Universitatea “Politehnica” din Bucureşti Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Algoritmi de combatere a congestiei prin metode ECN

Proiect de Diplomă

prezentat ca cerinţă parţială pentru obţinerea titlului de
Inginer în domeniul *Calculatoare şi Tehnologia Informaţiei*
programul de studii de licenţă *Ingineria Informaţiei*

Conducător ştiinţific
Conf. dr. ing. Ştefan Stăncescu

Absolvent
Ancuța Ionuț-Alexandru

2015

ACRONYMS

ACK Acknowledgment
AQM Active Queue Management
BDP Bandwidth Delay Product
CPU central processing unit
CRC cyclic redundancy check
CWND congestion window size
DCE Distributed Computing Environment
DNS Domain Name System
ECN Explicit Congestion Notification
FTP File Transfer Protocol
ICMP Internet Control Message Protocol
IEEE Institute of Electrical and Electronics Engineers
IP Internet Protocol
FTP File Transfer Protocol
LAN local area network
LCP link control protocol
HTTP Hypertext Transfer Protocol
MSL maximum segment lifetime
MSS maximum segment size
NS2 Network Simulator 2
OSI open systems interconnection
PDU protocol data unit
PPP Point-to-Point Protocol
PSH push flag, TCP header
RED Random Early Detection
RFC Request for Comment
RTT round-trip time
SMTP Simple Mail Transfer Protocol
SMS Systems Management Server
SSTHRESH Slow-start threshhold Value
SACK selective acknowledgment
SYN synchronize packet
TCP Transmission Control Protocol
TTL time-to-live
UDP User Datagram Protocol
VOIP Voice over Internet Protocol.
QoS Quality of Service
WWW World Wide Web

Universitatea "Politehnică" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul Electronică Aplicată şi Ingineria Informaţiei

Aprobat Director de Departament :

Prof. Dr. Ing. Sever Paşca

TEMA PROIECTULUI DE DIPLOMĂ
a studentului Ancuța I. Ionuț-Alexandru,441A

1. Titlul temei: **Algoritmi de combatere a congestiei prin metode ECN**

2. Contribuţia practică, originală a studentului va consta în (*în afara* părţii de documentare): *se vor simula algoritmi de combatere a congestiei folosind limbajul TCL al simulatorului Network Simulator pentru câteva configurații concludente de rețea;se vor estima performanțele procedurilor cunoscute de combatere a congestiei cu metode ECN;*
-se va optimiza o procedură ECN (Explicit Congestion Notification) de tratare a congestiei;
-se va simula modificarea prin introducerea sa in sursele simulatorului Network Simulator care se va recompila;
-se vor simula strategii de combatere a congestiei cu metoda modificată si se vor măsura performanțele;
-se va elabora un capitol de concluzii care va justifica eficiența modificării.

3. Proiectul se bazează pe cunoştinţe dobândite în principal la următoarele 3-4 discipline:
Rețele de calculatoare,
Sisteme de operare,
Arhitecturi de rețea si internet.

4. Proprietatea intelectuală asupra proiectului aparţine: *UPB*

5. Locul de desfăşurare a activităţii: *UPB*

6. Realizarea practică rămâne în proprietatea: *UPB/ETTI*

7. Data eliberării temei: 12 decembrie 2014

CONDUCĂTOR LUCRARE: *Conf. Dr. Ing. Ștefan Stăncescu*

STUDENT: *Ionuț-Alexandru Ancuța*

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul "*Algoritmi de combatere a congestiei prin metode ECN*", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul Calculatoare și Tehnologia Informației, programul de studii *Ingineria Informației* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București,

29.06.2015 *Ionuț-Alexandru Ancuța*

Cuprins

Listă de figuri

Introducere

În această lucrare se propune studiul algoritmilor de combatere a congestiei prin așa-numitele metode ECN. Se va studia comportamentul algoritmilor si principiile pe care aceştia se bazează, urmând o discuție despre cât de eficientă și ușor de implementat este această metodă.

Am ales algoritmii de combatere a congestiei prin metode ECN deoarece dețin avantajul de a semnala, controla și combate congestia la nivelul transportului de pachete funcționând totodată cu actualele protocoale de transport TCP/IP. Pentru a putea fi implementate aceste metode avem nevoie și de echipamente in rețea performante, dar trebuie ca fiecare echipament din rețea să aibă implementate mecanismele de suport al acestor metode.

Se va folosi simulatorul Network Simulator 3 și se vor simula anumiți algoritmi de combatere a congestiei pentru câteva configurații concludente de rețea, iar apoi se va optimiza o procedură ECN de tratare a congestiei ce se va recompila in sursele simulatorului. Pe baza optimizării se vor face comparații în care se va justifica eficiența modificării aduse.

Deoarece metodele ECN implică automat utilizarea nivelului TCP/IP se va face o descriere mai pe larg a nivelului de transport pentru a exemplifica utilizarea ECN în cadrul lucrării prezentate.

1. Clasificarea rețelelor de calculatoare

Rețelele de calculatoare s-au dezvoltat foarte mult în ultimul timp datorită faptului că au apărut în ultimul timp tehnologii software, hardware și de interconectare de ultimă generație. Aceste tehnologii de mare viteză au făcut posibilă utilizarea rețelelor de calculatoare în toate domeniile și cu rezultate impresionante.

O rețea de calculatoare desemnează practic o multitudine de calculatoare interconectate folosind o singură tehnologie. Interconectarea a două calculatoare este realizată atunci când acestea sunt capabile să schimbe informație între ele. Conectarea propriu-zisă poate fi realizată prin cablu, prin radiații, microunde sau prin sateliți de comunicare.

Rețelele pot fi clasificate după tipul mediului de transmisie în:

- Rețele cu fir – conectarea între componentele unei rețele se face prin cabluri de cupru sau prin fibră optică;
- Rețele wireless – la care mediul de transmisie este reprezentat de aer.

După suprafața pe care o ocupă:

- Rețele locale (LAN- Local Area Network) – reprezintă o rețea limitată la o anumită suprafață geografică. Ca exemplu putem da o rețea dintr-o întreprindere sau dintr-un campus universitar;
- Rețele metropolitane (MAN- Metropolitan Area Network) – o rețea ce acoperă aproximativ suprafața unei metropole cuprinsă ca dimensiuni între un LAN și un WAN;
- Rețele extinse (WAN – Wide Area Network) – reprezintă o rețea alcătuită din mai multe LAN-uri interconectate. Cuprinde suprafețe geografice foarte mari și poate utiliza ca legături pentru WAN-uri tehnologia satelit pentru conectarea calculatoarelor din țări sau continente diferite.

După proiectarea fizică a rețelei:

- Rețele bus – calculatoarele sunt aranjate astfel încât cablurile trec de la un calculator la altul în mod liniar;
- Rețele stea – toate calculatoarele sunt legate la un hub situat în mijlocul rețelei;
- Rețele inel (ring) – rețea în care ultima conexiune se întoarce la prima și formează un inel;
- Rețele full mesh – sunt rețele în care fiecare nod este legat la toate celelalte noduri din rețea astfel formând o plasă.

După metodele de acces la mediul de transmisie:

- Metode statice: - TDM (Time Division Multiplexing)
  - FDM (Frequency Division Multiplexing)
  - WDM (Wavelength Division Multiplexing)
- Metode dinamice: - deterministe (token passing);
  -nedeterministe (Ethernet).

După tipul sistemului de operare de rețea:

- TCP/IP;
- IPX/SPX;
- Apple Talk. [1]

1.1 Arhitectura OSI

Modelul de referință OSI (Open System Interconnection) este o stivă ierarhică de protocoale de comunicație foarte des folosită pentru a realiza o rețea de calculatoare. OSI reprezintă un standard al Organizației internaționale de standardizare ce a fost emis în anul 1984.

Acest model oferă metode generale pentru realizarea comunicației, sistemelor de calcul pentru ca acestea să poată schimba informații, indiferent de particularitățile constructive ale sistemelor. Modelul de referință are aplicații în toate domeniile comunicațiilor de date, nu doar in cazul rețelelor de calculatoare.

Modelul OSI divizează problema complexă a comunicării între două sau mai multe sisteme în 7 straturi numite și niveluri distincte (layers), într-o arhitectură ierarhică. Fiecare nivel are funcții bine determinate și comunică doar cu nivelele adiacente. Principiile folosite pentru a se ajunge la cele șapte niveluri sunt următoarele:

1. Un nivel trebuie creat atunci când este nevoie de un nivel de abstractizare diferit.
2. Fiecare nivel trebuie să îndeplinească un rol bine definit.
3. Funcția ficărui nivel trebuie aleasă acordându-se atenție definirii de protocoale standardizate pe plan intenațional.
4. Delimitarea nivelurilor trebuie facută astfel încât să se minimizeze fluxul de informații prin interfețe.
5. Numărul lor trebuie să fie suficient de mare pentru a nu fi nevoie să se introducă în același nivel funcții diferite și suficient de mic pentru ca arhitectura să rămână funcțională.

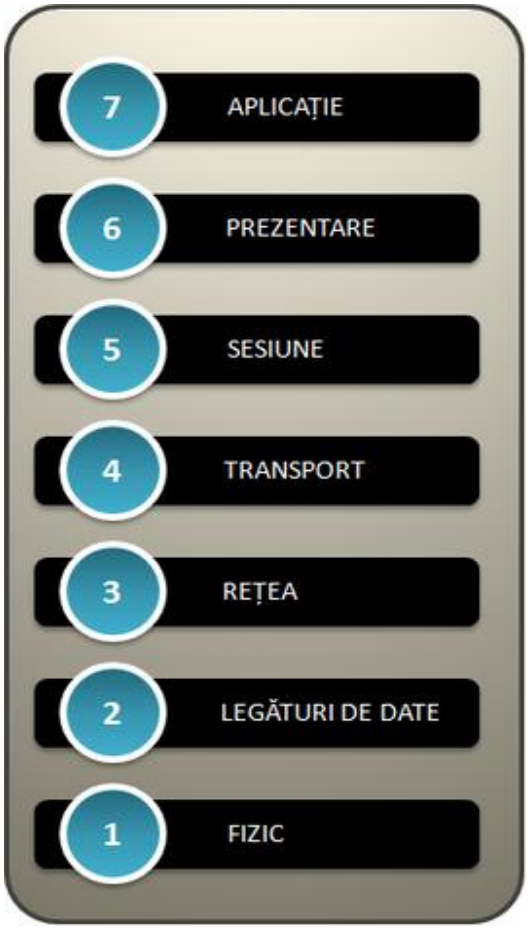

Figura 1.1-Stiva de protocoale după modelul OSI

Deși astăzi există și alte sisteme, cei mai mulți distribuitori de echipamente de comunicație folosesc OSI pentru a instrui utilizatorii în folosirea echipamentelor. Se consideră că OSI este cel mai bun mijloc prin care se poate face înțeles modul în care informația este trimisă și primită.

Nivelul APLICAȚIE

Este cel mai apropiat nivel de utilizator. Nivelul aplicație furnizează servicii de rețea pentru aplicațiile utilizatorului.Nu oferă servicii pentru nici un alt strat OSI.

Nivelul aplicație conține o varietate de protocoale frecvent utilizate: FTP, HTTP, Telnet, SMTP. Atunci când un browser accesează o pagină web, el trimite serverului numele paginii pe care dorește să o acceseze folosind HTTP.Serverul va trimite ca răspuns pagina. Alte protocoale de aplicație sunt folosite pentru transferul fișierelor, poștă electronică.

Nivelul PREZENTARE

Nivelul prezentare se asigură că informația trimisă de nivelul aplicație dintr-un sistem este descifrabilă pentru nivelul aplicație al altui sistem. Acesta se ocupă de probleme legate de formatul datelor: compresie, criptare translația protocoalelor. Nivelul prezentarese ocupă de sintaxa și semantica informațiilor transmise.

Nivelul SESIUNE

Nivelul sesiune stabilește, administrează și termină sesiunile între două sisteme care comunică între ele. Sincronizează mesajele dintre cele două nivele de prezentare aparținând sistemelor care comunică între ele și administrează schimbul de date între acestea. O sesiune se poate utiliza pentru ca un utilizator să se poată conecta la distanță sau pentru transferul unui fițier între două mașini.

Nivelul TRANSPORT

Are ca rol segmentarea datelor în sistemul gazdă – transmițător și reasamblarea acestora în sistemul gazdă receptor. Segmentarea se realizează tocmai pentru o trimitere mai eficientă a datelor deoarece în caz că apare o coliziune în rețea, iar datele ar fi pierdute, să fie retransmis doar segmentul ce nu a ajuns la destinație și nu întregul pachet de date. Acest nivel ajutat și de protocolul TCP se asigură ca toate segmentele ajung la destinație și că acestea sunt și reasamblate.

Legătura dintre nivelul Transport și nivelul Sesiune poate fi vazută ca o legătură între protocoalele aplicației și protocoalele flux de date astfel: nivelurile 7,6,5 se ocupă de problemele aplicației în timp ce 4,3,2,1 de problema transportului de date.

Nivelul REȚEA

Acesta furnizează prin adresare logică conectivitatea și selectarea căii între două sisteme gazdă ce pot aparține și unor rețele distincte. Atunci când un pachet trebuie să parcurgă un traseu dintr-o rețea în alta modul de adresare folosit de una din rețele poate fi diferit de celălalt mod al

celei de-a doua rețele astfel că transferul datelor la nivelul rețea se realizează prin 4 procese de bază: - adresare (trebuie să prezinte un caracter de unicitate);

-încapsularea datelor primite de la nivelul transport în pachete;

-direcționarea pachetelor prin rețea;

-decapsularea pachetelor.

Ca și echipamente utilizate la nivel de rețea folosim ruterele, iar ca protocoale de rețea IPv4, IPv6, IPX și AppleTalk.

Nivel Legătură de date

Nivelul legătură de date se ocupă cu adresarea fizică, topologia rețelei, accesul la rețea, detecția și anunțarea erorilor și controlul fluxului fizic. Rolul nivelului legăturii de date este de a marca și de a recunoaște delimitatorii amplasați între cadre de biți.

O problemă frecventă ce apare la nivelul legăturii de date este aceea de a supraîncărca un receptor lent de către un emițător mai rapid. Această problemă poate fi tratată prin anumite mecanisme de reglare a traficului prin determinarea spațiului liber în coada de recepție de către emițător.

Echipamentele ce sunt utilizate la nivelul legăturii de date pot fi placa de rețea, switch, ethernet.

Nivelul Fizic

Are ca rol transmiterea biților pe un canal de comunicație, iar ca scop de a crea semnale electrice, optice și electromagnetice pentru reprezentarea biților din fiecare cadru și transmiterea acestora pe mediul fizic.

Nivelul fizic definește specificații electrice, mecanice, procedurale și funcționale pentru activarea, întreținerea și dezactivarea legăturii fizice între terminale.

Echipamentele reprezentative nivelului fizic dintr-o rețea pot fi huburile și firele sau cablurile (coaxial, STP, UTP sau fibra optică), iar în rețelele wireless mediu de transmisiune aerul.[16]

1.2 Modelul TCP/IP

Modelul TCP/IP este cel mai utilizat protocol folosit în rețelele locale cât și pe Internet datorită disponibilității și flexibilității lui, având cel mai mare grad de corecție al erorilor. Acesta permite comunicarea între calculatoarele din întreaga lume indiferent de sistemul de operare instalat.

Bazele acestuia au fost puse la începutul anilor 1960 când guvernul SUA a finanțat proiectarea și dezvoltarea protocolului TCP/IP.Se dorea ca atâta timp cât conexiunea fizică dintre calculatoarele din rețea era funcțională să fie funcțională și cea logică chiar dacă acestea se opresc brusc. Era nevoie de o arhitectură flexibilă, mergând de la transferul de fișiere până la transmiterea vorbirii în timp real.

Crearea protocolului a rezolvat multe probleme dificile din acea vreme, devenind modelul standard pe care este bazat Internetul. La început a fost folosit pentru rețelele militare, apoi a fost furnizat și agențiilor guvernamentale, universităților, iar într-un final să fie folosit de publicul larg.

Protocolul TCP/IP este alcătuit din patru niveluri:

-Aplicație;
-Transport;
-Rețea;
-Acces la rețea.

Modelul TCP/IP este asemănător cu modelul OSI, dar nu trebuie făcută confuzia numelor nivelurilor TCP/IP cu numele nivelurilor din OSI.

Nivelul Aplicaţie

Nivelul aplicaţie se ocupă cu protocoalele de nivel înalt, codificarea şi controlul dialogului, împachetarea datelor şi trimiterea lor la următoarele niveluri. El este echivalentul nivelurilor Aplicaţie, Prezentare şi Sesiune din modelul OSI.

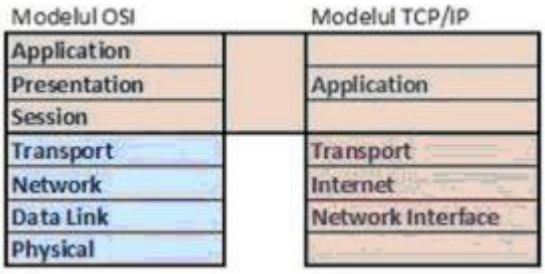

Figura 1.2-Nivelul Aplicaţie(TCP/IP)

Nivelul Aplicaţie conţine următoarele protocoale de nivel înalt:

- Transfer de fişiere : TFTP, FTP şi NFS;
- E-mail: SMTP;
- Remote: telnet, rlogin;
- Managementul de reţele: SNMP;
- Managementul de nume: DNS;
- HTTP.

Nivelul Transport

Nivelul Transport asigură conexiunea logică dintre sursă şi destinaţie, fluxul de date şi corecţia erorilor.

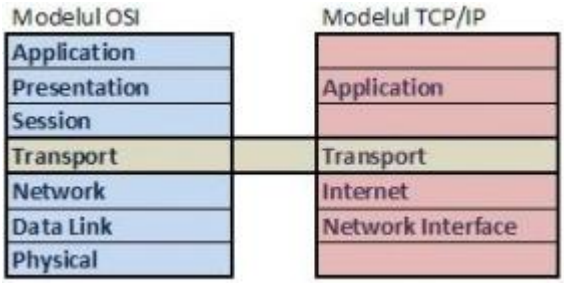

Figura 1.3-Nivelul Transport(TCP/IP)

Nivelul Transport include protocoalele TCP şi UDP. TCP(Transmission Control Protocol) este un protocol orientat pe conexiune care permite ca un flux de octeţi trimişi de la un calculator să ajungă fără erori pe orice alt calculator din Internet. Dacă la destinaţie un pachet ajunge cu erori, TCP cere retransmiterea pachetului.

TCP fragmentează fluxul de octeţi în mesaje discrete şi pasează fiecare mesaj nivelului Reţea. De asemenea, tratează controlul fluxului pentru a se asigura că sursa nu inundă destinaţia cu mai multe pachete decât poate aceasta prelucra. Toate acestea sunt realizate prin utilizarea secvenţelor de număr, sliding windows şi acknowledgments.

UDP(User Datagram Protocol) este un protocol nesigur, destinat pentru aplicaţii care trebuie să interogheze rapid, fără retrimiterea pachetelor eronate. UDP este folosit în aplicaţiile de transmisii video sau audio şi aplicaţii client-server.

Ca exemple de aplicaţii care folosesc protocolul UDP :

- DNS (Domain Name Server)
- TFTP (Trivial File Transfer Protocol)
- IPTV (TV prin Internet)
- VOIP (Telefonie prin Internet)

Nivelul Reţea

Scopul nivelului reţea este de a găsi cel mai optim traseu prin care poate trimite pachetele.Nivelul reţea corespunde cu acelaşi nivel reţea din modelul OSI.

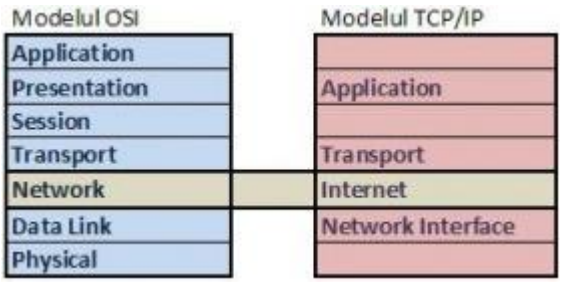

Figura 1.4-Nivelul Reţea(TCP/IP)

Protocoalele care lucrează la nivelul reţea din modelul TCP/IP sunt:

- IP (Internet Protocol)
- ICMP (Internet Control Message Protocol)
- ARP (Adress Resolution Protocol)
- RARP (Reverse Adress Resolution Protocol)

IP caută cea mai bună cale de a trimite pachetele. ICMP oferă capabilităţi de control şi în schimbul de mesaje. ARP determină adresa MAC pentru adresele IP. RARP determină adresa IP pentru o adresă MAC cunoscută.

Problemele majore se referă la dirijarea pachetelor şi evitarea congestiei în reţea. De aceea este ideal este ca nivelul reţea din TCP/IP să funcţioneze asemănător cu nivelul reţea din OSI.

Nivelul Acces la Reţea

Nivelul acces la reţea se ocupă cu toate problemele legate de transmiterea efectivă a unui pachet IP pe o legătură fizică, incluzând şi aspectele legate de tehnologii şi medii de transmisie, adică niveluri OSI Legătură de date şi Fizic. Modelul de referinţă TCP/IP nu spune mare lucru

despre ce se întâmplă acolo, însă menţionează că gazda trebuie să se lege la reţea, pentru a putea trimite pachete IP, folosind un anumit protocol. Acest protocol nu este definit şi variază de la gazdă la gazdă şi de la reţea la reţea.

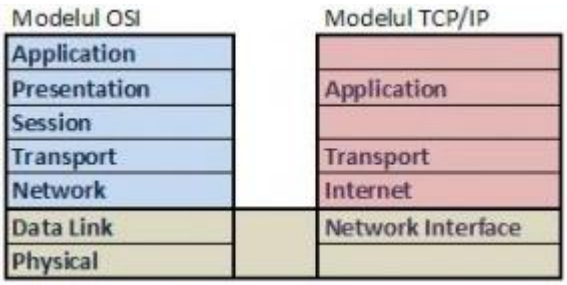

Figura 1.5-Nivel acces la reţea(TCP/IP)

Componentele ce fac parte din acest nivel pot fi drivere, modemuri, plăci de reţea etc. Nivelul de acces la reţea defineşte procedurile folosite pentru interogarea cu echipamente de reţea şi acces la mediu de transmisie. Protocolul standard, cum ar fi Serial Line Internet Protocol (SLIP) şi punct la punct (PPP) trebuie să asigure accesul la reţea prin intermediul unui modem de conectare. Multe protocoale sunt necesare pentru a determina elementele hardware şi software, precum şi specificaţiile de transmitere la acest nivel.

Ca o comparaţie alcătuită între modelul OSI şi modelul TCP/IP se poate observa că acestea sunt alcătuite aproximativ la fel fiind organizate pe niveluri chiar dacă unele dintre ele includ servicii diferite. Ca deosebiri se poate identifica din imaginile ataşate că protocolul TCP/IP combină în nivelurile sale Aplicaţie şi Acces la reţea mai multe niveluri din modelul OSI şi este oarecum mai simplu tocmai datorită faptului că acesta conţine mai puţine niveluri. În privinţa nivelului Transport, acesta este mai bun la nivelul modelului OSI, în privinţa livrării de pachete, în cazul în care se utilizează protocolul UDP în TCP/IP.

Ca şi o concluzie a acestor modele se poate afirma că Internetul are la bază standardele implementate de protocoalele TCP/IP şi că dezvoltarea sa este bazată într-o proporţie destul de mare datorită acestor protocoale. Modelul OSI nu este folosit în construcţia de reţele, dar are ca rol înţelegerea mai facilă a procesului de comunicare la nivel de reţea.[17]

2. Congestia

2.1 Definiţia congestiei

Din cauza faptului că apariţia congestiei se datorează supraîncărcării reţelelor, definiţia acesteia se concentrează foarte mult pe comportamentul reţelelor la încărcări mai sau foarte mari.

Această supraîncărcare apare atunci când se doreşte a fi transmis un volum mare de date pe o legătură cu o capacitate redusă. Ca şi definiţie putem spune că sarcina de a transmite un volum de date mai mare decât capacitatea legăturii pe care se va transmite duce la congestie. În prezent congestia este semnalată pe reţele prin detecţia pierderilor de pachete ca rezultat al aşteptării în coada ce depăşeşte o anumită limită. Algoritmul de control al congestiei specificat în TCP a rămas oarecum neschimbat semnalizarea pierderii de pachete şi eliminarea apariţiei unui colaps în reţea fiind îndeplinite de acest algoritm.

Ca exemplu se poate considera o reţea în care rata de sosire a pachetelor la un ruter depăşeşte rata de deservire a acestuia. În acest moment, pachetele sunt introduse într-o coadă de aşteptare, ceea ce produce întârzieri. Această întârziere poate determina sursele să retransmită pachetele astfel fiind încărcată şi mai mult reţeaua.Dacă se implementează un mecanism de control al traficului de la ruter la ruter, se poate ajunge în situaţia în care se poate inunda un ruter precedent şi astfel se poate îngheţa traficul.

Astfel se produc evenimente ce apar simultan şi care sunt întârzierea datorată cozilor de aşteptare care creşte, pierderea de pachete şi retransmisiile care duc la un trafic îngreunat cu o eficienţă scăzută. Pe baza acestor reguli apar diferite definiţii ale congestiei.

Aceste definiţii nu sunt satisfăcătoare din mai multe motive. În primul rând. Întârzierile şi pierderile de pachete sunt indicatori de performanţă ce sunt impropiu folosiţi drept indicatori pentru congestie, dat fiin că modificarea acestor indicatori se poate datora unor fenomene diferite de congestie. În al doilea rând, definiţia nu specifică în mod exact punctul de la care o reţea poate fi considerată congestionată. De exemplu, în timp ce o reţea care are timpi medii de întârziere în fiecare ruter de 1 până la 10 durate de deservire nu este cu siguranţă congestionată, nu este clar dacă o reţea ce are timpi medii de întârziere de 1000 durate de deservire este sau nu congestionată.

O reţea congestionată din punctul de vedere al unui utilizator nu este neapărat congestionată din punctul de vedere al altui utilizator. Dacă utilizatorul A poate tolera o rată de pierdere a pachetelor de 1 la 1000, iar utilizatorul B poate tolera o rată de pierdere de 1 la 100 şi rata de pierdere efectivă este de 1 la 500 atunci A va considera că reţeaua este congestionată, în timp ce pentru B reţeaua va funcţiona în parametri normali. O reţea trebuie considerată necongestionată numai dacă toţi utilizatorii săi sunt de acord cu această stare.

Astfel, pe baza celor prezentate se justifică presupunerea că starea de congestionare a unei reţele depinde de perspectiva utilizatorului. Un utilizator care cere puţin de la o reţea poate tolera o pierdere de performanţă mult mai bine decât un utilizator cu cerinţe mari. Spre exemplu, un utilizator care foloseşte reţeaua numai pentru mail va fi mulţumit cu o întârziere de o oră, pe când această stare nu este acceptabilă pentru un utilizator care foloseşte reţeaua pentru transmisiuni de voce în timp real.

Pentru a stabili o definiţie cu ajutorul celor prezentate se poate afirma că o reţea este congestionată din privinţa utilizatorului i dacă utilitatea lui I scade ca urmare a unei creşteri a încărcării reţelei.

Observaţii:

- O reţea poate fi congestionată din perspectiva unui utilizator şi necongestionată din perspectiva altuia.
- O reţea poate fi considerată strict necongestionată numai dacă nici un utilizator nu o percepe ca fiind congestionată.

O reţea care controlează congestia trebuie să fie sensibilă la funcţiile de utilitate ale utilizatorilor şi să fie capabilă să îşi direcţioneze resursele astfel încât să nu existe pierderi de utilitate odată cu creşterea încărcării. Aşadar, reţeaua trebuie să fie capabilă să facă diferenţa între conversaţii şi să prioritizeze conversaţiile în funcţie de stringenţa utilităţii participanţilor la o anumită conversaţie.

Congestia poate fi produsă de mai mulţi factori. Dacă dintr-o dată încep să sosească şiruri de pachete pe trei sau patru linii de intrare şi toate necesită aceiaşi linie de ieşire, 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, dar Nagle a descoperit în 1987 că dacă ruterele ar avea o cantitate infinită de memorie, congestia s-ar înrăutăţi în loc să se amelioreze, deoarece până să ajungă la începutul cozii pachetele au fost deja considerate pierdute (timeout repetat) şi s-au trimis duplicate. Toate aceste pachete vor fi trimise cu conştiinciozitate către următorul ruter, crescând încărcarea de-a lungul căii către destinaţie.

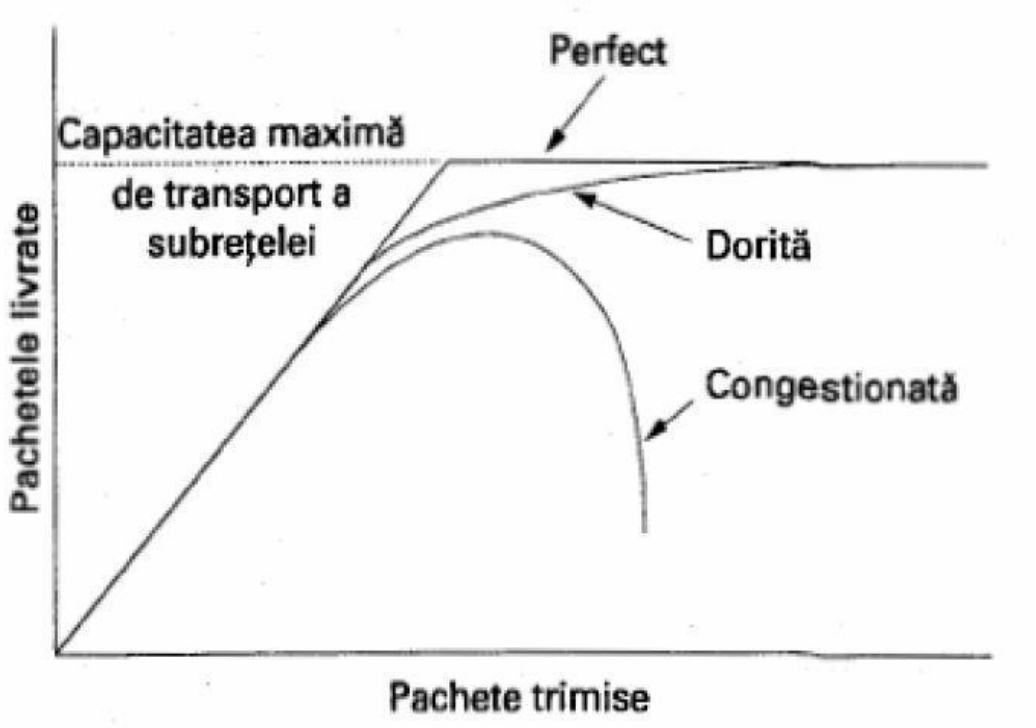

Figura 2.1-Simptomele unei reţele congestionate

În figură se scoate în evidenţă faptul că dacă traficul este prea intens, apare congestia, iar în consecinţă performanţele reţelei scad substanţial.

O altă cauză sunt procesoarele lente care pot cauza congestia. Dacă unitatea centrală a ruterului (CPU) este lentă în execuţia funcţiilor sale (introducerea în cozi, actualizarea tabelelor etc.), se pot forma cozi, chiar dacă linia de comunicaţie nu este folosită la capacitate maximă. Similar şi liniile cu lăţime de bandă scăzută pot provoca congestia. Schimbarea liniilor cu unele mai performante şi păstrarea aceluiaşi procesor sau invers de obicei ajută puţin, însă de cele mai multe ori nu fac decât să deplaseze punctul critic. Adevărata problemă este de multe ori o incompatibilitate între părţi ale sistemului. Ea va persista până ce toate componentele sunt în echilibru.

Este important de subliniat diferenţa dintre controlul congestiei şi controlul fluxului, deoarece relaţia este subtilă.

Controlul congestiei trebuie să asigure că subreţeaua este capabilă să transporte întreg traficul implicat. Este o problemă globală, implicând comportamentul tuturor calculatoarelor gazdă, al tuturor ruterelor, prelucrarea de tip store-and-forward din rutere şi toţi ceilalţi factori care tind să diminueze capacitatea de transport a subreţelei.

Controlul fluxului, prin contrast, se referă la traficul capăt la capăt între un expeditor şi un destinatar. Rolul său este de a împiedica un expeditor rapid să trimită date continuu, la o viteză mai mare decât cea cu care destinatarul poate consuma datele. Controlul fluxului implică frecvent existenţa unui feed-back de la receptor către emiţător, pentru a spune emiţătorului cum se desfăşoară lucrurile la celălalt capăt.

Pentru a vedea diferenţa dintre aceste două concepte, să considerăm o reţea cu fibre optice cu o capacitate de 1000 Gbps pe care un calculator încearcă să transfere un fişier către un calculator personal la 1 Gbps. Deşi nu există nici o congestie (reţeaua nu are nici un fel de probleme), controlul fluxului este necesar pentru a forţa supercalculatorul să se oprească des, pentru a permite calculatorului personal să primească datele.

La cealaltă extremă, să considerăm o reţea de tip store-and-forward cu linii de 1 Mbps şi 1000 de calculatoare mari, din care jumătate încearcă să transfere la 100 kbps către cealaltă jumătate. Aici problema nu apare datorită unui emiţător rapid care surclasează un receptor lent, ci pentru că traficul cerut depăşeşte posibilităţile reţelei. Motivul pentru care controlul congestiei şi controlul fluxului sunt adesea confundate este acela că unii algoritmi pentru controlul congestiei funcţionează trimiţând mesaje înapoi către diferitele surse, spunându-le să încetinească atunci când reţeaua are probleme. Astfel, un calculator gazdă poate primi un mesaj de încetinire fie din cauză că receptorul nu suportă încărcarea, fie pentru că reţeaua este depăşită.[1]

2.2 Principii generale ale controlului congestiei

Controlul congestiei este un proces foarte complex care nu şi-a găsit o rezolvare cât de cât mulţumitoare nici în ziua de azi. Chiar dimpotrivă, acest subiect este din ce în ce mai des abordat în cercetările din informatică din cauza aglomerării din ce în ce mai mări a reţelelor de calculatoare. Putem extinde problema congestiei chiar la nivelul general al unei reţele de telecomunicaţii. Aspectul care complică şi mai mult mecanismele de control al congestiei este acela că nu se poate localiza pe un nivel particular al stivei de protocoale. Algoritmii de control al congestiei pot fi implementaţi atât la nivelul routerelor de reţea cât şi la nivelul protocoalelor de transport implementate în softul de reţea al dispozitivelor comunicante. O soluţie exclusivă, bazată doar pe controlul unui singur nivel nu duce la o rezolvare mulţumitoare. De aceea se caută soluţii de compromis între cele două niveluri.

Multe din problemele care apar în sistemele complexe, cum ar fi reţelele de calculatoare, pot fi privite din punctul de vedere al unei teorii a controlului. Această abordare conduce la împărţirea tuturor soluţiilor în două grupe: în buclă deschisă şi în buclă închisă. Soluţiile în buclă deschisă încearcă să rezolve problema printr-o proiectare atentă, în esenţă să se asigure că problema nu apare. După ce sistemul este pornit şi funcţionează, nu se mai fac nici un fel de corecţii.

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 de 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, valorile crescătoare 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 ocoli zonele cu probleme. Unele staţii de radio au elicoptere care zboară deasupra oraşelor pentru a raporta congestiile de pe drumuri şi a permite ascultătorilor mobili să îşi dirijeze pachetele (maşinile) astfel încât să ocolească zonele "fierbinţi".

În toate schemele cu feedback se speră că informarea asupra producerii congestiei va determina calculatoarele gazdă să ia măsurile necesare pentru a reduce congestia. Pentru ca o schemă să funcţioneze corect, duratele trebuie reglate foarte atent. Dacă un ruter strigă STOP de fiecare dată când sosesc două pachete succesive şi PLEACĂ (eng.: GO) de fiecare dată când este liber mai mult de 20 μsec, atunci sistemul va oscila puternic şi nu va converge niciodată. Pe de altă parte, dacă va aşteaptă 30 de 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 celor mai potrivite constante de timp 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 împărţi algoritmii în cei în buclă deschisă şi cei în buclă închisă, aşa cum s-a precizat anterior. În continuare împart algoritmii cu buclă deschisă în unii care acţionează asupra sursei şi alţii care acţionează asupra destinaţiei. Algoritmii în buclă închisă 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 cea pe care o pot suporta resursele (unei părţi a sistemului). 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 sistemele bazate pe sateliţi, creşterea puterii de transmisie asigură de regulă creşterea lăţimii de bandă. Împarţirea 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.

Totuşi, 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 refuzul servirii anumitor utilizatori, degradarea serviciilor pentru o parte sau pentru toţi utilizatorii şi planificarea cererilor utilizatorilor într-o manieră mai previzibilă.

O schemă de control al congestiei trebuie să îndeplinească anumite cerinţe fundamentale. Acestea sunt:

1) Eficienţă
2) Heterogenitate
3) Capacitatea de a funcţiona cu surse defecte
4) Stabilitate
5) Scalabilitate 10

6) Simplitate
7) Echitate

Eficienţa

Sunt două aspecte de menţionat în ceea ce priveşte eficienţa unei scheme de control a congestiei. În primul rând trebuie să se ţină cont de cantitatea de trafic suplimentar pe care schema de control a congestiei o aduce în reţea. Am dori desigur că o asemenea schemă să presupună o povară minimă pentru reţea. În al doilea rând trebuie să ne gândim dacă schema de control nu conduce la o subutilizare a resurselor critice ale reţelei. Scheme de control ineficiente pot "gâtui" sursele chiar dacă nu există pericol de congestie, conducând la o subutilizare a resurselor. Nu se doreşte să se lucreze suboptimal într-o reţea.

Heterogenitatea

Odată cu creşterea dimensiunilor reţelelor cât şi a acoperirii lor, acestea tind să înglobeze o varietate din ce în ce mai mare de arhitecturi hardware şi software. O schemă de control care presupune o singură dimensiune de pachet, un singur tip de protocol la nivelul transport sau un singur tip de serviciu nu poate funcţiona în regulă într-un aşa mediu. Aşadar, ne dorim o schemă de control implementabilă peste o largă varietate de arhitecturi de reţea.

Capacitatea de a funcţiona cu surse defecte

În reţelele actuale, administratorul de reţea nu deţine mijloace administrative de control asupra surselor de mesaje. Deci, sursele (de exemplu, utilizatorii de PC-uri) sunt liberi să

manipuleze protocoalele de reţea pentru a maximiza utilitatea obţinută de la reţea. Atunci când un ruter informează o sursă să-şi reducă rata de trimitere, o sursă funcţională va face acest lucru. Totuşi, este posibil că o sursă „defectă" să ignore aceste semnale, în speranţa că acest lucru îi va permite să trimită mai multe date. O schemă de control a congestiei nu trebuie să eşueze în prezenţa unor asemenea surse (o modalitate este "pedepsirea" surselor defecte).

Stabilitatea

Controlul congestiei poate fi privit că o problemă clasică de feed-back negativ. Complexitatea suplimentară se datorează faptului că semnalele de control sunt întârziate. Cu alte cuvinte, există o întârziere finită între momentul detectării congestiei şi momentul recepţionării acestui semnal de către sursă. Mai mult, sistemul prezintă zgomot iar observările parametrilor sistemului pot fi corupte. Această complexitate poate induce instabilitate în reţea. Astfel, vrem că schema de control să fie robustă şi dacă este posibil, dovedit stabilă. 11

Scalabilitatea

Este însuşi natura sistemelor distribuite să crească odată cu trecerea timpului. Cheia succesului într-un asemenea mediu este scalabilitatea. O schemă de control performantă a congestiei ar trebui să fie scalabilă în ceea ce priveşte doi parametri: lăţimea de bandă şi dimensiunea reţelei.

Transmisiunile pe fibră optică au făcut posibilă dezvolarea de reţele cu lăţimi de bandă cu trei ordine de mărime mai mari decât predecesorii lor nu foarte îndepărtaţi (45000 kbps faţă de 56 kbps). În altă ordine de idei, sute de mii de reţele locale au fost interconectate într-o enormă inter-reţea ce acoperă întregul glob. Această expansiune face că scalabilitatea unei scheme de control a congestiei să fie imperativă.

Simplitatea

Simplitatea este întotdeauna un bun de preţ. Protocoale mai simple sunt mai uşor de implementat şi pot suporta mai uşor creşterile de lăţime de bandă. De asemenea, protocoalele simple sunt mai probabil acceptate că standarde. Aşadar, o schemă de control al congestiei ideală trebuie să fie uşor de implementat.

Echitatea

Poate fi necesar că anumite surse să-şi reducă transmisiile datorită congestiei. Modalitatea de a alege care surse vor fi restricţionate (fie cerându-le să-şi reducă transmisiile, fie renunţând la unele dintre pachetele trimise de acestea) determină cât de echitabil îşi alocă reţeaua resursele. De exemplu, dacă o cerere excesivă din partea sursei A „sufocă" sursa B, este clar că reţeaua nu îl tratează echitabil pe B. Nu este de dorit o schemă de control care să genereze o alocare neechitabilă a resurselor.

Când se vorbeşte despre echitate, sunt două aspecte de care trebuie ţinut cont: definirea şi implementarea. Numeroase criterii de definire a echităţii au fost propuse de-a lungul timpului dar nici unul nu este absolut. Implementarea unui crieteriu de echitate generează probleme care uneori depăşesc domeniul controlului congestiei. De exemplu, s-ar putea decide că este echitabil să se avantajeze anumite surse care sunt dispuse să plătească pentru acest lucru sau care trimit un anumit tip de pachete (mai importante - email).[1]

2.3 Politici pentru prevenirea congestiei

Voi începe studiul asupra metodelor de control al 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.

Prevenirea congestiei se bazează pe un set de mecanisme ce ajută cozile echipamentelor să evite congestia. Cozile se supraîncărca atunci când traficul sosit pe interfeţele de intrare depăşeşte capacitatea interfeţelor de ieşire. Atunci când mai mult trafic soseşte pe interfeţele de ieşire decât acestea pot suporta, cozile (memoriile && bufferele) routerelor se încarcă . Mecanismele de management al cozilor ne pot ajuta să evităm supraîncărcarea cozilor şi respectiv congestia. Practic, aceste mecanisme ne oferă posibilitatea de a alege între pierderea de pachete şi întârzierea sau jitterul – variaţiile în timp ale întârzierilor (parametri critici pe reţea).

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 foarte activ, care produce repede timeout şi retransmite toate pachetele în aşteptare folosind retransmiterea ultimelor *n,* va produce o încărcare mai mare decât un emiţător calm, care foloseşte retransmiterea selectivă. Strâns legată de acestea este politica de memorare în zonă tampon. Dacă receptorii distrug toate pachetele în afara secvenţei, acestea vor trebui retransmise ulterior, introducând o încărcare suplimentară. În ceea ce priveşte controlul congestiei, repetarea selectivă este, în mod sigur, mai bună decât retrimiterea ultimelor *n*.

Nivel	**Politică**
Transport	Politica de retransmisie Politica de memorare temporară a pachetelor în afară de secvenţă (out-of-order caching) Politica de confirmare Politica de control al fluxului Determinarea timeout-ului
Reţea	Circuite virtuale contra datagrame în interiorul subreţelei Plasarea pachetelor în cozi de aşteptare şi politici de servire Politica de distrugere a pachetelor Algoritmi de dirijare Gestiunea timpului de viaţă al pachetelor
Legătură de date	Politica de retransmitere Politica de memorare temporară a pachetelor în afară de secvenţă (out-of-order caching) Politica de confirmare Politica de control al fluxului

Figura 2.2 - Politici ce influenţează congestia

Şi politica de confirmare afectează congestia. Dacă fiecare pachet este confirmat imediat, pachetele de confirmare vor genera un trafic suplimentar. Totuşi, în cazul în care confirmările sunt preluate de traficul de răspuns, se pot produce timeout-uri şi retransmisii suplimentare. O schemă prea strânsă pentru controlul fluxului (de exemplu 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 subreţele cu 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 ce pachet este distrus dacă nu mai există spaţiu. O politică bună va ajuta la eliminarea congestiei, pe când una greşită o va accentua.

Un algoritm de dirijare bun poate ajuta la evitarea congestiei prin răspândirea traficului de-a lungul tuturor liniilor, în timp ce un algoritm neperformant ar putea trimite toate pachetele pe aceeaşi linie, care deja este congestionată. 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 ajunge la destinaţie, provocând astfel retransmisii.

La nivelul transport apar aceleaşi probleme ca la nivelul legăturii de date dar, 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. Dacă intervalul de timeout este prea mic, vor fi trimise inutil pachete suplimentare. Dacă este prea mare, congestia se va reduce, însă timpul de răspuns va fi afectat de pierderea unui pachet.

3. Tehnici cunoscute de prevenire a pierderii pachetelor în rețele

3.1. Introducere

Principalul scop al asigurării QoS este acela al controlării pierderilor de pachete.

Aceasta se realizează printr-un management adecvat al cozilor. Pachetele sunt pierdute din două motive: datorită erorilor de transmisie (fenomen relativ rar), sau datorită congestiilor din rețea. Pentru a controla și a evita congestiile din rețea trebuie implementate unele mecanisme în punctele de acces și în ruterele intermediare.

La capetele rețelei se află TCP-ul sau aplicațiile client care trebuie să implementeze un mecanism de adaptare la condițiile rețelei. În interiorul rețelei ruterele sunt cele care asigură, pe baza cozilor, un management al traficului. Cozile din arhitectura ruterelor sunt gândite să absoarbă vârfurile de trafic. Limitând dimensiunile cozilor scad și întârzierile pachetelor prin rețea. Din nefericire, atunci când cozile sunt pline, pachetele sunt eliminate.

În rețelele de mare viteză, gateway-urile sunt proiectate cu cozi de lungime maximă pentru a controla congestiile. Protocolul de transport TCP detectează o congestie numai după ce un pachet a fost aruncat de către gateway. Cu toate acestea, este evident că deși cozile au dimensiuni mari nu

este de dorit ca acestea să fie pline în majoritatea timpul – dacă s-ar întampla acest lucru, întârzierea medie din rețea ar crește semnificativ. Prin urmare, odată cu creșterea vitezei rețelei, este din ce în ce mai importantă prezența unor mecanisme care să mențină throughput-ul ridicat, însă dimensiunea cozilor mică.

În absența unui feedback explicit din partea gateway-ului, protocolul de nivel 4, transport, ar putea deduce apariția congestiei prin apariția gâtuirilor (bottlenecks), prin modificările ce apar în throughput, prin modificările întarzierilor punct la punct, dar și prin aruncarea pachetelor. De asemenea, perspectiva pe care o are o conexiune individuală este limitată de modificările ce apar în traficul acestei conexiuni, de modelul traficului, de lipsa cunoașterii numărului de gateway-uri ce sunt afectate de congestii, de schimbările de rutare posibile, dar și de propagarea întârzierilor.

Cea mai eficientă detecție a apariției congestiei o realizează gateway-ul. Numai gateway-ul are o vedere unică a modului în care evoluează cozile în timp. În plus, gateway-ul este împărțit între mai multe conexiuni cu timpi de răspuns variați, cu toleranțe față de întârzieri diferite și cerințe legate de throughput de asemenea diferite. Deciziile legate de durata și dimensiunea congestiei permise la nivelul gateway-ului sunt luate cel mai bine chiar de către gateway.

3.2 TCP

TCP implementează un mecanism de control al fluxului pe bază de fereastra. Propriu zis vorbind, un protocol pe bază de fereastră înseamnă că dimensiunea ferestrei curente definește o limită superioară a cantității de date nerecunoscută, care poate fi în tranzit între o pereche dată expeditor-receptor. Inițial, fluxul de control al TCP-ului a fost guvernat doar de dimensiunea ferestrei maximă permisă anunțate de către receptor și politică care a permis expeditorului să trimită noi pachete numai după ce a primit unidirecționat pachetul anterior.

Principalul scop al TCP este de a asigura circuit logic sigur sau serviciu de conexiune intre doua procese pereche. El nu se bazeaza pe siguranta altor protocoale de nivel inferior (cum este IP), asa ca TCP trebuie sa garanteze el insusi siguranta transmisiei. TCP poate fi caracterizat prin urmatoarele facilitati pe care le asigura pentru aplicatiile care il utilizeaza:

- Transfer de date in flux (Stream Data Transfer)

Din punctul de vedere al aplicatiei, TCP transfera un sir (stream) continuu de octeti prin retea. Aplicatia nu trebuie sa-si faca griji cu segmentarea datelor in blocuri de baza sau datagrame. O face TCP prin gruparea octetilor in fragmente TCP, care sunt transferati IP pentru a fi transmisi la destinatie. De asemenea, insusi TCP decide cum sa segmenteze datele si poate trimite datele mai departe intr-un mod convenabil.Uneori, o aplicatie are nevoie sa fie sigura ca toate datele trasnsmise catre TCP au fost in realitate transmise la destinatie. Pentru aceasta, o functie push este definita. Ea va forta transmiterea tuturor segmentelor TCP ramase in memorie catre destinatie. Functia de inchidere (close) normala forteaza si ea transmiterea datelor la destinatie.

- Siguranta

TCP atribuie un numar de secventa fiecarui octet transmis si asteapta o confirmare (acknowledgment - ACK) de la aplicatia care receptioneaza datele TCP. Daca confirmarea nu vine intr-un interval de timp prestabilit, datele sunt transmise din nou. Deoarece datele sunt transmise in blocuri (segmente TCP) numai numarul de secventa al primului octet este trimis calculatorului destinatie. Aplicatia TCP destinatie foloseste numerele de secventa pentru a le ordona atunci cand sosesc neordonate si sa elimine segmentele duplicate.

- Controlul transmisiei (Flow Control)

Aplicatia TCP destinatie, cand trasmite o confirmare (ACK) catre emitent, indica de asemenea numarul de octeti pe care ii poate receptiona pe langa ultimul segment TCP primit, fara sa se supraîncarce sau să aparã depasirea memoriilor tampon (internal buffers) ale sale. Aceasta este trimis in ACK sub forma numarului cel mai mare de secventa pe care-l poate receptiona fara probleme. Acest mecanism este cunoscut sub numele de fereastra glisanta (window-mechanism) si va fi discutat in detaliu in acest capitol.

- Multiplexarea

Este realizata prin utilizarea porturilor, la fel ca si la UDP.

- Conexiunile logice

Siguranţa si mecanismele de control ale transmisiei descrise mai înainte necesitã ca TCP sã iniţializeze si menţina cateva informaţii referitoare la starea fiecarui flux de date (data stream). Aceasta informaţie de stare ce include socketurile, numerele de secvenţã, dimensiunile ferestrelor se numeste conexiune logica. Fiecare conexiune este unic identificatã de cãtre o pereche de socketuri utilizate de cãtre procesele emitente si receptoare.

- Full Duplex

TCP asigură două fluxuri concurente de date în ambele sensuri.

În afară de advertismentul ferestrei receptorului, awnd, controlul congestiei TCP introduce două noi variabile pentru conexiune: fereastra de congestie "CWND" şi pragul de start lent. Dimensiunea ferestrei transmiţătorului, w, a fost definită ca:

$$w = \min(cwnd, awnd)$$

în loc să fie egală cu *awnd*. Fereastra de congestie poate fi considerată ca fiind o contra-fereastră de avertizare, unde variabila "*awnd*" este folosită pentru a atenţiona pe transmiţător sa suprafolosească resursele receptorului îm timp ce scopul "*cwnd*-ului" este acela de a preveni transmiţătorul de a trimite mai multe pachete decât rețeaua poate suporta în condiţiile de încărcare curentă.

Controlul congestiei (reactivă) = se aplica după ce rețeaua este supraîncărcată

Congestie Evitarea (Proactive) = se aplica înainte sa devina rețeaua supraîncărcată

TCP (Transport Control Protocol) este un protocol punct la punct, orientat pe conexiune, introdus pentru a proteja rețeaua împotriva supraîncărcării puternice și pentru a realiza o împărțire echitabilă a lățimii de bandă pentru diferite surse TCP. Protocolul TCP detectează erori precum pachete eronate, pierdute sau duplicate. Emițătorul atribuie o secvență numerică fiecărui pachet (segment) și solicită o confirmare pozitivă (ACK) de la receptor. Receptorul detectează 24

pierderea de pachete verificând secvența numerică și confirmând, la fiecare pachet primit, ultimul pachet recepționat în ordinea corectă. Pentru fiecare pachet care nu este în ordine, o confirmare duplicat este trimisă. Acest ACK duplicat are rolul de a notifica perechea de faptul că a fost detectată o pierdere, și pentru a îi transmite emițătorului următoarea secvență numerică așteptată.

Protocolul TCP asigură un flux de octeţi sigur, printr-o conexiune nesigură, cap la cap, a reţelei sau inter-reţelei. Inter-reţeaua este formată prin asocierea unor reţele care diferă ca: topologie, lăţime de bandă, întârzieri, dimensiuni de pachete, etc. TCP se adresează dinamic la reţeaua Internet şi este robust la mai multe tipuri de defecte. Entitatea de transport TCP gestionează fluxurile TCP si interfeţele către IP. Conexiunile TCP sunt:

-duplex;

-punct-la-punct (TCP nu suportă difuzarea);

-orientate pe fluxuri de octeţi nu de mesaje.

Fluxurile de date de la procesele utilizator sunt împărţite în segmente de maxim 64 kB, tipic de 1500 de octeţi. Apoi fiecare segment este expediat ca o datagramă IP separată. Datagramele IP recepţionate sunt furnizate entităţii TCP care reconstruieşte fluxul original de octeţi, în ordine, deoarece IP nu asigură livrarea sau transportul ordonat al datagramelor. Serviciul TCP este asigurat prin crearea de către emiţător şi receptor a unor puncte finale numite socluri sau socket-uri. Portul este un punct de acces la servicii de nivel transport, TSAP (Transport Service Acces Point). Numerele de port mai mici decât 256 sunt alocate porturilor general cunoscute, rezervate serviciilor standard.(telnet-23, http -80).

Serviciile TCP principale asigurate de TCP sunt:

- **expedierea datelor (SEND),** atunci când e convenabil pentru TCP,
- **urgentarea expedierii (PUSH),** cere transmiterea imediată dacă e posibil,
- **urgentarea recepţiei (URGENT DATA):** transmiterea şi recepţia imediată pentru a întrerupe o prelucrare distantă, deoarece PUSH nu are efect la distanţă, dar URGENT DATA are.

Formatul antetului TCP

Antetul TCP are 20 octeţi plus o parte opţională. Segmentul TCP, format din antet şi date, are lungimea maximă de 64kB, adică 65.536 B, dar e limitat de **MTU (Maximum Transfer Unit)** al fiecărei reţele traversate. Eventual reţelele din cale mai fragmentează segmentul TCP. Fiecare fragment va avea propriile antete TCP şi IP.

<table>
<tr><td colspan="8">Port SursA</td><td>Port destinatie</td></tr>
<tr><td colspan="9">Numar de secventa</td></tr>
<tr><td colspan="9">Numar de confirmare</td></tr>
<tr><td>Lungime Antet (4)</td><td>Rezervati (6)</td><td>U R G</td><td>A C K</td><td>P S H</td><td>R S T</td><td>S Y N</td><td>F I N</td><td>Dimensiunea ferestrei</td></tr>
<tr><td colspan="8">Suma de Control</td><td>Indicator URGENT</td></tr>
<tr><td colspan="9">Optiuni (1-n) cuvinte pe 32 biti</td></tr>
<tr><td colspan="9">Date (optional)</td></tr>
</table>

Figura 3.1-Formatul Antetului TCP

Unde:

- Portul sursa : Numărul portului sursă pe 16 biţi folosit de receptor pentru a răspunde.
- Portul destinaţie: Numărul portului destinaţie pe 16 biţi.
- Numărul de secvenţă: Numărul de secvenţă al primului octet de date din acest segment. Dacă bitul de control SYN este setat, numărul de secvenţă este numărul de secvenţă iniţial (n) şi primul octet de date este n+1.
- Numărul de confirmare: Dacă bitul de control ACK este setat, acest câmp conţine valoarea următorului număr de secvenţă pe care receptorul se aşteaptă să-l primească.

- Data Offset: Numărul de cuvinte de 32 biţi din antetul TCP. El indică unde incep datele.
- Reserved: Şase biţi rezervaţi pentru utilizare in viitor; trebuie sa fie zero.
- URG: Precizează cã articolul pointer de urgenta are semnificatie in acest segment.
- ACK: Precizează cã articolul de "Numar de secventa confirmat" are semnificatie in acest segment.
- PSH: Functia push.
- RST: Resetează conexiunea.
- SYN: Sincronizează numerele de secventa.
- FIN: Nu mai sunt date de la emitent.
- Fereastra: Folosit in segmentele ACK. El specifică numărul de octeţi de date începând cu acela indicat în "numărul de secvenţă confirmat" pe care receptorul (adică emitentul acestui segment) doreşte (şi poate) să îl accepte.
- Suma de control: Complementul faţă de unu al sumei în complement faţă de unu al tuturor cuvintelor de 16 biţi din pseudo- antet, antetul TCP şi datele TCP. La calcularea sumei de control, câmpul "Suma de control" este considerat zero.
- indicatorul de urgenţă – permite identificarea poziţiei unor date de urgenţă, în cadrul protocolului TCP;
- date – datele protocolului nivelului superior

3.3 Mecanismul de comunicare TCP

Stabilirea conexiunii:

- înţelegerea în 3 paşi (hand-shake) sau comunicarea cu confirmare SYN = 1 arată o cerere de conectare **datelor Terminarea conexiunii**
- fluxul: fiecare octet e numerotat modulo 2
- antetul conţine numărul de secvenţă al primului octet

- controlul fluxului: prin credite (număr de octeţi)
- datele sunt transmise când vrea entitatea TCP: - PUSH - transmite acum
-urgent: transmite această dată din fluxul normal care are indicatorul urgent
-dacă se recepţionează un TPDU nepotrivit acestei conexiuni, se pune reset =1 în segmentul care pleacă

Terminarea conexiunii
- deconectarea normală cu FIN = 1;
- deconectarea bruscă (abort) cu pierdere de date.

Politici de implementare

- **emisia** trebuie aleasa **dimensiunea segmentului** şi a **ferestrei de transmisie**:

- dacă segmentul este prea mic, antetul este o încărcare prea mare
- dacă segmentul este prea mare, rezultă întârzieri prea mari
- alegerea ferestrei este o problemă la liniile cu lăţime de bandă mare şi întârziere mare: cu fereastra de 64 kB şi linia T3 de 4,736 Mbps, transmiterea a 64 KB durează 12 msec. Dacă RTT

(Round Trip Time), întîrzierea dus-întors este de 50 msec, atunci 75 % din timp emiţătorul este inactiv, aşteptând confirmări.

- **livrarea datelor**

- datele pot fi memorate sau sunt livrate imediat, ordonat. Cu menţiunea urgent sunt livrate imediat, dacă e posibil
- segmentele în afara secvenţei sunt acceptate sau rejectate
- la expirarea timpului se retransmite numai primul segment, toate, sau individual, (se menţin cronometre separate per segment)
-confirmarea se face imediat sau cumulat (se aşteaptă date în sens opus sau expirarea timpului)

- **controlul fluxului şi al congestiei**

Există trei ferestre, pentru emisie *Wt,* pentru receptive *Wre,* si pentru congestive . Se alege fereastra de emisie :

$$Wt = \min (Wre , Wc) \qquad (2.2)$$

O sursă TCP este considerată saturată în cazul unei sesiuni FTP, atunci când unde toate segmentele au dimensiunea maximă posibilă, rezultând pachete de lungimi constante. Termenul de pachet este echivalent cu termenul segment. TCP-ul deține o fereastră de congestie și o limită pentru „slow-start" ce sunt descrise de variabilele CWND (Congestion Window) și SSTHRESH (**Slow-start threshhold Value)** . CWND determină numărul maxim de pachete ce pot fi neconfirmate în rețea, iar SSTHRESH controlează creșterea dimensiunii CWND. Dacă CWND este mai mică decât SSTHRESH, în faza de „slow-start", fereastra de congestie este incrementată cu câte un pachet pentru fiecare confirmare neduplicată primită. Aceasta duce la o creștere exponențială a CWND. Când CWND este mai mare sau egală cu SSTHRESH, în faza de evitare a congestiei, CWND crește cu câte un pachet la fiecare RTT (round trip time), ceea ce corespunde unei creșteri liniare a CWND.

TCP reacționează la pierderea pachetelor în două feluri diferite. Setează câte un cronometru pentru fiecare pachet trimis. În cazul în care nu este primită o confirmare pentru acel pachet, înainte de expirarea timpului, CWND își reduce dimensiunea la dimensiunea maximă a segmentului iar SSTHRESH este setat la *max:*

$$\text{SSTHRESH} = \frac{flig\,htsize}{2.2\ dimsegment} \qquad (2.3)$$

Flight size reprezintă cantitatea de informație neconfirmată în rețea. Presupunând că emițătorul este saturat, valoarea CWND este egală cu valoarea flight size. Când emițătorul recepționează trei ACK-uri duplicat, înainte de expirarea timpului, este realizată o recuparare rapidă (fast recovery): într-o manieră simplificată, TCP trimite pachetul pierdut din nou, reduce SSTHRESH la *max* și setează CWND la noua valoare calculată SSTHRESH.

Congestia reţelelor este o cauză a distribuției inegală a resurselor de rețea și a fluxului de rețea, iar algoritmii de control al congestiei pot fi împărţiţi în algoritmi „Source" şi „link". Algoritmul de control al congestiei TCP este cel mai utilizat algoritm „Source", dar utilizarea TCP-ului nu a putut evita posibilitatea de apariție a congestiei în rețele. Algoritmii „ Link" ar putea depăși în mod eficient impasul și problemele de sincronizare la nivelul de strategie drop-coadă,iar cercetarea se concentrează pe algoritmii de AQM (Active Queue Management).Există trei tipuri de

algoritmi AQM : Red (Random early Decetion), Algoritm AQM bazat pe teoria controlului şi Algoritm AQM bazat pe teoria optimizării.

3.4 Algoritmi de management al cozii la receptor

Principalele caracteristici de performanță ale AQM includ :

• Utilizarea eficientă a cozii: coada ar trebui să evite supraîncărcarea rezultată în pierderea de pachete şi retransmisii nedorite sau pachete goale= goliciuni au rezultat într-un link subincarcat

• Întârzierea cozii: este de dorit să se păstreze întârzierea şi micile sale variații.

• Robustețea: Schema AQM are nevoie de a menținerea un comportament robust, în ciuda diferitelor condițiile de rețea, cum ar fi variații în numărul de sesiuni de TCP sau variații în întârzierea propagarii și capacitatea legăturii.

TCP a fost optimizat pentru rețele cu fir. Orice pierdere de pachete este considerată a fi rezultatul unei congestie a rețelei și dimensiunea ferestrei de congestie este dramatic redusă ca o măsură de precauție. O serie de algoritmi alternativi de control al congestiei au fost propusi pentru a ajuta la rezolvarea problemei fara fir : cum ar fi Las Vegas, Westwood, Veno și Santa Cruz.

TCP îndeplinește două sarcini de control fundamentale:

1) De control end-to-end al debitului, pentru a evita supraincarcarii buffer-ului receptorului ;

2) Adaptarea controlului congestiei la conectare, adaptarea cantitatii de informații transferata în rețeaua la starea congestionarea rețelei. Această funcție este bazata pe recuperarea congestiei, mai degrabă decât de prevenire. Crește traficul TCP oferit rețelei până când rețeaua este obligata să renunțe la unele pachete; apoi se strange pentru a da timp rețelei de a reveni din congestie, și începe din nou să crească traficul oferit pentru a forța un alt episod de congestie. Controlul fluxului TCP este pus în aplicare prin intermediul receptorului, „*advertised window" (rcvwnd) Wre,* stabilit de către receptorul în antetul de confirmare a segmentelor (ACK). Controlul congestiei este pus în aplicare prin intermediul ferestrei de congestie (CNWD) *Wc*, calculata de către transmițător după algoritmul de control al congestiei TCP.

Controlul congestiei se referă la controlarea traficului ce pătrunde într-o rețea (de telecomunicații sau de calculatoare), ceea ce presupune evitarea supraîncărcării legăturilor dintre nodurile intermediare ale rețelei, prin reducerea numarului de pachete trimise. Așa cum s-a prezentat anterior, controlul congestiei nu trebuie confundat cu controlul fluxului care protejează receptorul de a fi „copleșit" de către emițător.

Slow-start este parte a strategiei de control al congestiei folosit de TCP, protocolul de transmitere a datelor utilizate de mai multe aplicații pe Internet. Slow Start este folosit în combinație cu alti algoritmi pentru a evita trimiterea de mai multe date decât rețeaua este capabilă să transmită, scopul este de a evita congestionarea rețelei. Algoritmul este specificat de RFC 5681.

Slow-start este unul dintre algoritmii care TCP ii foloseste pentru a controla congestie în interiorul rețelei. De asemenea, este cunoscut sub numele de faza de creștere exponențială. În

timpul fazei de creștere exponențială, începe creşterea in regim încet a creșterii ferestrei de congestie TCP de fiecare dată cand o confirmarea este primită. Aceasta crește dimensiunea ferestrei cu numărul de segmente recunoscute. Acest lucru se întâmplă până când o confirmare nu este primită pentru se ajunge la unele segment sau o valoare de prag predeterminată. În cazul în care are loc un eveniment de pierdere, TCP presupune că este din cauza congestiei rețelei şi ia măsuri pentru a reduce sarcina oferite pe rețea. Odată ce pragul a fost atins, TCP intră în fază de creștere liniară (de evitare a congestiei). In acest moment, fereastra crește cu un segment pentru fiecare RTT. Acest lucru se întâmplă până când se produce un eveniment de pierdere.

Cu toate că strategia este denumit "Slow-Start", creșterea fereastrei de congestie este destul de agresivă, mai agresivă decât faza de evitare a congestiei. Înainte de „Slow Start" a fost introdus în TCP, faza inițială de evitare pre-congestie fost chiar mai rapid.

Algoritmul de bază slow-start : Algoritmul începe în faza de creștere exponențială, inițial cu o dimensiune a ferestrei de congestie (cwnd) de 1, 2 sau 10 segmente și crește prin dimensiunea unui segment (SS) pentru fiecare nou ACK primit. În cazul în care receptorul trimite un ACK pentru fiecare segment, acest comportament se dublează în mod eficient dimensiunea ferestrei fiecare călătorie dus-întors a rețelei. În cazul în care receptorul acceptă un ACK întârziate, rata de creștere este mai mică, dar totuși crește cu un minimum de un SMS de fiecare dată pentru un drum dus-întors. Acest comportament continuă până când dimensiunea ferestrei congestiei (cwnd) atinge dimensiunea ferestrei de avertizare a receptorului sau până când se produce o pierdere.

Atunci când are loc o pierdere, jumătate din cwnd curent este salvat ca un prag de pornire lentă (SSThresh) și Slow Start începe din nou de la dimensiunea ferestrei cwnd inițială. Odată ce cwnd ajunge la SSThresh, TCP trece în modul de evitare a congestiei în cazul în care fiecare nou ACK crește cwnd de SS × SS / cwnd. Acest lucru duce la o creștere lineară a cwnd.

Fast Restransmit : Un expeditor TCP foloseste un cronometru pentru a recunoaște segmente pierdute. În cazul în care o confirmare nu este primită pentru un anumit segment într-un timp specificat (în funcție de estimat timpul de întârziere tur-retur), expeditorul va asuma segmentul a fost pierdut în rețea, și va retransmite segmentul. Confirmarea Duplicate este baza pentru mecanismul Fast Restransmit , care funcționează după cum urmează: după ce a primit un pachet (de exemplu, cu numărul de ordine 1), receptorul trimite o confirmare prin adăugarea de la 1 la numărul de ordine (de exemplu, numărul de ordine 2), ceea ce înseamnă că receptor primește numărul de pachete de 1 și se așteaptă ca numărul de pachete de 2 de la expeditor. Să presupunem că cele trei pachete ulterioare s-au pierdut. Intre timp receptorul primește un număr de pachete de 5 și 6. După ce a primit numărul de pachete de 5, receptorul trimite o confirmare, dar numai pentru numărul de ordine 2. Când receptorul primește numărul de pachete de 6, acesta trimite un alt valoare confirmare de 2. Deoarece expeditorul primeste mai mult de o confirmare cu acelaşi număr de ordine (2 în acest exemplu), aceasta se numește duplicat confirmare.

Recuperarea rapidă : Există o variație a algoritmului slow-start, cunoscut sub numele de recuperare rapidă, care utilizează retransmiterea rapidă, urmată de evitarea congestiei . În algoritmul de recuperare rapidă, în modul de evitare a congestiei, atunci când pachetele nu sunt primite (ceea ce ar insemna detectarea a trei ACK duplicat), dimensiunea ferestrei de congestie este redusă la pragul de Slow Start , mai degrabă decât valoarea inițială.

Există mai mulți algoritmi de prevenire a congestiei prevăzuți de TCP. Primii algoritmi apăruți sunt TCP Tahoe și TCP Reno, ambii având la bază regulile de „slow-start" și „fast retransmit". Cei doi algoritmi, Tahoe și Reno, diferă prin modul în care reacționează la pierderea de pachete:

- ***Tahoe:*** pierderea este detectată atunci când timpul de timeout expiră înainte ca o confirmare să fie recepționată. În acel moment, Tahoe reduce fereastra de congestie la valoarea maximă a dimensiunii unui segment (MSS – maximum segment size) și se resetează la faza de „slow-start".

- ***Reno:*** în cazul în care sunt recepționate trei confirmări duplicate (ACK duplicat), Reno înjumătățește fereastra de congestie, efectuează o retransmitere rapidă (fast retransmit) și intră în recuparare rapidă (fast recovery). Până la mijlocul anilor 1990, toate timeout-urile și întarzierile dus-intors erau bazate numai pe ultimul pachet din buffer. Cercetătorii Larry Peterson și Lawrence Brakmo de la Universitatea din Arizona, au introdus ***TCP Vegas*** în care timeout-ul este setat iar întarzierea dus-întors este măsurată pentru fiecare pachet din buffer. De asemenea, acest algoritm folosește o creștere aditivă a ferestrei de congestie. Această varianta nu a cunoscut o răspândire foarte mare în afara laboratorului cercetătorilor.

- ***TCP New Reno*** îmbunătățește retransmisia în timpul fazei de recuperare rapidă (fast recovery) a TCP Reno. Pe parcursul recuperării rapide, pentru fiecare ACK duplicat, este transmis un nou pachet netrimis de la sfârșitul ferestrei de congestie, pentru a menține fereastra plină. Pentru fiecare ACK parțial din spațiul de secvențe numerice, emițătorul trimite următorul pachet cu secvența numerică următoare ACK-ului. Întrucât cronometrul de timeout este resetat ori de cate ori există un progres în buffer-ul de transmisie, New Reno poate umple găuri mari sau mai multe găuri în spațiul de secvențe. Deoarece New Reno poate trimite pachete noi la sfârșitul ferestrei de congestie în timpul recuperării rapide, este menținut un throughput mare pe durata procesului de umplere a găurilor, chiar și atunci când există mai multe găuri, de mai multe pachete fiecare. Când TCP intră în recuperare rapidă, acesta înregistrează cea mai mare secvență numerică neconfirmată. Când această secvență numerică este confirmată, TCP revine la stare de evitare a congestiei. Atunci când nu au loc pierderi de pachete, dar in schimb pachetele sunt reordonate cu mai mult de 3 secvențe numerice, apare o problemă în New Reno. Când acest fenomen are loc, New Reno intră greșit în recuperare rapidă, însă când pachetul reordonat este livrat, are loc progresul secvenței numerice ACK și din acel moment până la sfârșitul recuperării rapide, fiecare bit al secvenței numerice produce un duplicat și retransmisia nu este necesară.

În rutare , management activ al cozii (AQM) reordoneaza arbitrar sau pierde pachetele de rețea în interiorul bufferul de transmisie a interfației unui controler de rețea. Sarcina este efectuată de către designerul de rețea. Un router de Internet menține de obicei un set de cozi, câte unul pe fiecare interfață. Sarcina activa a cozii este de a pierde sau marca pachetele înainte ca ea sa fie plina. De obicei, acestea operează prin menținerea uneia sau mai multor probabilități de pierdere sau de marcare .

Pentru a evita colapsul congestie, TCP foloseste o strategie multi-fațete de control a congestie. Pentru fiecare conexiune, TCP menține o fereastră de congestie, limitând numărul total de pachete nerecunoscute care pot fi în tranzit end-to-end. Acest lucru este oarecum analog cu fereastră glisantă TCP folositã pentru controlul fluxului. TCP folosește un mecanism numit start lent [29] pentru a mări fereastra de congestie, după o conexiune este initializatã si dupa un timeout. Acesta începe cu o fereastră de două ori dimensiunea maximă a segmentului (MSS). Deși rata inițială este scăzută, rata de creștere este foarte rapidă: pentru fiecare pachet recunoscut, fereastra de congestie crește cu 1 MSS, astfel încât fereastra de congestie dublează în mod eficient de fiecare dată dus-întors (RTT). Când fereastra de congestie depășește un prag ssthresh algoritmul intră într-o stare nouă, numită evitare a congestiei. În unele implementari (de exemplu, Linux), ssthresh inițială este mare, și astfel primul start lent, de obicei, se termină după o pierdere. Cu toate acestea, ssthresh este actualizat la sfârșitul fiecărui start lent, și va afecta de multe ori începe lent ulterioare declanșate de timeout.

Evitare a congestiei. Atâta timp cât ACK-uri non-duplicat sunt primite, fereastra de congestie este aditiv a crescut cu un SMS de fiecare dată dus-intors. Când un pachet este pierdut, probabilitatea de a fi primit ACK duplicat este foarte mare (este posibil, deși puțin probabil ca fluxul tocmai a suferit reordonare pachete de extremă, care ar putea, de asemenea, ACK duplicat prompte). Comportamentul de Tahoe si Reno diferă în modul în care detectează și reacționează la pierderea de pachete:

- Tahoe: ACK triple duplicat sunt tratate la fel ca un timeout. Tahoe va efectua "Fast Restransmit", a stabilit pragul de start lent la jumătate ferestrei de congestie curent, reduce fereastra de congestie la 1 MSS, și a reseta pentru faza de Slow Start. [29].
- Reno: În cazul în care sunt primeşte trei ACK duplicat (de exemplu, patru ACK recunoscând în același pachet, care nu sunt piggybacked pe date, și nu se schimba fereastra de publicitate receptorului), Reno va înjumătăți fereastra de congestie (în loc de a stabili o la 1 MSS ca Tahoe), a stabilit pragul de pornire lent egal cu noul ferestrei de congestie, efectua o retransmite rapid, și intră într-o fază numită de recuperare rapidă. În cazul în care un ori ACK din, început lent este utilizat ca acesta este cu Tahoe.

Recuperarea rapidă. (În cazul algoritmului New Reno) În această stare, TCP retransmite pachetul lipsește, care a fost semnalat prin trei ACK duplicat, și așteaptă o confirmare a întregii ferestrei de transmisie înainte de a reveni la evitarea congestiei. Dacă nu există nici o confirmare, TCP Reno experimentează un timeout și intră în starea de slow-start. Ambii algoritmi reduc fereastra de congestie la 1 MSS pe un eveniment de timeout.

TCP Cubic îmbunătățește corectitudinea împărţirii prin utilizarea în timp real pentru evoluția din fereastră de congestie. S-au făcut multe mai multe cercetări pentru a se găsi o metodă de tratare a congestiei şi pentru împărţirea corectă a lăţimii de bandă la nivel de rețea. Au fost propuşi mai mulţi algoritmi AQM bazaţi pe echitate. Ei folosesc o gamă largă de tehnici pentru a detecta care link mai multă de lățime de bandă decât ar trebuie și aruncă în rețea mai multe pachete ca urmare a ocupării acestui flux mai mare. În unii algoritmi (de exemplu CHOKE), nu există o detectare explicită a ocupării de prea multă lățime de bandă - echilibrul între consumul de bandă este menținut în mod automat şi datorită algoritmului ingenios de gestionare a cozii.

Acum trebuie subliniat faptul că cele mai multe studii anterioare au fost concentrate pe un singur tip de soluție și anume, fie pe nivelul rețea (AQM) sau nivelul transport (controlul congestiei TCP). Când noul AQM a fost propus, a fost testat pentru controlul congestiei clasice (New Reno) . Similar, după ce a fost inventată noua variantă de TCP, a fost evaluată cu managementul clasic al cozii.

4. ECN- Explicit Congestion Notification

4.1 Definiție ECN

Algoritmii de evitare şi de control al congestiei se bazează pe noţiunea că reţeaua este practic o "cutie neagră" (opacă). Starea reţelei este determinată de sistemele de la capătul acesteia care probează crescând gradual încărcarea pe reţea până când aceasta devine congestionată şi se pierde cel puţin un pachet. Tratând reţeaua ca o cutie neagră şi pierderea ca o indicare a congestiei în reţea ideal este ca să se întârzie sau să se piardă pachete individuale fără senzitivitate mare. În plus, algoritmii TCP de control al congestiei au implementate diferite tehnici (ca de exemplu Fast Retransmit sau Fast Recovery) ca să minimizeze pierderile. Totuşi aceste mecanisme nu intenţionează să ajute aplicaţiile care au o senzitivitate mare la pierderea sau întârzierea a unuia sau mai multor pachete individuale.

Deoarece TCP determină fereastra de congestie potrivită folosirii crescând gradual mărimea ferestrei până când experimentează un pachet pierdut, acest lucru provoacă apariţia sau chiar creşterea cozilor la ruterele congestionate.

Mecanismele de management al cozilor (AQM) pot folosi una sau mai multe metode pentru a indica prezenţa congestiei la nodurile finale. Una dintre acestea este de a folosi pierderea de pachete. Totuşi ruterele datorită mecanismului pot separa politicile de pierdere de pachete de cele care indică congestia. AQM permite ruterelor să folosească bitul CE (Congestion Experienced) în headerul unui pachet ca o indicare a congestiei, în loc să se bazeze doar pe pierderea de pachete.

RED este un mecanism AQM care a fost propus pentru a detecta congestia în starea sa incipientă şi este în momentul actual folosit în Internet. AQM este creat pentru a fi un mecanism general ce foloseşte una din mai multe alternative pentru indicarea congestiei, dar în absenţa ECN, AQM este restricţionat în vederea folosirii pierderii de pachete ca şi mecanism de indicare a congestiei. AQM pierde pachete bazându-se pe lungimea cozii medii ce depăşeşte un threshold, decât doar atunci când coada este "inundată". AQM poate arunca pachete înainte de inundarea efectivă a cozii, AQM nu este întotdeauna forţat de limitările de memorie să arunce pachete.

AQM poate seta bitul CE în headerul pachetului în loc să îl arunce, atunci când un asfel de câmp este asigurat în headerul IP şi înţeles de TCP. Folosirea acestui bit cu ECN permite receptorilor să primească pachetul, evitând întârzierile excesive datorate retransmisiei după pierderea pachetului. Se va folosi termenul pachet CE pentru a semnifica faptul că un pachet are setat bitul CE.[6]

4.2 Suportul ECN în IP

Acest paragraf specifică faptul că Internetul pune la dispoziţie o indicare a a congestiei în fază incipientă, iar notificarea poate fi uneori prin marcarea pachetelor decât prin aruncarea acestora. Astfel se foloseşte câmpul ECN în headerul IP cu doi biţi, formând 4 coduri ECN, de la 00 la 11. ECT sau ECN- Capable Transport (01 şi 10) sunt setate de transmiţător pentru a indica faptul

că protocolul TCP are capatele din reţea capabile ECN. Codul este format din ECT(0) şi ECT(1). Ruterele tratează aceste coduri ca fiind echivalente. Emiţătorii sunt liberi să folosească fie ECT(0) fie ECT(1) pentru a indica ECT pe bază pachet-pachet.

Folosirea ambelor coduri pentru ECT este în primul rând motivată de dorinţa de a permite mecanismelor ce emit date să verifice că elementele din reţea nu şterg codul CE şi că receptorii raportează corect emiţătorului recepţia unui pachet cu codul CE setat, după cum impune protocolul de transport.

Codul ”00” indică un pachet ce nu foloseşte ECN. Codul CE ”11” este setat de un ruter pentru a indica prezenţa congestiei în nodurile finale. Ruterele ce primesc un pachet într-o coadă deja plină, aruncă pachetul la fel cum se întâmplă în absenţa ECN.

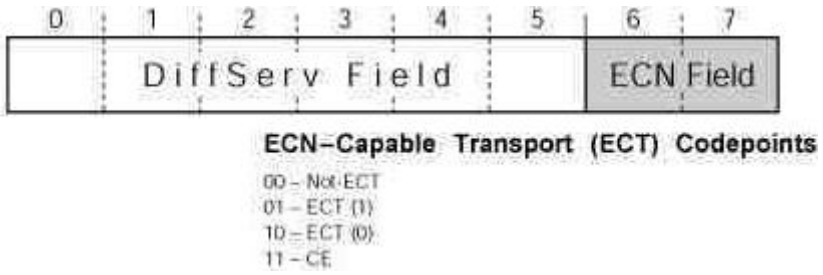

Figura 4.1 – Câmpul ECN în IP

În principiu, ECN a fost divizat în biţii ECT şi CE, iar câmpul ECN ce are setat doar bitul ECT corespunde lui ECT(0), iar cel care are setaţi ambii biţi (ECT şi CE) corespunde codului CE.

Biţii 6 şi 7 din Type of Service(ToS) IPv4 sunt dedicaţi câmpului ECN şi corespund octetului Traffic Class din IPv6, câmpul ECN fiind definit identic în ambele cazuri.

La primirea unui singur pachet CE de către un ECT, algoritmii de control al congestiei de la capetele reţelei trebuie să fie practic aceeiaşi cu răspunsul controlului congestiei la pierderea unui singur pachet. Un motiv pentru care se impune ca răspunsul controlului congestiei la un pachet CE să fie la fel cu răspunsul la un pachet pierdut îl reprezintă acomodarea la implementarea treptată a ECN-ului în rutere cât şi în sistemele de final. Unele rutere pot pierde pachete, în timp ce altele setează codul CE, pentru a indica prezenţa congestiei. Identic, un ruter poate pierde un pachet ce nu este capabil ECN, dar setează codul CE într-un pachet capabil ECN, petntru un level echivalent al congestiei. Dacă au fost diferite răspunsuri de control ale congestiei la un cod CE decât la un pachet pierdut, aceasta poate rezulta într-o tratare incorectă pentru fluxuri diferite. Un alt scop poate fi reprezentat de sistemele de final care trebuie să reacţioneze la congestie cel mult o dată pe fereastra de date pentru a evita reacţia de mai multe ori la mai multe indicaţii ale congestiei într-un interval de timp tur-retur.

Pentru un ruter, codul CE al unui pachet capabil ECN ar trebui doar să fie setat dacă ruterul ar fi aruncat un pachet ca să indice congestia nodurilor finale. Când bufferul ruterului nu este încă plin şi ruterul este pregătit să arunce un pachet ca să informeze nodurile finale de o congestie în fază incipientă , ruterul ar trebui mai întâi să verifice dacă codul ECT este setat în headerul IP al acelui pachet, iar daca este în loc să piardă pachetul, ruterul poate în schimb să seteze codul CE în headerul IP.

Într-un mediu în care toate nodurile finale au fost capabile ECN ar trebui să permită un nou criteriu pentru a îmbunătăţi setarea codului CE şi un nou mecanism de control al congestiei la reacţia nodurilor finale la pachete CE.

Când un pachet CE este recepţionat de un ruter, codul CE este lăsat neschimbat şi pachetul este trensmis ca de obicei. Când este depistată o congestie într-un stadiu avansat şi coada ruterului este plină, atunci ruterul chiar nu are nicio soluţie decât să arunce un pachet atunci când este recepţionat unul nou. În principiu această pierdere va deveni relativ rară când majoritatea sistemelor finale devin capabile ECN şi iau la rândul lor parte în TCP sau în alte mecanisme compatibile de control al congestiei. Într-un mediu capabil ECN care este provizionat adecvat, pierderea de pachete ar trebui să apară în timpul unor tranziţii sau în prezenţa unor surse ce nu cooperează.

Ceea ce se doreşte este ca ruterele să seteze codul CE ca răspuns împotriva congestiei fapt indicat de mărimea cozii medii, folosind algoritmii RED. În timp ce ECN este strâns legată de nevoia de a avea un mecanism rezonabil de management al cozii active de la ruter, reciproca nu este valabilă; mecanismele de management al cozii active au fost dezvoltate şi desfaşurate independent de ECN, folosind pierderea de pachete ca o indicaţie a congestiei în absenţa ECN-ului în arhitectura IP.

Evident este faptul că un singur pachet cu codul CE setat într-un pachet IP cauzează stratului transport, în termeni de control al congestiei, exact la fel cum ar fi făcut la pierderea unui pachet. Dimensiunea cozii instantanee este utilizată probabil pentru a observa variațiile considerabile chiar și atunci când un ruter nu experimentează congestia. Astfel, este important ca congestia tranzitorie la un ruter, reflectată de dimensiunea cozii instantanee ce atinge un prag mult mai mic decât capacitatea cozii, să nu declanșeze o reacție la nivelul stratului de transport, deci codul CE nu ar trebui setat de un ruter bazat pe mărimea cozii instantanee.[4]

Pachetele pierdute sau corupte

Pentru folosirea ECN-ului nodurile finale detectează pachetele de date pierdute, iar răspunsul congestiei nodurilor finale la pachetul de date pierdut este cel puțin la fel puternic ca răspunsul congestiei la un pachet CE recepționat. Pentru a asigura livrarea sigură a indicației congestiei dintr-un cod CE, un cod ECT nu trebuie să fie setat într-un pachet decât dacă pierderea pachetului în rețea ar fi detectată de nodurile finale și interpretată ca o indicație a congestiei.

Protocoalele de transport (exemplu TCP) nu detectează chiar toate pachetele pierdute, la fel ca pierderea unui pachet ACK. De exemplu, TCP nu reduce rata de sosire a pachetelor ACK ulterioare ca răspuns la un pachet ACK pierdut anterior. Orice propunere de extindere a capabilităților ECN la astfel de pachete vor adresa probleme ca de exemplu cazul unui pachet ACK care a fost marcat cu codul CE, dar a fost pierdut mai târziu în rețea.

La fel dacă un pachet CE este pierdut mai târziu în rețea datorită erorilor de bit, nodurile finale ar trebui să invoce controlul congestiei, la fel cum răspunde TCP la un pachet de date pierdut. Această problemă a corupției pacheteleor CE ar trebui considerată în orice propunere în rețea de distingere între pierderea de pachete datorită corupției li pierderea de pachete datorată depășirii bufferului.

Fragmentarea

Pachetele capabile ECN pot avea setat bitul DF. Reasamblarea unui pachet fragmentat nu trebuie să piardă indicarea congestiei. Cu alte cuvinte, dacă orice fragment al unui pachet IP ce va fi reasamblat are codul CE setat, atunci trebuie luată una din cele două opțiuni:

- trebuie setat codul CE al pachetului reasamblat. Totuși, asta nu trebuie să se întâmple dacă oricare din celelalte fragmente ce contribuie la reasamblare poartă bitul not-ECT.
- Pachetul este aruncat, în loc să fie reasamblat, din orice alt motiv.

Dacă ambele opțiuni pot fi aplicabile, oricare paote fi aleasă. Reasamblarea unui pachet fragmentat nu trebuie să schimbe codul ECN când toate fragmentele poartă același cod de biți. Implementările mai vechi se poate spune că nu realizau reasamblarea foarte corect, punând codul de biți CE doar într-un singur fragment. Emițătorul poate evita consecințele acestui comportament setând bitul DF în pachetele capabile ECN.

Totuți pot apărea situații neplăcute atunci când specificațiile reasamblării nu sunt destul de precise. De exemplu, dacă apare o nesincronizare a entității în calea înspre punctul de fragmentare sau chiar după, fragmentele pachetului pot căra o mixtură a codurilor de biți ECT(0), ECT(1) și not-ECT. Reasamblarea specificată mai sus nu impune cerințe pe reasamblarea fragmentelor în acest caz. În situații unde un comportament mai precis al reasamblarii este impus, specificațiile protocolului ar trebui să specifice că bitul DF trebuie setat în toate pachetele capabile ECN trimise de protocol.

4.3 Suportul din partea TCP

ECN impune suport din partea TCP în continuarea funcționalității date de câmpul ECN din headerul IP al pachetelor. TCP poate cere o negociere între capetele rețelei în timpul setărilor pentru a determina dacă toate componenetele sunt capabile ECN pentru ca emițătorul să seteze codul ECT în pachetele pe care le transmite. Dintr-o altă perspectivă TCP trebuie să fie capabil să reacționeze adecvat la recepția unui pachet CE. Această reacție poate fi prin notificarea emițătorului de către receptor de primirea unui pachet CE sau de alte acțiuni care în cele din urmă reduc rata de sosire a fluxului pe acea legătură congestionată.Pachetele CE indică mai degrabă o congestie persistentă decât una de tranziție și prin urmare reacțiile la primirea pachetelor CE ar trebui să fie cele adecvate pentru congestia persistentă.

Pentru TCP, ECN cere anumite părți de funcționalitate:

- Negocierea între capete în timpul setărilor pentru a determina dacă acestea sunt capabile ECN;
- Un steag ECE(ecou ECN) în headerul TCP pentru ca receptorul să poată informa emițătprul când a fost primit un pachet CE;
- Un steag CWR(Congestion Window Reduced) tot în headerul TCP pentru ca emițătorul să poată informa receptorul că a redus fereastra de congestie.

Aceste cerințe reprezintă doar suportul pentru protocolul ECN, suportul impus de alte protocoale poate fi diferit în special pentru protocoalele de transport multicast fie ele sigure sau nesigure.

Se pleacă de la presupunerea că TCP folosește algoritmii standard de control al congestiei Slow-start, Fast Retransmit și Fast Recovery.

Mecanismul TCP negociază capabilitatea ECN folosind steagul ECE din headerul TCP, acest steag este reprezentat de bitul 9 din câmpul Reserved al headerului TCP. Pentru ca receptorul TCP să știe când să se oprească din setarea din setarea steagurilor ECE s-a introdus un al doilea steag în headerul TCP, steagul CWR. Acestuia i-a fost asignat bitul 8 din câmpul Reserved al headerului TCP.

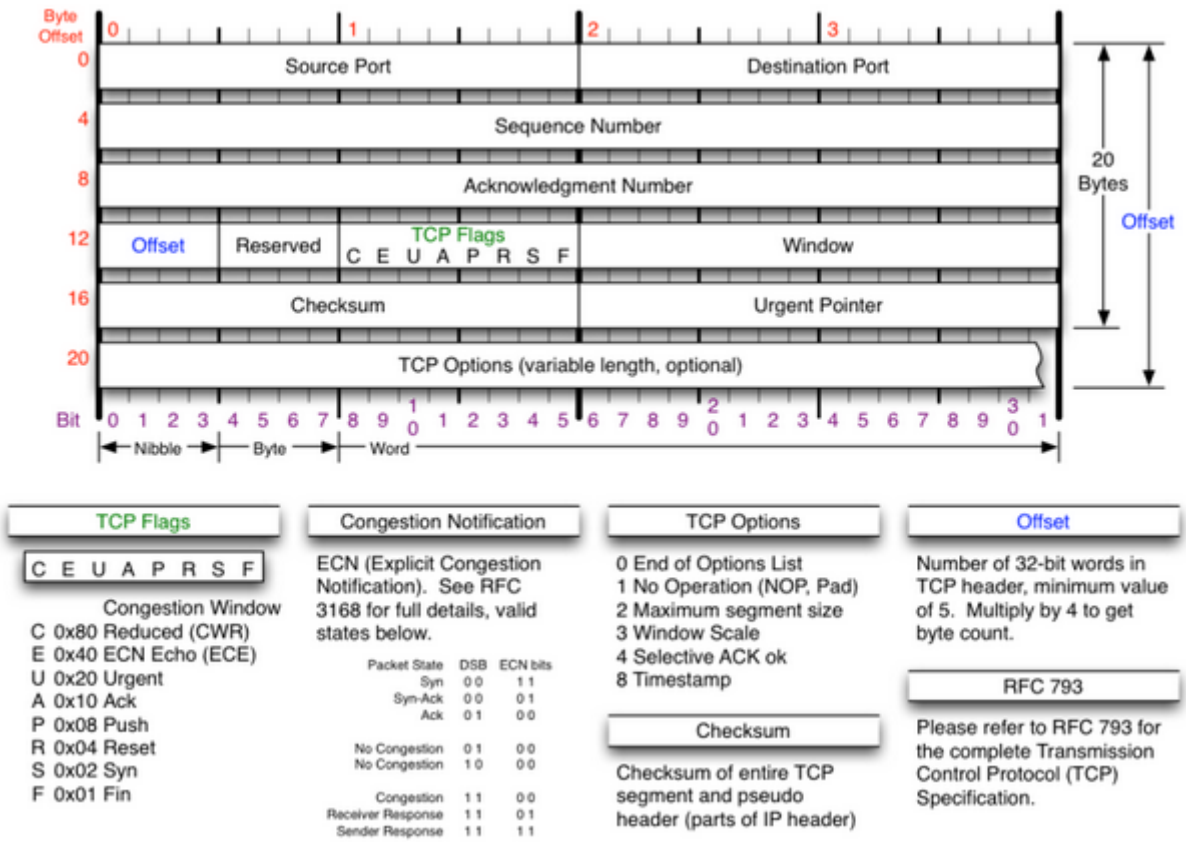

Figura 4.2- Headerul TCP

În figura 10 se prezintă întreg formatul headerului TCP, dar în continuare în această lucrare ne vom referi doar la doar la partea ce ține de ECN și anume biții specificați mai sus și importanța lor în cadrul lucrării.

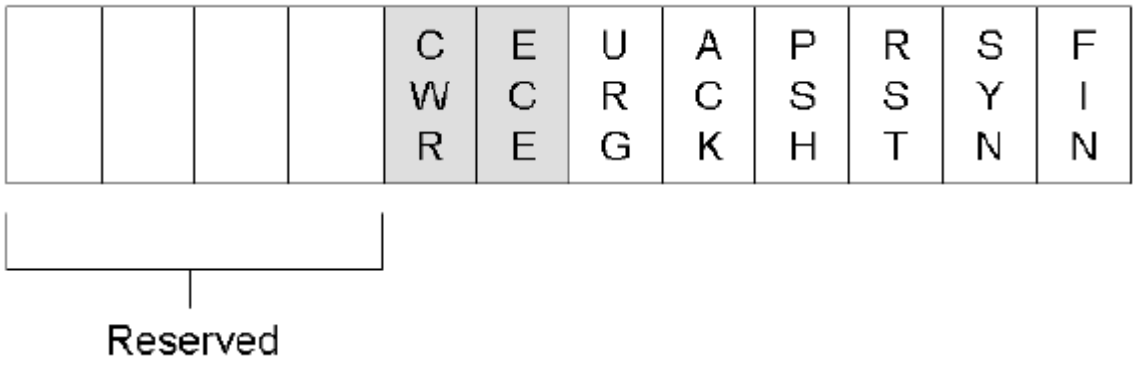

Figura 4.3-Biții rezervați ECN

Totuși ECN folosește steagurile ECT și CE din headerul IP pentru semnalizarea între rutere și captele de conectare și folosește steagurile ECE și CWR din headerul TCP pentru semnalizarea între capetele TCP. Pentru o conexiune TCP o secvență tipică de evenimente intr-o reacție bazată pe ECN împotriva congestiei poate fi:

- Un cod ECT este setat în pachetele transmise de emițător pentru a indica că Ecn este suportat de entitățile de transport ale acestor pachete.
- Un ruter capabil ECN detectează o congestie iminentă și că un cod ECT este setat într-un pachet ce urmează să fie aruncat.
- Receptorul primește un pachet cu codul CE setat și setează la rândul său steagul ECE în următorul TCP ACK trimis la emițător.
- Emițătorul primește un TCP ACK cu steagul ECE setat și reacționează la congestie ca și cum un pachet ar fi fost pierdut.
- Emițătorul setează steadul CWR din headerul TCP al următorului pachet trimis către receptor pentru a îi confirma acestuia recepția și reacția sa la un steag ECE.

Inițializarea TCP

În faza de setare a conexiunii TCP sursa și destinația schimbă informații despre disponibilitatea lor de a folosi ECN. Ulterior la finalizarae acestei negocieri, emițătorul TCP setează un cod ECT în headerul IP al pachetelor de date pentru a indica rețelei că transportul este capabil și dispus să participe la ECN pentru respectivul pachet.

Acest fapt indică ruterelor că acestea pot marca acest pachet cu codul CE, dacă acestea doresc să utilizeze această metodă pentru a notifica congestia. Dacă conexiunea TCP nu dorește să folosească notificarea prin ECN pentru un anumit pachet, emițătorul TCP setează codul ECN la non-ECT, iar receptorul ignoră codul CE din pachetul recepționat.

În continuare se va folosi asocierea de gazdă A și B pentru emițător și receptor. Un pachet SYN cu steagurile CWR și ECE setate se va numi un pachet SYN setat ECN, iar cu cel putin unul din cele 2 steaguri setate un pachet SYN nesetat ECN.

Înainte ca o conexiune TCP să poată folosi ECN, gazda A trimite un pachet SYN setat ECN, iar gazda B trimite un pachet SYN-ACK setat ECN. Pentru un pachet SYN, setarea ambelor

steaguri CWR și ECE în pachetul SYN setat ECN este definită ca o indicație că emițătorul TCP este capabil ECN, mai degrabă decât o indicație a congestiei sau un răspuns la congestie. Mai exact, un pachet SYN setat ECN indică facptul că implementare TCp care transmite un pachet SYN va participa în ECN și ca receptor și ca emițător. Ca receptor va răspunde la pachetele primite ce au codul CE setat în headerul IP prin setarea ECE în pachetele TCP ACK ce le va trimite, iar ca emițător va răspunde pachetelor primite ce au ECE setat ărin reducerea ferestrei de congestie și setarea steagului CWR când va fi nevoie. Un pachet SYN setat ECN nu permite emițătorului să seteze codul ECT în vreunul sau în toate pachetele pe care el le poate transmite. Totuși angajamentul de a răspunde adecvat la pachetele pe care le va primi cu codul CE setat rămâne chiar dacă emițătorul TCP într-o transmisie întârziată trimite un pachet SYN fără steagurile ECE și CWR setate.

Când gazda B trimite un pachet SYN-ACK setat ECN, setează steagul ECE dar nu și steagul CWR. Un pachet SYN-ACK este definit ca o indicație că TCP transmite un pachet SYN-ACK capabil ECN. La fel ca un pachet SYN, un pachet SYN-ACK setat ECN nu impune gazdei TCP să seteze un cod ECT în pachetele transmise.

În continuare vor fi prezentate anumite reguli care se aplică pachetelor setate ECN cu o conexiune TCP, unde această conexiune TCP este definită de anumite reguli standard pentru stabilirea și oprirea conexiunii.

- Dacă un pachet primește un pachet SYN setat ECN atunci acesta poate trimite un pachet SYN-ACK setat ECN, altfel poate să nu trimită un pachet SYN-ACK setat ECN.

- O gazdă nu trebuie să seteze ECT pe pachetele de date decât dacă a trimis cel puțin un pachet SYN setat ECN sau un pachet SYN-ACK setat ECN și a primit cel puțin un pachet SYN-ACK sau SYN setat ECN și a trimis din nou un pachet SYN sau SYN-ACK setat non-ECN. Dacă gazda a primit cel puțin un pachet non ECN atunci nu ar trebui să seteze ECT pe pachetele de date.

- Dacă o gazdă setează vreodată codul ECT pe un pachet de date, atunci acea gazdă trebuie să seteze corect bitul CWR pe toate pachetele ulterioare ale conexiunii.

- Dacă o gazdă a trimis cel puțin un pachet SYN sau SYN-ACK setat ECN și a primit un pachet Syn sau SYN-ACK non-ECN, atunci dacă acea gazdă primește pachete de date cu coduri ECT și CE setate în headerul IP, atunci acea gazdă trebuie să proceseze aceste pachete cum este specificat pentru o conexiune ECN capabilă.

- O gazdă care nu dorește să folosească Ecn pentru o conexiune TCP ar trebui să șteargă steagurile ECE și CWR din toate pachetele SYN și SYN-ACK setate non-ECN care sunt trimise pentru a arăta refuzul său. Receptorii trebuie să folosească corect toate formele din pachetele SYN sau SYN-ACK setate non-ECN.

- O gazdă nu trebuie să seteze pachetele SYN sau SYN-ACK.[13]

Un client TCP intră într-o stare de aşteptare după ce primeşte un FIN-ACK, şi tranzitează intr-o stare CLOSED după un timp de timeout. Multe implementări TCP creează o nouă conexiune TCP dacă primesc un pachet SYN în timpul stării de aşteptare. Când o gazdă TCP intră într-o stare de aşteptare sau una inchisă, ar trebui să ignore orice stare precedentă despre negocierea dintre ECN pentru acea conexiune.

Emiţătorul TCP

Pentru o conexiune TCP ce foloseşte ECN, noile pachete de date sunt transmise cu cu cod ECT setat în headerul IP. Când doar de un cod ECT de un emiţător pentru toate pachetele trimise pe o conexiune TCP, ar trebui folosit ECT(0). Dacă emiţătorul recepţionează un pachet ACK cu steagul ECE setat, atunci emiţătorul ştie că a avut loc congestia în reţea pe calea de la emiţător la receptor. Indicarea congestiei ar trebui tratată doar ca o pierdere a congestiei în protocolul TCP incapabil ECN. Astfel sursa TCP înjumătăţeşte fereastra de congestie (cwnd) şi reduce pragul de start lent (ssthresh). Emiţătorul TCP nu ar trebui să crească fereastra de congestie ca răspuns la recepţia unui pachet ACK cu steagul ECE setat.

TCP ar trebui să reacţioneze la indicarea congestie mai mult decât o dată la fiecare fereastră de date. Astfel fereastra de congestie a emiţătorului TCP ar trebui redusă o dată ca răspuns al unei serii de pierderi de pachete CE dintr-o singură fereastră de date. În plus, sursa TCP nu ar trebui să îşî micşoreze pragul de slow-start, dacă a fost micşorat în ultima tură dus-întors. Totuşi , dacă orice pachet retransmis este pierdut, atunci aceasta este interpretată de sursa TCP ca o nouă apariţie a congestiei.

Dacă fereastra de congestie constă doar într-un MSS(maximum segment size), iar emiţătorul TCP recepţionează un pachet ECK cu ECE setat, atunci emiţătorul TCP ar trebui în principiu să reducă în continuare fereastra sa de congestie la jumătate. Totuşi, valoareaferestrei de congestie este limitată sub o valoare a unui singur MSS. Este necesar să se reducă rata de trimite a emiţătorului şi mai mult, la recepţia unui pachet ECE când fereastra de congestie este 1. Se foloseţte timerul de retransmitere ca o măsură de reducere a ratei şi mai mult, în aceste circumstanţe. Prin urmare, emiţătorul TCP trebuie să reseteze timerul la recepţia unui pachet ECE când fereastra de congestie este 1.Emiţătorul TCP va fi disponibil să trimită un nou pachet doar când timerul de retransmitere expiră.

Când un emiţător capabil ECN îşî reduce fereastra de congestie din orice motiv acesta setează steagul CWR din headerul TCP al primului pachet de date trimis după reducerea ferestrei. Dacă acel pachet de date este pierdut în reţea, atunci emiţătorul va trebui să îşî reducă fereastra de congestie din nou şi să retrimită pachetul pierdut.

Când emiţătorul TCP este pregătit să seteze bitul CWR după ce a reduc fereastra de congestie, ar trebui să seteze bitul CWR doar pe primul pachet de date pe care îl va transmite.

TCP urmăreşte algorimti deja existenţi pentru trimiterea de pachete ca răspuns pentru ACK-uri sau timeouturi retransmise. El urmăreşte de asemenea procedura normală pentru a creşte fereastra de congestie când este primit un pachet ACK fără bitul ECE setat.

Receptorul TCP

Când TCP primește un pachet de date CE la sistemele de destinație, receptorul TCP setează steagul ECE în headerul TCP în următoarele pachete ACK. Dacă nu există nicio reținere a ACK implementate, ca și în implementările actuale ale Pachetelor ACK întârziate unde receptorul TCP poate trimite un ACK pentru două pachete de date primite, atunci steagul ECE din pachetul ACK va fi setat la ”1” dacă codul CE este setat în oricare din pachetele de date acceptate. Totul are loc dacă oricare dintre pachetele de date primite sunt pachete CE, atunci ACK returnat are setat steagul ECE.

Pentru a fi robust împotriva posibilității pierderii pachetelor Ack ce dețin un steag ECE, receptorul TCP setează steagul ECE într-o serie de pachete ACK trimise ulterior. Receptorul TCP folosește steagul CWR recepționat de la emițătorul TCP pentru a determina când să oprească setarea unui steag ECE.

După ce un receptor TCP trimite un pachet ACK cu bitul ECE setat, acel receptor continuă să seteze steagul ECE în toate pachetele ACK (chiar dacă au primit pachete de date CE sau non-CE) până când primește un paceht CWR (un pachet cu steagul CWR setat). După recepția unui pachet CWR, acceptările pachetelor CE ulterioare nu vor avea steagul ECE setat. Dacă un alt pachet CE este recepționat de receptor, acesta va trimite din nou pachete ACK cu steagul ECE setat. Cât timp recepția unui pachet CWR nu garantează că emițătorul a recepționat un mesaj ECE, aceasta sugerează că emițătorul a redus fereastra sa de congestie la un anumit punct după ce a trimis un pachet de date pentru care codul CE a fost setat.

A fost specificat faptul că un emițător TCP nu trebuie să-și reducă fereastra de congestie mai mult decât o dată pe fereastra de date. E nevoie atenție în cazul în care expeditorul TCP dorește să evite reduceri inutile ale ferestrei de congestie atunci când o fereastră de date include atât pachete pierdute cât și pachete CE marcate.

5. Simulatorul

Ns3 este un simulator de evenimente , orientat pe simularea in rețea care să permită imitarea într-o varietate de rețele locale și largi. El pune în aplicare diferite protocoale de retea (TCP, UDP), surse de trafic (FTP, web, CBR, exponențiale on / off), mecanisme de gestionare coadă (RED, DropTail), protocoale de rutare (Dijkstra), etc. NS3 este scris în C + + și Otcl la separa de control și implementări cale de date. Simulatorul are o ierarhie de clase în C + + (ierarhia compilat) și o ierarhie corespunzătoare în interpretul Otcl (ierarhie interpretat). Motivul pentru care ns3 folosește două limbi este că sarcini diferite au cerințe diferite: Un exemplu de simulare de protocoale necesită manipularea eficientă de bytes și antete de pachete de a avea viteza . Pe de altă parte, în studiile de rețea în cazul în care obiectivul este de a varia parametri și a examina rapid o serie de scenarii de timp pentru a schimba modelul și rulați-l din nou, este mult mai important. În NS3, C++ este utilizat pentru a aplica detaliat protocol și, în general, pentru astfel de cazuri în care fiecare pachet de un flux trebuie să fie prelucrate. De exemplu, dacă doriți să pună în aplicare o nouă disciplină de așteptare, apoi C++ este limbajul de alegere. Otcl, pe de altă parte, este adecvat pentru configurare. Otcl ruleaza destul de încet, dar poate fi schimbat foarte repede face construcția simulării mai ușor. În NS3, sursele compilate C++ ale obiectelor poate fi puse la dispoziția interpretului Otcl. În acest fel, obiectele gata făcute C++ pot fi controlate de la nivelul OTcl. Există destul de multe tutoriale de înțeles disponibile pentru utilizatorii NS-noi. Trecând prin, pentru exemplu, următoarele tutoriale ar trebui să vă dea o imagine destul de bună a modului de a crea scenarii simple, de simulare cu NS3: https://www.nsnam.org/docs/tutorial/html/ .

OTcl

Network Simulator este un simulator orientat pe obiecte, scris în C++, cu o interfață făcută de un interpretor OTcl. Simulatorul suportă o ierarhie a claselor în C++, și o ierarhie a claselor asemănătoare pentru interpretorul OTcl. Cele două ierarhii sunt strâns legate una de cealaltă. Din perspectiva utilizatorului, există o corespondență unu la unu, între o clasă din cadrul ierarhiei interpretorului și una din cadrul ierarhiei compilate. Rădăcina acestei ierarhii este clasa TclObject. Utilizatorii creează simulări noi prin intermediul interpretorului. Aceste obiecte sunt instanțiate în cadrul interpretorului și sunt oglindite de un obiect corespondent în ierarhia compilată. Ierarhia claselor interpretorului este realizată automat prin metode definite în clasa TclClass. Obiectele instanțiate de utilizator sunt oglindite prin metode definite in clasa TclObject. Există și alte ierarhii în codul C++ și scripturile OTcl. Aceste alte ierarhii nu sunt oglindite în maniera TclObject.

De ce două limbaje?

Network Simulator folosește două limbaje deoarece simulatorul are două tipuri de obiective pe care trebuie să le realizeze. Pe de o parte, simulări detaliate ale protocoalelor necesită un limbaj de programare al sistemului, care poate manipula eficient bytes, pachete, anteturi și care poate implementa algoritmi ce rulează cu seturi foarte mari de date. Pentru aceste obiective, viteza la rulare este importantă, în timp ce rularea simulării, găsirea bug-urilor și repararea acestora este mai puțin importantă. Pe de altă parte, o mare parte a cercetărilor în rețele implică variații mici ale parametrilor sau configurațiilor sau numărului de scenarii. În aceste cazuri, timpul necesar iterației (schimbarea modelului și rularea acestui din nou) este mai importantă. Din moment ce configurația rulează o singură dată (la începutul simulării), timpul de rulare pentru această parte este mai puțin important.

NS împacă cele două cerințe cu cele două limbaje, C++ și OTcl. C++ este rapid la rulare, însă mai greoi la schimbare, făcându-l potrivit pentru implementările detaliate ale protcoalelor. OTcl rulează mult mai greu, însă poate fi schimbat foarte rapid (și interactiv), făcându-l ideal pentru configurarea simulărilor.

Network Animator

Nam este o unealtă pentru animații, bazată pe Tcl/TK , folosită pentru vizualizarea datelor simulărilor de rețele. Teoria de implementare din spatele Nam a fost crearea unui animator care să poată citi seturi foarte mari de date pentru animații și care să poată fi îndeajuns de extensibil pentru a putea vizualiza o viariatate foarte mare de situații de rețele. Având această constrângere, **nam** a fost creat să poată citi comenzi simple de animație din interiorul unor fișiere foarte mari de urmărire. Pentru a putea lucra cu seturi de date foarte mari pentru animații, cantitatea de informație reținută în memorie este foarte mică. Comenzile sunt reținute în fișier și sunt recitite ori de câte ori este necesar.

Primul pas pentru a putea folosi **nam** este crearea unui fișier de urmărire. Acest fișier conține informații referitoare la topologii, noduri, legături dar și pachete. De obicei, acest fișier este creat de către Network Simulator.

După ce a fost generat fișierul pentru urmărire, acesta este pregătit pentru a putea fi animat de către nam. La pornire, nam citește fișierul, crează topologia, creează o fereastră pop-up, realizează layout-ul necesar și se oprește la momentul de timp 0. Network animator are o interfață pentru utilizator însă permite controlul asupra multor aspecte ale animației. [12]

Instalarea simulatorului

Simulătile au fost realizate cu ajutorul simulatorului Network Simulator v3.19 rulând pe un sistem de operare Ubuntu 14.04 instalat pe o mașină virtuală creată pe un Toshiba Satellite cu procesor Dual Core, memorie Ram 3 gb și o memorie externă de 300 gb.

Pentru instalarea simulatorului am urmat pașii ce urmează:

-descărcarea și instalarea fișierului ns-3-allinone prin comenzile:

```
wget   http: //www.nsnam.org/release/ns-allinone-3.19.tar.bz2
  tar   xjf ns-allinone-3.19.tar.bz2
  cd    ns-allinone-3.19/
```

Apoi se poate face construirea exemplelor prin comanda:

```
./build.py  --enable-examples  --enable-tests
```

Pentru a testa dacă totul este ok se poate folosi comanda:

```
./test.py   [15]
```

6. Testare algoritmi de congestie

Pentru început în această lucrare se va prezenta cum lucrează algoritmii de congestie din punctul de vedere a setărilor standard și care sunt performanțele acestora pentru o topologie de rețea dată.

Topologia aleasă este formată din:

- 4 link-uri cu o lățime de bandă de 10 Mbps și delayuri de 2,3,4 și 5 ms;
- 1 link bottleneck cu o lățime de bandă de 1.5 Mbps și un delay de 20 de ms.

Se consideră cele 4 link-uri cu conexiune FTP ce utilizează protocolul TCP cu dimensiunea maximă a ferestrei de 25. Fiecare din cele patru noduri transmit trafic FTP pe parcursul întregii perioade de simulare, iar pentru coada de bottleneck dimensiunea este de aleasă 15. Valorile folosite pentru parametrii sunt cele standard: min_{th}=5, max_{th}=15, w_q=0,002; unde min_{th} reprezintă coada minimă, max_{th} reprezintă coada maximă, iar w_q reprezintă încărcarea cozii.

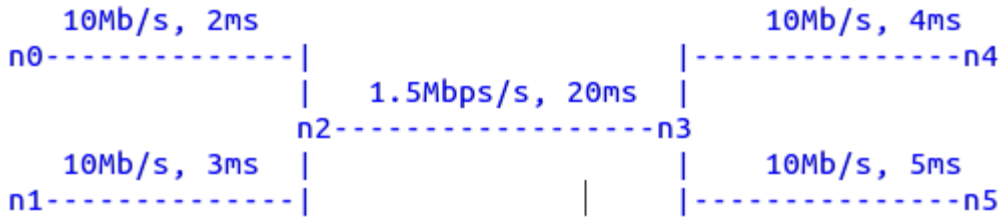

Figura 6.1 – Topologia rețelei

În continuare am realizat,în urma simulărilor algoritmilor. o comportare a acestora în condiții normale de rețea raportat la numărul de pachete aruncate:

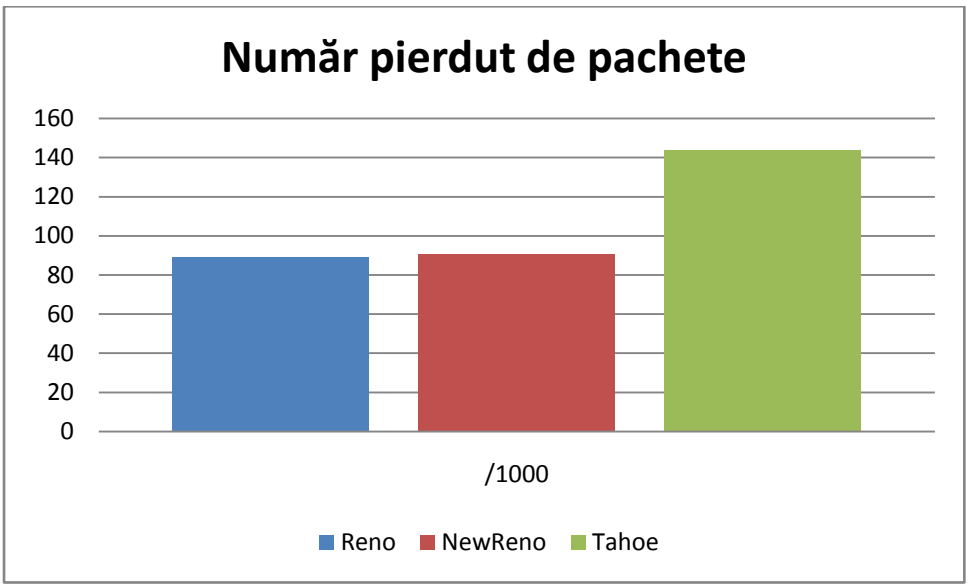

Figura 6.2 – Pachetele pierdute în rețea

Pe topologia realizată anterior formată din cele 6 noduri cu interfețele aferente se simulează toți cei trei algoritmi prezentați: Reno, New Reno și Tahoe. În urma rulării datele au fost centralizate în tabele din care au rezultat următoarele grafice:

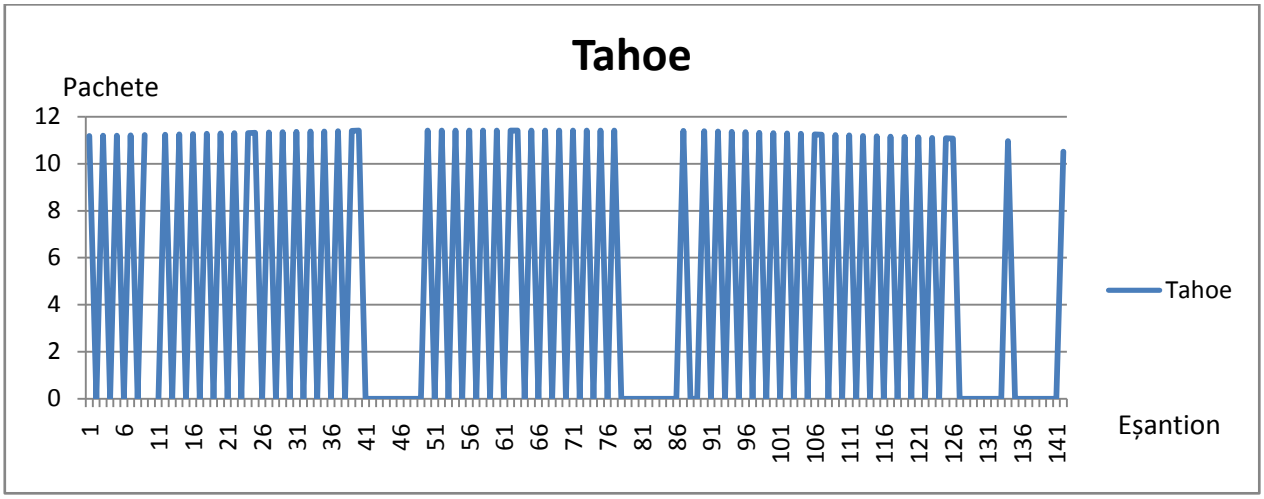

Figura 6.3 – Coada medie Tahoe

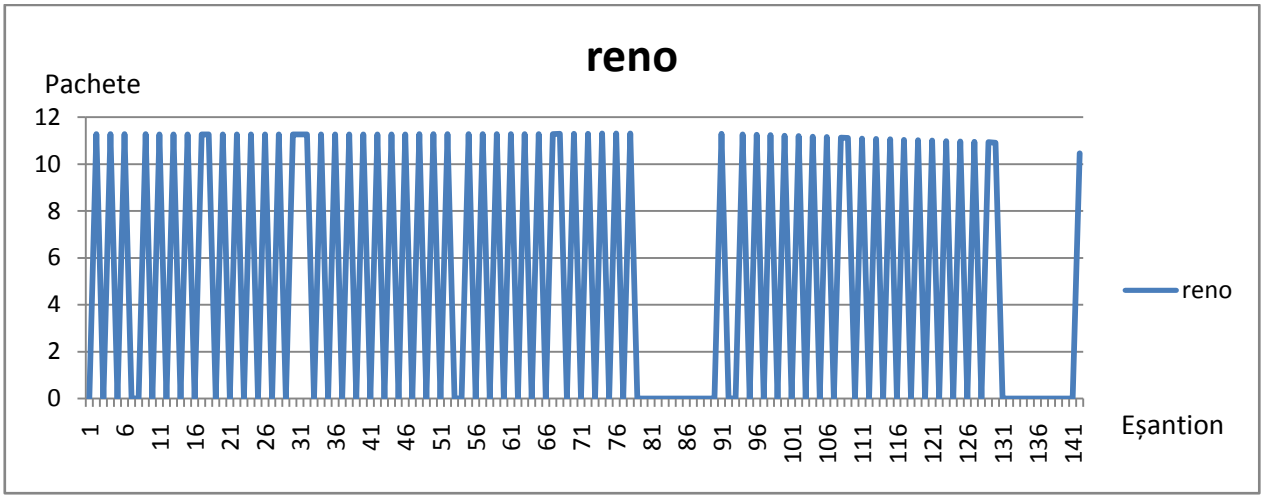

Figura 6.4 – Coada medie Reno

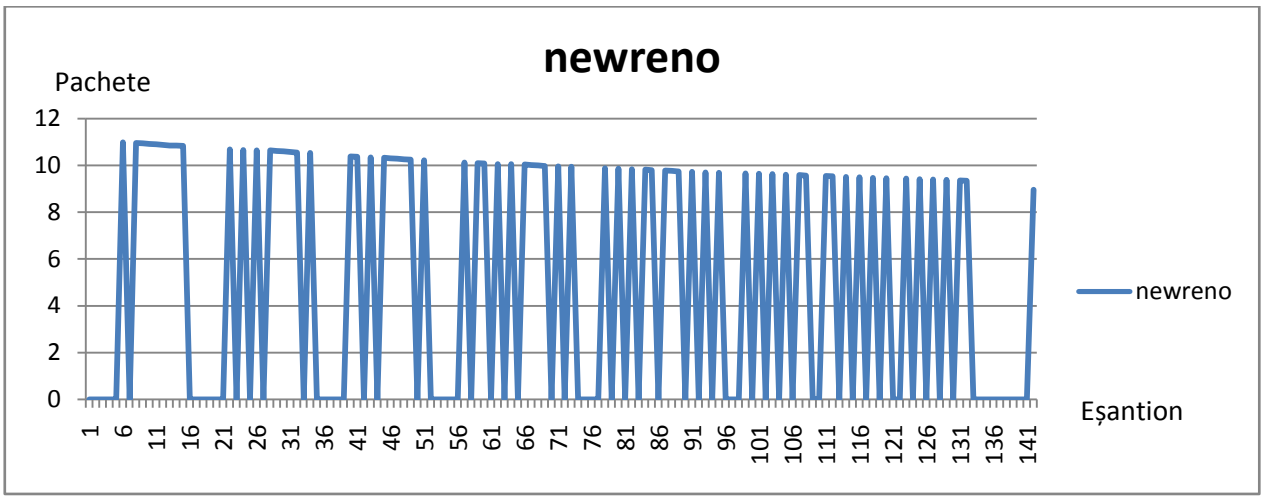

Figura 6.5 – Coada medie NewReno

Analizând, în paralel, cele 3 grafice rezultate din prelucrarea datelor extrase după rularea algorimilor se observă că cel mai eficient dintre acestea este algoritmul NewReno. Algoritmii Tahoe și Reno, fiind implementați și testați înainte au o eficiență oarecum mai scazută, New Reno prezentând o îmbunătățire deoarece s-a încercat optimizarea lipsurilor prezente în ceilalți algoritmi. De exemplu, NewReno în timpul algoritmului de recuperare rapidă(Fast Recovery) așteaptă să recupereze toate pachetele pierdute și trimite câteva pachete noi.

Se observă că panta scade cel mai constant și cu o valoare destul de mare la rularea algoritmului NewReno, îmbunătățirea fiind observată și în privința algoritmului Reno și foarte puțin și la Tahoe care totuși are o valoare destul de mare. Deși aceste grafice completează tabelul realizat cu numărul de pachete pierdute se poate spune că în urma testelor Tahoe este cel mai puțin eficient algoritm utilizat din cele 3, iar făcând o comparație între Reno și NewReno deși la numărul de pachete pierdute NewReno nu stă foarte bine acesta este foarte eficient când vine vorba de lungimea cozii medii pe care o ajustează foarte eficient menținând totodată și un număr scăzut de pachete pe care acesta le pierde tocmai datorită ajustării ferestrei de congestie la jumătate în timpul Fast Recovery .

7. Modificarea simulatorului

Ținând cont că schema ECN folosește porți RED pentru notificarea explicită a congestiei se poate face o descriere incipientă a schemei RED. O poartă RED menține două praguri ale cozii: min și max, și modifică încontinuu mărimea cozii medii. Când un număr de pachete are valoare mai mică decât min, nu se ia nicio acțiune, iar când mai multe pachete au valoare mai mare ca max, oricare pachet sosit este marcat sau aruncat. Când pachetele au valori între aceste două praguri , poarta RED calculează probabilitatea ca pachetul să poată fi marcat sau aruncat, proporțional lățimea de bandă a conexiunii. În plus, dacă se măsoară lungimea cozii medii mai degrabă in bytes decât în pachete, probabilitatea ca pachetul să fie marcat este de asemenea proporțională cu mărimea pachetului în bytes. Deci, dacă coada la ieșire este goală pentru o perioadă de timp, algoritmul estimează câte pachete ar fi putut fi transmise în acel timp, și modifică mărimea cozii medii în consecință.

Scopul schemei ECN este de a defini un răspuns la nivelul stratului de transport al nodului emițător pentru emiterea unei notificații a congestiei de către ruterul RED în cauză. Când schema ECN este activă, poarta RED este configurată pentru a marca mai degrabă decât a arunca pachetele.

Pe de-o parte, schema ECN folosește în acest raport un singur mesaj pentru a indica congestie, dar pe de altă parte schema încearcă să se asigure că TCP nu răspunde prea frecvent prin reacția la notificarea congestiei decât cel mult o dată pe un drum dus-întors. Urmând recepția unui pachet cu bitul ECN setat, emițătorul va înjumătăți fereastra de congestie și pragul de slow-start. Protocolul nu va înjumătăți acei parametri din nou ca răspuns la pachetele cu 3 ACK sau la alte pachete cu bitul ECN setat, până când toate pachetele în restante în timp ca răspuns la ECN au fost acceptate.

Modificarea ce a fost gândită astfel încât să fie urmați pașii prezentați mai sus va fi prezentată în cele ce urmează. Se va ține cont de toți parametrii ce trebuie modificați și de altfel se va elabora și o comparație a rezultatelor cu cele realizate mai sus pentru a putea observa performanțele ambelor metode și a identifica metoda mai eficientă și mai ușor de implementat.

Modificările se vor face direct în fișierul sursă și acestea sunt după cum urmează:

Renotcp [window = 15]

Red [min_thresh = 5 max_thresh= 15 q_weight = 0.002 max_p = 0.02 setbit = true]

Edge s1 to r1 bandwidth 10Mb delay 2ms
Edge s2 to r1 bandwidth 10Mb delay 3ms
Edge r1 to r2 bandwidth 1.5Mb delay 20ms

Forward [queue-type = red queue-size = 25]
Backward [queue-type = red queue-size = 25]

Edge s3 to r2 bandwidth 10Mb delay 4ms
Edge s4 to r2 bandwidth 10Mb delay 5ms

ftp conv from renotcp [ecn = 1] at s1 to sink at s3
ftp [start-at = 3.0] conv from renotcp [ecn = 1] at s2 to sink at s3

Astfel se va implementa Explicit Congestion Notification în sursele TCP și în porțile RED. În loc să fie forțate să arunce un pachet, porțile RED setează un bit ECN în headerul unui pachet în schimbul aruncării acestuia. Sursele TCP interpretează recepția unui pachet cu bitul ECN setat ca o indicație a congestiei.

În urma modificărilor aduse s-au extras din nou valorile prezentate anterior și s-au creat pentru aceiași parametri graficele următoare ce ilustrează o oarecare diferență de comportament al fiecărui algoritm la modificările aduse:

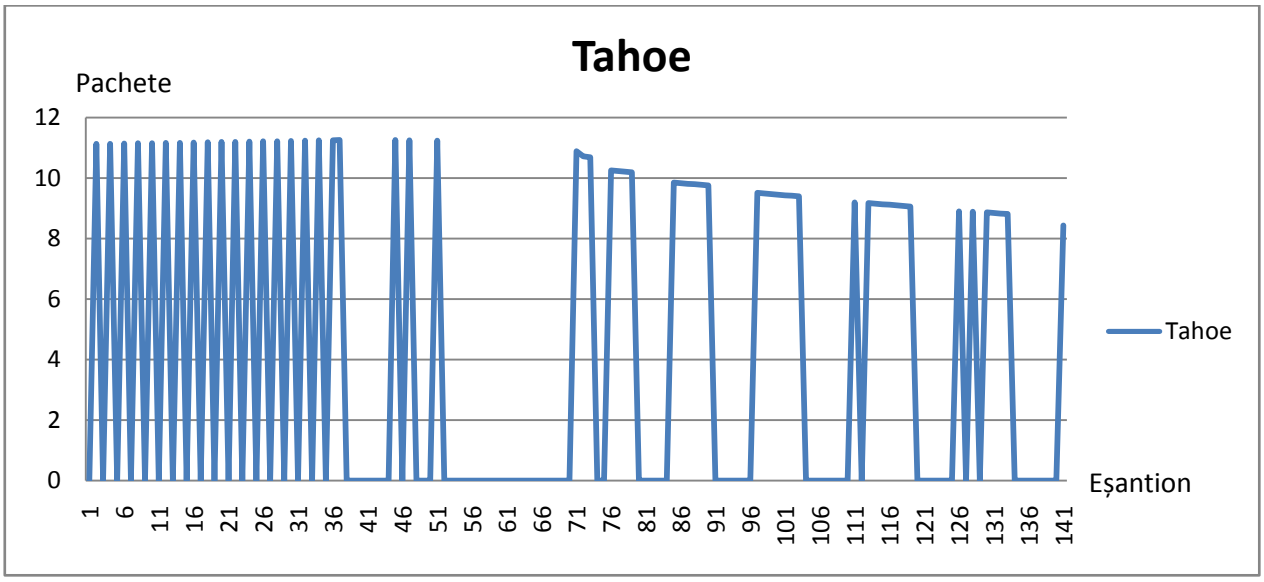

Figura 7.1 – Coada medie Tahoe

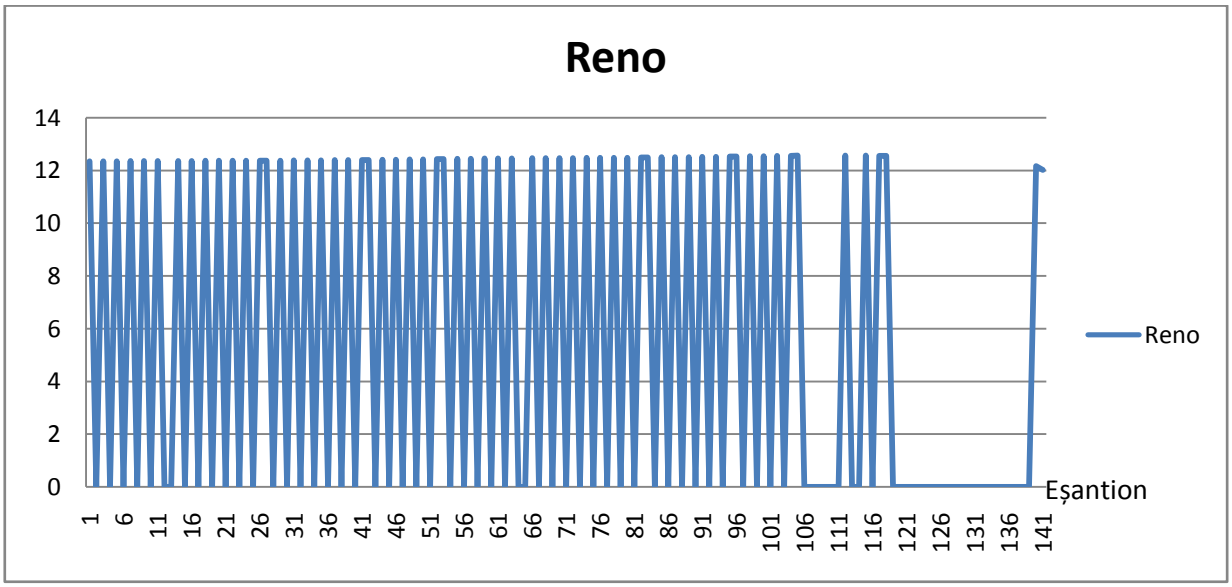

Figura 7.2 – Coada medie Reno

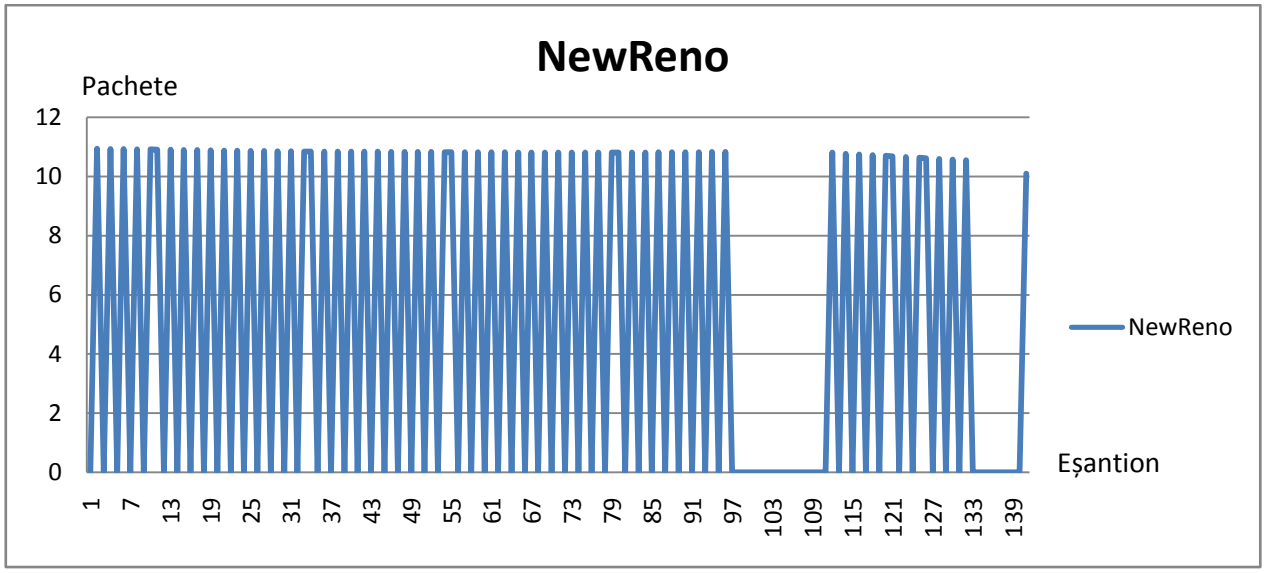

Figura 7.3 – Coada medie NewReno

Deși graficele diferă într-o oarecare măsură de cele prezentate anterior nu se poate da o decizie finală în privința metodei mai eficiente. Se observă că în privința algoritmilor Reno și New Reno schimbările au loc doar la momentul în care dimensiunea cozii se schimbă, numărul de pachete fiind în medie același ca în situația prezentată anterior. La Tahoe lucrurile se schimbă deoarece algoritmul este îmbunătățit prin folosirea metodei ECN, iar rezultatul se poate observa foarte ușor, dar totuși nu se poate cataloga ca o îmbunătățire substanțială deoarece algoritmul nu mai este așa de mult utilizat în prezent, iar implementarea sa este una oarecum învechită.

Totuși o îmbunătățire adusă de această metodă se observă în graficul ce urmează, grafic realizat prin centralizarea pachetelor pierdute în cazul rulării fiecărui algoritm cu metoda modificată.

Graficul a fost realizat prin numărarea pachetelor ce sunt aruncate în nodul numărul 2 al topologiei realizate, nod în care este simulată congestia, iar rezultatele se prezintă după cum urmează:

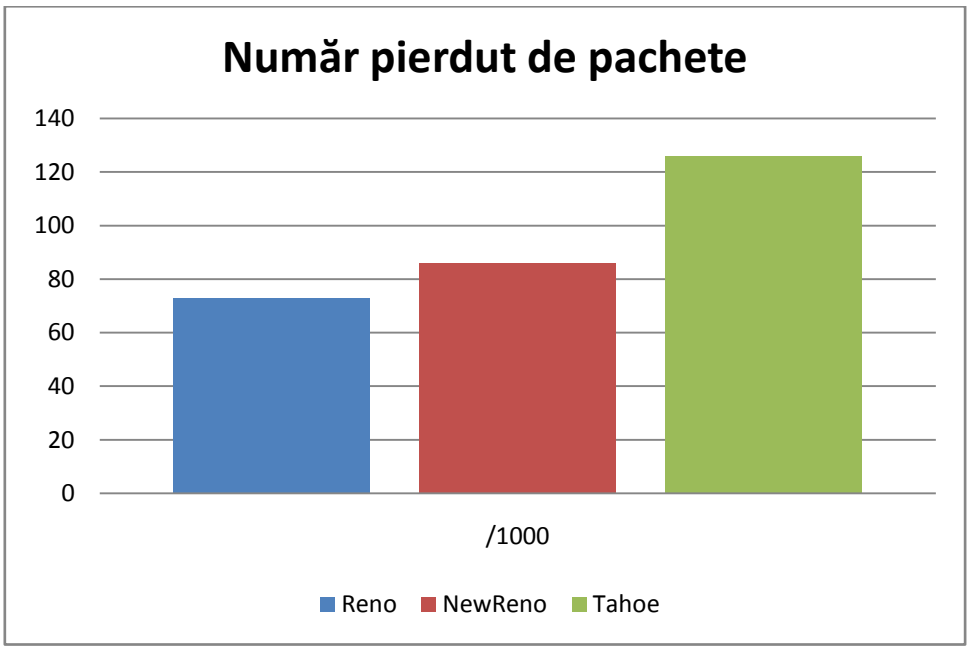

Figura 7.4 – Pachete pierdute în rețea

Avantajul metodei ECN este oarecum mai vizibil aici, dar nu foarte îmbunătățit deoarece procesul este același ca în cazul algoritmilor RED. Ceea ce diferă este doar semnalizarea faptului că a fost identificată congestia de către destinație asupra sursei, reacția fiind oarecum aceeași doar numărul de pachete pierdute fiind puțin mai mic datorită reacției sursei doar o singură dată a unui pachet 3-ACK.

A mai fost realizat un sondaj despre numărul pachetelor ce se află în coada RED la același moment de timp reprezentat și în graficele anterioare. Trebuie menționat că atunci când valoarea este 0 înseamnă că acea coadă RED este încă în așteptare deoarece a ajuns la limită și s-a saturat. Ea va menține această stare până în momentul când pachetele se vor retransmite și astfel se va crea loc în această coadă.

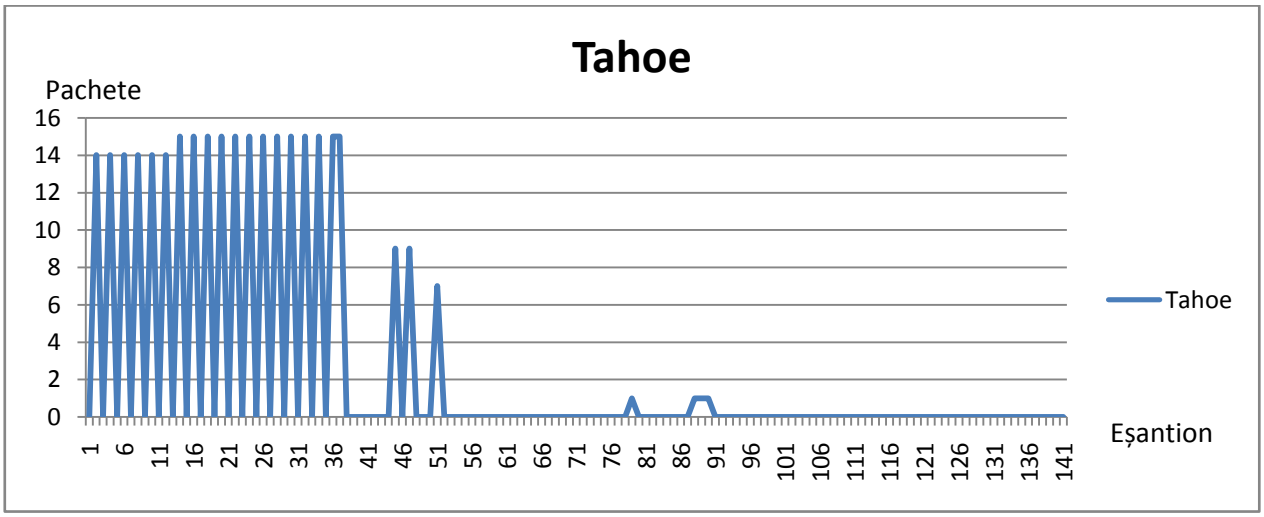

Figura 7.5 – Număr pachete în coada RED(TAHOE)

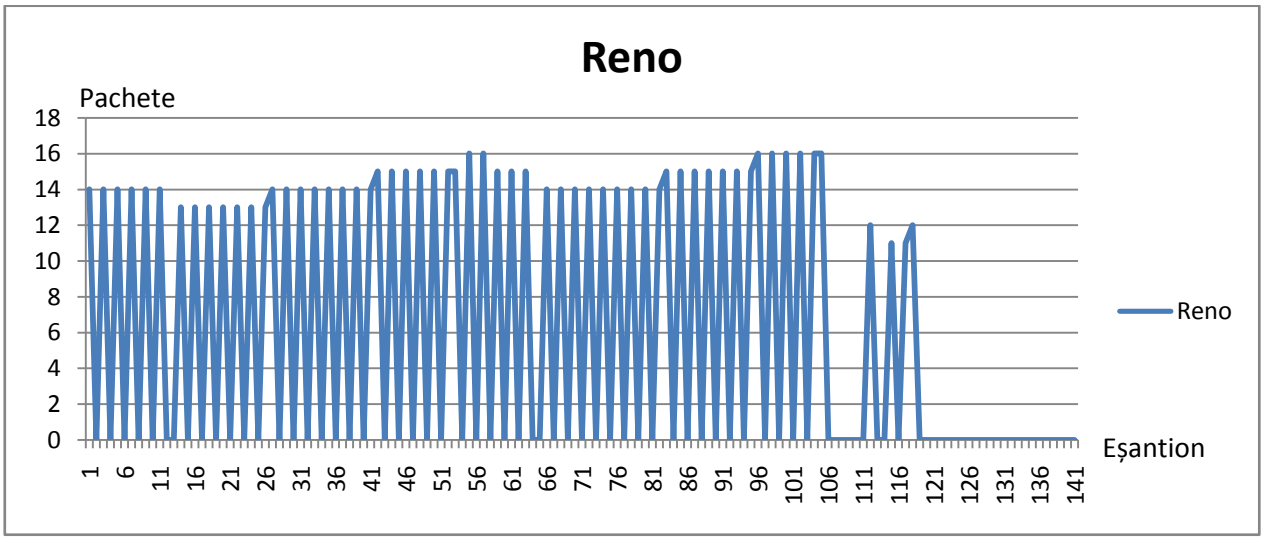

Figura 7.6 – Număr pachete în coada RED (RENO)

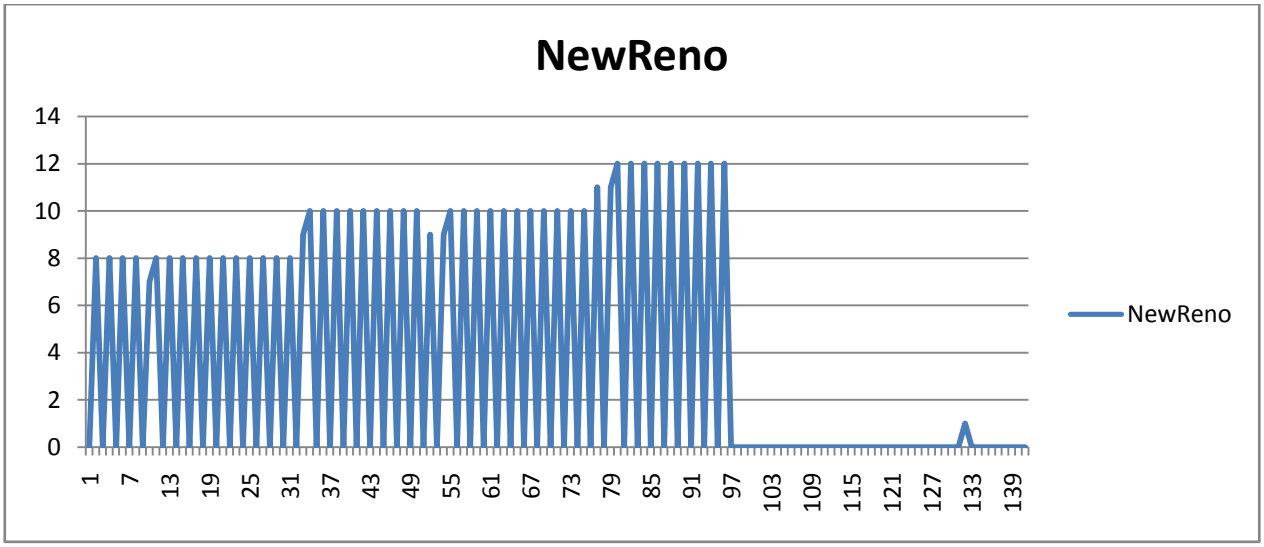

Figura 7.7 – Număr pachete în coada RED (RENO)

După cum se observă, cantitatea cea mai mică de pachete pierdute o are algoritmul Reno urmat de New Reno. Era de așteptat ca algoritmul Tahoe să fie pe ultimul loc, deoarece pierderea este detectată atunci când timpul de timeout expiră înainte ca o confirmare să fie recepționată. În acel moment, Tahoe reduce fereastra de congestie la valoarea maximă a dimensiunii unui segment (MSS – maximum segment size) și se resetează la faza de „slow-start".

8. Concluzii

După cum a fost prezentat și în introducere, scopul acestei lucrări este de a prezenta o altă modalitate de evitare a congestiei în afară de cea cu ajutorul algoritmilor RED și de a fi comparate rezultatele celor 2 metode precizându-se unde este mai eficientă una decât cealaltă și care sunt avantajele.

Metoda ECN propusă a fost implementată direct în codul sursă al programului, iar rezultatele au fost prelucrate astfel încât să poată fi centralizate în graficele ce au fost prezentate în conținutul lucrării. Folosind algoritmii detaliați se poate specifica faptul că la nivel de gateway congestia poate fi prevenită cu ajutorul rezultatelor pe care le primim rulând programele despre valoarea pragurilor ce sunt utilizate în program sau despre valoarea lungimii medie a cozii. Acest fapt se poate realiza folosind marcarea pachetelor ținând cont de starea congestiei la acel moment de timp, iar folosindu-se de acești parametri ruterele își pot reduce numărul de pachete emise.

Algoritmul New Reno este o variantă a algoritmului Reno folosit în desfășurarea acestei lucrări modificarea fiind în cadrul algoritmului de Fast Recovery. Această modificare are rolul de a rezolva problema timeout-ului ce apare la pierderea mai multor pachete într-un RTT(round trip time), dar totuși retransmiterea se face doar pentru un singur pachet.

Din punctul de vedere al pachetelor pierdute diferențele sunt foarte mici între cei doi algoritmi ceea ce este de așteptat deoarece dețin aceleași performanțe și aceiași parametri.

Modificarea făcută la nivelul algoritmului are ca rol o altă modalitate de notificare a congestiei în rețea și anume o notificare explicită a congestiei după cum îi spune și numele. Comportamentul rețelei simulate fiind oarecum același ca și în cazul unei experimentări a congestiei în parametri normali.

9. Bibliografie și referințe

1. Andrew S. Tanenbaum, ”Computer Networks”, ediţia a 4a

2. W. Richard Stevens – “TCP/IP Illustrated” ,Volume 1The Protocols “ Editia a doua , Addison-Wesley, 1994.

3. W. Stevens - TCP Slow Start, Congestion Avoidance,Fast Retransmit, and Fast Recovery Algorithms” Network Working Group , NoAO , 1997

4. Mark Allman, Vern Paxson, W. Richard Stevens (April 1999). "Fast Retransmit/Fast Recovery". *TCP Congestion Control*. IETF. sec. 3.2. RFC 2581. Retrieved 2010-05-01.

5. RFC2481 A Proposal to add Explicit Congestion Notification (ECN) to IP, K. Ramakrishnan and Sally Floyd

6. RFC3168 The Addition of Explicit Congestion Notification (ECN) to IP, K. Ramakrishnan and Sally Floyd

7. Explicit Congestion Notification (ECN) for TCP-IP Joseph Davies Revised October 2006

8. WRED — Explicit Congestion Notification Cisco IOS Release 12.2(8)T

9. Effectiveness of ECN and RED Based Congestion Detection, Yury Puzis and Vasily Tarasov Stony Brook University 2008

10. TCP and Explicit Congestion Notification, Sally Floyd , Lawrence Berkeley Laboratory

11. Measuring the state of ECN Readiness in Servers, Clients an Routers, Steven Bauer and Robert Beverly

12. http://web.opalsoft.net/qos/default.php?p=tcp-60 12.06.2015

13. https://www.nsnam.org/tutorials/NS-3-LABMEETING-1.pdf 10.05.2015

14. http://www.icir.org/floyd/papers/tcp_ecn.4.pdf 11.05.2015

15. http://www.ittc.ku.edu/~jpgs/courses/mwnets/lecture-lab-intro2ns3-display.pdf 05.05.2015

16. http://ns3help.blogspot.in/ 05.05.2015

17. http://ro.wikipedia.org/wiki/Modelul_OSI 05.06.2015

18. http://www.competentedigitale.ro/internet/internet_TCP_IP.html 09.06.2015