Reti e Sistemi Operativi

DaveRhapsody

3 Ottobre 2019

Indice

Capitolo 1

La rete

La rete è un insieme di nodi, che sono tendenzialmente pc, server, terminali, e dispositivi radio che consentono ai dispositivi senza fili di collegarsi (per intenderci, gli access point wi-fi), ma attenzione!

Qui si parla di wi-fi, non della rete a cella telefonica, non si parla di 4G etc.

1.1 Il modello TCP/IP

Al centro è presente un Router, dispositivo in grado di INDIRIZZARE quelli che sono i pacchetti IP (Internet Protocol). Cos'è un pacchetto?

Pacchetto : E' un "Record/Struct" composto da un campo dati detto payload), ed un campo header, che serve al protocollo di riferimento come etichetta che dice (ad un protocollo) "Questo pacchetto lo puoi leggere". Cosa contiene l'Header?

$$header = \begin{cases} indirizzi \\ livelli\ di\ priorit \\ contatori \end{cases}$$

A che servono i contatori?
Immagina di avere un problema nella rete in cui appare un routing loop (pensate alla lista circolare)

I pacchetti è auspicabile che siano di dimensioni piccine, immaginate un film di dimensioni tendenti al Gigabyte, un pacchetto da un Giga è pressochè (ad oggi) insostenibile, pertanto, si riduce in pezzettini. Soprattutto c'è una probabilità di errore più bassa! (Pensate di mangiarvi una Fiorentina in un solo boccone o di tagliarla prima un pochino, stesso concetto.)

Un altro motivo per cui vengono usati i pacchetti è legato al fatto che inviare un qualsiasi dato occupa un canale. Intendiamoci, se da A a B (terminali) abbiamo un solo cavo, non c'è problema, no?

Ok, ma se A e B sono degli switch aventi 10 pc connessi assieme? Se uno deve caricare un giga di roba la gente deve aspettare che questo finisca MA utilizzando un sistema

a pacchetti, otteniamo che ci sia una sorta di "turno" per ogni pacchetto. Di conseguenza non si intasa la luce.

Il sistema sopracitato si chiama multiplazione o multiplexing statistico, perchè quando io metto tanti flussi di info che viaggiano sullo stesso collegamento, potrebbero accadere congestioni di rete, oppure comunque uno deve aspettare. Ecco i pacchetti dei pacchetti che attendono si chiama buffer. Più hai host di rete, più ti conviene aumentare la dimensione del buffer.

Lo switch Sono dei dispositivi che consentono di costruire la rete con più livelli, proprio perchè ad essi vengono collegati altri dispositivi. Uno switch utilizza connessioni Ethernet, ed una serie di nodi collegati ad uno switch forma una LAN. ATTENZIONE, quando si menziona l'access point, la rete che si forma si chiama WLAN, ovvero (Wireless LAN (Local area network)).

Terminale Il terminale (o host) sarà tendenzialmente un qualsiasi dispositivo che hosta applicazioni che lavorano in rete.

ATTENZIONE la rete è identificata dal Router.
Quando noi diciamo Internet, di fatto, stiamo parlando di un insieme di reti, appunto inter - net, più reti, ergo, più router. C'è un agreement totale che consente ad un terminale in una zona, per poter arrivare ad un'altra.

Vi ricordate il cammino dei nodi? Ecco, stesso concetto.
Questi nodi che si attraversano per passare da una rete A ad una rete B si chiamano autonomous systems, ed il cammino tra mittente e destinatario di un messaggio è identificato dagli autonomous per cui il pacchetto passo.

Più precisamente questi Autonomous sono dei Gateway, gestiti con un protocollo (BGP, Border Gateway Protocol) in grado di regolare le comunicazioni tra questi AS.
Ogni sistema autonomo, a sua volta all'interno avrà un certo insieme di altri router e reti, e quindi sarà presente un altro protocollo (IGP, Internal gateway protocol).

Sicurezza di un AS Ogni sistema autonomo è gestito da un "master" in grado di poter decidere cosa possa passare da quella rete, specifica le famose policy di Routing. La domanda che sorge è.. Chi gestisce gli AS in toto? I gestori della rete, quelli telefonici.

1.2 Cosa influenza le prestazioni di una rete?

Analizzeremo due quantità

Latenza/ritardo La latenza (o ritardo) non è altro che il tempo che intercorre tra l'invio e la ricezione di un pacchetto. Esistono diversi tempi:

- Tempo di processing: E' una quantità infinitesimale che indica il tempo impiegato per elaboare un pacchetto

- Tempo di coda: Per misurarlo devo capire oggettivamente quanto viene usata quella coda (esistono interi studi sulle code, ma a noi ci importa una bellissima ;) (Almeno, per questo corso si intende))

  Per un sistema stabile è auspicabile che il rapporto tra i dati contenibili nel buffer ed il rate di dati trasmessi al secondo sia ≤ 1, detto meglio:

$$\frac{nL}{R} \leq 1$$

Throughput Il through put è il quantitativo di dati che riescono ad essere trasferiti in un determinato quantitativo di tempo (Avete presente Mbps, Gbps, Kbps, quella roba lì)

1.3 Stratificazione

Il concetto di stratificazione consiste nel includere le funzioni di una rete non in un solo protocollo ma in una serie di protocollo che abbiano uno specifico compito, e sia indipendente dagli altri

La filosofia KISS Keep it simple stupid, è la filosofia che regola anche il mondo Unix e Gnu/Linux, ovvero, ogni componente fa il suo mestiere, fa la sua funzione, evitare di mettere in mano ad un componente 10 compiti diversi, per intenderci.

Inoltre dividendo in più componenti io ho possibilità di testare il singolo componente, il singolo protocollo, il singolo problema SENZA, S E N Z A andare a danneggiare o danneggiare gli altri protocolli/componenti!

Stack ISO/OSI Questo è lo stack più preciso ingrandito dell'TCP/IP, visto dal livello più basso al livello più alto

- physical
- data link
- network
- transport
- session
- presentation
- application

1.4 livello Physical

Nel livello fisico viene convertito quello che è il segnale elettrico in una sequenza di 0 ed 1

1.5 Livello Data Link

Si implementa MAC (Medium Access Control), converte quelli che erano degli 0 ed 1 e li fa diventare dei pacchetti (yes, c'è il primo header, cioè quello del physical).

L'indirizzo MAC identifica univocamente un dispositivo in rete, QUALSIASI esso sia. E non solo, mondialmente, N O N esistono due dispositivi DIVERSI aventi lo STESSO Mac address. A che serve oltre questo? Serve al protocollo ip per capire CHI è il mittente o destinatario, e sì, è incluso nell'header

1.6 Rete

E' il livello di internet, l'IP protocol, con indirizzo ipv4 v6 etc.

1.7 Trasporto

E' il livello in grado di trasportare tramite internet quelli che sono i pacchetti applicazione, trasporta i datagram ip, trasporta i pacchetti appunto tramite internet, sono presenti due protocolli (TCP e UDP) che si occupano di fare questo

1.8 Applicazioni

Il nome parla da sè, non lo studieremo in questo corso.

Qui noi si studierà dal data link al transport MA in ordine invertito, ovvero partiamo da "Che cosa vuole l'applicazione?" e poi andiamo a scalare.

Precisazione: Dato un pacchetto in un qualsiasi livello (escluso lv application), quando viene aggiunto l'header, il nuovo pacchetto potrà esser letto SOLO dal protocollo del livello superiore!

Su questo stesso piano, se ho un pacchetto con già l'header aggiunto, questo potrà essere spacchettato SOLO da un protocollo del livello inferiore.

Ogni livello offre servizio al livello immediatamente superiore od inferiore in base al verso del flusso dati.

La magica teoria (o ricetta) dell'incapsulamento di Dave Per spiegare meglio cosa accade, immaginatevi una fetta di prosciutto, bene, questa fetta di prosciutto rappresenta il flusso elettricoche vien gestito dal livello fisico.
A questo punto, immaginatevi due fette di pane che si chiudono sulla fetta di prosciutto. Benissimo, il Data Link ha **INCAPSULATO** il prosciutto ed ha creato un panino (Pacchetto)!

Il data link passerà questo panino al Network, e lì ci saranno altre due fette di pane che includeranno il nostro panino. Ed in questo modo abbiamo ottenuto un panino dentro ad un altro panino! Ossia, un pacchetto incapsulato in un pacchetto che semplicemente aggiunge un header (o intestazione) ;) Ma torniamo seri adesso.
Premete qui per vedere un video corso che spiega con precisione l'incapsulamento.

Capitolo 2

Livello Trasporto

Il livello di trasporto è il livello che riceve messaggi dal livello application ed ha come compito quello di mettere in comunicazione end-to-end due nodi. Nel caso del livello di trasporto non si parla di pacchetti ma di Segmenti (Dall'esempio sopra riportato sì, è stato incapsulato il pacchetto del lv network).

Quando viene analizzato il segmento si effettua la DEMULTIPLAZIONE (Demultiplexing), ovvero si analizza l'header del segmento per vedere a quale applicazione sarà destinataria di un determinato messaggio.

Cosa contiene un segmento di trasporto?

- Numero di porta: presente in ogni segmento, rappresenta una applicazione e dal punto di vista dell'applicazione è come se fosse un punto di accesso, detto anche SOCKET.

  NON CONFONDIAMOCI Il livello transport vede una porta, mentre il livello applicazione vede un socket. All'atto pratico c'è una syscall che letteralmente può attivare qualsiasi socket.

- Protocollo Transport TCP o UDP.

Che differenza c'è tra TCP e UDP? UDP è quello che usate negli streaming di Rojadirecta , in cui pur di vedere la vostra squadra (che ovviamente è la Fiorentina, vero? <3) non vi preoccupate di perdere qualche dato (tipo immagine un po' sgranata ogni tanto e simili).

Infatti UDP se ne frega se un pacchetto è arrivato, è usato per comunicazioni anche per esempio le voice chat, roba di questo genere, poichè **NON** effettua alcun controllo sulla correttezza dei messaggi, diciamo che lui ti invia roba, se non ti arriva non gli importa, va avanti, perchè qui è importante che si sia il più aggiornati possibile.
Al contrario TCP effettua questi controlli, perchè con TCP si garantisce la correttezza (sia dei dati stessi che dell'ordine in cui arrivano), pertanto potrebbe essere un problema.

Il segmento può avere sia TCP che UDP, e nel caso di UDP si ha un IP + Porta di

destinazione, o meglio

$$PacchettoUDP = \begin{cases} SourcePort: \ Porta\ sorgente \\ DestinationPort: \ Porta\ di\ destinazione \\ Lunghezza: Indica\ la\ lunghezza\ totale\ del\ segmento\ (+payload \\ dei\ livelli\ inferiori) \\ Checksum: \ Verifica\ la\ correttezza\ di\ trasmissione \end{cases}$$

Supponiamo di avere un web server, (quello che consente di vedere una pagina web) , la porta d'accesso è la stessa ma hai più utenti con la stessa pagina

Nel caso di TCP invece un segmento è composto non solo dall'IP + la porta di destinazione, ma è composto in questo modo:

$$\begin{cases} IP_A \\ IP_B \\ PORT_A \\ PORT_B \end{cases}$$

A e B in questo caso sono tendenzialmente sorgente e destinataria, ed il calcolo del checksum precedente è effettuato direttamente dal livello Transport stesso.

Attenzione, nell'introduzione abbiamo anche detto che già nel datalink abbiamo dei controlli, pertanto ogni livello può svolgere la correzione dei pacchetti, pertanto non si attiva quasi mai, perchè Transport dà per scontato che Network e Datalink gli abbian passato qualcosa di corretto.

Il codice di implementazione del controllo è tipo di migliaia di righe di codice per correggere o prevenire errori, ed è giustissimo così. Contiamo che ormai TCP e UDP non si possono rompere, sono più che sicuri.

2.1 TCP

Il concetto è che si vuole trasferire dati in modo affidabile, o meglio, si vuole fare in modo che i dati arrivino a destianazione e siano corretti in generale.

Il trasferimento dati affidabile dipende da una serie di fattori, e si verifica quando ho una rete con le seguenti caratteristiche:

1. Nessuna perdita

2. Nessun errore

3. I dati son corretti e presenti tutti quanti

Nel caso avvengano errori si può ricorrere a diverse metodiche di soluzione, come ad esempio la tecnica di ritrasmissione. Ossia, non ti è arrivato il mio pacchetto? Ok, te lo rimando. Nell'atto pratico è una cosa del tipo

$$\begin{cases} if(ricevoAcknoledgment)\ then\ mandoSucessivo(); \\ else\ if(ricevoNotAcknoledgment)\ rimandoPacchetto(); \end{cases}$$

Quello che chiamo not acknoledgment (che da ora chiameremo solo ACK) viene inviato SE il pacchetto è non corretto, per esempio per il checksum.

E se il pacchetto poi è lo stesso? Ne ricevo due? No. I pacchetti sono numerati hanno un ordine, pertanto

$$\begin{cases} if(ricevoACK)m+1; \\ elsem \end{cases}$$

con m che è il "pezzetto" successivo o precedente, è importante che si capisca questo.

Ora cerchiamo di integrare, è necessario che il sistema mi mandi indietro sì se ha ricevuto ma ANCHE la posizione a cui è arrivato! (Sistema PAR, positive acknoledgmente with retransmition).

Quindi mi mandi un messaggio e io ti rispondo dicendo ok messaggio, tu mi mandi il due, io rispondo ok due, tu mi mandi tre io rispondo ok due, mi rimandi tre.

Perdite Sì, ok, ma se uno non mi riesce a rispondere? Si imposta un semplice timer la cui scadenza implica che ti rimando il pacchetto. Un po' come quando dici una cosa con la musica alta a uno e lui ti guarda tipo fisso... Passa tempo, e poi ripeti la cosa.

E se io continuo a rispondere? Lui che fa continua come un pazzo a ripetere la stessa cosa come Ciuchino che chiedeva a Shrek ogni 2 millisecondi "Siamo arrivati?"? NO! C'è un contatore che si aziona ad ogni mancata risposta, una volta che arriva ad un tot, chiudo la comunicazione.

$$\begin{cases} if(timerGoesOff)\ \ trasmetti(messaggio_i)\ AND\ setTimer(0); \\ else\ if(ACK_i)\ trasmetti(messaggio_{i+1}\ AND\ setTimer(0)); \\ else\ trasmetti(messaggio_i)\ AND\ setTimer(0); \end{cases}$$

ATTENZIONE Supponendo un timer troppo corto, accadrebbe che il destinatario non fa in tempo a rispondere, e quindi il mittente rifà la domanda prima di ricevere la risposta, hai già due domande con una risposta.

All'atto pratico questo problema esiste davvero, è un problema reale, pertanto si deve riuscire a capire il modo in cui il timer vada adattato alle condizioni di rete, che a seconda del traffico può essere più lenta o anche più veloce.

2.1.1 I tempi di una connessione

Riprendiamo il concetto di latenza (tempo che intercorre tra invio e ricezione), bene, quello è concettualmente il tempo di trasmissione, da qui possiamo calcolare che

$$utilizzoRete = \frac{\frac{L}{R}}{latenza + \frac{L}{R}}$$

in cui $\frac{L}{R}$ Ha R che è la velocità di trasferimento dei pacchetti, mentre invece L sarebbe la lunghezza , ossia: $\frac{L}{R} = \frac{Lunghezza}{Velocita'\ trasmissione}$ E la latenza... E' la latenza.

2.1.2 Protocolli a Finestra Scorrevole (Sliding Windows)

Io trasferisco un insieme di w pacchetti, i quali impiegheranno $w * \frac{L}{R} \geq \frac{L}{R} + RTT$ tempo ad esser trasposti, pertanto ne esce che

$$w \geq \frac{RTT \cdot R}{L} + 1$$

Questo ovviamente in mancanza di errori e perdite.

Osservazione: Stiamo ragionando su sistemi concettualmente indistruttibili, quindi diciamo che non si è quasi menzionata la possibilità che un host possa disconnettersi.

2.1.3 Go Back N

Poniamo caso che io trasmetto 5 messaggi: $m_{1,\dots,5}$ e mi arriva l'ok di ricezione dall'1 al 3, malgrado li abbia trasmessi tutti e 5, pertanto ciò che accade è che rinvierò dal messaggio 3 in poi.

2.1.4 Selective Repeat

Traccio la vita di ogni pacchetto individualmente, ossia ritrasmetto solo il pacchetto mancante, il problemino è legato al fatto di avere tanti timer, ma oggi c'è da considere che abbiamo processori nell'ordine dei Ghz, no problem.

Quando all'epoca il massimo processore esistente era di massimo 10 Mhz, insomma chiaro che ora essa sia la migliore delle soluzione. Ma cosa richiede più precisamente l'SR?

- Ack individuali per ogni pacchetto
- Un timer per ogni singolo pacchetto
- Buffer sul ricevitore (E' un array di booleani all'atto pratico)

Ci sono dei casi in cui ci sono interazioni tra numero di pacchetto e dimensione finestra, ma sono problematiche di cui non approfondiamo perchè non ci interessa siccome non può praticamente mai capitare