majú MIMD výhodu, lepšie sa adaptujú na linky s väčšou kapacitou a sú výrazne efektívnejšie ako AIMD. Naopak AIMD sú vhodnejšie a efektívnejšie pre malé okná (a teda veľkú vynútenú stratovosť). Preto väčšina MIMD protokolov sa pre malé okná správa ako AIMD a po prekročení určitej hranice ako MIMD protokol.

2.4.5 *Dynamika*

Dynamika alebo Dynamická funkcia (po anglicky Dynamics) podľa [23] je funkcia, ktorá charakterizuje dané rozšírenie. Jej všeobecný tvar je

$$D_i(t) = k_i(t)\left(1 - \frac{q_i(t)}{u_i(t)}\right) \tag{2.14}$$

kde $k_i(t)$ je nárastová funkcia (gain function) a $u_i(t)$ je prírastková užitočnosť (marginal utility function). $q_i(t)$ za predpokladu, že neuvažujeme explicitné informácie a potvrdzovania zo strany prvkov siete (spojovacích uzlov), je miera zahltenia. Tá podľa princípu zisťovania zahltenia môže byť definovaná ako vynútená stratovosť alebo miera zdržania dát na ceste.

Pomocou Dynamiky môžeme opísať model algoritmu zahltenia u protokolov. Dokonca u vysokorýchlostných sietí si rozšírenia berú Dynamiku ako predmet skúmania a tú potom menia, alebo presnejšie povedané menia funkcie k_i a u_i. Všeobecný tvar je preto rovnaký, zdedený z TCP Rena, a ostatné rozšírenia ho využívajú vhodne upravený.

Všeobecný model Dynamiky definovaný (2.14) rozdeľuje správanie sa rozšírenia na dve časti. Prvou je nárastová funkcia k_i, ktorá ovplyvňuje vlastnosti ako stabilita, zodpovednosť, ale priamo neovplyvňuje stabilný stav. Druhou je prírastková užitočnosť u_i, ktorá naopak určuje správanie sa daného rozšírenia v stabilnom stave. To sú vlastnosti ako rýchlosť/priepustnosť v stabilnom stave, dosiahnutie stabilného stavu, férovosť, TCP priateľskosť a iné.

Cieľom rozšírení je vyrovnanie prírastkovej užitočnosti $u_i(t)$ s mierou zahltenia $q_i(t)$, čím by sa rozšírenie dostalo do stabilného stavu (stavu, kedy sa nič v sieti nemení a teda prírastok k zmene okna je nulový). Inak povedané, $u_i(t)$ sa snaží aproximovať mieru zahltenia $q_i(t)$. To vedie k snahe rozšírení stanoviť zmenu (prírastok) okna zahltenia *cwnd* (window adjustment) na základe veľkosti vzdialenosti podielu $\frac{q_i(t)}{u_i(t)}$ od cieľovej hodnoty 1. Pre loss based systémy, to znamená porovnanie podielu $\frac{q_i(t)}{u_i(t)}$ s cieľovou hodnotou 1, a podľa výsledku (kladný resp. záporný) sa určí nárast/pokles (porovnávanie závislé len od *cwnd*). Delay based toto porovnávanie rozširuje o samotnú veľkosť nárastu/poklesu *cwnd*, podľa veľkosti rozdielu $\frac{q_i(t)}{u_i(t)}$ a cieľovej hodnoty 1. Chceme len poznamenať, že cieľová hodnota 1 vyplýva zo všeobecného tvaru Dynamiky a je to hodnota pomeru $\frac{q_i(t)}{u_i(t)}$ v stabilnom stave (kde predpokladáme rovnosť $q_i(t)$ a $u_i(t)$).

Ako $u_i(t)$ sa často volí vynútená stratovosť p, presnejšie povedané jej vyjadrenie z Reakčnej funkcie (pre AIMD je to $p = \alpha\frac{2-\beta}{2\beta w^2}$, pre MIMD $p = \frac{\alpha'}{\beta' w}$). To nám poskytuje pre dané rozšírenie postačujúcu informáciu o priemernej vynútenej stratovosti pre aktuálne okno w, ktorú porovnávame s aktuálnou mierou zahltenia $q_i(t)$. Z tohto porovnania už vieme určiť, či budeme zväčšovať okno (ak $\frac{q_i(t)}{u_i(t)} \leq 1$) alebo zmenšovať (ak $\frac{q_i(t)}{u_i(t)} > 1$).

Dynamiku môžeme chápať ako funkciu na výpočet zmeny okna zahltenia *cwnd* pre dané rozšírenie (teda pre stanovené α a β) v čase t. Dostávame ju ako rozdiel medzi nárastom a poklesom posielacieho okna. V stabilnom stave (teda pre priemerné RTT a priemerné okno) je dynamika konštantná. Podrobnejšie si výpočty Dynamík u konkrétnych rozšírení ukážeme v **Kapitole 4 Rozšírenia pre vysokorýchlostné siete**. Teraz si ukážeme všeobecné vyjadrenie Dynamiky pre AIMD a MIMD systémy.

Vyjadrenie Dynamiky u AIMD

Model všeobecnej Dynamiky algoritmu AIMD (obrázok 2.5) sa dá stanoviť nasledovne [38, 39]. Nech v čase t, i-ty odosielateľ vysiela rýchlosťou $x_i(t) = \frac{w_i(t)}{T_i(t)}$ paketov za sekundu. Za predpokladu bezstratovej spoľahlivej linky dostávame, že odosielateľovi sa vracajú potvrdenia na poslané pakety rýchlosťou $x_i(t - T_i(t))$. Nech $(1 - q_i(t))$ paketov je pozitívne potvrdených a $q_i(t)$ paketov sa zahodí v spojovacích uzloch - označíme ich ako negatívne potvrdené pakety. Každý pozitívne potvrdený paket zväčší okno $w_i(t)$ o $\frac{\alpha}{w_i(t)}$, teda celková zmena je $x_i(t - T_i(t))(1 - q_i(t))\frac{\alpha}{w_i(t)}$. Podobne negatívne potvrdenia sú prijímané priemernou rýchlosťou $x_i(t - T_i(t))q_i(t)$, pričom každé zmenší okno $w_i(t)$ o $\beta w_i(t)$, kde $\beta < 1$. Z toho dostávame vývoj okna pre AIMD model

$$D_i(t) \quad = \quad x_i(t - T_i(t))(1 - q_i(t))\frac{1}{w_i(t)} - x_i(t - T_i(t))q_i(t)w_i(t)\beta$$

kde $q_i(t)$ je miera zahltenia definovaná ako súčet vynútených stratovostí (pravdepodobnosť zahodenia paketu) spojovacích uzlov, cez ktoré prechádza komunikácia.

Tento model ešte môžeme zjednoduchšiť nasledovne. Keďže $q_i(t)$ je vždy malé, položíme $(1-q_i(t)) \approx 1$. Ďalej predpokladáme, že každý paket je potvrdený (pozitívne alebo negatívne - zisťuje sa strata), teda veľkosť poslaného okna sa rovná veľkosti prijatého okna ($w_i(t) = w_i(t - T_i(t))$).

Zároveň môžeme položiť $T_i(t) = T_i(t-T_i(t))$, lebo nová hodnota časovača sa upravuje až po doručení potvrdení a je primerane vyhľadená. Zároveň z pozorovania, že pokles okna nastáva, až keď okno presiahne maximálnu kapacitu W, môžeme použiť vzťah (2.3) na vyjadrenie maximálneho okna $W = \frac{2}{2-\beta}cwnd$ (kde $cwnd$ je naše $w_i(t)$) v závislosti od aktuálneho okna. Za týchto predpokladov dostávame Dynamiku (2.14), ako je definovaná v [23]

$$\begin{aligned} D_i(t) &= \frac{\alpha}{T_i(t)} - x_i(t)q_i(t)\beta W \\ &= \frac{\alpha}{T_i(t)} - x_i(t)q_i(t)\beta\frac{2}{2-\beta}w_i(t) \\ &= \frac{\alpha}{T_i(t)}\left(1 - \frac{q_i(t)}{\frac{\alpha(2-\beta)}{w_i(t)^2 2\beta}}\right) \\ &= k_i(t)\left(1 - \frac{q_i(t)}{u_i(t)}\right) \end{aligned} \tag{2.15}$$

kde posledný tvar je všeobecným modelom Dynamiky, pre $k_i(t) = \frac{\alpha}{T_i(t)}$ a $u_i(t) = \frac{\alpha(2-\beta)}{w_i(t)^2 2\beta}$.

Vyjadrenie Dynamiky u MIMD

Podobne ako u AIMD aj pri MIMD vychádzame z podobného postupu. Model všeobecného algoritmu MIMD (obrázok 2.6) stanovíme z nárastu okna *cwnd* pri potvrdení správy a poklesu pri strate. Nárast *cwnd* sa deje raz za RTT o faktor α a závisí aj od pôvodnej veľkosti *cwnd*. Z toho dostávame nárast o $\frac{\alpha w_i(t)}{T_i(t)}$. Pokles veľkosti okna *cwnd* pri zistenej strate je rovnaký ako u AIMD, keďže oba systémy používajú multiplikatívny pokles. Veľkosť poklesu je teda

$$x_i(t)q_i(t)\beta W = x_i(t)q_i(t)\beta\frac{2}{2-\beta}w_i(t) \tag{2.16}$$

kde $W = \frac{2}{2-\beta}w_i(t)$ je maximálne dosiahnuteľná veľkosť okna, $q_i(t)$ je miera zahltenia a $x_i(t) = \frac{w_i(t)}{T_i(t)}$ je rýchlosť posielania paketov.

Z toho vyplýva model Dynamiky pre MIMD

$$\begin{aligned} D_i(t) &= \frac{\alpha w_i(t)}{T_i(t)} - x_i(t)q_i(t)\beta W \\ &= \frac{\alpha w_i(t)}{T_i(t)} - x_i(t)q_i(t)\beta\frac{2}{2-\beta}w_i(t) \\ &= \frac{\alpha w_i(t)}{T_i(t)}\left(1 - \frac{q_i(t)}{\frac{\alpha(2-\beta)}{w_i(t)2\beta}}\right) \\ &= k_i(t)\left(1 - \frac{q_i(t)}{u_i(t)}\right) \end{aligned} \tag{2.17}$$

kde posledný tvar je všeobecným modelom Dynamiky, pre $k_i(t) = \frac{\alpha w_i(t)}{T_i(t)}$ a $u_i(t) = \frac{\alpha(2-\beta)}{w_i(t)2\beta}$.

2.4.6 *Vzťah Dynamiky a Reakčnej funkcie*

Veľa rozšírení pri popise funkčnosti spomína len Reakčnú funkciu, iné zasa len Dynamiku. Ako si ukážeme Reakčná funkcia a Dynamika sú vo vzájomnom vzťahu.

Reakčná funkcia nám určuje veľkosť okna zahltenia *cwnd* v závislosti od vynútenej stratovosti p v stabilnom stave. Do úvahy neberieme žiadny časový údaj, preto informáciu o veľkosti *cwnd* berieme ako statickú, vzťahujúcu sa na celý stabilný stav.

Dynamika ráta zmenu okna *cwnd*. Ako sme si ukázali v (2.15) a (2.17), je to rozdiel nárastu a poklesu okna *cwnd* za RTT. Túto informáciu už počítame v čase t. Od toho závisí stratovosť/miera zahltenia, veľkosť aktuálneho okna, či namerané RTT.

Pre porovnanie: Reakčná funkcia dáva statickú informáciu o stabilnom stave, Dynamika poskytuje dynamickú informáciu v závislosti od času. Uvažujme priemerné hodnoty namerané v jednom cykle v stabilnom stave nasledovne (berieme priemer hodnôt od začiatku cyklu až po jeho koniec). Nech *cwnd* je priemernou veľkosťou okna $w_i(t)$, nech q je priemerná miera zahltenia hodnôt $q_i(t)$ a nech T je priemerný RTT $T_i(t)$ nameraný počas cyklu. Keďže uvažujeme dobu celého cyklu v stabilnom stave, začíname na hodnote $(1-\beta)W$ a po dosiahnutí maximálneho okna W, znova padáme na hodnotu $(1-\beta)W$. Celkový priemerný nárast je preto nulový (nasledujúce je pre AIMD model - TCP Reno)

$$D_i(t) = 0 = \frac{\alpha cwnd}{T}\left(1 - \frac{q}{\frac{\alpha}{cwnd}\frac{(2-\beta)}{2\beta}}\right) \tag{2.18}$$

Po úprave (2.18) a vyjadrení cwnd dostávame rovnicu Reakčnej funkcie

$$\begin{aligned} 0 &= \frac{\alpha}{T} - \frac{cwnd}{T} q\beta \frac{2}{2-\beta} cwnd \\ cwnd^2 q\beta \frac{2}{2-\beta} &= \alpha \\ cwnd &= \sqrt{\frac{\alpha \frac{2-\beta}{2\beta}}{q}} \end{aligned}$$

pričom priemerná miera zahltenia q je rovná vynútenej stratovosti p v stabilnom stave.

Reakčná funkcia a Dynamika pojednávajú o tej istej informácii len v rozdielnych situáciách. Kým Reakčná funkcia je statickou informáciou veľkosti okna pre určitú vynútenú stratovosť v stabilnom stave, dynamika nám ponúka veľkosť zmeny okna v určitom čase *t*. Pokiaľ Reakčná funkcia poukazuje vzťah vynútenej stratovosti a veľkosti okna pre dané rozšírenie, Dynamika opisuje správanie sa Kontroly zahltenia (okrem Pomalého štartu) daného algoritmu.

V ďalších kapitolách predstavíme problémy štandardnej implementácie algoritmov Kontroly zahltenia a ich riešení. Zároveň budeme vychádzať z poznatkov popísaných v tejto kapitole na poukázanie problematických častí štandardnej implementácie v rôznych podmienkach a jej rozšírení.

Kapitola 3

Problémy štandardného TCP pre špecifické typy sietí

Štandardný protokol TCP je stavaný pre štandardné siete. Berie do úvahy ich vlastnosti a využíva ich vo svoj prospech. Na začiatku Internetu protokol TCP plne vyhovoval vtedajším požiadavkám. Postupom času sa zväčšovala priepustnosť liniek, ktorými sa dáta prenášali a vznikali stále väčšie požiadavky na prenos dát. Narastal počet užívateľov prihlásených do siete Internet. S týmto zväčšovaním prichádzali aj nové problémy. Pod pojmom štandardné TCP budeme rozumieť pôvodné špecifikácie TCP s implementovanou Kontrolou zahltenia podľa van Jacobsona. Typickým príkladom je TCP Reno.

Štandardné TCP bolo stavané pre drôtové siete. Tie sa vyznačovali malou pravdepodobnosťou strát paketov (stratovosťou) z dôvodu nespoľahlivosti liniek a zároveň mali dostatočné bufferovanie (dostatok miesta vo vyrovnávacích pamätiach) pre danú kapacitu siete. V poslednej dobe však Internet zaznamenal radu zmien. Kostra Internetu a mnohé ďalšie časti sú poprepájané vysokorýchlostnými a veľkokapacitnými linkami, ktoré zabezpečujú prenos veľkého množstva dát na obrovské vzdialenosti. Do Internetu sa užívatelia pripájajú cez bezdrôtové siete, alebo dokonca sa bezdrôtový prenos využíva na prepojenie drôtových sietí.

Každý takýto nový spôsob spojenia je špecifický vlastnosťami, ktoré nie sú v štandardnom TCP zohľadnené. Preto vzniká potreba rozšíriť protokol TCP tak, aby efektívne pracoval na novom spojení.

Otázkou je, či namiesto hľadania možných rozšírení už aj tak starého TCP a jeho algoritmov kontroly premávky, nie je jednoduchšie spraviť nový protokol, ktorý by tieto problémy naraz vyriešil. Môžeme spomenúť dva dôvody, prečo je výhodné rozširovať a meniť TCP: Prvým dôvodom je, že TCP dokázal, že je veľmi efektívny a silný v prenášaní dát sieťou a regulovaní premávky. Preto je dobré, aby veľa jeho vlastností zostalo zachovaných. Druhým dôvodom je predpoklad, že TCP zostane aj naďalej hlavným trans-

portným protokolom na rozličných sieťach. Pre každý nový protokol by sa musela brať do úvahy koexistencia a spätná kompatibilita s TCP, keďže celková zmena na úplne nový TCP nekompatibilný protokol by si vyžadovala zmenu u všetkých prvkov siete. A to je prakticky nereálne.

Definujme si *delay-bandwidth* produkt (resp. bandwidth-delay produkt) ako súčin prenosovej kapacity linky *bandwidth* a propagačného oneskorenia *delay* (oneskorenie v dôsledku šírenia signálu v linke). Delay-bandwidth produkt určuje kapacitu linky, t.j. koľko dát môže byť maximálne v danom časovom okamihu vo fáze prenosu.

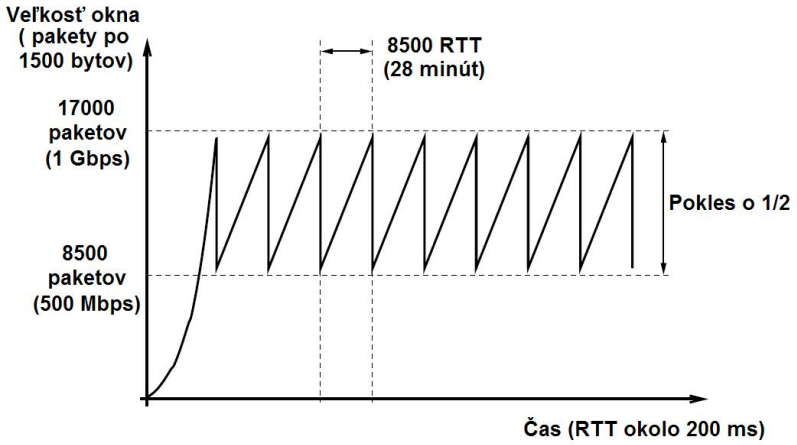
3.1 *Problémy TCP na vysokorýchlostných sieťach*

Vysokorýchlostné linky (high-speed) v súčasnosti tvoria chrbticu Internetu. Prepájajú vzdialené časti siete a zaručujú rýchly prenos veľkého množstva dát. Postupom času sa aj ostatné linky vymieňajú za výkonnejšie a je predpoklad, že kapacity liniek budú ďalej rásť.

Tento typ sietí sa vyznačuje veľkou kapacitou liniek a veľkým časom odozvy (RTT - Round Trip Time). Zároveň musí čeliť veľkému náporu dát a preto musí mať efektívne vyriešený systém bufferovania paketov. Vysokorýchlostné siete sa vyznačujú veľkým delay-bandwidth produktom a v mnohých literatúrach sa nazývajú aj Long, Fat Pipe LFN (vyslovuje sa elefant).

Ako príklad vysokorýchlostných sietí (LFN) môžeme uviesť veľkokapacitné satelitné spojenia, napríklad DS1-speed satellite channel má delay - bandwidth produkt väčší ako 10^6 bitov, čo je porovnateľné so 100 súčasne bežiacimi TCP paketmi o veľkosti 1200 bytov. Ďalším typom vysokorýchlostných sietí sú optické siete, kde sa napríklad pre zdržanie 30 ms na DS3 sieťach (s kapacitou 45Mbps) presahuje delay-bandwidth produkt 10^6 bitov, pre ktoré je stavaný štandardný TCP. Ten bol navrhovaný pre prostredia s propagačným oneskorením linky v rozmedzí 1 ms - 100 sekúnd a s prenosovou kapacitou od 100 bit/s - 10^7 bit/s.

Štandardný TCP s kontrolou toku je pre tento typ sietí pomalý, má pomalé dosiahnutie stabilného stavu. Týka sa to hlavne algoritmu predchádzania zahlteniu, ktoré uvažuje lineárne zväčšovanie posielacieho okna. Pre vysokorýchlostné siete je to však veľmi pomalé. Ďalším problémom je pokles posielacieho okna u TCP Rena pri strate o polovicu. Pri vysokorýchlostných sieťach s veľkou kapacitou je tento pokles privysoký a TCP potrebuje veľa času na dosiahnutie pôvodnej rýchlosti. Výsledkom štandardného predchádzania zahlteniu je nevyužitie plnej kapacity linky a výrazné poddimenzovanie možností prenosového pásma linky. Ako dôsledok, TCP zdroje slabo reagujú na zmeny v sieti. Zároveň dlhodobé nevyužívanie linky vedie aj k malej priepustnosti celej siete. Toto všetko je znakom zvolenia neprimeraného spôsobu práce s linkou a poddimenzovaným bufferovaním. No práve vhodné bufferovanie u vysokorýchlostných liniek je problém. Štandardný



Obrázok 3.1: Problém dlhého dosahovania maximálnej kapacity u vysokorýchlostných sietí.

TCP a jeho algoritmy kontroly zahltenia rátajú s tým, že na linke je dostatočne veľký buffer (vyrovnávacia pamäť) v porovnaní s delay-bandwidth produktom linky. Toto sa však z technických a ekonomických dôvodov nedá na takýchto sieťach zaručiť.

Pre veľké posielacie okná zároveň nastáva ďalší problém - viacnásobné straty paketov v jednom posielacom okne. Táto situácia je tým kritickejšia, čím väčšie je okno. V pôvodnom TCP, algoritmus Rýchleho zotavenia rieši problém straty jedného paketu za okno. Pre viacnásobné straty nám však tento algoritmus nepomôže, lebo väčšinou vyprší časovač skôr ako dôjdu tri *dupAck* a TCP je donútené prejsť do fázy Pomalého štartu. Riešením môže byť selektívne potvrdzovanie SACK (RCF 1072), ale to je len nepovinné a nie všetky rozšírenia ho podporujú.

Podľa [19] sú tri hlavné dôvody prečo siete s veľkou prenosovou kapacitou majú problémy so štandardným TCP. Prvým je limit veľkosti posielaného okna *wnd*, ktorá sa dohodne na začiatku spojenia a nemôže sa prekročiť. Je rozumné uvažovať o vhodnej veľkosti okna a aj o veľkosti vyrovnávacích pamätí u vysielateľa a prijímateľa. Druhým je mechanizmus merania RTT. V štandardom TCP sa meria RTT pre jeden paket v jednom okne ako vzorka. To však môže byť nedostatočné pre veľké okná, práve ktoré potrebujeme pre vysokorýchlostné rozšírenia. Preto sa vyvinuli a vyvíjajú mnohé mechanizmy na dodatočné výpočty RTT (napr. lit [6]). Tretím dôvodom je mechanizmus zotavenia z viacerých strát pre linky s dlhou odozvou. Tu po stratách musia spojenia čakať oveľa dlhší čas, ako je potrebný na riešenie zahltenia (čas potrebný na vyprázdnenie vyrovnávacích pamätí spojovacích uzlov). Taktiež ak niektoré spojenia čakajú, ďalšie môžu využiť situáciu a zmocniť sa linky, čím vzniká problém s udržaním férovosti.

Ako príklad neefektívneho hospodárenia so sieťovými prostriedkami mô-

Typ siete	Kapacita siete (Bandwidth)		Čas jedného kola rátania poradového čísla
ARPANET	56 kbps	7 KBps	~ 3,6 dňa
DS1	1,4 Mbps	190 KBps	~ 3 hod
Ethernet	10 Mbps	1,25 MBps	1700 s (~ 30 min)
DS3	45 Mbps	5,6 MBps	380 s
FDDI	100 Mbps	12,5 MBps	170 s
Gigabit	1 Gbps	125 MBps	17 s

Obrázok 3.2: Čas jedného kola rátania (wrapped cycle) pre niektoré typy sietí.

žeme uviesť (podľa [21]) spojenie s RTT 200 ms a veľkosťou paketu 1500 bytov na 1 Gbps linke (obrázok 3.1). Veľkosť okna, ktoré naplno využíva kapacitu tejto linky, musí mať veľkosť 17000 paketov. Hneď po zistení straty paketu sa okno zahltenia *cwnd* nastaví na 8500 paketov, čo zodpovedá klasickému algoritmu kontroly zahltenia. Toto zodpovedá rýchlosti 500 Mbps a na dosiahnutie plnej rýchlosti potrebuje TCP okolo 8500 RTT, čo znamená zhruba 28 minút. Tento postup je pomalý, neefektívne využíva prostriedky linky a umožňuje neférovosť medzi spojeniami s rozdielnou dĺžkou RTT.

Ďalším problémom protokolu TCP na vysokorýchlostných sieťach je konflikt poradových čísel (sequence number) poslaných paketov pre dané spojenie. Ako sme už spomínali v predchádzajúcej kapitole, posielané pakety sú usporiadané do postupnosti a je im priradené poradové číslo. Od tohto čísla ďalej závisí samotné potvrdzovanie paketov, preposielanie stratených a posielanie nových paketov. Pre poradové číslo je v hlavičke vyhranených 16 bitov. Po dosiahnutí maximálneho čísla preto musíme začať rátať od začiatku. Hovoríme o novom kole rátania (wrapped cycle).

Problém s nedostatočným množstvom poradových čísel nastáva práve pri sieťach s veľkým delay-bandwidth produktom (široké siete). Tento problém je podrobnejšie popísaný a ďalej riešený v RFC 1323 [33]. Pri vysokorýchlostných sieťach sa môže stať, že poradové čísla sa začnú opakovať za čas (teda nastane ďalšie kolo rátania), kedy je paket na ceste, resp. čaká vo vyrovnávacích pamätiach. Potom pri jeho strate sa paket, ktorý má rovnaké poradové číslo (pridelené v nasledujúcom kole), dostane do prijímateľa a je zle klasifikovaný ako stratený paket (predchádzajúceho kola). Tu vzniká problém zlého usporiadania dát a z toho vyplývajúci zlý výstup. Obrázok 3.2 ukazuje pre niektoré typy sietí čas potrebný na vypršanie jedného kola rátania poradového čísla.

Podobný problém môže nastať, ak sa ukončí spojenie a začne sa vzápätí ďalšie (s rovnakým párom soketov). Vtedy sa môžu pomiešať pakety pôvodného spojenia zdržané na ceste s paketami nového spojenia. To sa dá ale zamedziť použitím MSL - Maximum Segment Lifetime, čo je hranica život-

nosti paketu. TCP sa v tomto spolieha na protokoly nižšej vrstvy, napríklad IP, ktorý v sebe implementuje TTL (time to live) pole pre segmenty. Tým IP vie určiť, ktorý segment je dosť starý nato, aby ho zrušil.

3.2 *Problémy TCP na bezdrôtových sieťach*

V poslednej dobe sa začali dostávať do popredia bezdrôtové (wireless) siete. Ich rozšírenie bolo a je zapríčinené viacerými ukazovateľmi. Bezdrôtová sieť je lacná na vybudovanie, nie je závislá od fyzického média (linky) a teda sa dá zaviesť aj na miesta, kde je neekonomické alebo inak nemožné ťahať drôty.

Typov bezdrôtových sietí je niekoľko a každý má svoje klady a zápory. Preto je možné si zvoliť najvhodnejší typ pre dané nasadenie. Možnosti sú od satelitných spojení cez bezdrôtové LAN siete až po mobilné telefóny (GPRS).

Takéto siete majú vlastnosti, ktoré treba brať do úvahy pri prenose dát sieťou. Podľa typu bezdrôtovej siete a teda aj spôsobu prenášania paketov sieťou nás môže zaujímať kapacita prenosu, RTT, miesto využitia siete (napríklad v budove, na otvorenej ploche) a mnohé ďalšie vlastnosti ovplyvňujúce prenos. Všetky tieto fakty musíme zobrať do úvahy pri návrhu rozšírenia pre štandardný TCP nad danou sieťou.

Štandardný TCP, jeho kontrola a predchádzanie zahlteniu sú stavané na siete s vysokou spoľahlivosťou liniek. Značnú časť stratených paketov pripisujú zahlteniu linky a teda spúšťajú mechanizmy na jeho zabránenie a spamätanie sa z neho. Rátajú až s 99% spoľahlivosťou linky.

U bezdrôtových liniek je stratovosť dát z dôvodu nespoľahlivých liniek oveľa väčšia ako u bežných. Prenos môžu ovplyvniť mnohé faktory, napríklad šum z vonkajšieho prostredia, vzájomné sa rušenie uzlov v sieti (interferencia), reselekcia (handover) a mnohé ďalšie. TCP tieto straty interpretuje ako príznak zahltenia na linke, preto zníži vysielaciu rýchlosť. V prípade veľmi šumivého prostredia môže dôjsť k zníženiu vysielania na veľmi nízku hranicu. Pre rozšírenia TCP na takýchto typoch sietí by mala byť snaha odchytiť takéto straty a správať sa k nim inak ako k stratám, ktoré sú spôsobené zahltením.

Nasledujúce faktory priamo či nepriamo ovplyvňujú transport dát a mali by byť zohľadnené pri rozširovaní a implementácii daného rozšírenia TCP v praxi:

- Stratovosť paketov - hovorí o tom, ako veľmi je linka nespoľahlivá, teda aká je veľká pravdepodobnosť, že poslaný paket bude stratený pri prenose,
- Priepustnosť linky (prenosová kapacita) - bezdrôtové siete sa od seba líšia prostredím použitia, preto je aj ich priepustnosť (a požiadavky na

priepustnosť) rozdielna podľa toho, kto a ako bude služby danej siete využívať (GPRS siete majú oveľa menšiu priepustnosť ako bezdrôtové LAN),

- RTT (Round Trip Time) - štandardne ako u drôtových sietí aj tu potrebujeme zohľadniť vzdialenosť medzi jednotlivými uzlami.

Z dôvodu zjednodušenia popisu problémov týchto sietí si v tejto podkapitole rozdelíme bezdrôtové siete na LAN (statické - také čo ich komponenty nemenia polohu) a mobilné (ktoré musia riešiť dodatočnú réžiu s pohybom komponentov).

3.2.1 *Problémy TCP na bezdrôtových LAN sieťach*

Bezdrôtové LAN siete (WLAN) sú jedným z najrozšírenejších typov bezdrôtových sietí. Využívajú sa v nich mikrovlnné, rádiové, satelitné, infra, laserové alebo aj iné spojenia. Najrozšírenejšie sú však asi prvé dve.

Najväčším problémom je stratovosť liniek. Tá nastáva z dôvodu interferencie prostredia (zmena počasia, rušenie, steny, ostatné uzly siete). Prenosová kapacita môže byť variabilná, ale zväčša je postačujúca pre prenos. Ostatné problémy nezohrávajú v prenose až takú veľkú úlohu. Preto hlavným cieľom rozšírení nad týmito sieťami je rozlíšenie strát na straty spôsobené zahltením (kde sa bude rozšírenie správať ako štandardné TCP) a straty spôsobené nespoľahlivosťou linky (kedy sa musí nájsť spôsob, ako sa má rozšírenie správať v danej situácii).

Podľa [34] je u mnohých komerčných nasadení WLAN sietí (napr. 802.11) problém aj s férovosťou tokov. Mnohé WLAN siete majú nepomer vo veľkosti uplink (posielanie) a downlink (sťahovanie) - rozdelenie kapacity linky na posielanie a prijímanie paketov. Vyplýva to z predpokladu, že odosielateľ vysiela dátové pakety, ktoré sú pomerne veľké a prijíma potvrdenia *ACK*, ktoré sú naopak malé. Preto je rozumné prideliť väčšiu kapacitu na posielanie dát, ako na prijímanie *ACK*. To ale môže viesť k zahlteniu downlink-u, kde vo vyrovnávacích pamätiach čakajú na poslanie *ACK* pakety, čím trpia všetky toky prechádzajúce touto linkou. Zároveň s týmto vzniká problém férovosti nad takto zahlteným downlink-om. Nové a malé toky sú často vyhladované, lebo dostanú málo miesta vo vyrovnávacích pamätiach.

3.2.2 *Problémy TCP na mobilných (GPRS, EDGE, UMTS-3G) sieťach*

Hlavný problém mobilných sietí je riešiť stálosť pripojenia užívateľa aj pri pohybe. Ich štruktúra je v jednoduchosti popísateľná ako plocha zložená z malých buniek. Každá takáto bunka obsahuje vysielač/prijímač, s ktorým užívatelia v danej bunke komunikujú. Bunky sú charakteristické pásmom,

pod ktorým sa v danej bunke komunikuje. Toto pásmo musí byť rozdielne od pásma buniek susedov.

Sieť musí riešiť prechod užívateľa z jednej bunky do druhej bez straty spojenia. Na tento problém existuje niekoľko riešení. Dôležité je, aby užívateľ nepocítil prechod medzi bunkami resp. aby to pocítil čo najmenej. O toto všetko sa stará proces - reselekcia. Musí zabezpečiť komunikáciu užívateľa a obidvoch buniek, tú z ktorej užívateľ vychádza a tú, do ktorej vchádza. Ďalej musí dohliadať nad tým, aby sa vymenil dostatok informácií na presmerovanie komunikácie z jednej bunky do druhej. Tento proces si vyžaduje poslanie niekoľkých paketov a teda trvá určitý čas. Preto treba zaručiť čo najkratšie trvanie reselekcie, aby nedošlo k prerušeniu spojenia z dôvodu rýchleho prechodu medzi bunkami.

Pri mobilných sieťach musíme rátať so silnou interferenciou a šumom z prostredia. Tieto faktory sú dosť veľké vzhľadom na veľkosť využívania siete. Ich dôsledkom dochádza k náhodným výpadkom liniek a stratám paketov.

Iným faktorom je aj dlhotrvajúci výpadok. Dôvodom môžu byť náhla degradácia signálu, dlhotrvajúce zoslabenia signálu, reselekcia (hlavne rýchla alebo prebiehajúca v zlých podmienkach pre komunikáciu medzi bunkami napríklad slabý signál z jednej bunky). Môžu sa tu radiť aj úplné straty signálu, napr. prechod tunelom a pod.

Pri krátkych výpadkoch dochádza k pozdržaniu vysielania a po opätovnom nadviazaní spojenia, sa posielajú pakety, ktoré čakali na poslanie, ale boli pozastavené kvôli indikácii výpadku. Kritické sú dlhé výpadky, ktoré vedú k masovým stratám prenášaných dát až k úplnému zrušeniu komunikácie.

Ďalším problémom sú nízka a hlavne kolísavá prenosová kapacita liniek (*jitter*) zapríčinená použitím nepomerných šírok up-link (pásmo pre upload), down-link (pásmo pre download) a nedostatočným bufferovaním hlavne u mobilných telefónov a variabilná latencia - nastáva veľké zoskupovanie paketov (hlavne po stratách). Jitter resp. Packet Delay Variation je v RFC 3393 definované ako rozdiel v koncovom propagačnom oneskorení vybraných paketov toku (pričom ignorujeme stratené pakety).

Kvôli vyššie uvedeným vlastnostiam má (napríklad podľa [17, 16, 36, 37]) štandardný TCP problémy s efektívnym transferom dát nad takouto sieťou. Tieto problémy sa týkajú:

- Štartovacia fáza TCP - Ak je dobrý signál a malá stratovosť, Pomalý Štart (slow-start) zaručí efektívne využitie linky až za nejakú dobu od nadviazania spojenia (kým sa dosiahne veľkosť okna zahltenia (congestion window) *cwnd* danej linky). Pre mobilné siete (aspoň v súčasnosti) je charakteristický prenos malých objektov. Preto v značnej časti spojení dôjde ku koncu prenosu skôr, ako sa naplno využije kapacita linky pre daný prenos.

- Kompresia Potvrdzovacích správ (ACK compression) - odosielateľ po-

siela zhlukovo pakety ako odpoveď na viacero potvrdzovacích *ACK* (v implementáciách väčšinou štyri) správ prichádzajúcich rýchlo za sebou. Ak mobilné zariadenie má dáta aj na vysielanie aj na prijímanie, potom kvôli pozdržaniu poslania ACK, z dôvodu preplnenej výstupnej linky, môže dôjsť k neefektívnemu využitiu downlinku a možným vypršaním časovačov na strane posielateľa.

- Nadmerné bufferovanie (Excess queuing) - kvôli nižšej prenosovej kapacite mobilných sietí oproti klasickým sa hranica medzi drôtovou a mobilnou sieťou stáva kritickou. Dochádza k bufferovaniu paketov pred vstupom do mobilnej siete a je potrebné použiť rozšírené spôsoby bufferovania. Navyše ak nejaké spojenie s dlhou životnosťou príde na túto hranicu, je schopné obsadiť značnú časť kapacity vyrovnávacej pamäte, kým dôjde k indikácii zahltenia a následným opatreniam. Toto implikuje aj nasledujúce problémy.

- Hustenie RTT (RTT inflation) - väčšie zdržania pri bufferovaní vedú k degradácií správania sa TCP. V prípade veľkého množstva dát vo vyrovnávacej pamäti môže dôjsť až k vypršaniu časovačov na strane odosielateľa.

- Zväčšovanie hodnoty preposielacieho časovača (Retransmit Timer Value) - hustenie RTT spôsobuje zväčšovanie času v preposielacích časovačoch (vyplýva to z výpočtu veľkosti RTT pre preposlané pakety). Môže sa to odzrkadliť napríklad v neaktuálnosti preposlaných dát alebo viacnásobnej strate paketu.

- Staré dáta - čakanie paketu v rade (queue) vo vyrovnávacích pamätiach môže viesť k jeho neaktuálnosti, napríklad ak niekto ukončí prácu s danou aplikáciou alebo zruší sťahovanie. Posielanie takýchto paketov zabraňuje efektívnemu využitiu linky, zisťovanie a vyhadzovanie takýchto paketov stojí čas.

- TCP zotavenie zo strát (TCP loss recovery) - spamätanie sa zo straty dát u GPRS liniek trvá dlho. Je to spôsobené veľkým množstvom nevybavených dát z iných spojení, ktoré čakajú na poslanie a nízkou prenosovou kapacitou linky.

- Férovosť liniek - dlhé čakania v radoch vyrovnávacích pamätí umožňujú istý stupeň neférovosti. Ak nie je použité vhodné riešenie udržania férovosti nad vyrovnávacími pamäťami, môže dôjsť k vybavovaniu paketov jedného spojenia prednostne. Napríklad, ak sú v preplnenej vyrovnávacej pamäti dáta jedného spojenia a príde ďalšie spojenie, môže dôjsť k prednostnému vybavovaniu prvého spojenia, kým dôjde na rad druhé. Čiastočne sa to dá vyriešiť zahadzovaním paketov z radu, keď predpokladáme, že bude vyrovnávacia pamäť plná. Na to

sa používajú rôzne techniky (buffer management) ako Fair Queue FQ alebo iných. Vtedy treba zaručiť, aby bolo dosť miesta vo vyrovnávacej pamäti aj pre druhé spojenie, čo chvíľu potrvá. Techniky bufferovania a buffer management-u sme spomínali v predchádzajúcej kapitole pri popise protokolu TCP.

3.2.3 *Podpora nižších vrstiev*

V niektorých nasadeniach bezdrôtových sietí je spoľahlivosť prenosu dát priamo zaručená (resp. podporovaná) protokolmi pod TCP (fyzická, linková alebo sieťová vrstva). Ako príklad môžeme uviesť štandard IEEE 802.11 (bližšie popísaný v [35]), ktorý je v súčasnosti veľmi populárny, hlavne pre WLAN siete alebo mobilné ad-hoc siete. Preto si ho v skratke popíšeme a poukážeme na niektoré jeho problémy s koexistenciou s TCP.

Linková vrstva MAC pre štandard 802.11 obsahuje tri spôsoby komunikácie:

1. Distributed coordination function (DCF) - distribuovaný spôsob komunikácie. Vyznačuje sa použitím CSMA/CA (carrier sense multiple access protocol with collision avoidance) protokolom, ktorý sa stará o zdieľanie prenosového média a zaručuje riešenie kolíznych situácií.

2. rozšírenie DCF o RTS/CTS framy (správy Request-to-send a Clear-to-send), ktoré rozširujú funkcionalitu DCF o rezervovanie času na komunikáciu.

3. Point coordination function (PCF) - komunikácia sprostredkovaná a riadená jedným centrálnym uzlom Access Point (AP).

Pretože bezdrôtová sieť je vlastne zdieľané médium, tieto prístupy rôznymi spôsobmi riešia jednu hlavnú úlohu - zaručiť prideľovanie linky tokom a riešenie sporov (kedy sa viaceré uzly naraz rušia). Na zaručenie tejto úlohy používajú systém potvrdzovania správ, teda odosielateľ pošle správu a čaká na jej potvrdenie. To je posielané prijímateľom v prípade, že správa bola doručená v poriadku a bez chýb. Ak odosielateľ nedostane potvrdenie, po vypršaní časovača prepošle správu. To sa deje niekoľkokrát, pokým sa nedosiahne stanovený limit počtu preposielaní.

Ako si môžeme uvedomiť, týmto spôsobom sa zisťujú a prepošlú aj správy poškodené alebo stratené z dôvodu nespoľahlivosti linky. Teda MAC vrstva sa nám sama stará o zaručenie spoľahlivej komunikácie. Prečo potom uvažovať o ďalších rozšíreniach TCP nad zariadeniami využívajúcimi štandard 802.11? Na túto otázku existuje niekoľko odpovedí. V tejto časti sme vychádzali z [34, 35, 36, 37].

Tým, že MAC vrstva sa stará o preposielanie správ, zakrýva pred TCP straty paketov spôsobených nespoľahlivosťou linky. Teda TCP si vysvetlí

stratu ako signál zahltenia a zaháji algoritmy Predchádzania zahlteniu. Uvažujme situáciu, kedy MAC zistí stratu a preposiela správu. Toto vedie k zdržaniu paketu na ceste, ktoré si TCP nesprávne vysvetlí ako zahltenie a zníži veľkosť posielacieho okna. Loss based rozšírení sa to týka, len ak zdržanie paketu je dosť veľké, aby vypršal časovač. U delay based rozšírení však takéto zdržiavanie skresluje informáciu o stave linky a rozšírenie si z toho vyvodí, že linka je zahltená.

Podobne pri dlhotrvajúcich výpadkoch siete, kedy MAC nie je schopný preposielať správy na linke (a neoznámi to vyšším vrstvám), si TCP vysvetlí straty paketov ako signál zahltenia. Vtedy zníži okno až na minimum a začne fázu Pomalého štartu namiesto zachovania si veľkosti okna. Zároveň nastáva problém s plytvaním energie u mobilných zariadení, ktoré sa cez takéto výpadky snažia posielať pakety.

Ďalšími problémami 802.11 a jeho spoľahlivej komunikácie sú Problém skrytého terminálu (the Hidden terminal problem) a Problém exponovaného terminálu (the Exposed terminal problem). Prvý problém nastáva, keď tretí uzol, ktorý nie je v dosahu odosielateľa, ruší komunikáciu prichádzajúcu do prijímateľa. Vtedy nevie tretí uzol, že ruší, dochádza k opätovnému preposielaniu správ a následnému zdržiavaniu paketov na ceste (niekedy až straty). Druhý problém nastáva, keď nejaký tretí uzol v dosahu odosielateľa nie je v dosahu prijímateľa. Vtedy nemôže tretí uzol vysielať (lebo cíti sporné prostredie) aj keď vysiela prijímateľ, hoci nekoliduje s prijímateľom. Tu dochádza k neefektívnemu priestorovému využitiu siete, čím sa dáta pozdržiavajú na ceste. To môže spôsobiť vypršanie časovačov u TCP.

Samotné riešenie kolízií v MAC vrstve tiež môže viesť k pozdržiavaniu paketov na ceste. V prípade zistenia kolízie napr. u DCF sa uzly snažiace o posielanie na linke, musia umlčať o nejaký náhodný čas a potom znova skúsiť obsadiť linku. Medzi tým sa ale môže zmocniť linky iný uzol. To vedie k určitej miere neférovosti medzi tokmi.

Nasledujúce dva problémy 802.11 implementácií sa týkajú správania sa TCP na bezdrôtových sieťach len okrajovo a ich riešenie si vyžaduje zásah do 802.11 zariadení. Tu si ich pre úplnosť spomenieme. V 802.11 vyrovnávacia pamäť MAC vrstvy využíva Tail Drop systém zahadzovania paketov (tých, čo sú na konci). To zaručuje istý krok neférovosti. Existujú riešenia implementujúce férovejšie algoritmy správy vyrovnávacích pamätí.

Z dôvodu zaručenia spoľahlivého prenosu dát v MAC vrstve, trpia toky, ktoré si nevyžadujú spoľahlivý prenos ale požadujú rýchlosť. Takými sú toky (napr. UDP) prenášajúce video alebo audio záznam. Zároveň je problém aj s implementovaním kvalít služieb (Quality of Services). Aj tu je potrebná úprava MAC vrstvy pre riešenie tohto problému.

Kapitola 4

Rozšírenia TCP pre vysokorýchlostné siete

Vysokorýchlostné siete využívajúce TCP ako transportný protokol potrebujú kvôli už vyššie spomínaným problémom nejaké dodatočné riešenia alebo rozšírenia, aby plne využívali prenosovú kapacitu linky a pritom aj vhodne umožňovali kontrolu tokov (shaping).

Podľa [29] by mali rozšírenia mať nasledujúce vlastnosti:

- mali by sa správať ako klasické TCP, keď sú nasadené na pomalších sieťach a sieťach s malou vzdialenosťou,
- mali by byť TCP-priateľské, čo znamená, že by nemali vyhladovať ostatné súperiace toky na vysokorýchlostných linkách,
- mali by byť férové, teda mali by si zhruba rovnako prerozdeliť kapacitu linky,
- mali by byť zodpovedné, teda mali by rýchlo reagovať na zmeny vo voľnej kapacite linky (po štarte alebo po strate paketu),
- mali by zaručiť, že linky najviac vyťažené (tzv. úzke hrdlá - bottleneck) by mali byť stále efektívne využité.

4.1 *Klasifikácia rozšírení TCP pre vysokorýchlostné siete*

Rozšírenia nad vysokorýchlostnými sieťami môžeme klasifikovať do viacerých skupín. Nasledovnú klasifikáciu sme zvolili z pohľadu rozsahu zmien potrebných na implementáciu do štandardného TCP.

4.1.1 *Použitie TCP options*

Najskôr rozdelíme rozšírenia na dve skupiny, podľa toho, či využívajú špeciálne TCP *options* (možnosti) v hlavičke na riešenie problémov neefektívneho správania sa TCP na vysokorýchlostných sieťach alebo nie. Používanie špeciálnych TCP možností si vyžaduje ich implementáciu v oboch komunikujúcich stranách. Ak teda aspoň jedna strana nepozná alebo nesúhlasí s použitím danej TCP možnosti, komunikácia sa musí zaobísť bez nej. To znamená, že dané rozšírenie sa musí správať ďalej ako štandardné. TCP možnosti zároveň prinášajú ďalšie zväčšenie veľkosti paketu, preto treba zvážiť, nakoľko sa oplatí využiť danú TCP možnosť.

V RFC 1323 autori pomocou TCP možností riešia niektoré problémy so správaním sa TCP na vysokorýchlostných sieťach. Jednou TCP možnosťou, ktorá sa vymieňa pri vytváraní spojenia, je *Window scale option* (Škálovacia možnosť). Tá má za úlohu zväčšiť veľkosť maximálneho posielacieho okna (receive window). V štandardnom TCP si veľkosť maximálneho posielacieho okna komunikujúce strany dohodnú 16 bitovým poľom v TCP hlavičke. To nám dá maximálne okno $2^{16} = 65$ KB. Pre vysokorýchlostné siete je to však veľmi málo. Škálovacia možnosť nám umožňuje rozšírenie veľkosti maximálneho okna na 32 bitov. Toto zaručuje dostatočne veľké okná pre široké siete.

Škálovacia možnosť sa nezaoberá riešením problémov nad vysokorýchlostnou sieťou, ale poskytuje pre ňu možnosť vhodne si zvoliť maximálne množstvo dát, ktoré v jednom okne do nej pošleme. Táto TCP možnosť sa používa aj v ďalších rozšíreniach.

Ďalším vylepšením v RFC 1323 je meranie RTT. Autori navrhli Timestamp option (Možnosť časovej pečiatky, v ďalšom len timestamp), ktorá slúži na presnejšie meranie RTT. Najskôr odosielateľ pridá do paketu do timestamp možnosti aktuálny čas (údaj z jeho timestamp hodín). Po doručení paketu, prijímateľ prekopíruje timestamp možnosť do potvrdenia a pošle ho. Odosielateľ po obdržaní potvrdenia odráta timestamp od aktuálneho času timestamp hodín, a tak získa RTT meranie pre paket. Výhodou tohto prístupu je, že sa to dá urobiť aj pre viac paketov v okne. Potom namiesto jedného merania RTT ako u štandardného TCP, dostávame niekoľko meraní, ktoré nám dávajú lepšiu informáciu o stave v sieti. Z nich vieme vypočítať priemernú hodnotu RTT.

S použitím Timestamp možnosti, autori RFC 1323 riešia aj kolíziu poradových čísel (popísanej v predchádzajúcej kapitole). V pôvodnom TCP bol problém s malou hodnotou poľa pre poradové číslo v hlavičke paketu. To viedlo k skorému vyčerpaniu všetkých čísel v jednom kole rátania vo vysokorýchlostných sieťach a začatie druhého kola rátania poradového čísla. Problém nastal vtedy, keď pakety s rovnakým poradovým číslom boli prenášané tesne za sebou a došlo k preusporiadaniu alebo ku strate paketov. Autori RFC 1323 riešia tento problém rozšírením poradového čísla údajom o čase z timestamp možnosti. Ako príklad uvedieme nasledovnú situáciu.

Nech paket X s poradovým číslom x a timestamp-om tx sa stratí v sieti. Nech pred zistením straty paketu sa začne nové rátanie poradového čísla pre odosielané pakety (nové kolo rátania) a nech paket Y s poradovým číslom x a timestamp-om $ty > tx$ dorazí do prijímateľa. V štandardnom TCP by bol paket Y považovaný za X, ale s použitím timestamp možnosti porovnáme tx s ty a vidíme, že toto je paket ďalšieho kola. Preto nepošleme potvrdenie pre paket X, ale správne pre paket Y.

Musíme poznamenať, že timestamp možnosť je zakódovaná 32 bitovým číslom a timestamp hodiny musia byť vhodne zvolené tak, aby neboli príliš pomalé (inak by neriešili problém kolízie poradových čísel) a ani príliš rýchle (aby rýchlo nedošlo k narátaniu do posledného čísla). Autori navrhujú tik timestamp hodín medzi 1 ms a 1 sek. To nám umožní vypršanie timestamp hodín od 24,8 dní do 24800 dní, čo prevyšuje niekoľkonásobne životnosť veľkej väčšiny TCP spojení. V prípade spojení, ktoré sú živé dlhšie, pri vypršaní timestamp hodín a začatí rátania od začiatku vieme, že pakety s poradovým číslom blízkym nule poslané na začiatku kola sú už dávno staré (doručené alebo zahodené). Preto nové pakety s nízskym timestamp môžeme považovať za väčšie ako pakety s poradovým číslom blízkym k maximálnej hodnote timestamp hodín.

RFC 1072 definuje ďalšiu špeciálnu TCP možnosť - SACK (selective Ack). Táto TCP možnosť rieši problém viacnásobných strát paketov v rámci jedného okna. Tým sme schopný určiť, ktoré pakety prišli do prijímateľa mimo poradia a vieme ich vyhlásiť za potvrdené. V pôvodnom TCP s algoritmami Rýchle zotavenie a Rýchle preposlanie by sme museli čakať na kumulatívne potvrdenie paketov doručených mimo poradia. To však pre viacnásobné straty na vysokorýchlostných sieťach znamená vypršanie časovačov pre pakety doručené mimo poradia. Použitím SACK ale vieme, že tie pakety už boli doručené a potvrdíme ich. Tým zabránime klesnutiu posielacieho okna na 1 paket a začatie fázy Pomalého štartu.

Tieto TCP možnosti môžu byť zahrnuté aj v ďalších rozšíreniach, ak ich tie podporujú. Samotné TCP možnosti preto chápeme viac ako dodatočnú funkcionalitu než ako samotné rozšírenie. Zároveň si musíme uvedomiť, že pre používanie TCP možností musia najskôr súhlasiť obe komunikujúce strany a že TCP možnosti zaberajú určité miesto v pakete, čím ho zväčšujú.

4.1.2 *Explicitná vs. Implicitná informácia (feedback)*

Rozšírenia pre vysokorýchlostné siete si ďalej rozdelíme na dve skupiny, podľa toho, či potrebujú explicitnú (dodatočnú) potvrdzovaciu a informačnú podporu zo strany prvkov siete alebo nie. Takúto dodatočnú informáciu nazývame *feedback* (*spätná väzba*).

Ak si rozšírenie vyžaduje explicitný informačný (feedback) systém, môže dosiahnuť veľmi dobré výsledky vo využívaní prostriedkov siete, dosahovaní

stabilného a férového stavu. Informácie poskytované spojovacími uzlami dokážu podať presný pohľad na stav linky v danom okamihu a podľa nich už vieme prispôsobiť posielanie na strane odosielateľa. Problém je však pri nasadzovaní, lebo je potrebné zaručiť zmenu prvkov siete (väčšinou všetkých po ceste) zároveň s nasadením rozšírenia. Toto je dosť vážny problém pri už existujúcich sieťach. Zároveň použitím explicitnej informácie zo spojovacích uzlov sa nám ruší jedna zo základných vlastností TCP, to že je end-to-end (teda že sa zaoberá komunikáciou len medzi koncovými uzlami).

Na druhej strane implicitný systém, akým je napríklad vypršanie časovača alebo doručenie duplicitných potvrdzovacích správ *dupAck*, síce nevyžaduje žiadny zásah do prvkov siete okrem vysielajúcich a prípadne prijímajúcich, ale kontrola nie je tak efektívna.

Ďalšie rozdelenie rozšírení využívajúcich explicitné informácie už nie je potrebné. Vyplýva to z dôvodu postačujúcej informácie o stave siete. Ďalej sa touto skupinou rozšírení nebudeme zaoberať, lebo z pohľadu rozsahu zmien ide prakticky o vytvorenie nového protokolu (so zachovaním niektorých vlastností TCP) ako o rozšírenie TCP.

Tu aspoň spomenieme ECN (explicit Congestion Notification) [31] a XCP (eXplicit Control Protocol) [32]. ECN si vyžaduje potvrdzovanie pre každý paket v špeciálnych ENC routroch. Ak sa zistí zahltenie v sieti, ECN routre nastavia špeciálny bit v hlavičke potvrdenia a pošlú ho posielateľovi. Ten po obdržaní tejto informácie vie, že nastáva zahltenie a zníži posielaciu rýchlosť. Kým ECN využíva jednobitovú informáciu o stave zahltenia, XCP používa viacbitovú informáciu. Nekóduje mieru zahltenia ako 0 a 1, ale posiela veľkosť zahltenia v danom XCP routery (teda informáciu, na koľko je vyrovnávacia pamäť ešte voľná resp. predpoklad o ďalšom priebehu).

4.1.3 *Rozdelenie podľa spôsobu zisťovania zahltenia*

Toto rozdelenie sa týka rozšírení s implicitným spôsobom zisťovania zahltenia. Tu spomíname dva spôsoby, založené na stratách (loss based) alebo založené na zdržaní v zásobníkoch/vyrovnávacích pamätiach (queueing delay based). Tie sú bližšie popísané v kapitole **2.4.1 Spôsoby zistovania zahltenia**.

Loss based systém, využívajúci stratu ako signál zahltenia, je zahrnutý napríklad v implementácii TCP Reno [28], HighSpeed TCP [24], Scalable TCP [21], H-TCP [29], BIC TCP [26] alebo CUBIC TCP [27]... Delay based systém je použitý v implementácii TCP Vegas [22] a Fast TCP [23]. Tu sa uvažuje rozšírenie signálu straty o meranie zdržaní paketov. Tak dostávame presnejšiu informáciu o stave zahltenia v sieti.

Výhody Delay based princípu oproti Loss based zisťovaniu strát sú pre malé rýchlosti takmer zanedbateľné, no pri vysokorýchlostných sieťach sú značné. Závisí to hlavne od faktu, že pre veľkokapacitné siete sú straty menej pravdepodobné (k množstvu dát). Vyplýva to z Reakčnej funkcie

TCP. Ona znázorňuje závislosť okna *cwnd* od stratovosti *q*. Podľa RCF 1323, štandardné TCP bolo navrhnuté pre siete s prenosovou kapacitou od 100 bit/s do 10^7 bit/s pre propagačné oneskorenia v rozmedzí 1 ms - 100 s. To ale nezodpovedá vysokorýchlostným sieťam. Ako je ukázané na obrázku 2.7, pre veľké *cwnd* zodpovedá malá stratovosť. Preto je výhodnejšie, ako umelo vyvolávať straty v Loss based systémoch a tým zapríčiniť značný pokles (o násobok veľkého *cwnd*), zabrániť možným stratám u Delay based systémov.

4.1.4 *Dynamika a Reakčná funkcia*

Dynamiku a Reakčnú funkciu sme si už definovali pre štandardné TCP v kapitole 2, v časti o Modeli TCP. Ako sme si popísali, Dynamika vyjadruje veľkosť zmeny *cwnd* v čase t a Reakčná funkcia vyjadruje veľkosť priemerného okna *cwnd* v stabilnom stave.

Pomocou Dynamiky a Reakčnej funkcie môžeme opísať model algoritmu zahltenia u protokolov. Dokonca u vysokorýchlostných sietí si rozšírenia berú Dynamiku ako predmet skúmania a tú potom menia, alebo presnejšie povedané menia funkcie k_i a u_i. Všeobecný tvar je preto rovnaký, zdedený z TCP Rena a využívajú ho ostatné rozšírenia vhodne upravený.

Reakčná funkcia zase pomáha autorom pri návrhu rozšírenia stanoviť požadovanú závislosť *cwnd* od stratovosti *q*. Tento vzťah potom slúži na výpočet parametrov prenosu α a β pre stanovenú závislosť.

Obrázok 2.7 zobrazuje závislosť veľkosti okna a stratovosti v stabilnom stave pre AIMD a MIMD rozšírenia. Ako sme už spomínali, pre veľké okná majú MIMD výhodu, lepšie sa adaptujú na linky s väčšou kapacitou a sú výrazne efektívnejšie ako AIMD. Naopak AIMD sú vhodnejšie a efektívnejšie pre malé okná.

Nasledovné riešenia si ďalej rozdelíme najskôr podľa loss-based a delay-based princípov a potom podľa Dynamiky.

4.1.5 *Loss-Based*

Pozrime sa na skupinu rozšírení, ktoré pracujú na princípe Loss-Based. Teda strata je braná ako signál zahltenia, po ktorom sa znižuje prenosová rýchlosť. Väčšina protokolov používa algoritmy AIMD (aditional increase multiplicative decrease - aditívny nárast multiplikatívny pokles), MIMD (multiplicative increase multiplicative decrease - multiplikatívny nárast multiplikatívny pokles), ale existujú aj iné prístupy. Rozšírenia si preto rozdelíme podľa nich.

AIMD

K tejto skupine rozšírení patrí TCP Reno [28], HighSpeed TCP [24]. Tu sa po zistení zahltenia - strata paketu - zníži vysielacia rýchlosť u vysielača

o nejaký násobok a potom sa postupne lineárne zväčšuje, kým nedôjde k ďalšej strate.

Ako sme už ukázali v časti **2.4.5 Dynamika**, model všeobecného algoritmu AIMD (obrázok 2.5) sa dá stanoviť z nárastu okna *cwnd* pri úspešnom doručení $\frac{\alpha}{T_i(t)}$ a poklesu veľkosti okna *cwnd* pri zistenej strate a z toho model Dynamiky pre AIMD

$$D_i(t) = \frac{\alpha}{T_i(t)} - x_i(t)q_i(t)\beta\frac{2}{2-\beta}w_i(t) \tag{4.1}$$

TCP Reno

Tu si opíšeme štandardný TCP tak, ako je implementovaný v TCP Reno [28].

Správanie sa protokolu:

i) keď nastane strata paketu okno zahltenia *cwnd* sa nastaví na
$cwnd \longleftarrow \frac{cwnd}{2}$
teda je to pokles o
$x_i(t)q_i(t)\frac{2}{3}w_i(t)$ paketov v čase t
keď $\beta = 1/2$

ii) pri pozitívnom potvrdení paketu sa *cwnd* zväčšuje nasledovne
$cwnd \longleftarrow cwnd + \frac{1}{cwnd}$
teda nárast je o $\frac{1}{T_i(t)}$, keď $\alpha = 1$

Z tohto algoritmu dostávame Dynamiku

$$D_i(t) = \frac{1}{T_i(t)} - x_i(t)q_i(t)\frac{2}{3}w_i(t) = \frac{1}{T_i(t)}\left(1 - \frac{q_i(t)w_i(t)^2}{1,5}\right) \tag{4.2}$$

Teda $k_i(t) = \frac{1}{T_i(t)}$ a $u_i(t) = \frac{1,5}{w_i(t)^2}$.

Reakčná funkcia v stabilnom stave u TCP Rena vyzerá nasledovne $cwnd = \frac{1,2}{\sqrt{p}}$.

Toto je klasický algoritmus zahltenia, ktorý sa budeme snažiť ďalej nejakým spôsobom upraviť, aby sme dosiahli dostatočné využitie linky na vysokorýchlostných sieťach.

HighSpeed TCP

S riešením problémov štandardného TCP na vysokorýchlostných sieťach prišiel Sally Floyd v RFC 3649 [24]. Tu si popíšeme ideu rozšírenia.

Dynamika a Reakčná funkcia u HighSpeed TCP závisí od prostredia. Keď je na sieti veľká premávka a tým pádom aj veľká stratovosť paketov, alebo ak je rozšírenie nasadené na pomalších linkách, používa algoritmus

na predchádzanie zahlteniu a Dynamiku ako TCP Reno. Tým zabraňuje zahlteniu liniek zo svojej strany. Keď naopak sieť je vysokorýchlostná, stratovosť je malá a linky na ceste nevykazujú veľké zahltenie, rozšírenie použije upravenú Dynamiku.

Ako prvý krok k vytvoreniu rozšírenia, autor zvolil vzťah medzi vynútenou stratovosťou a veľkosťou okna (Reakčná funkcia) v stabilnom stave nasledovne

$$cwnd = \frac{0,12}{p^{0,835}} \tag{4.3}$$

Tým chcel zaručiť lepšie využitie okna pri rovnakej stratovosti ako je u TCP Reno.

Druhým krokom bolo stanovenie algoritmu:

i) pri potvrdení paketu sa rozšírenie správa nasledovne

$cwnd \longleftarrow cwnd + \frac{\alpha(cwnd)}{cwnd}$

čo predstavuje nárast o $\frac{\alpha(w_i(t))}{T_i(t)}$

ii) ako odpoveď na stratu je algoritmus nasledovný

$cwnd \longleftarrow (1 - \beta(cwnd))cwnd$

čo predstavuje pokles o $x_i(t)q_i(t)w_i(t)\frac{2\beta(w_i(t))}{(2-\beta(w_i(t)))}$ paketov v čase t

Rozšírenie má na rozhodovanie, ktorý algoritmus použiť, parameter Low_Window (okolo 38 MSS, čo zodpovedá možnej vynútenej stratovosti $Low\_P \approx 10^{-3}$). Ak momentálna veľkosť $cwnd$ je menšia ako Low_Window, použijeme štandardný TCP algoritmus, v opačnom prípade náš upravený. Teda pre $cwnd \leq Low\_Window$ máme $\alpha(cwnd) = 1$ a $\beta(cwnd) = 0,5$.

Parametre pre $cwnd$ väčšie ako Low_Window závisia od veľkosti $cwnd$ (teda sú funkciou $cwnd$) a musia sa raz za čas prepočítať. Tu sa uvažuje hodnota $High\_Window$ (83000 segmentov), čo je maximálne dosiahnuteľná veľkosť $cwnd$ a ku nej prislúchajúca stratovosť $High\_P$ (10^{-7}) vypočítaná zo vzťahu (4.3).

Závislosť medzi parametrami $\alpha(cwnd)$ a $\beta(cwnd)$ je nasledovná - zo všeobecnej rovnice Dynamiky (4.1) si vyjadríme parameter $\alpha(cwnd)$, čím jeho výpočet bude závislí od $\beta(cwnd)$, $q(cwnd)$ a $cwnd$ a to nasledovne

$$\alpha(cwnd) = cwnd^2 q(cwnd)\frac{2\beta(cwnd)}{(2 - \beta(cwnd))} \tag{4.4}$$

kde $q(cwnd)$ je stratovosť pri nejakej veľkosti okna a vyjadríme ju z očakávanej rovnice Reakčnej funkcie (4.3) ako $q(cwnd) \approx \frac{0,0789}{cwnd^{1,1976}}$ (kde stratovosť $q(cwnd)$ aproximujeme vynútenou stratovosťou v stabilnom stave p). Teda dostávame

$$\alpha(cwnd) \quad = \quad cwnd^2 \frac{0,0789}{cwnd^{1,1976}} \frac{2\beta(cwnd)}{(2 - \beta(cwnd))}$$

$$= \quad 0,0789 cwnd^{0,8024} \frac{2\beta(cwnd)}{(2-\beta(cwnd))} \tag{4.5}$$

Z (4.5) si vyjadríme závislosť okna $cwnd$ od parametrov $\alpha(cwnd)$ a $\beta(cwnd)$ nasledovne

$$\alpha \frac{2-\beta(cwnd)}{2\beta(cwnd)} = 0,0789 cwnd^{0,8024} \tag{4.6}$$

Nakoniec ešte potrebujeme vypočítať $\beta(cwnd)$. Autor zvolil nasledovný vzťah

$$\beta(cwnd) = \frac{(High\_Decrease - 0{,}5)(\log(cwnd) - \log(Low\_Window))}{(\log(High\_Window) - \log(Low\_Window))} + \\ +0,5 \tag{4.7}$$

kde $High\_Decrease$ určuje rýchlosť zmenšovania okna, napr. u TCP Reno je to 0,5.

Zo všeobecného modelu Dynamiky (4.1) a vzťahu medzi parametrami $\alpha(cwnd)$, $\beta(cwnd)$ a oknom $cwnd$ (4.6) dostávame Dynamiku pre HighSpeed TCP ([24])

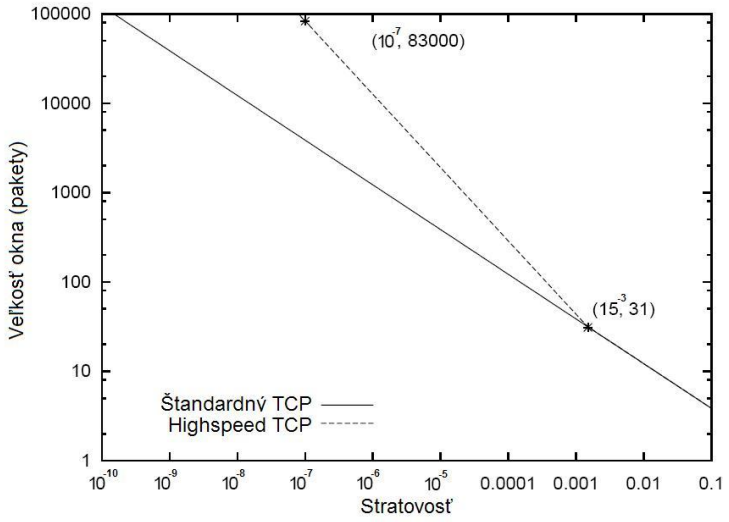
$$\begin{aligned} D_i(t) &= \frac{\alpha(w_i(t))}{T_i(t)} - x_i(t) q_i(t) w_i(t) \frac{2\beta(w_i(t))}{(2-\beta(w_i(t)))} \\ &= \frac{2\beta(w_i(t))}{T_i(t)(2-\beta(w_i(t)))} \left(\alpha \frac{2-\beta(w_i(t))}{2\beta(w_i(t))} - q_i(t) w_i^2(t) \right) \\ &= \frac{2\beta(w_i(t))}{T_i(t)(2-\beta(w_i(t)))} \left(0,0789 w_i^{0,8024}(t) - q_i(t) w_i^2(t) \right) \\ &= \frac{0,1578 \beta(w_i(t)) w_i^{0,8024}}{T_i(t)(2-\beta(w_i(t)))} \left(1 - \frac{q_i(t)}{\frac{0,0789}{w_i^{1,1976}(t)}} \right) \end{aligned} \tag{4.8}$$

alebo inak

$$k_i(w_i, T_i) = 0,1578 \beta(w_i) w_i^{0,8024} \frac{1}{T_i(2-\beta(w_i))} \tag{4.9}$$

$$u_i(w_i, T_i) = \frac{0,0789}{w_i^{1,1976}} \tag{4.10}$$

Porovnanie Reakčných funkcií štandardného TCP a Highspeed TCP je zobrazené na obrázku 4.1. Tu si môžeme všimnúť fakt, že Highspeed TCP pri malých oknách (približne 31 paketov v okne) sa správa ako štandardné TCP. Nad touto hranicou sa používa pozmenená Reakčná funkcia znázornená prerušovanou čiarou, ktorá zaručuje využitie väčšieho okna pri väčšej stratovosti ako štandardné TCP. Inak povedané pri rovnakej stratovosti Highspeed TCP



Obrázok 4.1: Priebeh AIMD modelu a jeho správanie sa v stabilnom stave.

má väčšiu priepustnosť ako štandardné TCP, čo znamená lepšie využitie kapacity vysokorýchlostnej linky.

HighSpeed TCP svojou Reakčnou funkciou simuluje emuláciu N paralelných tokov TCP spojení, kde N je na začiatku 1 a postupne stúpa ako funkcia veľkosti *cwnd*.

Pozmenenou Dynamickou funkciou a algoritmom zahltenia chce rozšírenie dosiahnuť nasledovné ciele:

- dosiahnutie veľkej priepustnosti na spojeniach (per-connection) bez požadovania nerealisticky nízkej stratovosti,
- dosiahnutie vysokej priepustnosti bez prílišných zdržaní pri zotavovaní z viacnásobných preposielacích timeoutov alebo keď zrýchľujeme z malého *cwnd*,
- žiadna dodatočná informácia alebo iná pomoc od spojovacích uzlov,
- v prostredí so stredným a veľkým zahltením sa správať aspoň ako TCP,
- férovosť medzi tokmi.

MIMD

Do tejto skupiny patria rozšírenia, ktoré implementujú algoritmy s multiplikatívnym nárastom a poklesom. Tu spomenieme Scalable TCP [21].

Pri MIMD rozšírení budeme vychádzať zo všeobecného modelu Dynamiky pre MIMD (2.17)

$$D_i(t) = \frac{\alpha w_i(t)}{T_i(t)} - x_i(t)q_i(t)\beta\frac{2}{2-\beta}w_i(t) \tag{4.11}$$

Pri MIMD je samotný nárasť okna závislý od jeho predchádzajúcej veľkosti. Teda MIMD má lepšie správanie sa na vysokorýchlostných linkách, lebo nárast je dynamický a pre siete s väčšou kapacitou vykazuje skoršie dosiahnutie stabilného stavu (čím je okno väčšie, tým predpokladáme väčšiu kapacitu linky a teda zvýšime viac okno).

Scalable TCP [21]

Do tejto skupiny patrí Scalable TCP. Ide o zjednodušenie Highspeed TCP zmenou AIMD modelu na MIMD.

Algoritmus pracuje nasledovne:

i) keď nie je zistená žiadna strata

$cwnd \longleftarrow cwnd + \alpha$

kedy je nárast o $\alpha\frac{w_i(t)}{T_i}$

ii) pri zistení zahltenia signalizovaného stratou paketu

$cwnd \longleftarrow cwnd - \lceil\beta * cwnd\rceil$

kde pokles je o $\frac{2\beta}{2-\beta}x_i(t)q_i(t)w_i(t)$ paketov v čase t

Z toho dostávame Dynamiku charakterizujúcu rozšírenie

$$\begin{aligned} D_i(t) &= \alpha\frac{w_i(t)}{T_i(t)} - \frac{2\beta}{2-\beta}x_i(t)q_i(t)w_i(t) \\ &= \alpha\frac{w_i(t)}{T_i(t)}\left(1 - \frac{q_i(t)}{\frac{\alpha(2-\beta)}{2\beta w_i(t)}}\right) \end{aligned} \tag{4.12}$$

alebo inak

$$k_i(w_i, T_i) = \alpha\frac{w_i(t)}{T_i(t)} \tag{4.13}$$

$$u_i(w_i, T_i) = \frac{\alpha(2-\beta)}{2\beta w_i(t)} \tag{4.14}$$

Podobne ako u HighSpeed TCP aj tu existuje hranica, až od ktorej sa rozšírenie správa lepšie ako štandardné, preto je vhodné prepínať medzi algoritmami, keď okno prekročí túto hranicu. Vo svojej podstate je Scalable TCP veľmi podobné HighSpeed TCP, líši sa len vybratím inej reakčnej funkcie. Tvorcovia ho navrhli tak, že sa používajú konštanty pre zväčšovanie a zmenšovanie *cwnd* namiesto parametrizácie ako je to u HighSpeed. Tým

chceli dosiahnuť podobný výsledok vo využívaní siete a zároveň umožniť väčšiu jednoduchosť protokolu a viac zrozumiteľný model.

Reakčná funkcia Scalable TCP má tvar $cwnd = \frac{\alpha'}{\beta' p}$ a je znázornená na obrázku 2.7. Môžeme si všimnúť, podobne ako u Highspeed TCP, potrebu využitia štandardného TCP pri malých oknách (približne 16 paketov a stratovosť $5,86 \times 10^{-3}$). Pri menšej stratovosti Scalable TCP dovoľuje dosiahnuť väčšie okná ako štandardné TCP.

Nastavenie parametrov α a β je vec rozhodnutia autorov rozšírenia. V [21] je β stanovené na $\frac{1}{8}$, čo ponúka kompromis medzi výkyvmi rýchlosti a časom, potrebným na dosiahnutie stabilného stavu. α je potom určená ako $\frac{1}{100}$ (z Reakčnej funkcie pre rozšírenie).

Iné tvary Reakčnej funkcie

Tu patria viaceré rozšírenia, ktoré berú do úvahy iné funkcie na kontrolu premávky ako sú AIMD a MIMD.

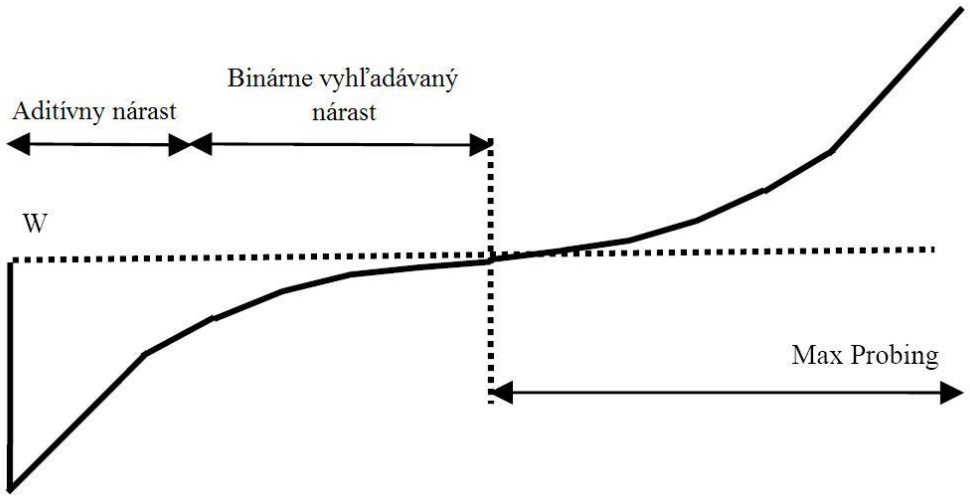
Binary Increase TCP

Binary Increase TCP (BIC) [26] sa pozerá na problém zahltenia siete ako na binárny vyhľadávací problém - áno/nie na otázku, či posielacia rýchlosť je väčšia ako kapacita linky. Algoritmus sa skladá z dvoch častí, *Nárast pomocou binárneho vyhľadávania* a *Aditívny nárast*.

Na začiatku ide klasický algoritmus pomalého štartu, na konci ktorého poznáme hodnotu maximálneho okna. V zotavení zo straty sa potom používajú už spomínané dva algoritmy. Minimálne okno je počítané ako veľkosť okna, pri ktorom tok nemá žiadne straty. Ak je známe maximálne okno (z algoritmu pomalého štartu - slow start), technikou binárneho vyhľadávania vypočítame cieľové okno, čo je stred medzi minimálnym a maximálnym oknom. Ak rastieme k cieľovému oknu a zistíme stratu, maximálnemu oknu nastavíme hodnotu súčasného okna a redukované okno (upravené algoritmom predchádzanie zahlteniu) sa stane naším novým minimálnym oknom. Ich stred bude naším novým cieľovým oknom. No ak dosiahneme cieľové okno bez straty, súčasné okno sa stane minimálnym oknom a znovu prepočítame cieľové okno. Toto je počítané, pokiaľ nie je rozdiel medzi minimálnym a maximálnym oknom menší ako nejaká konštanta (minimálny inkrement).

No ak je vzdialenosť medzi súčasným oknom a cieľovým veľká (prekračuje nejaký konštantný maximálny inkrement *Smax*), tak použijeme namiesto predchádzajúcej techniky aditívny nárast o *Smax*. Tým zaručíme, že nebudeme zväčšovať o veľké skoky.

Táto technika sa na začiatku správa agresívnejšie, keď je väčšia vzdialenosť medzi súčasným a cieľovým oknom, no čím sa viac približujeme, posielanie sa spomaľuje. Zmenšovanie okna sa deje o multiplikatívny faktor β.



Obrázok 4.2: Priebeh nárastu okna u BIC.

Hodnota maximálneho okna sa hľadá na začiatku v pomalom štarte a po každom prekročení doteraz zistenej hodnoty maximálneho okna bez zistenia straty paketu. Technika sa volá Max-probing a je to vlastne otočenie binárneho vyhľadávacieho algoritmu. Najskôr sa pomaly stúpa a postupne sa rýchlosť zväčšuje. Stúpanie je exponenciálne, preto ho treba od nejakého bodu spomaliť. To znamená, že ak prekročíme nejakú konštantnú hranicu, začneme stúpať lineárne, aby sme veľkými oknami nezahltili sieť.

Priebeh nárastu okna u BIC je znázornený na obrázku 4.2. Nárast sa skladá z dvoch častí, aditívneho nárastu a nárastu pomocou binárneho vyhľadávania. Po prekročení doteraz zistenej maximálnej veľkosti okna vbiehame do Max-Probing cyklu, ktorý je inverzným k doterajšiemu nárastu.

Reakčná funkcia je zložením logaritmickej a lineárnej funkcie. Predpokladajme stratu každého q-teho paketu (stratovosť je $\frac{1}{q}$), W maximálne dosiahnuté okno a pokles okna na hodnotu $(1-\beta)W$ po strate. BIC prepína z aditívneho nárastu na binárne vyhľadávaný nárast, keď vzdialenosť medzi aktuálnym oknom a cieľovým oknom je menšia ako *Smax*. Pre počet RTT kôl inkrementálneho nárastu podľa [26] platí

$$N_1 = \max\left(\left\lceil \frac{W\beta}{Smax} \right\rceil - 2, 0\right)$$

Z toho počet kôl binárneho vyhľadávacieho nárastu môžeme vyjadriť ako $W\beta - N_1 Smax$ alebo

$$N_2 = \log_2\left(\frac{W\beta - N_1 Smax}{Smin}\right) + 2$$

kde $Smin$ je veľkosť okna po strate.

Počas aditívneho nárastu sa okno zväčšuje o $\frac{1}{Smax}$ za RTT. To dáva celkový počet paketov

$$Y_1 = \frac{1}{2}(W(1-\beta) + W(1-\beta) + (N_1 - 1)Smax)N_1 \tag{4.15}$$

Počas binárne vyhľadávaného nárastu je nárast logaritmický a dostávame celkový počet paketov

$$Y_2 = WN_2 - 2(W\beta - N_1 Smax) + Smin \tag{4.16}$$

Celkový počet RTT v cykle je potom $N = N_1 + N_2$ a celkový počet paketov v cykle je $Y = Y_1 + Y_2$. Zároveň z toho, že strata je každých p paketov, Y môžeme vyjadriť ako $Y = \frac{1}{p}$ a použitím rovníc (4.15) a (4.16) vyjadriť W. Tu uvažujeme 2 prípady.

Najskôr uvažujme $W\beta > 2Smax$ a nech je $W\beta$ deliteľné $Smax$. Potom dostávame

$$W = \frac{-b + \sqrt{b^2 + 4a\left(c + \frac{1}{p}\right)}}{2a}$$

kde $a = \beta(2-\beta)/(2Smax)$, $b = \log_2(Smax/Smin) + (2-\beta)/2$ a $c = Smax - Smin$. Reakčná funkcia stabilného stavu je potom

$$cwnd = \frac{Y}{N} = \frac{(2-\beta)\frac{1}{p}}{\sqrt{b^2 + 4a(c + \frac{1}{p})} + (1-\beta)b + \frac{\beta(2-\beta)}{2}} \tag{4.17}$$

V prípade $W\beta >> 2Smax$ a pre fixné $Smin$, platí $N_1 >> N_2$. Preto posielacia rýchlosť závisí hlavne od aditívneho nárastu a stratovosti. V tomto prípade teda môžeme Reakčnú funkciu aproximovať

$$cwnd \approx \sqrt{\frac{Smax(2-\beta)}{2\beta p}} \tag{4.18}$$

(4.18) je veľmi podobná (2.1) s $\alpha = Smax$. Preto pri veľkých oknách má BIC podobné správanie ako AIMD model.

Teraz uvažujme prípad $W\beta \leq 2Smax$, teda $N_1 = 0$. Predpokladajme $1/p >> Smin$, potom

$$W \approx \frac{1}{\left(\log_2\left(\frac{W\beta}{Smin}\right) + 2(1-\beta)\right)p}$$

z toho dostávame Reakčnú funkciu

$$cwnd = \frac{Y}{N} = \frac{\frac{1}{p}}{\log_2\left(\frac{W\beta}{Smin}\right) + 2} \approx W\left(1 - \frac{2\beta}{\log_2\left(\frac{W\beta}{Smin}\right) + 2}\right)$$

Ak $2\beta << \log_2(W\beta/Smin) + 2$, potom

$$T \approx W \tag{4.19}$$

Z (4.19) vyplýva, že posielanie je nezávislé od *Smax*.

Ako zhrnutie môžeme poukázať na fakt, že pre veľké okná je rýchlosť posielania funkciou *Smax* a pre malé okná je funkciou *Smin* pri fixnom β.

CUBIC TCP

CUBIC TCP [27] je vylepšením BIC z pohľadu férovosti, TCP priateľskosti a zjednoduhšenia modelu protokolu. Ide o zmenu reakčnej funkcie na real-timeovú, čím sa nárast stane nezávislým od merania RTT. Kým BIC bol agresívny pre toky s malou hodnotou RTT alebo na pomalej linke, CUBIC sa viac adaptuje na podmienky siete.

CUBIC má v sebe kubickú Reakčnú funkciu

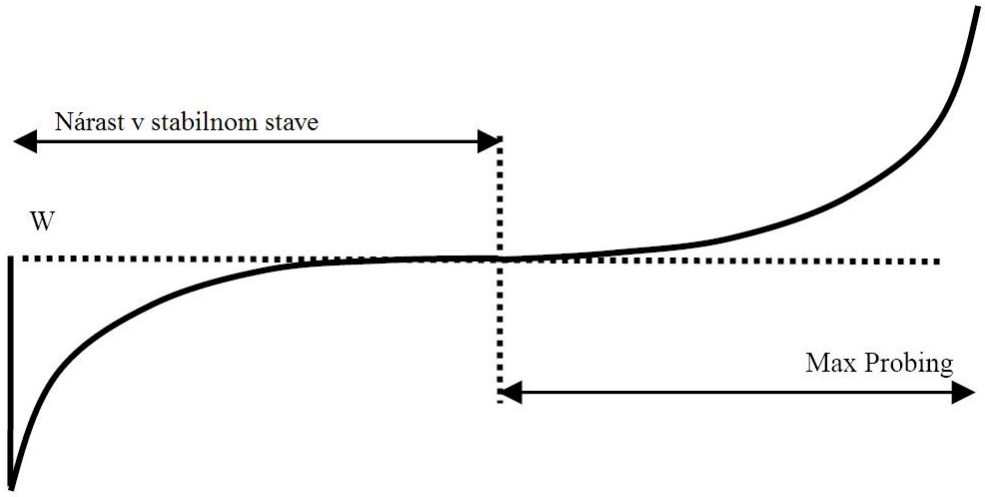
$$cwnd = C(t-K)^3 + W$$

kde C je zväčšujúci faktor, t je vypršaný čas od posledného zmenšovania, W je veľkosť okna pred redukciou, K je $\sqrt[3]{\frac{W\beta}{C}}$, β je konštantný zmenšovací faktor.

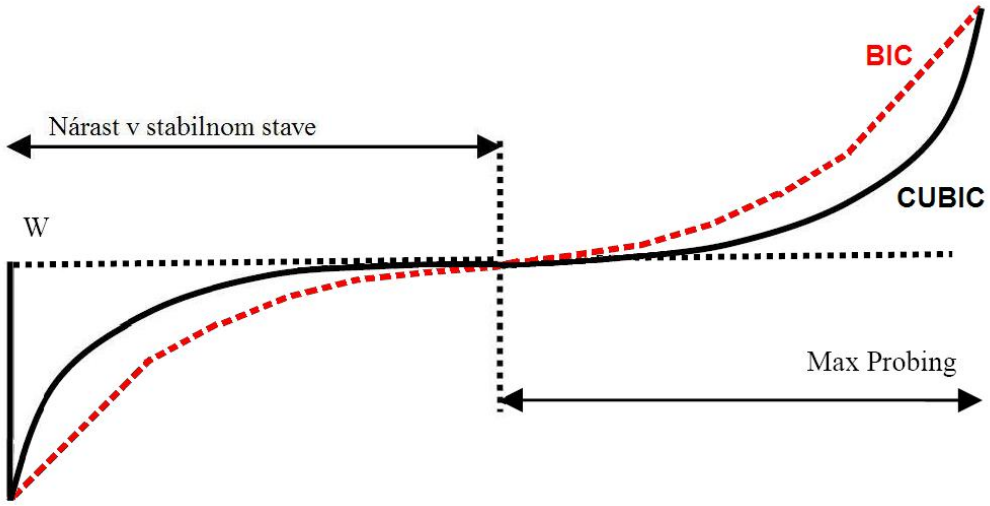
CUBIC sa snaží o aproximáciu zložitej Reakčnej funkcie BIC TCP zvolením jednoduhšej kubickej funkcie. Priebeh nárastu okna CUBIC-u je ukázaný na obrázku 4.3 a je možné porovnanie s priebehom nárastu okna protokolu BIC (obr. 4.4).

H-TCP

Podobne do tejto skupiny môžeme zaradiť aj H-TCP [29], rozšírenie, ktoré zmenou Dynamiky rieši problémy štandardného TCP na vysokorýchlostných sieťach. Funguje podobným spôsobom, ako predošlé rozšírenia, preto naznačíme len jeho princíp fungovania.



Obrázok 4.3: Priebeh nárastu okna u CUBIC.



Obrázok 4.4: Porovnanie BIC (prerušovaná čiara) a CUBIC (plná čiara).

Ako u predchádzajúcich rozšírení, aj tu sa rieši problém, kedy je vhodné použiť štandardné TCP algoritmy na predchádzanie zahlteniu a kedy treba už použiť nové metódy. Nech V_L je konštantná hodnota, kedy sa rozhodujeme, aký spôsob využijeme, v_i je čas od poslednej straty zistenej v i-tom toku, $\frac{RTT_{min,i}}{RTT_{max,i}}$ je pomer minimálnej a maximálnej RTT zistenej i-tym tokom, B_i je priepustnosť (kapacita) siete dosiahnutá i-tym tokom tesne pred zistenou stratou. Samotný algoritmus funguje nasledovne:

i) pre každé potvrdenie paketu nastavíme nárastový faktor α_i podľa toho, či sa nachádzame vo fáze štandardného TCP nárastu (pre $v_i \leq V_L$) alebo vo fáze H-TCP (pre $v_i > V_L$) nasledovne

$$\begin{aligned} \alpha_i &\longleftarrow & 1 & & \text{pre } v_i \leq V_L \\ \alpha_i &\longleftarrow & 1 + 10(v_i - V_L) + (\tfrac{v_i - V_L}{2})^2 & & \text{pre } v_i > V_L \end{aligned}$$

a potom nastavíme

$\alpha_i \longleftarrow 2(1 - \beta_i)\alpha_i$

Samotné okno *cwnd* nastavíme ako pre štandardné TCP s novo vypočítaným α_i

$$cwnd_i \longleftarrow cwnd_i + \frac{\alpha_i}{cwnd_i} \tag{4.20}$$

ii) pre každú zistenú stratu

$$\begin{aligned} \beta_i(k+1) &\longleftarrow & 0,5 & & \text{ak} \left| \frac{B_i(k+1) - B_i(k)}{B_i(k)} \right| > 0,2 \\ \beta_i(k+1) &\longleftarrow & \tfrac{RTT_{min,i}}{RTT_{max,i}} & & \text{inak} \end{aligned}$$

kde $\beta_i(k+1)$ je faktor poklesu pre *k+1* cyklus, k nám určuje poradie cyklu. Teda o koľko znížime okno *cwnd* v *k+1* cykle závisí od posledných dvoch nameraných priepustností siete.

Okno *cwnd* pri poklese v *k+1* cykle nastavíme ako pri štandardnom TCP s novou hodnotou $\beta_i(k+1)$

$$cwnd_i \longleftarrow \frac{cwnd_i}{\beta_i(k+1)} \tag{4.21}$$

Autori H-TCP vychádzajú z pozorovania, že zväčšovací faktor α_i by mal byť malý pre štandardné siete s malou kapacitou a patrične veľký pre vysokorýchlostné siete s veľkou kapacitou. Preto sa hlavne sústredili na úpravu faktora α_i, ktorý je funkciou času od poslednej straty. Faktor poklesu β_i musí rýchlo reagovať na zmeny siete, preto na stanovenie miery poklesu sledujeme zmenu priepustnosti siete. To nám dá informáciu o zmene stavu siete, na ktorú vieme reagovať nastavením vhodného β_i.

H-TCP podľa [29] svojím dynamickým prepočítavaním faktorov α_i a β_i rieši problém efektívneho využitia linky nielen pre vysokorýchlostné siete, umožňuje zodpovedné správanie sa tokov na linkách a zaručuje férové zdieľanie prostriedkov siete.

4.1.6 *Delay-Based*

Idea zmeniť spôsob zisťovania zahltenia prišla s TCP Vegas [22]. To síce nie je stavané pre vysokorýchlostné siete, ale správa sa na nich lepšie. Princíp fungovania delay-based princípu je popísaný v predchádzajúcej kapitole, tu si bližšie popíšeme fungovanie TCP Vegas a jeho vylepšenie - rozšírenie Fast TCP [23].

Delay based princípy sa vyznačujú približovaním sa k určitej hranici využitia siete. Táto hranica je stanovená (väčšinou empiricky) tak, aby sa blížila k maximálnej hranici využitia (kapacity) siete (teda čo najefektívnejšie využívala linku), ale aby stále bola od nej dosť ďaleko v prípade jej prekročenia (aby nedošlo k zahlteniu liniek, keď ju prekročíme a pokým sa vrátime k stanovenej hranici). Teda delay based toky oscilujú v stabilnom stave okolo stanovenej hranice, čím zabraňujú prekročeniu maximálnej kapacity liniek (zahlteniu), zahadzovaniam paketov a následným opatreniam algoritmov Predchádzania zahlteniu (hlavne problém s poklesom o polovicu, ako je to u TCP Reno, čo je príliš veľký pokles pre veľké okná).

Delay based princípy zároveň zmierňujú nepomer veľkej kapacity liniek a malých vyrovnávacích pamätí na nich a to tak, že vhodným stanovením hranice, okolo ktorej budú toky oscilovať, sa snažíme udržať posielaciu rýchlosť na linkách v takej miere, aby sa vyrovnávacie pamäte neprepĺňali (resp. plnili do určitej miery). Tým budeme mať na linkách vždy voľné miesto pre prípad výkyvov v posielacích rýchlostiach a nebude potrebné zahadzovať pakety z vyrovnávacích pamätí.

Dynamika Delay-based je rovnakého tvaru ako Dynamika Loss-based rozšírení, vychádza zo vzťahu (2.14). Je však prispôsobená meraniu pozdržania na ceste (queue delay) ako ukazovateľa zahltenia, čo mení výber vhodnej nárastovej funkcie $k_i(w_i, T_i)$, prírastkovej užitočnosti $u_i(w_i, T_i)$ a merania zahltenia, čo tu je miera pozdržania na ceste (queueing delay).

TCP Vegas [22]

Vegas je rozšírenie len na vysielateľovi, preto nie je problém s jeho nasadením. Ide o eliminovanie množstva strát paketov, ktoré samotné TCP vyvoláva, aby zistilo, že sieť je zahltená. Namiesto toho používa proaktívny systém porovnávania očakávaných a nameraných posielacích rýchlostí s predpokladom, že keď sa sieť plní, čas potvrdzovania sa predlžuje.

Vegas zahŕňa tri vylepšenia:

i) Nový preposielací mechanizmus - tu chce Vegas obmedziť čakanie na vypršanie systémových časovačov, ktoré môžu byť veľké (500ms v Linuxe). Vegas si zapisuje čas vyslania paketu. Keď príde potvrdenie, vypočíta nové RTT (presnejší odhad). Keď príde dupAck, skontroluje, či rozdiel medzi súčasným časom a časom, kedy bol poslaný nepotvrdený (preskočený) paket, je väčší ako hodnota časovača. Ak áno, Vegas nečaká na ďalšie dupAck a prepošle segment. Zároveň po strate kontroluje aj obyčajné potvrdenia podobným spôsobom, čím zabraňuje nasledujúcim paketom, aby zbytočne čakali na *dupAck* alebo vypršania časovačov. Zároveň žiadne straty zistené pred zmenou *cwnd* už jeho veľkosť neovplyvňujú. To je z dôvodu, že tu zahltenia zisťujeme ešte pred tým ako sa stanú, teda ak vykonáme nejaké opatrenie, nemôžeme sa pozerať, čo sa stalo pred ním.

ii) Algoritmus predchádzania zahlteniu - Vegas je proaktívny, teda predvída zahltenie. Idea spočíva v myšlienke, že ak zväčšíme *cwnd*, tak sa zväčší aj priepustnosť (posielacia rýchlosť). Ale to sa môže diať len do nejakého bodu, pokiaľ to dovoľuje kapacita linky. Potom každé zväčšenie spôsobuje zaberanie miesta vo vyrovnávacích pamätiach na ceste, čím vzniká zahltenie. Vegas porovnáva predpokladanú a súčasnú posielaciu rýchlosť a snaží sa udržať vhodné množstvo extra dát v sieti, tj. dát, ktoré by neboli poslané, keby kapacita využitá spojením bola rovnaká ako kapacita liniek, cez ktoré ide spojenie (inak povedané dáta, ktoré sa musia bufferovať). Pre zisťovanie posielacích rýchlostí je potrebné meranie a porovnávanie RTT. Rozdiel medzi očakávanou a aktuálnou posielacou rýchlosťou porovná s dvoma hranicami: α (dolná hranica extra dát) a β (horná hranica extra dát). Ak je rozdiel menší ako α, Vegas lineárne zväčší *cwnd* na ďalší RTT, ak je väčší ako β, tak ho zase v ďalšom RTT lineárne zmenší a ak je medzi α a β, nemení nič. Obrazne povedané Vegas kontroluje spojenie, aby použilo na ceste aspoň α a najviac β extra dát vo vyrovnávacích pamätiach. Štandardne zvolená hodnota pre α je 1 a pre β je 3.

iii) Modifikovaný pomalý štart - aby umožnil zistiť a potom aj riešiť zahltenie už počas pomalého štartu, Vegas umožňuje exponenciálny nárast len pre každý druhý RTT. Medzi tým *cwnd* zostáva nezmenené, čo umožní reálnejšie porovnania predpokladanej a aktuálnej rýchlosti. Ak

aktuálna rýchlosť klesne pod očakávanú, Vegas prepne z pomalého štartu na lineárne zrýchľovanie/spomaľovanie. Toto je účinné pri veľkých stratách pri pomalom štarte.

FAST TCP [23]

Ideu o kontrolovaní úrovne zaplnenia siete si zobralo od Vegas-u rozšírenie FAST TCP. Tu sa snažíme zabrániť stratám pri zahltení tým, že si strážime posielaciu rýchlosť spojenia v nejakom intervale.

Problémy, ktoré sa autori rozhodli riešiť pri FAST TCP:

- na úrovni posielania paketov
  - lineárny nárast o jeden paket za RTT je pomalý a multiplikatívny pokles na stratu je veľmi dramatický,
  - oscilácia je neodstrániteľná z dôvodu, že TCP používa binárny signál na zistenie zahltenia (stratu paketu),
- na úrovni tokov
  - udržovanie veľkého *cwnd* potrebuje extrémne malé výkyvy od stabilného stavu (equilibrium),
  - dynamika je nestabilná, vedie k osciláciám, ktoré môžu byť redukované správnym odhadom stratovosti paketov.

Autori rozdelili funkcionalitu rozšírenia na štyri časti:

1. odhad ceny (Estimation) - používané ostatnými troma časťami,
2. kontrola dát - stará sa o to, ktoré pakety poslať,
3. kontrola okna - stará sa o to, koľko paketov poslať,
4. kontrola zhlukovosti (Burstiness control) - stará sa o to, kedy poslať.

Algoritmus:

- ***Odhad ceny*** - 2 typy informácií pre poslané pakety - multibit queuing delay a one-bit loss-or-no-loss indicator. Keď príde pozitívne potvrdenie na paket, tak počítame RTT pre ten paket a použijeme ho na výpočet minimálneho RTT a priemerného zdržania na ceste. Keď obdržíme negatívne potvrdenie (3 dupAck alebo vypršanie časovača), vygenerujeme informáciu o strate a pošleme ho ostatným komponentom. Takto máme pre jeden paket jednobitovú informáciu o strate a multibitovú informáciu o zdržaní na ceste. Miera zdržania (queueing delay) je vyhladená a upravená mierou závislou od veľkosti okna.

- ***Kontrola dát*** - Táto časť sa zaoberá zaradením paketu na poslanie do 3 skupín: 1) nové, 2) negatívne potvrdené a 3) poslané ešte nepotvrdené. Keď nie je zistená strata, posielajú sa nové pakety z prvej skupiny ako odozvy na potvrdenia starých paketov - toto zodpovedá mechanizmu self-clocking (ack-clocking). Počas zotavovania sa zo straty (loss recovery) treba rozhodnúť, čo pošleme. Tu sa kontrola dát stará o vhodné rozdelenie prenosovej kapacity medzi 3 skupiny. To sa deje v závislosti od štatistík získaných z časti na odhad ceny.

- ***Kontrola okna*** - Samotné kontrolovanie zmeny okna a jeho veľkosti je podobné klasickému TCP. Rozdiel je v tom, že výpočet okna je nezávislý od aktuálneho stavu vysielateľa. Kontrola okna reaguje na stratu paketu a jeho zdržanie. Podľa týchto informácií upravuje veľkosť okna.

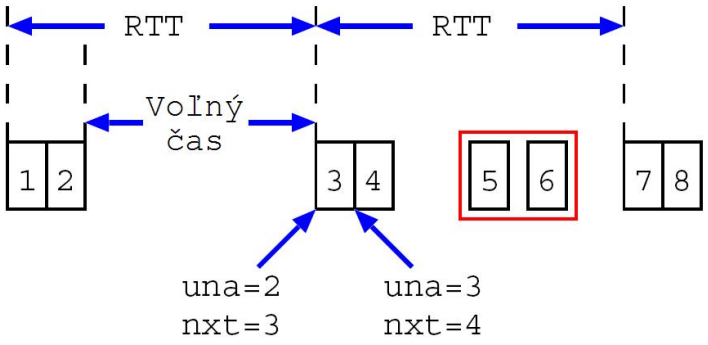
  $w \longleftarrow \min\{2w, (1-\beta)w + \beta(\frac{baseRTT}{RTT}w + \alpha(w, qdelay))\}$

  kde $\beta \in (0, 1]$, $baseRTT$ je doteraz najmenšie zistené RTT a $qdelay$ je end-to-end zdržanie (priemerné). Tu uvažujeme zmenu okna len každé druhé RTT, čo nám zaručuje výpočet priemerného prípadu ak sa stane niečo neočakávané. Autori zvolili funkciu $\alpha(w, qdelay)$ ako konštantnú, čo zaručuje lineárnu konvergenciu k stabilnému stavu ak $qdelay$ je nulové.

- ***Kontrola zhlukovosti*** - Stará sa o vyvážené posielanie dát. Jej funkcionalita sa skladá z dvoch algoritmov

  - *Redukcia zhlukovosti* - Regulovanie vysielacej rýchlosti na nejaký limit, ktorý je počítaný pri každom prijatí potvrdenia ACK. Nové pakety sú poslané, ak zistená rýchlosť je menšia ako stanovená hranica, inak sú pozdržané. Tu sa rozhoduje, koľko paketov sa má poslať po obdržaní potvrdenia, ktoré môže zvýšiť *cwnd* na veľkú hodnotu, a tým sa snaží redukovať zhlukové posielanie dát na malých intervaloch v rámci jedného RTT.
  - *Posúvanie okna (window pacing)* - Snaha o hladké zväčšovanie veľkosti *cwnd* tak, že sa jeho zväčšenie nevykoná naraz, ale zväčšuje sa aj vtedy, keď sa nič nevysiela. Tu treba nájsť práve tie intervaly, kedy sa nič neposiela. To sa zisťuje z rozdielu časov posledného poslania paketu a aktuálneho času po prijatí potvrdenia. Ak ten rozdiel je väčší ako nejaký limit, považujeme ho za dosť dlhý na použitie pacingu a rozdistribuovanie prírastku (window increment) na daný interval.

    Obrázok 4.5 zobrazuje princíp fungovania algoritmu počas pomalého štartu. Najskôr sa v prvom cykle (prvý RTT) pošlú pakety 1, 2 a zisťuje sa voľný čas, ktorý sa neskôr využije na dodatočné poslanie paketov. **una** znamená prvý nepotvrdený paket, **nxt** je



Obrázok 4.5: Posúvanie okna (window pacing) u FAST TCP.

nasledujúci paket na poslanie. Následne po potvrdení paketov 1 a 2 sa pošlú pakety 3 a 4 a ako nárast v pomalom štarte sú pakety 5 a 6, ktoré sú poslané počas zisteného voľného času.

Celkové fungovanie je nasledovné. Po prijatí každej ACK, Odhad ceny vypočíta priemerné zdržanie (queueing delay). Kontrola zhlukovosti určí, či poslať pakety do siete. Pre každý poslaný paket si Odhad ceny uloží čas odoslania a Kontrola zhlukovosti si aktualizuje svoje štruktúry. Kontrola okna si pravidelne vypočíta novú veľkosť okna. Na konci RTT burstiness reduction vypočíta cieľovú priepustnosť (target throughput) s použitím okna a merania RTT z minulého kroku. Window pacing potom naplánuje rozdelenie veľkého prírastku *cwnd* na menšie a určí čas, kedy sa nič neposiela. Počas zotavovania zo straty *cwnd* je stále upravované ako normálne s použitím informácií o zahltení zo siete. Po každom zistení straty paketu odosielateľ určí, či preposlať alebo pozdržať paket.

Dynamika FAST TCP je charakterizovaná dvoma funkciami a tým, že miera zahltenia q_i je vyjadrená pozdržaním na ceste (queueing delay).

$$D_i(t) = \beta\alpha_i \left(1 - \frac{q_i(t)}{\frac{\alpha_i}{x_i}}\right) \tag{4.22}$$

alebo

$$k_i(w_i, T_i) = \beta\alpha_i \tag{4.23}$$

$$u_i(w_i, T_i) = \frac{\alpha_i}{x_i} = \alpha_i \frac{T_i}{w_i} \tag{4.24}$$

kde $x_i = \frac{w_i}{T_i}$ je rýchlosť posielania, β a α_i sú parametre pre FAST TCP.

Reakčná funkcia v stabilnom stave má tvar

$$cwnd = \frac{\alpha_i}{q_i} \tag{4.25}$$

4.1.7 *Loss-based vs. Delay-Based*

Štandardné princípy zisťovania zahltenia Loss based a Delay based majú spoločný základ. V podstate Delay based princíp u TCP Vegas je len upravenou a vylepšenou verziou Loss based u TCP Rena. Ako sme ukázali, pre obidva princípy existujú aj ich rozšírenia stavané na vysokorýchlostné siete. Otázkou je, ktorý princíp je vhodnejší na použitie u takýchto sietí. Bohužiaľ tu sa nedá jednoznačne povedať, ktorý je lepší a ktorý je horší. Každé má svoje výhody a problémy v závislosti od ich plánovaného využitia a nasadenia.

Ďalej sa budeme zaoberať niektorými porovnaniami oboch princípov. Tu chceme podotknúť, že rozdiely na štandardných sieťach sú zanedbateľné, preto sa zaoberáme len vysokorýchlostnými sieťami.

Výhody Delay-based systémov nad Loss-based:

- pozdržanie v rade (queuing delay) je presnejšie vypočítateľné ako stratovosť (loss probability), lebo na vysokorýchlostných sieťach sú straty zriedkavé,
- na rozdiel od jednobitovej informácie v loss-based o strate máme v delay-based viacbitovú informáciu o zahltení, vďaka čomu môžeme ľahšie dosiahnuť stabilný stav a zaručiť určitý stupeň férovosti,
- dynamika delay-based rozšírení sa zdá, že má presnejšie nastavenie miery na kapacitu linky. Vyplýva to z jej zmeny v závislosti od zaplnenia vyrovnávacích pamätí na ceste a meraní RTT.

Nevýhody Delay-based systémov oproti Loss-based:

- počítanie zdržaní na ceste u delay-based systémov je závislé od prostredia a okolitých vplyvov - takzvaný šum. Najpresnejší odhad je v prípade nulového šumu. Tento šum môže nastať pri nasadení systému v heterogénnom prostredí, kde je problém v rozdielnych bufferovacích schémach, samotné operačné systémy svojimi plánovacími algoritmami (scheduling) tiež skresľujú merania zdržaní na ceste (queue delay) a RTT, a zároveň značným problémom sú obľúbené firewall-y, ktoré preverujú pakety, na čo potrebujú určitý čas, ktorý pridáva šum v meraní queue delay. Ďalším generátorom šumu sú aj súperiace spojenia, ktoré sa nesprávajú TCP priateľsky, teda nekontrolujú zapĺňanie vyrovnávacích pamätí na ceste a neupravujú svoju rýchlosť podľa toho. Tieto spojenia sú väčším problémom pre Delay-based princípy ako Loss-based, ktoré sa správajú agresívnejšie vo využívaní zdrojov siete.

Kapitola 5

Rozšírenia TCP pre bezdrôtové siete

Bezdrôtové siete, ako sme opísali vyššie, majú veľa vlastností, ktoré bránia štandardnému TCP efektívne zabezpečiť prenos dát sieťou. Aby sme mohli plne využívať takéto typy sietí, musíme brať do úvahy charakteristiky danej siete a oblasti jej použitia.

Pre rôzne typy bezdrôtového spojenia už vznikli mnohé rozšírenia protokolu TCP. Situácia je však taká, že ich súčasná evidencia je nepostačujúca. Navyše mnohé z týchto rozšírení sa týkajú inej oblasti ako ostatné a vyzdvihujú svoje kladné stránky. Nie je zavedená žiadna klasifikácia a porovnávanie podľa určitých kritérií.

V tejto kapitole podávame klasifikáciu bezdrôtových sietí z pohľadu viacerých vybraných kritérií. Podľa nich zatriedime niektoré vybrané rozšírenia.

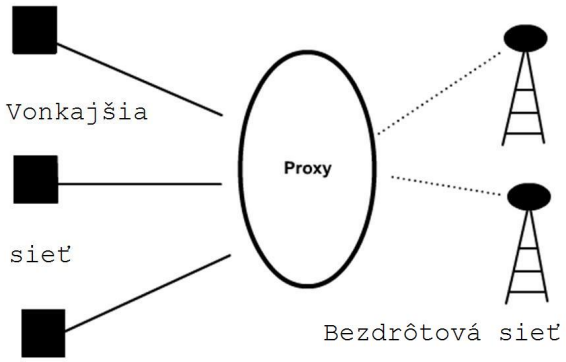
5.1 *Klasifikácia rozšírení TCP pre wireless siete*

V tejto podkapitole vychádzame zo [6] a pridávame ďalšie možnosti klasifikácie. Ku každému deleniu opíšeme jeho vlastnosti, stanovíme klady a zápory a určíme dôležitosť daného delenia.

Nasledujúce delenia nie sú disjunktné, preto sa budeme snažiť stanoviť ich optimálne spojenie. Toto by malo tvoriť kľúč k prehľadu jednotlivých rozšírení a vhodnosť ich použitia pre danú oblasť.

5.1.1 *End-to-end v.s. Split*

Štandardný TCP je vo svojej podstate End-to-end protokol. To znamená, že celé spojenie sa tvári, akoby bolo len medzi komunikujúcimi zariadeniami. Napríklad aj potvrdzovanie paketov sa prevádza až po doručení paketu do cieľa.



Obrázok 5.1: Princíp rozdelenia siete na dve časti: klasickú a mobilnú bezdrôtovú.

Idea End-to-end je správať sa k heterogénnej sieti (klasická drôtová časť - wired a bezdrôtová časť - wireless) ako k celku, teda zachovať pôvodné chápanie siete z pohľadu TCP.

Idea Split je založená na myšlienke rozdeliť heterogénnu sieť na časti: drôtovú wired a bezdrôtovú wireless. Potom by komunikácia a hlavne potvrdzovanie paketov fungovalo v dvoch častiach (resp. v toľkých častiach, na koľko máme rozdelenú sieť). Poslaný paket prichádzajúci do rozhrania (medzirozhrania) je potvrdený odosielateľovi ako doručený, a ďalej sa posiela prijímateľovi. Pritom posielajúce medzirozhranie čaká na potvrdenie ACK od prijímateľa (resp. ďalšieho medzirozhrania), kde poslal paket. Takto sú dáta preposielané a potvrdzované po častiach. Toto je načrtnuté na obrázku 5.1.

V rôznych literatúrach a pre rôzne rozšírenia TCP je toto rozdelenie brané inak. Niektoré sa spoliehajú na štandardný end-to-end systém, ktorý je prirodzený pre TCP a presne vie určiť stav doručenia dát. Na druhej strane prívrženci Split schémy tvrdia, že použitím rozdelenia siete na časti a postupným overovaním dokážu efektívne určiť problém straty paketu (zistenie, či ide o zahltenie alebo šum) a jeho preposlanie, keďže paket sa musí v rozhraniach bufferovať až do potvrdenia.

Výhody a nevýhody End-to-end:

+ nie je potrebné robiť zásahy do prepojovacích rozhraní, TCP funguje ako štandardné - teda medzi komunikujúcimi zariadeniami a nie je potrebné využívať prepojovacie zariadenia, ktoré by museli mať dané rozšírenie TCP,

+ po obdržaní potvrdzovacej správy ACK vieme povedať, že paket určite dosiahol cieľ,

+ štandardne sa dá identifikovať zahltenie,

- často sú bezdrôtové wireless časti spomaľujúce faktory siete (kvôli prenosovej kapacite a stratovosti), preto musia všetky sieťové komponenty na ceste paketu čakať práve na doručenie paketu v drôtovej sieti a tá musí mať vhodne vyriešené bufferovanie,

- kvôli častým stratám paketov v bezdrôtovej časti, musia byť pakety preposielané zo zdroja,

- aby bola zachovaná komunikácia a využívanie daného rozšírenia, museli by mať oba komunikujúce zariadenia rovnaké rozšírenie TCP.

Výhody a nevýhody Split:

+ rozdelením siete na časti wired (drôtová) a wireless (bezdrôtová) umožňuje použitie rozdielnych techník pre dané časti, čo zvyšuje efektivitu prenosu na jednotlivých častiach. Nutné je potom prepojenie na rozhraní,

+ umožňuje efektívne preposielanie paketov, ktoré jednou časťou prešli v poriadku a v druhej sa stratili - preposlanie z rozhrania,

+ ľahké zistenie pôvodu straty paketu - vieme, že wired časti majú vysokú spoľahlivosť, preto predpokladáme straty spôsobené nespoľahlivou linkou len pre wireless,

- potrebné bufferovanie v rozhraní. Tu treba spomenúť aj dodatočné riešenie problému, ak máme rozhranie medzi vysoko priepustnou a nízko priepustnou sieťou tzv. úzke hrdlo (bottleneck). Potom je potrebné kontrolovať zahltenie vyrovnávacej pamäte a predchádzanie i zotavovanie sa z neho,

- nekonzistencia medzi doručením ACK do zdroja a doručením paketu. Keď odosielateľovi príde ACK z rozhrania, neznamená to, že došli aj do cieľa resp. či došli do neho dostatočne skoro. Niektoré Split riešenia riešia tento problém čakaním poslania ACK z rozhrania na potvrdenie z druhej časti siete. Toto si zase vyžaduje dodatočnú réžiu,

- pri mobilných sieťach treba vyriešiť prechod medzi bunkami a doručením paketu z rozhrania k zariadeniu, ktoré medzičasom prešlo pod ďalšiu bunku. Je potom starosťou rozhrania, aby preposlalo paket na novú adresu do novej bunky, pričom musí sledovať prechody zariadení medzi bunkami. To je dosť náročné na implementáciu a efektivitu.

5.1.2 *Local vs. Glogal*

Toto kritérium sa zaoberá umiestnením zmien rozšíreného TCP. Lokálne zmeny sa týkajú len komponentov siete, ktoré sú pod správou poskytovateľa služieb na danej bezdrôtovej sieti. Týka sa to či už vysielačov/prijímačov (base station) ako aj mobilných hostov. Ak zavedenie daného rozšírenia si vyžaduje aj zmeny mimo bezdrôtovej siete, považujeme ho za globálne rozšírenie.

Globálne riešenia majú možnosť využitia v menších uzavretých sieťach alebo v častiach veľkých sietí, ktoré sú potom nejakým spôsobom prepojené s ostatným svetom. Lokálne riešenia sú vhodné pre pripájanie bezdrôtových sietí k väčším sieťam ako je aj Internet.

Výhody a nevýhody Global riešení:

+ jednotnosť v prenosovom protokole - nevyžaduje sa žiadne dodatočné úsilie na prepojenie častí siete s rozdielnym rozšírením TCP,

+ možnosť inkrementálnej úpravy/rozširovania - ale vyžaduje si sledovať a dodržiavať upgradovanie protokolu u všetkých zariadení,

- náročné na zavádzanie - kvôli niekoľkým bezdrôtovým linkám, ktorých v porovnaní s klasickými drôtovými spojeniami je podstatne menej (v súčasnom Internete), si pevný host (v bezdrôtovej časti) nebude rozširovať TCP, ktorý by aj tak priniesol prospech pre zlomok spojení,

- vyžaduje správu nad celou sieťou a zaručením jednotnosti protokolov.

Výhody a nevýhody Local riešení:

+ vyžaduje si len lokálne zmeny v komponentoch bezdrôtovej siete,

+ ľahká správa,

- treba spojenie medzi drôtovými a bezdrôtovými časťami na úrovni rôznych protokolov (rozšírení).

5.1.3 *Transparent vs. Snooping*

Hlavným problémom u bezdrôtových sietí je stratovosť paketov z dôvodu nespoľahlivosti liniek a ich odlíšenie od straty paketu z dôvodu predchádzania zahlteniu. Toto kritérium hovorí o spôsobe zisťovania príčiny straty paketov.

Snooping riešenie je postavené na dodatočnej informácii o doručovaní a potvrdzovaní paketov. Na zistenie tejto dodatočnej informácie potrebuje čítať alebo aj zapisovať do hlavičky paketu. Toto sa väčšinou odohráva na rozhraní wired/wireless (base station). Snooping riešenia často požadujú

podporu zo strany nižších vrstiev, aby získali informáciu o stave linky a tak určili dôvod straty.

Transparentné riešenia sú také, ktoré nevyžadujú čítanie hlavičky TCP paketov.

Nasledujúce dva protokoly sú príslušníkmi snooping riešenia. **Snoop Protocol** [6] v base station má tzv. Snoop agenta. Ten pozerá do hlavičiek TCP paketov vstupujúcich do bezdrôtovej siete a cache-uje ich. Zároveň si cache-uje aj prichádzajúce ACK na pakety. Porovnaním odoslaných a potvrdených dát potom vie odhaliť stratu paketu na nespoľahlivej linke a preposlať príslušný paket, poprípade s použitím ďalších techník ju odlíšiť od straty z dôvodu zahltenia.

Ďalším snoop rozšírením je **Explicit Loss Notification** (ELN), ktorý miesto cache-ovania paketov využíva bit v TCP hlavičke na identifikáciu miery zahltenia siete.

Výhody a nevýhody Snoop riešení:

+ dokážu odhaliť pravú príčinu straty paketov, a tak efektívne na ňu reagovať,
+ už počas komunikácie získame informácie o doručených paketoch,
- vyžaduje dodatočný mechanizmus na kontrolu paketov a ich ACK potvrdení,
- najväčší problém u takýchto riešení je potreba kontroly hlavičky TCP protokolu. Tým pádom sa nemôžu aplikovať v sieťach, ktoré používajú šifrovaný prenos dát čitateľný len v koncových bodoch spojenia (encrypted networks - IPSec).

Výhody a nevýhody Transparent riešení:

+ nie je potreba žiadnej dodatočnej funkcionality na odhalenie dôvodu straty,
+ možnosť použitia schém, ktoré zabezpečia šifrovanie dát a možnosť ich čítania len v koncových hostoch - IPSec,
- problém je nájsť vhodný algoritmus na zistenie dôvodu straty a jej riešenia.

5.1.4 *Two-way vs. One-way*

V závislosti od toho, či samotná bezdrôtová komunikácia je jednosmerná alebo dvojsmerná, môžeme podľa rovnakého kritéria deliť aj rozšírenia TCP pre tieto siete. Napriek spomenutiu tohto kritéria si myslíme, že nie je také významné ako ostatné.

Výhody a nevýhody One way riešení:

+ v mobilných sieťach, kde je mnohonásobne väčšia prenosová kapacita v smere download, alebo v sieťach, ktoré vyžadujú hlavne komunikáciu v jednom smere alebo majú komunikáciu oddelenú,

- väčšinou treba riešiť komunikáciu tam aj naspäť.

Výhody a nevýhody Two way riešení:

+ rieši komunikáciu ako celok,

- ak vyslovene treba kontrolovať len jeden smer komunikácie.

5.1.5 *Intermediate-link vs. Last-Hop*

Toto kritérium sa zaoberá umiestnením bezdrôtovej siete v rámci globálnej siete (Internet). Intermediate link riešenie predpokladá umiestnenie bezdrôtovej siete do ľubovoľnej časti globálnej siete. Toto berie do úvahy prepojenia častí bezdrôtovými linkami, napríklad satelitné spojenie. Last Hop riešenie je podmnožinou prvej možnosti. Týka sa umiestnenia bezdrôtových sietí na okraj globálnej siete.

Dôvodom, prečo vôbec rozmýšľať len o použití na okraji siete, sú siete, ktorých priepustnosť dát je malá, má veľkú stratovosť dát alebo má iné nepriaznivé vlastnosti, kvôli ktorým nie je vhodné použitie na spojenie častí ostatnej siete. Aj napriek týmto nedostatkom je ich použitie na okraji vhodné. Preto je snaha optimalizovať premávku na takejto sieti. Ako príklad môžeme uviesť mobilnú komunikáciu.

Výhody a nevýhody Intermediate-link riešení:

+ univerzálnosť použitia,

- neefektívnosť pri špecifických sieťach (mobilné,...).

Výhody a nevýhody Last-Hop riešení:

+ pre špecifické siete, efektívne využitie vlastností,

- nie je vhodné použiť pre iné ako okrajové pripojenie siete.

5.1.6 *Signaling vs. Hiding*

Celé toto kritérium sa týka protokolu TCP len okrajovo, ale v podstate rieši základný problém TCP bezdrôtových sietí a to straty paketov z dôvodu nespoľahlivosti linky. Ide o vysporiadanie sa nižších vrstiev linky so stratami paketov.

Rozšírenie je signalujúce (Signaling), ak nižšie vrstvy zisťujú dôvody chýb a posielajú to na riešenie TCP protokolu. Takými sú napríklad ELFN (Explicit Link Failure Notification) a BEAD (Best Effort ACK Delivery) popísané v [36].

Rozšírenie je Hiding, ak zakrýva wireless straty vyšším vrstvám (v našom prípade TCP). V tomto prípade ide často o vyriešenie strát z dôvodu nespoľahlivosti wireless linky na úrovni nižšej ako TCP (napr. link layer). Medzi takéto riešenia patrí napríklad štandard 802.11 a jeho MAC vrstva (popísané v [35]) a TCP ACK Congestion Control and Filtering [34].

Ako ich doplnok sú siete, keď nižšie vrstvy neriešia stratovosť a ani neposielajú žiadnu informáciu o strate. Vtedy sa musí TCP samo vysporiadať so stratou.

Výhody a nevýhody Signaling riešení:

+ presné zistenie straty a jej dôvodu,
+ dá sa zobrať vhodné riešenie daného stavu a použiť aj na samotné spojenia (nie len pakety) a využiť aj neštandardné riešenia - v závislosti od danej siete (štatistiky pre spojenia,...),
- treba meniť TCP, aby implementovalo signály z nižších vrstiev.

Výhody a nevýhody Hiding riešení:

+ skrýva nespoľahlivosť fyzickej linky, teda nie je nutné meniť TCP,
- zväčša potreba špecifickej schémy pre špecifické zariadenia/siete,
- pri použití tvarovania siete môže dôjsť k mätúcim informáciám - z dôvodu zahadzovania paketov pri prevencii proti zahlteniu.

Niektoré špecifické nevýhody Hiding riešenia napr. u MAC vrstvy pre 802.11 sme popísali v kapitole **3. Problémy štandardného TCP pre špecifické typy sietí** v časti **3.2.3. Podpora nižších vrstiev**.

5.1.7 *Centralizované vs. Distribuované*

Toto rozdelenie sa zaoberá správou bezdrôtovej siete. Centralizovaná sieť má koordinátora, ktorý prideľuje bezkolízny prístup uzlom, ktoré si vyžiadajú komunikáciu od koordinátora.

Distribuovaná sieť (Ad-hoc) nemá v sieti entitu, ktorá by zobrala zodpovednosť za správu a prideľovanie prostriedkov siete jednotlivým uzlom. Uzly sa pripájajú samostatne na základe vopred stanovenej distribučnej funkcie.

IEEE 802.11 má v sebe zabudované obe riešenia - centralizované Access Point a distribuované Distributed coordinator function (DCP).

Výhody a nevýhody Centralizovaných riešení:

+ celá sprava je kontrolovaná jednou entitou, ktorá má prehľad o situácii v sieti,
+ možné vypracovanie (centralizovaných) štatistík a meraní,
+ centralizovaná správa zahltení a kontroly tokov,
- vyžaduje, aby každý uzol vedel, kto je koordinátor,
- algoritmus pre koordinátora - tak aby zodpovedal danej sieti.

Výhody a nevýhody distribuovaných riešení:

+ netreba mať znalosť o architektúre, resp. poznať koordinátora,
- algoritmus pre distribučnú funkciu - vhodné, aby bola rovnaká pre všetky uzly.

5.2 *Niektoré vybrané rozšírenia*

5.2.1 *Multi-Radio Unification Protocol*

Multi-Radio Unification Protocol (MUP) [11] je rozšírenie nad IEEE 802.11 pre multi-hop siete. V tomto rozšírení si autori vybrali za cieľ transparentnosť nad existujúcou sieťou, rozšírenie si nevyžaduje žiadne zmeny nad aplikáciami, transport a routing protokolmi, funguje so starým hardware-om a nevyžaduje znalosť o topológii siete. Rozšírenie je zaujímavé z pohľadu centralizovaného rozdelenia.

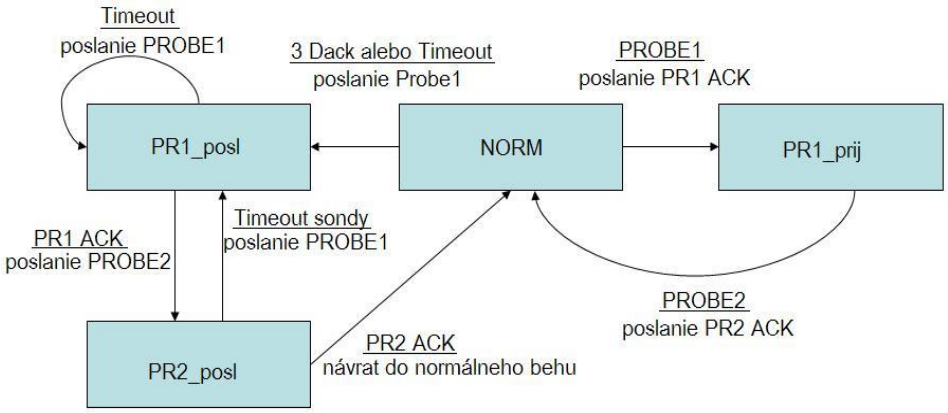
MUP rieši problém úzkeho hrdla (bottleneck) u bezdrôtových sietí a tým pomáha predísť k zahlteniu hranice medzi dvoma časťami siete.

Ako hlavný problém, ktorý rieši MUP, je jednostranné prijímanie alebo vysielanie, s ním spojené potrebné bufferovanie a využívanie len malého pásma na vysielanie.

Riešením je vytvoriť skupinu hlavných uzlov v sieti, ktoré sa správajú ako jeden a ony si rozdeľujú pásma na komunikáciu. Tým sa dá čiastočne kontrolovať a usmerňovať premávka na sieti. Algoritmus funguje na princípe Work Balancing.

Komunikácia medzi uzlami v sieti a hlavným uzlom prebieha cez hlavný uzol, ktorý vytvorí asociáciu medzi uzlom v sieti a uzlom vo svojej skupine hlavných uzlov a ďalšia komunikácia už prebieha medzi nimi.

Rozšírenie je End-to-end, transparent, hiding, local a centralizované.



Obrázok 5.2: Stavový diagram mechanizmu sondovania.

5.2.2 *TCP-Probing*

TCP-Probing [13, 14] - toto rozšírenie má za úlohu určiť príčinu straty a podľa toho zvoliť vhodnú stratégiu na jej zotavenie. Probing pracuje nad klasickou drôtovou architektúrou s koncovými bezdrôtovými hostami. Riešenie si stanovilo 3 ciele: zaručiť férovosť nad tokmi, optimálne využitie prostriedkov siete a Congestion avoidence (predchádzanie zahlteniu).

Rozšírenie chce rozlíšením zahodenia paketu rozhodnúť, či bol paket zahodený z dôvodu straty na linke alebo z dôvodu zahltenia. Vtedy môže použiť vhodnú stratégiu na zotavenie sa zo straty.

Algoritmus: Po strate paketu algoritmus zastaví posielanie a vojde do Probing cyklu. V ňom nastáva posielanie tzv. sond (Probes), ktoré sú vlastne prázdne pakety len s hlavičkou. Tento cyklus má za úlohu zistiť príčinu straty paketu. Využíva odhadovanie na základe miery zahltenia (contention level) a podľa toho sa rozhodne, či použije agresívny algoritmus Error Recovery, alebo pokračuje ďalej v posielaní v prípade straty paketu z dôvodu nespoľahlivej linky.

Probing cyklus je zobrazený stavovým diagramom na obrázku 5.2 a popísaný nasledovne. Pri zistení straty (3 DupACK alebo vypršanie časovača) v stave NORM (normálny beh bez zistenia zahltenia) posielame sondu PROBE1 a prechádzame do stavu PR1_posl. Ak tam zistíme timeout, preposielame sondu PROBE1, ak dostávame PR1_ACK - potvrdenie na PROBE1, tak posielame sondu PROBE2 a prechádzame do stavu PR2_posl. Ak v PR2_posl identifikujeme timeout, posielame sondu PROBE1 a vraciame sa do stavu PR1_posl, inak po obdržaní potvrdenia PR2_ACK na sondu PROBE2 ukončujeme Probing cyklus a vraciame sa do stavu NORM. Naopak, keď sme v stave NORM a dostaneme správu PROBE1, posielame na ňu potvrdenie PR1_ACK a prechádzame do stavu PR1_prij. Tam čakáme

na doručenie sondy PROBE2, ktorú potvrdzujeme poslaním PR2_ACK a vraciame sa do stavu NORM.

Error-Recovery Mechanism: Probing cyklus neskončí, ak nenastanú aspoň 2 úspešné RTT merania (doručenie dvoch sond). Z nich potom získa informácie pre zotavovaciu (recovery) stratégiu a to nasledovne: ak obe merania ukážu veľkú mieru sporu (contention), pokračujeme ako TCP Reno v zotavovaní. Ak merania ukážu, že máme dostatok linky, zotavovanie nenastáva a pokračujeme v prenose ako pred stratou paketu, ktorý cyklus vyvolal (Immediate Recovery). Odosielateľ vojde do Probing cyklu po 3 DupACK, alebo keď vyprší časovač.

Odhad sporu: každá RTT sonda počíta Congestion Predictor podobne ako je to u TCP-Vegas (vegas predictor - miera zahltenia). Ak predictor je väčší ako congestion window *cwnd*, teda sme zaregistrovali zahltenie, tak po obdržaní 3 DACK nevbehneme do Probing cyklu, ale do Slow Start-u ako u TCP-Reno.

Zároveň je vhodné uvažovať v nespoľahlivom prostredí o zachovávaní rovnakého času pre časovač, lebo nie je potrebné čakať čoraz dlhšiu dobu na skončenie výpadku, ale posielať prázdne sondy, aby sme mali informáciu o znovu priechodnej linke.

Rozšírenie posielaním prázdnych sondovacích paketov počas Probing cyklu sleduje aj ďalší problém - šetrenie energie. V štandardnom TCP by sme počas výpadku posielali toľko paketov, koľko by nám dovolilo posielacie okno. Tu naopak pošleme len jednu sondu a navyše bez obsahu. Tým ušetríme veľkú časť energie, hlavne pri veľkých výpadkoch.

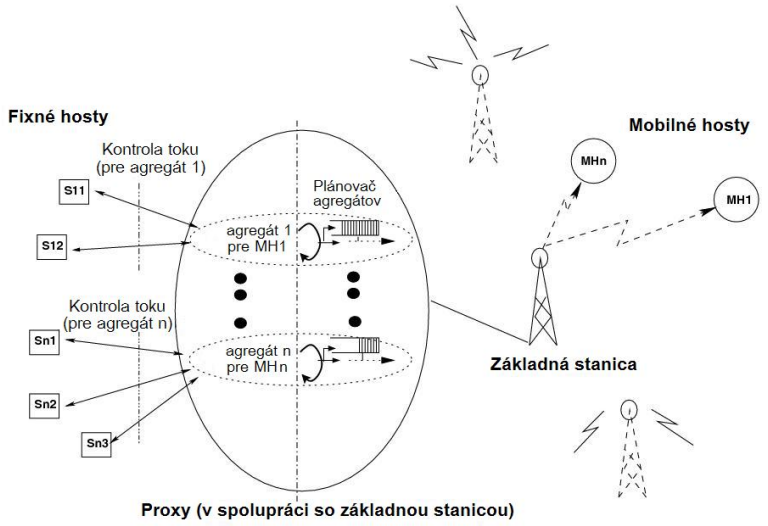
Toto rozšírenie je End-to-end (ale môže sa použiť aj v Split systémoch), lokálne (týka sa len zmien u odosielateľa), transparentné (ak sa použije ako Split, potom si vyžaduje čítanie hlavičiek paketov - Snooping) a nepotrebuje podporu z nižších vrstiev. Je vhodné pre mobilné siete, kde nastávajú často dlhšie výpadky (čiže sa používa ako Last-Hop rozhranie).

5.2.3 *Split TCP*

Split TCP [15] je riešenie zamerané hlavne pre mobilné GPRS siete. Tie predpokladajú malú premávku a presúvanie sa. Cieľom rozšírenia je zlepšiť správanie sa TCP nad GPRS a pritom nemeniť sieťový protokol.

Rozšírenie predpokladá existenciu proxy uzlu a je postavené na rozdelení siete na dve časti: z vonkajšej drôtovej siete po proxy a od proxy k mobilným uzlom. Obrázok 5.3 nám ukazuje možné nasadenie rozšírenia, kde vonkajšia sieť je spojená s mobilnou sieťou cez proxy uzol v spolupráci so základnými stanicami (base station), v ktorom sa riešia problémy vyplývajúce z bezdrôtovej a mobilnej komunikácie.

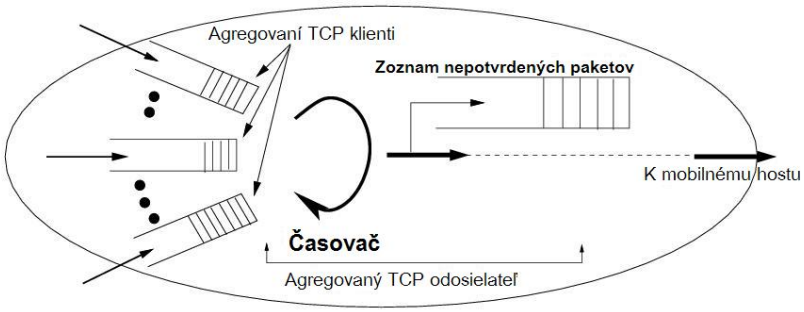
Táto architektúra má vylepšiť štandardné TCP nad GPRS v týchto bodoch: využitie linky, férovosť (založenej na čase alebo RTT), zistenie chýb a zotavenie z nich, efektívny mechanizmus na kontrolu premávky. Posledný



Obrázok 5.3: Princíp rozdelenia siete na dve časti: klasickú a mobilnú bezdrôtovú. Sij je j-ty fixný host komunikujúci s agregátom i a HMi je i-ty mobilný host.

bod rieši tzv. Agregate flow mechanism (agregovanie tokov), kde dochádza k spájaniu tokov do jedného agregátu na základe nejakého kritéria napr. toky vedúce do/z jedného uzla - majú podobné adresovacie informácie (shared state learning) a informácie o aktuálnom stave linky (share congestion wnd a RTT). Tieto informácie sa ukladajú do blokov pre každý tok - Agregate Control Block (ACB). Celé spájanie prebieha v proxy uzle.

Proxy má dve fronty, jedna pre drôtovú a jedna pre bezdrôtovú časť linky. Pre každé spojenie má vlastnú frontu, kde agreguje spojenia. Paket, ktorý má byť doručený mobilnému uzlu je prijatý do fronty v proxy. Ten pošle predpotvrdenie vysielajúcej strane a potom sa snaží doručiť paket do cieľa.



Obrázok 5.4: Logický agregát v proxy pre daný mobilný host.

Ak sa stratí paket v mobilnej sieti, paket sa z proxy uzla prepošle. Tento mechanizmus je zobrazený na obrázku 5.4. Zároveň tu môže proxy spájať dokopy aj jednotlivé pakety ak boli dostatočne malé, čím zvyšuje využitie linky. Tu je problém s bufferovaním prichádzajúcich paketov a znížením prenosovej kapacity v mobilnej časti siete. Preto sa na predpotvrdzovanie používajú viaceré techniky napr. predpotvrdenie sa posiela, až keď sa pošle paket koncovému uzlu, resp. až keď sa potvrdí ten paket. Zároveň sa nastavuje aj veľkosť prijímacieho okna, čím sa reguluje množstvo dát, ktoré môže v danom okamihu proxy prijať. Toto regulovanie prijímacieho okna regulujeme podľa voľného miesta vo vyrovnávacej pamäti a podľa stavu mobilných liniek.

Mechanizmus má zdieľanú informáciu o veľkosti okna zahltenia *cwnd* pre celý agregát tokov. Tá je fixná na relatívne počítanú hodnotu Bandwith Delay Product (BDP). Tým vlastne preskakujeme pomalý štart, ktorý spôsobuje nevyužitie linky na začiatku prenosu malých dát, ktoré tvoria podstatnú časť prenosu cez GPRS linky.

Error detection algoritmus na mobilnej časti siete - v tomto rozšírení vďaka proxy architektúre sú dve hlavné príčiny chýb na mobilnej časti siete. Prvou príčinou sú zhlukové straty (bursty radio losses), ktoré trvajú dlhšie ako je linková vrstva pripravená k preposlaniu paketu. Druhou príčinou sú reselekcie nastávajúce pri prechode medzi bunkami a môžu vyústiť až do pozastavenia linky na niekoľko sekúnd. TCP štandardne zisťuje tieto straty z doručenia dupAck alebo v najhoršom prípade z vypršania časovača. Tu by štandardný TCP vyvolal algoritmus rýchleho preposlania, no keďže sa linka väčšinou po takýchto stratách vráti do pôvodného stavu, algoritmy predchádzajúce zahlteniu by viedli k nevyužitiu linky. Mechanizmus TCP-SACK nám vie povedať, že sme doručili dáta mimo poradia. Keď použijeme tento mechanizmus s agregovaným systémom a berieme do úvahy, že nad GPRS linkami nedochádza k preusporiadaniu dát (veľmi malá pravdepodobnosť kvôli priamemu spojeniu proxy s uzlami resp. využitie len niekoľko skokov pri ad-hoc sieťach), vieme určiť dôvod chyby. Pri zhlukových chybách sa môžu naraz stratiť niektoré pakety. V našom mechanizme máme pre každý odoslaný paket dve informácie. Jedna hovorí o doručení mobilnému uzlu a druhá (skipack) je kvôli preposielaniu. Ak obdržíme SACK o doručení nejakého novšieho paketu, všetkým skôr odoslaným a nepotvrdeným paketom sa zväčší hodnota skipack o jeden. Keď skipack dosiahne určitú hodnotu (empiricky zistenú ako 3), paket sa prepošle. Tento systém je relatívne rýchly a nedôjde k vypršaniu časovačov. Tie preto signalizujú dlhodobé výpadky siete. Vtedy by bolo veľmi neefektívne preposielanie dát. Preto po vypršaní časovača sa nastaví hodnota congestion window *cwnd* na 1 a posiela sa jediný paket pokiaľ sa nedoručí potvrdenie. Vtedy sa nastaví hodnota *cwnd* na pôvodnú hodnotu a linka sa vráti do pôvodného stavu.

V štandardnom TCP doručenie 3 dupAck vyvolá preposlania, no tu sa o to stará skipack systém, teda dupAck nám len dajú informáciu, že linka

stále funguje.

Podobne ako u predchádzajúceho rozšírenia, aj tu sa preposielaním jedného paketu v kolíznom stave šetrí energia, ktorá je potrebná pre mobilné hosty.

Rozšírenie je klasickým prípadom Split riešení. Zároveň je lokálne, two-way, nepotrebuje žiadnu pomoc od nižších vrstiev. Z predpokladu použitia na GPRS sieťach ho môžeme klasifikovať aj ako Last-Hop rozšírenie. Na radenie do front, posielanie, preposielanie a potvrdzovanie potrebuje čítať hlavičky paketov, teda je to snooping riešenie.

5.2.4 *Delay Injection*

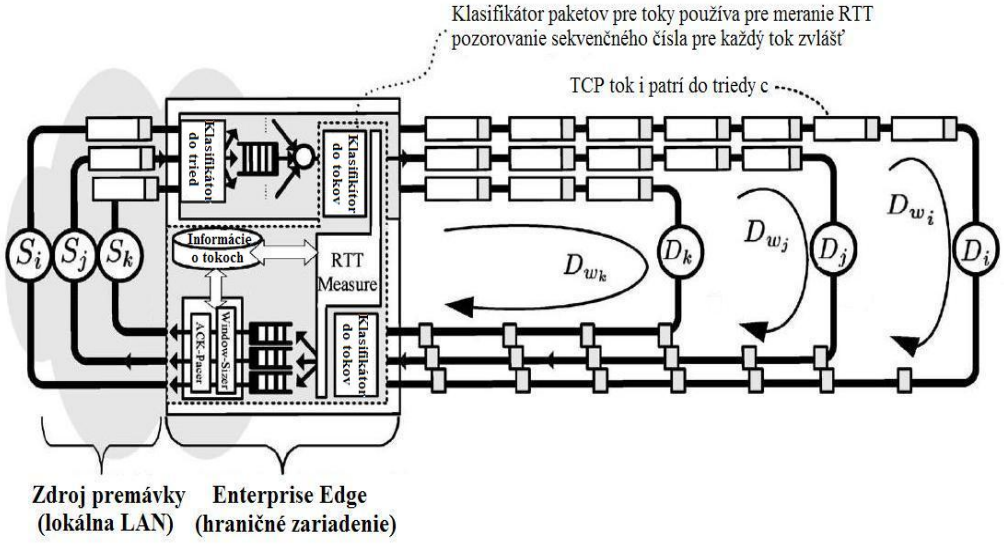
Delay Injection (DI) [16] rozšírenie sa zameriava na tzv. Delay Spikes (špicaté zdržania) a ich vplyv na TCP timeou-ty a priepustnosť. Delay Spikes sú definované ako náhle a výrazné zmeny v RTT medzi TCP odosielateľom a jeho prijímateľom, spôsobené výpadkom pri prechode medzi bunkami alebo nízkou kvalitou linky. Tieto situácie si TCP vyloží ako stratu paketu z dôvodu zahltenia linky a zareaguje neefektívne.

Toto rozšírenie si nevyžaduje zmeny v TCP. Idea je umelo vsunúť dodatočné zdržania v round-trip ceste. Tým sa zvýši čas doručovania - časovač RTT - timeout treshold RTO (inak povedané je to mechanizmus na umelé zvýšenie zmeny RTT bez výrazného zväčšovania priemerného RTT). Zvýšenie časovača RTO môžu ovplyvniť len dva parametre: Smoothed avarage (vyhladený priemer) alebo Mean deviation of RTT (spodná odchýlka RTT). DI mení Mean deviation a to pridávaním konštantných alebo náhodných zdržaní. To docieľuje tým, že pozdrží paket alebo jeho potvrdzovaciu správu niekde na round-trip ceste.

Tvorcovia algoritmu sa zamýšľali nad použitím fixných a náhodných prístupov. Prvý je Fixed Time - Fixed Delay (FTFD), kde pravidelne pridá konštantný čas, Random Time - Fixed Delay (RTFD), ktorý pridá náhodným paketom konštantný čas, a nakoniec Random Time - Random Delay (RTRD), ktorý náhodne pridáva náhodný čas. Po porovnávaní správania sa jednotlivých modelov tvorcovia došli k záveru, že náhodnosť sa nespráva veľmi efektívne, ale pri vhodne nastavených parametroch sa fixný model správa veľmi dobre a očakávane eliminuje falošné timeout-y, čo zvyšuje priepustnsť linky.

Taktiež treba podotknúť, že minimalizovanie timeout-ov pomáha k využitiu linky len do určitej miery. Keby sme pridávali veľké zdržania, vyriešili by sme síce negatívny efekt Delay Spikes ale pri strate by sme museli dlho čakať na vypršanie timeout-ov. Pri predpoklade, že TCP timeout-y väčšinou prichádzajú v zhlukoch (kvôli výpadkom liniek), sa toto rozšírenie javí ako efektívne. Zároveň nepožaduje žiadne dodatočné informácie a zmeny do TCP.

DI môžeme klasifikovať ako transparentné, lokálne - zmenu si vyžadujú



Obrázok 5.5: Princíp fungovania TCR.

len uzly, ktoré pridávajú zdržania - najčastejšie odosielateľ alebo prijímateľ, resp. proxy medzi drôtovou alebo bezdrôtovou sieťou (čo je vhodné pre Last-hop siete napr. mobilné, kedy sa cez jeden uzol pripájajú bezdrôtové hosty do Internetu), podobne od toho závisí, či bude End-to-end alebo Split. DI nepotrebuje pomoc od nižších vrstiev.

5.2.5 *TCP rate control*

TCP rate control (TCR) [18] je rozšírenie založené na policeroch (klasifikátoroch do určitých skupín na základe stanovených pravidiel), s triedami na klasifikovanie tokov a ich tvarovanie - Hierarchické triedy. Je založené na dvoch technikách: window-sizing, ktorý vraví, koľko môžeme poslať a ACK-pacing, ktorý určuje čas poslania. TCR je kombináciou oboch mechanizmov a používa sa na hraniciach drôtových a bezdrôtových sietí. TCR má pre každú triedu časovač, ktorý odrátava poslanie ACK odosieľatelovi. Idea je v tom, že pozdržanie nejakých paketov, vedie k pozdržaniu aj ich odpovede a tým k dlhšiemu RTT. A to už má za následok samotné spomalenie vysielacej rýchlosti u odosielateľa.

Pakety prechádzajú cez rozhranie do/z bezdrôtovej siete, ktoré ich buffe-ruje a preposiela na druhú stranu. Tu sa rieši aj strata paketov na bezdrôtovej strane, ale podstatné pre toto rozšírenie je regulácia tokov a liniek tak, aby sa predišlo zahlteniu. Vtedy vieme, že strata je z dôvodu chybovosti linky. Samotná kontrola a tvarovanie premávky sa vykonáva pomocou window-sizing a ACK-packing.

Obrázok 5.5 nám ukazuje, ako sa správa TCR k premávke vstupujúcej do bezdrôtovej siete. Najskôr sa pakety klasifikujú do jednotlivých tried, potom sa v rámci jednej triedy stará o ne klasifikátor tokov. Ten na základe nameraných RTT vysiela pakety do bezdrôtovej siete. Po prijatí potvrdení sa aktualizujú namerané RTT a vytvorí/zmení sa informácia o danom toku. Zároveň sa uváži použitie window-sizing a ACK-pacing.

Na kontrolu paketov má rozšírenie vyrovnávacie pamäte (buffer), v ktorých po klasifikácii dát do tried si drží pakety, pokiaľ sa nedoručí ich potvrdenie. Po ich doručení z mobilných uzlov sa uváži použitie ACK-pacingu, teda či je potrebné a ako dlho pozdržať potvrdenie o doručení paketu odosielateľovi. Zároveň sa nastaví aj veľkosť prijímacieho okna podľa voľného miesta v danom buffery (window-sizing). Tým sa spraví vlastné predchádzanie zahlteniu. Nastavenia času poslania odpovede sa počítajú z round-trip WAN delay - teda času od rozhrania k uzlu a naspať v mobilnej sieti.

Ak sa nám stratí paket v mobilnej časti siete, vieme si ho nájsť v buffery pre nepotvrdené pakety a preposlať ho.

Problémy: meranie RTT na bezdrôtovej časti siete a menenie hlavičky TCP potvrdzovacej správy má niektoré nevýhody:

1. window-sizing je stále dosť závislý od prenosovej kapacity bezdrôtovej linky a pri viacerých stratách sa prijímacie okno výrazne zníži, čo vedie k neefektívnemu využitiu.

2. Tiny window - pre toky s malým Bandwith Delay Product (BDP) môže window-sizing zmenšiť prijímacie okno tak, že sa do buffera nezmestí viac ako tri nepotvrdené pakety, čo je neefektívne.

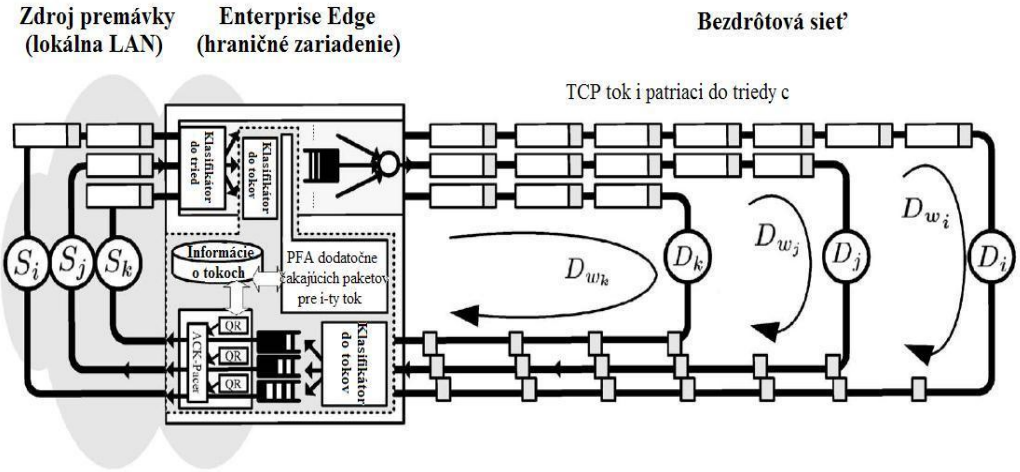
3. Zlý odhad round-trip WAN delay (D) kvôli veľkým možným výkyvom spôsobených bufferovaním. Teda D nie je počítané len od vzdialenosti ale závisí aj od bufferov.

TCR má podobnú ideu pozdržiavania paketov resp. potvrdení ako predchádzajúce DI. Rozdielne sú v tom, že TCR číta hlavičky a teda je snooping rozšírenie, pokiaľ DI nie. TCR pridáva zdržania podľa výpočtu RTT a priepustnosti, DI pridáva náhodne alebo pravidelne určité zdržanie, ktoré nezávisí od okamžitého stavu siete a liniek, a teda je jednoduchšie na implementáciu.

TCP je Split, lokálne, snooping (vyžaduje čítanie a menenie hlavičky), môže byť Intermediate-link alebo aj Last-hop, nevyžaduje si pomoc nižších vrstiev.

5.2.6 *PostAck*

PostAck [18] je alternatívou k predošlému TCR, ktorá rieši niektoré jeho problémy. Líši sa tým, že nemeria WAN delay (D) a nezmenšuje prijímacie



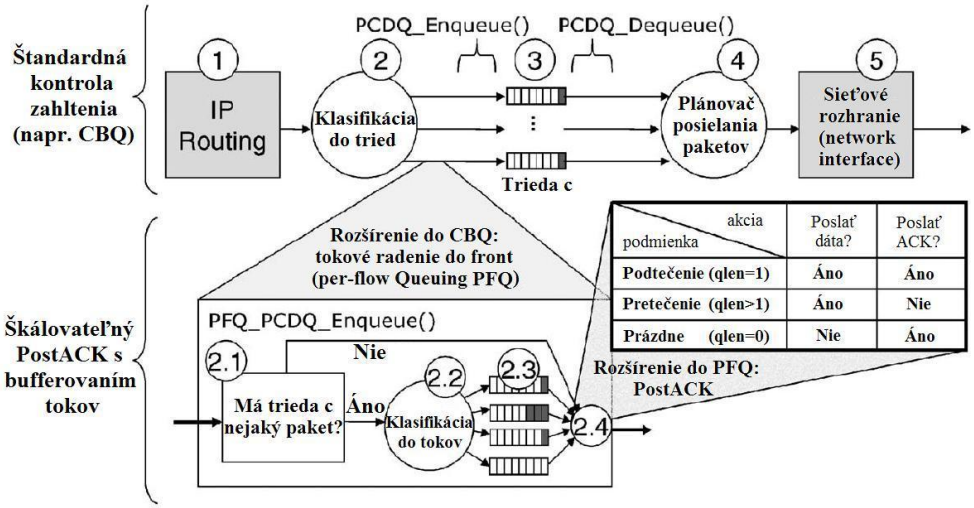
Obrázok 5.6: Princíp fungovania PostACK.

okná. Systém je odlišný od TCR myšlienkou bufferovania. V TCR sa bufferovali pakety - vlastné dáta, ktoré boli zaradené do vyrovnávacích pamätí (buffer) v jednotlivých triedach. PostAck pakety posiela rovno (resp. používa len obmedzenú vyrovnávaciu pamäť), ale bufferuje prijaté potvrdzovacie správy a tie potom pozdržiava. To má rovnaký efekt na kontrolu premávky, pričom to má ďalšie výhody: menšia veľkosť vyrovnávacích pamätí, keďže bufferujeme potvrdzovacie správy zložené len z hlavičiek. Malé bufferovanie implikuje aj malé dátové oneskorenie. Keďže nemeriame WAN delay a nezmenšujeme prijímacie okná, nenastávajú problémy ako u TCR.

Fungovanie PostACK je zobrazené na obrázku 5.6.

V PostAck sa nepoužíva meranie RTT, ale Per-Flow Accounting (PFA), ktorý pracuje nasledovne: Keď prvý buffer PCDQ (per-class data queue) zistí, že niektorý tok presiahol svoju hranicu, namiesto bufferovania paketov pre tok v PCDQ bufferuje pre dodatočné pakety toku ich ACK potvrdzovacie správy v PFAQ (per-flow acnowledge queue). Tým sa zaberie miesto pre ďalšie potvrdzovacie správy toku a tak sa tok spomalí. Inak povedané pre toky, ktoré presiahnu určitú hranicu, pozdržíme ich potvrdenia, čím zmenšíme celkovú rýchlosť.

Nevýhoda je v časovačoch - potrebujeme veľa časovačov, pre každý tok jeden. Preto tvorcovia vymysleli aj verziu bez časovačov, ktorá používa PFQ (per flow queue) v každej triede a používa deficit round robin (DRR). Tým vlastne neuvoľňujeme potvrdzovacie správy na základe časovača, ale podľa PFQ a DRR a to tak, že keď je prázdne PFQ, vypustíme ACK, ak je v ňom voľné miesto, vypustíme paket a ACK, a ak máme preplnený PFQ, nepošleme ACK len paket.



Obrázok 5.7: Implementácia PostACK.

Možnú implementáciu PostACK do existujúceho systému (v tomto prípade implementácia Per-Flow Queuing PFQ do Štandardného mechanizmu predchádzania zahlteniu a následné implementovanie PostACK ako rozšírenie PFQ) znázorňuje obrázok 5.7. Kroky 1, 2, 3, 4 a 5 zobrazujú správanie sa štandardného spojovacieho uzla s mechanizmom správy vyrovnávacích pamätí a klasifikáciou tokov (v tomto prípade CBQ - Class Based Queuing, čiže bufferovanie založené na klasifikáciách do tried). Ako rozšírením tohto mechanizmu je PFQ. Ten pridáva medzi kroky 2 a 3 ďalšie (2.1, 2.2, 2.3 a 2.4). Krok 2.4 je spustený CBQ volaním PCDQ_Dequeue a vyberá ďalší paket na poslanie použitím Deficit Round Robin (DRR). PostACK je zase rozšírenie PFQ a to také, že v kroku 2.4 kontrolujeme stav vybraných paketov s tabuľkou a podľa toho rozhodujeme, či pridáme zdržanie alebo nie. PCDQ_Enqueue a PCDQ_Denqueue sú klasické funkcie na správu front pre triedy, PFQ_PCDQ_Enqueue je funkcia pre vkladanie paketov do front pre jednotlivé toky.

PostAck ako aj TCP je Split, lokálne a snooping rozšírenie.

5.2.7 *Signaling a hiding rozšírenia*

Tieto rozšírenia sa protokolu TCP týkajú len okrajovo, pracujú na sieťovej, linkovej a/alebo fyzickej vrstve a tým riešia problém TCP predchádzanie zahlteniu na bezdrôtových sieťach, a preto ich aspoň okrajovo spomenieme. Podľa našej kategorizácie ich môžeme zaradiť medzi signaling alebo hiding rozšírenia, lebo sa samé starajú o straty, resp. poskytujú informácie o stratách protokolu TCP.

Asi najpopulárnejším štandardom medzi bezdrôtovými sieťami je 802.11.

Ten špecifikuje aj linkovú vrstvu MAC. Tú sme si stručne popísali a ukázali jej problémy v kapitole **3. Problémy štandardného TCP pre špecifické typy sietí** v časti **3.2.3. Podpora nižších vrstiev**. Pre bližšie informácie odporúčame preštudovať si štandard [35]. Spomenieme, že 802.11 MAC vrstva je hiding rozšírenie, teda rieši straty paketov a skrýva ich pred TCP.

V súčasnosti existuje viacero vylepšení štandardného MAC 802.11, napr. ELFN (posielanie explicitnej informácie TCP o strate paketu z dôvodu nespoľahlivej linky) a BEAD (pri strate ACK sa posiela informácia jej odosielateľovi, pri strate viacerých sa používa najväčšie poradové číslo stratených ACK). Obe sú popísané v [36] a radia sa medzi signaling rozšírenia štandardu 802.11.

Nasledujúce rozšírenia spomínané v literatúre [17] sa zameriavajú na široko pásmové bezdrôtové siete ďalšej generácie (2G-BWA). Do úvahy berú zmeny v Mac-linkovej a fyzickej vrstve.

Mac vrstva:

Automatic Retransmission/Fragmentation (ARQ/F) - potvrdzovanie s malým oneskorením zabezpečuje preposielanie stratených častí atomic data units (ADU). Dáta z vyšších vrstiev (PDU) sú fragmentované do ADU, ktoré sú jednoznačne identifikované, bufferované a posielané. U prijímateľa sú ADU tiež bufferované. Keď sa doručia všetky ADU pre dané PDU, ADU sa spoja a prepošlú do vyššej vrstvy. Tu vieme zistiť, kedy nám nedošlo nejaké ADU a vieme ho spätne preposlať. K tomu prijímateľ pravidelne generuje potvrdzovaciu správu s informáciou o doručených ADU. Tie sú potom s väčšou prioritou preposielané pred normálnymi ADU.

Weighted Round Robin Scheduling - tu sa rozhoduje či a ako sa doručí frame (napr. časový slot, modulácia, kódovanie).

Fyzická vrstva:

Adaptive Modulation (AM) - umožňuje prispôsobiť rýchlosť (data rate) ako funkciu stavu linky - využívaním všetkých častí rádiového spektra.

Spatial Diversity (SD) - ide o kombinovanie signálov rôznych antén, čím sa zníži stratovosť a tak sa linka správa skôr ako drôtová.

Multiple-input Multiple-output (MIMO) and Spatial Multiplexing (SM) - multiplexovanie signálu pre viaceré antény - variácia SD.

Frequency Diversity - mechanizmus, ktorý rozloží signál po celej šírke posielacieho pásma, čím zmenšuje efekt tlmenia signálu v niektorých častiach pásma.

Kapitola 6

Záver

Tak ako sa Internet vyvíja, budú do neho stále viac pribúdať nové technológie s rôznym využitím. Aby sa im Internet prispôsobil, bude potrebné meniť a vylepšovať jeho časti - protokoly. TCP ako najpoužívanejší protokol nielen v Internete dokázal, že pracuje zodpovedne a dobre. Ale napriek tomu sú situácie, kedy je potrebné rozmýšlať o jeho vylepšení.

Problematika Kontroly zahltenia u TCP je značne rozsiahla, zaujímavá a dôležitá. Paradoxne pri štúdiu protokolu TCP sa jej kladie veľmi malý dôraz. Prínosom tejto práce je uviesť čitateľa do problematiky Kontroly zahltenia u TCP a podanie obšírnejšieho prehľadu o technikách Kontroly zahltenia nielen v štandardných implementáciách TCP. Podávame aj model algoritmov Kontroly zahltenia (Dynamika, stabilný stav a Reakčná funkcia), ktoré sa v rôznych literatúrach podávajú rôzne a poukazujeme na vzájomný vzťah medzi nimi.

Prínosom práce je taktiež zmapovanie správania sa TCP nad vysokorýchlostnými a bezdrôtovými sieťami, kde TCP vykazuje rôzne problémy s efektívnym využívaním prostriedkov siete. Úlohou je podať čitateľovi obraz o problematike v takýchto sieťach a poukázať na problémy TCP a ich riešenie. V práci ďalej kategorizujeme jednotlivé prístupy k riešeniu problémov nad takýmito sieťami a popisujeme niektoré vybrané riešenia.

V súčasnosti existuje veľa rôznych rozšírení pre jednotlivé typy sietí, preto sme do práce vybrali len také, čo sme považovali za zaujímavé.

Ako ukázala história počítačových sietí, siete ako Internet a ich časti sa budú aj naďalej meniť. Preto je potrebné ďalšie mapovanie problémov a ich riešení v nových situáciách. V práci sa venujeme problémom nad vysokorýchlostnými a bezdrôtovými sieťami, problém heterogénneho spojenia týchto sietí medzi sebou a s klasickými sieťami však spomíname len okrajovo. Pokračovaním práce by mohla byť štúdia takéhoto heterogénneho spojenia.

V práci sa zaoberáme rozšíreniami hlavne z pohľadu efektívneho využí-

vania siete. Mnohé rozšírenia si však kladú aj iné ciele (napríklad férovosť z pohľadu RTT, oscilácia tokov) a používajú rôzne metriky na ich meranie. Ich zmapovanie by mohlo byť náplňou ďalších prác.

Rozšírenia využívajúce explicitnú informáciu na riešenie problémov vysokorýchlostných sietí sme si spomínali len okrajovo. Náplňou ďalších prác by mohol byť podrobnejší prehľad techník využívajúcich explicitnú informáciu zo siete.

V práci spomíname a popisujeme podporu nižších vrstiev (fyzická, linková, sieťová) pre riešenie niektorých problémov bezdrôtových sietí. Bolo by zaujímavé sa na túto problematiku pozrieť podrobnejšie. Zároveň by bolo zaujímavé preskúmanie podpory nižších vrstiev pre vysokorýchlostné siete.

Zaujímavým rozšírením práce by mohla byť štúdia zaručenia diferencovaných služieb nad špeciálnymi sieťami. To sa týka zaručenia určitých kvalít služieb (Quality of services - QoS) zo strany štandardných a rozšírených TCP.

Literatúra

[1] Charles M. Kozierok: „The TCP/IP guide, TCP/IP Transmission control protokol (TCP)“, URL http://www.tcpipguide.com, September 2005.

[2] Stevens, W. Richard: „TCP/IP Illustrated, Volume 1 The Protocols“.

[3] Wright, Gary R.: Stevens, W. Richard: „TCP/IP Illustrated, Volume 2 The Implementation“.

[4] Martin Moravek: „Ovplyvňovanie prenosových charakteristík TCP/IP komunikácie“, Diplomová práca, FMFI UK, 2005.

[5] Hamilton Institute: „TCP Congestion Control for Next Generation Networks“, URL http://www.hamilton.ie/net/tcp.htm.

[6] Chunlei Liu, Raj Jain: „Approaches of Wireless TCP Enhancement and A New Proposal Based on Congestion Coherence“, *System Sciences*, 2003. *Proceedings of the 36th Annual Hawaii International Conference*, Január 2003, strany 6-9.

[7] Rajiv Chakravorty, Sachin Katti, John Crowcroft, Ian Pratt: „Flow aggregation for enhanced TCP over WIde-Area wireless“, University of Cambridge Computer Laboratory, 2003. *IEEE INFOCOM*, 2003.

[8] Sally Floyd, Van Jacobson: „Link-sharing and Resource management models for packet networks!, *IEEE/ACM Transaction on Networking*, Vol. 3 No. 4, August 1995.

[9] Sally Floyd, Kevin Fall: „Promoting the End-to-End congestion control in the Internet“, *IEEE/ACM Transactions on networking*, 3. Máj 1999.

[10] Ratul Mahajan, Sally Floyd, David Wetherall: „Controlling High-bandwidth flows at the Congested Router“, *Proceedings of the Ninth International Conference on Network Protocols*, 2001, strana 192.

[11] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, Lidong Zhou: „A Multi-Radio Unification Protocol for IEEE 802.11 Wireless

Networks“, *Proceedings of the First International Conference on Broadband Networks*, 2004, strany 344-354.

[12] Imad Aad, Qiang Ni, Chadi Barakat, Thierry Turletti: „Enhancing IEEE 802.11MAC in congested environments“, *Applications and Services in Wireless Networks*, 2004. *ASWN* 2004. *2004 4th Workshop*, 2004, strany 82-91.

[13] I. Psarras, L. Mamatas, V.Tsaoussidis: „Shaping TCP Traffic in Heterogeneous Networks“, Dept. of Electrical and Computer Engineering Demokritos University of Thrace, Greece, Júl 2004.

[14] V. Tsaoussidis, H. Badr: „TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains“, *The 8th IEEE Conference on Network Protocols*, *ICNP* 2000, Osaka, Japan, November 2000, strany 12-21.

[15] Rajiv Chakravorty, Sachin Katti, Jon Crowcroft Ian Pratt: „Flow aggregation for enhanced TCP over wide-ares wireless“, University of Cambridge Computer LAboratory, JJ Thomson Avenue, Cambridge CB3 0FD, U.K.

[16] Thierry E. Klein, Kin K. Leung, Richard Parkinson, Louis G. Samuel: „Avoiding Spurious TCP Timeouts in Wireless Networks by Delay Injection“, *Global Telecommunications Conference*, IEEE Volume 5, Issue 29. November - 3. December 2004, strany 2754s-2759.

[17] Ivana Stojanovic, Manish Airy, David Gesbert, Huzur Saran: „Performance of TCP/IP Over Next Generation Broadband Wireless Access Networks“, *Proc. of the International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Aalborg, Denmark, 2001. URL http://citeseer.ist.psu.edu/stojanovic01performance.html.

[18] Huan-Yun Wei, Shih-Chiang Tsao, Ying-Dar Lin: „Assessing and Improving TCP Rate Shaping over Edge Gateways“, *IEEE transactions on computers*, Vol. 53, No. 3, Marec 2004, strany 259-275.

[19] Huan-Yun Wei, Shih-Chiang Tsao, Ying-Dar Lin: „Shaping TCP traffic in ATM networks“, *IEEE transactions on computers*, Vol. 53, No. 3, Marec 2004.

[20] Sally Floyd, Sylvia Ratnasamy, Scott Shenker: „Modifying TCP's Congestion Control for High Speeds“, 5. Máj 2002, URL http://www.icir.org/floyd/notes.html, http://citeseer.ist.psu.edu/floyd02modifying.html.

[21] Tom Kelly: „Scalable TCP: Improving Performance in Highspeed Wide Area Networks“, *Computer Communication Review* 32(2), Apríl 2003.

[22] Lawrence S. Brakmo, Larry L. Peterson: „TCP Vegas: End to End Congestion Avoidance on a Global Internet“, *IEEE Journal on selected areas in communications*, Vol. 13, No. 8, Október 1995, strany 1465-1480.

[23] Cheng Jin, David X. Wei, Steven H. Low: „FAST TCP: motivation, architecture, algorithms, performance“, Caltech CS Report CaltechCSTR:2003:010, December 17, 2003, An Abridged version appears in the *Proceedings of IEEE Infocom*, Hong Kong, Marec 2004.

[24] Sally Floyd: „RFC 3649: HighSpeed TCP for Large Congestion Windows“, RFC 3649, Experimental, December 2003

[25] Sally Floyd: „RFC 3742: Limited Slow-Start for TCP with Large Congestion Windows“, RFC 3742, Experimental, Marec 2004

[26] Lisong Xu, Khaled Harfoush, Injong Rhee: „Binary Increase Congestion Control for Fast, Long Distance Networks“, *INFOCOM* 2004, *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 4, 7-11 Marec 2004, strany 2514-2524.

[27] Injong Rhee, Lisong Xu: „CUBIC: A New TCP-Friendly High-Speed TCP Variant“, 2005.

[28] J. Padhye, V. Firoiu, D. Towsley, J. Kurose: „Modeling TCP Reno Performance: A Simple Model and its Empirical Validation“, *IEEE/ACM Transactions on Networking*, Vol. 8, No. 2, Apríl 2000, strany 133-145.

[29] D. Leith, R. Shorten: „H-TCP: TCP for high-speed and long-distance networks“, *Proc. PFLDnet*, Argonne, 2004.

[30] S. Floyd, M. Handley, J. Padhye: „A comparison of equation-based and aimd congestion control“, Február 2000. URL http://www.aciri.org/tfrc/.

[31] K. K. Ramakrishnan, S. Floyd: „Proposal to add explicit congestion notification (ECN) to IP“, RFC 2481, Január 1999.

[32] Dina Katabi, Mark Handley, Chalrie Rohrs: „Congestion Control for High Bandwidth-Delay Product Networks“, *The proceedings on ACM Sigcomm*, 2002, strany 89-102.

[33] V. Jacobson, R. Braden, D. Borman: „RCF 1323: TCP Extensions for High Performance“, RCF 1323, Máj 1992.

[34] Feyza Keceli, Inanc Inan, Ender Ayanoglu: „TCP ACK Congestion Control and Filtering for Fairness Provision in the Uplink of IEEE 802.11 Infrastructure Basic Service Set“, RCF 1323, *Communications, 2007. ICC '07, IEEE International Conference*, 24-28 Jún 2007.

[35] the IEEE 802.11 working group: „Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications“, *ANSI/IEEE Std 802.11*, 1999.

[36] Xin Yu: „Improving TCP performance over Mobile Ad Hoc Networks by Exploiting Cross-Layer Information Awareness“, *MobiCom'04*, 26. September - 1. Október 2004, Philadelphia, Pennsylvania, USA.

[37] Xiang Chen, Hongqiang Yhai, Jianfeng Wang, Yuguang Fang: „TCP Performance over Mobile Ad Hoc Networks“, Wireless Network Laboratory (WINET), Departament of Electrical and Computer Engineering, University of Florida, USA.

[38] S. H. Low, F. Paganini, J. Wang, J. C. Doyle: „Linear stability of TCP/RED and scalable control“, *Computer Networks Journal*, 43(5):633 647, 2003. URL http://netlab.caltech.edu.

[39] Frank P. Kelly: „Mathematical modelling of the Internet“, In B. Engquist and W. Schmid, editors, *Mathematics Unlimited* - 2001 and Beyond, strany 685-702. Springer-Verlag, Berlin, 2001.