Gruppo di Reti e Telematica Applicata

DIMES
UNIVERSITÀ DELLA CALABRIA

Corso di
Fondamenti di Reti di Telecomunicazioni
LT - ELE / LM-TLC

Lo strato di Trasporto

Lezione - 03

Il dimensionamento del timeout (RTO) è un aspetto critico per le prestazioni del TCP.

- minore è il timeout, in meno tempo si interrompe la trasmissione. Però se il suo valore è troppo piccolo, i segmenti in ritardo, a causa di congestione, potrebbero essere considerati persi e quindi ri-trasmessi, ciò aumenterebbe la congestione e di conseguenza le ri-trasmissioni finché la portata tende a zero
- se il suo valore è troppo grande, la risposta ad un evento di perdita sarebbe troppo lenta con conseguente perdita di efficienza e minore velocità di trasferimento

Il timeout è fissato con uno schema "adattativo" in base al valore stimato del round-trip delay (RTT)

TCP misura il RTT, cioè il tempo di andata e ritorno tra sorgente e destinazione impiegato da un'unità dati, ovvero il tempo che passa tra l'invio di un segmento e la ricezione del relativo ACK

- RTT è somma di 3 contributi: ritardo di trasferimento in un verso della comunicazione; ritardo di trasferimento nell'altro verso; tempo necessario al destinatario per rispondere (tempo di risposta)

In generale, la determinazione dei tempi di trasferimento nel RTT è complessa in Internet:

- una connessione può attraversare una LAN ad alta velocità o più sottoreti lente in diversi continenti; perciò il ritardo di trasferimento è molto variabile ed è influenzato dallo stato di congestione momentanea delle sottoreti attraversate

Se si potesse effettuare il recupero di errore a strati inferiori al TCP, sottorete per sottorete, si avrebbero stime precise del ritardo di trasferimento, ma in Internet ciò non è possibile, visto che richiederebbe

- la cooperazione tra sottoreti diverse (impossibile)
- il controllo di errore effettuato da ogni sistema di interconnessione intermedio (da evitare)

In base al valore di RTT misurato in maniera regolare, il TCP aggiorna "dinamicamente" il valore di RTO; TCP potrebbe scegliere un valore di RTO maggiore o uguale al "valore medio" osservato di RTT

Tuttavia, la misura del RTT basata sulla "media" delle osservazioni può essere affetta da errori:

- l'emissione degli ACK da parte del TCP ricevente può essere non immediata (riscontri cumulativi non riferiti a uno specifico datagramma)
- se è stata effettuata una ritrasmissione è impossibile distinguere se l'ACK si riferisce alla trasmissione iniziale o alla ritrasmissione (l'RTT si calcola dal datagramma originale o da quello ritrasmesso?)
- lo stato di congestione della rete può cambiare molto rapidamente

Soluzione: si usa una stima del RTT che fornisce una media pesata del RTT (media esponenziale), denominata Smoothed Round Trip Estimate (SRTT) (RFC793)

L'entità TCP mittente inizializza un temporizzatore (timer) nell'istante in cui inizia la trasmissione di un segmento e ne memorizza l'istante di partenza in una memoria destinata a contenere le informazioni di gestione della connessione

Alla ricezione di un riscontro valido (che include il numero di sequenza del segmento trasmesso), il mittente calcola il tempo trascorso, cioè l'RTT corrente, e aggiorna il valore stimato di SRTT

L'SRTT privilegia i valori dei campioni di RTT più recenti:

$$SRTT(k+1) = a\,SRTT(k) + (1-a)\,RTT(k+1)$$

$0 \leq a \leq 1$; tanto più il valore del peso (smoothing factor) a è vicino a 0, tanto maggiore sarà il peso dato all'ultima osservazione di RTT (normalmente $0.8 \leq a \leq 0.9$)

Valori minori del peso corrispondono ad un aggiornamento veloce del SRTT; valori maggiori rendono il SRTT insensibile a brevi variazioni del ritardo di trasferimento

Usando un valore costante di a (0<a<1), indipendentemente dal numero di osservazioni passate, si considerano comunque tutte le osservazioni ma si dà minor peso a quelle più distanti

Infatti:

$$SRTT(k+1) = (1-a)\,RTT(k+1) + a\,(1-a)\,RTT(k) + a^2(1-a)\,RTT(k-1) + \ldots + a^k(1-a)\,RTT(1) + a^{k+1}\,RTT(0)$$

dato che a e (a –1) sono ≤ 1 ogni termine successivo è più piccolo; per es. per a =0.8 (=7=/8), si ha:

$$SRTT(k+1)=0.2\ RTT(k+1)+0.16\ RTT(k)+0.128\ RTT(k-1)+ \ldots$$

La specifica TCP (RFC 793) prevede che il valore di RTO sia proporzionale a SRTT secondo la seguente espressione:

RTO(k+1) = min[UBOUND, max [LBOUND, b SRTT(k+1)]]

- LBOUND (valore tipico 1 s) e UBOUND (valore tipico 1 m) sono i prefissati limiti inferiore e superiore di RTO
- b è una costante il cui valore deve essere scelto 1.3<b<2.0 (delay variance factor)
- Il lower bound DOVREBBE essere misurato in frazioni di secondo (per accomodare LAN ad alta velocità) e l'upper bound dovrebbe essere 2*MSL, cioè 240 secondi

Algoritmi di Karn e Jacobson

Ci sono due problemi legati al calcolo del RTO specificato nel RFC-793:

- la misura accurata del RTT è difficile quando ci sono ritrasmissioni
- l'algoritmo per calcolare SRTT è inadeguato, perché assume erroneamente che la varianza nei valori di RTT è piccola e costante

Questi problemi sono rispettivamente risolti dagli algoritmi di Karn e Jacobson

- Nel calcolo di SRTT, l'algoritmo di Karn ignora i riscontri di segmenti ritrasmessi
- L'algoritmo di Jacobson incorpora la misura della varianza del RTT ed è importante soprattutto su link a bassa velocità, dove la variazione delle dimensioni dei pacchetti causa un'ampia variazione di RTT. L'algoritmo migliora l'utilizzazione di un link da 9.6 kbps dal 10% al 90%

Il metodo di misura standard di SRTT non è adatto a situazioni in cui la varianza del ritardo di rete è elevata

- se il bit rate della connessione TCP basso, allora il ritardo di trasmissione può essere elevato rispetto al ritardo di propagazione e perciò la varianza del ritardo dipende dalla dimensione dei datagrammi IP (dai dati e non dalla rete)
- variazioni repentine di traffico e di carico in rete possono provocare brusche variazioni di RTT
- l'entità TCP ricevente può inviare i riscontri in maniera cumulativa o comunque dopo un certo tempo di processamento

L'RFC 793 specifica il timeout come il doppio del valore di RTT stimato:

$$SRTT[k+1] = (1-a)\,SRTT[k] + a\,RTT[k+1]$$

$$0<a<1;\ \text{tipicamente } a=1/8=0.125$$

$$RTO = b\,SRTT = 2\,SRTT$$

In realtà, SRTT oscilla in maniera random attorno al valor medio con una deviazione standard SDEV(RTT)

Primo problema (sovrastima di RTO!):
supponiamo una rete non-congestionata tale che l'RTT resti quasi costante per un lungo intervallo. Improvvisamente un pacchetto si perde. Dopo RTT secondi, l'ACK non è ancora stato ricevuto. E' quasi certo che il pacchetto è stato scartato, tuttavia bisogna aspettare RTT secondi prima di ritrasmetterlo!

Secondo problema (sottostima di RTO):
dalla teoria delle code, se il carico di rete aumenta, il valore medio di RTT aumenta e (cosa peggiore) la sua varianza cresce anche di più (inversamente proporzionali a (1-r)). In queste circostanze, fissare RTO pari solo al doppio del valore misurato di RTT potrebbe essere troppo piccolo (anche minore del RTT reale). Cioè, i pacchetti che impiegano tanto tempo ad arrivare per via della congestione di rete (ma che arriveranno) sono ritrasmessi aumentando ancora la congestione

Jacobson afferma che questo succede con carichi di rete superiori al 30%

J. propose di usare un valore di b grossolanamente proporzionale alla deviazione standard della pdf del tempo di arrivo degli ack

Per misurare la variazione di RTT ci sono varie alternative: la scelta convenzionale è la varianza e quindi la deviazione standard:

$$s^2 = 1/n \; S \; |RTT\text{-}SRTT|^2$$

La stima della deviazione standard di RTT è troppo complessa (calcolo di quadrati e radici quadrate)

J. suggerì di usare la DEVIAZIONE MEDIA MDEV(RTT) dei campioni del RTT (o errore di predizione medio) come stimatore della deviazione standard di RTT:

$$MDEV(x) = E\,[\,|X - E[X]|\,]$$

$$MDEV^2(RTT) = 1/n\,(S\,|RTT - SRTT|)^2$$

Se gli errori di predizione sono distribuiti normalmente, $MDEV = (p/2)^{1/2} SDEV \approx 1.25$, cioè MDEV è una buona approssimazione di SDEV ed è molto più facile da calcolare

J. propose di calcolare RTO come:

$$RTO(k+1) = SRTT(k+1) + u * MDEV(RTT)$$

dove u è di solito fissato a u=4

Giustificazione

- Se c'è poca differenza tra i valori campionati di RTT e la stima del SRTT per lungo tempo (MDEV(RTT)->0), si può pensare che SRTT è una buona stima di RTT. Quindi, se un riscontro impiega più di SRTT sec ad arrivare, si può pensare, a ragione, di ritrasmettere il segmento
- D'altra parte, se RTT ha un'alta varianza (grande MDEV(RTT)), allora la stima non è molto affidabile ed è opportuno avere un margine di sicurezza proporzionale all'incertezza

- calcolo della stima di RTT, SRTT(K+1):

$$SRTT(k+1) = (1-a)\ SRTT(k) + a\ RTT(k+1) =$$

$$SRTT(k) + a\ (RTT(k+1) - SRTT(k))$$

  - valori tipici di a sono 0.1-0.2
  - calcolo dell'errore nella predizione di RTT, SERR(K+1):

$$SERR(K+1) = RTT(K+1) - SRTT(K)$$

  - quindi:

$$SRTT(k+1) = SRTT(k) + a\ SERR(k+1)$$

  - la nuova predizione è basata sulla vecchia più una frazione dell'errore nella predizione

Calcolo della deviazione media MDEV(K+1)

J. propose di usare la stessa tecnica di media esponenziale (exponential smoothing) usata per la stima di RTT anche per la stima di MDEV, indicata con SDEV, quindi:

$$SDEV(k+1) = (1-h)\, SDEV(k) + h\, |SERR(k+1)|$$

calcolo del valore del timeout RTO(K+1)

$$RTO(k+1) = SRTT(k+1) + u\, SDEV\,(k+1)$$

a=0.125 (1/8), h=0.25 (1/4), u=4 (all'inizio J. disse u=2, poi si accorse che u=4 aveva maggiori vantaggi: (1) la moltiplicazione per 4 può essere fatta con un solo shift; (2) minimizza timeout e ritrasmissioni non necessarie perché meno dell'1% di tutti i pacchetti arrivano con più di 4 deviazioni standard di ritardo)

Con l'algoritmo di J. i valori di RTO calcolati sono piuttosto conservativi rispetto ai valori misurati di RTT finché i campioni di RTT variano, poi cominciano a convergere verso RTT quando i valori dei campioni si stabilizzano, cioè la stima della variazione SDEV si riduce (vedi fig. libro Stallings pg. 260)

Jacobson's Algorithm

L'algoritmo di base

Per calcolare il valore corrente di RTO, un sender TCP mantiene variabili di stato, SRTT e RTTVAR (round-trip time variation Inoltre, assumiamo una granularità del clock di G secondi

Le regole per il calcolo di SRTT, RTTVAR, e RTO sono:

(1) Finché non viene misurato un RTT per un segmento, il sender DOVREBBE porre RTT=0, RTO=3 sec (RFC 1122), anche se si applica il "back off" sulle ritrasmissioni ripetute

(2) Quando si effettua la prima misura di RTT, R, l'host DEVE porre

SRTT <- R
RTTVAR <- R/2
RTO <- SRTT + max (G, K*RTTVAR)

dove K = 4

(3) Quando si effettua una misura successiva R' di RTT, l'host DEVE porre

RTTVAR <- (1-beta) * RTTVAR + beta * |SRTT-R'|
SRTT <- (1-alpha) * SRTT + alpha * R'

Il valore di SRTT usato nell'aggiornamento di RTTVAR è il suo valore prima dell'aggiornamento di SRTT stesso usando il secondo assegnamento. Il calcolo DOVREBBE essere eseguito usando alpha=1/8 e beta=1/4

Dopo il calcolo, un host DEVE aggiornare

RTO <- SRTT + max (G, K*RTTVAR)

(4) Una volta che RTO è calcolato, se è minore di 1 sec allora RTO DOVREBBE essere arrotondato a 1 sec. Un valore minimo di RTO più grande è necessario per mantenere il TCP conservativo e per evitare ritrasmissioni spurie

(5) Un valore massimo PUO' essere fissato per RTO purché sia almeno 60 sec

Altri due fattori devono essere considerati per migliorare le prestazioni del TCP:

1. Quale valore di RTO usare su un segmento ritrasmesso? Si usa l'algoritmo di backoff esponenziale
2. Quali campioni usare in input all'algoritmo di Jacobson. L'algoritmo di Karn determina quali campioni di RTT usare per non inficiare la stima

Backoff esponenziale di RTO

Quando scade il timeout relativo a un segmento, il TCP sender ritrasmette il segmento stesso; secondo la specifica originale RFC793 il sender usa sempre lo stesso valore di RTO per tutte le successive ritrasmissioni del pacchetto

Questa tecnica non è consigliabile se la scadenza del timeout è legata a uno stato di congestione di rete perché lo aggraverebbe

- è consigliabile variare il valore di RTO delle sorgenti che sono coinvolte nella congestione per evitare ritrasmissioni contemporanee
- è consigliabile aumentare RTO ogni volta che il TCP sender ritrasmette lo stesso segmento (backoff)

Una tecnica semplice per implementare il backoff è il backoff esponenziale

Ogni volta che deve ritrasmettere un segmento, TCP moltiplica il valore di RTO precedentemente calcolato per un opportuno valore q, finchè il segmento non vada a buon fine

La sorgente TCP aumenta il valore di RTO per ogni ritrasmissione (backoff process)

$$RTO_{i+1} = q\ RTO_i$$

normalmente q=2 (binary exponential backoff)

Normalmente, il valore di RTO viene raddoppiato a ogni ritrasmissione fino al raggiungimento di un fattore moltiplicativo pari a 64, ottenuto alla settima trasmissione

In base a quest'equazione, RTO cresce esponenzialmente ad ogni ritrasmissione

Oltre questo valore, la connessione viene reinizializzata (con una procedura di reset): il valore di RTO torna al valore precedente solo dopo la ricezione di un riscontro relativo ad un segmento che è stato trasmesso una sola volta

TCP usa l'algoritmo di Karn per scegliere i campioni di RTT da usare per la stima di SRTT

In caso di ritrasmisssione TCP non distingue se il riscontro si riferisce

- alla prima trasmissione del segmento
- alla ritrasmissione del segmento

Un errore di attribuzione può causare

- timeout troppo elevato (perdita di efficienza e inutili ritardi)
- timeout troppo breve (ritrasmissioni eccessive nuovi errori di misura)

Nel calcolo di SRTT e SDEV, secondo l'algoritmo di Karn, NON DEVONO essere inseriti i campioni di RTT relativi a segmenti ritrasmessi (per i quali è ambiguo se la reply si riferisce alla prima istanza del pacchetto o all'ultima)

Quindi TCP aggiorna SRTT e SDEV solo con riferimento ai segmenti trasmessi una sola volta

L'unico caso in cui il TCP può prendere campioni RTT dai segmenti ritrasmessi è quando si usa l'opzione "timestamp" che rimuove l'ambiguità

Inoltre l'algoritmo di Karn stabilisce di calcolare il valore di RTO dei segmenti ritrasmessi con la procedura di exponential backoff

Si usa il backoff esponenziale nel calcolo di RTO finché arriva un riscontro relativo a un segmento che non è stato ritrasmesso

A questo punto si riattiva l'algoritmo di Jacobson per il calcolo di RTO