Proiect Curs

Implementarea unui algoritm de control al congestiei

TCP Vegas

TCP Vegas a fost introdus în 1994 ca o alternativă al mecanismului de control al congestiei bazat pe sursă pentru Internet. Vegas anticipează apariția congestiei monitorizând diferența dintre rata pe care o așteaptă să o vadă și rata pe care o realizează în realitate. Strategia Vegas constă în ajustarea ratei de expediere a sursei (fereastra de congestie) în încercarea de a menține un număr mic de pachete în coada de așteptare, de-a lungul căii de transmitere.

TCP Vegas identifică congestionarea la o etapă incipientă pe baza creșterii valorilor timpului de round-trip (RTT) ale pachetelor din conexiune. Algoritmul se bazează în mod semnificativ pe calculul precis al valorii de bază a RTT-ului. Dacă această valoare este prea mică, capacitatea conexiunii va fi mai mică decât lățimea de bandă disponibilă, în timp ce, dacă este prea mare, va depăși capacitatea conexiunii.

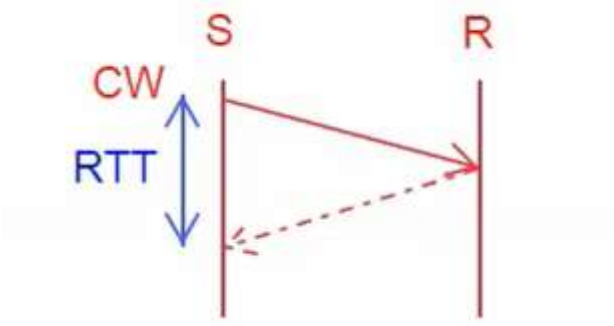
Calcularea RTT-ului se face de la momentul trimiterii unui pachet și primirea unui ACK (acknowledgement) înapoi.

S – sender

R – receive

CW – fereastra glisantă curentă

RTT – timpului de round-trip



Pentru controlul congestiei, TCP Vegas pleacă de la următoarea formulă:

Diff = ((CW / BaseRTT) - (CW / RTT)) * BaseRTT, unde:

Diff – diferența dintre timpul de round-trip așteptat (BaseRTT) și timpul de round-trip real în funcție (RTT), de dimensiunea ferestrei glisante (CW)

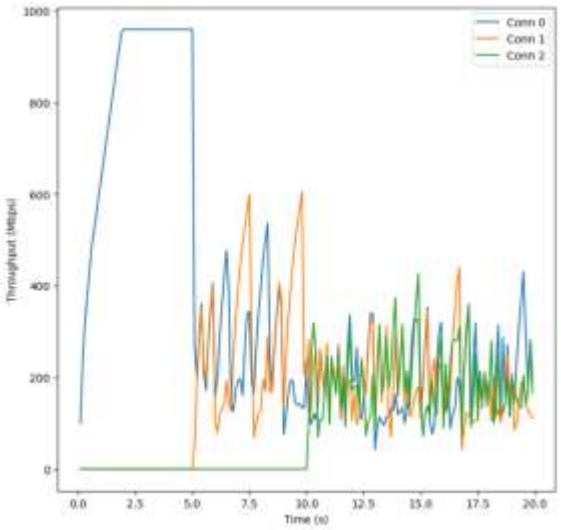
Rezultatul obținut este comparat cu două praguri, numite alpha și beta, prin intermediul cărora algoritmul va decide dacă este posibilă creșterea ferestrei glisante astfel:

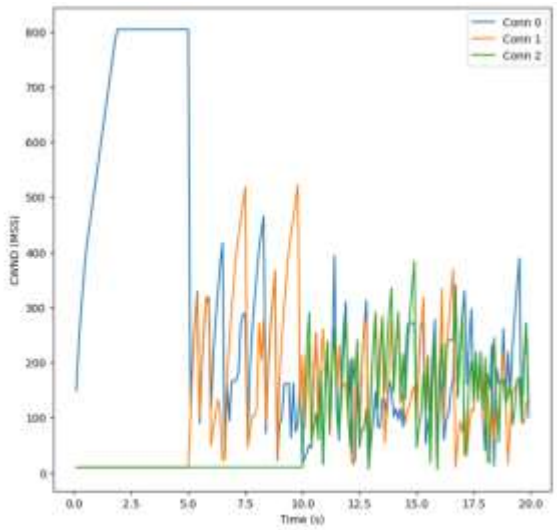
- if Diff < alpha, atunci CW += MSS
- if Diff > beta, atunci CW -= 2 * MSS, unde:

MSS – unitatea de segment

Dacă este congestie (prea multe pachete în coadă), RTT-ul devine prea mare, astfel crescând Diff. Atunci, CW scade. Acest lucru este activat de către pragul beta.

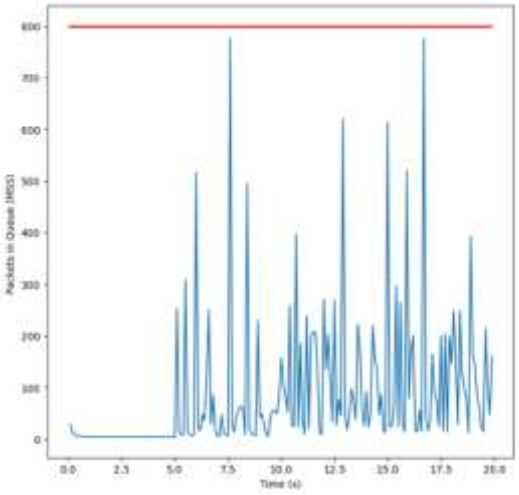
Dacă nu este congestie (coada e goală), RTT-ul devine mic (la fel și Diff). Astfel, CW trebuie să crească pentru a evita riscul de a nu utiliza toata lățimea de bandă, în integritate. Această situație se petrece în cazul activării pragului alpha.





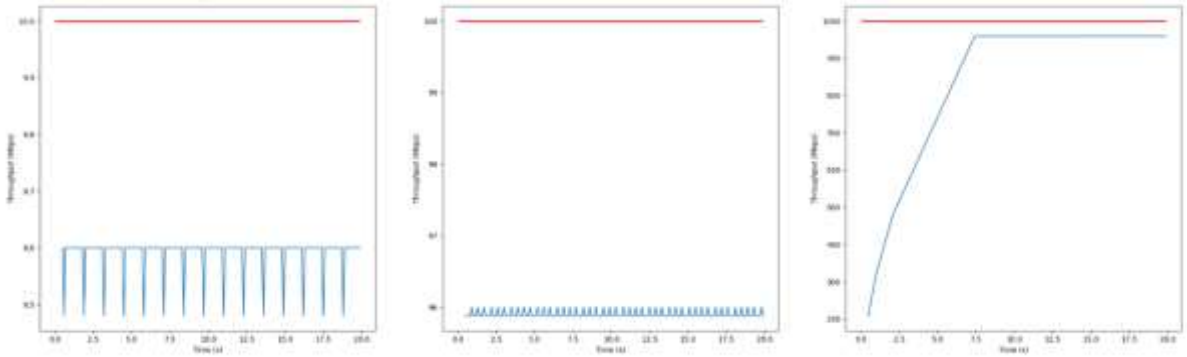
În graficele de mai sus se poate observa cum algoritmul manageriază cele 3 conexiuni pe care le primește router-ul.

La început se poate observa faptul că, la server, se conectează doar primul client (linia albastră). Acesta primește throughput maxim până la timpul de 5 secunde. Atunci, când clientul numărul 2 se conectează (linia portocalie), datorită corectitudinii, scade throughput-ul clientului 1 pentru a permite celor două conexiuni să funcționeze în mod optim. Acest comportament se poate observa și atunci când al treilea client se conectează (linia verde).



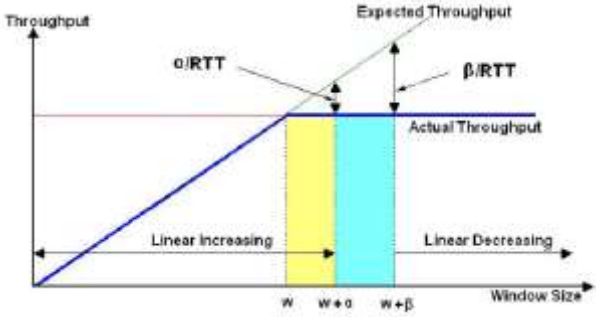
În acest grafic se prezinta cum se ocupa coada router-ului la diferite momente. Se poate observa, că până în secunda 5, când este doar conexiunea 1, coada este goală. Atunci când apar și cele două noi conexiuni, coada se populează.

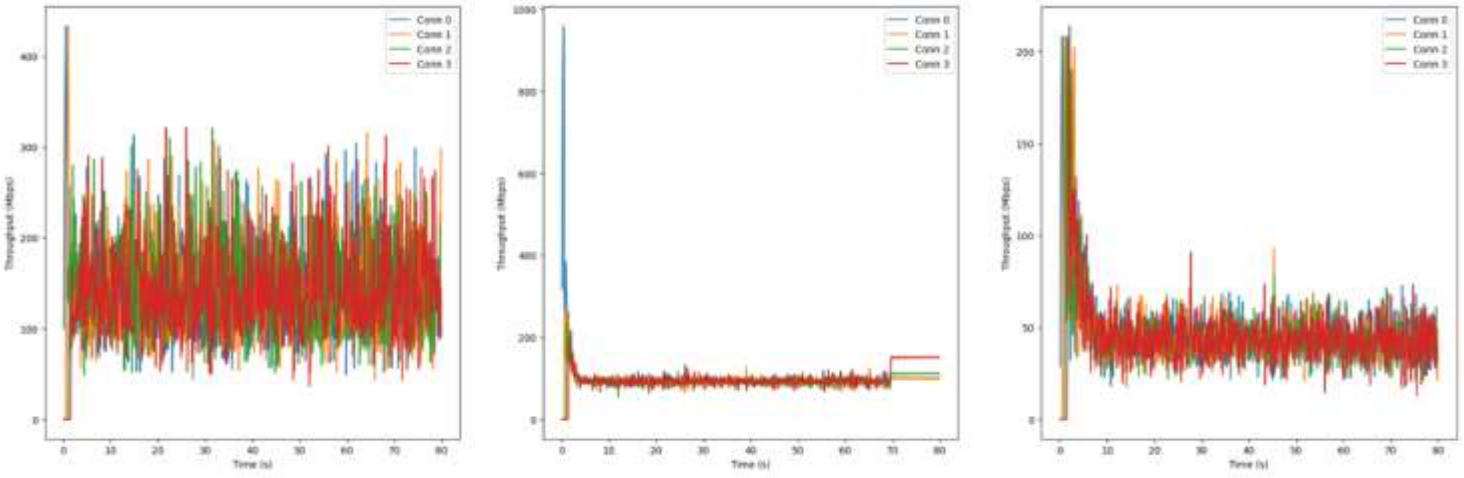
Spike-urile reprezintă primirea multor pachete de la cele 3 cele conexiuni. Limita setata asupra RTT-ului, duce la limitarea cozii sub 800 de pachete în coadă.



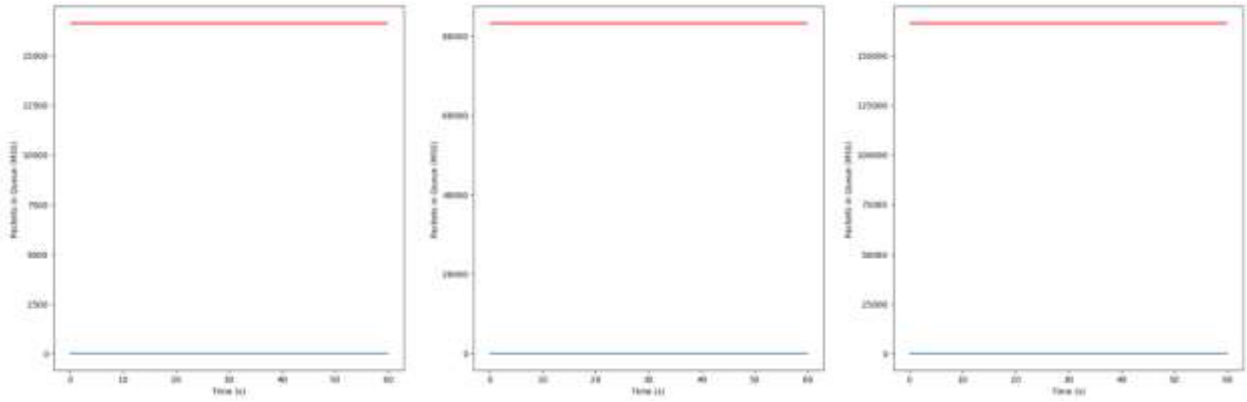
În aceste grafice, se observă throughput-ul algoritmului TCP Vegas având lățimea de bandă diferită. În primul grafic, lățimea de bandă este de 10 Mbps, iar TCP Vegas oscilează între 9.5 și 9.6. În al doilea grafic, lățimea de bandă este de 100 Mbps și se obține aceiași oscilație, între 95 și 96 Mbps. În al treilea grafic, având o lățime de bandă de 1000 Mbps se observă cel mai bine creșterea liniară până la 960 Mbps.

Pentru a demonstra creșterea corectă a throughput-ului, am preluat un grafic de TCP Vegas unde se observă aceiași creștere similară cu al treilea grafic.





Cele trei grafice prezintă corectitudinea lățimii de bandă pentru cele 3 conexiuni. Inițial, fiind singură, prima conexiune preia toată lățimea de bandă disponibilă. Ulterior, când apare o nouă conexiune, se diminuează lățimea de bandă pentru conexiunea 1. Același lucru se întâmplă pentru orice nouă conexiune, celelalte își scad lățimea de bandă. Se observă pe fiecare lățime de bandă furnizată, cum algoritmul TCP Vegas o împarte eficient cele patru conexiuni (reprezentate de cele 3 linii de grafic) fiind aproape complet suprapuse după stabilirea tuturor conexiunilor.



Aceste grafice prezintă latența per pachet introdusă de algoritmul de congestie implementat. Astfel, se observă că algoritmul TCP Vegas are latență minimă, datorită conceptului pe care acesta îl are la bază.

BIBLIOGRAFIE:

1. https://en.wikipedia.org/wiki/TCP_Vegas
2. https://www.youtube.com/watch?v=_X45oCyY69Y&t=692s
3. https://www.cs.princeton.edu/techreports/2000/616.pdf
4. https://www.geeksforgeeks.org/basic-concept-of-tcp-vegas/