PROIECT RC

Berevoiana Adelina

Epure Adriana

Țintă Mihăiță

CUPRINS

Controlul congestiei 3

1.Algoritmi pentru controlul congestiei
2.Principii generale ale controlului congestiei
3.Politici pentru prevenirea congestiei
4.Formarea traficului
5.Algoritmul găleţii găurite
6.Algoritmul găleţii cu jeton
7.Specificarea fluxului
8. Controlul congestiei în subreţelele bazate pe circuite virtuale
9.Concluzii

RED 18

1.Calculul dimensiunii cozii medii
2. Calculul probabilitatii de marcare a pachetelor
3. Caracteristici ale algoritmului RED

Controlul Congestiei Distribuite 28

1.Problema
2.Solutie
3.Introducere teoretica
A.Controlul Congestiei dependent de intarzieri
B. Controlul congestiei intarzierilor independente
C. Controlul Kelly classic
4.Exemplu de stabilitate intarziata
5.Implementare
6.Rolul router-ului

Bibliografie 37

Repartizarea documentatiei 39

I. Controlul Congestiei

Această lucrare are ca scop proiectarea unui sistem de control al congestiei care se potrivește cu capacitatea rețelei, oferind un nivel înalt de utilizare, stabilitate dinamica și corectitudine în rândul consumatorilor. Accentul este pus pe dezvoltarea legăturilor de control descentralizate de la capătul sistemelor si ruterelor la nivelul care poate satisface astfel de proprietăți în rețele arbitrare şi, ulterior, poate aproxima aceste caracteristici prin intermediul implementării practice la nivel de pachete.[1]

Legile celor două familii de control sunt dezvoltate. Prima este capabila sa realizeze primele trei obiective pentru rețele arbitrare și întârzieri, dar este forțată să constrângă politica de alocare de resurse. Vom dezvolta ulterior o lege *primal – dual* care depășește limitările și alocă resurse pentru a se potrivi cu preferințele stării lor de echilibru la o scară de timp mai lentă.[2]

Mecanismele de control ale congestiei în internet constau în algoritmi ai ferestrei de congestie TCP [16] , care rulează la sfârșitul sistemelor și algoritmi de management *AQM* la routere, cautând să se obțină o performanță înaltă de utilizare, întârzieri mici şi un anumit grad de corectitudine în rândul utilizatorilor. Aceste implementări sunt rezultatul unui ciclu evolutiv care include simulări si experimentări la scală mică și dezvoltare; dat fiind că acest proces a apărut la momentul unei creşteri explozive a rețelei, perfomanțele obținute de aceste sisteme trebuie considerate un mare succes. [4]

Cu toate acestea, există motive de întrebare dacă această cale evolutivă poate atinge limitele sale : deficiențele protocolului curent în medii wireless; dificultăți în furnizarea de servicii de calitate de garanții (întârziere, alocarea resurselor). [12], [19]

Avem obiectivul de a găsi un protocol care poate fi implementat într-un mod descentralizat de către surse şi routere, şi controlează sistemul de la un punct de echilibru stabil, care îndeplineşte

unele cerinţe de bază: utilizare ridicata a resurselor de reţea, cozi mici, şi un grad de control bun asupra alocării resurselor. [7]

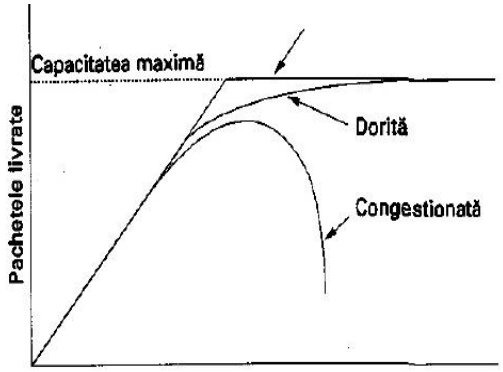
Toate acestea sunt necesare pentru a fi scalabile, de exemplu, pentru o arbitrare de reţea, eventual cu o capacitate mare şi întârziere. Acest fapt, şi structura de informaţii descentralizate, în mod semnificativ conduc spre stabilirea unei legi de control. În secţiunea III vom prezenta o primă soluţie de candidat, care este capabil să obţină primele douăobiectivele de echilibru şi de stabilitate, într-un mod complet scalabil, dar care constrânge politica de alocare a resurselor. În Secţiunea IV am tendinţa de a include teoria dinamica la surse de TCP, conservarea caracteristicilor de mai devreme, la timp-scala rapid (frecvenţe înalte), dar permitand surselor sa se potriveasca cu starea lor de echilibru (frecvenţă redusă) ; limitarea numai la scalabilitatea este că o conexiune pe tur-retur, se presupune a fi cunoscuta.

1. Algoritmi pentru controlul congestiei

Un potențial dezavantaj al algoritmului este acela că se comportă deficitar în cazul rețelelor extinse. Să presupunem că o rețea are n grupuri, fiecare cu un număr mediu de m membri. Pentru fiecare grup, trebuie memorați m arbori de acoperire retezați, deci un total de mn arbori. Dacă există grupuri mari, este necesară o cantitate tot mai mare de memorie pentru a memora acești arbori. [1]

O alternativă de proiectare folosește arbori de bază (core-base trees). Aici se calculează un singur arbore de acoperire pentru fiecare grup, având rădăcina (core – inima, nucleu) lângă mijocul grupului. Pentru a trimite un mesaj multicast, un ruter trimite mesajul către rădăcină, care apoi îl trimite de-a lungul arborelui de acoperire. Deși acest arbore nu va fi optim pentru toate sursele, reducerea costului memorării de la m arbori la unul singur este o realizare majoră.[1]

Atunci când foarte multe pachete sunt prezente într-o subrețea (sau parte a unei subrețele), performanțele se degradează. Situația care apare se numește congestie. Când numărul de pachete emise în subrețea de calculatoare gazdă nu depășește capacitatea de transport, ele sunt livrate integral (cu excepția celor care sunt afectate de erori de transmisie), iar numărul celor livrate este proporțional cu numărul celor emise. Totuși, atunci când traficul crește prea mult, ruterele încep să nu mai facă față și să piardă pachete. Aceasta tinde să înrăutățească lucrurile. La un trafic foarte intens performanțele se deteriorează complet și aproape nici un pachet nu mai este livrat.



Dacă traficul este prea intens, apare congestia și performanțele se degradează puternic.

Congestia poate fi produsă de mai mulți factori. Dacă dintr-o dată încep să sosească șiruri de pachete pe trei sau patru linii de intrare și toate acestea necesită aceeași linie de ieșire, atunci se va forma

o coadă. Dacă nu există suficientă memorie pentru a le păstra pe toate, unele se vor pierde. Adăugarea de memorie poate fi folositoare până la un punct, Nagle (1987) descoperind că dacă ruterele ar avea o cantitate infinită de memorie, congestia s-ar înrăutăți în loc să se amelioreze, deoarece în timpul petrecut de pachete pentru a atinge la începutul cozii ele au fost deja considerate pierdute (timeout repetat) și s-au trimis mai multe duplicate. Toate aceste pachete vor fi cu greu trimise către următorul ruter, crescând încărcarea de-a lungul căii către destinație.

Și procesoarele lente pot cauza congestia. Dacă unitatea centrală (CPU) a ruterului este lentă în execuția funcțiilor sale (introducerea în cozi, actualizarea tabelelor etc), cozile pot crește, chiar dacă linia de comunicație nu este folosită la capacitate. Similar și liniile cu lățime de bandă scăzută pot provoca congestia. Schimbarea liniilor cu unele mai performante și păstrarea aceluiași procesor sau vice-versa de obicei ajută puțin, însă de cele mai multe ori doar deplasează punctul critic. De asemenea, îmbunătățirea parțială și nu totală a sistemului cel mai adesea mută punctul critic în altă parte. Adevărata problemă este de multe ori o incompatibilitate între părți ale sistemului. Ea va persista pâna ce toate componentele sunt în echilibru.[1]

Congestia tinde să se auto-alimenteze și să devină tot mai rea. Dacă un ruter nu mai are zone tampon libere, el trebuie să ignore mesajele noi pe care le recepționează. Când un pachet este distrus, ruterul care l-a trimis (un vecin) poate iniția retransmiterea sa pe motiv de timeout de mai multe ori. Deoarece nu poate distruge pachetul pâna ce nu a fost confirmat, congestia de la receptor îl va forta pe emițător să nu elibereze o zonă tampon pe care în mod normal ar fi eliberat-o în felul acesta congestia se accentuează, ca în cazul mașinilor care se apropie de un punct de taxare de pe autostradă.

Este important să subliniem diferența dintre controlul congestiei și controlul fluxului, deoarece relația este subtilă. Controlul congestiei trebuie să asigure că subrețeaua este capabilă să transporte întreg traficul implicat. Este o problemă globală, implicând comportamentul tuturor calculatoarelor gazdă, ale tuturor ruterelor, prelucrarea de tip *memorează și trimite* (store-and forward) din rutere și toți ceilalți factori care tind să diminueze capacitatea de transport a subrețelei.

Controlul fluxului, prin constrast, se referă la traficul capăt la capăt între un expeditor și un destinatar. Rolul său este de împiedica un expeditor rapid să trimită date continuu, la o viteză mai mare decât cea cu care destinatarul poate consuma datele. Controlul fluxului implică aproape întotdeauna existența unui feed-back de la receptor către emițător, pentru a spune emițătorului cum se desfășoară la celălalt capăt.

Pentru a vedea diferența dintre aceste două concepte, să considerăm o rețea cu fibre optice cu o capacitate de 1000 gigabiți/sec pe care un supercalculator încearcă să transfere un fișier către un calculator personal la 1 Gbps. Deși nu există nici o congestie (rețeaua nu are nici un fel de probleme), controlul fluxului este necesar pentru a forța supercalculatorul să se oprească des, pentru a perminte calculatorului personal să și respire.

La cealaltă extremă, să considerăm o rețea de tip *memorează și trimite* (store –and – forward), cu linii de 1 Mbps și 1000 de calculatoare mari, din care jumătate încearcă să transfere fișiere la 100 kbps către cealaltă jumătate. Aici problema nu apare datorită unui emițător rapid care surclasează un receptor lent, ci pentru că traficul cerut depășește posibilitățile rețelei.

Motivul pentru care controlul congestiei şi controlul fluxului sunt adesea confundate este acela că unii algoritmi pentru controlul congestiei funcționează trimițând mesaje înapoi către diferitele surse, spunându-le să încetinească atunci când rețeaua are probleme. Astfel, un calculator gazdă poate primi un mesaj de încetinire fie din cauzî că receptorul nu suportă încărcarea, fie pentru că rețeaua este depășită. Vom reveni asupra acestui punct mai târziu.

Vom începe studiul algoritmilor pentru controlul congestiei prin studiul unui model general al congestiei. Apoi ne vom ocupa de principalele măsuri pentru prevenirea sa. După aceea, vom vedea o serie de algoritmi dinamici pentru tratarea congestiei odată ce a apărut.

2. Principii generale ale controlului congestiei

Multe din problemele care apar în sistemele complexe, cum ar fi rețelele de calculatoare, pot fi privite din punctul de vedere al unei teorii a controlului. Această abordare conduce la împărțirea tuturor soluțiilor în două grupe: în buclă deschisă şi în buclă închisă. Soluțiile în buclă deschisă încearcă să rezolve problema printr-o proiectare atentă, în esență să se asigure că problema nu apare. După ce sistemul este pornit şi funcționează, nu se mai fac niciun fel de corecții.[1]

Instrumentele pentru realizarea controlului în buclă deschisă decid când să se accepte trafic nou, când să se distrugă pachete şi care să fie acestea, realizează planificarea deciziilor în diferite puncte din reţea. Toate acestea au ca numitor comun faptul că iau decizii fără a ţine cont de starea curentă a reţelei.

Prin contrast, soluţiile în buclă închisă se bazează pe conceptul de reacţie inversă (feedback loop). Această abordare are trei părţi, atunci când se foloseşte pentru controlul congestiei:

1. Monitorizează sistemul pentru a detecta când şi unde se produce congestia.

2. Trimite aceste informaţii către locurile unde se pot executa acţiuni.

3. Ajustează funcţionarea sistemului pentru a corecta problema.

în vederea monitorizării subreţelei pentru congestie se pot folosi diverse metrici. Cele mai utilizate sunt procentul din totalul pachetelor care au fost distruse din cauza lipsei spaţiului temporar de memorare, lungimea medie a cozilor de aşteptare, numărul de pachete care sunt retransmise pe motiv de timeout, întârzierea medie a unui pachet, deviaţia standard a întârzierii unui pachet. în toate cazurile, numerele mari indică creşterea congestiei.

Al doilea pas în bucla de reacţie este transferul informaţiei legate de congestie de la punctul în care a fost depistată la punctul în care se poate face ceva. Varianta imediată presupune trimiterea unor pachete de la ruterul care a detectat congestia către sursa sau sursele de trafic, pentru a raporta problema. Evident, aceste pachete suplimentare cresc încărcarea reţelei exact la momentul în care acest lucru era cel mai puţin dorit, subreţeaua fiind congestionată.

Există însă şi alte posibilităţi. De exemplu, poate fi rezervat un bit sau un câmp în fiecare pachet, pentru a fi completat de rutere dacă congestia depăşeşte o anumită valoare de prag.

Când un ruter detectează congestie, el completează câmpurile tuturor pachetelor expediate, pentru a-şi preveni vecinii. O altă abordare este ca ruterele sau calculatoarele gazdă să trimită periodic pachete de probă pentru a întreba explicit despre congestie. Aceste informaţii pot fi apoi folosite pentru a dirija pachetele în zonele cu probleme. Unele staţii de radio îşi trimit elicopterele să zboare deasupra oraşelor pentru a raporta congestiile de pe drumuri, în speranţa că ascultătorii lor îşi vor dirija pachetele (maşinile) astfel, încât să ocolească zonele fierbinţi.[1]

In toate schemele cu feedback se speră că informarea asupra producerii congestiei va determina calculatoarele gazdă să ia măsurile necesare pentru a reduce congestia. Pentru a funcţiona corect, uratele trebuie reglate foarte atent. Dacă de fiecare dată când două pachete sosesc într-o linie, un ruter strigă STOP şi de fiecare dată când rutenii e liber mai mult de 20 usec el strigă PLEACĂ, atunci sistemul va oscila puternic şi nu va converge niciodată. Pe de altă parte, dacă el aşteaptă 30 minute pentru a fi sigur înainte de a spune ceva, mecanismul pentru controlul congestiei reacţionează prea lent pentru a fi de vreun folos real. Pentru a funcţiona corect sunt necesare unele medieri, dar aflarea constantelor de timp cele mai potrivite nu este o treabă tocmai uşoară.

Se cunosc numeroşi algoritmi pentru controlul congestiei. Pentru a oferi o modalitate de organizare a lor, Yang şi Reddy (1995) au dezvoltat o taxonomie pentru algoritmii de control al congestiei. Ei încep prin a diviza algoritmii în cei în buclă închisă şi cei în buclă deschisă, aşa cum s-a precizat mai sus. în continuare împart algoritmii cu buclă deschisă în unii care acţionează asupra sursei şi unii care acţionează asupra destinaţiei. Algoritmii în buclă deschisă sunt de asemenea împărţiţi în două subcategorii, cu feedback implicit şi cu feedback explicit. în algoritmii cu feedback explicit, pachetele sunt trimise înapoi de la punctul unde s-a produs congestia către sursă, pentru a o avertiza. în algoritmii impliciţi, sursa deduce existenţa congestiei din observaţii locale, cum ar fi timpul necesar pentru întoarcerea confirmărilor.

Prezenţa congestiei înseamnă că încărcarea (momentană) a sistemului este mai mare decât resursele gestionate de sistem. Pentru rezolvare vin imediat în minte două soluţii: sporirea resurselor sau reducerea încărcării. De exemplu subreţeaua poate începe să folosească linii telefonice pentru a creşte temporar lăţimea de bandă între anumite puncte. în sisteme ca SMDS (vezi Cap. 1), se poate cere furnizorului o suplimentare temporară a lăţimii de bandă. în sistemele bazate pe sateliţi, creşterea puterii de transmisie asigură de regulă creşterea lăţimii de bandă. Spargerea traficului pe mai multe căi în locul folosirii doar a celei mai bune poate duce efectiv la creşterea lăţimii de bandă. în fine, ruterele suplimentare, folosite de obicei doar ca rezerve pentru copii de siguranţă (backups) (pentru a face sistemul tolerant la defecte), pot fi folosite pentru a asigura o capacitate sporită atunci când apar congestii serioase.[1]

Oricum, uneori nu este posibilă creşterea capacităţii sau aceasta a fost deja crescută la limită. Atunci singura cale de a rezolva congestia este reducerea încărcării. Sunt posibile mai multe metode pentru reducerea încărcării, cum ar fi interzicerea unor servicii către anumiţi utilizatori, degradarea serviciilor pentru o parte sau pentru toţi utilizatorii şi planificarea cererilor utilizatorilor într-o manieră mai previzibilă.

Unele din aceste metode, pe care le vom studia pe scurt, pot fi aplicate cel mai bine circuitelor virtuale. Pentru subreţelele care folosesc intern circuite virtuale aceste metode pot fi utilizate la nivelul reţea. Pentru subretele bazate pe datagrame ele pot fi totuşi folosite uneori pentru conexiuni

la nivelul transport. în acest capitol, ne vom concentra pe folosirea lor în cadrul nivelului reţea. în următorul, vom vedea ce se poate face la nivelul transport pentru a controla congestia.

3. Politici pentru prevenirea congestiei

Să începem studiul nostru asupra metodelor pentru controlul congestiei prin analiza sistemelor cu buclă deschisă. Aceste sisteme sunt proiectate astfel, încât să minimizeze congestia, în loc să o lase să se producă şi apoi să reacţioneze. Ele încearcă să-şi atingă scopul folosind politici corespunzătoare, la diferite niveluri. în Fig. 5-23 sunt prezentate diferite politici pentru nivelul legătură de date, reţea şi transport, care pot influenţa congestia (Jain, 1990).

Să începem cu nivelul legătură de date şi să ne continuăm apoi drumul spre niveluri superioare. Politica de retransmisie stabileşte cât de repede se produce timeout la un emiţător şi ce transmite acesta la producerea timeout-ului. Un emiţător vioi, care produce repede timeout şi retransmite toate pachetele în aşteptare folosind retrimiterea ultimelor n, va produce o încărcare mai mare decât un emiţător calm, care foloseşte retrimiterea selectivă. Strâns legată de acestea este politica de memorare. Dacă receptorii distrug toate pachetele în afara secvenţei, acestea vor trebui retransmise ulterior, introducând o încărcare suplimentară.

Şi politica de confirmare afectează congestia. Dacă fiecare pachet este confirmat imediat, pachetele de confirmare vor genera un trafic suplimentar. Dacă confirmările sunt preluate de traficul de răspuns, se pot produce timeout-uri şi retransmisii suplimentare. O schemă prea strânsă pentru controlul fluxului (fereastră mică) reduce volumul de date şi ajută în lupta cu congestia.

La nivelul reţea, alegerea între folosirea circuitelor virtuale şi datagrame influenţează congestia, deoarece mulţi algoritmi pentru controlul congestiei funcţionează doar pe subretele bazate pe circuite virtuale. Plasarea în cozi de aşteptare a pachetelor şi politicile de servire specifică dacă ruterele au o coadă pentru fiecare linie de intrare, o coadă pentru fiecare linie de ieşire sau ambele.

Mai precizează ordinea în care se prelucrează pachetele (de exemplu round robin sau bazată pe priorităţi). Politica de distrugere a pachetelor este regula care stabileşte pachetele distruse dacă nu mai este spaţiu. O politică bună va ajuta la eliminarea congestiei, pe când una greşită o va accentua. Algoritmul de dirijare poate ajuta la evitarea congestiei prin răspândirea traficului de-a lungul tuturor liniilor; un algoritm neperformant ar putea trimite toate pachetele pe aceeaşi linie, care deja este congestionată. în fine, gestiunea timpului de viaţă asociat pachetelor stabileşte cât de mult poate trăi un pachet înainte de a fi distrus. Dacă acest timp este prea mare, pachetele pierdute vor încurca pentru mult timp activitatea, iar dacă este prea mic, este posibil să se producă timeout înainte de a atinge destinaţia, provocând astfel retransmisii.[1]

La nivelul transport apar aceleaşi probleme ca la nivelul legăturii de date, în plus, determinarea intervalului de timeout este mai dificil de realizat, deoarece timpul de tranzit prin reţea este mai

greu de prezis decât timpul de tranzit pe un fir între două rutere trimise inutil pachete suplimentare. Dacă este prea mare, congestia se va reduce, însă timpul de răspuns la pierderea unui pachet se va mări. [1]

4. Formarea traficului

Una dintre principalele cauze ale congestiei este aceea că traficul este de obicei în rafală.

în cazul în care calculatoarele gazdă ar putea fi făcute să transmită cu o rată uniformă, congestia nu ar mai fi ceva atât de obişnuit. O altă metodă buclă deschisă care ajuta la controlul congestiei este impunerea unei rate previzibile cu care să fie transmise pachetele. Această abordare a gestiunii congestiei este larg răspândită în reţelele ATM şi se numeşte formarea traficului (traffic shaping).

Formarea traficului se ocupă cu uniformizarea ratei medii de transmisie a datelor (atenuarea rafalelor). în contrast, protocoalele cu fereastră glisantă pe care le-am studiat anterior limitează volumul de date în tranzit la un moment dat şi nu rata la care sunt transmise acestea. La momentul stabilirii unui circuit virtual, utilizatorul şi subreţeaua (clientul şi furnizorul) stabilesc un anumit model al traficului (formă) pentru acel circuit. Atât timp cât clientul îşi respectă partea sa de contract şi trimite pachete conform înţelegerii, furnizorul promite livrarea lor în timp util. Formarea traficului reduce congestia şi ajută furnizorul să-şi ţină promisiunea. Astfel de înţelegeri nu sunt foarte importante pentru transferul de fişiere, însă sunt deosebit de importante pentru datele în timp real, cum ar fi conexiunile audio sau video, care nu suportă bine congestia.

Pentru formarea traficului clientul spune furnizorului: „Modelul m eu de transmisie arată cam

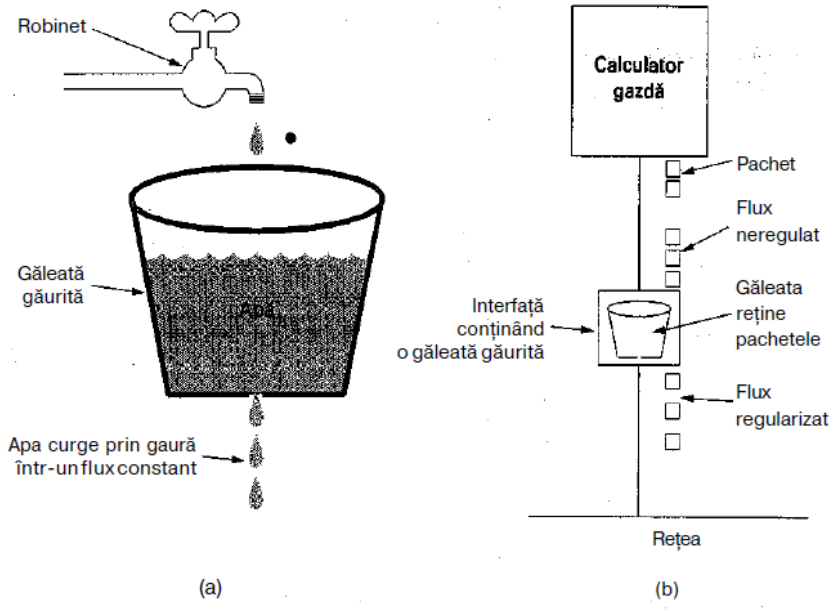
aşa. Poţi să te descurci cu el?" Dacă furnizorul este de acord, problema care apare este cum poate spune furnizorul dacă clientul respectă înţelegerea şi ce să facă dacă nu o respectă. Supravegherea fluxului traficului se numeşte politica traficului (traffic policing.) Stabilirea unei forme a traficului şi urmărirea respectării ei se fac mai uşor în cazul subreţelelor bazate pe circuite virtuale decât în cazul subreţelelor bazate pe datagrame. Cu toate acestea, chiar şi în cazul subreţelelor bazate pe datagrame, aceleaşi idei pot fi aplicate la conexiunile nivelului transport.

5. Algoritmul găleţii găurite

Să ne imaginăm o găleată cu un mic orificiu în fundul său, aşa cum este prezentată în Fig. 5- 24(a). Nu contează cu ce rată curge apa în găleată, fluxul de ieşire va fi la o rată constantă, p, dacă este o cantitate de apă în găleată şi zero dacă găleata e goală. De asemenea, odată ce găleata s-a umplut, orice cantitate suplimentară de apă se va revărsa în afara pereţilor şi va fi pierdută (adică nu se va regăsi în fluxul de ieşire de sub orificiu).

Aceeaşi idee poate fi aplicată şi în cazul pachetelor, aşa cum se arată şi în Fig. 5-24 (b). Conceptual, fiecare calculator gazdă este conectat la reţea printr-o interfaţă conţinând o găleată găurită, în fapt o coadă internă cu capacitate finită. Dacă un pachet soseşte în coadă atunci când aceasta este plină, el este distrus. Cu alte cuvinte, dacă unul sau mai multe procese de pe acel calculator gazdă încearcă trimiterea unui pachet atunci când coada conţine deja numărul maxim de pachete, pachetele noi vor fi distruse în modul cel mai nepoliticos cu putinţă. Acest aranjament poate fi implementat în interfaţa hardware sau poate fi simulat de către sistemul de operare gazdă. A fost propus pentru prima dată de către Turner (1986) şi numit algoritmul găleţii găurite (Ihe leaky bucket algorithm). De fapt nu este altceva decât un sistem de cozi cu un singur server şi cu timp de servire constant.

Fig. 5-24. (a) O găleată găurită umplută cu apă (b) O găleată găurită, cu pachete.

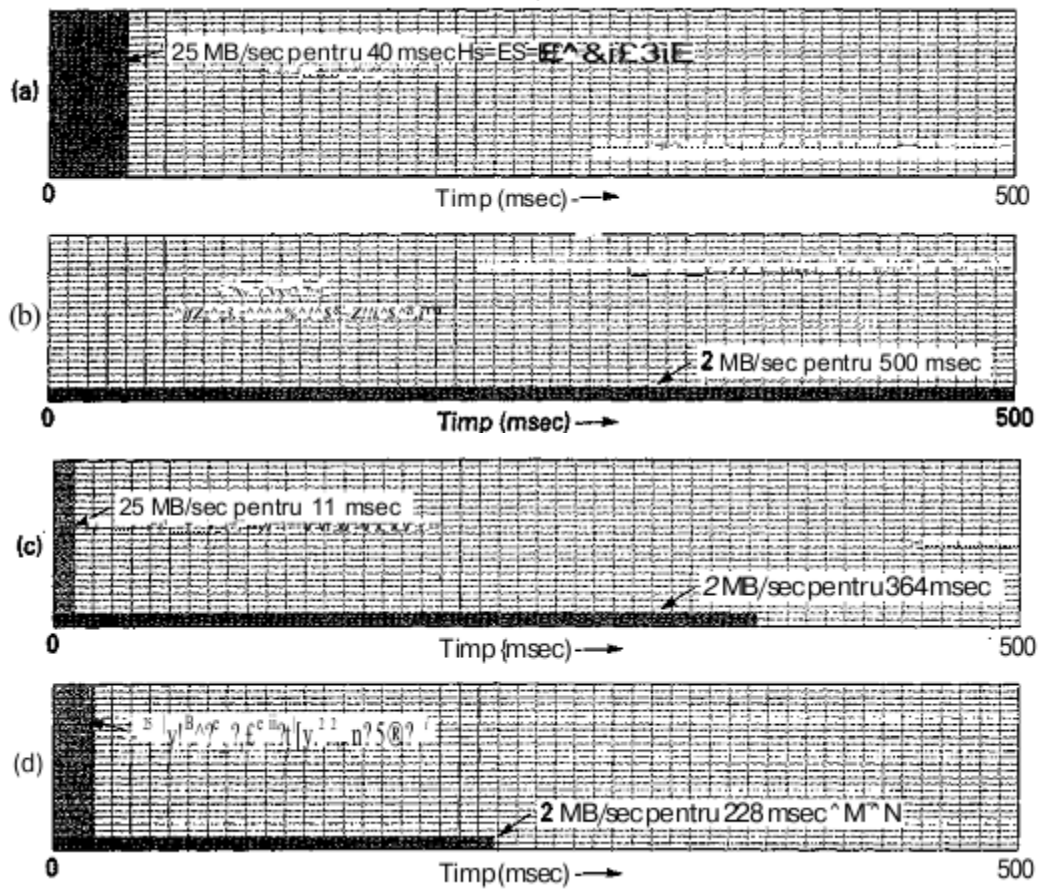


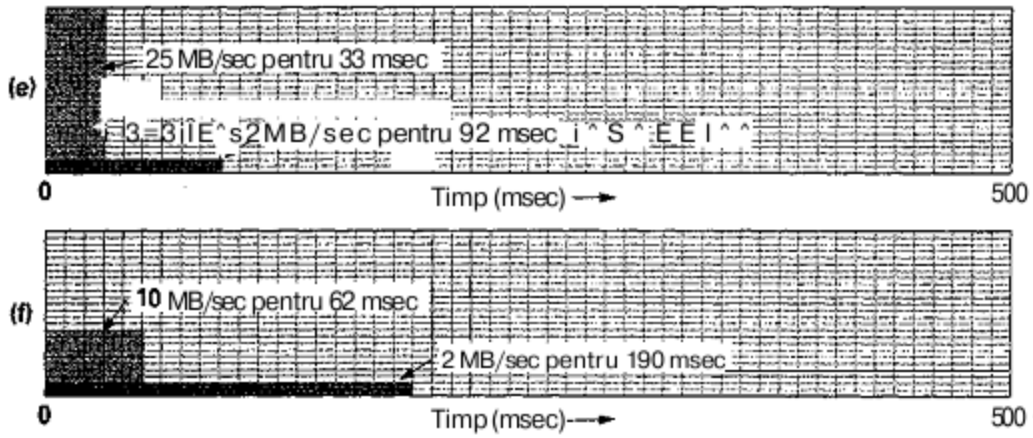
Calculatorul gazdă poate pune pe reţea câte un pachet la fiecare tact al ceasului. Din nou aceasta poate fi implementată în hardware sau poate fi simulată de către sistemul de operare. Acest mecanism transformă un flux neregulat de pachete de la procesele de pe calculatorul gazdă într-un flux uniform de pachete care se depun pe reţea, netezind rafalele şi reducând mult şansele de producere a congestiei.

Dacă pachetele au toate aceeaşi dimensiune (de exemplu celule ATM), algoritmul poate fi folosit exact aşa cum a fost descris. Dacă se folosesc pachete de lungimi variabile, este adesea mai convenabil să se transmită un anumit număr de octeţi la fiecare tact şi nu un singur pachet. Astfel, dacă regula este 1024 biţi la fiecare tact, atunci se pot transmite un pachet de 1024, două de 512 octeţi, sau patru de 256 octeţi ş.a.m.d. Dacă numărul rezidual de octeţi este scăzut, următorul pachet va trebui să aştepte următorul tact.

Implementarea algoritmului iniţial al găleţii este o treabă uşoară. Găleata găurită constă de fapt dintr-o coadă finită. Dacă la sosirea unui pachet este loc în coadă, el este adăugat la sfârşitul cozii, în caz contrar, este distrus. La fiecare tact se trimite un pachet din coadă (bineînţeles dacă aceasta nu este vidă).

Algoritmul găleţii folosind contorizarea octeţilor este implementat aproximativ în aceeaşi manieră. La fiecare tact un contor este iniţializat la *n*. Dacă primul pachet din coadă are mai puţini octeţi decât valoarea curentă a contorului, el este transmis şi contorul este decrementat cu numărul corespunzător de octeţi. Mai pot fi transmise şi alte pachete adiţionale, atât timp cât contorul este suficient de mare. Dacă contorul scade sub lungimea primului mesaj din coadă, atunci transmisia încetează până la următorul tact, când contorul este rescris şi valoarea sa pierdută.





(a) Intrarea pentru o galeata gaurita
(b) Iesirea pentru o galeata gaurita
(c) – (e) iesirea pentru o galeata cu jeton de capacitate 250 KB, 500 KB, 750 KB
(f) Iesirea pentru o galeata cu jeton de capacitate 500 KB care alimenteaza o galeata gaurita de 10 MB/sec.

Ca un exemplu de galeata gaurita, sa ne imaginam un calculator care produce date cu rata de 25 milioane octeti/sec (200 Mbps) si o retea care functioneaza la aceeasi viteza. Cu toate acestea, ruterele pot sa gestioneze datele la aceasta rata doar pentru un timp foarte scurt, pentru interval mai mari ele lucreaza optimi pentru rate care nu depasesc valoarea de 2 milioane octeti/sec. Sa presupunem acum ca datele vin in rafale, 1 milion de octeti, cate o rafala de 40 msec in fiecare secunda.

Pentru a reduce rata medie la 2 MB/sec, puntem folosi o galeata gaurita avand p=2MB/sec si o capacitate, C, de 1 MB. Aceasta inseamna ca rafalele de pana la 1 MB pot fi gestionate fara pierderi de date si ca acele rafale sunt imprastiate de-a lungul a 500 msec, indifferent cat de repede sosesc.

6. Algoritmul găleţii cu jeton

Algoritmul găleţii găurite impune un model rigid al ieşirii, din punct de vedere al ratei medii, indiferent de cum arată traficul. Pentru numeroase aplicaţii este mai convenabil să se permită o creştere a vitezei de ieşire la apariţia unor rafale mari, astfel încât este necesar un algoritm mai flexibil, de preferat unul care nu pierde date.

Un astfel de algoritm este **algoritmul găleţii cu jeton (the token bucket algorithm).** în acest algoritm, găleata găurită păstrează jetoane, generate de un ceas cu rata de un jeton la fiecare $\check{a}T$ sec. în Fig. 5-26(a) vedem o găleată păstrând trei jetoane şi având cinci pachete care aşteaptă să fie transmise. Pentru ca un pachet să fie transmis, el trebuie să captureze şi să distrugă un jeton. în Fig. 5-26(b) vedem

că trei din cele cinci pachete au trecut mai departe, în timp ce celelalte două aşteaptă să fie generate alte două jetoane.

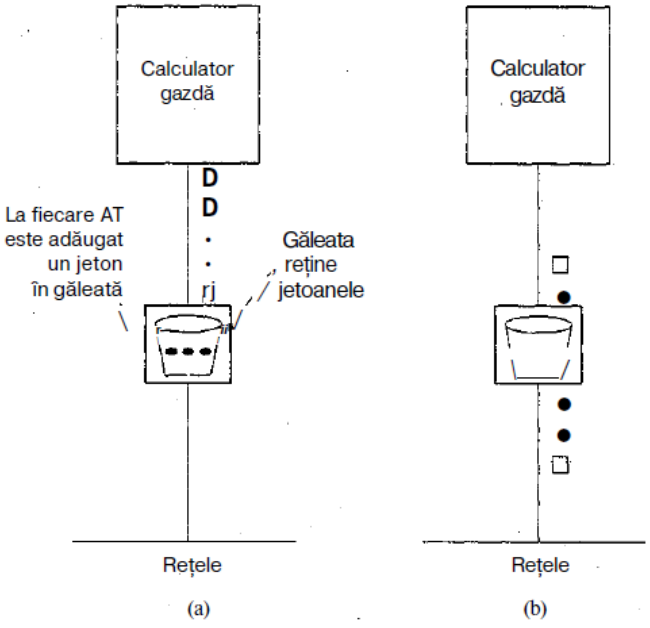


Fig. 5-26. Algoritmul găleţii cu jeton, (a) înainte, (b) După.

Algoritmul găleţii cu jeton asigură o formare diferită a traficului, comparativ cu algoritmul găleţii RE:*!Î'."Algoritmul găleţii găurite nu permite calculatoarelor gazdă inactive să acumuleze permisiuni de transmitere pentru a trimite o rafală mai mare ulterior. Algoritmul găleţii cu jeton permite această acumulare, mergând până la dimensiunea maximă a găleţii, *n.* Această proprietate permite ca rafale de până la *n* pachete să fie trimise simultan, permiţând apariţia unor rafale la ieşire, asigurând un răspuns mai rapid la apariţia bruscă a unor rafale la intrare.

O altă diferenţă între cei doi algoritmi este aceea că algoritmul găleţii cu jeton aruncă jetoanele la umplerea găleţii, dar niciodată nu distruge pachete. Prin contrast, algoritmul găleţii găurite distruge pachete la umplerea găleţii.

Şi aici este posibilă o variantă, în care fiecare jeton reprezintă dreptul de a trimite nu un pachet ci *k* biţi. Un pachet va putea fi trimis doar dacă se deţin suficiente jetoane pentru a-i acoperi lungimea în biţi. Pentru variantele viitoare se prevede folosirea jetoanelor fracţionare. Algoritmul găleţii găurite şi algoritmul găleţii cu jeton pot fi folosite şi pentru a uniformiza traficul între rutere, la fel de bine cum pot fi folosite pentru a regla informaţia de ieşire a unui calculator gazdă, ca în exemplul prezentat. Oricum, o diferenţă evidentă este aceea că algoritmul găleţii cu jeton folosit pentru a regla ieşirea unui calculator gazdă îl poate opri pe acesta să trimită date atunci când apare vreo restrictive.

A spune unui ruter să oprească transmisiile în timp ce datele de intrare continuă să sosească, poate duce la pierderea de date. Implementarea de bază a algoritmului găleţii cu jeton se face printr-o variabilă care numără jetoane. Acest contor creşte cu 1 la fiecare *AT* şi scade cu 1 ori de câte ori este un

pachet. Dacă acest contor atinge valoarea zero, nici un pachet nu mai poate fi trimis. în varianta la nivel de biţi, contorul este incrementat cu *k* biţi la fiecare *AT* şi decrementat cu lungimea pachetului trimis.

Caracteristica esenţială a algoritmului găleţii cu jeton este că permite rafalele, dar până la o lungime maximă controlată. Priviţi spre exemplu Fig. 5-25(c). Aici avem o găleată cu jeton, de capacitate 250 KB. Jetoanele sosesc cu o rată care asigură o ieşire de 2 MB/sec. Presupunând că găleata este plină când apare o rafală de 1 MB, ea poate asigura scurgerea la întreaga capacitate de 25 MB/sec pentru circa 11 milisecunde. Apoi trebuie să revină la 2 MB/sec până ce întreaga rafală este trimisă.

Calculul lungimii rafalei de viteză maximă este uşor înşelător. Nu este doar 1 MB împărţit la 25 MB/sec deoarece, în timp ce rafala este prelucrată, apar alte jetoane. Dacă notăm *S* cu lungimea rafalei în secunde, cu C capacitatea găleţii în octeţi, cu p rata de sosire a jetoanelor în octeţi/secundă, cu *M* rata maximă de ieşire în octeţi/secundă, vom vedea că o rafală de ieşire conţine cel mult C + pS octeţi. De asemenea, mai ştim că numărul de octeţi într-o rafală de viteză maximă de *S* secunde este MS. De aici avem $\mathbf{C + pS = MS}$

Rezolvând această ecuaţie obţinem $S = C / (M - p)$. Pentru parametrii consideraţi $C = 250$ KB, $M = 25$ MB/sec şi $p = 2$ MB/sec vom obţine o durată a rafalei de circa 11 msec. Fig. 5-25(d) şi Fig. 5-25(e) prezintă găleţile cu jeton de capacităţi 500 KB şi respectiv 750 KB.

O problemă potenţială a algoritmului găleţii cu jeton este aceea că şi el permite apariţia unor rafale mari, chiar dacă durata maximă a unei rafale poate fi reglată prin selectarea atentă a lui p şi *M*.

De multe ori este nevoie să se reducă valoarea de vârf, dar fără a se reveni la valorile scăzute permise de algoritmul original, al găleţii găurite.

O altă cale de a obţine un trafic mai uniform este de a pune o găleată găurită după cea cu jeton. Rata găleţii găurite va trebui să fie mai mare decât parametrul p al găleţii cu jeton, însă mai mică decât rata maximi a reţelei, Fig. 5-25(f) ilustrează comportarea unei găleţi cu jeton de 500 KB, urmată de o găleată găurită de 10 MB/sec.

Gestionarea tuturor acestor scheme poate fi puţin mai specială. în esenţă, reţeaua trebuie să simuleze algoritmul şi să se asigure că nu se trimit mai multe pachete sau mai mulţi biţi decât este permis. Pachetele în exces sunt distruse sau pierdute, aşa cum s-a discutat anterior.

7. Specificarea fluxului

Formarea traficului este eficientă atunci când emiţătorul, receptorul şi subreţeaua sunt toţi de acord. Pentru a realiza înţelegerea, este necesar să se specifice modelul traficului într-o manieră precisă. O astfel de înţelegere se numeşte **specificarea fluxului.** Ea constă dintr-o structură de date care descrie atât modelul traficului introdus, cât şi calitatea serviciului dorit de aplicaţie.

O specificare a fluxului poate fi aplicată atât pachetelor trimise pe un circuit virtual, cât şi unei secvenţe de datagrame trimise între o sursă şi o destinaţie (sau chiar cu destinaţii multiple). în această secţiune vom descrie un exemplu de specificare a fluxului propusă de Partridge (1992). Ea este prezentată în Fig. 5-27. Ideea este că înainte de stabilirea unei conexiuni sau înainte de trimiterea unei secvenţe de datagrame, sursa supune specificarea fluxului spre aprobare subreţelei. Subreţeaua poate fie să o accepte sau rejecteze, fie să vină cu o contrapropunere („Nu pot să-ţi ofer întârziere medie de 100 msec; te mulţumeşti cu 150 msec? „). După ce emiţătorul şi subreţeaua au ajuns la o înţelegere, emiţătorul poate interoga receptorul dacă este şi el, de acord.

Caracteristicile intrării	Servicii dorite
Dimensiunea maximă a pachetului (octeţi)	Senzitivitate la pierderi (octeţi)
Rata găleţii cu jeton (octeţi/sec)	Interval de pierderi (nsec)
Dimensiunea găleţii cu jeton (octeţi)	Senzitivitatea la pierderi a rafalei (pachete)
Rata maximă de transfer (octeţi/sec)	Cea mai mică întârziere observată (jjsec)
	Cea mai mare variaţie a întârzierii (\\sec)
	Calitatea garanţiei

Să examinăm acum parametrii exemplului nostru de specificare a fluxului, începând cu specificarea traficului. *Dimensiunea maximă a pachetului* spune cât de mari pot fi pachetele.

Următorii doi parametri presupun implicit că traficul va fi format cu algoritmul găleţii cu jeton, lucrând la nivel de octeţi. Ei precizează câţi octeţi intră în găleata cu jeton într-o secundă şi cât de mare este găleata. Dacă rata este *r* octeţi/sec şi dimensiunea găleţii este *b* octeţi, atunci în orice interval arbitrar At, numărul maxim de octeţi ce pot fi trimişi este *b + rAt.* Aici primul termen reprezintă conţinutul maxim posibil al găleţii la începutul intervalului, iar al doilea reprezintă noile intrări sosite în timpul intervalului. *Rata maximă de transfer* este cea mai mare rată pe care o poate produce calculatorul gazdă, indiferent de condiţii şi specifică implicit cel mai scurt interval de timp în care poate fi golită găleata cu jeton.

A doua coloană specifică ce vrea aplicaţia de la reţea. Primii doi parametri reprezintă respectiv numărătorul şi numitorul unei fracţii care ne dă rata maximă acceptabilă a pierderilor (de exemplu 1 octet pe oră). *Senzitivitatea rafalei la pierderi* stabileşte câte pachete consecutive pierdute pot fi tolerate.

Următorii doi parametri se ocupă de întârzieri. *Cea mai mică întârziere observată* specifică durata de întârziere la care aplicaţia sesizează acest lucru. Pentru un transfer de fişiere ea ar putea fi 1 secundă, dar pentru un flux audio limita ar fi 3 msec. *Cea mai mare variaţie a întârzierii* încearcă să cuantifice faptul că anumite aplicaţii nu sunt sensibile la întârzieri, însă sunt foarte sensibile la **fluctuaţii (jitter),** adică variaţia timpului de tranzit al pachetelor capăt la capăt. Este dublul numărului de microsecunde cu care întârzierea unui pachet poate varia faţă de medie.

Deci, o valoare de 2000 înseamnă că un pachet poate sosi mai devreme sau mai târziu cu cel mult 1 msec. în fine, *Calitatea garantei* precizează dacă aplicaţia are într-adevăr nevoie de aşa ceva. Pe de o parte, caracteristicile de pierdere şi întârziere pot fi scopuri ideale, fără să se întâmple nimic dacă ele nu

sunt realizate. Pe de altă parte, ele ar putea fi aşa de importante, încât dacă nu pot fi atinse aplicaţia se termină. Sunt posibile, evident şi situaţii intermediare.

Deşi am privit specificarea fluxului ca o cerere a aplicaţiei pentru subreţea, ea poate fi de asemenea o valoare returnată care spune ce poate face subreţeaua. Astfel, ea poate fi folosită pentru o negociere mai extinsă a nivelului serviciului.

O problemă inerentă a oricărei aplicaţii referitoare la specificarea fluxului este că aplicaţia ar putea să nu ştie cu adevărat de ce are nevoie. De exemplu, un program aplicaţie rulat la New York ar putea fi mai mult decât fericit cu o întârziere de 200 msec până la Sydney, dar total nefericit cu aceeaşi întârziere până la Boston. Aici "serviciul minim" este în mod clar funcţie de ce se crede că se poate.

8. Controlul congestiei în subreţelele bazate pe circuite virtuale

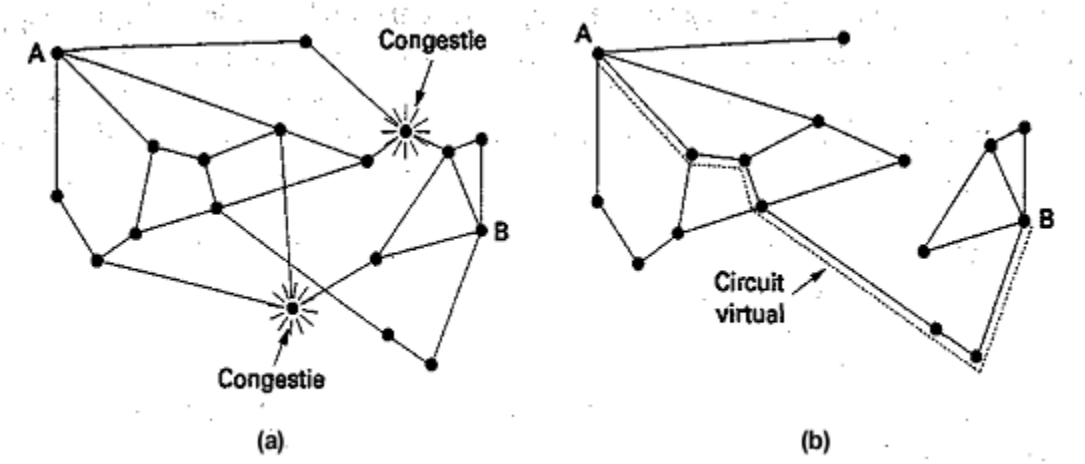
Metodele pentru controlul congestiei descrise anterior sunt în principiu în buclă deschisă: ele încearcă în primul rând, să prevină apariţia congestiei în primul rând, în loc să o trateze după ce a apărut. în această secţiune, vom descrie unele abordări ale controlului dinamic al congestiei în subreţelele bazate pe circuite virtuale. în următoarele două, vom studia tehnici ce pot fi folosite în orice subreţea.

O tehnică larg răspândită pentru a controla congestia care tocmai s-a produs şi de a o împiedica să se agraveze este controlul admisiei. Ideea este simplă: odată ce s-a semnalat apariţia congestiei, nu se mai stabilesc alte circuite virtuale până ce problema nu s-a rezolvat. Astfel, încercarea de a stabili o nouă conexiune la nivel transport eşuează. Lăsând tot mai mulţi utilizatori să stabilească conexiuni, nu ar face decât să agraveze lucrurile. Deşi această abordare este crudă, ea este simplă şi uşor de întreţinut. în sistemul telefonic, dacă o centrală devine supraaglomerată, ea practică controlul admisiei, nemaifurnizând tonul.

O alternativă este de a permite stabilirea de noi circuite virtuale, dar de a dirija cu atenţie aceste noi circuite prin zonele cu probleme. De exemplu, să considerăm subreţeaua din Fig. 5-28(a), în care două rutere sunt congestionate, aşa cum se poate observa.

Să presupunem că un calculator gazdă ataşat ruterului^l doreşte să stabilească o conexiune cu un altul, ataşat rutenilui/î. în mod normal, această conexiune ar trece prin unul dintre cele două rutere congestionate. Pentru a evita situaţia aceasta, putem redesena subreţeaua aşa cum se arata, omiţând ruterele congestionate şi toate liniile asociate. Linia punctată arată o posibilă

cale pentru un circuit virtual, care evită ruterele congestionate.



9. CONCLUZII

O metoda de a evita congestia care poate obține o utilizare importantă, cu o întârziere mică, libertate de oscilare si corectitudine în alocalea benzii a fost un obiectiv major al cercetărilor din domeniul rețelisticii din ultimii ani. O întrebare fundamentala în această problemă este cât de departe se poate merge pentru obținerea obiectivelor în tehnologia existentă end-to-end a internetului. [20]

Rezultatele arată că dacă aspectul de corectitudine este exprimat funcții de utilitate, stabilitatea locala de-a lungul acestor puncte de echilibru putând fi obțiută cu un mechanism minimal de feedback, un semnal scalar care reflect nivelul concestiei agregat al legăturilor traversate pentru fiecare sursă. Mai mult, rezultatele convergentei pot fi realizate indepenedent de topologia rețelei de routing, parametric şi întârziere pentru destinația dorită.

Lucram acum la o întelegere a dinamicii nonliniare , care are un impact semnificant asupra vitezei a controlului echilibrului.[3]

Am demonstrate o versiune practica a protocolului bazat pe ECN marking, care aproximează cu success aceste obiective în retele de capacitate înaltă unde protocoalele curente au limitari. Comparat cu alte soluții de protocol, această versiune reprezintă o schimbare minora în implementările de protocol ale routerelor si ale end-system [19]. Totuși, ar fi de preferat sa upgradăm doar sfarşitul lucrurilor; acest fapt ne-a motivate să considerăm implementarea bazată pe întârziere, similar cu TCP Vegas, care pare să fie capabila sa aduca majoritatea beneficiilor cu nicio participare active a routerelor din retea. Bazata pe

succesul nostrum preliminary in simulari, acum vom obtine exeperimental datele acestor tipuri de protocoale.

O parte din acest efort implica testarea coexistentei unor astfel de protocale cu versiuni cu anumite versiuni alte TCP. [8]

II. Random Early Drop (RED)

În 1993, Sally Floyd şi Van Jacobson au venit cu o lucrare interesantă despre "cum să detectezi congestia în reţele folosind ruterele, cu ajutorul unui mecanism de detecţie aleatoare timpurie".

Algoritmul RED – Random Early Detection, cunoscut şi ca Random Early Discard sau Random Early Drop, este un mecanism de evitare a congestiei, care are în plus câteva metode pentru detectarea congestiei şi pentru alegerea conexiunii care să notifice această congestie. Ruterele RED sunt folosite de obicei cu reţele TCP/IP, fiind proiectate pentru reţele unde un singur pachet marcat sau aruncat este suficient pentru semnalarea congestiei la nivelul transport.

Random Early Detection (RED) reduce congestia în cozi aruncând pachete astfel încât anumite conexiuni TCP pe o perioadă limitată de timp transmit mai puţine pachete pe reţea. În loc să se aştepte până când o coadă se supraîncarcă, cauzând un număr mare de pachete aruncate, RED în mod intenţionat aruncă o parte din pachete înainte ca bufferul să fie plin. Această acţiune este menită să facă transmiţătorii ce generează traficul să reducă volumul de date transmis pe reţea.

Numele "detecţie aleatoare timpurie" descrie funcţionarea algoritmului. RED alege aleator pachete de descărcat după ce a luat decizia de descărcare de pachete. Logica de funcţionare a RED conţine 2 aspecte principale: RED trebuie să detecteze congestia înainte ca aceasta să se intample, cu alte cuvinte, RED trebuie să decidă care sunt condiţiile care vor duce la decizia de aruncare de pachete, iar când se ia această decizie, trebuie să fie definit numărul de pachete de aruncat. În primul rând RED verifică adâncimea cozii echipamentului. Apoi RED calculează adâncimea medie pentru acea interfaţă, apoi decide dacă va apărea fenomenul de congestie. RED foloseşte încărcarea medie pentru că încărcarea reală îşi schimbă valoarea mult mai des. Pentru că RED evită efectele sincronizării, este necesar ca aruncare de pachete să fie egal proporţionată.

Avantajul acestui mecanism de control al congestiei la nivelul ruterului este că funcţionează cu actualele protocoale de transport, şi nu necesită ca toate ruterele din Internet să folosească acelaşi mecanism de control al congestiei. Ruterele RED reprezintă un mecanism simplu de evitare a congestiei, şi poate fi implementat gradual în actualele reţele TCP/IP fără a schimba protocolul de transport.

Scopul ruterelor RED este de a preveni apariţia congestiei prin controlul dimensiunii medii a cozii de pachete.

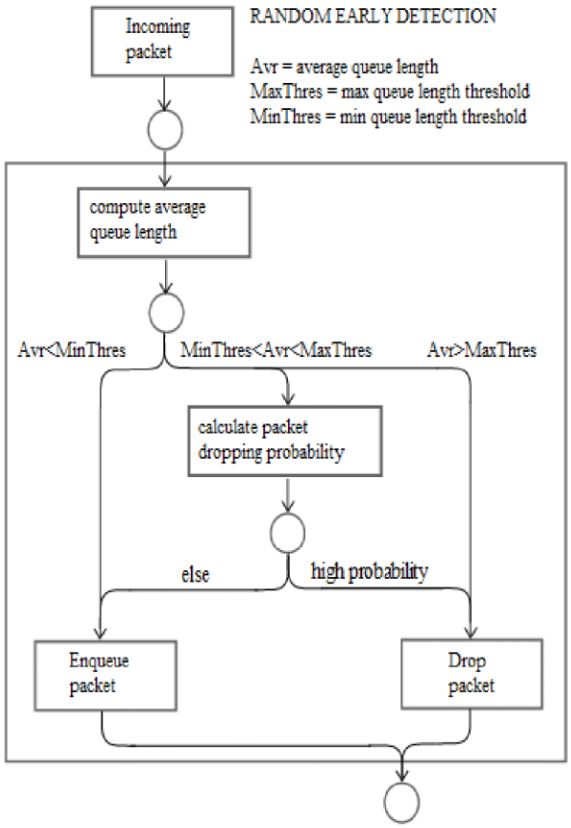
Dimensiunea medie a cozii este calculată de RED folosind un filtru trece-jos cu o medie ponderată mobilă, exponenţială. Dimensiunea medie a cozii este comparată cu 2 praguri, un prag *minim* şi un prag *maxim*. Atunci când dimensiunea medie a cozii este mai mică decat pragul minim, nu este marcat nici un pachet. Atunci când dimensiunea medie a cozii este mai mare decât pragul maxim, fiecare pachet ce soseşte este marcat. Dacă pachetele marcate sunt de fapt aruncate, sau dacă toate nodurile sursă cooperează, acest procedeu poate asigura ca dimensiunea medie a cozii să nu depăşească semnificativ pragul maxim.

Atunci când dimensiunea medie a cozii se situează între minim şi maxim, fiecare pachet ce soseşte este marcat cu probabilitatea p_a, unde p_a este o funcţie de dimensiunea medie a cozii avg. De fiecare dată când un pachet este marcat, probabilitatea ca acel pachet să aparţină unei anumite conexiuni, este proporţională cu cota pe care acea conexiune o deţine din lăţimea de bandă totală ce traversează gateway-ul.

```
Pentru fiecare pachet sosit
     Calculează dimensiunea medie a cozii avg
     Dacă min_th <= avg < max_th
          Calculează probabilitatea p_a
          Marchează pachetul sosit cu probabilitatea p_a
     Altfel dacă max_th <= avg
          Marchează pachetul sosit
```

Figura 8. Algoritmul RED

Prin urmare este alcătuit din 2 algoritmi separați. Algoritmul care calculează dimensiunea medie a cozii determină gradul maxim de *rafale* ce va fi permis în coada gateway-ului. Algoritmul care calculează probabilitatea de marcare a unui pachet, determină cât de des marchează gateway-ul pachete, la un anumit nivel de congestie. Obiectivul este ca gateway-ul să marcheze pachete la intervale de timp egal distanțate, pentru a evita sincronizarea globală, și de a marca pachete suficient de des pentru a putea controla dimensiunea medie a cozii.



Calculul dimensiunii medii a cozii, are în vedere perioada cât timp coada este goală (perioadă *idle*), estimând numărul ***m*** de pachete care *ar fi putut* să fie transmise de către gateway pe durata acesteia. După perioada *idle*, gateway-ul calculează dimensiunea medie a cozii ca și când ***m*** pachete ar fi ajuns pe durata acesteia, într-o coadă goală.

Cum ***avg*** variază între $\boldsymbol{min_{th}}$ și $\boldsymbol{max_{th}}$, probabilitatea de marcare a pachetelor $\boldsymbol{p_b}$ variază liniar între ***0*** și $\boldsymbol{max_p}$:

$$p_b = max_p \frac{(avg - min_{th})}{(max_{th} - min_{th})}$$

Probabilitatea finală de marcare a pachetelor $\boldsymbol{p_a}$ crește încet pe măsură ce numărătoarea crește față de ultimul pachet marcat.

$$p_a = \frac{p_b}{(1 - count * p_b)}$$

În acest mod, se asigură că gateway-ul nu așteaptă prea mult pentru a marca un nou pachet.

Gateway-ul marchează fiecare pachet ce ajunge aici, atunci când dimensiunea medie a cozii ***avg*** depășește $\boldsymbol{max_{th}}$.

O opțiune pentru algoritmul RED este măsurarea cozii în *bytes* și nu în *pachete*. În acest mod, dimensiunea medie a cozii reflectă precis întârzierea medie la gateway. Când este folosită această opțiune, algoritmul este modificat pentru a asigura faptul că probabilitatea cu care este marcat un pachet, este proporțională cu dimensiunea pachetului în bytes.

$$p_b = max_p \frac{(avg - min_{th})}{(max_{th} - min_{th})}$$

$$p_b = p_b \frac{PacketSize}{MaximumPacketSize}$$

$$p_a = \frac{p_b}{(1 - count * p_b)}$$

În acest fel, un pachet FTP are șanse mai mari să fie marcat în comparație cu un pachet TELNET.

Inițializare
- $avg \leftarrow 0$
- $count \leftarrow -1$

Pentru fiecare pachet sosit
- Calculează noua dimensiune medie a cozii ***avg***
  - Dacă coada **nu** este goală
    - $avg \leftarrow (1 - w_q)avg + w_q q$
  - Altfel
    - $m \leftarrow f(time - q_{time})$
    - $avg \leftarrow (1 - w_q)^m avg$
- Dacă $min_{th} \leq avg < max_{th}$

Incrementează ***count***
Calculează probabilitatea p_a:

$$p_b = max_p \frac{(avg - min_{th})}{(max_{th} - min_{th})}$$

$$p_a = \frac{p_b}{(1 - count * p_b)}$$

Marchează pachetul sosit cu probabilitatea p_a
$count \leftarrow 0$

Altfel dacă $max_{th} \leq avg$
Marchează pachetul sosit
$count \leftarrow 0$

Altfel $count \leftarrow -1$

Când coada devine goală
$q\_time \leftarrow time$

Variabile salvate:

avg: dimensiunea medie a cozii
q_time: momentul de început al perioadei *idle*
count: pachete de la ultimul pachet marcat

Parametrii fixați

w_q**:** ponderea cozii
min_{th}**:** pragul minim al dimensiunii cozii
max_{th}: pragul maxim al dimensiunii cozii
max_p: valoarea maximă pentru p_b

Alți parametri

p_a: probabilitatea de marcare a pachetelor, curentă
q: dimensiunea curentă a cozii
time: timpul curent
f(t): o funcție liniară de timp ***t***

Ponderea cozii w_q este determinată de către mărimea și durata *rafalelor* în coadă, ce sunt permise de către gateway. Pragurile min_{th} și max_{th} sunt determinate de dimensiunea medie a cozii dorite. Dimensiunea medie a cozii ce stabilește compromisurile dorite (cum ar fi compromisul dintre maximizarea throughput-ului și minimizarea întârzierii), depinde de caracteristicile rețelei.

1. Calculul dimensiunii cozii medii

Pentru calcularea lungimii medii a cozii, este folosit un filtru trece-jos. Astfel, creşterile de scurtă durată ale lungimii cozii datorate traficului în rafale nu vor duce la o creştere semnificativă a lungimii medii a cozii.

Filtrul trece jos este o medie ponderată mobilă,exponențială:

$$avg \leftarrow (1 - w_q)avg + w_q q$$

Ponderea w_q determină constanta de timp a filtrului trece jos. Mai jos sunt discutate limitele de sus şi de jos ale lui w_q.

Dacă w_q este prea mare, atunci procedura de mediere nu va filtra congestia tranzitorie de la nivelul gateway-ului.

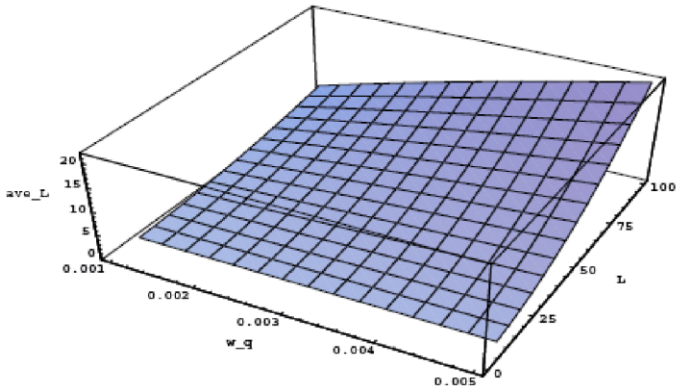
Presupunem că la început coada este goală, cu o lungime medie a cozii de valoare zero, după care, coada creşte de la 0 la L pachete, după L pachete sosite. După sosirea celui de-al L pachet, dimensiunea medie a cozii avg este [4]:

$$avg_L = \sum_{i=1}^{L} i w_q (1 - w_q)^{L-i}$$

$$= w_q (1 - w_q)^L \sum_{i=1}^{L} i \left(\frac{1}{1 - w_q}\right)^i$$

$$= L + 1 + \frac{(1-w_q)^{L+1} - 1}{w_q} \quad (1) \text{ [4]}$$

În figura de mai jos [4] este prezentată dimensiunea medie a cozii avg_L pentru mai multe valori ale lui w_q şi L. Pe axa x este reprezentat w_q, cu valori între 0.001 şi 0.005, iar pe axa y este reprezentat L cu valori cuprinse între 10 şi 100. De exemplu, pentru w_q = 0.001, după o creştere a cozii de la 0 la 100 de pachete, dimensiunea medie a cozii avg_{100} este 4.88 pachete.



Având setat un prag minim min_{th} și știind că dorim să permitem rafale de câte L pachete, atunci w_q ar trebui ales asftel încât să fie satisfăcută ecuația de mai jos, pentru $avg_L < min_{th}$:

$$L + 1 + \frac{(1-w_q)^{L+1} - 1}{w_q} < min_{th} \quad (2)\ [4]$$

De exemplu, pentru min_{th} = 5 și L=50, w_q trebuie ales <=0.0042.

Gateway-urile ce au implementat algoritmul RED trebuie să mențină valoarea calculată a lungimii medii a cozii, ***avg*** sub un anumit prag. Cu toate acestea, aceasta nu are nici un sens dacă media calculată ***avg*** nu este o reflecție rezonabilă a lungimii medii curente. Dacă w_q este setat prea jos, atunci ***avg*** răspunde prea încet la schimbările actuale ale lungimii cozii. În acest caz, gateway-ul este incapabil să sesizeze apariția unei congestii.

Presupunem trecerea cozii de la 0 la 1 pachet, și că pe masură ce pachetele ajung și pleacă cu aceeași rată, coada rămâne la un pachet. De asemenea presupunem că inițial, dimensiunea medie a cozii era zero. În acest caz, vor fi necesare $\frac{-1}{ln(1-w_q)}$ sosiri de pachete (cu dimensiunea cozii rămânând la 1) până când dimensiunea medie a cozii ***avg*** devine 0.63=1-1/e. Pentru w_q = 0.001, sunt necesare 1000 de sosiri de pachete; pentru w_q = 0.002 sunt necesare 500 de sosiri de pachete; pentru w_q = 0.003 sunt necesare 333 sosiri de pachete.

Valorile optime pentru min_{th} și max_{th} depind de dimensiunea medie a cozii. Dacă traficul este în general în rafale, atunci min_{th} trebuie să fie corespunzător de mare pentru a permite menținerea utilizării legăturii la un nivel suficient de înalt. Pentru trafic tipic cu întârzieri destul de mari, un prag minim de un pachet ar duce la o legătură inaccetabil de proastă.

Valoarea optimă pentru max_{th} depinde în parte de întârzierea medie maximă ce este permisă de către gateway.

Algoritmul funcționează eficient atunci când $max_{th} - min_{th}$ este mai mare decât creșterea dimensiunii medii a cozii într-o perioadă de timp. De obicei, max_{th} este cel puțin de 2 ori min_{th}.

2. Calculul probabilității de marcare a pachetelor

Probabilitatea inițială de marcare a pachetelor p_b este calculată ca o funcție liniară de lungimea medie a cozii. Mai jos sunt comparate două metode pentru calculul probabilității finale de marcare a pachetelor și este ilustrat avantajul folosirii celei de-a doua metode. În prima metodă, când lungimea medie a cozii este constantă, numărul de pachete sosite între 2 pachete marcate, este o *variabilă aleatoare geometrică*. În cea de-a doua metodă, numărul de pachete sosite între 2 pachete marcate este o *variabilă aleatoare uniformă*.

Probabilitatea inițială de marcare a pachetelor este calculată cu ajutorul relației

$$p_b = max_p \frac{(avg - min_{th})}{(max_{th} - min_{th})}$$

Parametrul max_p reprezintă valoarea maximă pentru probabilitatea de marcare a pachetelor p_b atinsă când lungimea medie a cozii ajunge la pragul maxim.

$$Prob[X = n] = (1 - p_b)^{n-1} p_b$$

Prin urmare, X este o variabilă aleatoare *geometrică*, cu parametru p_b și $E[X]=1/p_b$.

Cu o dimensiune medie a cozii constantă, scopul este de a marca pachetele la intervale destul de regulate. Nu este de dorit să avem prea multe pachete marcate foarte apropiate, și de asemenea nu este de dorit ca intervalele de timp dintre marcări să fie foarte mari. Ambele variante pot duce la sincronizări globale, cu mai multe conexiuni reducându-și ferestrele în același timp, și ambele variante se pot intampla când X este o variabilă aleatoare geometrică.

O variantă mai bună este ca X să fie o variabilă aleatoare *uniformă* din *{1, 2, ..., $1/p_b$}* (presupunând pentru simplitate că $1/p_b$ este un întreg). Acest lucru se întâmplă când probabilitatea de marcare pentru fiecare pachet sosit este $\frac{p_b}{(1-count*p_b)}$, unde ***count*** este numărul de pachete nemarcate ce au sosit de când a fost marcat ultimul pachet. În acest caz,

$$Prob[X = n] = \frac{p_b}{1-(n-1)p_b}\prod_{i=0}^{n-2}\left(1-\frac{p_b}{1-ip_b}\right)$$

$$= p_b pentru\ 1 \leq n \leq \frac{1}{p_b}$$

Și

$$Prob[X = n] = 0\ pentru\ n > \frac{1}{p_b}$$

În acest caz $E[X] = \frac{1}{2p_b} + \frac{1}{2}$.

În figura de mai jos, este prezentat un experiment ce compară cele două metode de marcare a pachetelor. Linia de sus reprezintă metoda 1, unde fiecare pachet este marcat cu probabilitatea p, pentru p = 0.02. Linia de jos, reprezintă metoda 2, unde fiecare pachet este marcat cu probabilitatea $\frac{p}{(1+ip)}$, pentru p = 0.01 și cu i numărul de pachete nemarcate de la ultimul pachet marcat. Ambele metode au marcat aproape 100 de pachete din 5000 sosite. Axa x reprezintă numărul de pachete. Pentru fiecare metodă, este reprezentat câte un punct pentru fiecare pachet marcat. Așa cum era de așteptat, pachetele marcate sunt mai grupate în cazul primei metode, decât în cea de-a doua.

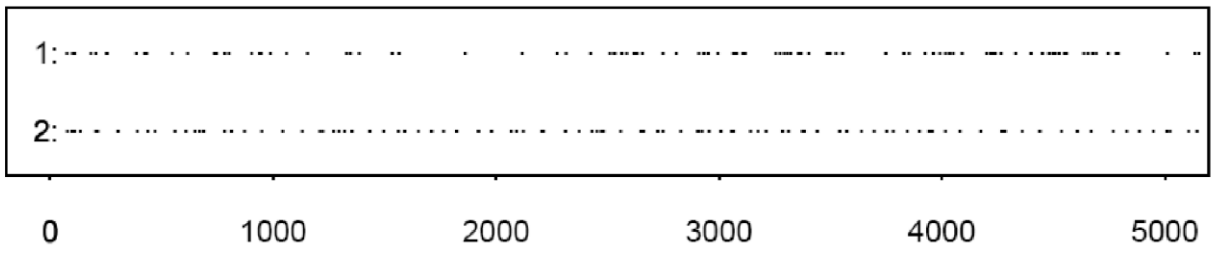


Figura 12. Pachete marcate aleator, comparație între 2 metode de marcare

3. Caracteristici ale algoritmului RED

Evitarea congestiei

Gateway-ul RED garantează faptul că dimensiunea medie calculată a cozii nu depăşeşte pragul maxim prin aruncarea pacheteleor ce ajung la gateway atunci când dimensiunea medie a cozii atinge acest prag. Dacă ponderea w_q pentru procedura mediei mobile ponderate exponenţial a fost stabilită corespunzător, atunci ruterul RED va fi cel care va controla de fapt dimensiunea medie a cozii. Atunci când ruterul RED setează un bit în

antetul pachetelor, în momentul în care dimensiunea medie a cozii depăşeşte pragul maxim, înseamnă că se bazează pe colaborarea dintre surse pentru a controla dimensiunea medie a cozii.

Scală de timp adecvată

După ce este notificată conexiunea asupra apariţiei congestiei, prin marcarea unui pachet, intervalul de timp necesar gateway-ului pentru a sesiza o descreştere a ratei de primire a pachetelor, este de cel puţin o perioadă dus-întors. În cazul gateway-urilor RED, scala de timp pentru detecţia congestiei se potriveşte aproximativ cu scala de timp necesară conexiunilor să răspundă la congestie. Gateway-urile RED nu notifică conexiunile să-şi reducă ferastra ca rezultat al congestiei tranzitorii de la gateway.

Lipsa sincronizării globale

Rata cu care gateway-ul RED marchează pachetele, depinde de nivelul congestiei. Pe durata unei congestii scăzute, gateway-ul are o probabilitate mică de a marca fiecare pachet ce soseşte. Pe măsură ce congestia creşte, probabilitatea de a marca fiecare pachet creşte şi ea. Gateway-urile RED, evită sincronizarea globală marcând pachetele cu o rată cât mai mică posibil.

III. Controlul Congestiei Distribuite

1. Problema:

Congestia apare in cazul in care numarul de pachete transmise prin retea este aproape egal cu capacitatea de manipulare a retelei. Controlul congestiei presupune pastrarea numarului de pachete sub nivelul la care performanta scade dramatic.

2. Solutie:

Intr-o retea, ruterele isi pot controla ratele proprii (individuale), interactionand cu scopul de a obtine o rata globala optima de folosire a retelei. Apare asadar acest concept de congestie distribuita, prin care ruterele sunt modelate ca indivizi intr-o teorie dezvoltata prima data de Frank Kelly. Acesta a apelat la teori din microelectronica si statistica pe care urmeaza sa le detaliem. (Detalii despre conceptele referite se gasesc in bibliografie la rubrica: Controlul Congestiei Distribuite)

Eforturile recente de cercetare au dus la o mai bună proiectare a protocoalelor de transport in internet combinate cu un AQM scalabil (Active Queue Management). Ele au condus catre progrese semnificative in controlul congestiei. Unul dintre cele mai fierbinţi subiecte în acest domeniu este proiectarea unor algoritmi de control al congestiei, care sunt asimptotic stabili în întârzieri eterogene de feedback şi al căror control cu ecuatii nu depind în mod explicit pe RTTs.

În această lucrare, se demonstrează că daca o singură legătură în controlul congestiei foloseste metode cu un Jacobian radial stabil rămâne stabil în întârzieri de feedback-ul arbitrare (inclusiv întârzierile eterogene) şi că starea de stabilitate a unor astfel de metode nu implică nici a întârzierilor. Am extins apoi acest rezultat pentru reţele generice cu misiuni consistente fixe *gat de* sticla si retele de feedback max-min. Pentru a demonstra practic rezultatul obţinut, vom schimba operatorul original al lui Kelly pentru a deveni solide în conformitate cu întârzierile, feedback-ul aleatoriu şi Constantele xed ale ecuatiei de control. Facem un apel de unde rezultă Max-cadru min Kelly de control (M K C) Acesta arată că rata de trimitere converge neted la o exponenţiala a efcientei, şi converge rapid la corectitudine, toate facand mai atrăgătoare pentru viitor pentru reţelele de mare viteză.

3. Introducere teoretica

In ultimii 15 ani, controlul congestiei in internet a evoluat de la metode de feedback binary AIMD/TCP la dezvoltari mult mai interesante bazate pe teoria optimizarii, teoria jocului si teoria

controlului. Este bine stiut ca controlul congestiei TCPurilor este in starea sa inadecvata pentru multe retele de mare viteza. Totusi s-a depuns un efort semnificativ pentru a intelege mai bine proprietatile dorite ale controlului congestiei si pentru a dezvolta noi algoritmi care pot fi utilizati in viitor in retele AQM.

Unul dintre cei mai importanţi factori în proiectarea de congestia de control este stabilitatea asimptotică, care este capacitatea de de protocol, de a evita oscilaţiile în starea de echilibru şi în mod corespunzător a răspunde la perturbatii externe generate de sosire / plecare de variaţie în feedback-ul, şi alte efectele tranzitorii.

Dovezi de stabilitate pentru congestionarea distribuita de control devin progresiv mai complicate ca intarzierile de feedback care sunt luate în considerare, ceea ce este valabil mai ales pentru cazul intarzierilor heterogene unde fiecare user I primeste feedbackul retelei cu intarziere de un anumit timp D_i.

În această lucrare, ne-am stabilit obiectivul nostru de a construi sistem de control de congestie discret care menţine atât stabilitatea şi corectitudinea în temeiul feedback-ului eterogen întârziat care permite utilizatorilor să folosească parametric ficsi pentru ecuatia de control si admite o implementare low-overhead in interiorul ruterelor.

Am rezolva această problemă prin care arată că orice legătură într-o singură-max-min, cu un sistem echitabil stabil radial Jacobian ramane asymptotic stabil in intarzieri directionale arbitrare, extinzand rezultatul catre retele multilink cu sarcini bottleneck fixe, si o aplicat controlerului original propus de Kelly.

Vom numi rezultatul acestor eforturi Controlul Kelly Max-Min (MKC) si vom demonstra ca stabilitatea si corectitudinea nu depend de parametric retelei.

Totodata vom arata ca daca alegem varianta buna de feedback AQM, MKC converge catre o eficienta exponentiala rapida, ceea ce demonstreaza stabilitatea si corectitudinea in intarzieri aleatoare si converge catre o functionare buna aproape la fel de rapid ca AIMD si nu are nevoie de rutere pentru a estima alti parametric ai scaparilor individuale.

Prin izolarea blocajelor pe fiecare traseu si raspunzand numai celor mai congestionate resurse, framework-ul MKC permite pentru dovezi simple de stabilitate, care speram ca vor conduce catre o intelegere mai buna a framework-ului Kelly in comunitatea sistemelor si un eventual rezultat in implementarea actuala a acestor metode in retele reale.

A. Controlul Congestiei dependent de intarzieri

Recent s-a depus mult efort theoretic si experimental pentru a crea o un control robust al congestiei. Lucrarea originala a lui Kelly ofera o interpretare economica a modelului resursa-user in care intregul sistem obtine performanta optima prin maximizarea utilitatii individuale a fiecarui user terminal. Pentru a implementa acest model in o retea descentralizata, Kelly a

descries doi algoritmi (primal si dual) si a demonstrate stabilitatea lor globala in absenta intarzierii feedback-ului. Totusi, daca intarzierea feedback-ului este prezenta in bucla de control, analiza stabilitatii lui Kelly formeaza o arie active de research. In framework-ul lui Kelly fiecare user I apartine [1,N] si primeste o ruta unica r_i care consta in una sau mai multe resurse de retea (rutere). Intarzierile de feedback din retea sunt heterogene si directionale. Intarzierile dus si intors dintre userul I si userul j sunt notate D_{ij} si D_{ji} . Acestea respecta

$$D_i = D_{ij}^{\rightarrow} + D_{ij}^{\leftarrow}$$

$$x_i(n) = x_i(n-1) + \kappa_i(\omega_i - x_i(n - D_i)\eta_i(n)) \qquad (1)$$

Unde k_{ij} este parametru strict pozitiv

ω_i poate fi interpretat ca dorinta userului I sa plateasca pretul pentru folosirea retelei si pentru feedback-ul retelei

$$\eta_i(n) = \sum_{j \in r_i} p_j(n - D_{ij}^{\leftarrow}) \qquad (2)$$

al userului I care este pretul agregat p_i al tuturor resurselor j in calea r_i. Aici $p_j(n)$ este o functie a ratei combinate $y_i(n)$ a tuturor fluxurilor care intra in routerul j:

$$y_j(n) = \sum_{u \in s_j} x_u(n - D_{uj}^{\rightarrow}) \qquad (3)$$

unde s_j reprezinta setul de useri care impart resursa j. Este important faptul ca folosim o notatie in care Di=1 inseamna feedback imediat iar D_i>=2 implica feedback intarziat.

Mai departe se observa ca intarzierea omogena D a sistemului (1) este stabile local daca :

$$\kappa_i \sum_{j \in r_i} \left((p_j + p_j' \sum_{u \in s_j} x_u) \Big|_{x_u^*} \right) < 2 \sin\left(\frac{\pi}{2(2D-1)}\right) \qquad (4)$$

unde x_u^* este un punct stationar al userului u si $p_j(.)$ este presupus a fi diferentialul lui x_u^*.

Pentru intarzieri eterogene o combinatie de conjuncture dezvoltata de Johari, Massoulie si Vinnicombe sugereaza ca intarzierea D poate fi inlocuita simplu cu intarzieri individuale D_i pentru a forma un sistem de N ecuatii de stabilitate. Totuusi, dovada exista doar pentru versiunile continue ale functiei (1) si conduce catre conditia urmatoare de stabilitate:

$$\kappa_i \sum_{j \in r_i} \left(\left(p_j + p_j' \sum_{u \in s_j} x_u \right) \Big|_{x_u^*} \right) < \frac{\pi}{2D_i}. \qquad (5)$$

Inspirata din frameworkul de optimizare al lui Kelly, o metoda aditionala cunoscuta sub numele de MaxNet este propusa si considerate ca avand proprietati de convergenta imbunatatie fata de modelele traditionale ale feadbackului aditiv. In MaxNet, fiecare user I obtine feedback

$\eta_i(t) = \max_{j \in r_i} p_j(t)$ de la cel mai congestionat router in calea sa si aplica $\eta_i(t)$ pentru o lege de control nespecificata $x_i(t) = f_i(\eta_i(t))$. Bazata pe tehnica dezvoltata, autorii demonstreaza ca MaxNet este stabil local in retele generice cu sarcini de constrangere fixe daca $0 < f_i'(\eta_i^*) < x_i^*/D_i,$ unde x_i^* and η_i^* sun respective rata de echilibru si feedbackul stationar al fluxului i.

B. Controlul congestiei intarzierilor independente

Pentru a intelege mai bine, prima conditie de stabilitate a intarzierilor independente este formulate datorita lui Vinnicombe, care prpune si examineaza urmatoarele modele fluide continue ale unei retele cu surse ce opereaza cu algoritmi TCP

$$\dot{x}_i(t) = \frac{x_i(t - D_i)}{D_i}(\alpha_i(t) - (\alpha_i(t) + \beta_i(t))\eta_i(t)) \qquad (6)$$

where $\alpha_i(t) = a(x_i(t)D_i)^n$, $\beta_i(t) = b(x_i(t)D_i)^m$, a, b, m, n sunt constant, $\eta_i(t)$ este feedbackul retelei si pretul $p_j(t) = (y_j(t)/C_j)^B$ i este o aproximare a pierderii pachetului la legatura j cu capacitatea C_j si dimensiunea bufferului B. Este demonstrate ca mai sus controlerul este stabil local daca

$$a(x_i^* D_i)^n < \frac{1}{B}. \qquad (7)$$

Setand n=0 si $a < 1/B$, sistemul ce rezulta va obtine stabilitate independent de intarzieri. Un rezultat aditional este formulat de Ying, care considera urmatorul controller :

$$\dot{x}_i(t) = \kappa_i x_i(t - D_i)\left(\frac{1}{x_i^n(t)} - x_i^m(t)\eta_i(t)\right) \qquad (8)$$

Unde k_I este o constanta. Autorii au demonstrate ca (8) este stabile global indifferent de intarzierea in topologiile retelei generale daca m+n>B. Acest lucru este similar cu cazul nostrum. Totusi, analizele si metodele propuse sunt diferite.

C. Controlul Kelly classic

In aceasta sectiune vom discuta intuitive despre exemple care explica formulele criptice din sectiunile anterioare si demonstreaza in simulari cum intarzierile afecteaza stabilitatea. Vom arata apoi ca controlul Kelly original, sau alt mechanism ce se bazeaza pe suma functiilor de feedback de la rutere individuale, expun un schimb intre convergenta liniara pana la eficienta si pierderea persistenta si stationara de pachete.

4. Exemplu de stabilitate intarziata

Urmatorul exemplu ilustreaza problemele de stabilitate de la (1) cand intarzierile feedback-urilor sunt mari. Presupunem o singura sursa, configuratie single-link si utilizam o functie de congestive care calculeaza pierderea de pachete estimate folosin rate de raspuns instantanee:

$$p(n) = \frac{x(n) - C}{x(n)} \qquad (9)$$

Unde C este capacitatea si x(n) este rata fluxului la un pas discret n. Notam faptul ca functia de pret p(n) in Controlul Kelly original este nonnegativa.
Aplicand Kelly (1) vom obtine

$$x(n) = x(n-1) + \kappa\omega - \kappa(x(n-D) - C). \qquad (10)$$

Mai departe presupunem un set particular de parametric $\kappa = 1/2, \omega = 10$ mb/s si C=1000 mb/s. Rezolvand conditia in (4), vom avea un sistem stabil doar daca intarzierea D este mai mica decat patru unitati de timp.

La intarzieri mai mari D=2 si D=3 fluxul are oscilatii care cresc progresiv inainte de a intra in stationare. Eventual, pe masura ce D devine egal cu patru unitati de timp, sistemul diverge ca in figura.

Folosind acelasi parametru k si reducand omega la 20 kb/s, vom examina (10) via simulari ns2 in care un singur flux trace printr-o legatura cu capacitatea 50 mb/s.
Vom rula fluxul in doua configuratii de retele cu intarziere egala cu 90 si 120 ms. Primul flux va devein stationar dupa oprirea oscilatiilor, in timp ce al doilea flux nu afiseaza nicio conversie .

Din moment ce controlurile Kelly sunt instabile doar in cazul in care conditia (4) este satisfacuta, o strategie naturala de a mentine stabilitatea este pentru fiecare user I pentru a ajusta adaptive parametrul de castig $\kappa_i \sim 1/D_i$ astfel incat (4) sa nu fie incalcata.
Totusi, aceasta metoda depinde de estimarea intarzierilor D_i si conduce catre erori intre fluxurile diferitelor RTTuri.

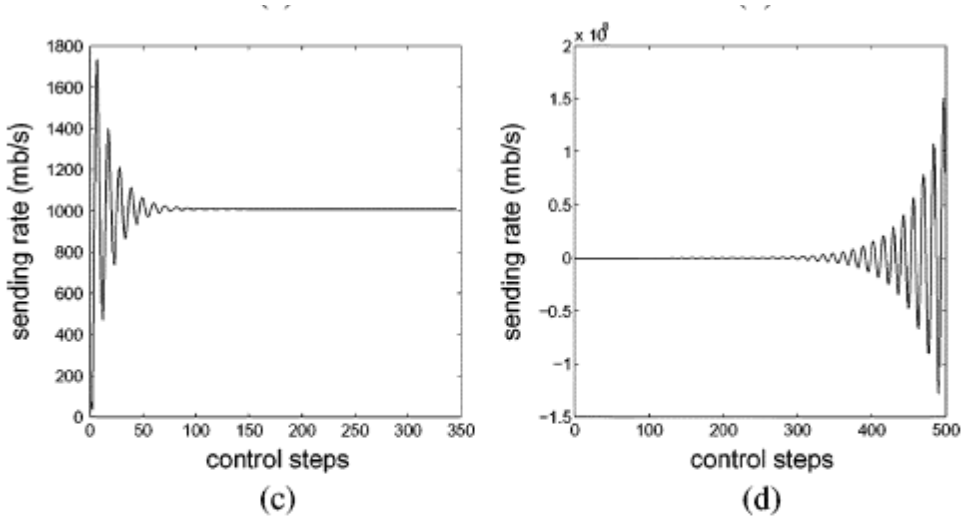
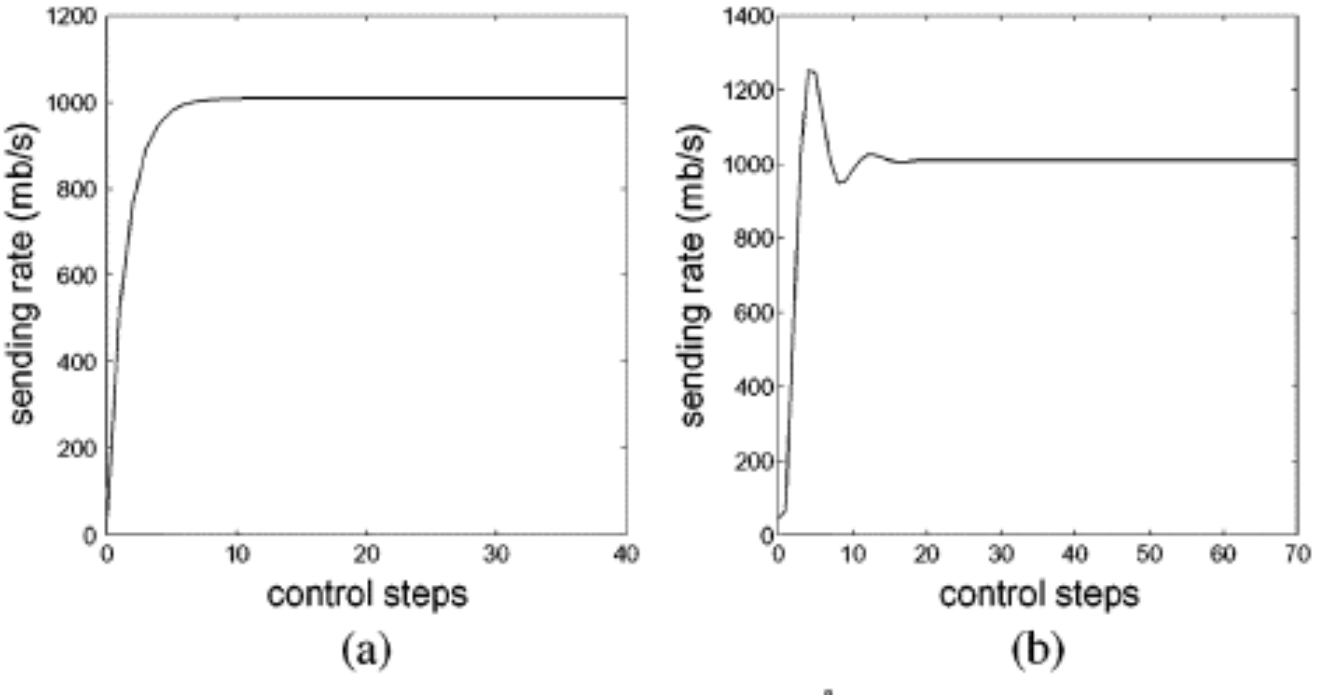


Fig. 1. Stability of Kelly control under different feedback delays ($\kappa = 1/2$, $\omega = 10$ mb/s, and $C = 1000$ mb/s). (a) $D = 1$, (b) $D = 2$, (c) $D = 3$, and (d) $D = 4$.



Rata de alocare stationara

Asa cum am mentionat in subsectiunea anterioara, functia de pret ar trebui sa poata lua valori negative cum in asa fel incat viteza de convergenta a controlului Kelly sa poata fi imbunatatita de la linear la exponential. Totusi, vom arata in continuare ca modificarile impugn o problema in

alocarea stationara a resurselor. Considerand o retea de M resurse si N useri omogeni. Mai departe vom presupune ca nu exista reduntanta in retea. Totusi putem define matricea de ruting A_{NxM} cum ar fi A_{ij}=1 daca userul I trece de resursa j si A_{ij} = 0 in caz contrar.
Deci coloana j a A din vectorul V_j. In mod cert V_j identifica setul s_j al fluxului de trecere prin ruterul j.
Fie

$$\mathbf{x}_j(n) = (x_1(n-D_{1j}^{\rightarrow}), x_2(n-D_{2j}^{\rightarrow}), \ldots, x_N(n-D_{Nj}^{\rightarrow}))$$

Vectorul al ratelor trimise de userii individuali observati de ruterul j la momentul n.

$$p_j(n) = \frac{\mathbf{x}_j(n) \cdot \mathbf{V}_j - C_j}{\mathbf{x}_j(n) \cdot \mathbf{V}_j} \qquad (11)$$

Unde operatorul (.) reprezinta inmultire vectoriala. Atunc vom avea urmatorul rezultat

Lema 1

$\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_N^*)$ este rata de alocare stationara a controlului Kelly cu functia (11).

Atunci $\mathbf{x}^*$ satisfice

$$\sum_{i=1}^{N} M_i x_i^* = \sum_{j=1}^{M} C_j + N\omega. \qquad (12)$$

Demonstratie :
In cazul initial putem scrie ecuatia de control a userului i ca :

$$\begin{aligned} x_i^* &= (1 - \kappa\eta_i^*)x_i^* + \kappa\omega \\ &= \left(1 - \kappa \sum_{j \in r_i} \frac{\mathbf{x}^* \cdot \mathbf{V}_j - C_j}{\mathbf{x}^* \cdot \mathbf{V}_j}\right) x_i^* + \kappa\omega \qquad (13) \end{aligned}$$

Unde η_i^* reprezinta feedbackul stationar trimis de userul i. Folosind operatii simple in (13) vom avea:

$$M_i x_i^* - \sum_{j \in r_i} \left(\frac{x_i^* C_j}{\mathbf{x}^* \cdot \mathbf{V}_j}\right) = \omega. \qquad (14)$$

$$\sum_{i=1}^{N} M_i x_i^* = \sum_{i=1}^{N} \sum_{j \in r_i} \left(\frac{x_i^* C_j}{\mathbf{x}^* \cdot \mathbf{V}_j}\right) + N\omega. \qquad (15)$$

$$\sum_{i=1}^{N} M_i x_i^* = \sum_{j=1}^{M} \sum_{i \in s_j} \left(\frac{x_i^* C_j}{\mathbf{x}^* \cdot \mathbf{V}_j} \right) + N\omega$$
$$= \sum_{j=1}^{M} (\mathbf{x}^* \cdot \mathbf{V}_j) \left(\frac{C_j}{\mathbf{x}^* \cdot \mathbf{V}_j} \right) + N\omega$$
$$= \sum_{i=1}^{M} C_i + N\omega \qquad (16)$$

Ceea ce demonstreaza ipoteza.

Lema 1 produce o conexiune intre alocarea stationara a resurselor si lungimea caii fiecarui flux. Conform (12), ratele stationare x_i^* sunt constranse de capacitatea tuturor resurselor in loc sa fie constranse de constrangerile individuale. De fapt, aceasta observatie arata importanta diferentelor dintre caile reale de retea, care sunt limitate de cea mai lenta resursa si intre un model de corectitudine proportional, fapt ce ne duce la capacitatea tuturor resurselor din retea. Asa cum am demonstrate, aceasta diferenta ne duce la un overflow semnificativ al ruterelor lente si al subutilizarii celor rapide intr-o cale data.

In urmatoarea sectiune vom propune un controller care depaseste problemele celui dinainte.

5. Implementare

Vom studia acum cum putem implementa functia AQM scalabila in interiorul ruterelor pentru a obtine feedbackul necesar fluxurilor MKC. Acesta este o problema de design nontrivial din moment ce pierderea de pachete ideala se bazeaza pe suma ratelor $x_i(n)$ instantanee, care nu sunt niciodata cunoscute de router. In astfel de cazuri, o abordare frecventa este de a aproxima modelul cu cateva functii medii de timp calculate in ruter. Totusi, cum am mentionat in introducere, aceast lucru nu conduce direct la un framework fara oscilatii atat timp cat intarzierile directionale ale retelelor reale introduce diverse inconsistente in bucla de feedback si conduc gresit ruterul sa produca estimate incorecte ale $X(n) = \sum_i x_i(n)$.

In cele ce urmeaza, vom face o descriere variata a diferitelor problem de implementare AQM si vom simula EMKC in ns2 cu intarzieri de feedback eterogene.

A. Headerul de pachet

Asa cum se vede in figura 5, headerul pachetului MKC consta in doua parti – un header de 16 biti si un header de 4 biti.
Headerul routerului incapsuleaza informatia necesara pentru ca routerul sa genereze feedback AQM precis si subsecvential pentru ca ultimul user sa ajusteze rata de trimitere.

Campul de *id* este unica eticheta care identifica ruterul care a generat feedback-ul. Acest camp este folosit de fluxuri pentru a detecta schimbari in constrangeri, in cazul in care asteapta un extra RTT inainte sa raspunda congestiei semnalului unui nou router. Campul *seq* este o variabila locala incrementata de router de fiecare daca cand produce o noua valoare a pierderii pachetelor p. Campul Δ contine lungimea intervalului mediu folosi de ruter in calculul feedbackului.

Campul *usr* este necesar pentru fluxurile terminale pentru a determina rata $x_i(n-D_i)$ care era sub efectul RTT mai inainte.

Cea mai usoara metoda de implementare a functionalitatii este de a injecta valoarea lui $x_i(n)$ in fiecare pachet ce pleaca si apoi de a-i cere receptorului sa returneze acest camp. Un mod de folosire mai sofisticat este discutat in aceasta sectiune.

6. Rolul router-ului

MKC decupleaza operatiile userilor si pe cele ale routerelor lasand o implementare descentralizata. Task-ul major al routerului este de a genera feedbackul AQM si de a-l insera in header pachetelor ce trec prin el. Totusi, trebuie mentionat ca routerul niciodata nu stie rata exacta a fluxurilor ce vin. Dar, pentru a aproxima rezultatul ideal al pierderii pachetelor, routerul conduce calculul lui p(n) catre o scala discrete in timp a unitatilor de timp Δ. Pentru fiecare pachet ce ajunge in intervalul current Δ, routerul insereaza in headerul pachetului informatia feedbackului gasita in intervalul anterior Δ. Ca o consecinta, feedbackul este modificat de unitatile de timp Δ in router .

In timpul intervalului Δ, routerul retine o variabila locala S, care urmareste intreaga cantitate de date care a ajuns de la inceputul intervalului. Specific pentru fiecare pachet ce ajunge k din fluxul I, routerul incrementeaza S dupa dimensiunea pachetului $S=S+s_i(k)$. Routerul estimeaza daca $\tilde{p}$ este mai mare decat cea condusa in pachet. Daca da, routerul rescrie intrarile corespunzatoare in pachet si pleaseaza propriul ID de router, si numarul secvential in header. In acest mod, dupa traversarea intregii cai, fiecare pachet inregistreaza informative de la majoritatea legaturilor congestionate. La sfarsitul intervalului Δ, routerul aproximeaza rata de sosire combinata $X(n) = \sum_{i=1}^{N} x_i(n - D_i^{\rightarrow})$ prin normarea lui S cu timpul $X(n) = \sum_{i=1}^{N} x_i(n - D_i^{\rightarrow})$:

$$\tilde{X} = \frac{S}{\Delta}. \qquad (71)$$

Bibliografie:

Controlul congestiei:

[1] Tannenbaum, Retele de calculatoare , NJ: Prentice-Hall

[2] D. Berstekas and R. Gallager, Data Networks. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[3] L. S. Brakmo and L. Peterson, “TCP Vegas: End to end congestion avoidance on a global internet,” IEEE J. Select. Areas Commun. , vol. 13, pp. 1465–1480, Oct. 1995.

[4] D. H. Choe and S. H. Low, “Stabilized Vegas,” in Proc. IEEE INFOCOM, 2003, pp. 2290 –2300.

[5] D. D. Clark, “The design philosophy of the DARPA internet protocols,” in ACM Comput. Commun. Rev., Proc. ACM SIGCOMM ’88 , vol. 18, 1988, pp. 106 –114.

[6] M. E. Crovella and A. Bestavros, “Self –similarity in World Wide Web traf fic: Evidence and possible causes,” IEEE/ACM Trans. Networking , vol. 5, no. 6, pp. 835–846, Dec. 1997.

[7] S. Deb and R. Srikant, “Global stability of congestion controllers for the internet,” IEEE Trans. Automat. Contr. , vol. 48, no. 6, pp. 1055–1059, Dec. 2003.

[8] FAST Project [Online]. Available: http://netlab.caltech.edu/FAST/

[9] C. Hollot, V. Misra, D. Towsley, and W.-B. Gong, “A control theoretic analysis of RED, ” in Proc. IEEE INFOCOM, Apr. 2001, pp. 1510 –1519.

[10] V. Firoiu and M. Borden, “A study of active queue management for congestion control, ” in Proc. IEEE INFOCOM , Mar. 2000, pp. 1435–1444.

[11] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” IEEE/ACM Trans. Networking , vol. 1, no. 4, pp. 397–413, Aug. 1993.

[12] S. Floyd. (2002) HighSpeed TCP for Large Congestion Windows. IETF, Internet Draft. [Online]. Available: http://www.ietf.org/internet-drafts/draft- floyd-tcp-highspeed-00.txt

[13] R. J. Gibbens and F. P. Kelly, “Resource pricing and the evolution of congestion control, ” Automatica, vol. 35, pp. 1969 –1985, 1999.

[14] , “Distributed connection acceptance control for a connectionless network,” in Proc. 16th Int. Teletraffic Congr., Edinburgh, U.K., Jun. 1999.

[15] F. Baccelli and D. Hong, “Interaction of TCP flows as billiards,” in Proc. IEEE INFOCOM, 2003, pp. 895–905.

[16] V. Jacobson, “Congestion avoidance and control,” in Proc. ACM SIGCOMM, 1988, pp. 314–329.

[17] D. M. Chiu and R. Jain, “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks,” Comput. Networks ISDN Sys. , vol. 17, pp. 1 –14, 1989.

[18] R. Johari and D. Tan, “End-to-end congestion control for the internet: Delays and stability,” IEEE/ACM Trans. Networking, vol. 9, no. 6, pp. 818–832, Dec. 2001.

[19] D. Katabi, M. Handley, and C. Rohrs, “Internet congestion control for high bandwidth-delay product networks, ” in Proc. ACM SIGCOMM , Pittsburgh, PA, Aug. 2002.

[20] F. P. Kelly, A. Maulloo, and D. Tan, “Rate control for communication networks: Shadow prices, proportional fairness and stability, ” J. Oper. Res. Society , vol. 49, no. 3, pp. 237 –252, Mar. 1998.

[21] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management," IEEE Network , vol. 15, no. 3, pp. 48 –53, May-Jun. 2001.

RED:

S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions*
on Networking, August 1993, 1(4): 397-413.
J. Heinanen, F. baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, June 1999.
W. Feng, D. Kandlur, D. Saha, and K Shin, "A selfconfiguring RED gateway," in *Proc*. IEEE INFOCOM, March 1999, pp. 1320-1328.
P. Ranjan, E. H. Abed, and R. J. La, "Nonlinear instabilities in TCP-RED," in Proc. IEEE INFOCOM, 2002.
W. Feng, K. Kandlur, D. Saha, and K. Shin, "Techniques for eliminating packet loss in congested TCP/IP Network,"
University of Michigan CSE-TR-349-97, November 1997.
http://www.icir.org/floyed

Controlul Congestiei Distribuite

R. Bronson*, Schaum's Outline of Theory and Problems of Matrix Operations*. New York: McGraw-Hill, 1988.
D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comput. Netw.*
ISDN Syst., vol. 17, no. 1, pp. 1–14, Jun. 1989.
M. Dai and D. Loguinov, "Analysis of rate-distortion functions and congestion control in scalable internet video streaming," in *Proc. ACM NOSSDAV*, Jun. 2003, pp. 60–69.
Y. Zhang, S.-R. Kang, and D. Loguinov, "Delayed stability and performance of distributed congestion control," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 307–318.
Y. Zhang, D. Leonard, and D. Loguinov, "JetMax: Scalable maxmin congestion control for high-speed heterogeneous networks," in *Proc.*
IEEE INFOCOM, Apr. 2006.
Y. Zhang and D. Loguinov, "Local and global stability of symmetric heterogeneously-delayed control systems," in *Proc. IEEE CDC*, Dec. 2004, pp. 5004–5009.
http://www2.fz-juelich.de/nic-series/volume33/761.pdf

Repartizarea documentatiei:

Epure Adriana

I.Controlul Congestiei
- 1.Algoritmi pentru controlul congestiei
- 2.Principii generale ale controlului congestiei
- 3.Politici pentru prevenirea congestiei
- 4.Formarea traficului
- 5.Algoritmul găleţii găurite
- 6.Algoritmul găleţii cu jeton
- 7.Specificarea fluxului

Berevoianu Adelina

II. Random Early Drop (RED)
- 1.Calculul dimensiunii cozii medii
- 2. Calculul probabilitatii de marcare a pachetelor
- 3. Caracteristici ale algoritmului RED

Tinta Mihaita

III.Controlul Congestiei Distribuite
- 1.Problema
- 2.Solutie
- 3.Introducere teoretica
  - A.Controlul Congestiei dependent de intarzieri
  - B. Controlul congestiei intarzierilor independente
  - C. Controlul Kelly classic
- 4.Exemplu de stabilitate intarziata
- 5.Implementare
- 6.Rolul router-ului

Bibliografie