Protocollo TCP - 5

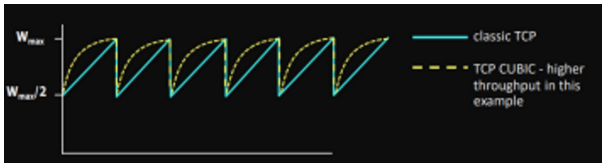
martedì 22 agosto 2023 11:45

TCP CUBIC

TCP CUBIC differisce da TCP Reno solo nella fase di congestion avoidance, in questo modo:

- Sia W_{Max} la dimensione della finestra del controllo di congestione TCP nell'istante in cui viene rilevata l'ultima perdita e sia *K* l'istante nel futuro in cui la dimensione della finestra di TCP CUBIC raggiungerà nuovamente W_{Max}, *assumiamo che non vi sia alcuna perdita*.
- CUBIC aumenta la finestra di congestione in funzione del cubo della distanza tra l'istante corrente *t* e *K*. Pertanto se il tempo *t* corrente e il tempo *K* sono distanti, gli incrementi della dimensione della finestra di congestione sono molto maggiori rispetto a quando l'istante corrente *t* è vicino da *k*.

Il concetto è: CUBIC aumenta rapidamente la velocità di invio di TCP per avvicinarsi alla velocità W_{Max} e solo quando è vicino viene esaminata con cautela la larghezza di banda.
K è un valore fisso nel tempo, *t* varia, **al variare di *t* varia la finestra di congestione**: quando *t* è vicino a *K* gli aumenti di *cwnd* sono piccoli, quando *t* supera *K* la distanza fra i due aumenta sempre più, di conseguenza anche la finestra di congestione aumenta sempre più.
Quest'ultima cosa permette a CUBIC di trovare più rapidamente un nuovo punto di lavoro qualora il livello di congestione del collegamento che ha causato la perdita sia cambiato in modo significativo.



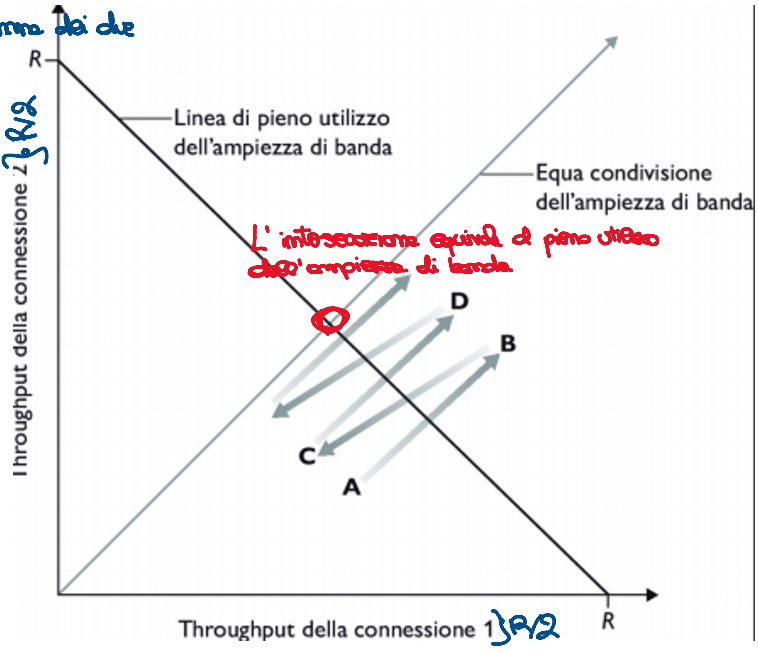
CUBIC si avvicina a W_{Max} più rapidamente godendo così di un throughput complessivo maggiore rispetto a TCP reno.
Se noti graficamente, TCP CUBIC tenta di mantenere per tutto il tempo possibile il flusso appena al di sotto della soglia di congestione.

Controllo della congestione equo: Fairness

Supponiamo di avere *K* connessioni TCP con un diverso percorso end-to-end, ma che passano attraverso uno stesso collegamento con capacità trasmissiva di R bps che costituisce il collo di bottiglia del sistema.
Un meccanismo di controllo della congestione è equo se la velocità di trasmissione media di ciascuna connessione sullo stesso collegamento è approssimativamente *R/K* : ovvero ciascuna connessione ottiene la stessa porzione di banda del collegamento.
L'obiettivo è quindi quello che: se *K* sessioni TCP condividono lo stesso collegamento, collo di bottiglia, di larghezza di banda pari a R, allora ciascuno dovrebbe avere una velocità media di *R/K*.
Prendiamo in considerazione un numero fisso di sessioni TCP nelle fase di congestion avoidance (AIMD) dove le diverse connessioni hanno gli stessi valori di MSS e RTT (->*stessa finestra di congestione e stesso trhoughput*) e devono trasmettere una grande quantità di dati…

Nota: il controllo di congestione di TCP Additive increase, Multiplicative decrease (AIMD) è equo.

In ogni istante la somma dei due throughput non da R



L' obiettivo è quello di tenere il throughput vicino questa intersezione...

Se le connessioni 1 e 2 subiscono una perdita quando raggiungono i thorughput indicati dal punto B, applicando l'algoritmo di congestion avoidance le loro finestre vengono dimezzate. Ora ci ritroviamo quindi nel punto C, a questo punto entrambe le connessioni cominciano ad incrementare di 1 MSS fino ad arrivare al punto D in cui dimezzeranno e così via sempre con questo comportamento.
La banda utilizzata dalle due connessioni può fluttuare lungo la linea di equa condivisione della banda e le connessioni convergeranno a questo comportamento indipendentemente dal punto in cui si trovano inizialmente (ricorda però che abbiamo fatto le ipotesi di RTT e MSS uguale).

Tali meccanismi consentono al protocollo di TCP di regolare il tasso di trasmissione delle applicazioni, questo è il motivo per cui molte applicazioni multimediali non fanno uso di TCP; queste applicazioni pur di non avere *trasmission rate* ridotto usano UDP senza alcun controllo di congestione, implica che il traffico TCP viene **soffocato,** questo perché TCP diminuisce il proprio traffico anche in presenza di chi non lo fa, permettendo quindi ad UDP di "rubare" la maggior parte della larghezza di banda.

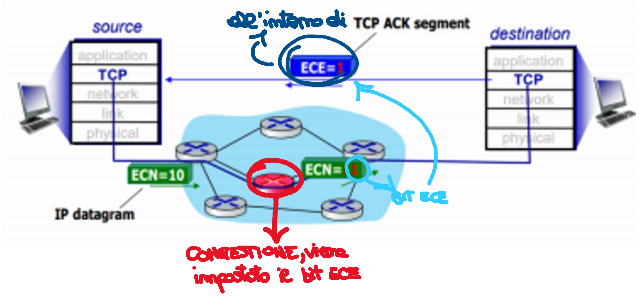
Fin ora abbiamo approfondito il controllo di congestione end-to-end, approfondiamo ora brevemente il controllo della congestione assistito dalla rete…

Politiche di gestione del buffer dei router

Le politiche attive di gestione dei buffer, cercano di desincronizzare le perdite.
L'idea è quella di avere un processo che osserva le dimensioni della coda, quando quest'ultima è piccola non faccio nulla, quando la dimensione è media significa che *potrebbe* andare in **overflow** e dopo, come già sappiamo, abbiamo il "**drop tail**" ovvero tutta la coda aggiuntiva di pacchetti siccome non ci stanno vengono scartati.
Nel caso di coda media avviso il mittente e lo invito ad abbassare la velocità di trasmissione: un pacchetto che avrebbe spazio lo butto ugualmente via, in modo tale da mantenere la coda stabile.
Questa politica si chiama **RED** *(random early detection)*.

Notifica esplicita di congestione assistita dalla rete

L'**explicit Congestion Notification, ECN**, è la forma di controllo di congestione assistita dalla rete effettuata in internet, sono coinvolti sia TCP che IP.
Infatti al livello di rete vengono utilizzati due bit *(per indirizzare 2^2 = 4 possibili valori)* nel campo *"tipo di servizio"* nell'intestazione del datagramma IP. Se un router è congestionato, imposta tali bit ed invia il pacchetto al destinatario.



Tramite ECE il mittente viene avvisato della congestione dal destinatario; quando quest'ultimo riceve per primo l'ECE riduce la sua finestra di congestione, il mittente successivamente reagisce allo stesso modo.

Questo permette ai mittenti di ridurre le velocità d'invio in anticipo, auspicabilmente prima della perdita di pacchetti, evitando così perdite e ritrasmissioni.

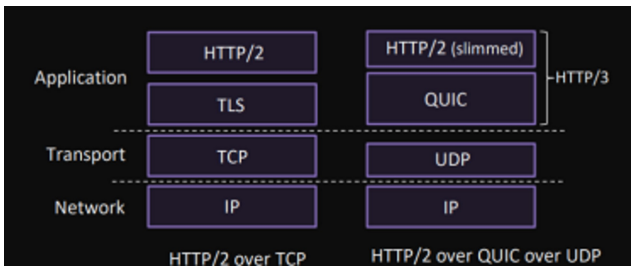
Controllo della congestione basato sul ritardo con TCP Las Vegas

L'obiettivo è quello di rilevare l'insorgenza di una congestione in modo proattivo ovvero prima che si verifichi la perdita di pacchetti: **se un pacchetto è in ritardo viene rilevata la congestione**.
Sia *RTTmin* il valore minimo delle misurazioni del percorso dalla sorgente alla destinazione per tutti i pacchetti riscontrati fino ad allora calcolato lato mittente. Se la dimensione della finestra di congestione di TCP è di *cwnd / RTTmin*, abbiamo due situazioni:

1) Se il throughput effettivo misurato dal mittente è vicino a *cwnd / RTTmin*, allora la velocità d'invio può essere aumentata, fino a che il percorso non diventa congestionato;
2) Se invece la velocità effettiva del mittente è significativamente inferiore al tasso di throughput in assenza di congestione (*RTTmin*), il percorso è congestionato e il mittente TCP Las Vegas diminuisce la velocità di invio.

QUIC

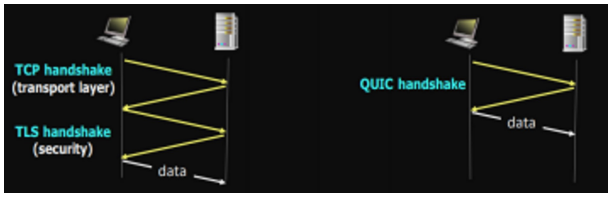
Qualora i modelli di servizio offerti da UDP e TCP non fossero adatti ai servizi del livello di trasporto richiesto da una applicazione, gli sviluppatori dell'applicazione possono usare un proprio protocollo a livello applicativo:
Quic è un nuovo protocollo a livello di applicazione progettato da zero per migliorare le prestazioni del livello di trasporto per HTTP sicuro, *QUIC usa UDP*.
Prossimamente HTTP/3 incorporerà nativamente QUIC, ecco dove quest'ultimo si posiziona:



Le sue caratteristiche:

- **Protocollo orientato alla connessione**: fase di handshaking tra gli end-point per impostare lo stato della connessione, due parti di questo sono ID origine ed ID destinazione; inoltre tutti i pacchetti QUIC sono

crittografati…



- Flussi: QUIC consente il multiplexing di diversi "flussi" a livello applicativo attraverso una singola connessione, una volta stabilita nuovi flussi possono essere aggiunti, come? Ogni connessione ha un ID di connessione e ogni flusso, all'interno di una connessione, a sua volta ha **un ID di connessione + un ID di flusso**: entrambi sono contenuti nell'intestazione del pacchetto QUIC, *insieme ad altre informazioni*. I dati di più flussi possono essere contenuti all'interno di un singolo segmento QUIC che viene trasferito su UDP;
- Trasferimento dati affidabile e controllo della congestione: questo viene fornito ad ogni flusso separatemente all'interno di una stessa connessione, utilizzando meccanismi di acknowledgment simili a quelli di TCP.