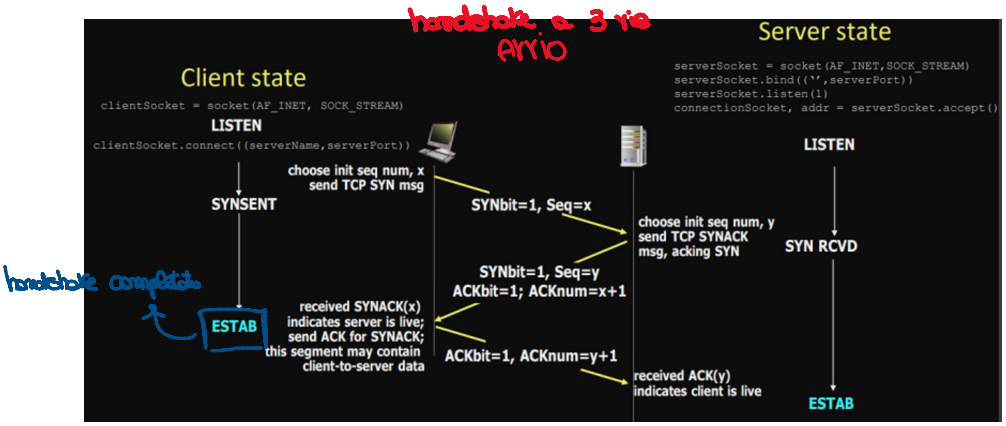
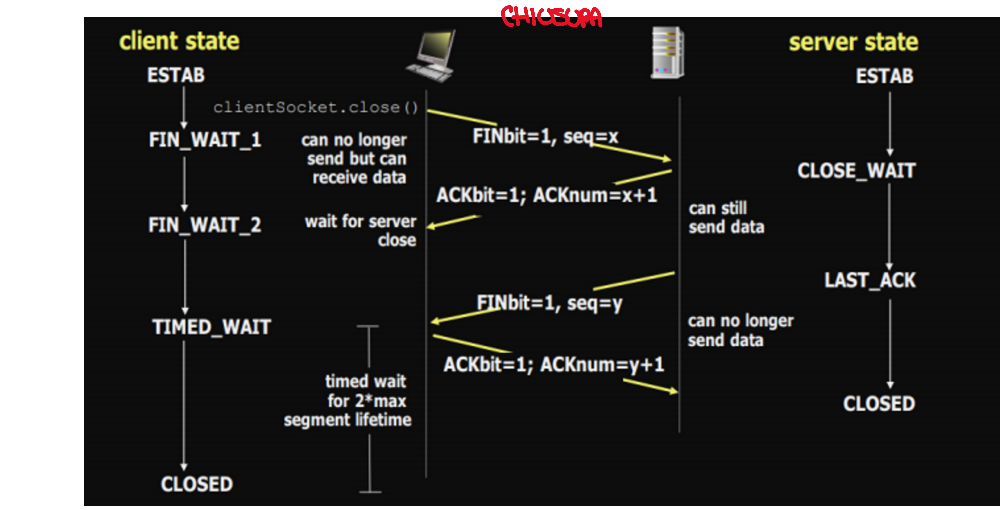
Protocollo TCP - 3

venerdì 18 agosto 2023 15:48

Gestione della connessione TCP

In questa fase di **handshake a tre vie**, vengono scambiate informazioni, da queste **vengono stabilite le variabili di connessione**: come le dimensioni dei buffer, i sequence number iniziali, gli ACK corrispondenti ecc.





Avvio di una connessione

Dopo che il processo applicativo client informa il suo TCP di avviare una connessione, verso un processo server, ecco cosa succede:

- Passo 1: TCP lato client invia uno speciale segmento al TCP server senza alcun contenuto applicativo, con il *bit SYN posto ad 1*, da questo il segmento prende il nome di **segmento SYN.** Il sequence number iniziale (per questo segmento) **client_isn** viene scelto casualmente;
- Passo 2: Quando il datagramma IP (corrispondente al segmento SYN inviato dal client al passo 1, supponendo non sia stato perso) arriva al processo server, quest'ultimo estrae il segmento SYN, alloca il buffer e le variabili TCP per la connessione, infine invia al client un segmento "connessione approvata". Anche questo non contiene dati applicativi e nella sua intestazione il bit SYN è impostato, inoltre:
    - Il campo ACK assume valore client_isn + 1;
    - Il server sceglie il proprio sequence number, server_isn, e lo pone nel campo "numero di sequenza".

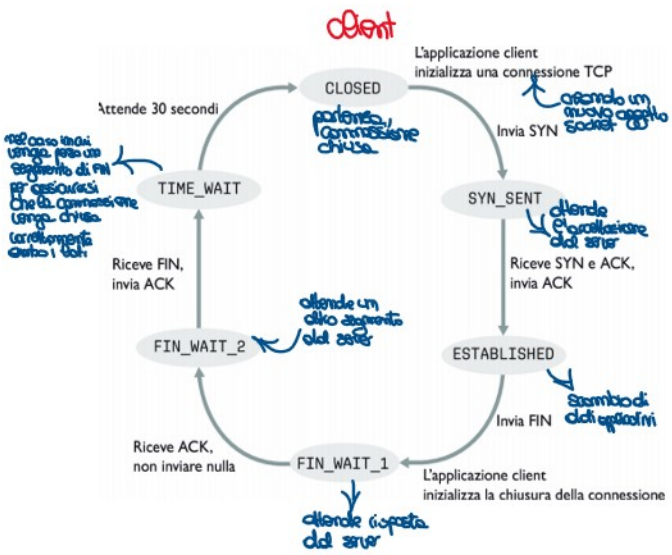
  Il segmento di "connessione approvata" prende il nome di SYNACK;
- Passo 3: Alla ricezione del segmento SYNACK, anche il client alloca il suo buffer e variabili di connessione. A questo punto il client invia al server un altro segmento in risposta al segmento SYNACK (del server) dove il campo ACK avrà valore server_isn + 1. Per quest'ultimo segmento il bit SYN è posto a 0 (connessione stabilita) e il campo dati potrebbe contenere un messaggio.

poi farlo anche il server

Chiusura di una connessione

Il client vuole chiudere la connessione: il processo client invia un comando di chiusura che forza il suo TCP ad inviare un segmento speciale di chiusura al processo server. Nell'intestazione di tale segmento abbiamo il bit FIN che ha valore 1. Quando il server riceve questo segmento risponde inviando il rispettivo acknowledgment al client; il server a sua volta spedisce il proprio segmento di chiusura (shut down) con anch'esso il bit FIN pari a 1. Infine, il client manda a sua volta il rispettivo acknowledgment a quest'ultimo segmento del server.
Ora tutte le risorse sui sistemi periferici degli host risultano deallocate.

Ora vediamo i diversi stati visitati dal client e dal server TCP...



*Casi patologici: l'host (server) riceve un pacchetto TCP SYN con porta di destinazione 80, ma lui non accetta connessioni su quella porta. L'host ricevente quindi invia al mittente un segmento speciale di **reset** con il bit RST impostato a 1; Questo comunicherà alla sorgente "non ho una socket per quel segmento.*

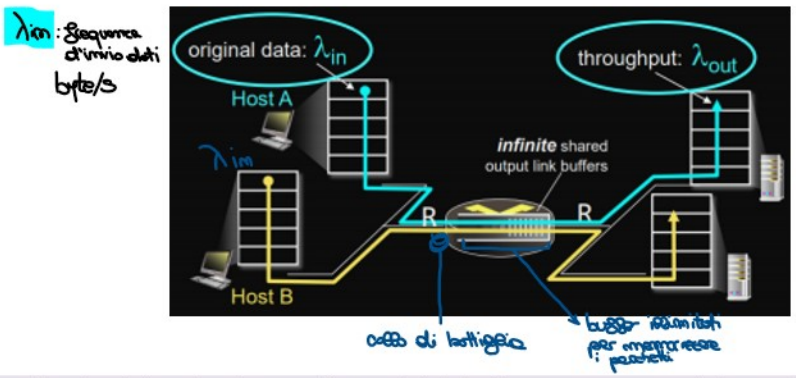
Attacco SYN flood

Se l'handshake a tre vie non viene completato, ovvero non viene inviato l'ACK finale, il server mantiene la connessione "mezza aperta"... In un attacco SYN flood l'aggressore manda un gran numero di segmenti TCP SYN, senza completare il terzo passo dell'handshake *(l'attacco può essere amplificato mandando i segmenti SYN da più sorgenti creando in questo modo un attacco SYN flood di tipo Ddos)*.
Questa inondazione di segmenti TCP SYN, le risorse del server riservate alle connessioni possono esaurirsi velocemente: perchè allocate alle connessioni "mezze aperte" ma mai usate.
Se le risorse sono esaurite, agli utenti legittimi del servizio è negato il servizio.
Quando il server riceve un pacchetto apre metà della connessione e memorizza in una **TCB queue** tutti quelli che hanno fatto i primi due passi di una connessione, inoltre, dato che la TCB queue, è mantenuta dal sistema operativo, oltre che intasare il servizio, manda in crisi l'intero server o il suo sistema operativo.
Difesa: Una difesa efficace agli attacchi SYN flood è chiamata SYN Cookie, *inclusa nella maggior parte dei sistemi operativi*.
Chiamato così perché funziona proprio come i cookie, lo stato viene mantenuto dal client stesso.
Viene calcolato un sequence number tramite una funzione hash, in cui come input viene usato l'indirizzo mittente, la porta mittente, l'indirizzo destinatario, la porta destinatario e un numero casuale che ha solo il server. Quest'ultimo in ricezione del pacchetto SYN del client, con le sue informazioni, a sua volta si calcola un sequence number sempre tramite una funzione hash.
A questo punto se il client è legittimo mi invia come terzo segmento, il segmento ACK con valore sequence number server + 1; un client non legittimo nemmeno mi risponde, ma tanto il sever in ogni caso non ha inserito alcun TCB record perché lo stato è contenuto dal client.

Principi del controllo di congestione

Una rete è congestionata quando tante sorgenti inviano dati a ritmi troppo elevati. Vediamo ora 3 scenari:

1. **Due mittenti e un router con buffer illimitati:**



Quando il tasso di arrivo dei pacchetti supera la capacità del collegamento uscente R, la rete diventa congestionata!
Fino a quando il throughput del ricevente non supera il valore R/2 *(-> il valore del ricevente equivale al tasso di invio del mittente)* la rete non è congestionata; se il tasso di invio supera R/2, il throughput resta R/2, quindi R/2 costituisce un limite superiore sul throughput.
Inoltre quando il tasso di invio si avvicina sempre più ad R/2 il ritardo medio cresce, questo perché avvicinarsi a quel valore vuol dire che troveremo sempre più pacchetti in coda, questi crescono sempre di più, comporta che il ritardo medio mittente-destinatario tende all'infinito.

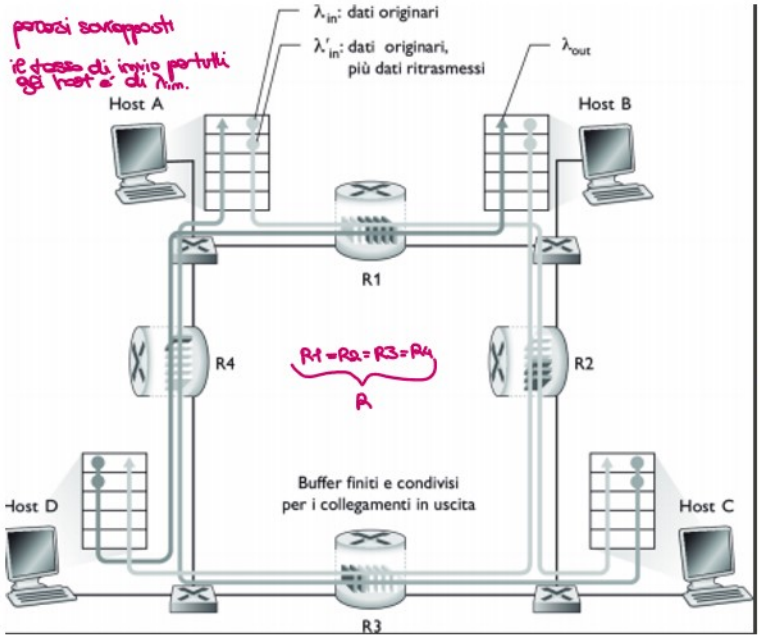
2. **Due mittenti e un router con buffer limitati:** i pacchetti che giungono in un buffer già pieno vengono scartati.

Siamo nello stesso caso dell'immagine di prima solo che questa volta il buffer ha capacità limitata, quindi abbiamo λin byte/s *(frequenza d'invio dei dati)* e λ' in byte/s *(il carico, dati nuovi e ritrasmessi, offerto dalla rete, offered load)*. Consideriamo tre casi:

- Caso in cui l'host è in grado di determinare se il buffer del router abbia spazio oppure no e di conseguenza trasmettere un pacchetto solo quando il buffer è libero; il throughput = λin byte/s e la velocità di trasmissione non supera R/2 perché non vengono persi pacchetti;
- Caso in cui il mittente ritrasmette solo quando è certo che un pacchetto sia andato perduto, il tasso di consegna all'applicazione destinazione cala a R/3, questo perché la perdita di pacchetti comporta una ritrasmissione.
- Caso in cui il mittente subisce un timeout prematuro e ritrasmette un pacchetto che ha subito importanti ritardi ma non è andato perduto, è uno spreco di risorse!

Questo è un altro costo legato alla congestione di rete: il mittente deve effettuare ritrasmissioni per compensare i pacchetti scartati a causa del fatto che il buffer era già pieno quando sono arrivati.
All'aumentare del tasso di invio, iniziamo a perdere pacchetti che dovranno essere ritrasmessi.
Il costo della congestione è un utilizzo inefficiente delle risorse.

3. **Quattro mittenti, router con buffer finiti e percorsi da più collegamenti:**



Se vaste R2, il traffico di A verso C è in competizione con il traffico di B verso D per lo spazio limitato nel buffer, all'aumentare del traffico di una coppia e'altra diminuisce...

A seconda del valore di λin distinguiamo 3 casi:

i. **Il valore di λin è estremamente piccolo:** in questo caso l'overflow del buffer è raro e il throughput è approssimativamente uguale al traffico inviato in rete (λin);
ii. Per piccoli valori di λin, **un incremento di questo valore provoca un incremento di λout** (throughput);
iii. **Caso in cui λin è molto grande, quindi anche λ'in** *(tasso invio dati originali + ritrasmissioni)* **lo è**.