**Algoritmi de tratare a congestiei în reţea de tip bucla inchisa**

 Student: Corin-Adrian Chirigiu

 Grupa : 443A

 Profesor coordonator: Conf. Dr.Ing. Ştefan Stăncescu

Contents

[1.Congestia 3](#_Toc410496381)

[1.1 Introducere 3](#_Toc410496382)

[1.2 Clasificarea algoritmilor de control ai congestiei 4](#_Toc410496383)

[2. Algoritmi de control ai congestiei 5](#_Toc410496384)

[2.1 Load Shedding 5](#_Toc410496385)

[2.2 Additive-Increase/Multiplicative-Decrease (AIMD) 6](#_Toc410496386)

[2.3 Random early detection (RED) 7](#_Toc410496387)

[2.2.1 WRED 8](#_Toc410496388)

[2.2.2 ARED 9](#_Toc410496389)

[2.2.3 RRED 12](#_Toc410496390)

[3. Concluzii 14](#_Toc410496391)

[4. Bibliografie 15](#_Toc410496392)

# 1.Congestia

## 1.1 Introducere

**Ce este congestia?**

Congestia reprezinta fenomenul de pierdere a pachetelor, ce apare in momentul in care se doreste transmiterea unui numar de pachete mai mare decat capacitatea maxima a retelei. Congestia este o problema care nici pana in prezent nu are o solutie satisfacatoare, aceasta putand duce la pierderea pachetelor, blocarea conexiunilor noi sau scaderea performantei retelei din cauza timpului de asteptare asociat blocarii. Imaginea de mai jos este o reprezentare grafica a cazului in care poate apare congestia.



 Fig. 1 Fenomenul de congestie (sursa :http://rast.orgfree.com/retele/retele\_2002\_07\_1.jpg)

 Pentru a pastra parametrii de functionare a retelei la nivel optim, nivelul de transport si cel de retea trebuie sa coopereze pentru a se reduce sarcina retelei.

 In principiu pentru a rezolva problema congestie avem 2 optiuni: fie reducem sarcina la nivelul retelei fie marim resursele disponibile. Uneori putem modifica dinamic resursele disponibile atunci cand este vorba de o congestie serioasa a retelei (ex: pornirea unui router de rezerva ).

## 1.2 Clasificarea algoritmilor de control ai congestiei

 Solutiile controlului congestiei in retea pot sa nu tina cont de starea retelei ( caz in care discutam despre un concept bucla deschisa) sau poate urmari starea retelei (caz in care discutam despre conceptul de bucla inchisa). Abordarea cu bucla inchisa se bazeaza pe conceptul de reactie inversa si are 3 parti[[1]](#footnote-2):

1. Monitorizeaza sistemul pentru a detecta cand si unde se produce congestia.
2. Trimite aceste informatii catre locurile unde se pot executa actiuni.
3. Ajusteaza functionarea sistemului pentru a corecta problema.

 Clasificarea algoritmilor poate fi extinsa, astfel algoritmii in bucla deschisa pot fi impartiti in algoritmi ce actioneaza asupra destinatiei si algoritmi ce actioneaza asupra sursei. Algoritmii ce functioneaza in bucla inchisa sunt impartiti in 2 subcategorii: cu feedback explicit si feedback implicit.

 Algoritmii cu feedback explicit functioneaza pe principiul retransmiterii pachetelor din punctul in care s-a produs congestia catre sursa pentru a o avertiza. Algoritmii cu feedback implicit presupun ca sursa sa deduca existenta congestiei pe baza parametrilor de transmitere a pachetelor (ex: timpul pentru intoarcerea confirmarilor).

 In continuare vom analiza 3 algortimi de control ai congestiei : Load Shedding , AIMD si RED (Random Early Detection).

# 2. Algoritmi de control ai congestiei

## 2.1 Load Shedding

 Este un algoritm de bucla inchisa relativ simplu si foarte eficient. Acesta algoritm are la baza o idee foarte simpla, atunci cand apare congestia pachetele sunt eliminate pentru a evita eventuale inconveniente. Modul in care pachetele sunt eliminate depinde foarte mult de domeniul de aplicatie in care este utilizat, de exemplu pentru aplicatii media in timp real pachetele cele mai importante sunt cele “noi”, cu cat trece mai mult timp cu atat valoarea pachetului scade (ex: O videoconferinta cu imaginea video a prezentarii intarziate nu ofera informatii utile). Pe de alta parte daca discutam despre tipul de aplicatii in care pachetele “noi” nu fac nimic altceva decat sa umple buffer-ul sursei (lucru care ar duce la congestie si implicit pierderi si intarzieri) atunci putem elimina pachetele “noi” in favoarea celor “vechi”.

 Se pune insa problema la eliminarea eronata a unor pachete de importanta maxima, aici intervine prioritatea pachetelor. Orice pachet poate contine in header-ul sau informatii privind importanta lui, astfel putem avea pachete importante marcate ca atare, lucru ce ar duce la pastrarea acestora neexistand riscul de a elimina informatii valoroase. Load shedding este deseori utilizat, considerat a fi un “rau” necesar pentru a spori performantele retelei.

 Un principal dezavantaj al acestui algoritm il reprezinta faptul ca nu poate actiona decat in cazul in care congestia este detectata (de obicei consecintele sunt grave la acest nivel), aceasta aparitie a congestiei este un lucru nefericit de aceea se doreste eliminarea pachetelor inainte de efectul ingrijorator al congestiei, aici intervine algoritmul RED.

## 2.2 Additive-Increase/Multiplicative-Decrease (AIMD)

AIMD este un algoritm de control de tip bucla inchisa folosit adesea in tratarea congestiei prin TCP. Principiul de baza al acestui algoritm este de a mari putin (prin adunare) fereastra de congestie din TCP dupa fiecare transmisie a pachetelor, iar atunci cand se detecteaza congestia micsorarea drastica (prin impartire) a acesteia.

Acest mecanism intra in functiune atunci cand este sesizata pierderea de pachete sau se poate detecta cu ajutorul mecanismului Explicit Congestion Notification (ECN). Aceasta lege de control din TCP a devenit unanim acceptata ca fiind cea mai buna metoda de control a congestiei bazandu-se pe o actiune drastica dar foarte eficienta (micsorarea prin impartire) pentru rezolvarea problemei. O mare problema in adoptarea acestei reguli de control a fost interactiunea cu celelalte protocoale de transport care la randul lor au reguli proprii , de aici a derivat notiunea de TCP-friendly prin care celelalte protocoale sunt dezvoltate in asa fel incat sa poata lucra eficient cu TCP (cel mai raspandit protocol). AIMD are (dupa cum se poate vedea in figura de mai jos) o forma de “dinte de fierastrau” datorita regulii de control:



Fig.2 Legea de control AIMD (sursa: Andrew Tanenbaum – "Computer Networks" editia V, Ed. Prentice Hall 2011)

AIMD este foarte eficient in tratarea congestiei cand este vorba de un transfer de date cu un numar foarte mare de pachete, problema se pune atunci cand se doreste utilizarea acestei metode in aplicatii in timp real unde micsorarea drastica a ratei de transfer duce la rezultate nesatisfacatoare. Totusi ramane unul dintre cele mai bune mecanisme de control iar unele variante de AIMD ale caror parametrii de control (rata de crestere si scadere) pot fi modificate dinamic duc la rezultate satisfacatoare.

## 2.3 Random early detection (RED)

Este tot un algoritm bazat pe eliminarea de pachete doar ca spre deosebire de Load Shedding acest algoritm actioneaza inainte ca situatia sa fie grava. El se bazeaza pe monitorizarea cozii router-elor din retea. Este un mecanism de eliminare bazat pe probabilitati, cu cat buffer-ul de la intrarea router-ului este mai plin (coada mai mare) cu atat probabilitatea de eliminare a pachetelor tinde catre 1. Daca buffer-ul este gol atunci toate pachetele noi vor fi acceptate. Astfel se imbunatateste nivelul de control asupra congestiei (fata de Load Shedding).

O mare problema conceptuala a acestui algoritm il reprezinta faptul ca nu se tine cont de calitatea serviciului (QoS), lucru nefericit ce poate duce la eliminarea pachetelor importante. Pentru a evita aceasta situatie neplacuta s-au produs modificari la algoritmul de baza RED si s-au creat cateva variante imbunatatite ale acestuia:

 - WRED (Weighted random early detection) tine cont de prioritatea pachetelor astfel modificandu-se valoarea probabilitatilor de eliminare a pachetelor.

 - ARED (Adapted random early detection) acesta modifica nivelul de interventie a algoritmului RED pe baza analizei dimensiunii medie a cozii. Daca valoarea cozii este in jurul valorii minime atunci detectia timpurie a congestiei este prea “agresiva”, daca dimpotriva valoarea oscileaza in jurul valorii maxime atunci detectia este prea “permisiva”.

- RRED (Robust random early detection) este o adaptare a algoritmului traditional pentru a se proteja impotriva atacurilor DoS(denial-on-service). Acesta imbunatateste foarte mult performantele protocolului de control al transferului.

## 2.2.1 WRED

 WRED (weighted random early detection) a aparut din nevoia de pastrare a pachetelor importante. Algoritmul RED clasic prezenta riscul de eliminare nedorita a pachetelor de mare importanta. Ideea de baza a acestui algoritm RED modificat este de a tine cont de importanta pachetelor in calculul lungimii medie a cozilor, astfel calculandu-se probabilitati diferite pentru pachetele de o anumita importanta. De exemplu o coada poate avea o anumita limita pentru pachete de importanta minora diferita de cea pentru pachetele prioritare astfel putem elimina pachetele de prioritate mai mica si minimizam riscul de a elimina pe cele mai importante. Pasii algoritmului WRED sunt urmatorii[[2]](#footnote-3):

 -Se calculeaza lungimea medie a cozii.

 -Pachetul ajuns este introdus in coada doar daca dimensiunea acesteia este sub limita minima.

 -Depinzand de probabilitatea de eliminare a pachetului, acesta este fie eliminat fie introdus in coada daca dimiensiunea medie a acesteia este cuprinsa intre valorile limita minim si maxim a cozii.

 -Pachetul este automat eliminat daca dimensiunea cozii este mai mare decat limita superioara.

 Calculul dimensiunii medie se face cu ajutorul formulei[[3]](#footnote-4):

 media = oa\*(1-2-n)+cs\*(2-n)

n- factor exponential configurabil de catre utilizator

oa- media veche cs- dimensiunea actuala a cozii

 O caracteristica a WRED este faptul ca eliminarea pachetelor se efectueaza tinandu-se cont de precedenta IP-urilor de provenienta. Pachetele ce provin cu preponderenta de la o anumita adresa IP au o probabilitate mult mai mica de eliminare, de asemenea WRED poate fi modificat pentru a ignora aspectul provenientei lor (se ignora IP-ul de provenienta).

## 2.2.2 ARED

 RED traditional tine cont de 2 praguri (minim si maxim) ale unei cozi si in functie de aceasta se stabileste statistic o rata de eliminare a pachetelor. Problemele apar deoarece intarzierea data de algoritmul RED traditional depind foarte mult de intensitatea traficului si parametrii retelei. In 1997 se propune introducerea unor modificari ale algoritmului RED de catre Feng et al. pe baza caruia in 2001 Sally Floyd, Ramakrishna Gummadi, si Scott Shenker elaboreaza o noua lucrare ce sustine utilizarea algoritmului Adaptive RED.

 Ideea de baza a algoritmului ARED o reprezinta modificarea probabilitatii de eliminare a pachetelor astfel incat lungimea medie a cozii sa fie cuprinsa intre cele 2 praguri. In plus s-au propus o serie de modificari ale ARED traditional pentru a imbunatati performantele[[4]](#footnote-5):

 -Probabilitatea de eliminare a pachetelor se modifica astfel incat lungimea medie a cozii sa fie cuprinsa la jumatatea intervalului min-max.

 -Modificarea probabilitatii de eliminare a pachetelor se modifica treptat si trebuie sa fie pastrata in intervalul [1%,50%].

 -Rata de modificare a probabilitatii de eliminare a pachetelor se modifica nu multiplicativ ci se utilizeaza o politica de crestere prin adunare si scadere prin impartire.

 Algoritmul pentru adaptive RED este urmatorul:



 Fig. 3 Algoritmul ARED (sursa: <http://www.icir.org/floyd/papers/adaptiveRed.pdf>)

minth->pragul de minim al cozii

maxth->pragul de maxim al cozii

avg-> dimensiunea medie a cozii

interval->pasul de prelucrare al algoritmului ARED

target->dimensiunea medie dorita a cozii

maxp->probabilitatea de eliminare maxima a pachetelor

α->constanta de crestere a probabilitatii de eliminare a pachetelor

β->constanta de scadere a probabilitatii de eliminare a pachetelor

 Simulari ale acestui algoritm imbunatatit indica faptul ca modificarea probabilitatii de eliminare a pachetelor duce la o stabilitate sporita a algoritmului RED in fata modificarii fluxului de informatii si a parametrilor de retea. Trebuie sa se tina cont insa de faptul ca aplicarea ARED tine foarte mult de expertiza operatorilor cu reteaua deoarece trebuie sa se ajunga la un echilibru intre accesul la informatie si intarzierea din cauza cozilor.

## 2.2.3 RRED

 Cum s-a prezentat anterior motivatia aparitiei RRED este protectia impotriva atacurilor DoS (denial of service). Un atac DoS este o incercare de a impiedica accesul utilizatorilor la o resursa disponibilia in mod normal lor. In general aceste atacuri sunt utilizate pentru a bloca sau intrerupe accesul la internet intr-o retea. Experimentele au demonstrat ca datorita variatiei lungimii cozilor TCP algoritmul RED este foarte sensibil. Astfel RRED are rolul de a detecta si filtra pachetele care stau la baza unui atac DoS inainte ca acestea sa fie supuse algoritmului RED. Ideea din spatele RRED este ca atunci cand se trimite un pachet la scurt timp dupa ce un altul a fost eliminat noul pachet este suspectat ca facand parte dintr-un atac tip DoS.

 Algoritmul RRED[[5]](#footnote-6):

RRED-ENQUE(pkt)

01 f←RRED-FLOWHASH(pkt)

02 Tmax←MAX(Flow[f].T1, T2)

03 if pkt.arrivaltime is within [Tmax, Tmax+T\*] then

04 reduce local indicator by 1 for each bin corresponding to f

05 else

06 increase local indicator by 1 for each bin of f

07 Flow[f].I←maximum of local indicators from bins of f

08 if Flow[f].I >=0 then

09 RED-ENQUE(pkt) //pass pkt to the RED block

10 if RED drops pkt then

11 T2←pkt.arrivaltime

12 else

13 Flow[f].T1←pkt.arrivaltime

14 drop(pkt)

15 return

f.T1 este timpul la care a ajuns ultimul pachet eliminat de filtru de detectie din fluxul f

T2 este timpul la care a ajuns ultimul pachet eliminat de RED

Tmax=max(f.T1,T2)

T\* este o fereastra de timp scurt determinata empiric ca fiind 10ms intr-un algoritm RRED traditional

pkt-> pachetul analizat

pkt.arrivaltime-> timpul la care a ajuns pachetul analizat

RED-ENQUE->functia filtrului RRED

# Concluzii

O data cu utilizarea intensa a retelelor de calculatoare si a aplicatiilor ce tin de interactiunea cu acestea se observa o importanta deosebita a evitarii intarzierilor si a pierderii de pachete. Astfel congestia a devenit una dintre cele mai importante probleme de tratat, care din nefericire in marea majoritate a cazurilor nu poate fi evitata. Abundenta de algoritmi de combatere a congestiei reprezinta o unealta importanta in lupta impotriva acesteia, dar fiecare algoritm presupune avantaje si dezavantaje.

Am analizat 3 tipuri de algoritmi Load Shedding,AIMD si RED (cu variantele acestuia) toate se bazeaza pe o analiza a starii retelei si pe eliminarea de pachete dupa anumite criterii. Se observa ca toti algoritmi prezinta anumite dezavantaje (Load Shedding prezinta imposibilitatea de predictie, AIMD este ineficient in aplicatiile in timp real, RED este sensibil la atacuri si la eliminarea pachetelor importante) dar cu toate acestea cercetarea din domeniu si imbunatatirea algoritmilor ii fac pe zi ce trece din ce in ce mai puternici si mai stabili.

Beneficiind de aceste unelte pentru combaterea congestiei responsabilitatea din urma cade asupra administratorului/operatorului de retea care trebuie sa ia decizii in functie de necesitati asupra algoritmilor implementati si asupra parametrilor acestora. Sarcina ii revine acestuia din urma pentru a gasi echilibrul perfect intre castig si pierdere pentru a asigura o functionare cat mai buna a retelei si a asigura buna functionare a acesteia.

# Bibliografie

Stefan Stancescu – Note de curs

Andrew Tanenbaum – "Retele de calculatoare" editia IV, Ed. Byblos 2003

<http://www.icir.org/floyd/papers/adaptiveRed.pdf>

http://rast.orgfree.com/retele/

http://en.wikipedia.org/wiki/Weighted\_random\_early\_detection

http://www.cisco.com/c/en/us/td/docs/ios/12\_0s/feature/guide/fswfq26.html

<http://en.wikipedia.org/wiki/Robust_random_early_detection>

<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5456075>

<http://people.mpi-inf.mpg.de/~akessel/papers/aimd.pdf>

http://en.wikipedia.org/wiki/Additive\_increase/multiplicative\_decrease

1. Andrew Tanenbaum – "Retele de calculatoare" editia IV, Ed. Byblos 2003 [↑](#footnote-ref-2)
2. <http://en.wikipedia.org/wiki/Weighted_random_early_detection> [↑](#footnote-ref-3)
3. <http://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fswfq26.html> [↑](#footnote-ref-4)
4. http://www.icir.org/floyd/papers/adaptiveRed.pdf [↑](#footnote-ref-5)
5. <http://en.wikipedia.org/wiki/Robust_random_early_detection> [↑](#footnote-ref-6)