



# Proiect Rețele de Calculatoare

## **Tema:**

Aplicație pentru transfer de fișiere – implementare printr-un mecanism de control al congestiei

## **Realizat de:**

Anițoaiei Teodor – 1305B

Stanciu Ioan – 1305B

## Protocolul TCP

Protocolul TCP (Transmission Control Protocol) este unul dintre protocoalele nivelului Internet al modelului TCP/IP. A fost special conceput pentru a oferi un transport de biți de tip capăt-la-căpăt, fără erori sau pierderi de date. El a fost definit în standardul RFC 793 (Request For Comment) scris în 1981.

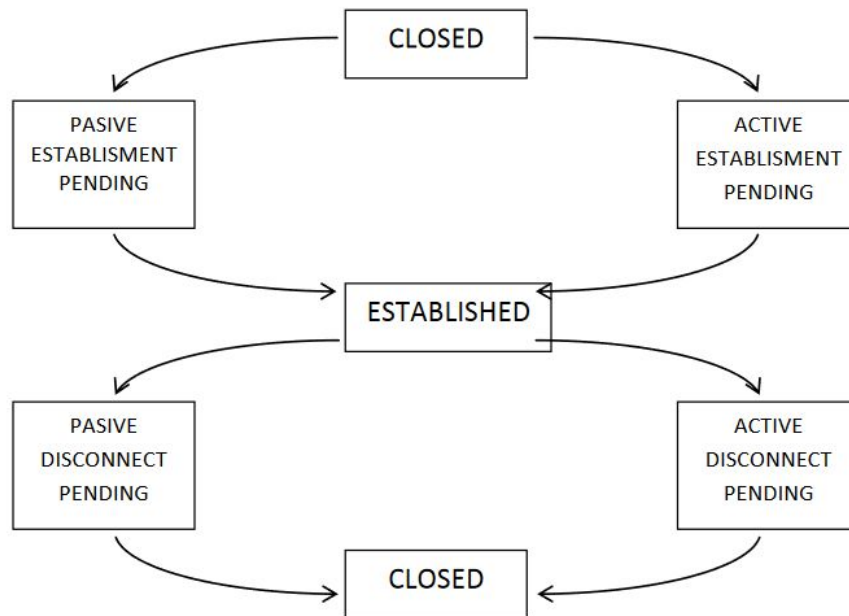
O rețea de internet diferă de una globală deoarece diferite părți pot avea topologii, lățimi de bandă, timpi de întârziere diferiți, dimensiuni ale pachetelor sau alți parametri diferiți. Astfel TCP a fost astfel conceput să se adapteze la proprietățile diferite ale componentelor folosite în cadrul rețelei.

Protocolul TCP reprezintă unul din componentele modelului TCP/IP (Protocol de control al transmisiei/ Protocol Internet) care a fost creat de US DoD (Departamentul de apărare al Statelor Unite) din necesitatea unei rețele care ar putea supraviețui în orice condiții. Scopul rețelei TCP/IP era ca orice conexiune să nu se rupe în interiorul rețelei; rețeaua în ansamblu să rămână intactă.

Protocolul TCP este un protocol orientat pe conexiune care permite ca un flux de octeți transmiși să ajungă la destinație, fără erori, la orice altă mașină din cadrul aceleiași rețele.

Serviciul TCP trebuie asigurat atât de emitator cât și de receptor deoarece se bazează pe principiul capăt-la-căpăt și folosește socket-uri. Fiecarui socket îi corespunde un număr cu rol de adresă numit adresă IP a destinatarului și un alt număr alcătuit din 16 biți numit port. Pentru ca serviciul TCP să fie obținut trebuie realizată explicit o conexiune între mașina care transmite mesajul și cea care îl recepționează. Simplificat o conexiune poate fi reprezentată astfel:

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



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

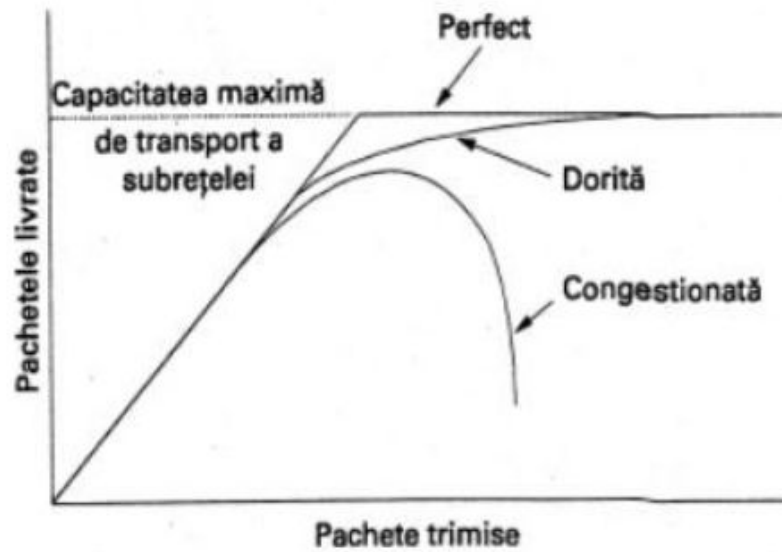
Un socket poate fi folosit pentru mai multe conexiuni simultane. Astfel 2 sau mai multe conexiuni se pot termina la același socket. Acestea sunt identificate prin identificatori de socket aști la ambele capete.

## Congestia rețelei

Congestia apare în rețelele de calculatoare atunci când încărcarea depășește posibilitățile acestora 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 în noduri și de mecanismele de confirmare folosite de receptor.

Controlul acestui fenomen intră în 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 întârzierea 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 înrăutățirea congestiei și nu la ameliorarea ei.

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

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

## Algoritmul BITCP

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

### *Binary search increase*

În această etapă privim controlul congestiei ca pe o problemă de căutare în 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 știe dimensiunea maximă a ferestrei, putem să aplicăm o tehnică de căutare binară pentru a seta dimensiunea ferestrei curente la dimensiunea medie dintre minimul estimat și 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.

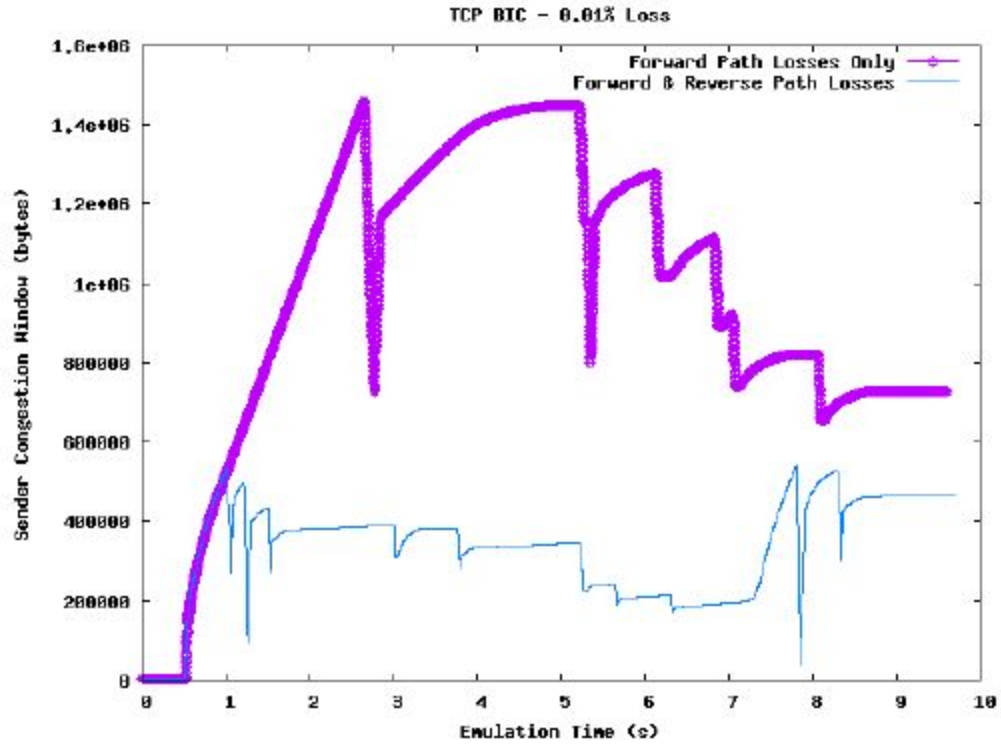
Acest proces de recalculare a minimului și maximului are loc până când diferența dintre dimensiunea maximă și cea minimă scade sub un nivel prestabilit numit incrementare minimă  $S_{min}$ .

### *Additive increase*

Pentru a asigura o convergență rapidă a algoritmului și corectitudinea, se combină căutarea binară cu o strategie de creștere aditivă. Când distanța dintre dimensiunea medie și minimul curent este foarte mare, creșterea dimensiunii ferestrei direct la cea medie este un factor de risc în rețea. Astfel, când distanța dintre dimensiunea ferestrei curente și cea medie este mai mare decât o valoare presetată, numită increment maxim  $S_{max}$ , în loc să creștem dimensiunea ferestrei direct la acea valoare medie în următorul RTT, o să o creștem cu  $S_{max}$  până când distanța va deveni mai mică decât  $S_{max}$ , moment în care dimensiunea va fi setată direct la valoarea medie.

### *Multiplicative decrease*

La fel ca în cazul altor algoritmi de control al congestiei, BITCP folosește tehnica multiplicativă de 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 următoarele parametrii presetate:

smax - incrementul maxim

smin - incrementul minim

wmax - dimensiunea maximă a ferestrei

beta - factor de scădere multiplicativă

cwnd - dimensiunea ferestrei de congestie

bic\_inc - incrementul ferestrei la fiecare RTT



La fiecare pas de marire al cwnd:

```
if (cwnd < wmax)           // binary search OR additive

    bic_inc = (wmax - cwnd) / 2;

else                          // slow start OR additive

    bic_inc = cwnd - wmax;

if (bic_inc > Smax)          // additive

    bic_inc = Smax;

else if (bic_inc < Smin) // binary search OR slow start

    bic_inc = Smin;

cwnd = cwnd + (bic_inc / cwnd);
```

La fiecare pas al scaderii cwnd:

```
if (cwnd < wmax) // fast convergence

    wmax = cwnd * (2-β) / 2;

else

    wmax = cwnd;

cwnd = cwnd * (1-β);
```

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://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-q6KH6AmuWVXcxIsik7QOrzhHCSRY8LxOyB7PI\\_EGtl720u2OKhO0](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-q6KH6AmuWVXcxIsik7QOrzhHCSRY8LxOyB7PI_EGtl720u2OKhO0)

[http://stst.elia.pub.ro/news/RC/Teme\\_RC\\_IVA\\_2012\\_13/1\\_SoareBo\\_TCP%20CONTROLUL%20CONGESTIEI.pdf](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>



Facultatea de Automatica si Calculatoare