Algoritmi de combatere a congestiei în buclă deschisă

-Rețele de calculatoare-

Prof.coord. : Student:

**Prof.dr.ing.Ștefan Stăncescu Oțelea Ștefan,443A**

Cuprins

Capitolul 1:Introducere........................................................................................3

 1.1.Ce este congestia și cum poate fi evitată.........................................3

 1.2.Controlul Congestiei vs. Controlul Fluxului......................................5

Capitolul 2:Algoritmi de combatere a congestiei în buclă deschisă.....................8

 2.1.Mecanismul de control “Call Admission Control”............................8

 2.2.Supraaprovizionarea capacității rețelei.........................................10

 2.3.Leaky Bucket..................................................................................11

 2.4.Token Bucket..................................................................................12

Capitolul3:Concluzii și considerații personale....................................................14

Capitolul4:Bibliografie........................................................................................15

**Capitolul 1: Introducere**

**1.1.Ce este congestia și cum poate fi evitată?**

Congestia,în contextul rețelelor de comunicații, se referă la o stare anume a rețelei unde un nod sau o legatură între noduri transportă atât de multă informație încât se poate deteriora calitatea serviciilor rețelei, rezultând următoarele: întârzieri ale cozilor, pierderi ale pachetelor și blocarea conexiunilor. Într-o rețea congestionată, timpul de răspuns încetinește rezultând o rată de transfer mai mică în rețea.Congestia apare când lățimea de bandă este insuficientă iar traficul de date depășește capacitatea rețelei.

Pierderile de pachete datorate congestiei sunt parțial cuantificate de protocoalele de retransmisie, fapt ce menține starea de congestie a rețelei după reducerea inițială a încărcării rețelei. Acest comportament cauzat de protocoalele de retransmisie poate crea două stări stabile ale rețelei : una în care routerele și celelalte dispozitive din componența rețelei se ocupă cu reducerea traficului inițial, iar cealaltă se ocupă cu menținerea reducerii transferului în rețea după ce s-a produs retransmisia.

În rețelele moderne, evitarea colapsului rețelei cauzat de congestie, implică aplicarea unor tehnici de combatere a congestiei în paralel cu controlul acesteia, deoarece indiferent de ce măsuri se aplică pentru a evita congestia aceasta se întâmplă datorită încărcării tot mai mari a rețelelor și a existenței unui număr din ce în ce mai mare de utilizatori(gazde/hosturi) conectate în același timp la o rețea.

Congestia poate fi descrisă ca un efect fundamental al resurselor limitate ale rețelei, în special viteza de procesare a ruterelor dar și transferul mare la care sunt supuse legăturile(link-urile).Procesele de direcționare a traficului realizate de rutere, folosesc un microprocesor. Timpul cumulativ de procesare a ruterelor afectează în mod direct congestia rețelelor. De aceea, ruterele intermediare unei conexiuni virtuale,pot îndepărta pachetele de date când acestea depășesc capacitatea de transfer a conexiunii respective.Când se întâmplă acest lucru,sunt trimise pachete de date adiționale pentru a compensa lipsa pachetelor care au fost îndepărtate de către ruterele intermediare. De aceea congestia rețelelor conduce adesea la colapsul rețelelor.Congestia tinde să se hranească pe sine insăsi pentru a obține probleme din ce în ce mai grave. Deci, cauza majoră a congestiei este adesea calitatea de rafală a traficului de pachete, acestea nemaiputând să fie gesionate conform protocoalelor de transport TCP/IP.

**Colapsul congestiv** constă într-o stare în care poate ajunge o rețea de calculatoare bazată pe schimb de pachete, atunci când nu se mai poate desfășura niciun fel de comunicație în rețeaua respectivă datorită congestiei. Colapsul congestiv apare în general, în puncte ale rețelei în care traficul este “sufocat”, unde tot traficul care converge într-un singur nod depășește lățimea de bandă alocată conexiunilor nodului respectiv. Cel mai adesea vom întâlni această stare în punctele de legătură între rețelele locale și cele extinse(LAN->WAN).

Internetul a experimentat pentru prima dată o problemă numita “**Congestion Collapse”** ,în anii 80. Această problema poate fi descrisă în felul următor:mai mulți biți ai unor pachete ar fi dispărut și reapărut în interiorul anumitor pachete. Acest lucru s-a întâmplat însă doar pentru protocolul TCP. Operații precum „**ping”** sau obținerea unor pachete UDP se puteau realiza, însă protocoale precum „**Telnet**” sau „**FTP**” au eșuat. Dar acest lucru depindea de locul unde se dorea a fi transmise pachetele(unele host-uri erau în regulă însă altele nu) dar și de perioada zilei(în weekend-uri rețeaua era mult mai liberă, iar în timpul săptămânii pe în jurul orei 1 pm pe coasta de Est a Americii rețeaua funcționa foarte prost deoarece in acel moment populația de pe coasta de Vest a Americii începea programul de lucru).

O altă experiență neplăcută a acestui „**Congestion Collapse**” a fost faptul că rețeaua nu funcționa în zile în care te așteptai mai puțin. De aceea, o mulțime de încercări de a crea aplicații care să rezolve aceste probleme s-au dovedit a fi inutile în fața eșecurilor rețelelor.

Una dintre ele a constat în ideea de a avea doua servere diferite pentru fiecare domeniu de adrese. Această idee și-a cunoscut eșecul în momentul în care s-a pus problema comunicării între cele două servere. În perioadele de congestie comunicarea între cele două servere primare ar eșua în mod repetat.

În exemplul din **figura 1.1** sunt expuse două furnizoare de servicii („**Service Providers**”: ISP1 si ISP2) împreună cu doi clienți fiecare.Ele sunt interconectate printr-o legatură de 300 kbps cele două ISP-uri necunoscând configruațiile celelilalte.

Clientul 0 trimite date clientului 4 în timp ce clientul 1 trimite date clientului 5, ambele trimițând cât de multe date pot(100kbps), neexistând nicun mecanism de control al congestiei în această configurație. ISP1 va notifica faptul ca legătura sa de ieșire nu este utilizată la capacitate maximă(200kbps = 2/3\* 300kbps).Totuși decide să actualizeze legătura clientului 0 cu routerul 2 la 1Mbps . În urma acestei decizii, logic ar fi ca și legătura între cele două ISP-uri să fie actualizată de catre ISP1. Acest lucru nu este cunoscut de către ISP1 și, datorită diferenței mari de lațime de bandă asignată clienților se va produce congestia.



Figura 1.1([https://books.google.ro/”Network](https://books.google.ro/) “**Congestion Control**” de Michael Welzl)

**Combaterea congestiei** constă în controlul traficului de la intrarea unei rețele de telecomunicații, prin încercarea de a evita supraîncărcarea a oricăruia dintre nodurile intermediare conectate la rețeaua respectivă. Pe scurt, evitarea se realizează prin reducerea resurselor rețelei treptat sau prin reducerea ratei de emisie a pachetelor.

**1.2.Controlul Congestiei vs. Controlul Fluxului**

Asigurându-ne că tot traficul este transportat într-o subrețea se numește **Controlul Congestiei**, iar controlul din punct în punct a traficului între un emițător si un receptor se numește **Controlul Fluxului**.

**Controlul Congestiei** implică toate gazdele, routerele, „procesele store-and-forwarding”și alți factori care au ceva în comun cu capacitatea subrețelei.**Controlul Fluxului** ar trebui să încetinească un emițător care încearcă sa trimită mai mult decât poate să recepționeze un receptor.Unii algoritmi de control a congestiei implementează, de asemenea încetinirea ratei de emisie a surselor, deci, **Controlul Fluxului** și **Controlul Congestiei** pot coexista într-un context.

Din moment ce nodurile intermediare pot acționa ca puncte de control și măsură în același moment, schema de control a congestiei poate exista teoretic si în subrețele unde emitorul și receptorul nu sunt implicate. Totuși, aceasta nu este o metodă recomandată și practică din moment ce multe tehnologii sunt proiectate să funcționeze în medii cu diverse condiții, incluzând până și cea mai mica setare posibilă: un emitor sau un receptor, interconectate printr-un singur link. Cât timp fenomenul de „**Congestion Collapse”** e mai puțin problematic în acest scenariu, receptorul trebuie sa dețină destule mijloace care să poată să încetinească rata de emisie, dacă acesta este ocupat să facă mai multe operațiuni pe lângă primirea de pachete sau pur și simplu nu este destul de rapid. În acest caz, funcția care îl informează pe emitor să încetinească emisia se numește **Controlul Fluxului**.

Scopul **Controlului Fluxului** este de a proteja emitorul de supraîncărcare,pe când scopul **Controlului Congestiei** este de a proteja rețeaua.Ambele funcții se întrepătrund în implementări combinate deoarece mecanismul care stă la baza lor este similar: feedback-ul este folosit pentru a actualiza rata fluxului de date. Din moment ce pare rezonabil să protejăm atât receptorul cât și rețeaua de supraîncărcare în același timp, astfel de implementări ar trebui să fie astfel:emitorul folosește o rată care constă din minimul rezultatelor obținute prin calculele operațiilor de control al fluxului și control al congestiei.

Datorită acestor asemănări, termenii de **Controlul Fluxului** și **Controlul Congestiei** sunt de multe ori sinonime, sau unul din ele este considerat un caz particular al celuilalt.

**Controlul Congestiei**



 Figura 1.2(<http://166.111.64.8/~zhengyouquan/networking.html>)

În **figura 1.2.** este reprezentată rețeaua Internet în situația în care apare fenomenul de congestie. Aceasta se încearcă a fi rezolvată prin tehnica numită “**Controlul Congestiei”**, al cărui scop constă în asigurarea faptului că tot traficul este transportat într-una din cele șase subrețele. Congestia apare la unul din cele trei routere prezente în schemă, nod al rețelei în care pachetele care depășesc capacitatea de transport(lățimea de bandă) fie sunt aruncate și retransmise(ceea ce ocupă foarte mult timp), fie sunt marcate pentru a transmite emițătorului ce pachete depăsesc capacitatea și nu pot fi trimise mai departe, cu scopul de a fi retrimise.

Emițătorul “știe” ce pachete să retrimită sau cât de mult să încetinească rata de de emisie bazându-se pe informația furnizată de protocolul TCP. De asemenea nodurile de rutare transmit emițătorului informații despre unde a apărut congestia și cât să fie redusă rata de emisie.

 **Controlul Fluxului**



 Figura 1.3(<http://lea.hamradio.si>)

În **figura 1.3** este ilustrată metoda de control al transferului datelor numită “**Controlul Fluxului**”. Spre deosebire de “**Controlul Congestiei**”, “**Controlul Fluxului”** verifică din punct în punct starea traficului pentru a interveni în situția în care apare congestia. Se utilizează un feedback constant între nodurile consecutive ce apar pe calea dintre un emițător și un receptor. De exemplu, în figură se observă cum emițătorul trimite un flux mare de date nodului #1. Nodul #1 constată că nu mai are destul spațiu în buffer și ia următoarele decizii:

1. Dacă nu este depășit bufferul de memorie, transmite emițătorului să reducă rata de emisie și transmite mai departe către nodul #2 cu o rată mai mică.
2. Dacă este depășit bufferul de memorie o parte din pachete sunt aruncate(cele care depășesc capacitatea bufferului) o parte sunt transmise nodului #2 iar nodul #1 transmite sursei să emită cu o rată mult mai mică pentru a se putea goli bufferul nodului #1.
3. Se repetă cei doi pași în funcție de situație până când toate pachetele ajung la destinație.

**Capitolul 2:Algoritmi de combatere a congestiei în buclă deschisă**

În timpul setării conexiunii, se determină rata de transmisie a conexiunii. Odată ce este setată conexiunea, pachetele sunt trimise și procesate de sistem fără ca rețeaua în sine sau destinatarul pachetului să trimita un feedback. Din acest motiv, mecanismul de combatere a congestiei este denumit ca “mecanism de control a congestiei în buclă deschisă ”.

**2.1.Mecanismul de control “Call Admission Control”(Gestionarea Apelurilor)**

În timpul inițializării sesiunii unei legături, amblele capete ale conexiunii interschimbă informațiile de control și parametrii de calitate a serviciului QoS**(„Quality of Service”)** cum ar fi: rata maximă de transfer, rata minimă de transfer, dimensiunea bufferelor, și întârzierea maximă permisă. Informațiile de control și parametrii de calitate a serviciului(QoS) formează împreună descriptorul traficului. Rețeaua admite unui șir de date accesul la rețea numai dacă există resurse necesare disponibile în rețea. Traficul poate fi susținut prin asigurarea unei rate de transfer maxime la destinație cu scopul de a evita supraîncărcarea bufferelor din rutere.

Ca exemplu pentru gestionarea apelurilor, putem considera circuitele comutate din rețelele de telefonie. În timpul configurării unui apel, o cale(conexiune) particulară cu o anumită lățime de bandă și rată de transfer este rezervată în rețea pentru a asigura integritatea parametrilor QoS în timpul unei sesiuni de transfer. Dezavantajul acestui model constă în faptul că nu sunt utilizate la maxim toate resursele rețelei.

**Protocolul TCP în mecanismul CAC(Call Admission Control)**

Sistemele specifice CAC tradiționale iau în calcul doar performanțele nivelului de rețea și sunt în principiu destinate rețelelor wireless cu circuite comutate. Din moment ce comunicațiile wireless se vor baza pe sisteme de pachete comutate, vor fi vizate caracteristicile nivelului de pachete pentru a îmbunătăți performanțele sistemului.

Pentru o utilizare eficientă a resurselor sistemului din punctul de vedere al controlului admisiei, este propus un sistem de control CAC(„**Call Admission Control”**) care să țină cont de protocolul TCP pentru a reglementa dinamicitatea nivelului de pachete a fluxului TCP.

În condițiile în care tehnologia comutării de pachete oferă o utilizare mai eficientă a resurselor “invizibile” a rețelelor wireless decât tehnologia comutării de circuite, viitoarele rețele wireless 3G si 4G vor fi bazate pe comutări de pachete. În rețelele wireless cu comutare de pachete, dinamicitatea nivelului de pachet este centrată pe performanța nivelului de apel, de aceea atat nivelul de pachet cât și nivelul de apel vor trebui să ia în considerație mecanismul de gestiune a apelurilor(„**Call Admission Control”**) .

Totuși, atât sistemul CAC tradițional cât și sistemul CAC adaptat la protocolul TCP, nu au avut în vedere efectul aplicațiilor TCP în performanța nivelului de pachet. Deoarece protocolul TCP este direct responsabil de transportul datelor în viitoarele rețele wireless de comutare de pachete , este deosebit de important studiul efectelor TCP asupra sistemului CAC.

În descrierea sistemului “**Call Admission Control**” va fi descris felul în care dinamica nivelului de pachet afectează controlul admisiei la nivelul de apel. Spre exemplu, rata de sosirii a pachetelor a traficului de voce este fix, iar probabilitatea de pierdere a unor pachete trebuie să fie mai mică decât un anumit prag. Dacă sistemul deservește doar anumite clase de apeluri cu profiluri QoS fixe, sistemul nu arată deloc flexibilitate (fie acceptă pachetele cu QoS-ul potrivit, fie le respinge).

Totuși, pentru un apel TCP , cerințele QoS nu sunt fixe, fiind încadrate într-o gamă largă de QoS-uri(„**Quality of Service”**), de aici caracteristica TCP de protocol adaptiv. Dacă este disponibilă mai multă lățime de bandă, TCP va crește rata de transmisie.Dacă nu există, iar congestia apare(ori tinde să apară), TCP va reduce rata de transmisie, pentru a se potrivi lățimii de bandă disponibile. Totuși, TCP este capabil să se confrunte cu o gamă largă de situații, iar această caracteristică permite stației de bază să controleze apelurile TCP.

Spre exemplu, să presupunem că există un nou apel iar sistemul nu are resurse suficiente pentru a admite acest apel. Decât să respingă unul sau mai multe pachete asociate apelului respectiv,sistemul va încerca să reducă rata de transmisie, a apelurilor TCP controlând astfel probabilitatea de pierdere a pachetelor.Această procedură va crește șansa de acceptare a pachetelor după ce s-a efectuat un apel, iar probabilitatea de respingere va scădea.



Figura 2.1a Figura 2.1.b

(<http://www.cisco.com/c/en/us/products/collateral/unified-communications/rsvp-agent/solution_overview_c22-521818.html>)

**Figura 2.1a.** ilustrează un sistem tradițional de gestiune a apelurilor **(“Traditional Call Admission Control”)** bazat pe comutare de circuite, în care un circuit este stabilit de fiecare dată când un apel este efectuat.

**Figura 2.1.b.** ilustrează un sistem de gestiune a apelurilor actual (“**Call Admission Control”**) care folosește comutația de pachete, sub controlul protocoalelor TCP și IP.

Această legătură este teoretic independentă de conexiune însă pentru mai mult de două apeluri de voce simultane calitatea s-ar degrada în mod normal. Aici intervine sistemul CAC(“**Call Admission Control”**) care prin managerul de comunicații gestionează ordinea de efectuare a apelurilor cu scopul de a limita degradarea calității apelurilor. În practică nu sunt necesare doar trei sau mai multe apeluri (într-o conexiune) pentru a degrada calitatea lor, ci mult mai multe deoarece lățimea de bandă este crescută proporțional cu numărul de utilizatori sau cu numărul de servicii alocate acestora.

**2.2**. **Supraaprovizionarea capacității rețelei („Over-provisioning”)**

O soluţie uşoară pentru rezolvarea problemei congestiei este aceea de a furniza ruterului suficientă capacitate, spaţiu tampon şi lăţime de bandă încât pachetele să zboare pur şi simplu. Problema cu această soluţie este aceea că este costisitoare. Odată cu trecerea timpului și datorită faptului că proiectanţii au o idee mai clară despre cât de mult înseamnă suficient, această tehnică ar putea deveni chiar realistă. Până la un punct, sistemul telefonic este supraapovizionat. Se întâmplă rar să ridici receptorul telefonului şi să nu ai ton de formare instantaneu. Pur şi simplu există atâta capacitate disponibilă încât cererea va fi întotdeauna satisfăcută.

Supraaprovizionarea capacității rețelelor se bazează pe furnizarea de suficientă lățime de bandă pentru a face supraîncărcarea să pară nesemnificativă, pentru a obține prin urmare calitatea dorită a sistemului.Capacitățile de legătură sunt alese în așa fel încât ele sunt rare ori depășite de traficul preconizat.

Rata fluxului Internetului este adesea greu de estimat datorită dependenței pe termen lung de o serie de factori care au fost stabiliți începând de la scara de LAN(„**Local Area Network”**), care mai apoi au fost scalați pentru dimensiunile traficului WWW și pentru o rată reală de bitflow.

Măsurătorile arată că fluctuațiile de trafic pe scară de măsură mică tind să fie decorelate în timp ce dezvăluie o structură asemănătoare pe o scară de 1 sau mai mult.Acest lucru este problematic din moment ce similaritățile provoacă pierderi importante de pachete în buffere.Totuși, o superpoziție a unui număr suficient de mare de surse de distanță mare poate duce la valori pentru o utilizare moderată în limita unei probabilități de supraîncărcare a bufferului.

**2.3.Algoritmul găleții găurite(„Leaky Bucket”)**

Inițial, vom considera un recipient(mai exact o găleată) care este alimentată cu un debit variabil de apă. Această găleată are un orificiu în partea inferioară pe unde este lăsată sa curgă cu un debit constant cantitatea de apă din interiorul recipientului. Inițial, recipientul nu este plin, deci vom considera că nu are loc starea de congestie.

Analog, vom considera un buffer la intrarea căruia există un flux variabil de pachete, iar la ieșire bufferul se descarcă cu viteză constantă (de obicei, mai mică decât viteza de alimentare a bufferului), în momentul în care în buffer este atinsă capacitatea maximă de stocare.

Astfel vom presupune că, pachetele de intrare generate de nodul sursă sunt generate cu rata de 10Mbps la fiecare 2 secunde, și ieșirea transmite cu 10 Mbps.Acest fenomen cauzează o discontinuitate al fluxului de date la ieșire. Această problemă poate fi rezolvată cu algoritmul “**Leaky Bucket**” care modifică ieșirea, astfel încât vom avea un flux continuu de informație transmis cu 2 Mbps pe o durată de 10 secunde.

Deci , ceea ce algoritmul “**Leaky Bucket**” face, este să formateze traficul în rafală de la intrarea nodului în trafic cu rată fixă, având dezavantajul că pachetele sunt îndepărtate atunci când recipientul se umple.

Un alt dezavantaj al acestui algoritm este că , în orice moment al transferului de date, rețeaua este încetinită chiar dacă nu apare fenomenul de congestie. Acest algoritm reduce traficul în loc să controleze pachetele în aceeași manieră.Această limitare a algoritmului

poate fi compensată folosind algoritmul găleții cu jeton(„**Token Bucket Algorithm”**).

**Probleme care utilizează algoritmul „Leaky Bucket”**

1. O găleată găurită se găsește în interfața rețelei unui host. Rata de transfer a informației este 2 Mbytes/s iar rata de transfer în găleată este de 2.5Mbytes/s
2. Presupunând că gazda are 250 Mbytes de trimis si trimite datele într-o singură rafală, care ar trebui să fie capacitatea minimă a găleții cu condiția ca nicio dată să nu fie pierdută.

**R**:Timpul în care sunt trimise datele : 250 Mbytes/2.5Mbytes=100s;

Timp de 100 secunde, din galeată se vor scurge:

100s\*2 Mbytes/s =200 Mbytes

 Deci, capacitatea minimă necesară găleții pentru a nu pierde apa pe la margini este de:250MB-200MB=50MB

 b)Presupnand că găleata are capacitatea de 100MBytes, care este cea mai lunga rafală pe care o poate trimite gazda sursa pentru a nu pierde apa?

**R**:Diferența între cele două rate de transfer este:2.5MB/s-2MB/s=0.5MB/s, ceea ce inseamna ca găleata se umple cu 0.5MB/s când ambele linkuri transmit la capacitate maximă. Deci, capacitatea de 100 MB va fi umplută după o rafală de 100MB/0.5MB/s=200s;

****

**Figura 2.2** Schema Algoritmului „**Leaky Bucket”**(<http://sun.iwu.edu/~jhaefner/CS390/Lecture11/lec11.html> )

**2.4.Algoritmul găleții cu jeton(“Token Bucket”)**

În algoritmul găleții cu jeton păstrăm aceeași presupunere ca în cazul găleții găurite cu excepția faptului că în cazul găleții cu jeton sunt adăugate la interval generat de un counter câte un jeton în găleată. În momentul în care în “galeată” sosește un pachet , se verifcă dacă există un token în găleată și doar atunci pachetele sunt lăsate să părăsească găleata(bufferul) cu aceeași rată cu care au sosit.

După ce toate jetoanele au fost epuizate, pachetele vor fi introduse în buffer cu rata de 1 token/tick (tick dat de counter).

Vom considera același exemplu ca în cazul **„Leaky Bucket”**:

 Presupunem că pachetele de intrare generate de nodul sursă sunt generate cu rata de 10Mbps la fiecare 2 secunde, și ieșirea transmite cu 10 Mbps.Acest fenomen cauzează o discontinuitate al fluxului de date la ieșire, și apare inevitabil caracterul de rafală al traficului de intrare.Presupunând că au fost generate cinci jetoane pe parcursul emisiei, pentru un interval de cinci secunde , deci toate cele 5 pachete vor fi transmise mai departe în cinci secunde.

Pe de altă parte, de fiecare dată când fiecare pachet este transmis, counterul se decrementează. Cu alte cuvinte, decrementarea counter-ului se face pentru fiecare pachet care părăsește bufferul, iar verificarea counterului se face la fiecare perioadă de ceas astfel încât să fie furnizată valoarea corectă a acestuia pentru implementarea algoritmului **„Token Bucket”** fie la nivel hardware, fie software de către utilizator.

Se observă că algoritmul “**Token Bucket”** poate stoca un număr maxim de N jetoane, astfel permițând trimiterea în rafală a N pachete simultan micșorând timpul de răspuns.

Se poate notifica faptul că timpul de introducere a pachetelelor în rețea este mai mic pentru **“Token Bucket”** decât pentru **“Leaky Bucket”**, asigurându-se un transfer mai mare în cazul “**Token Bucket”**.

 

**Figura 2.3.** Schema Algoritmului **“Token Bucket”(** <http://ecomputernotes.com/computernetworkingnotes/communication-networks/what-is-congestion-control-describe-the-congestion-control-algorithm-commonly-used> )

**Capitolul 3:Concluzii și considerații personale**

**1)**Atât algoritmul „**Leaky Bucket”** cât și algoritmul „**Token Bucket”** pot fi implementați de către sistemul de operare dar și de o interfață de rețea ce este conectată la rețea cu ajutorul unui counter care este inițializat cu 0 la început și apoi este incrementat la fiecare tick.

**2)** **Ce este o congestie? De ce apar congestiile?**

Într-o rețea cu comutare de pachete, pachetele sunt introduse în noduri, iar nodurile dirijează pachetele în rețea.În cazul în care "sarcina oferită" depășește o anumită limită, atunci există o scădere bruscă a traficului. Acest fenomen se numeste congestie.În fiecare nod de comutare de pachete, cozile sunt păstrate si au rolul primească și să transmită pachete (store/foreward network). Datorita naturii de rafală a traficul din rețea pot exista situatii in care cozile sunt depășite. Ca urmare, vor fi retransmise mai multe pachete, fapt care crește și mai mult traficul în rețea. Acest lucru duce în cele din urmă la congestionare.

**3)Cum se realizeaza controlul congestiei prin algoritmul găleții găurite?**

În algoritmul găletii găurite, se introduce un mecanism de „buffering” între calculatorul gazda și rețea pentru a reglementa fluxul de trafic.Traficul in rafale este generat de calculatorul gazdă și este introdus în rețeaua de mecanismul găletii gaurite in felul urmator:

- Pachetele sunt introduse în rețea 1/tick

 - În caz de „buffer overflow” pachetele ce nu au loc in „galeata” sunt eliminate

 - Pachetele care părăsesc calculatorul gazdă fac parte dintr-un șir constant eliminându-se caracterul de rafală inițial.

 **4)În ce fel algoritm găleată cu jeton este superior algoritmului găleata gaurită?**

Algoritmul cu galeată găurită controlează viteza cu care pachetele sunt introduse în rețea, dar este foarte conservatoare. Algoritmul găleată cu jeton introduce mai multa flexibilitate. In cazul găleții cu jeton sunt generate jetoane la fiecare tick.Pentru ca un pachet de intrare să fie transmis, acesta trebuie să captureze un jeton și transmiterea are loc în același ritm. Prin urmare, unele dintre pachete venite in rafală sunt transmise la aceeași rată dacă token-uri sunt disponibile. Deci ,este introdusa o anumită cantitate de flexibilitate în sistem. Acest lucru îmbunătățește, de asemenea,performanțele sistemului.

**Capitolul 4:Bibliografie**

1. [**http://www.cs.toronto.edu/~marbach/COURSES/CSC358\_S14/congestion.pdf**](http://www.cs.toronto.edu/~marbach/COURSES/CSC358_S14/congestion.pdf)
2. [**http://textofvideo.nptel.iitm.ac.in/106105082/lec22.pdf**](http://textofvideo.nptel.iitm.ac.in/106105082/lec22.pdf)
3. [**http://en.wikipedia.org/wiki/Network\_congestion#Congestion\_control**](http://en.wikipedia.org/wiki/Network_congestion#Congestion_control)
4. [**http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4362498**](http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4362498)
5. [**http://www.cisco.com/c/en/us/products/collateral/unified-communications/rsvp-agent/solution\_overview\_c22-521818.html**](http://www.cisco.com/c/en/us/products/collateral/unified-communications/rsvp-agent/solution_overview_c22-521818.html)
6. [**http://ecomputernotes.com/computernetworkingnotes/communication-networks/what-is-congestion-control-describe-the-congestion-control-algorithm-commonly-used**](http://ecomputernotes.com/computernetworkingnotes/communication-networks/what-is-congestion-control-describe-the-congestion-control-algorithm-commonly-used)

**7)**  **C. W. Leong, W. Zhuang, Y. Cheng, and L. Wang, “Call admission**

**control for integrated on/off voice and best-effort data services in mobile**

**celluar communications”**