Dalle lezioni del prof

MSF - Attivazione della connessione - 3way handshake

In questa immagine possiamo vedere come lo stato del client passa da closed a SYN SENT e poi a ESTAB, quando viene ricevuto un SYN/ACK da parte del server.

Per quanto riguarda il server abbiamo **l'apertura passiva** determinata dalla funzione listen() e quindi passiamo da closed a **LISTEN**; nello stato listen vi è un'evento, ovvero la **ricezione di un SYN** che ci fa passare allo stato **SYN RCVD**; viene inoltre inviato un SYN/ACK, che fa passare il client nello stato ESTAB.

Nello stato SYN RCVD ci aspettiamo di ricevere un ACK per passare allo stato ESTAB anche per il server:

Bonus: è possibile che il server debba chiudere la connessione e fermarsi allo stato SYN RCVD perchè chi ha chiesto la connessione la chiuda subito dopo. E' possibile anche ricevere un SYN invece di un SYN/ACK (nello stato SYN RCVD del server): stiamo dicendo che il client ha spedito un segmento per l'attivazione della connessione, ma invece di ricevere un riscontro per il suo segmento, riceve un altro segmento di attivazione della connessione:

Siamo in una situazione particolare dell' **attivazione simultanea di una connessione**. Nel caso in cui venisse ricevuto solo il SYN (il client spedisce il riscontro) è possibile raggiungere lo stato ESTAB.

Automa completo:

Gestione della connessione: Closing

Cosa accade per la chiusura della connessione? Si impiega un handshake simile che viene chiamato **Four way handshake** perchè lo scambio di segmenti è di 4 segmenti, e non di 3.

Ipotizziamo che il client promuova la connessione (e non il server)

Quando il client chiude la connessione abbiamo un bit flag **FIN posto ad 1**, che ci dice che il client è intenzionato a chiudere la connessione; il numero di sequenza è quello a cui la connessione era arrivata in quel momento.
Viene quindi spedito questo segmento ed il server risponde con un ACK con un numero di riscontro pari ad **x + 1**, quindi anche in questo caso si assume, in assenza di dati, che il segmento abbia dimensione 1 byte (possiamo immaginarlo come il byte che rappresenta il fine stringa).

Dopo il primo ACK con numero di riscontro pari ad x+1, il server invia un nuovo segmento simile a quello che ha inviato il client per chiudere la connessione: questo segmento ha il bit FIN posto ad 1 ed un numero di sequenza pari ad y.
Il client deve quindi rispondere con un ACK pari ad y+1 (come è avvenuto da parte server quando il client ha inviato un segmento FIN=1) e da questo momento in poi la connessione è **definitivamente chiusa**.

La differenza sostanziale con il 3-way handshake è che i due segmenti centrali (riscontro per il primo FIN ed il FIN che va dal server al client) **non sono sovrapposti!** Sono infatti separati. Solo quando il server invoca la funzione close() (ad es in java) sarà inviato il segmento FIN. Se il server non fa questa operazione, il server può continuare ad inviare dati al client.

Nel caso in cui si usa close() la socket viene chiusa. Quindi sebbene lato client sia possibile ricevere dati, la socket non consente di farlo! non è possibile invocare sulla socket la funzione read().

Chiusura parziale della connessione

Nel caso in cui si vuole consentire al client di ricevere dati una volta chiusa la connessione, dobbiamo usare una funzione diversa; non usiamo più la funzione **close()**, ma **shutdown()**, che abbiamo già visto negli esercizi.

Con shutdown il client chiede la chiusura parziale della connessione (invio di FIN e ricezione di ACK) ma al tempo stesso la socket **non viene completamente chiusa**: con **shutdownOutput()**, ad esempio, specifichiamo che vogliamo chiudere la socket di scrittura.

Stato time wait

E' lo stato che caratterizza la socket che ha chiesto la chiusura della connessione (client) quando viene ricevuto il segmento FIN dal server:

Quando viene ricevuto il segmento FIN dal server, il client potrebbe chiudere la connessione e rilasciare le risorsa; il problema è che il server **potrebbe non ricevere correttamente il segmento ACK** di riscontro. Di conseguenza è presente un timeout che ci permette di attendere un tempo abbastanza lungo da poter attendere un'eventuale **rispedizione del FIN da parte del server**, in modo da poter rispondere nuovamente con un ACK.

Quanto aspettiamo? Il client deve attendere un tempo legato (superiore) al **Round Trip Time**, in modo da dare al server il tempo sia di ricevere l'ACK sia di rispedirlo; dobbiamo quindi attendere un eventuale timeout del server.

TCP - Fast retransmit

Una tecnica che consente di migliorare le prestazioni del TCP è la ritrasmissione veloce: evita di dover aspettare che si presenti l'evento di timeout per ritrasmettere un segmento. Il TCP rileva un malfunzionamento all'interno della rete non solo grazie al timout, ma anche grazie ad altri eventi.

In questa immagine notiamo che su 5 segmenti il secondo viene perso. All'Host A viene quindi spedito un numero di riscontro pari a 100 (ovvero l'ultimo segmento atteso ma non ricevuto); il numero di riscontro per i segmenti successivi continuerà ad essere 100, perchè è presente un "buco" dato dalla non ricezione del segmento con numero di seq 100.

Abbiamo quindi 3 riscontri duplicati, e di conseguenza il client rispedisce il segmento che il server non ha ricevuto. Dobbiamo notare che questa rispedizione **anticipa l'evento di timeout del client**.

Di conseguenza la definizione di **fast transmit** corrisponde alla rispedizione di un segmento dopo la ricezione di 3 (in questo esempio) ACK duplicati.

Principi del controllo della congestione

Quando facciamo riferimento alla congestione il problema riguarda **le risorse di rete**. Questo problema, se diffuso in diverse parti della rete, si traduce in un'impossibilità da parte della rete di gestire il traffico (che i dati vengano consegnati ai destinatari).

Scenario 1 - buffers infiniti

Assumiamo che ci sia un unico router da attraversare e che i buffer contenuti all'interno del router siano buffer di dimensione infinita. Immaginiamo che ci siano due macchine A e B che danno vita a due flussi verso altre macchine C e D, e questi dati devono essere inoltrati su un link di capacità R che passa per il router:

λin è il throughput in entrata mentre λout è in uscita. Assumiamo che non ci siano ritrasmissioni (caso ideale).

Cosa succede se se λin (throughput in ingresso) laggiunge R/2 (ovvero la metà della larghezza di banda del link)?

Se il throughput si avvicina ad R/2 (da parte di due macchine quindi R/2 * 2) avremo che il throughput sul link sarà pari ad R. Di conseguenza, aumentare alla sorgente il throughput in uscita a più di R/2 **non ci consente di avere un throughput maggiore di R/2 in ricezione**.

Quando il throughput arriva a R/2 (valore massimo consentito) il **ritardo di propagazione (consegna)** aumenta significativamente.

Siccome stiamo considerando il buffer di dimensione infinita, il ritardo sui pacchetti può raggiungere un valore infinito.

Scenario 2 - buffers limitati

Nella realtà i buffers non sono di dimensione infinita; cosa accade in un router quando la dimensione del buffer è finita ed il throughput in ingresso al router si avvicina al valore R?

Quando abbiamo un buffer limitato, o i pacchetti arrivati precedentemente vengono sovrascritti, o semplicemente **non accettati**. A differenza di prima è possibile che nel router (intermediario tra due nodi che vogliono comunicare) si **verifichino delle perdite**. Questo spiega perchè un canale di comunicazione può perdere pacchetti.

Poiché è possibile che si verifichino delle perdite, il throughput può cambiare a seconda del livello applicativo o di trasporto:

A livello di trasporto λin può crescere rispetto a quello a livello applicativo per via delle **ritrasmissioni** (i dati effettivi vengono inviati maggiormente, ma sono sempre gli stessi). Siccome chi spedisce non sa se il buffer del router è libero o meno, accade ciò:

- Viene spedito un pacchetto
- Il pacchetto arriva al router ma non c'è spazio, il pacchetto viene scartato.
- Il pacchetto viene rispedito
- C'è spazio nel buffer quindi il pacchetto viene consegnato.

🏁 1:57

Osserviamo che all'approssimarsi di λ'in ad R/2 (massima capacità) il throughput non sarà più R/2, ma un valore più basso:

Questo perchè i pacchetti vengono rispediti (visto che una grande quantità di essi viene scartata dal router).

Fine lezione 24