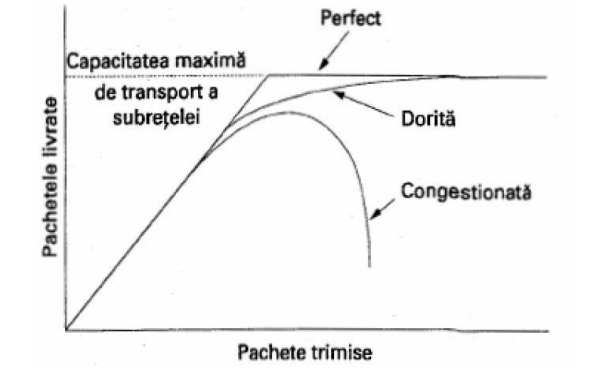
**Transferul fișierelor – controlul congestiei**



Graficul prezent apare in majoritatea documentelor bazate pe “Computer Networks” atunci când se dezbate tema partajării informațiilor, congestiei și cum poate fi aceasta controlată.

Când vine vorba de transfer masiv de fişiere între două sau mai multe calculatoare (care implicit, fac parte dintr-o rețea de calculatoare), ne dorim ca timpul așteptat pentru a avea acces la fișierele dorite sa fie cât mai nesemnificativ.

**Info just in case (baza):**

În timpul transmisiei de la un calculator-sursă la un calculator-destinație, datele suferă o serie de modificări:

* datele sunt transformate în caractere alfanumerice, apoi sunt împărțite în segmente, care sunt mai ușor de manevrat.
* fiecărui segment îi corespunde un antet (header) care conține informații suplimentare (de ex. IP-ul calculatorului sursa-destinație) => segmentul modificat se numește **pachet.**
* fiecărui pachet îi corespunde un al doilea antet care conține adresa MAC a PC-ului sursă-destinație => cadru, care circulă prin mediul de transmisie sub forma unui flux de biți.

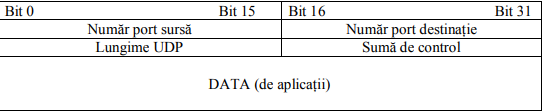
Ajunse la calculatorul-destinație, aceste șiruri de biți suferă procesul invers pentru a ajunge într-o formă ce poate fi citita de utilizator.

**UDP (User Datagram Protocol)**

Este un protocol nesigur, fără conexiuni, destinat aplicaţiilor care doresc să utilizeze propria lor secvenţiere. Cu alte cuvinte, atunci când o aplicaţie are de transmis date, aceasta trimite pur şi simplu acele date.

Un client trimite o cerere scurtă spre server şi aşteaptă un răspuns scurt. Dacă acesta nu vine într-un timp aşteptat, atunci repetă cererea. UDP nu ține cont de ordinea în care au fost transmise pachetele, iar cele pierdute nu se retransmit. Atunci când datele ajung la destinație, în funcție de importanță, aceasta va trebui să identifice secvența corectă.

Antetul pentru UDP:



Revenind la subiectul principal, atunci când se doreşte trimiterea unui volum mare de date pe o legătură cu o capacitate redusă, rețeaua devine supraîncărcată, ca mai apoi să apară riscul de a pierde pachete. Acest proces definește, pe scurt, congestia.

Apar, așadar, numeroși algoritmi pentru controlul fenomenului nedorit descris mai sus: de ex. algoritmul TCP Reno, cel pe care îl vom aplica și noi în cadrul proiectului.

TCP sugerează diferite scheme de control a congestiei folosind fereastra de congestie (cwnd) și tehnici de întârziere. Cu toate acestea, scopul este de a crește dimensiunea a cwnd , fără pierderi de pachete.

TCP folosește un mecanism numit „**slow start**” ( start lent ) pentru a mări fereastra de congestie, dacă o conexiune este inițializată după un timeout. Valoarea inițială a cwnd este 1. Receptorul trimite o confirmare (ACK) pentru fiecare segment, iar de fiecare dată când un ACK este recepționat de către expeditor, fereastra de congestie este crescută cu 1 segment. Dacă un ACK recunoaște două segmente, cwnd este incrementat cu doar 1 segment.

Când fereastra de congestie depășește un prag „ssthresh” algoritmul intră într-o stare nouă, numită **evitarea congestiei**. În unele implementări, „ssthresh” inițial 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 cele mai multe ori un început lent ulterior declanșat de timeout.

**Evitarea congestiei:** Atâta timp cât ACK-uri non-duplicate sunt primite, fereastra de congestie este aditivă, crescând cu un SMS de fiecare dată când este efectuat un ciclu dus-întors. Când un pachet este pierdut, probabilitatea de a fi primit ACK duplicat este foarte mare. În cazul în care sunt primite 3 ACK-uri duplicate, Reno va înjumătăți fereastra de congestie, va stabili pragul de pornire lent egal cu noua fereastra de congestie, efectuând o retransmitere rapidă și intră într-o fază de recuperare rapidă.

**Retransmitere rapidă:** Un expeditor TCP folosește un cronometru pentru a recunoaște segmente pierdute, în urma expedierii așteptând o confirmare (ACK) de la aplicația care recepționează datele. Dacă această confirmare nu vine într-un interval de timp prestabilit, expeditorul va presupune ca segmentul a fost pierdut în rețea, iar datele sunt transmise din nou.

Deoarece datele sunt transmise in blocuri, numai numărul de secvența al primului octet este trimis calculatorului-destinație. După cum am amintit mai sus, aplicația TCP destinație folosește numerele de secvența pentru a le ordona atunci când sosesc neordonate si sa elimine segmentele duplicate.

Cum funcționează, de fapt, **Fast-Retransmit**?

Dacă un receptor primește un segment de date care nu este în regulă, trimite imediat expeditorului o confirmare duplicată. Dacă expeditorul primește trei confirmări duplicate, presupune că segmentul de date indicat de confirmări este pierdut și retransmite imediat segmentul pierdut.

Algoritmul de retransmisie rapida consideră că un pachet a fost pierdut atunci când primește 3 ACK duplicat, înainte de expirarea timerului de retransmisie.

**Fast Recovery:** Deci, în cazul în care sunt recepționate trei confirmări duplicate, Reno înjumătățește fereastra de congestie, efectuează o retransmitere rapidă (fast retransmit) și intră în recuperare rapidă (fast recovery).

In această stare, TCP Reno utilizează retransmiterea rapidã, urmatã de evitarea congestiei. Practic, retransmite pachetul care lipsește, semnalat in urma celor 3 ACK duplicate și așteaptă o confirmare a întregii ferestre 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.

Pentru controlul congestiei TCP se definesc următorii parametri:

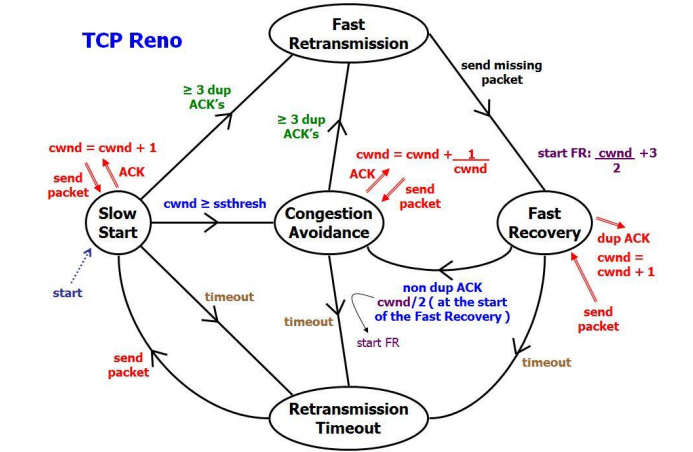
* dimensiunea maximă a segmentului expeditor (SMS-uri) reprezintă suma maximă de date care pot fi trimise într-un singur segment TCP, fără a include antetul.
* fereastra expeditorului (swnd) reprezintă numărul maxim de bytes care poate fi trimiși. Valoarea sa are valoarea cea mai mică dintre fereastra receptor și fereastra de congestie.
* fereastra destinatarului (rwnd) este cel mai nou fereastra de publicitate de către receptor.
* fereastra de congestie (cwnd) este o variabilă de stare TCP, care limitează cantitatea de date care pot fi trimise.
* fereastra pentru pierderi (lw) este valoarea ferestrei de congestie după o pierdere de pachete a fost detectatã.
* prag slow-start (ssthresh) este o altă variabilă TCP, care determină ca algoritmul de control al congestiei sa fie declansat sau.

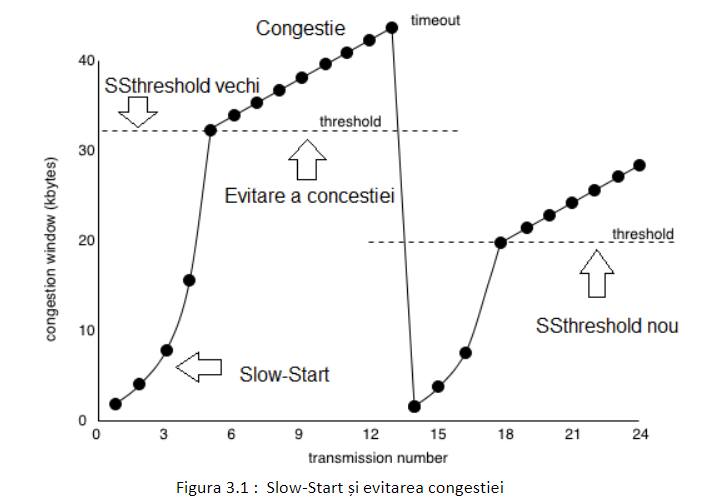
După punerea în aplicare a algoritmului , noi probleme apar.

Prima este legata la detectarea pierderii de pachete. În mod normal, o pierdere de pachete se deduce pe baza unei expirări a timer-ului de retransmisie. Aceasta poate duce la întârzieri semnificative în transmisii de date.

A doua problemă este legată de scăderea drastică a ferestrei de congestie după o detectare a pierderilor de pachete.

Mai jos, am găsit o schemă interesantă:





Ce incercam sa urmarim in implementarea aplicatiei?

Etapele proiectului:

-

-

-

-

Concluzii: