SOARE BOGDAN 441A

TCP CONTROLUL CONGESTIEI

Prezenatare Generală

Au fost nişte discuţii serioase despre prăbuşirea Internetului datorită supraîncărcării reţelei sau congestiei. Până în prezent, internetul a supravieţuit, dar au existat o serie de incidente de-a lungul anilor în care problem grave au dezactivat mari părţi ale reţelei. Unele dintre aceste incidente au fost o rezultatul algoritmilor folosiţi sau nu în Transmission Control Protocol (TCP). Altele sunt un rezultat ale problemelor în domenii cum ar fi securitatea, sau poate mai exact lipsa acesteia.

Popularitatea internetului a sporit nevoia de lăţime de bandă în toate nivelurile reţelei. Utilizatorii casnici au nevoie de lăţime de bandă mai mare decât 64 Kb/s ce oferă de obicei furnizorul de telefonie. Video, muzică, jocuri, partajarea de fişiere şi navigarea pe Web necesită lăţime de bandă mai multă, pentru a evita "World Wide Wait", aşa cum aceasta a ajuns să fie cunoscut de către cei cu conexiuni mai lente şi adesea puternic congestionate.

Uimitor TCP s-a schimbat foarte puţin de la designul iniţial la începutul anilor 1980. Câteva mici ajustări au fost adăugate, dar pentru cea mai mare parte, protocolul a rezistat testului timpului. Cu toate acestea, există încă o serie de probleme de performanţă privind Internetul şi software-ul de TCP.

Teoretic, congestia poate fi controlată pe baza unui principiu împrumutat din fizică: legea conservării pachetelor. Ideea de bază este de a nu injecta un nou pachet în reţea până când un pachet mai vechi nu o părăseşte (de exemplu este furnizat receptorului). TCP încearcă să atingă acest scop prin manipularea dinamică a dimensiunii ferestrei.

Primul pas în controlul congestiei este detecţia ei. Mai demult, detecţia congestiei era dificilă. O depăşire de timp datorată pierderii unui pachet putea fi cauzată fie de zgomotul de pe linia de transmisie, fie de eliminarea pachetului de către un ruter congestionat. Diferenţierea celor două cazuri era dificilă.

Proiectarea TCP a fost puternic influenţată de ceea ce a ajuns să fie cunoscut sub numele de argumentul end-to-end. Componenta cheie a argumentului end-to-end pentru scopurile noastre este metoda sa de manipulare congestionării şi a suprasarcinii reţelei.

Premisa argumentului și fundamental pentru design-ul TCP este că stațiile de capăt sunt responsabile pentru controlul ratei de flux de date. În acest model, nu există mecanisme explicite de semnalizare din rețea care spun stațiilor finale cât de repede să transmită, când să transmită, când să accelereze sau când să încetinească.

Algoritmi TCP de control al congestiei

Cei patru algoritmi: Start lent, Evitarea congestiei, Retransmisie Rapidă și Recuperare Rapidă sunt descriși mai jos.

1. Start Lent

O cerință pentru implementarea software TCP este un mecanism utilizat de expeditor pentru a controla rata de transmisie, altfel cunoscut sub numele de control al fluxului de expeditor. Acest lucru este realizat prin rata de returnare a confirmărilor de la receptor. În alte cuvinte, rata de confirmărilor returnate de către destinatar determină rata la care expeditorul poate transmite date.

Atunci când o conexiune TCP începe pentru prima oară, algoritmul de Start Lent initializeaza o fereastră de congestive la un segment, care este dimensiunea maximă a segmentului inițializat de către receptor în timpul fazei de stabilirea conexiunii. Atunci când sunt recunoașteri returnate de către destinatar, fereastra de congestie crește cu un segment pentru fiecare confirmare returnată. Astfel, expeditorul poate transmite minimul dintre fereastra de congestie și fereastra receptorului, care este pur și simplu numită fereastra de transmisie.

Start Lent nu este de fapt foarte lent atunci când rețeaua nu este în congestie și timpul de răspuns al rețelei este bun. De exemplu, prima transmisiune cu succes și recunoaștere a unui segment TCP crește fereastra la două segmente. După transmiterea cu succes a acestor două segmente și confirmări complete, fereastra a crescut la patru segmente. Apoi opt segmente, șaisprezece segmente și așa mai departe.

La un moment dat fereastra de congestie poate deveni prea mare pentru rețea sau condițiile rețelei se pot schimba astfel încât pachetele pot fi pierdute. Pachetele pierdere vor declanșa un timeout la expeditor. Atunci când se întâmplă acest lucru, expeditorul merge în evitarea congestionării.

2. Evitarea Congestieii

În timpul fazei inițiale de transfer de date a unei conexiuni TCP, algoritmul Start Lent este utilizat. Cu toate acestea, poate exista un punct în timpul algoritmului Start Lent când rețeaua este forțată să renunțe la unul sau mai multe pachete din cauza supraîncărcării sau congestie. În cazul în care se întâmplă acest lucru, Evitarea congestiei este utilizată pentru a încetini rata de transmisie. Cu toate acestea, Start Lent este utilizat în conjuncție cu Evitarea Congestiei ca mijloc de a obține pornirea transferului de date din nou astfel încât să nu încetinească și să rămână lent.

În Evitarea Congestiei Algoritmul are un timer de retransmisie de expirare sau recepția ACK-uri duplicate poate semnala implicit expeditorul că o situație de congestionareaa rețelei se va produce. Expeditorul stabilește imediat fereastra de transmisie la o jumătate din dimensiunea ferestrei curente, dar la cel puțin două segmente. În cazul în care congestia a fost indicată de către un timeout, fereastra de congestie este resetată la un singur segment, care pune în mod automat expeditorul în modul Start Lent. În cazul în care congestia era indicată de către ACK-uri duplicate, algoritmii Retransmisie Rapidă și Recuperare Rapidă sunt invocați.

Pe măsură ce datele sunt primite în timpul Evitării Congestiei, fereastra de congestie este mărită. Cu toate acestea, Start Lent este folosit numai până la punctul de la jumătatea distanței în cazul în care congestia a avut loc inițial. Acest punct de jumătatea distanței a fost înregistrat mai devreme ca și fereastra noua de transmisie. După acest punct la jumătate a distanței, fereastra de congestie este majorată cu un segment pentru toate segmentele în fereastra de transport, care sunt recunoscute. Acest mecanism va forța expeditorul să crească mai încet rata de transmisie, așa se va apropia de punctul în care congestia a fost anterior detectată.

3. **Retransmisie Rapidă**

Atunci când un ACK duplicat este primit, expeditorul nu știe dacă este pentru că o conexiune TCP segment a fost pierdută sau pur si simplu că un segment a fost întârziat și a primit din comanda de la receptor. Dacă receptorul poate re-comanda segmente, nu ar trebui să fie mult până receptorul trimite ultima confirmare așteptată. De obicei nu mai mult de unul sau două ACK-uri duplicat ar trebui să fie primite atunci când o simplă condiție de defect există. Dacă totuși mai mult mult de două ACK-uri duplicat sunt primite de către expeditor, acesta este un indiciu puternic că cel puțin un segment a fost pierdut. Expeditorul TCP va asuma că a trecut destul timp pentru toate segmentele pentru a fi corect re-comandate de faptul că receptorul a avut timp suficient pentru a trimite trei ACK-uri duplicat.

Atunci când trei sau mai multe ACK-uri duplicat sunt primite, expeditorul nici nu mai așteaptă pentru ca un timer de retransmisie să expire înainte de a retransmite segmentul.

4. **Recuperare Rapidă**

Deoarece algoritmul Retransmisie Rapidă este utilizat atunci când ACK-uri duplicat sunt primate, expeditorul TCP are cunoștință implicită că nu există date care curg în continuare la receptor. Motivul este că ACK-uri duplicat pot fi generate doar atunci când un segment este primit. Acesta este un indiciu puternic că congestionarea gravă a rețelei poate să nu existe și că pierderea unui segment a fost un eveniment rar. Deci, în loc de a reduce fluxul de date brusc intrând în modul Start Lent, expeditorul intra numai în modul Evitarea congestiei.

Decât să înceapă de la o fereastră cu un segment în modul Start Lent, expeditorul își reia transmisia, cu o fereastră mai mare, incrementând ca și cum ar fi în modul Evitarea Congestiei.

Tehnici TCP

1.TCP Tahoe

Tahoe se referă la un algoritm de control al congestiei TCP care a fost sugerat de Van Jacobson. TCP este bazat pe un principiu de "conservare a pachetelor ", adică în cazul în care conexiunea se execută la capacitatea de lățime de bandă disponibilă, apoi un pachet nu este injectat în rețea excepția cazului în care un pachet este scos de asemenea. TCP pune în aplicare acest principiu, prin utilizarea confirmărilor pentru a urmări pachete care ies, deoarece o confirmare înseamnă că un pachet a fost scos de pe fir de către de receptor. Cu toate acestea, există anumite aspecte, care trebuiesc să fie rezolvate pentru a asigura acest echilibru.

1) Determinarea lățimii de bandă disponibile.
2) Asigurarea că echilibrul este respectat.
3) Cum să reacționeze la congestie.

Sunt folosiți algoritmii Start Lent și Evitarea Congestiei.

Problema cu Tahoe este că durează un interval timeout complet pentru a detecta pierderea de pachete,și de fapt, în cele mai multe implementari pe care le ia durează chiar mai mult.

2. TCP Reno

Reno păstrează principiul de bază de la Tahoe, cum ar fi start lent și cronometru re-transmisie. Totuși se adaugă ceva inteligență peste el, astfel încât pachetele pierdute sunt detectate mai devreme și pipeline-ul nu este golit de fiecare dată când un pachet este pierdut.

Reno impune ca să primim confirmare imediată ori de câte ori un segment este primit. Logica din spatele acestui lucru este faptul că ori de câte ori primim un ACK duplicat, atunci acest ACK ar fi putut fi primit dacă segmentul următor din secvența așteptată ar fi fost întârziată în rețea și pachetul pierdut.

Dacă primim o serie de duplicat recunoașteri atunci înseamnă că a trecut suficient timp și chiar dacă ar fi avut-o segmentul de mai mult cale, ar fi trebuit să ajuns la receptor de acum. Există o cerere foarte mare probabilitatea ca acesta a fost pierdut.

Dacă primim o serie de ACK-uri duplicate atunci înseamnă că a trecut suficient timp și chiar dacă segmental a luat cale mai lungă, ar fi trebuit să fi ajuns la receptor până acum. Există de asemnea o probabilitatea foarte mare ca acesta a fost pierdut.

Reno sugerează folosirea algoritmului **Retransmisie Rapidă.**

Reno funcționează foarte bine peste TCP în cazul în care pierderile de pachete sunt mici. Dar atunci când avem mai multe pierderi de pachete într-o fereastră, RENO nu mai funcționeaza la fel de bine si performanță sa este aproape la fel ca în Tahoe în condiții de pierdere unui număr mare de pachete. Motivul este faptul că acesta poate detecta doar un singur pachet pierdut. Dacă există mai multe pachete pierdute atunci primele informații despre pachetul pierdut apar când primir ACK-ul duplicat.

3. New-Reno

Noul Reno este o ușoară modificare a lui TCP-Reno. Acesta este capabil să detecteze pierderi de pachete multiple și, prin urmare, este mult mai mai eficient ca Reno, în caz de pierderi de pachete multiple.

Ca și Reno, New-Reno, de asemenea intră în retransmisie rapidă atunci când primește mai multe pachete duplicate, cu toate acestea, diferă de la Reno, în sensul că nu iese din retransmisia rapidă până când toate datele care au intrat sunt confirmate. Astfel, depășește problema cu care se confruntă Reno.

Faza retransmisie rapidă este aceeași ca și la Reno. Diferența este faza care permite mai multe re-transmisii la New-Reno.

4. Vegas

Vegas este o implementare TCP care este o modificare a lui Reno. Acesta se bazează pe faptul că măsura proactivă de a întâlni congestii este mult mai eficientă decât cea reactivă. De asemenea, ea rezolvă problema de a cere sufficiente ACK-uri pentru a detecta un pachet pierdut, și sugerează de asemenea un algorim Start Lent modificat care împiedică congestia rețelei. Acesta nu depinde numai de pierderile de pachete ca un semn de congestie. Detectează congestie înainte ca pierderile de pachete să apară. Cu toate acestea, încă mai păstrează celelalte mecanisme Reno și Tahoe.

Concluzia

În ultimul deceniu s-a investit multă cercetare și experimentare în performanța TCP și în controlul congestiei. O mare parte din muncă a fost un success datorită faptului că internetul continuă să funcționeze foarte bine, chiar și odată cu creșterea traficului.

Rămâne de văzut cât de departe tehnicile curente de control a congestiei pot susține Internetul cum creșterea acestuia continuă.

Bibliografie

[1] - http://www.ietf.org/rfc/rfc2581.txt
[2] - http://condor.depaul.edu/jkristof/technotes/congestion.pdf
[3] - http://en.wikipedia.org/wiki/TCP_Vegas
[4] - http://inst.eecs.berkeley.edu/~ee122/fa05/projects/Project2/SACKRENEVEGAS.pdf
[5]http://tools.ietf.org/html/rfc3782, http://web.cs.wpi.edu/~rek/Grad_Nets/Spring2013/TCP_Congestion_Control_S12.pdf
[6] - http://www.comp.nus.edu.sg/~ooiwt/cs5229/archives/0708s1/slides/04-tcp.pdf