Proiect Reţele de Calculatoare

Tema:

Aplicaţie pentru transfer de fişiere – implementare printr-un mecanism de control al congestiei

Realizat de:

Anițoaei Teodor – 1305B

Stanciu Ioan – 1305B

Protocolul TCP

Protocolul TCP (Transmission Control Protocol) este unul dinte protocoalele nivelului Internet al modelului TCP/IPTCP a fost special conceput pentru a oferii un transportde biţi de tip capăt-la-căpat, fără erori sau pierderi de date. El a fost definit în standardul RFC 793 (Request For Comment) scris în 1981.

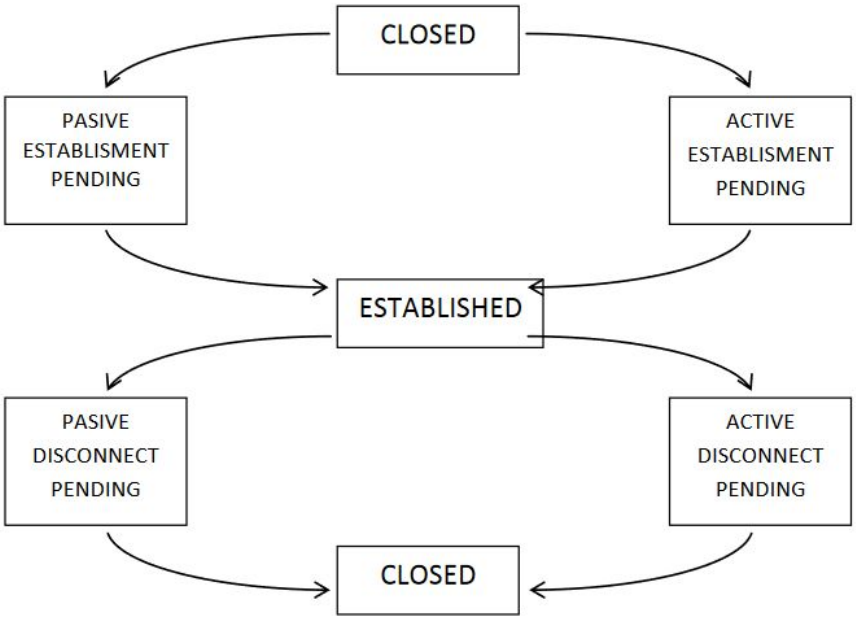
O reţea de internet diferă de una globală deoarece diferite păţi pot avea topologii, laţimi de bandă, timpi de întârziere diferiţi, dimeniuni ale pachetelor sau alţi parametrii diferiţi. Astfel TCP a fost astfel conceput să se adapteze la proprietăţile diferite ale componentelor folosite în cadrul reţelei.

Protocolul TCP reprezintă unul din compnentele modelul TCP/IP (Protocol de control al transmisiei/ Protocol Internet) care a fost creat de US DoD (Departamentul de aparare al Statelor Unite) din necesitatea unei reţele care ar putea supravieţui în orice condiţii. Scopul reţelei TCP/IP era ca orice conexiune s-ar rupe în interiorul reţelei reţeaua în ansamblu să rămână intacta

Protocolul TCP este un protocol orientat pe conexiune care permite ca un flux de octeţi transmişi să ajunga la destinaţie ,fără erori, la orice altă maşină din cadrul aceleiaşi reţea.

Serviciul TCP trebuie asigurat atat de emitator cat si de receptor deoarece se bazează pe principiul capăt-la-capăt şi foloseşte socketi. Fiecarui socket îi corespunde un număr cu rol de adresă numit adresa IP adestinatarului si un alt numar alcatuit din 16 biti numit port. Pentru ca serviciul TCP să fie obţinut trebuie realizată explicit o conexiune între maşina care transmite mesajul şi cea care îl recepţionează. Simplificat o conexiune pate fi reprezentatastfel:

Astfel TCP poate fi comparat cu o conexiune telefonica din punct de vedere al modului în care utilizatorul vede această conexiune. Diferenţa de baza este dată de faptul că mesajele sunt transmise prin intermediul pachetelor fără a bloca o linie telefoincă în cazul netransmiterii de informaţie.



Astfel un anumit port se poate afla în diferite stări, cum ar fi IDLE, ESTABLISHED sau stari intermediare în cadrul cărora se realizează stabilirea conexiunii sau deconectarea. Conexiunea presupune 3 faze: realizarea conexiunii (IDLE->ESTABLISHED), transmiterea informației (starea ramane ESTABLISED) și intreruperea conexiunii (ESTABLISHED->IDLE).

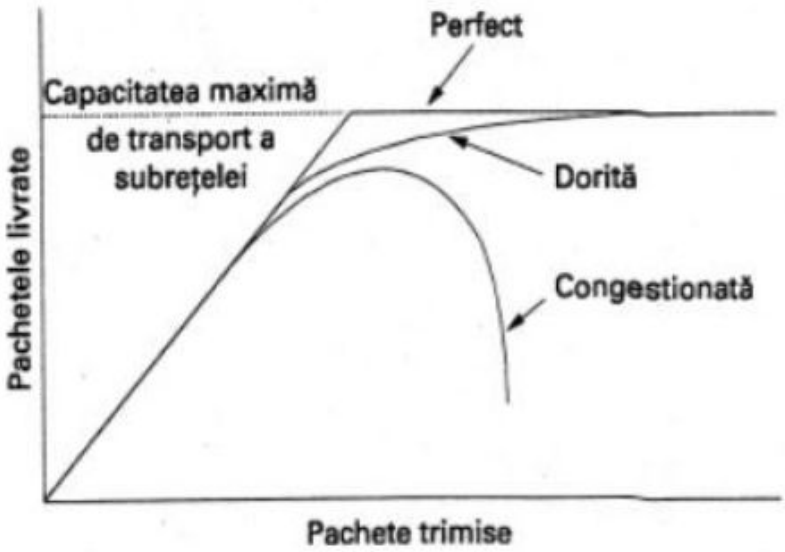
Un socket poate fi folosit pentru mai multe conexiuni simultane. Astfel 2 sau mai multe conexiuni se pot termina la același socket. Acestea sunt identificate prin identificatori de socket aflati la ambele capete.

Congestia retelei

Congestia apare in rețelele de calculatoare atunci când incărcarea depășește posibilitățile acesteia de a transfera datele. Incărcarea unei rețele la un moment dat nu depinde numai de

capacitatea ei de transmitere ci și de erorile care apar în mediul de transmisie, de viteza de prelucrare in noduri și de mecanismele de confirmare folosite de receptor.

Controlul acestui fenomen intră in responsabilitatea nivelelor de transport și de rețea. Apare ca rezultat al traficului intens la nivelul transport, propagându-se la nivelul rețea, performanțele totale ale sistemului degradându-se prin intârzierea și pierderea pachetelor de date.



[5]

Congestia apare ca rezultat al mai multor factori. In situația in care la sosirea unui număr mare de pachete provenind de pe mai multe linii de intrare intr-o singură linie de ieșire, atunci se va forma o coada. Pentru a le păstra pe toate, sistemul necesită memorie, iar creșterea acestei capacități de memorare duce la inrăutățirea congestiei și nu la ameliorarea ei.

Un alt factor care cauzează congestia este reprezentat de viteza procesoarelor. Dacă unitatea centrală a ruterelor este lentă in execuția funcțiilor sale, cozile pot crește. Și liniile cu lățime de bandă scăzuta pot provoca congestia. Schimbarea liniilor cu unele mai performante și păstrarea aceluiași procesor sau invers ajută puțin, deoarece problema ține de incompatibilitatea intre părțile sistemului și aceasta va persista până la aducerea la echilibru a tuturor componentelor.

Controlul congestiei trebuie să asigure capabilitatea rețelei de a transporta intreg traficul implicat. Este o problemă globala care implică comportamentul tuturor calculatoarelor gazdă și al ruterelor.

Algortimul BITCP

Binary Increase TCP este un algoritm de control al congestiei care adaptează controlul ferestrei in funcție de dimensiunea acesteia. Acesta constă in două părți: căutare prin creștere binară (binary search increase) și creșterea aditivă a dimensiunii ferestrei (additive increase).

Binary search increase

In această etapă privim controlul congestiei ca pe o problemă de căutare in care sistemul poate răspunde prin da sau nu atunci când vrem să stim dacă rata de transmisiune curentă depășește sau nu capacitatea rețelei. Fereastra minimă curentă poate fi estimată ca dimensiunea ferestrei atunci când nu avem pierderi.

Dacă se ştie dimensiunea maximă a ferestrei, putem să aplicăm o tehnică de căutare binară pentru a seta dimensiunea ferestrei curente la dimensiunea medie dintre minimul estimat şi maximul cunoscut. Dacă această dimensiune dă pierderi,noua fereastră poate fi tratată ca un nou maxim, iar dimensiunea ferestrei după pierderea pachetelor devine noul minim. Media acestor valori devine noul optim.

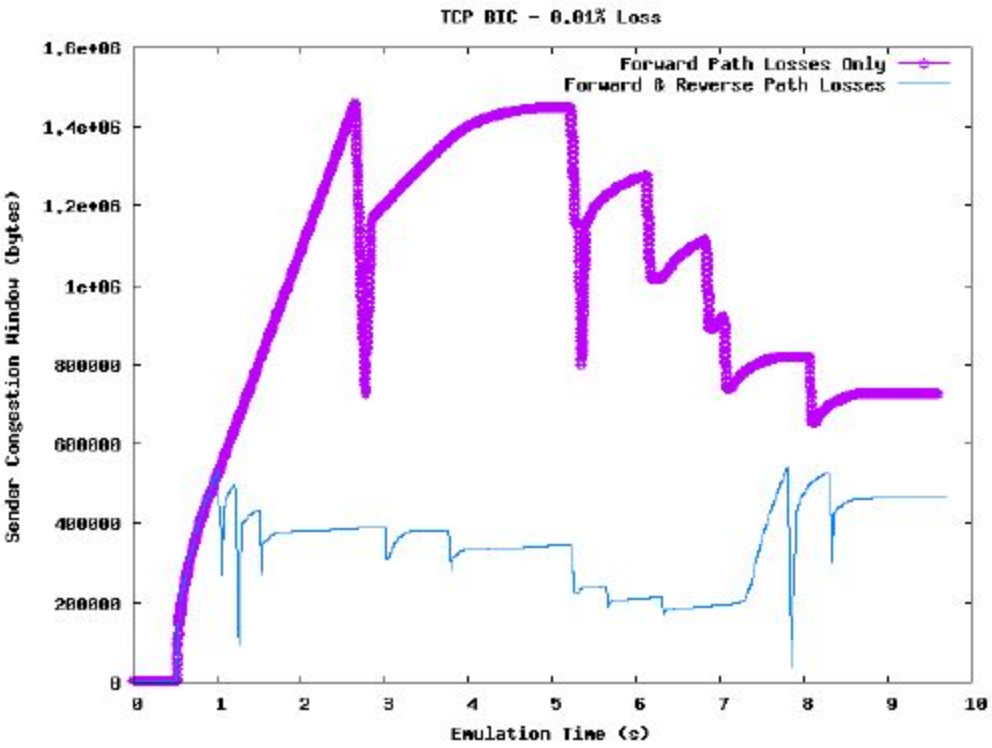
Acest proces de recalculare a minimului şi maximului are loc până când diferența dintre dimensiunea maximă şi cea minimă scade sub un nivel prestabilit numit incrementare minimă Smin.

Additive increase

Pentru a asigura o convergență rapidă a algoritmului şi corectitudinea, se combină căutarea binară cu o strategie de creştere aditivă. Când distanța dintre dimensiunea medie şi minimul curent este foarte mare, creşterea dimensiunii ferestrei direct la cea medie este un factor de risc in rețea. Astfel, când distanța dintre dimensiunea ferestrei curente şi cea medie este mai mare decât o valoare presetată, numită increment maxim Smax, in loc să creştem dimensiunea ferestrei direct la acea valoare medie in următorul RTT, o să o creştem cu Smax până când distanța va deveni mai mica decat Smax, moment in care dimensiunea va fi setata direct la valoarea medie.

Multiplicative decrease

La fel ca în cazul altor algoritmi de control al congestiei, BITCP foloseşte tehnica multiplicative decrease în cazul în care au fost detectate pachete lipsă în timpul comunicației. De asemenea, dimensiunea curentă a ferestrei de congestive va deveni noul maxim în pasul de căutare binară. Acest lucru se traduce într-o convergență mai rapidă a aflării limitei canalului de comunicație.



Evoluția ferestrei de congestie în cazul algoritmului BITCP

Implementare:

Se folosesc urmă torii parametrii presetați:

low_window – acest algoritm este activat in momentul in care dimensiune ferestrei curente depaseste aceasta valoare

Smax – increment maxim

Smin – increment minim

β – factorul de scadere multiplicativa a dimensiunii ferestrei

default_max_win – maximul presetat

Se folosesc urmă toarele variabile:

max_win – dimensiunea maxima a ferestrei care la inceput a valoarea presetata

min_win – dimensiunea minima a ferestrei

prev_win – dimensiunea maxima chiar inainte de setarea nouui maxim

target_win – media dintre minim si maxim cwnd – dimeniunea ferestrei de congestie

is_BITCP_ss – variabila booleana care indica daca suntem sau nu in etapa slow start. Se initializeaza cu false.

ss_cwnd – o varibila care stocheaza cu cat se creste cwnd curent in etapa slow start

ss_target – stocheaza fiecare valoare a cwnd dupa fiecare RTT in etapa slow start

Algoritmul BI-TCP:

Când se intră in fast recovery:

```
if(low_window<=cwnd){
        prev_max=max_win;
        max_win=cwnd;
        cwnd=cwnd*(1-β);
        min_win=cwnd;
         if(prev_max>max_win)  max_win=(max_win+min_win)/2;
         target_win=(max_win+min_win)/2;
} else { cwnd=cwnd*0.5; }
```

Cand nu se află in fast recovery şi ajunge un ACK pentru un nou pachet:

```
if(low_window>cwnd) { cwnd=cwnd+1/cwnd; return;}
if(is_BITCP_ss is false )
        { if(target_win-cwnd<Smax) cwmd+= (target_win - cwnd)/cwnd;
        if(max_win>cwnd)
```

```
        {
        min_win=cwnd;
        target_win= (max_win+min_win)/2; }
         else
         { is_BITCP_ss=true;
         ss_cwnd=1;
         ss_target=cwnd+1;
        max_win=default_max_win;
         } } else {
         cwnd+=ss_cwnd/cwnd;
        if(cwnd>=ss_target)
        {
                ss_cwnd=2*ss_cwnd;
                ss_target+=ss_cwnd;
         }
         if(ss_cwnd>=Smax) is_BITCP_ss=false;
}
```

Pentru realizarea aplicației, vom folosi:

limbajul de programare Python

biblioteca socket pentru comunicarea între cele două entități

biblioteca tkinter pentru realizarea interfețelor grafice

Bibliografie:

https://docs.python.org/3/library/socket.html

https://wiki.python.org/moin/UdpCommunication#Using_UDP_for_e.g._File_Transfers

https://squidarth.com/demonstrating-congestion-control.html

http://stst.elia.pub.ro/news/RC/Teme_RC_IVA_2011_12/Lecu%20tica%20vidrascu%20442A%20Algoritmi%20de%20control%20al%20congestiei%20.pdf?fbclid=IwAR3OHA-q6KH6AmuWVXcxlsik7QOrazHCSRY8LxOyB7PI_EGtl720u2OKhO0

http://stst.elia.pub.ro/news/RC/Teme_RC_IVA_2012_13/1_SoareBo_TCP%20CONTROLUL%20CONGESTIEI.pdf

https://rcp.atlassian.net/secure/RapidBoard.jspa?projectKey=PR&rapidView=1