Reti e calcolatori

Introduzione

Una rete è un insieme di macchine collegate tra di loro attraverso una sottorete di comunicazione (una connessione di fili, macchine e applicazioni distribuite).

Il problema all'interno di una rete è che le macchine devono comunicare tra loro e non sono sincronizzate le une con le altre e quindi non essendo sincronizzate non posssono scambiare informazioni.

Come soluzione a tale problema nascono i protocolli, ossia insieme di regolee software che permettono il dialogo tra diverse macchine in una rete.

Tipi di reti (classificazione geografica)

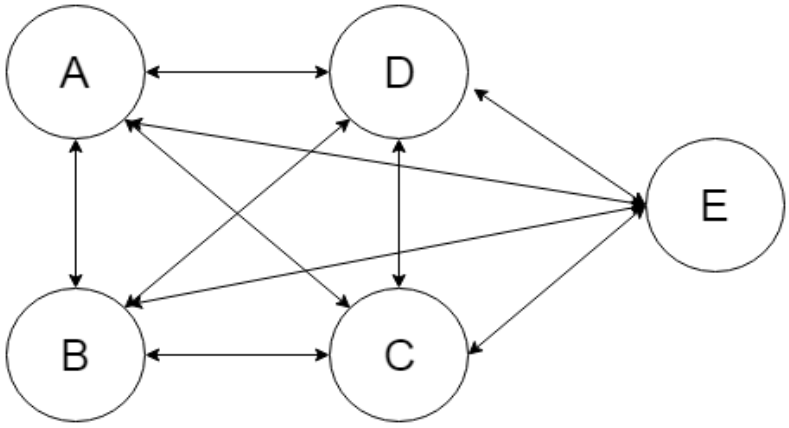
- LAN : Local Area Network (es. La rete di un ufficio)
- WAN : Wide area network (reti geografiche)
- MAN : Metropolitan Area Network (geografiche più estese)
- Internet : Rete Interconnessa (a livello globale)

Le sottoreti

La sottorete di comunicazione più informalmente vine e chiamata “rete”.

Come fa una app su un host A a comunicare con un altro host B?

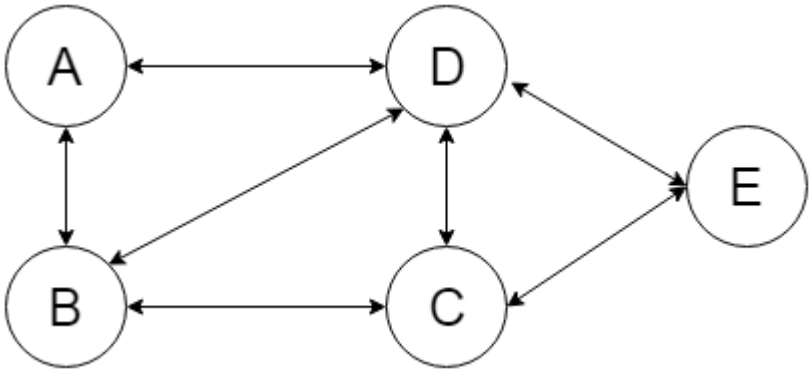
La rete è possibile rappresentarla mediante un grafo.



Questa tipologia di rete a maglia totalmente connessa non è tuttavia consigliabile perché la sua costruzione diventa troppo dispendiosa.

Infatti in questo caso ogni host ha bisogno di un cavo per ogni macchina con cui deve comunicare. Una maglia totalmente connessa tuttavia è molto efficiente: ogni comunicazione avviene tramite un solo hop.

Quali altre soluzioni esisono?



Questa soluzione parzialmente connessa non è efficiente perchè la comunicazione tra due host potrebbe richiedere più di un hop.

Tuttavia abbiamo ridotto il numero di cavi che devono essere connessi ad ogni host.

E’ meno affidabile della maglia totalmente connessa perché se si dovesse rompere un cavo è possibile che si interrompa la comunicazione.

Nonostante sia meno affidabile, è la soluzione che viene adottata ora.

Se prima con 1 hop impiegavo 1 secondo, adesso per andare da A ad E impiego 3 hop, quindi 3 secondi.

Come migliorare la comunicazione quindi? Posso aumentare la velocità della comunicazione tra ogni host in modo da far durare ogni hop 1/3 sec, quindi 3 x 1/3 = 1 secondo per effettuare 3 hop.

Passando dall'utilizzo del cavo in rame alla fibra ottica l'affidabilità aumenta (nessuna interferenza elettrica etc…) e quindi un eventuale errore di comunicazione risiede negli hosts e non sui cavi.

In una maglia parzialmente connessa è determinare qual è la strada che le informazioni devono percorrere.

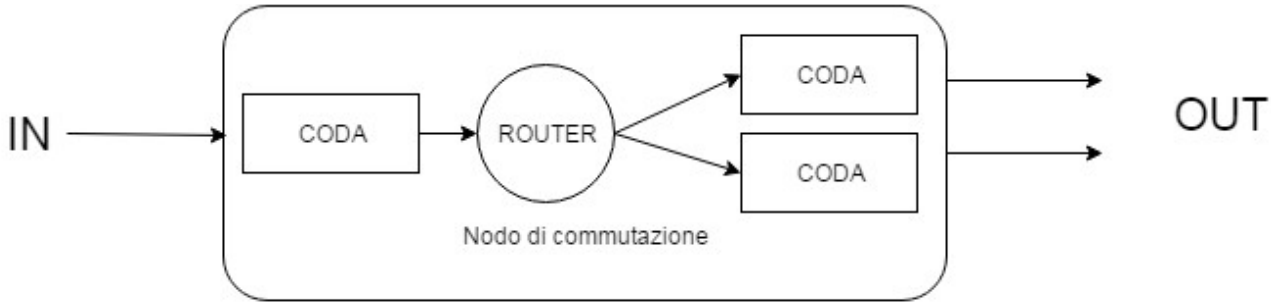
Per esempio tutto ciò che deve andare da A ad E posso dire che deve passare per B, quindi A saprà che se deve comunicare con E deve passare da B.

Allo stesso modo tutto quello che deve andare ad E da B dovrà essere mandato a C il quale se ne occuperà.

Diverse possibilità esistono per creare questi percorsi. Una possibilità è quella scrivere nella configurazione degli host queste informazioni: tale modalità è scomoda perché non permete la flessibilità della rete e ci potrebbero essere problemi nel caso un host si staccasse dalla rete o uno nuovo si inserisse.

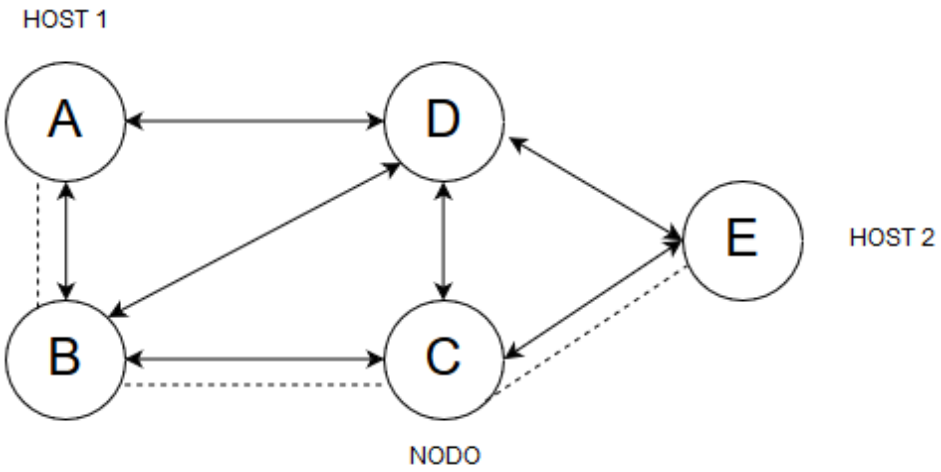
Dinamicità delle reti e nodi

Risulta necessario introdurre dinamicità nella rete.



Il router permette l'instradamento dei pacchetti ma se ha troppi dati in ingresso (riempie la coda) allora sarà obbligato a fare drop di alcuni pacchetti (vengono ignorati e quindi buttati via) perdendo così informazioni durante la comunicazione: da qui deriva una delle fonti di errore all'interno delle reti.

Instradamento → IP → Sottorete di comunicazione



Un pacchetto che deve andare da A ad E deve attraversare diversi nodi (routers) i quali dovranno :

→ instradare : smistare le richieste in base alle sue regole

→ indirizzare

Metriche utilizzate per la valutazione della rete:

- Tempo impiegato
- Hop (numero di nodi da attraversare) necessari

L'identificatore dell'indirizzamento è l'IP (Internet Protocol address).

Ogni device connesso in rete ha un indirizzo IP.

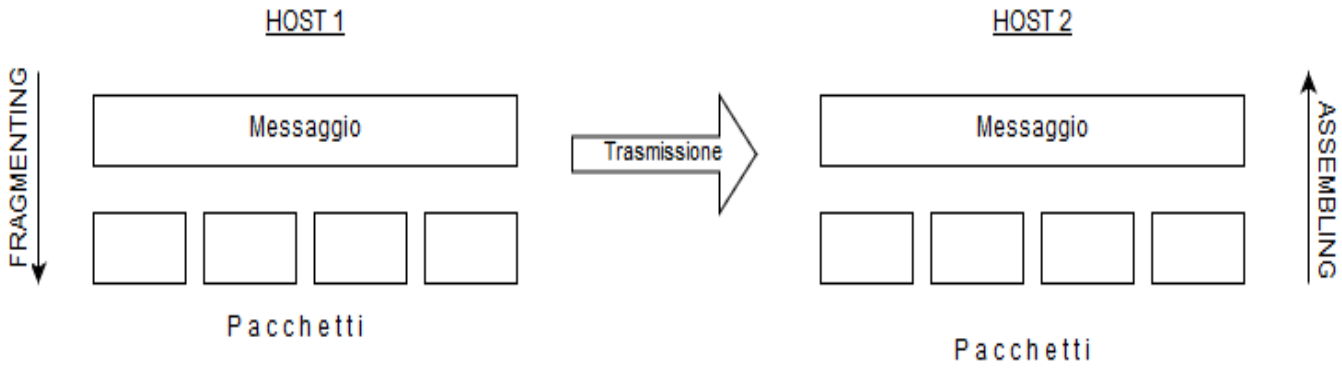
Pacchetti

Se ho la necessità di inviare un file di diversi MB dall' host 1 all' host 2 non ho la possibilità di inviarlo integralmente ma sarà necessario frammentarlo in tanti pacchetti più piccoli.

Dato che sulla rete sono diversi i pacchetti che transitano da e per diversi hosts, risulta necessario dimensionare i pacchetti in modo tale che tutti abbiano la possibilità di parlare sulla rete (come risulta migliore la paginazione rispetto alla segmentazione nei sistemi operativi).

L'utilizzo di questa modalità di frammentazione permette inoltre una più comoda comunicazione nel caso un pacchetto venisse droppato da un router (o ci fosse un errore al suo interno) e dovesse essere ritrasmesso: non servirà ritrasmettere tutto il file di k MB ma solo il singolo pacchetto che non è arrivato a destinazione o risulta corrotto.

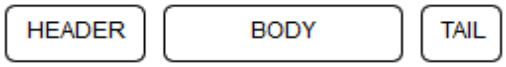
L'host 1 (il mittente) dovrà frammentare quindi l'unità dati iniziale (il messaggio) in tanti pacchetti.



A livello di sottorete si perde l'idea di file in quanto la rete vede solo pacchetti e non i files in trasmissione.

Come è fatto un pacchetto?

- Header:
  - Destinazione
  - Sorgente
  - Numero di segmento
- Body : dati da trasmettere (payload)
- Tail : codice di rilevazione errori



Il numero del pacchetto è utile perchè essendo l'instradamento dinamico, i pacchetti possono prendere strade diverse e arrivare non in ordine e quindi dover essere ricostruiti all'arrivo e riordinati in base al numero contenuto nell'header.

In base alla tecnologia utilizzata, i pacchetti potrebbero avere dimensioni diverse.

Per evitare problemi di comunciazione, i pacchetti vengono dimensionati uguali in base ad uno standard o di dimensione minima in modo che possano essere trattati allo stesso modo da qualunque tecnologia si stia utilizzando.

Ethernet ed errori

La tecnologia predominante attualmente è Ethernet che è una famiglia di tecnologie standardizzate per le reti locali sviluppato.

La dimensione di payload del pacchetto Ethernet è minimo 46 bytes e massimo 1500 bytes.

Quando la dimensione del payload non raggiunge i 46 bytes viene aggiunto un padding.

Solitamente viene usato un padding che permette di raggiungere dimensioni standard indipendentemente dalla lunghezza minima del pacchetto.

In caso di errore rilevato dalla tail (correction code) viene effettuato un recovery tra gli hosts e non a livello di sottorete.

Errori :

1. pacchetto mancante
2. pacchetto errato → viene droppato

Probabilità di errore sul bit cavi in fibra ottica ≈ 10^{-12}

Probabilità di errore su un pacchetto ≈ N x P_{bit} → Numero di bit x probabilità di errore sul singolo bit

Bisogna perciò trovare il giusto trade off tra dimensione del pacchetto e probabilità di errore.

Da sorgente a destinatario i pacchetti possono passare attraverso diverse sottoreti (non solo una).

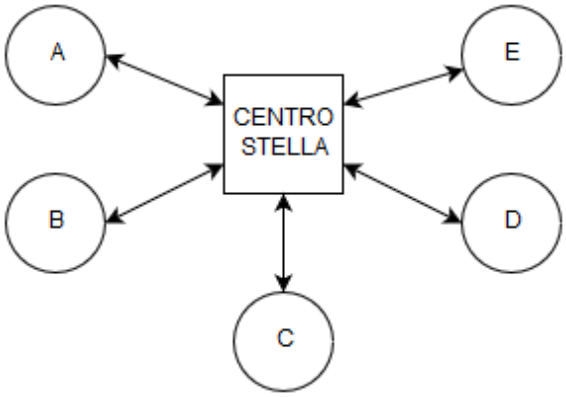
Esistono infatti due livelli di astrazione:

1. Interna alla sottorete
2. Esterna alla sottorete

ed i problemi di instradamento si presentano in entrambi i livelli.

Tipologie di broadcast

STELLA CON CENTROSTELLA PASSIVO



E' una modalità in cui il centro stella propaga a tutti i nodi ciò che passa da lui.

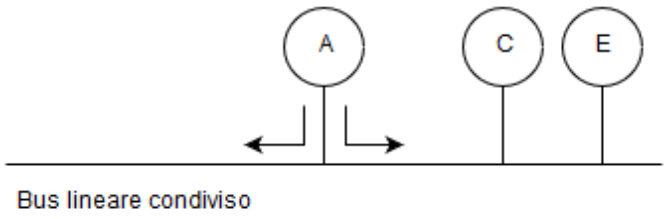
Non legge gli header che transitano per lui.

In questo caso non c'è problema di instradamento in cui ogni pacchetto viene inviato a tutti.

Questa è una modalità in cui il canale di trasmissione viene condiviso da più risorse: le tecniche così strutturate vengono chiamate reti ad accesso multiplo , il formato del pacchetto rimane invariato e si fa ancora indirizzamento.

ETHERNET

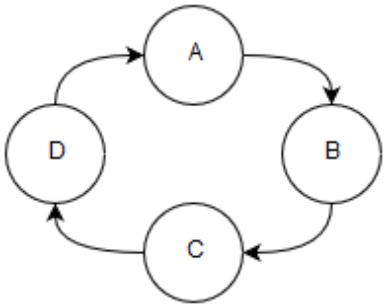
La risorsa è condivisa (per esempio un cavo coassiale) al quale si collegano tutte le macchine e quando un host immette bits sul canale, tutti lo ascoltano se non ci sono state collisioni (due host stanno trasmettendo nello stesso momento).



Bus lineare condiviso

In questo tipo di rete, una macchina può collegarsi e staccarsi dalla rete quando vuole. Il tutto continua a funzionare.

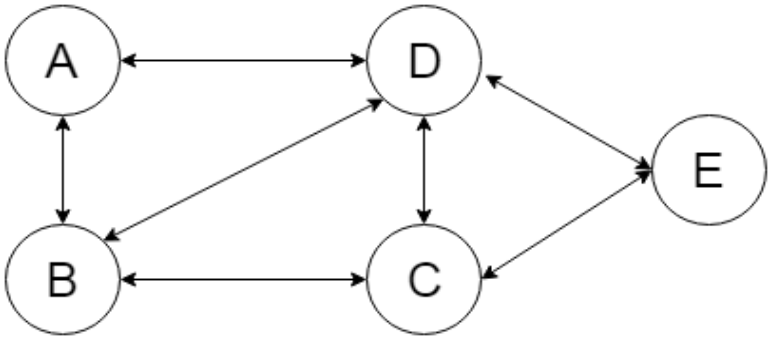
ANELLO UNIDIREZIONALE



Al contrario del bus condiviso ethernet, questo modello ha delle limitazioni. Infatti un nodo non può collegarsi o scollegarsi quando vuole dalla rete, pena la perdita della connettività di tutti gli altri hosts.

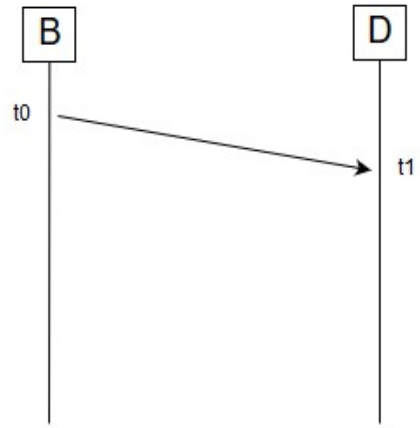
Store and forward: tecnica mediante la quale un'informazione suddivisa in pacchetti deve essere ricevuta integralmente (store) prima di poter essere rispedita (forward).

<u>Trasmissione e recovery</u>



Tornando alle reti parzialmente magliate, come è possibile effettuare il recovery di un pacchetto arrivato errato o non arrivato del tutto? Quanto tempo impiega la trasmissione?

$t_1 - t_0 = t_x + \ldots$

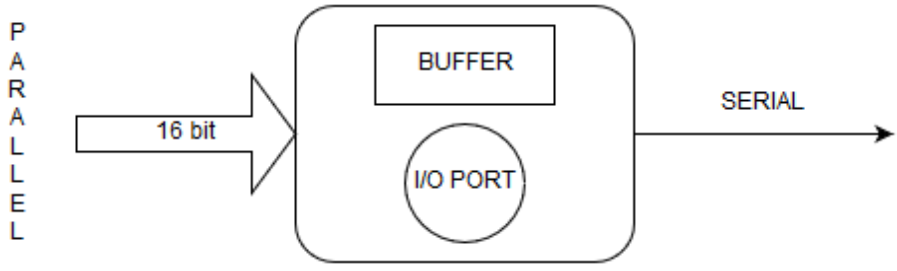


x → tempo di trasmissione che impiega B a trasmettere un pacchetto a D (dipende dalla velocità del canale e dalla dimensione del pacchetto).

La distanza non conta?

Parallel Input Serial Output : tecnica per trasmettere dal buffer a seriale.

Necessita l'utilizzo di un clock.



$$t_x = 16 / C_k$$

dove C_k è la frequenza di clock

Nella formula non è presente la distanza.

Il trasmettitore deve essere sincronizzato con il ricevente e viceversa.

$$t_x = L / V$$

Lunghezza del pacchetto / Velocità di clock di trasmissione

In realtà bisogna aggiungere un valore che è il tempo di propagazione t_p .

La formula diventa quindi:

$$t_1 - t_0 = t_x + t_p$$

t_p è il tempo che impiega il cavo a trasmettere.

$$t_p = D / C$$

Distanza / velocità di trasmissione

C = 3 x 10^8 mt/sec (la velocità della luce per la fibraottica)

C = 2 x 10^8 mt/sec (per il rame)

Esempio:

3 km di fibra ottica

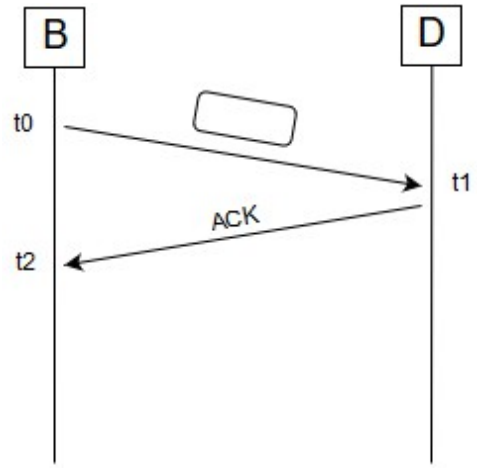
$$t_p = (3 \times 10^3) / (3 \times 10^8) = 10^{-5} \text{ secondi}$$

Inoltre per ogni hop, in base alla tecnologia in utilizzo è necessario avere dei limiti di lunghezza sui cavi.

Come effettuare il recovery di un pacchetto?

D dovrebbe verificare che B abbia inviato tutto correttamente.

Nel momento in cui B invia il pacchetto a D, questo si prepara ad inviare un altro pacchetto speciale chiamato “Acknowledge” (validazione).



Il tempo tra t_1 e t_2 non viene considerato nel calcolo del tempi di trasmissione perchè ack è molto piccolo e non influisce sul tempo di trasmissione (è formato dall'header ed un boolean).

D manda sempre un ack in modo da notificare B sullo stato di ricezione.

t_p rimane invariato in quanto non contiene la dimensione del pacchetto.

Quanto tempo impiega B a sapere se la trasmissione è andata a buon fine?

$$\Delta T = t_2 - t_0$$

ΔT è chiamato Round Trip Time ossia il tempo di invio e ricezione di una risposta di un pacchetto tra due hosts.

B avrà incorporato un timer che si preoccuperà di definire dei lassi temporali tali che B possa aspettarsi il pacchetto di acknowledge da parte di D. Ovviamente il timer di B sarà dimensionato tale da contenere il Round Trip Time.

Si presentano in questo caso due scenari differenti:

- il mittente riceve l'ack entro lo scadere del timer → comunicazione andata a buon fine. Il mittente cancella il pacchetto e resetta il suo timer.
- Il mittente non riceve l'ack → il mittente va nel suo buffer e rinvia il pacchetto.

Lo svantaggio di questo metodo è che bisogna utilizzare e mantenere sincronizzato un timer e che il trasmettitore deve aspettare ΔT per poter inviare il pacchetto successivo.

$$RTT = t_x + 2t_p$$

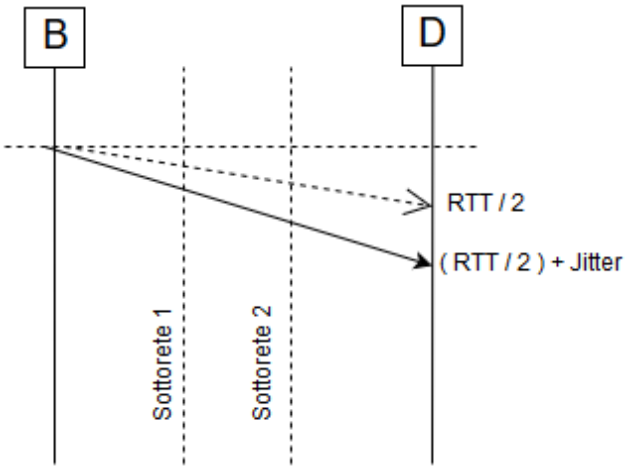
ΔT o RTT () quindi è formato dal tempo di trasmissione del pacchetto più 2 volte il tempo di propagazione del pacchetto (per l'ack non si considera la dimensione del pacchetto perchè troppo piccolo ma si considera solo il tempo di propagazione).

Esistono due modalità di trasmissione di un pacchetto:

- Best effort : la sorgente invia tutto il pacchetto "facendo del suo meglio" ma non si preoccupa se arriverà a destinazione correttamente l'informazione (non si preoccupa di ascoltare gli ack del ricevente) → Ethernet
- Rete a circuito : l'opposto di best effort.

Le reti che utilizziamo normalmente sono best effort. Solo a livello 3 (3 TCP/IP → livello 4 ISO/OSI) (TCP protocollo end-to-end non point-to-point) avviene normalmente il recovery lavorando con gli ack tra i due punti estremi (source e destination).

In realtà lo Standard Ethernet è stato costruito per funzionare sia best effort che a circuito ma quello a circuito di solito non viene utilizzato.



Jitter : parametro caratteristico di una sottorete. Varianza del tempo di trasmissione.

Rappresenta il tempo di computazione del pacchetto durante il transito nei nodi (tempo di permanenza nei buffer dei router...).

Il jitter si presenta particolarmente nel traffico multimediale in quanto il multimedia deve essere riprodotto fluidamente (basti pensare al buffering dei video) ad un determinato Constant Bit Rate (da garantire per un servizio di qualità come skype, video etc..). Servirà impostare delle priorità sui pacchetti per il corretto smistamento di tali in base al servizio che portano.

Protocollo

Un protocollo è un insieme di regole.

I protocolli sono software installati su hosts e device nella trasmissione tra mittente e destinatario che permettono di scambiare informazioni (pacchetti).

Le informazioni scambiate possono essere dati ma anche informazioni di controllo come gli ack.

Oltre ai dati, una parte del pacchetto dati è informazione di controllo e si chiama “header”.

Queste info di controllo servono per sincronizzare la comunicazione.

Più informazioni di controllo rendono meno efficiente la comunicazione (minore è il numero di pacchetti di controllo e più piccoli gli header, maggiore è l'efficienza).

Errore nel pacchetto = (grandezza del pacchetto) x (probabilità di errore sul bit)

- un pacchetto troppo grande ha più rischio di essere inviato in modo errato
- un pacchetto piccolo utilizza meglio la memoria ma il suo header inciderà maggiormente
- un pacchetto deve avere un buon trade off tra le varie variabili in gioco nella comunicazione.

I protocolli devono essere standard.

IETF (Internet Engineering Task Force) : organismo internazionale per la standardizzazione di TCP.

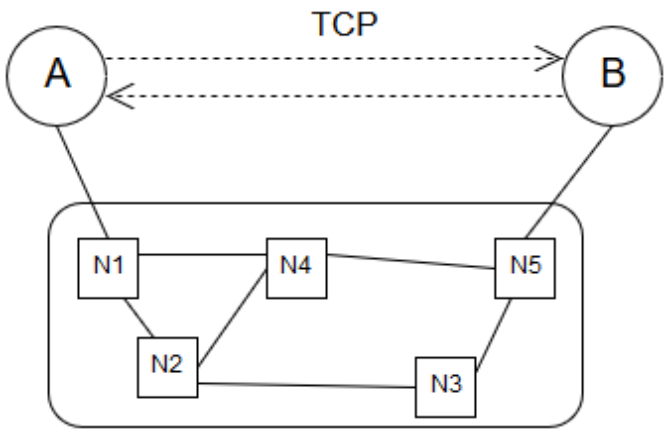
Un protocollo è affidabile se riesce a ritrasmettere gli eventuali pacchetti corrotti.

TCP lavora tra A e B ma la comunicazione avviene attraverso la sottorete in quanto non esiste un collegamento diretto tra i due host.

Da qualche parte ci deve quindi essere un modulo che lavora nella sottorete per instradamento e addressing.

N1 e N5 :

- hanno 1 cavo di I/O
- funzione di indirizzamento
- funzione di instradamento

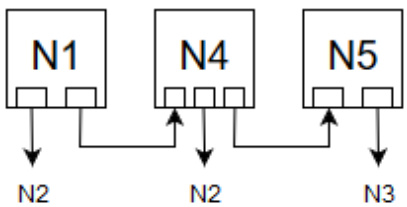


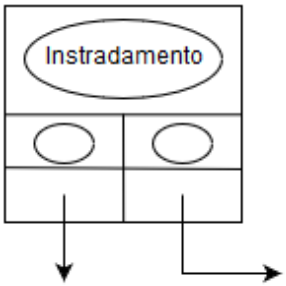
<u>Instradamento e indirizzamento</u>

La velocità nominale di un cavo è poco utile se l'architettura sopra non è adatta alla tecnologia di trasmissione utilizzata. Inoltre la velocità di trasmissione sul cavo è diversa da quella che A e B sentono realmente.

La funzione di instradamento ha una visione multi-hop e fa anche da indirizzamento.

Risiede su ogni nodo.





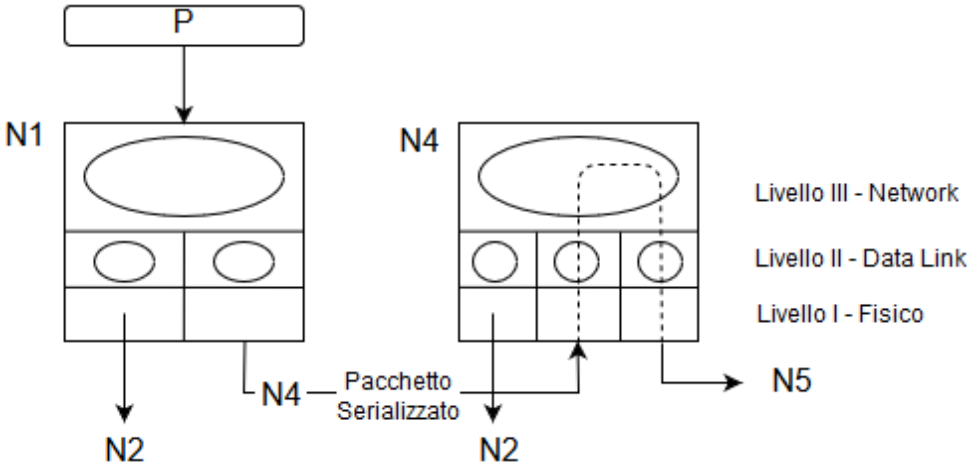
Indirizzamento invece è miope alla rete sottostante.

Ogni canale che utilizziamo (rame, fibra …) ha un blocco in input e uno in output.

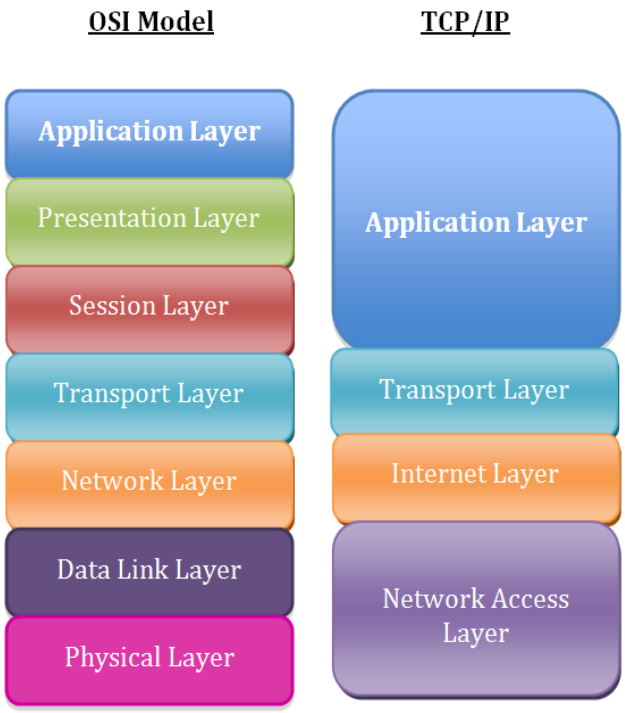
La serializzazione avviene alla velocità dettata dal clock di trasmissione.

Cosa succede quanto un pacchetto arriva ad un router?

La destinazione è scritta nell'header. Il pacchetto perciò viene aperto e letta l'intestazione. Successivamente il router guarda nelle sue tabelle qual è il percorso più veloce per l'inoltro del pacchetto.



Modelli ISO/OSI e TCP/IP



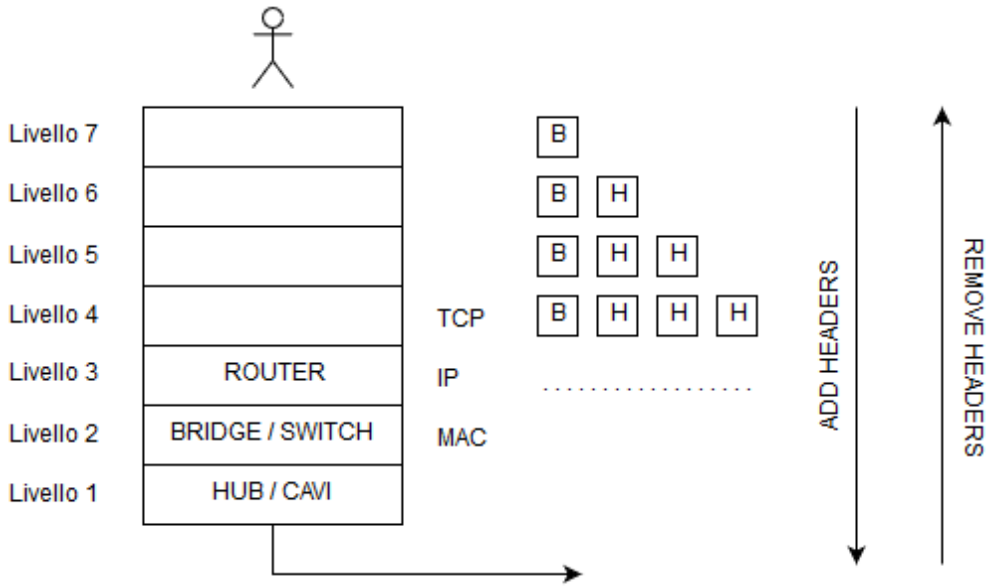
A livello 4 ISO/OSI può esserci TCP (Transmission Control Protocol) o UDP (User Datagram Protocol) che non garantisce affidabilità.

Di solito si utilizza TCP + best efforta livello 2 in quanto chiedere affidabilità a livello 2 sarebbe troppo costoso.

Tra i livelli > 1 dei vari nodi ci sono dei collegamenti logici e non fisici (al contrario del livello 1).

Il livello 4 ISO/OSI ha come protocollo TCP ed è il primo protocollo end-to-end che non risiede sui nodi intermedi come routers ma solo sugli hosts.

Socket → API di livello 4



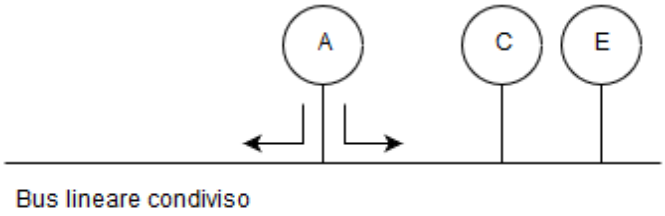
In una rete broadcast non ho la necessità di instradare.

A livello 3 il router viene usato per connettere pc in rete esterna (non interni ad una lan).

A livello 4 ci sono le socket che sono le API di TCP / UDP.

Ethernet

Se due hosts devono comunicare, è possibile che utilizzino insieme il canale per scrivere dati. Nel caso in cui perciò due hosts immettono bits nel canale allo stesso momento, si crea un problema di collisione.



Risulta necessario utilizzare un protocollo che faccia accedere gli hosts in mutua esclusione.

Si potrebbe utilizzare un token condiviso in modo che solo chi ha il token in modo deterministico possa utilizzare il canale per trasmettere.

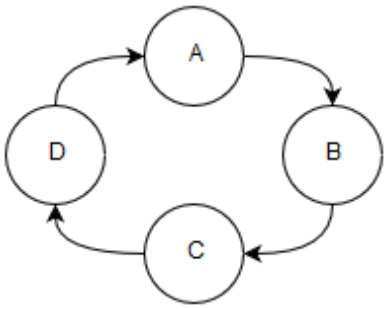
Si crea però un problema di fairness. Come assicurare che un host rilasci correttamente il token?

Se il token si perde?

Token ring non funziona perchè le reti devono essere dinamiche perciò il determinismo non funzionerebbe.

Risulta necessario accedere ad una rete dove non ci sia una macchina dedicata a semaforo.

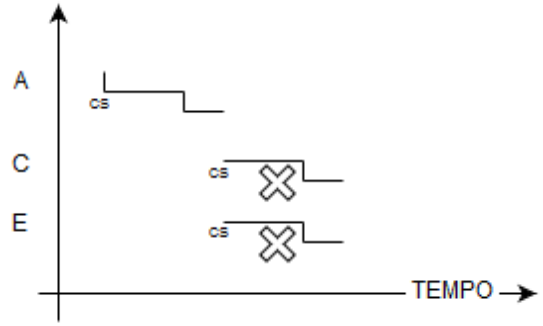
Nasce perciò Ethernet che lavora a livello probabilistico.



Ethernet infatti tollera la collisione e da un algoritmo di risoluzione.

L'efficienza di tale protocollo è dato dall'accesso aggressivo al canale condiviso.

L'efficienza è inversamente proporzionale al numero di nodi.



I nodi per verificare che non ci sono comunicazioni in corso, effettuano quello che viene chiamato "Carrier Sense" ossia si mettono in ascolto del canale per sentire se ci sono altri dati che stanno transitando sul canale.

Se nessuno sta trasmettendo, allora iniziano la trasmissione.

In questo caso abbiamo che C ed E fanno carrier sense nello stesso momento e quindi iniziano a trasmettere, collidendo.

Come capisce un nod se c'è stata collisione? Se ascoltando il canale il nodo si accorge che ciò che riceve è ciò che sta inviando allora è l'unico che sta trasmettendo. Al contrario se vede che sta collidendo, interrompo la trasmissione ed entrambi liberano il canale.

In questo caso però i due hosts non dovrebbero rincominciare la trasmissione dopo un tempo finito identico perchè collliderebbero ancora.

Quando allora C ed E possono riprovare la trasmissione del pacchetto? In base alla funzione Binary Exponential Backoff

$$\tau (0 \sim 2^{i} - 1)$$

dove τ è il doppio del peggior tempo di propagazione registrato ($2t_p$) ed i è il numero di collisioni precedenti sullo stesso pacchetto (i = 0 ... 16). Il valore di i rimarrà 16 fintanto che c'è collisione, altrimenti se la collisione sul pacchetto è stata risolta, il valore di i ritorna a 0 per il pacchetto successivo.

CSMA – CD (Carrier Sense Multiple Access with Collision Detection) → Ethernet

- 1 – persistente (appena trovo libero con cs, trasmetto)
- 0 – persistente (attendo un tempo di wait prima di trasmettere dopo cs – non è il caso di ethernet)

Se ci sono troppi hosts, è consigliabile dividere la rete a metà per abbassare la probabilità di collisione (per esempio usando un bridge).

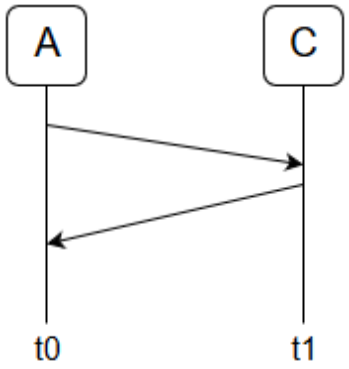
Quando un host esegue cs, sente tutta la rete a meno di quelle che si stanno ancora propagando (tempo di propagazione t_p).

$$t_p = d / 2 \times 10^{8}$$

distanza / velocità sul cavo o fibra ottica

Significa che il tempo che impiega un bit in trasmissione, potrebbe essere usato da un altro host per fare cs e trovando il canale libero, iniziare la trasmissione collidendo.

Infatti collision detection e carrier sense non sono istantanei.



C → t1 – t0

A → 2(t1 – t0)

Inoltre A non può trasmettere per un periodo di tempo inferiore a 2(t1 – t0) perchè non verrebbe rilevata la collisione (ossia A finirebbe di trasmettere prima che il primo bit di C arrivi ad A e venga rilevata la collisione). Servono perciò delle dimensioni minime e massime di pacchetto.

Esempio:

Cavo COAX di 2,5 km suddiviso in 4 segmenti con repeaters (propagatori di segnale per evitare l'indebolimento del segnale e la relativa perdita di informazione da parte del ricevente)

d = 2,5 km

tp = $(2{,}5 \times 10^3)$ / (2×10^8) = 12,5 µs

2tp = 25 µs

dimensioni pacchetto : 512 bit – 12.000 bit (64 bytes – 1.500 bytes)

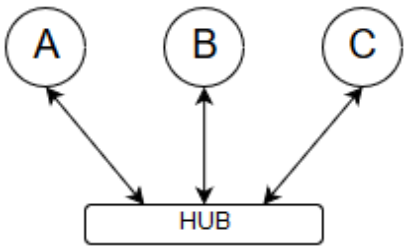
Tempo totale = 25 + 26,2 = 51,2 (2tp + tempo di ritardo sui repeater)

τ (0 ~ 2^i – 1) → (0 ~ 1) x 51,2 µs (la prima volta)

Un hub è un dispositivo di livello 1 (fisico) su cui non risiede alcuna scheda di rete. E' un dispositivo stupido che semplicemente inoltra i pacchetti a tutti i device connessi (come nel centro stella passivo).

Le schede di rete risiedono sui nodi A, B, C.

Lo switch invece è un dispostivo di livello 2 e non ha un comportamento passivo.



Non è broadcast ma point-to-point, non soffre di collisione (al contrario del bridge) in quanto è full duplex (bidirezionale – un cavo in input e uno in output).

Lo switch è un bridge full duplex sul quale risiede la mac table.

Gli switch vengono ancora chiamati CSMA – CD solo perchè cono compatibili con i pacchetti Ethernet.

MAC (Multiple Access Control)

MAC = IEEE 802.3 → Standard schede Ethernet

Composizione di un frame Ethernet (min 64 bytes ~ max 1.518 bytes)

Preamble 7	Start frame delimiter 1	Destination 2 / 6	Source 2 / 6	Lenght 2	Data 0 / 1500	PAD 0 / 46	Checksum 4

Livello 7	
Livello 6	
Livello 5	
Livello 4	SEGMENTO
Livello 3	PACCHETTO
Livello 2	FRAME
Livello 1	BIT

In una rete broadcast viene bufferizzato solo il pacchetto (frame) interessato (l'host verifica che il campo destinazione contenga il proprio indirizzo, quindi verrà messo nel buffer di ricezione).

Ogni produttore ha a disposizione uno slot di 2^{48} MAC address da assegnare alle proprie schede di rete (infatti ogni scheda di rete è identificata univocamente mediante il MAC address che è scritto direttamente nella scheda).

Il campo PAD viene utilizzato per raggiungere la dimensione minima del frame.

Destination + Source + Length + Checksum = 18 + PAD = 64 bytes

Nel calcolo della massima dimensione del frame, il preambolo e lo start frame delimiter non vengono considerati. Inoltre il PAD esiste solo se il payload (data) non raggiunge la dimensione minima.

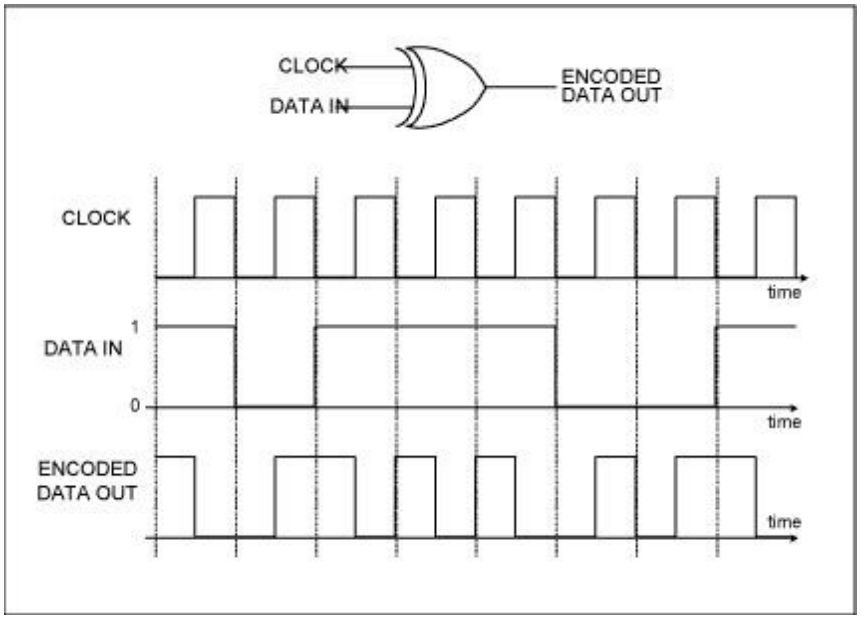
In un broadcast, esiste un indirizzo specifico per l'invio di un frame a tutti gli host appartenenti ad una sottorete che è l'indirizzo 255.255.255.255 (1111 1111.1111 1111.1111 1111.1111 1111).

Gli indirizzi MAC di sorgente e destinazione possono essere anche di 2 bytes perchè è possibile utilizzare Ethernet anche su reti piccole con pochi device (come in un'azienda possono essere connessi in rete alcuni macchinari).

Frame Check Sequence (FSC) campo che contiene il CRC (Cyclic Redundancy Check).

Codifica Manchester

La codifica Manchester è un particolare codifica utilizzata per il trsferimento di dati.



Due computer che devono parlare utilizzano la codifica Manchester che è basata sul clock di ricevente e destinatrio, quindi ogni bit è segnalato da una transizione.

La serie dei dati che viene inviata è preceduta da un prembolo che è composto d 8 bytes di cui i primi sette 10101010 mentre l'ottavo 10101011.

Il preambolo permette al destinatario di sincronizzare una unità apposita che permette la ricezione dei dati.

Invio dei dati: la sequenza viene codificata eseguendo uno XOR bit a bit tra il clock e la sequenza di invio.

Ad ogni salita/discesa del clock (in base alla modalità utilizzata), il bit della sequenza viene codificato in XOR e inviato sul canale.

L'host ricevente, attiverà la propria unità di sincronizzazione sul preambolo e leggerà i valori in ingresso. Tali valori verranno letti sfalsati di mezzo di clock tale da poter leggere il bit in entrata esattamente alla metà del suo segnale (per evitare errori di lettura dovuti a disturbi del segnale sul canale).

La trasmissione funziona in quanto tale codifica è self-clocking, ossia permette una accurata sincronizzazione del flusso dati.

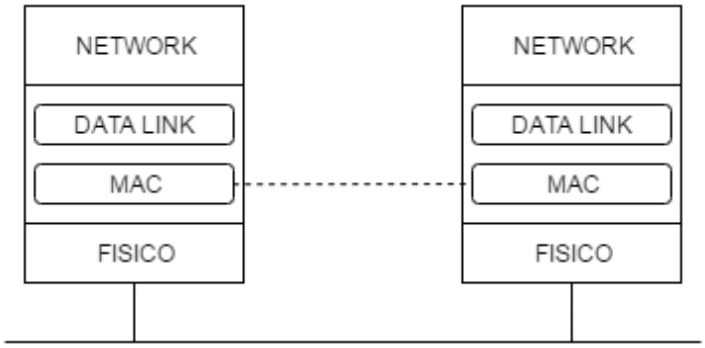
La codifica permette di non perdere sincronismo durante una lunga serie di 0 o 1 e funziona se entrambi utilizzano la stessa frequenza di clock o suoi sottomultipli.

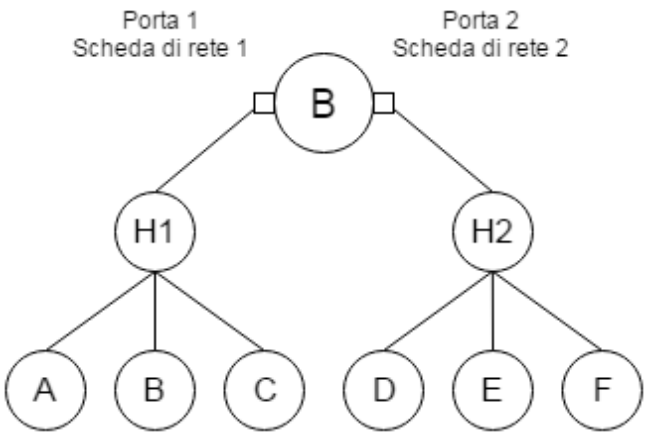
Hub e Bridge

A livello fisico si inserisce la codifica Manchester insieme a clock, cavi, hub, preambolo, input bit…

Hub e switch fanno broadcast.

Qualsiasi dispositivo connesso broadcast è soggetto a collisione.





Bridge: divide il dominio di collisione in 2 parti.

A, B, C non collidono più con D, E, F.

Il bridge ha una tabella di forwarding per conoscere chi è connesso a sé stesso e poter inoltrare correttamente i frame.

Se il traffico sul bridge è diretto da B a C, il bridge non fa nulla e blocca i frame sulla porta 2.

Se un frame sta viaggiando da B a D invece il bridge ripropaga sulla porta numero 2 il frame (l'indirizzo sorgente rimane quello di B).

Inizialmente quando il bridge viene inserito nella rete, esegue funzione di hub in quanto non conosce la topologia della rete e la tabella di forwarding è vuota.

Quando avviene una trasmissione, il bridge scopre chi è la sorgente (MAC) di quel frame e quindi lo inserisce nella tabella in corrispondenza della porta da cui è arrivato il frame iniziando a popolare la TdF.

Il processo di learning è continuo anche quando gli host vengono connessi o scollegati.

Ogni entry della tabella ha un TTL (Time To Live – 300 / 1.500 sec solitamente) breve in modo che il bridge non debba risolvere ogni volta il pacchetto e aggiornare la tabella ma allo stesso tempo non impedisca ad un eventuale host, appena riconnesso su un'altra porta diversa dalla precedente a cui era collegato, di non riceve più frame (a causa della vecchia entry nella TdF).

Volendo il TTL può essere anche infinito (per i server che non si scollegano di solito).

Quindi il bridge di livello 2 fa forwarding senza instradare che è compito del livello 3.

Tabella di forwarding :

MAC	PORT (I/O fisico)
00-08-74-4C-7F-1D	2

Nel bridge avrò perciò una memoria locale in cui verranno salvati i frame per poterli ispezionare e popolare la tabella (Store and Forward).

Switch ◊ bridge che collega canali full-duplex (un canale in input e uno in output).

Gli switch vengono usati per le MAN e sono molto potenti.

Meglio utilizzare uno switch fisico invece che un algoritmo come csma-cd per risolvere le collisioni.

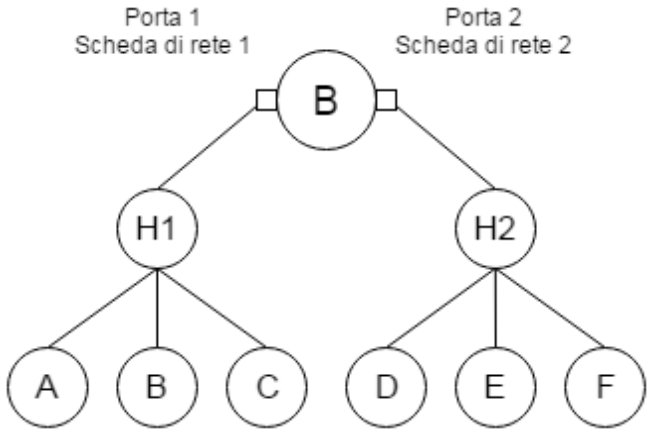
VLAN

Virtual Local Area Network utilizzata per creare degli strati virtuali su una rete fisica.

Ipotizziamo di voler riunire i diversi host in diverse sottoreti virtuali diverse da quelle rappresentate fisicamente.

- A, B, F : rete amministrazione
- C, D : marketing
- E : vendite

Allora utilizzo una v-lan per suddividere tali host nella rete virtuale di appartenenza (sottoreti funzionali).



Posso creare tali v-lan aggiungendo un campo nella forwarding table.

MAC	PORT	V-LAN
00-08-74-4C-7F-1D [C]	1	100
C0-34-68-4C-8A-2B [D]	2	100
….	….	200

Si dice, in gergo, che si assegna un “colore” alla v-lan (in realtà è un numero identificativo come 100, 200 etc...).

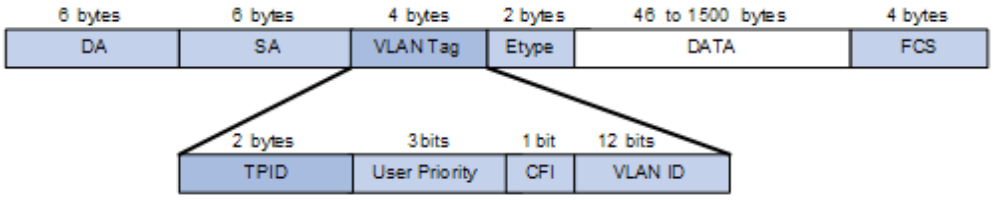
Il formato della frame in relazione alle v-lan dovrà ovviamente cambiare perchè dovrà trasportare anche l'indicazione di quale v-lan fa parte il frame.

In particolare è l'amministratore che decide di “colorare” le porte dello switch. Infatti le porte possono essere di due tipi:

1. Tagged (colorate) redirigono il traffico di una determinata v-lan in base al colore assegnato
2. Untagged (non colorate) redirigono il traffico normalmente

Lo standard 802.3 a cui vengono aggiunte le v-lan si chiama 802.1Q .

Formato del frame:



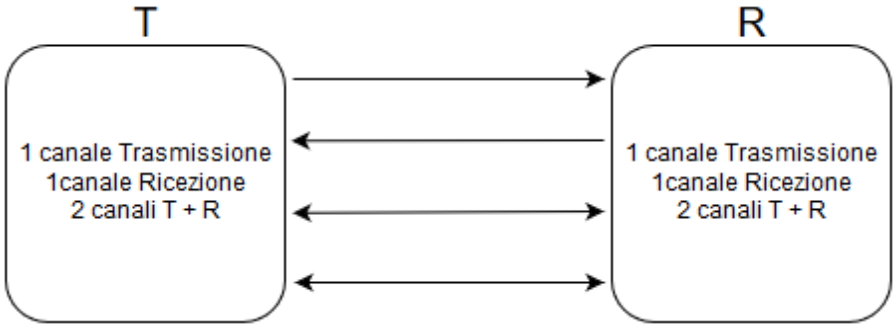
Come fa un host o uno switch a sapere che sta processando un frame tagged? Guardando la lunghezza del frame se è > di 1.500 quello che dovrebbe essere il payload. In particolare viene inserito in 16 bits (2 bytes del campo TPID) il numero 8.100_{16} (33.024_{10}).

Infatti in una frame normale untagged, il campo vlan corrisponderebbe al campo length, perciò nella computazione del frame verrà letto nel campo vlan un valore > 1.500 e quindi si capirà che in realtà quello processato non è untagged ma tagged.

Reti Ethernet

- Ethernet 10MBit/sec
- Fast Ethernet 100MBit/sec
- Giga Ethernet 1GBit/sec

Quanto è più alta la frequenza di clock, tanto più diventa difficile da parte del lettore capire la trasmissione.



Un canale da 100 MBit in rame si è visto che non si riesce a farlo. Infatti utilizzando i 3 cavi, ognuno trasprterebbe 33,3 MB ma non si riesce: il massimo che si raggiunge sono 25 MB.

Si potrebbero utilizzare 4 cavi così da poter portare 25 MB ciascuno ma non esistono in commercio cavi con 5 canali.

Come portare 100 MB su solo 3 cavi quindi?

Si è pensato di rimappare ogni sequenza binaria su una sequenza ternaria (-V , 0, +V).

$$2^8 \rightarrow 3^6 \ \text{[8B6T]}$$

Questa codifica da 8 bit binaria si sposta perciò ad una codifica ternaria su 6 bit. In questo modo dovendo trasportare solo 6 bit invece che 8, la comunicazione risulta più veloce e si raggiungono i 100 MB.

Risulta necessario però che lato trasmissione e ricezione venga fatto questo lavoro di traduzione da codifica binaria a ternaria mediante delle apposite tabelle di traduzione.

Inoltre la tabella è stata pensata in modo tale da ridurre qualsiasi possibile errore di lettura del canale.

$$((100 \text{ bit} \times 10^6) \times 6/8) / 3 = 25 \times 10^6 \ \text{Bps}$$

25 Mbit/sec su codifica ternaria su singolo cavo → 33,3 Mbit/sec su codifica binaria

Sul 10 MB si usavano solo 2 cavi. Su codifica ternaria la frequenza è più bassa in modo ch il cavo riesca a trasmettere correttamente l'informazione in modo che all'inzio e alla fine del cavo ci sia una corretta traduzione.

Lo switch essendo full-duplex, ogni host collegato ad esso, ha una porta di input e una di output per evitare le collisioni.

Al centro dello switch c'è una memoria condivisa e può funzionare in diverse modalità:

1. cut-through (lettura del MAC e forward del frame durante l'invio. Non attenda la ricezione completa)
2. store-and-forward (attesa di ricezione di tutto il frame. Check del codice di correzione e inoltre)
3. fragment-free (combinazione dei due precedenti)

Se la memoria va in overflow, ci sono determinate implementazioni di protocolli per il controllo del flusso che permettono di rallentare l'invio dei dati ed evitare il drop dei frame.

Nell'invio di una frame, il sender invia ad una frequenza di 10 MB, ciò significa che in un determinato lasso di tempo T, riuscirà a recapitare K bit. Se però aumentiamo la frequenza di invio a 1 GB (gigabit Ethernet) , allora nello stesso lasso di tempo T, la quantità di bit che riesco a recapitare sarà > K.

Posso decidere perciò di aumentare la dimensione della frame per poter recapitare questo maggior numero di bit. Tuttavia la soluzione migliore non è quella di aumentare la dimensione della frame ma di diminuire la lunghezza dei cavi (200 mt standard) mantenendo la dimensione sempre di 512 bit (64 byte).

Da standard il massimo possibile è stato messo a 800 mt suddiviso in 4 porzioni da 200 mt collegati da 4 repeater.

$T_p = 8 * 10^2 / (2 * 10^8) = 4 * 10^{-6} = 4\ \mu s$

Il tempo perciò per riconoscere una collisione diventa

$$4\ \mu s * 10^9 = 4000 bit \rightarrow 500\ byte \rightarrow default\ 512\ byte = 64\ kb$$

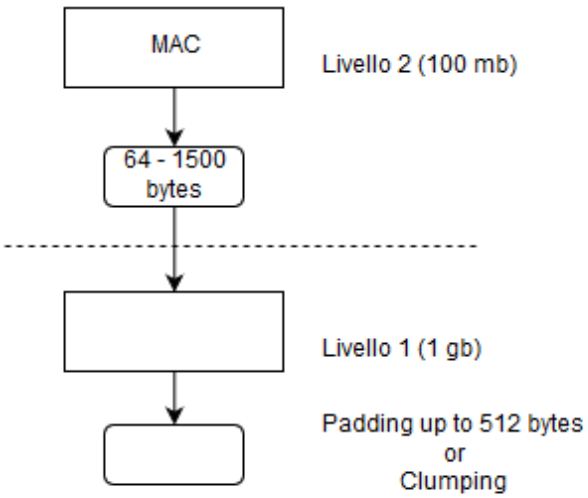
Posso avere un livello 1 a gigabit e un livello 2 a fast ethernet? Come è compatibile con csma-cd?

Viene paddato il payload fino a raggiungere la dimensione minima.

Il fatto di avere una connessione GBit a livello 1 e mantenere la 100 MBit a livello 2 ha comunque un beneficio perchè permette una migliore risoluzione delle collisioni.

Clumping: non faccio padding ma aspetto dal livello 2 che venga riempito tutto il frame di 512 bytes. Quale diventa il beneficio? Il livello 1 quanto deve aspettare?

E se i frames non arrivano? Quanto tempo bisogna aspettare?



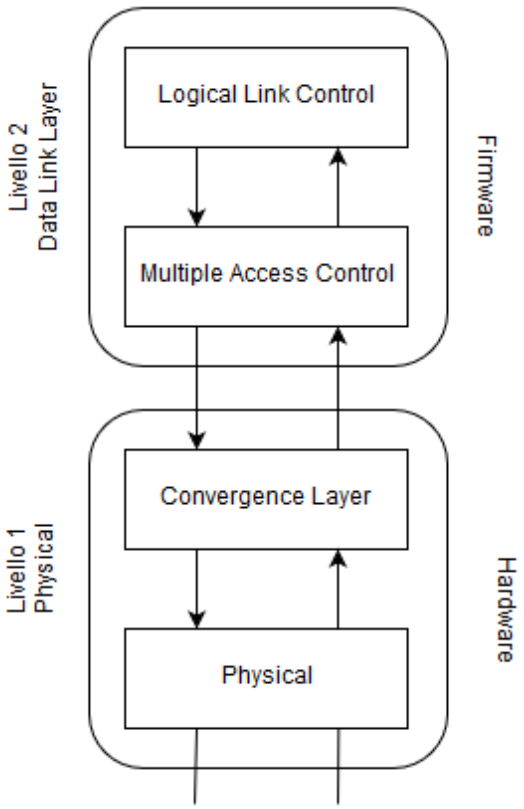
Livello Data Link

Il livello di collegamento (data link) è suddiviso in due sottolivelli.

Logical Link Control :

- Multiplexing / Demultiplexing
- Logical Link Service

Il sottolivello superiore è Logical link control (LLC), e può fornire servizi di controllo di flusso, conferma, rilevazione (o correzione) degli errori. I protocolli PPP e HDLC fanno parte di questo sottolivello.



MAC :

- Framing / Deframing
- Interact with level 1
- Collision resolution

Il sottolivello inferiore è Media Access Control o Medium Access Control. Il suo scopo è quello di disciplinare l'accesso multiplo di molteplici nodi ad un canale di comunicazione condiviso evitando o gestendo l'occorrenza di collisioni. Una collisione si verifica quando due o più nodi trasmettono simultaneamente dati sul canale condiviso. Ciò comporta l'inevitabile perdita dei dati trasmessi con conseguente spreco di banda.

Esistono molteplici algoritmi e protocolli standard per il controllo dell'accesso multiplo. Ad esempio, il MAC IEEE 802.3 adotta l'algoritmo CSMA/CD mentre il MAC IEEE 802.11 si basa sull'algoritmo CSMA/CA. Il primo è comunemente adottato in LAN cablate, il secondo in WLAN.

Due sono le principali tipologie di algoritmi di accesso multiplo: casuale e ordinato. Nell'accesso multiplo casuale è possibile che si verifichino delle collisioni ma vengono implementati degli opportuni meccanismi per ridurne la probabilità di occorrenza e per ritrasmettere le trame collise. Nell'accesso ordinato, invece, l'evenienza di una collisione è del tutto impossibile poiché i nodi seguono un preciso ordine di accesso al canale (stabilito nella fase di inizializzazione della rete) che li rende utilizzatori esclusivi del mezzo trasmissivo (a meno di guasti o malfunzionamenti).

A livello MAC, inoltre, si definisce il formato della trama, che tipicamente conterrà i campi di inizio/fine, i campi di indirizzo MAC mittente/destinatario, il pacchetto incapsulato di livello LLC, il codice per la rilevazione degli errori (FEC), ed opzionalmente dei byte di padding per garantire che la dimensione della trama non scenda al di sotto di una soglia minima.

LLC utilizzabile in due configurazioni:

- best-effort
- affidabile

Teoricamente su una rete broadcast servirà un MAC mentre su rete punto - punto non serve (n punti di livello 2 in base a quanti host sono connessi) .

Frame generica di livello 2:

Flag	Header	Data	Flag

Il flag contiene la sequenza 0111 1110 per identificare inizio e fine della frame.

Se il campo information (data) contiene questa sequenza? Vieni interpretato male il pacchetto?

Il flag funziona come il preambolo. Come risolvere il problema?

Si inserisce uno zero nel payload dopo cinque “1” sempre per distinguere una eventuale sequenza uguale al flag ma all'interno del campo dati. Quando il ricevente leggerà 6 volte un 1 vorrà dire che sono il flag di apertura e chiusura; ogni volta che il ricevente leggerà cinque 1 e uno zero, tale zero verrà rimosso.

Nota bene: lo zero dopo i cinque 1 viene aggiunto comunque anche se il sesto bit è uno zero (in tal caso la sequenza 1111100 diventerà 111110).

Questa tecnica si chiama “Bit Stuffing”.

In Ethernet non si esegue bit stuffing.

PPP

Formato frame PPP (point–to–point protocol)

Flag	Address	Control	Protocol	Information	FCS	Flag

Formato della trama PPP

Address: indirizzo della stazione trasmittente / ricevente in base al tipo di comunicazione.

Control: tipo di trama utilizzata durante la trasmissione.

Address + Control + Protocol = Header

Protocollo di livello 2 a connessione punto – punto utilizzato principalmente nelle due varianti

1. PPPoE point-to-point protocol over Ethernet
2. PPPoA point-to-point protocol over ATM (Asynchronous Transfer Mode)

dagli ISP (Internet Service Provider) per stabilire una DSL (Digital Subscriber Line) nella connessione coi clienti.

Possibili configurazioni del protocollo PPP:

1. Autenticazione: scambio di messaggi mediante protocollo di autenticazione con password
2. Compressione: riduce la quantità di dati trasferiti tra un peer e l'altro
3. Error detection
4. Multilink: bilanciamento del carico con l'utilizzo di diverse interfacce

Protocolli

Un protocollo si dice affidabile quando è corretto, non duplicato e i frame arrivano in ordine.

Quando un frame viene inviato è possibile che l'ack non arrivi o arrivi fuori tempo (del timer) o il frame venga perso.

Allora la frame viene tenuta nel buffer di memoria del mittente e rinviata.

Tuttavia è possibile in certi casi che il mittente invii due volte tale pacchetto (se non dovesse arrivare l'ack, il mittente non sa se il frame è comunque stato ricevuto o meno).

Quando il duplicato arriva al destinatario, e viene passato al livello 3, viene letto il numero di sequenza e viene scartato.

Esiste una variabile di ricezione (destinatario) che permette di indicare univocamente qual è il numero di sequenza del pacchetto in attesa che crescerà in maniera incrementale.

Esisterà anche una variabile di trasmissione del mittente che verrà invece incrementata quando arriverà l'ack dal destinatario.

Date queste caratteristiche si assume che l'invio dei frame sia in ordine, infatti le variabili non funzionano nel caso l'invio dei frame non sia ordinato.

Si può rendere più semplice il protocollo utilizzando una convenzione sulla variabile:

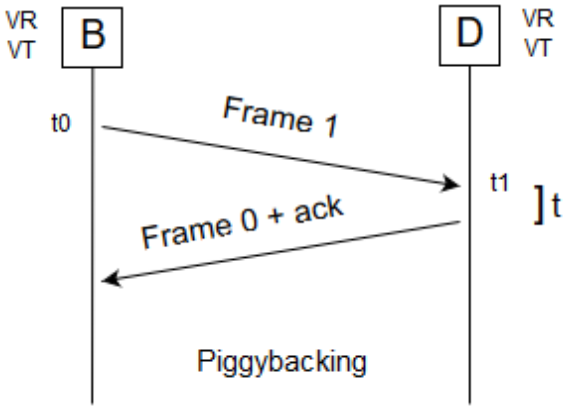
- 1 : se il frame ricevuto è un duplicato
- 0 : richiesta di un nuovo frame

In questo tipo di comunicazione (se bilaterale) l'ack viene messo a bordo di un frame in risposta al mittente in modo da poter comunicare e ricevere contemporaneamente.

Frame:

Destination address	Source address	Ack	Sequence #	Data

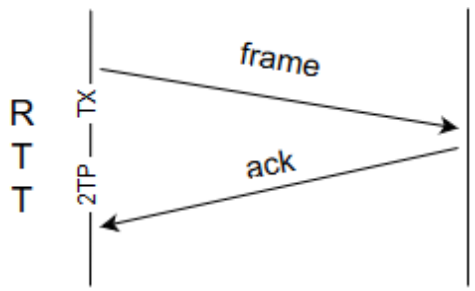
Se D alla ricezione di f1 ha pronto, entro un tempo t piccolo, il frame 0 da inviare, allora gli aggancia sopra l'ack di f1 e ne approfitta per inviare entrambi a minor costo.



Questa tecnica si chiama “piggybacking”.

Il protocollo che attende l'ack per ogni frame non è efficiente perché ogni volta che viene inviato un frame il mittente deve attendere la risposta di controllo dal ricevente prima di inviare il frame successivo.

Esempio:



$RTT = t_x + 2\ t_p$

f = 1.000 bit

$ct_x = 10^6$ bps → 1 mb

cavo = 2 km

$t_x = (10^3 / 10^6) = 10^{-3} \rightarrow 1$ ms

RTT = 1ms + 20 µs

$t_p = (2 \times 10^3) / (2 \times 10^8) = 10^{-5} \rightarrow 10$ µs

In questo caso è più grande il valore di t_x rispetto a $2t_p$, per tutto t_x sto trasmettendo e non posso inviare altre frame nel frattempo → efficienza alta.

Se $2t_p$ fosse più ampio, non potrei trasmettere altre frame, "sprecando" più tempo.

Tuttavia è possibile ottimizzare l'uso del driver si trasmissione inviando frames anche durante il tempo $2t_p$.

Calcolo della misura di utilizzo del canale Ux.

$$Ux = t_x / RTT = t_x / t_x + 2t_p$$

Più l'utilizzo è vicino al 100%, maggiore è l'efficienza.

Se il valore di 2tp è basso, il rapporto Ux tenderà ad 1 → pieno utilizzo del canale.

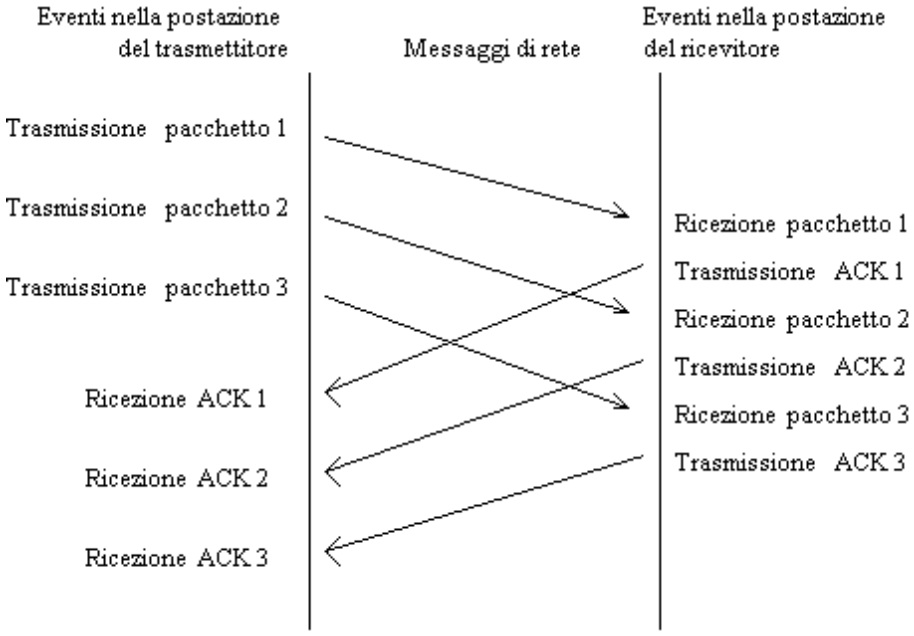
Se tp dovesse diventare dominante, l'utilizzo del canale e quindi la sua efficienza aumenterebbe (infatti solo inviando più frames è possibile aumentare l'efficienza sfruttando quasi al 100% il canale).

$$U = 1ms / 1ms + 2ms = 1/3$$

Come creare un protocollo che sfrutti questa idea?

Utilizzando i "protocolli a finestra".

Protocolli a finestra



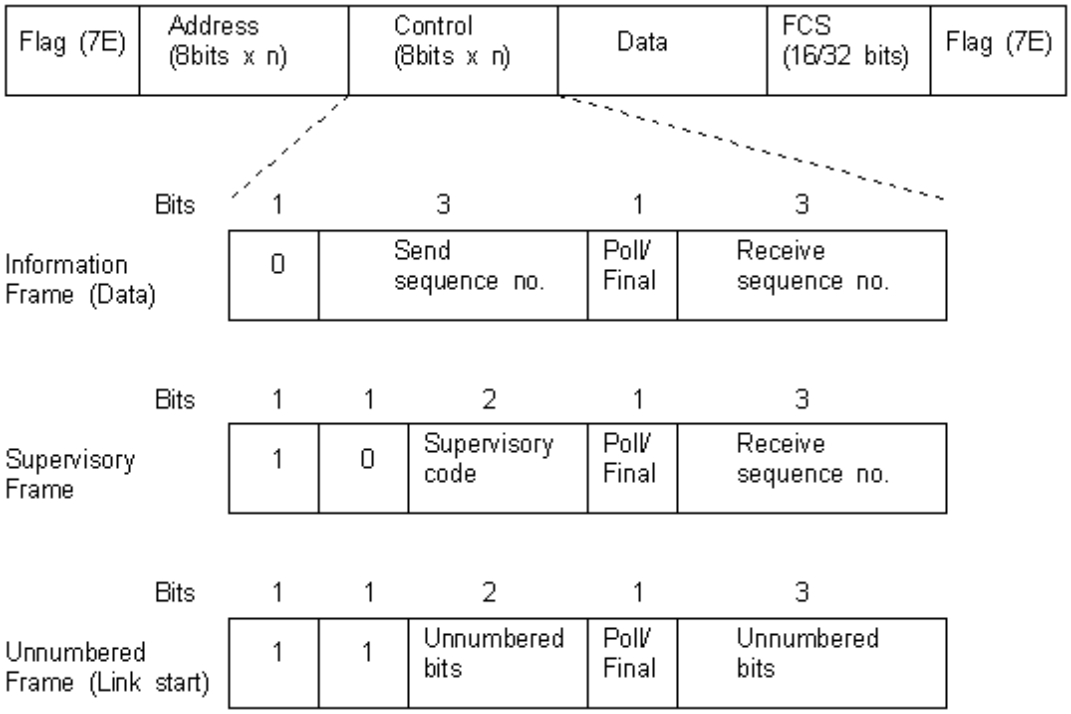
Nella figura (relativo al frame/pacchetto 1):

- t_x è il tempo di trasmissione del pacchetto 1

- $2t_p$ è il tempo dalla trasmissione del pacchetto 2 alla ricezione dell'ack del pacchetto 1

TCP è un protocollo a finestra.

Se per esempio aumentassi la lunghezza del cavo, il tempo di propagazione aumenterebbe consentendomi di inviare altri frame nel tempo $2t_p$.



I protocolli a timer visti sono considerabili protocolli a finestra di 1 frame.

I frame di questo tipo sono chiamati HDLC (High level Data Link Control) e utilizzano il bit stuffing, piggyback etc...

PPP è una derivazione di HDLC.

In un protocollo a finestra i frames vengono tenuti in memoria dal mittente fintanto che non viene ricevuto l'ack. A quel punto il buffer viene svuotato e viene incrementata la variabile di trasmissione e resettato il timer se tutto funziona correttamente.

Se ci sono errori?

1. Un frame può arrivare prima di uno con numero di sequenza successivo
2. Un frame può arrivare dopo la scadenza del timer
3. Un frame può non arrivare

Esistono due soluzioni:

1. GoBack_N : drop di quelli sbagliati da parte del ricevente e attesa di quello corretto.
2. Selective Repeat : aggiunta in un buffer temporaneo i frames fuori sequenza per poter essere riordinati successivamente in attesa di quello corretto

Queste modalità funzionano a livello 2 ma anche a livello 4.

Inoltre l'ack può essere utilizzato in due modalità:

1. Selettivo : è stato ricevuto l'ack relativo ad un determinato frame n

2. Cumulativo : sono stati ricevuti tutti i frame correttamente, precedenti a quello con numero di sequenza n

Esempio:

- Selective ack 4 → ricevuto correttamente il frame 4
- Cumulative ack 4 → ricevuti correttamente i frame 0, 1, 2, 3, 4

Esiste un pacchetto di controllo selettivo chiamato NACK ossia “Non - ACK” che identifica il frame che non è stato ricevuto correttamente.

I protocolli in cui viene inviato un frame alla volta, viene verificato e inviato l'ack, si chiamano Idle-RQ (chiamati anche “Stop & Wait”).

I protocolli a finestra in cui vengono inviati più frames alla volta si chiamano Continuous-RQ.

GoBack_N

GoBack_N : La ricezione di frames fuori sequenza (non in ordine o fuori tempo di timer) non è contemplata e tale frames vengono droppati dal ricevente, altrimenti invia un ack cumulativo di avvenuta ricezione corretta.

Esempio:

N+2 arriva prima di N+1 → N+2 droppato e mandato cumulative ACK N → resto in attesa di N+1

Perché inviare le frame di controllo?

Il tutto funzionerebbe anche se non venissero inviati gli ack e si aspettasse lo scadere del timer per il rinvio dei frame errati ma questo comporterebbe una perdita di tempo nella comunicazione.

Durante la comunicazione si potrebbe perdere non solo il frame dei dati ma anche il frame ack.

Tuttavia in uno scenario di comunicazione con ack cumulativi, quando viene inviato un nuovo ack relativo ad un nuovo pacchetto inviato, il ricevente sa indirettamente che il frame relativo all'ack non ricevuto, è arrivato correttamente e quindi svuoterà il buffer dai frame già inviati.

Un protocollo Goback_N tuttavia droppa anche i pacchetti corretti se fuori sequenza.

Come risolvere tale problema? Con un protocollo Selective Repeat.

Il numero di sequenze utili per identificare tutti i frame in una finestra di k frame, sono k+1 → variabile di ricezione da 0 a k (cioè k + 1).

Selective Repeat

Tutti i frame corretti arrivati al ricevente, vengono salvati in un buffer temporaneo che resta in attesa della ricezione del pacchetto desiderato per così ricostruire l'informazione in modo ordinato. Mittente e ricevente possono lavorare in due modalità una volta arrivato il frame a destinazione:

- Ack cumulativo : attende che siano arrivato il frame desiderato in sequenza per poi inviare un ack cumulativo fino a tutti i pacchetti già arrivati correttamente e precedentemente salvati nel buffer temporaneo.
- Nack selettivo / non invio (ricezione) dell'ack : il mittente ritrasmette il frame desiderato del ricevente.

La variabile di ricezione del destinatario rimane fissa sul numero dell'ultimo frame arrivato correttamente e la richiesta continua a rimanere quella del frame successivo (desiderato) all'ultimo ricevuto correttamente.

Il comportamento perciò varia in base a quale tipo di comportamento assumono M e D.

Se si perde un frame? Viene rinviato e se è un doppione verrà droppato da D il quale rimanderà l'ack a M, altrimenti verrà bufferizzato e verrà inviato l'ack.

Il numero di sequenze utili per identificare tutti i frame in una finestra di k frame, sono 2k → variabile di ricezione da 0 a 2k-1 (ossia 2k).

Conclusioni livello 2

L'indirizzamento è svolto a livello 2 e da tutti i livelli superiori in maniera indiretta:

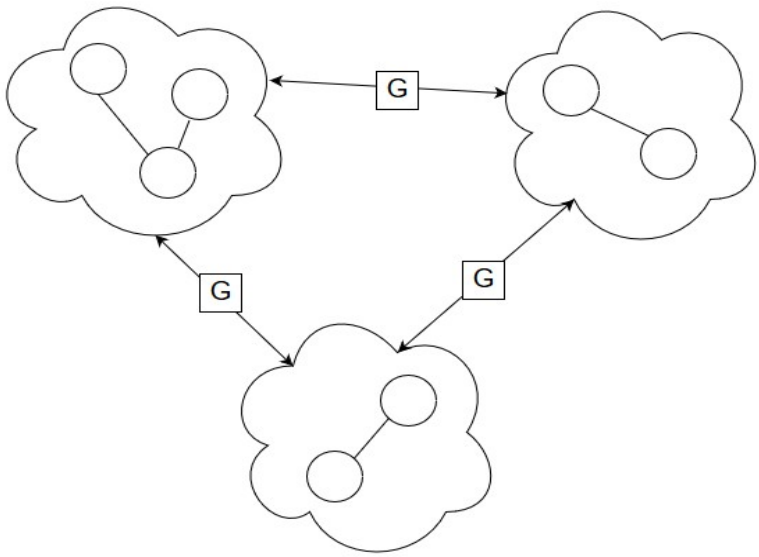
- MAC → identificazione del mezzo con cui si vuole comunicare
- Livello 3 → scoprire la topologia della rete ed i percorsi esistenti

Indirizzamento sul livello 2 point-to-point non avrebbe senso, pertanto la frame non contiene indirizzo source e destination.

Livello 3 (Network)

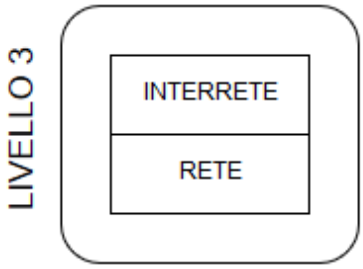
Il lavoro di instradamento caratterizzante del livello 3 è applicabile su due diversi livelli:

- interno ad una rete LAN
- esterno tra più LAN



Le diverse LAN sono connesse mediante gateway (qualsiasi tipo di dispositivo di rete come routers, switch...).

Gateway è un termine generico per indicare il device che permette di veicolare pacchetti all'esterno di una rete LAN.



IP è una funzione di indirizzamento che lavora a livello 3 alla quale dovrà essere aggiunta la funzione di routing R.

Gerarchia di Internet:

- Livello 1 → international ISP
- Livello 2 → ISP locali
- Livello 3 (accesso) → tutte le LAN Ethernet etc... locali

Protocolli di livello 3 : IP (coordina tutti gli altri moduli), ICMP, ARP, DHCP, OSPF, RIP …

IPv4: 32 bit di indirizzo

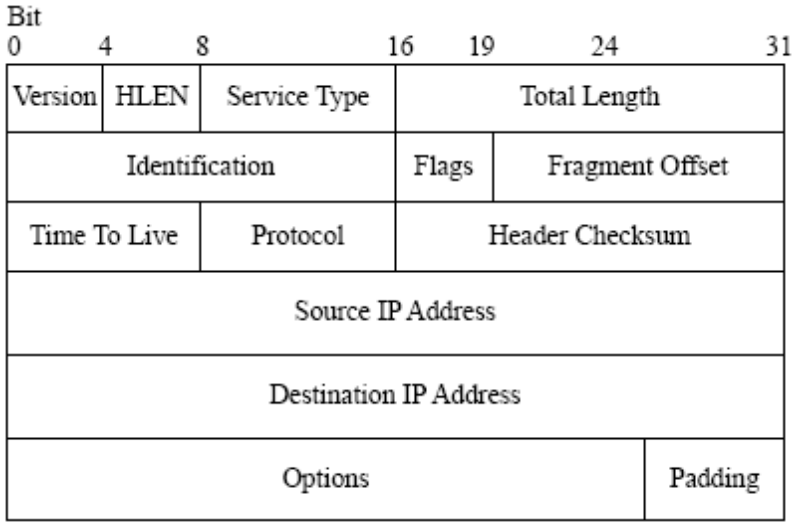
IPv6: 128 bit di indirizzo

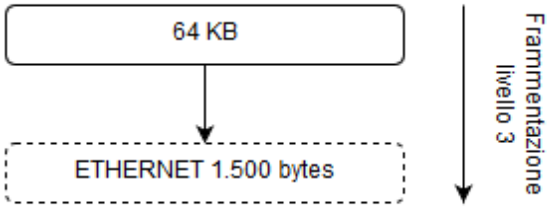
IP versione 6 assicura un indirizzo unico ad ogni device connesso in rete.

Header senza Options (Opzionale) è lungo 20 bytes.

La massima dimensione di un pacchetto è 64 kb.

Il campo service type viene usato per definire il tipo di servizio relativo ad un determinato pacchetto (email, multimedia...) e fare la gestione delle code in modo più efficiente.



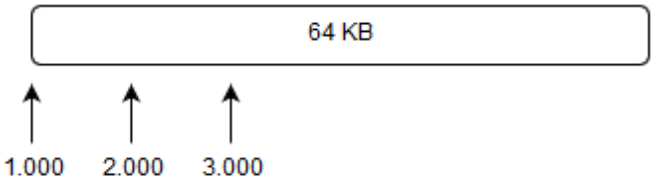


TTL : i pacchetti scaduti vengono droppati quando passano da un router

HCRC : header correction code (checksum)

Il protocol selector permette per esempio di definire quale protocollo si sta utilizzando tra TCP e UDP a livello 4.

Infatti a livello 4 c'è un numero che identifica i processi: tale numero viene inserito nel Protocol Selector affinché il destinatario sappia come interpretare i pacchetti in base a numeri di selettori standard.



Fragment offset : offset del pacchetto nel pacchetto originale

Essendo la lunghezza del campo Fragment Offset uguale a 13, è possibile indirizzare fino a 2^{13} blocchi di byte (per questo il pacchetto

deve essere suddiviso in blocchi di da 8 byte in modo da poterlo indirizzare tutto con 13 bit, altrimenti ne servirebbero di più di bit per indirizzare ogni byte dell'intero pacchetto). $8 * 2^{13} = 65.536$ (RFC 791)

Il campo opzionale può essere usato per diversi scopi per esempio per effettuare source routing ossia inserire l'elenco dei routers, dei nodi da cui deve transitare un pacchetto (quale strada voglio far seguire al pacchetto per arrivare a D).

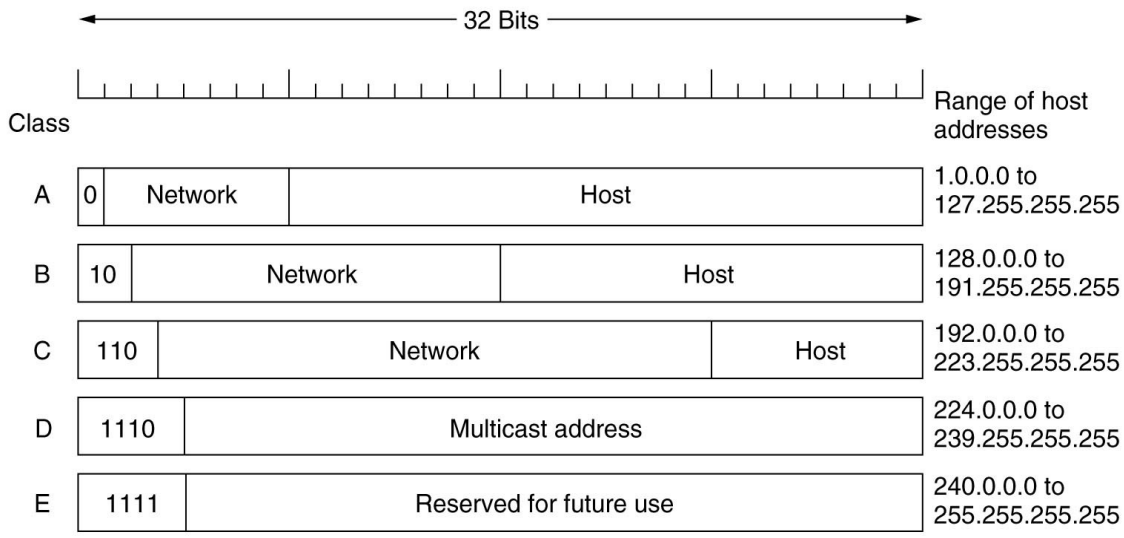
Nel campo flag viene inserito il more fragment:

- 1 → D deve attendere altri pacchetti dopo quello appena computato
- 0 → la trasmissione è finita. Quello indicato con 0 è l'ultimo pacchetto della comunicazione.

Se a livello 2 ho Ethernet, è necessario frammentare i pacchetti in blocchi da 1.480 bytes (1.480 + 20 di header = 1.500 bytes dimensione massima).

Indirizzi di rete

Gli indirizzi IP sono composti come : < Network ID , Host ID >



I gateway "spacchettano" i pacchetti che gli arrivano ma non li ricompongono.

Le classi A e B soffrono di una grande frammentazione interna.

CIDR

Classless InterDomain Routing

L'utilizzo delle classi sugli indirizzi IP ha portato ad avere una grande frammentazione (infatti in una rete di classe B non è detto che tutti gli indirizzi host vengano utilizzati, lasciando dei buchi inutilizzati).

Data la scarsità di indirizzi IP si era pensato di suddividere a blocchi di 256 indirizzi le classi A e B (2^8 indirizzi per blocco) colmando così i vuoti e creando tante classi C che altrimenti sarebbero andate perse.

Ipv6 nasce con l'avvento dell' IoT (Internet of Things).

Esempio:

Milano 2.048 indirizzi da 190.24.0.0

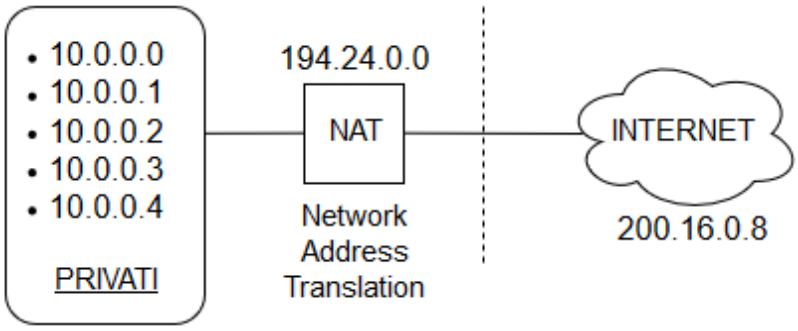
- 190.24.0.255
- 190.24.1.255
- 190.24.2.255
- ……..
- 190.24.7.255 (256 * 8 = 2.048)

Roma 4.096 indirizzi → 190.24.16.0 ~ 190.24.31.255

Torino 1.024 indirizzi → 190.24.8.0 ~ 190.24.11.255

Quando un indirizzo entra in un router di Milano, a questo viene applicato l'indirizzo di maschera di Milano in AND logico che restituisce un indirizzo IP: tale indirizzo viene poi confrontato con quello base di Milano (net-id di Milano) e se è uguale significa che l'host destinatario è interno alla rete cittadina, altrimenti è esterno bisognerà provare la maschera di Roma e poi di Torino.

NAT



Network Address Translation permette la traduzione da indirizzo pubblico (unico del NAT) a privato e viceversa.

Come fa un NAT a reindirizzare correttamente un pacchetto?

Utilizza una tabella interna di conversione che interroga ad ogni transito di pacchetto.

IPSource	IPDestination	IPNATSource
10.0.0.0	200.16.0.8	194.24.0.0
10.0.0.3	200.16.0.8	194.24.0.0

Quando NAT riceve il pacchetto di risposta dalla rete,come IPD avrà il proprio indirizzo in quanto l'host remoto non conosce gli IP interni e quindi utilizzerà quello pubblico del NAT.

Per sapere a chi inviare il pacchetto correttamente, il NAT utilizza il concetto di “PORT”. Ad ogni processo che usa TCP, il sistema operativo assegna un numero random ossia la porta da utilizzare per la comunicazione.

Quando avviene una comunicazione tra un host in LAN e uno remoto, le comunicazioni possono avvenire anche mediante porte ben stabilite (80 per http, 21 per ftp….) dette “well-known” o “porte standard”, altrimenti la comunicazione delle porte su cui gira un determinato servizio deve avvenire preventivamente.

Ipotizzando di avere un web server ad un determinato indirizzo IP, sulla porta 80 ci sarà un solo web server attivo che permette la gestione delle richieste web.

Tuttavia è possibile che due sistemi operativi, non comunicando tra loro, abbiano assegnato lo stesso numero di porta a due processi evidentemente differenti su macchine perciò diverse.

Il NAT aggira la problematica assegnando un proprio numero di porta ai processi che vi transitano.

Infatti il NAT, a dispetto degli altri dispositivi di rete, è in grado di svolgere funzioni su diversi livelli (è un device non convenzionale: funziona ma è stato studiato apposta per tale problematica) e legge i pacchetti e prende decisioni.

La tabella NAT viene integrata quindi così:

IPS	IPD	IPNATS	PORTS	PORTD	NATPORT
10.0.0.1	200.16.0.8	194.24.0.0	2037	80	1500
10.0.0.3	200.16.0.8	194.24.0.0	2037	80	1501

Le porte vengono usate come identificativo al posto dell'indirizzo MAC in quanto nel pacchetto che dal livello 3 scende al livello 2, non è presente l'indirizzo MAC che viene per l'appunto aggiunto a livello 2. Il NAT e gli apparati a livello 3, agendo su IP non hanno a disposizione tale informazione.

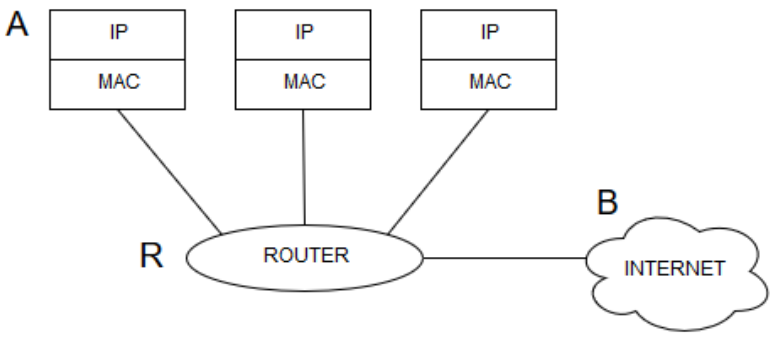
Il NAT crea una DMZ (DEmilitarized Zone). Come fa il PING a funzionare dietro ad un NAT?

PING è di livello 3 ma solo a livello 4 vengono aggiunte le porte. Come fa quindi? A bordo di ICMP

Internet Control Message Protocol che è un protocollo di gestione dei pacchetti di controllo per malfunzionamenti, viene inserito un campo QUERY-ID (numero random) che identifica la richiesta oltre all'utilizzo di un QUERY-ID NAT

ARP

ARP (Address Resolution Protocol) è un protocollo di livello 3 la cui funzione è quella di fornire una associazione (traduzione) di indirizzi IP – MAC e MAC – IP.



Il router contiene una tabella di risoluzione (cache) che viene popolata e modificata a runtime durante il routing dei pacchetti.

Pacchetto ARP

HW TYPE		Ethernet
PROTOCOL TYPE		IP
HLEN	PLEN	HW ADDRESS LENGTH (MAC) IP ADDRESS LENGTH
OPERATION		1 ARP REQUEST, 2 ARP RESPONSE, 3, 4
SENDER HW ADDR		
SENDER IP ADDR		
DEST IP ADDR		
DEST IP ADDR		

Il pacchetto ARP viene inserito nell'header di un pacchetto IP (che è di 20 bytes).

Il pacchetto ARP viene quindi inviato broadcast e solo la macchina che corrisponde a tale indirizzo IP richiesto, risponderà con l'indirizzo MAC (funziona anche con architetture non Ethernet).

Il router fa anche da proxy arp in questo caso, infatti ogni volta che A vuole parlare con B, manda una richiesta arp broadcast ed R risponde ad A con un arp contenente il proprio indirizzo MAC e non quello reale di B.

In questo modo A inserirà come indirizzo destinatario l'indirizzo IP di B ma userà come gateway il MAC di R).

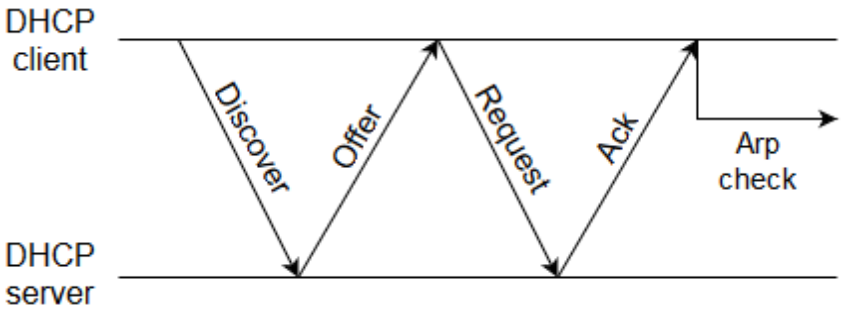
Il pacchetto ARP viene incapsulato all'interno del payload di un pacchetto IP.

DHCP

Il Dynamic Host Configuration Protocol è un protocollo di gestione automatico degli indirizzi IP.

Tale protocollo viene installato su un server che permette la richiesta e l'assegnamento di un indirizzo IP ad un host.

Standard DHCP → RFC 2131



- Discover: inizialmente l'indirizzo sorgente della richiesta è 0.0.0.0 poiché il nuovo device che si connette alla rete non è identificato con nessun IP mentre il destination è broadcast 255.255.255.255. Contiene Transaction-ID e Mac
- Offer: l'host riceve tante offerte di indirizzi IP quanti sono i server DHCP nella rete locale. L'host ne selezionerà uno solo. Il pacchetto contiene transaction-ID, IP address offerto, mask, times.
- Request: l'host ha scelto l'indirizzo IP da utilizzare e lo comunica broadcast a tutti. I server dhcp non selezionati rilasciano il lock attivo sull' indirizzo IP offerto da ognuno rendendoli nuovamente disponibili; il

server dhcp selezionato lo aggiungerà come indirizzo IP occupato. I dati contenuti nel pacchetto sono transaction-id, ip address offerto, mask e times.

- Ack: la richiesta di occupare un indirizzo IP è stata ricevuta correttamente.

DHCP deve assicurare che non vengano usati due indirizzi IP nella stessa rete.

Se al client non arrivano proposte di IP entro un determinato timer, viene rimandata una nuova richiesta DHCP con un nuovo t-id.

Alla fine viene eseguita una arp check: il client chiede al server di risolvere il proprio indirizzo IP e se non ottiene risposta significa che l'indirizzo IP a cui è stato associato è unico nella rete, altrimenti se ottiene risposta con un MAC, significa che ad un altro host nella sottorete è stato assegnato lo stesso IP.

Potrebbero esserci degli errori come il fallimento di arp-check (seguito dalla “decline” dell'IP) da parte del client o il ripensamento da parte del server dell'offerta dell'IP (non viene inviato l'ack).

Nella DECLINE non esiste un timer ma il client continua ad inviare il messaggio di decline sperando che almeno un burst (un pacchetto) arrivi al server.

Routing

L'operazione di instradamento o di routing è di livello 3 ed è compiuta dai routers.

Un router ha il compito di decidere qual è la strada migliore da intraprendere per un pacchetto.

Ogni router è una macchina che fa forwarding.

Quali sono le variabili in gioco nella determinazione del percorso migliore?

- tempo
- numero di hop
- affidabilità

La tabella di routing indica al pacchetto in transito presso un router, quale porta di I/O migliore usare per la trasmissione. La tabella non connette MAC ma solo indirizzi IP.

Il cammino minimo dovrebbe essere uguale su tutti i nodi della rete (altrimenti potrebbero crearsi loop etc...): per questo motivo le tabelle di routing dovrebbero convergere, deve esserci una conoscenza condivisa.

Inoltre la conoscenza condivisa permette l'invio dei pacchetti alla destinazione entro il loro TTL.

Tuttavia in base alle architetture di rete ed in base quindi agli algoritmi di routing utilizzati dai vari routers, la strada potrebbe non convergere o non essere la stessa.

Si creano così problemi di consistenza sul cammino.

Algoritmi di routing:

- distance vector
- link state
- ….

Static routing

Con l'algoritmo di static routing, durante la connessione di un router alla rete, viene immessa nella tabella di routing una entry per ogni host connesso alla rete. In questo modo il router sa dove indirizzare ogni pacchetto da lui in transito.

E' un metodo che non funziona per la rete internet in quanto si modifica costantemente. La rete è dinamica e non può essere utilizzato un algoritmo statico.

Flooding

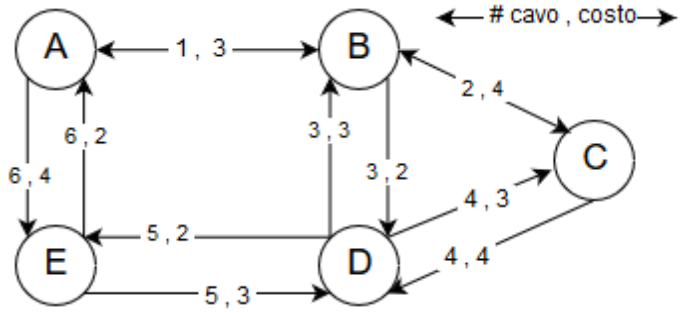
In questa modalità, ogni pacchetto in transito presso un router, viene inviato su tutti i link connessi a tale router. In questo modo il pacchetto sicuramente la destinazione nel minor tempo disponibile perché il pacchetto intraprende tutte le strade possibili.

Questa modalità è dinamica e si adatta bene alla dinamicità di internet.

Tuttavia questo algoritmo nonostante assicuri la buona riuscita dell'invio anche a fronte di una connessione compromessa, ha lo svantaggio di creare innumerevoli copie dello stesso pacchetto inondando la rete e rendendola poco scalabile.

Questo algoritmo viene utilizzato in una prima fase quando i routers devono determinare la topologia della rete a cui sono stati connessi.

Distance Vector e Path Vector



Ogni nodo misura la distanza (utilizzando ad esempio la metrica del tempo...) dal suo vicino.

I tempi sono misurabili (ping etc...).

Ogni nodo conosce la distanza con i propri nodi adiacenti.

Più frequentemente avviene la misurazione dei tempi, più precisa è la conoscenza della rete ma viene occupata più banda e toglie tempo alla trasmissione: per questo bisogna trovare un trade-off (infatti i tempi di trasmissione tra due hosts potrebbero cambiare in base a diversi fattori come la congestione della rete etc...).

Ogni sequenza di misura è indipendente (asincrona) dagli altri aggiornamenti degli hosts adiacenti.

Tabelle delle adiacenze

A :	B	3	1
	E	4	6

B :	A	3	1
	C	4	2
	D	2	3

E :	A	2	6
	D	3	5

Inizialmente ogni nodo conosce solo i propri vicini. Successivamente queste informazioni devono essere propagate anche agli altri nodi. Si potrebbe inserire tali informazioni in un pacchetto ed inviarlo broadcast.

In questo caso però si creerebbe molto traffico. Come fa perciò C a conoscere che esiste A?

B manda le informazioni che conosce non broadcast ma solo ai suoi nodi adiacenti, in questo modo C conosce l'esistenza di A che a sua volta invierà le sue informazioni ai suoi nodi adiacenti e così via...

In particolare il pacchetto che viene inviato è così composto (distance vector – vettore delle distanze):

DV_A:	B	3	DV_B :	A	3	DV_E :	A	2
	E	4		C	4		D	3

In questo pacchetto viene omesso il link in quanto è informazione che non serve e sarebbe solo di peso nella comunicazione.

Storm of distance vector: situazione particolare in cui una parte di sottorete (un gruppo di nodi) invia i distance vector nello stesso momento congestionando la rete (causato da un sincronizzamento dei tempi di aggiornamento).

Le tabelle sono dinamiche: si creano e si modificano durante il loro funzionamento.

Con un numero di step (invio dei pacchetti) pari al diametro del grafo, ogni nodo ha la conoscenza di tutta la rete.

ROUTING TABLE : A			
	A	0	-
	B	3	1
	E	4	0
DV_B :	C	7	1
	D	5	1
DV_E :	D	7	…
…	…	…	…

La entry della tabella relativa a DV_E (D, 7) viene scartata in quanto la tabella manterrà in memoria solo i cammini minimi e dato che esiste un cammino più veloce (D,5,1) allora scarterà quello più lungo.

E' un algoritmo che permette ad ogni router di trovare il cammino minimo su cui far transitare un pacchetto.

Inizialmente ogni router è configurato con una tabella in cui conosce solo la rete a cui è connesso e i router a lui adiacenti.

Tale tabella contiene inoltre il peso associato ad ogni link (hop, delay, tempo…) e si chiama tabella delle adiacenze.

In un secondo momento, ogni router invia ai suoi adiacenti, la propria tabella delle adiacenze.

In questa fase, ogni router aggiunge alla propria tabella, una entry per ogni nuovo router di cui è venuto a conoscere l'esistenza e aggiunge anche il costo per raggiungerlo; inoltre verifica che il cammino per un altro router che ha nella propria tabella di routing sia inferiore al cammino identificato dai distance vector che gli sono stati propagati dai suoi adiacenti: in questo caso nulla cambia, altrimenti il nuovo valore viene aggiornato con il nuovo router su cui far transitare i pacchetti per raggiungere la destinazione ed il relativo costo.

Ad ogni entry è associato un TTL entro il quale devono essere riverificate le entry: per tale motivo ogni intervallo regolare (30 secondi), ogni router invia il proprio distance vector.

Il protocollo che usa tale algoritmo si chiama RIP (Routing Information Protocol).

Problemi:

1. generazione di loop
2. count-to-infinity (data la non conoscenza dell'intera rete, due router continuano a scambiarsi un pacchetto in seguito ad un link non funzionante)

Una soluzione al problema del count-to-infinity è dato dalla tecnica dello Split Horizon, ossia un router A non condivide le conoscenze su un percorso con un determinato router B se tale conoscenza è stata data proprio da quest'ultimo router B.

Esempio:

il router "A" invia pacchetti a "C" attraverso "B" (la topologia di rete è **A—B—C**). "A" apprende quindi dell'esistenza di "C" proprio da "B", ma quando "A" condivide la sua tabella di routing con "B" ometterà il percorso relativo a "C". È utile notare che "B" non ha bisogno dell'informazione omessa da "A" poiché già la conosce.

In caso di interruzione del collegamento fra "B" e "C", "B" cancella inizialmente il percorso relativo a "C" dalla sua tabella di routing. Senza lo split horizon, "A" comunicherebbe in seguito a "B" che conosce un percorso verso "C". "B" non sa che questo percorso prevede il passaggio attraverso "B" stesso (questa informazione non è condivisa dai protocolli di routing) quindi lo aggiunge alla sua tabella di routing. In questo particolare scenario, se dovesse giungere un pacchetto da inoltrare a "C", i router "A" e "B" se lo inoltrerebbero a vicenda senza riuscire a trovare un percorso verso "C".

Utilizzando la regola dello split horizon questo particolare scenario di anello non viene a crearsi.

Path Vector è una variante di Distance Vector ed è anche questo una soluzione ai loop in quanto PV non si affida solo alla distanza tra i router ma ricorda il percorso che compie un pacchetto e quindi in grado di identificare i loop. Inoltre PV permette di utilizzare metriche diverse tra diversi AS perché proprio in quanto sistemi autonomi differenti possono avere esigenze differenti.

Count-to-infinity e Bouncing effect

A parità di costo sui link, ci possono essere routers che instradano sul link con identificativo più basso oppure altri che fanno load balancing (suddivisione del traffico sulle diverse porte).

Ci sono problemi quando:

[nn capisco]

RIP

Routing Information Protocol usa Distance Vector ed è possibile utilizzare i “triggered update” ossia degli update di DV che avvengono direttamente al rilevamento di un guasto (possibile storm).

A causa del possibile storm, RIP usa un ritardo random quando viene generato il triggered update.

Le reti piccole possono usare RIP e contano come costo il numero di hop (da 0 a 15, 16 equivale ad infinito).

Il DV si propaga ogni 6 volte il tempo del timer (6 x 30 sec.).

Ogni entry ha associato un timer che se scade viene impostato un costo infinito.

Link state e Shortest Path First

Questo algoritmo è sostanzialmente diverso da distance vector e si compone di due fasi.

La prima fase corrisponde alla creazione della topologia della rete da parte dei router.

Inizialmente ogni router ha a disposizione una tabella che contiene l'indicazione della rete interna a cui è connesso e l'elenco dei router adiacenti.

A intervalli regolari, ogni router condivide con i propri vicini tale tabella mediante dei pacchetti chiamati link-state packet in modo da espandere sempre di più la conoscenza: infatti ogni router inserisce nella propria tabella l'indicazione di quale rete interna è connessa ad un determinato router.

Alla fine di questa prima fase, tutti i router conoscono la topologia della rete.

Inizia poi una seconda fase che utilizza l'algoritmo di SPF, in particolare ogni router sfrutta l'algoritmo di Dijkstra per trovare i cammini minimi per arrivare ad ogni router presente nella mappa appena creata.

OSPF e Designated Router

Algortimo Open Shortest Path First che usa LS-SPF. Permette di risolvere il problema del flooding (invio in massa del pacchetto LS) utilizzando un Designated Router.

Il DR esiste con lo scopo di ridurre il traffico di rete fornendo una sorgente per aggiornamenti di routing, il DR memorizza una tabella completa sulla topologia della rete e manda gli aggiornamenti agli altri router attraverso il multicast. In questo modo tutti i router non devono costantemente aggiornarsi l'un l'altro, e possono ricevere tutti gli aggiornamenti da una singola sorgente. L'uso del multicasting riduce ulteriormente il carico di rete.

Il DR inoltre calcola tutti i cammini migliori per inviare i pacchetti sulla rete. Come fanno gli altri router a mandarsi i pacchetti con il designated router?

Costruendo quello che si chiama "Spanning Tree" a partire dalla radice (il DR). Lo spanning tree è un grafo di copertura non orientato che permette di definire la topologia della rete . Questo grafo non è privo di cicli, anzi, i cicli esistono per permetter la ridondanza e la robustezza della rete nel caso dovesse guastarsi qualche link. Tuttavia solo alcuni link (le porte fisiche) sono segnati come "attivi" mentre altri sono dichiarati "bloccati". Tenendo conto solo dei link attivi è possibile invece ottenere una rete senza cicli che permette un invio corretto senza loop all'intero della rete.

Utilizzando questa cosa risolvo il problema del flooding proprio con l'utilizzo dello spanning tree.

In caso di rottura di un link, lo spanning tree va modificato e ritrasmesso.

BGP

BGP è un protocollo che gira su ogni router. Durante la comunicazione con BGP, un router non invia il costo dei link ai router adiacenti ma solo il percorso che questo intraprende per raggiungere tutti gli altri router.

BGP è più un protocollo a livello applicazione più che un protocollo a livello di rete.

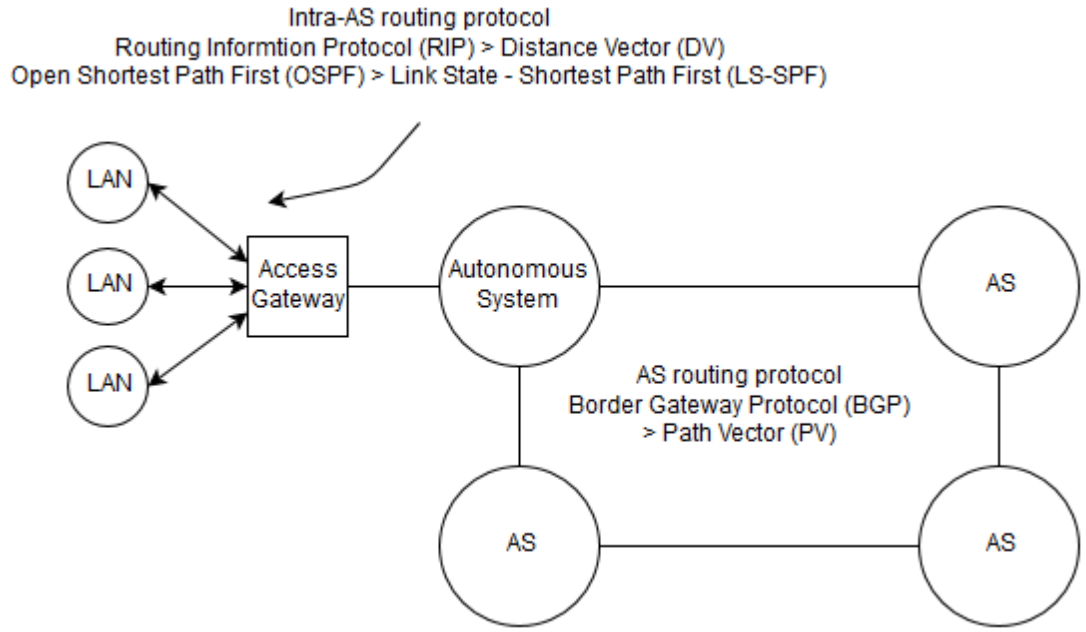
Ogni router BGP poi calcolerà al suo interno il percorso minimo da intraprendere in base a quanto acquisito dalla comunicazione con i suoi vicini.

BGP lavora con il protocollo di routing Path Vector, molto più raramente con Distance Vector.

I pacchetti BGP PV sono più grandi di quelli DV.

Ci sono 4 tipi di messaggi in BGP:

1. Open: usato per creare una associazione con gli altri AS boundary routers con cui il router è connesso
2. Update: usato per trasmettere le informazioni di routing
3. Keepalive: usato come acknowledge del messaggio di open e per confermare periodicamente la relazione con un altro boundary router
4. Notification: usato per informare i ricevitori che un errore è stato trovato.

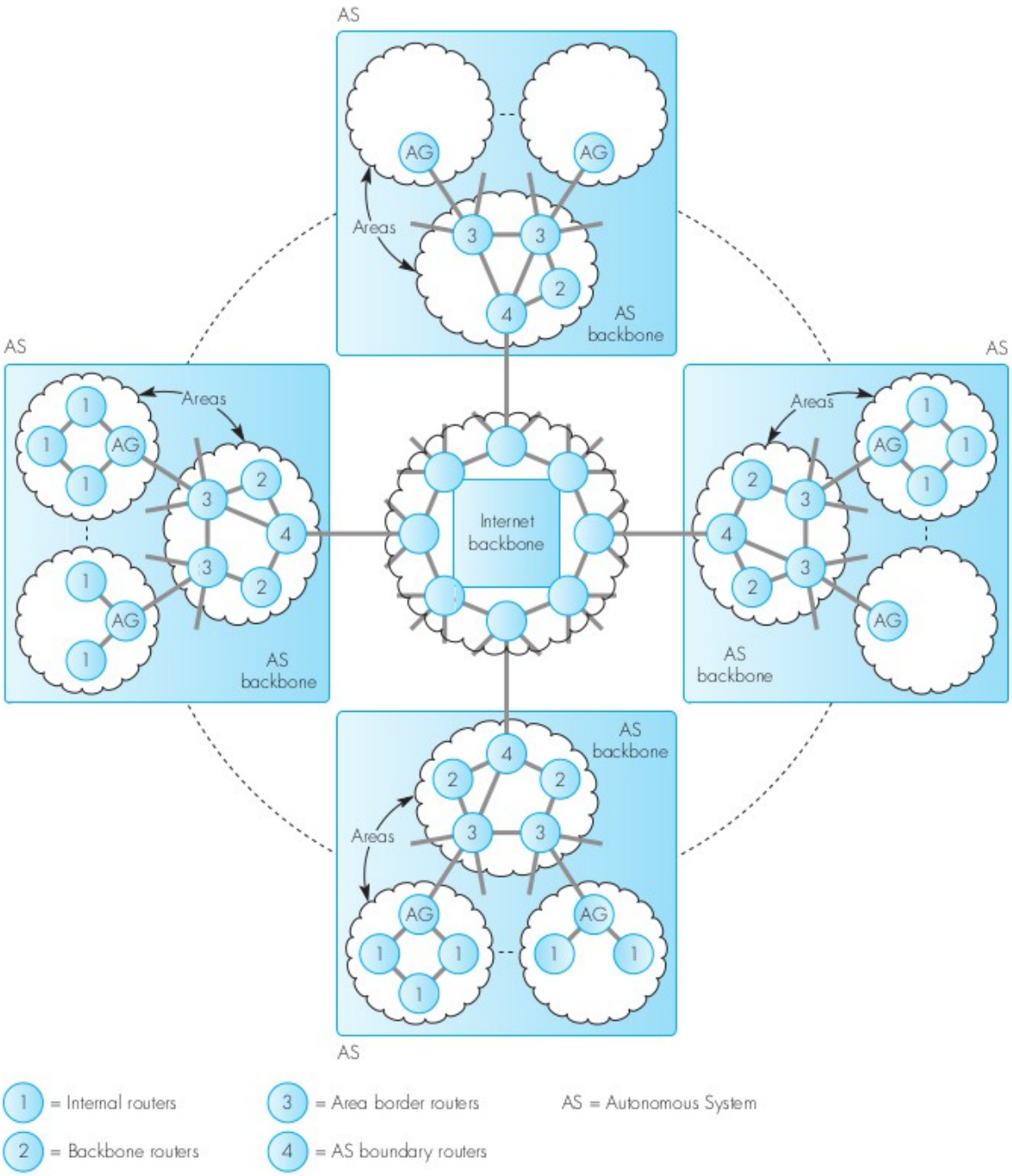


Schema della rete internet

All'interno della Internet backbone viene utilizzato il protocollo BGP.

All'interno degli AS viene usato OSPF.

In ogni singola rete interna ci sono diversi algoritmi utilizzati (anche se di solito sono RIP e OSPF).



Congestion control

Se in una rete viene a mancare un link, la capacità di un nodo potrebbe non essere sufficiente a gestire la congestione magari diventando l'unica porta per accedere ad altri nodi e quindi essere il punto di passaggio di tutto il traffico.

Ogni classe di servizio ha esigenze di traffico diverse, per questo è necessario attuare una gestione delle code.

Esiste un pacchetto particolare che si chiama “Choke Packet” e permette la segnalazione di incipiente congestione.

Stima di utilizzo della rete calcolata con la formula:

$$u = \alpha n + (1- \alpha)\, m$$

con α il peso della misura, n lo storico e m la misura attuale.

Se u >= 0,6 viene inviato un warning che è la soglia sopra la quale viene inviato il choke packet.

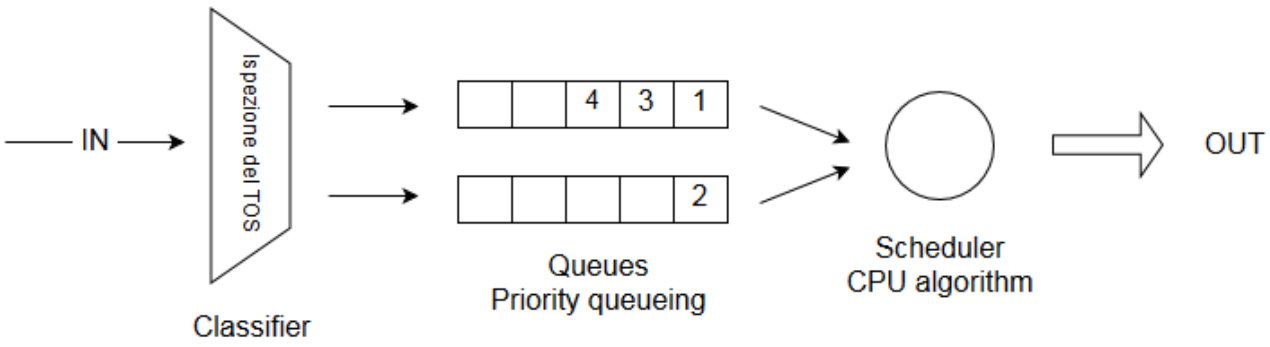
Alla ricezione di tale pacchetto la sorgente abbassa la frequenza di invio dei pacchetti verso il nodo che ha diramato il choke packet.

Le nostre reti infatti sono costruite non per killare processi (connessioni) ma per droppare pacchetti se la rete è troppo congestionata e non si riesce a gestire il flusso.

Alcune configurazioni permettono di inviare il choke packet ai nodi vicini chiedendo a loro di fare coda per un determinato nodo che ha inviato il CP ed eventualmente a loro volta inviare a propri vicini il CP etc...

Nel caso ci fosse del dropping, cosa sarebbe meglio droppare? TCP o UDP? Logicamente ha senso droppare dei pacchetti che non vengono più ritrasmessi in modo da non creare ulteriore traffico, quindi i primi ad essere droppati sono i pacchetti udp.

Packet scheduling



Weighted fair queueing permette uno scheduling pesato : ad ogni coda viene assegnato un peso in base alla quale verrà assegnato un diverso “quanto di tempo” per il servizio sulla coda.

$$\text{VcodaServizioI} = w_i / \sum w_j$$

Il tempo è dato dal peso per una determinata coda, diviso per la somma di tutti i pesi (i, j).

MPLS

Il Multi Protocol Label Switching è un protocollo utilizzato per l'indirizzamento dei pacchetti tra routers.

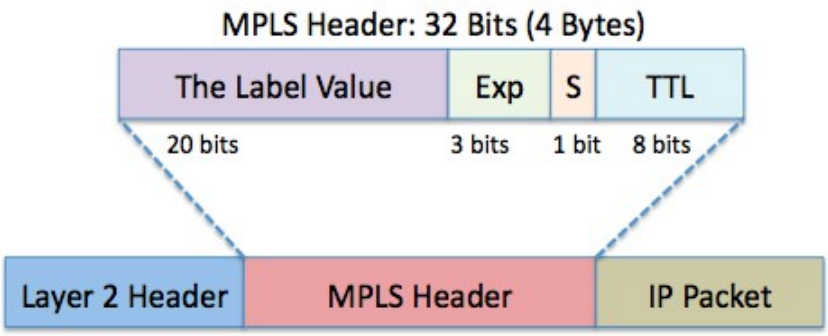
Tale protocollo può essere usato tra diversi AS come estensione di BGP oppure come protocollo interno di un AS a patto che un protocollo di routing come OSPF abbia preventivamente creato la conoscenza della topologia della rete in modo univoco.

MPLS utilizza il concetto di label tra coppie di router. Infatti un router IP/MPLS incapsula il pacchetto IP all'interno di un header di 32 bit (20 di label, 8 di TTL, 3 di priorità e 1 per le successive label) e lo spedisce al router destinatario.

Tale metodo risulta essere più performante del solo instradamento mediante IP e permette perciò un maggior valore di throughput.

Una volta che un pacchetto è stato inviato al router per essere inviato, quest'ultimo incapsula il pacchetto IP come pacchetto MPLS. La struttura del pacchetto rimarrà uguale fino a che l'ultimo router MPLS rimuova tale header per inviarlo poi alla destinazione finale.

Tra ogni coppia di router ci sono label differenti per ognuno dei flussi di traffico.



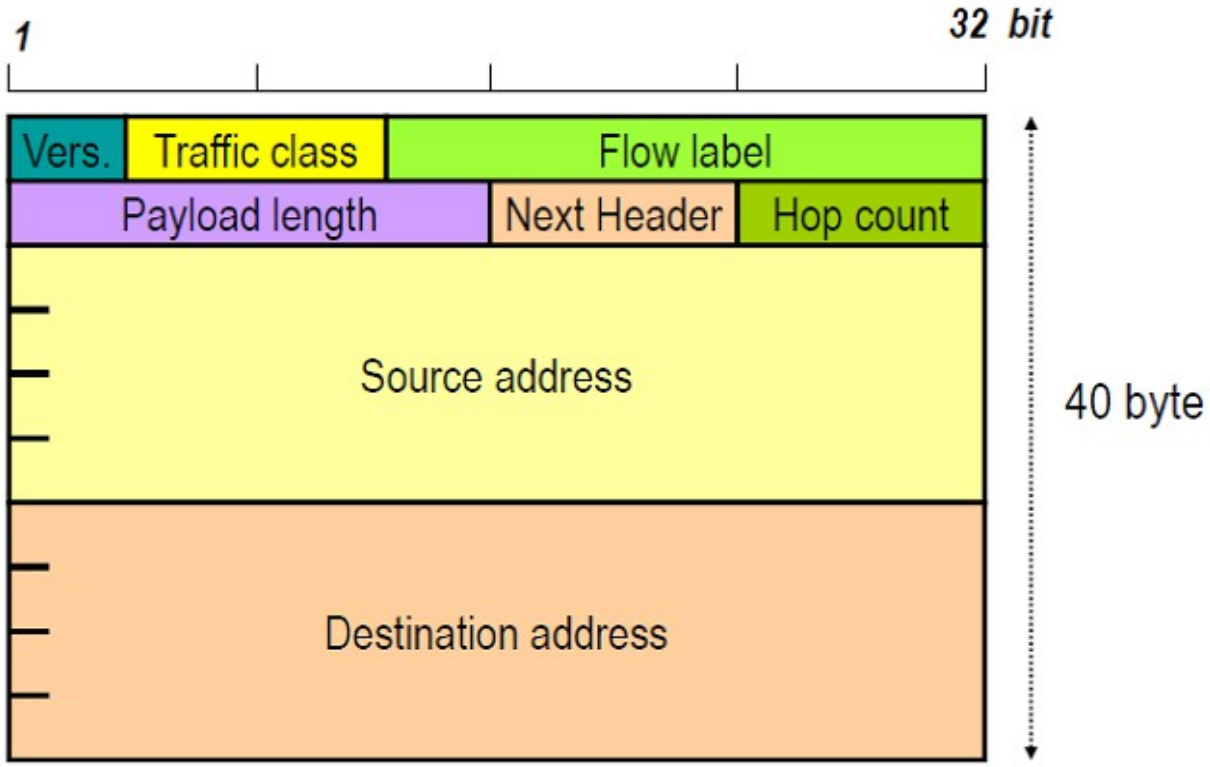
Label: etichetta assegnata al flusso

Exp: rappresenta il Type Of Service

IP v6

Il pacchetto IP versione 6 cambia la lunghezza degli indirizzi da 32 bit a 128bit (16 bytes) e nasce per far fronte alla scarsità di indirizzi attualmente presenti sulla rete.

Payload min : 536 byte / 16 kb



Formato indirizzo

010	REG	TLA	NLA	SLA	INTERFACE ID
3	5	8	32	16	64

- REG : Registry (ICANN , IANA ...)
- TLA : Top Level Aggregator (Internazionale)
- INTERFACE ID : host ID
- NLA : National Level Aggregator (next level)
- SLA : Site Level Aggregator

Struttura del pacchetto del livello 3

HEADER LEVEL 3	PAYLOAD LEVEL 3		
HEADER IPV6	HEADER LEVEL 4 TCP	HEADER LEVEL 7	DATA

HEADER	PAYLOAD				
HEADER IPV6	NEXT HEADER	HEADER LEVEL 4 TCP	NEW HEADER K	HEADER LEVEL 7	DATA

Next Header

Può esserci la necessità di aumentare le dimensioni del pacchetto o aggiungere dei servizi, quindi utilizzare altri pacchetti con altri formati. Questa possibilità in Ipv6 esiste ed è possibile grazie al campo Next Header.

Ogni pacchetto speciale ha un header speciale definito da degli standard ed un numero identificativo K.

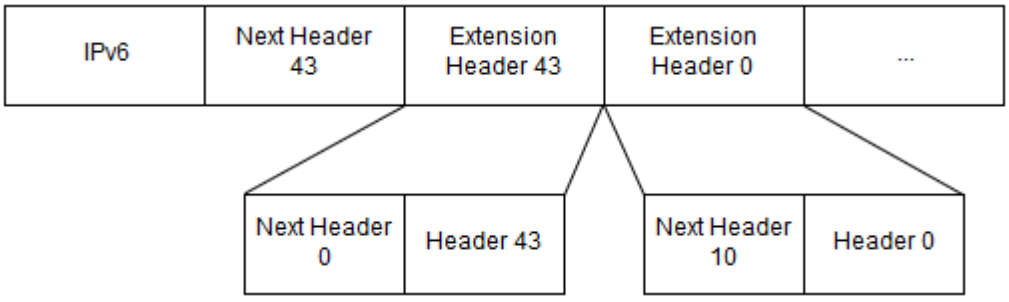
Per esempio quando dei dati devono essere spediti su cavi molto lunghi è possibile utilizzare dei pacchetti chiamati Jumbo Packets che fanno clumping di diversi pacchetti anche di utenti diversi (K = 0).

Esiste la possibilità di creare dei pacchetti con source routing ossia specificando quali sono i router da cui deve transitare il pacchetto (K = 43) che sono al max 23. Inoltre all'interno dell'header una bitmap di 23 bit (24 per comodità di cui uno non usato) che identificano se un router è raggiungibile in modalità strict (1: il pacchetto deve passare solo da quel determinato router come cammino obbligato del prossimo hop) o loose (0: il pacchetto deve passare da quel router ma può anche passare in un hop che non è il successivo).

Opzioni pacchetti:

- Hop by Hop
- Routing Extension Header
- Fragment
- Auth
- Security

Se il next header ha valore 10 significa che non ci sono altri extension header.



Comunicazione IPv4 ↔ Ipv6

La comunicazione v6 → v4 è possibile grazie al tunnelling. Infatti un indirizzo v4 è composto da 32 bit che stanno all'interno del campo dell'indirizzo v6 di 128.

Un pacchetto v4 è possibile inserirlo all'interno di un v6 ed inviarlo facendo così tunnelling.

I problemi si presentano con la comunicazione v4 → v6. Infatti un indirizzo di 128 bit non ci sta in un campo da 32 bit quindi in fase di comunicazione il v4 non saprebbe che indirizzo utilizzare essendo il destinatario un v6. Tale problematica è risolvibile con l'utilizzo di un NAT al quale è associato un pool di indirizzi v4 con i corrispondenti v6: in questo modo un v4 che deve comunicare con un v6, in realtà utilizza un v4 e il NAT poi provvederà a tradurre tale v4 nel corrispettivo v6. In fase di risposta, il v6 risponderà con un v4 (possibile con il tunnelling).

Anche quando si deve utilizzare una comunicazione a pacchetto ip su una rete che non parla ip è possibile effettuare tunnelling.

TCP

TCP (Transmission Control Protocol) è un protocollo che lavora a livello 4 ed è il primo protocollo end-to-end che si incontra.

Pacchetto → livello 3

Segmento → livello 4

Tcp è un protocollo affidabile al contrario di UDP che è best effort.

Le porte sono interfacce tra i livelli e sono univocamente assegnate all'interno di un sistema.

La porta è un numero che il sistema operativo associa ad ogni sessione di comunicazione con tcp.

Le socket sono le API che permettono di aprire un canale di comunicazione utilizzando le porte (una socket è paragonabile ad una “open file”).

<IP,PORT> identifica un processo univocamente in una rete.

Una socket è composta da: < < PORT X, IPs> Protocol Selector <PORT Y, IPd> >

La sorgente non conosce PORT Y ma conosce l'indirizzo IP (o il nome logico) e la porta è standardizzata in base al servizio offerto (server web = 80, ftp = 20...).

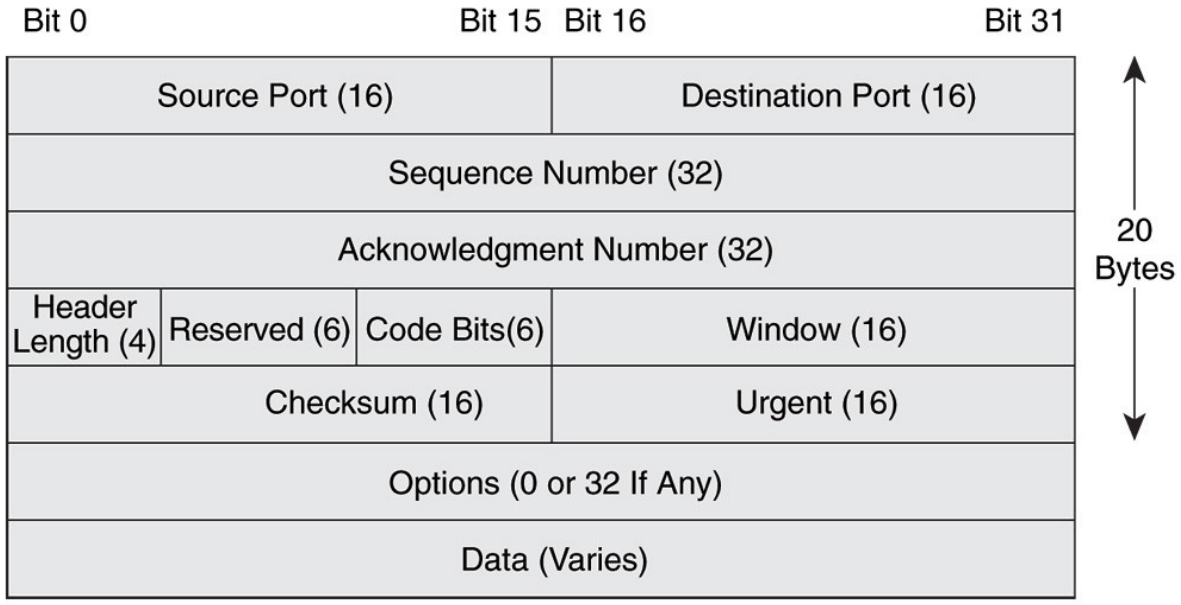
Le porte non standard si trovano sopra la 1.500 e devono essere comunicate preventivamente (fare advertising della porta).

Le funzionalità di connessione peer-to-peer non sono facili perché non si conosce a priori le porte da utilizzare trai due host, al contrario la comunicazione client-server è molto più facile mediante l'utilizzo di porte well-known.

Se due processi accedono alla stessa porta, il server non saprebbe a chi rispondere: il server perciò fa una fork del processo ad ogni richiesta di un client e crea una tabella di memoria tcp con l'elenco della porta k (80) e di tutte quelle che si sono aperte nuove (k+1, k+2)...

Inoltre esiste un numero massimo di client che un server può servire che viene specificato inizialmente: ogni altro server che eccede la capacità massima, non viene servito.

Apertura connessione → data transfer → chiusura connessione



La grandezza massima di un segmento è 64kb (Maximum Segment Size).

Source port [16 bit] - Identifica il *numero di porta sull'host mittente* associato alla connessione TCP.

- Destination port [16 bit] - Identifica il *numero di porta sull'host destinatario* associato alla connessione TCP.
- Sequence number [32 bit] - Numero di sequenza, indica lo scostamento (espresso in byte) dell'inizio del segmento TCP all'interno del flusso completo, a partire dall'Initial Sequence Number (*ISN*), negoziato all'apertura della connessione.
- Acknowledgment number [32 bit] - Numero di riscontro, ha significato solo se il flag ACK è impostato a 1, e conferma la ricezione di una parte del flusso di dati nella direzione opposta, indicando il valore del prossimo *Sequence number* che l'host mittente del segmento TCP si aspetta di ricevere (affidabilità).
- Data offset [4 bit] - Indica la lunghezza (in dword da 32 bit) dell'header del segmento TCP; tale lunghezza può variare da 5 dword (20 byte) a 15 dword (60 byte) a seconda della presenza e della lunghezza del campo facoltativo *Options*.
- Reserved [4 bit] - Bit non utilizzati e predisposti per sviluppi futuri del protocollo; dovrebbero essere impostati a zero.
- Code bits [6 bit] - Bit utilizzati per il controllo del protocollo:
    - URG - se impostato a 1 indica che nel flusso sono presenti *dati urgenti* alla posizione (offset) indicata dal campo *Urgent pointer*. *Urgent Pointer* punta alla fine dei dati urgenti;
    - ACK - se impostato a 1 indica che il campo *Acknowledgment number* è valido;
    - PSH - se impostato a 1 indica che i dati in arrivo non devono essere bufferizzati ma passati subito ai livelli superiori dell'applicazione;
    - RST - se impostato a 1 indica che la connessione non è valida; viene utilizzato in caso di grave errore; a volte utilizzato insieme al flag ACK per la chiusura di una connessione.
    - SYN - se impostato a 1 indica che l'host mittente del segmento vuole *aprire una connessione TCP* con l'host destinatario e specifica nel campo *Sequence number* il valore dell'Initial Sequence Number (*ISN*); ha lo scopo di sincronizzare i numeri di sequenza dei

due host. L'host che ha inviato il SYN deve attendere dall'host remoto un pacchetto SYN/ACK.
    - FIN - se impostato a 1 indica che l'host mittente del segmento vuole *chiudere la connessione TCP* aperta con l'host destinatario. Il mittente attende la conferma dal ricevente (con un FIN-ACK). A questo punto la connessione è ritenuta chiusa per metà: l'host che ha inviato FIN non potrà più inviare dati, mentre l'altro host ha il canale di comunicazione ancora disponibile. Quando anche l'altro host invierà il pacchetto con FIN impostato la connessione, dopo il relativo FIN-ACK, sarà considerata completamente chiusa.
- Window size [16 bit] - Indica la dimensione della *finestra di ricezione* dell'host mittente, cioè il numero di byte che il mittente è in grado di accettare a partire da quello specificato dall'acknowledgment number. (senza mandare in overflow il destinatario)
- Checksum [16 bit] - Campo di controllo utilizzato per la verifica della validità del segmento. È ottenuto facendo il complemento a 1 della somma complemento a uno a 16 bit dell'intero header TCP (con il campo checksum messo a zero),dell'intero payload, con l'aggiunta di uno pseudo header composto da: indirizzo IP sorgente(32bit),indirizzo IP destinazione(32bit), un byte di zeri, un byte che indica il protocollo e due byte che indicano la lunghezza del pacchetto TCP (header + dati).
- Urgent pointer [16 bit] - Puntatore a dato urgente, ha significato solo se il flag URG è impostato a 1 ed indica lo scostamento in byte a partire dal *Sequence number* del byte di dati urgenti all'interno del flusso.
- Options - Opzioni (facoltative) per usi del protocollo avanzati.
- Data - rappresenta il carico utile o *payload* da trasmettere cioè la PDU proveniente dal livello superiore.

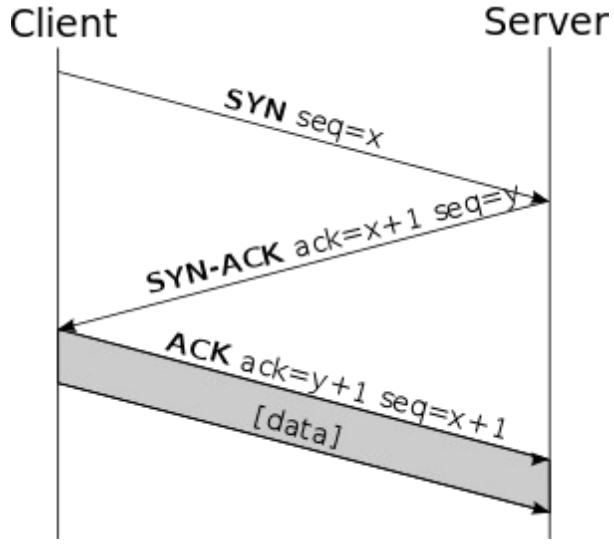
TCP invia sempre un ack cumulativo o un nack.

TCP tiene traccia di quanti byte può sopportare sia il destinatario, sia la rete.

Il protocollo crea un header per ogni segmento ed anche uno pseudo-header che non viene inviato ma utilizzato nel calcolo del codice di controllo e viene usato dal destinatario per verificare che i segmenti in ricezione siano quelli che effettivamente devono essere trasmessi sul determinato canale attivo.

Gli header di tutti i segmenti, pacchetti etc... sono dei campi fissi che vengono riempiti all'atto dell'invio.

3 Way Handshake



In fase di apertura della connessione è possibile specificare il window size, altrimenti di default è 64 kb.

La procedura utilizzata per instaurare in modo affidabile una connessione TCP tra due host è chiamata three-way handshake (stretta di mano in 3 passaggi), indicando la necessità di scambiare 3 messaggi tra host mittente e host ricevente affinché la connessione sia instaurata correttamente. Consideriamo ad esempio che l'host A intenda aprire una connessione TCP con l'host B; i passi da seguire quindi sono:

1. A invia un segmento SYN a B - il flag SYN è impostato a 1 e il campo Sequence number contiene il valore x che specifica l'Initial Sequence Number di A;
2. B invia un segmento SYN/ACK ad A - i flag SYN e ACK sono impostati a 1, il campo Sequence number contiene il valore y che specifica l'Initial Sequence Number di B e il campo Acknowledgment number contiene il valore x+1 confermando la ricezione del ISN di A;
3. A invia un segmento ACK a B - il flag ACK è impostato a 1 e il campo Acknowledgment number contiene il valore y+1 confermando la ricezione del ISN di B.

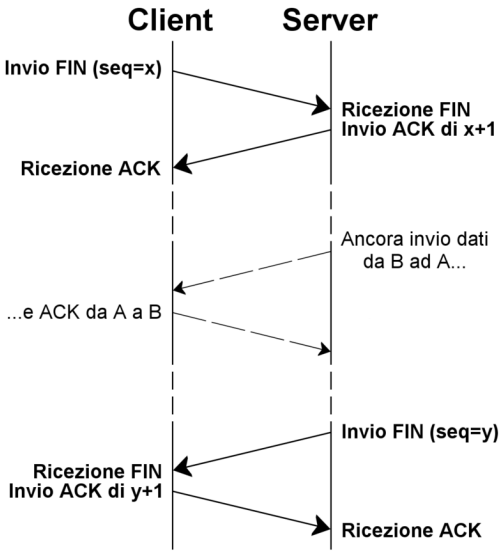
Il terzo segmento non sarebbe, idealmente, necessario per l'apertura della connessione in quanto già dopo la ricezione da parte di A del secondo segmento, entrambi gli host hanno espresso la loro disponibilità all'apertura della connessione. Tuttavia esso risulta necessario al fine di permettere anche all'host B una stima del timeout iniziale, come tempo intercorso tra l'invio di un segmento e la ricezione del corrispondente ACK.

Il flag SYN risulta utile nell'implementazione pratica del protocollo, e nella sua analisi da parte dei firewall: nel traffico TCP i segmenti SYN stabiliscono nuove connessioni, mentre quelli con il flag non attivo appartengono a connessioni già instaurate.

I segmenti utilizzati durante l'handshake sono solitamente 'solo header', ossia hanno il campo Data vuoto essendo questa una fase di sincronizzazione tra i due host e non di scambio di dati.

Il three-way handshake si rende necessario poiché la sequenza numerica (ISN) non è legata ad un clock generale della rete, inoltre ogni pacchetto IP può avere il proprio modo di calcolare l'Initial Sequence Number. Alla ricezione del primo SYN non è possibile sapere se la sequenza ricevuta appartenga ad un ritardo dovuto ad una precedente connessione. Tuttavia, viene memorizzata l'ultima sequenza usata nella connessione, potendo così essere richiesta la verifica all' host mittente del SYN appartenente alla vecchia connessione.

4 Way Handshake



Connessione tra A e B chiusa

Dopo che è stata stabilita, una connessione TCP non è considerata una singola connessione bidirezionale, ma piuttosto come l'interazione di due connessioni monodirezionali. Pertanto, ognuna delle parti deve terminare la *sua* connessione, e possono esistere anche *connessioni aperte a metà*, in cui solo uno dei due terminali ha chiuso la connessione e non può più trasmettere, ma può (e deve) ricevere i dati dall'altro terminale.

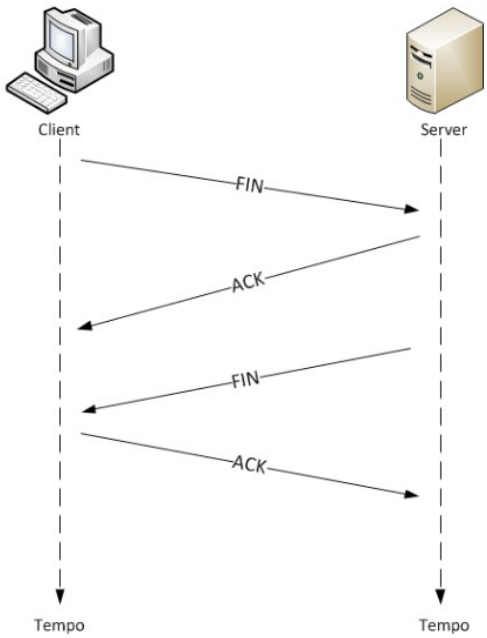
Di conseguenza, la chiusura della connessione si può effettuare in due modi: con un handshake a tre vie, in cui le due parti chiudono contemporaneamente le rispettive connessioni, o con uno a quattro vie, in cui le due connessioni vengono chiuse in tempi diversi.

L'handshake a 3 vie è omologo a quello usato per l'apertura della connessione, con la differenza che il flag utilizzato è il FIN invece del SYN. Un terminale invia un pacchetto con la richiesta FIN, l'altro risponde con un FIN + ACK, ed infine il primo manda l'ultimo ACK, e l'intera connessione viene terminata.

L'handshake a 4 vie invece viene utilizzato quando la disconnessione non è contemporanea tra i due terminali in comunicazione. In questo caso uno dei due terminali invia la richiesta di FIN, e attende l'ACK. L'altro terminale farà poi altrettanto, generando quindi un totale di 4 pacchetti.

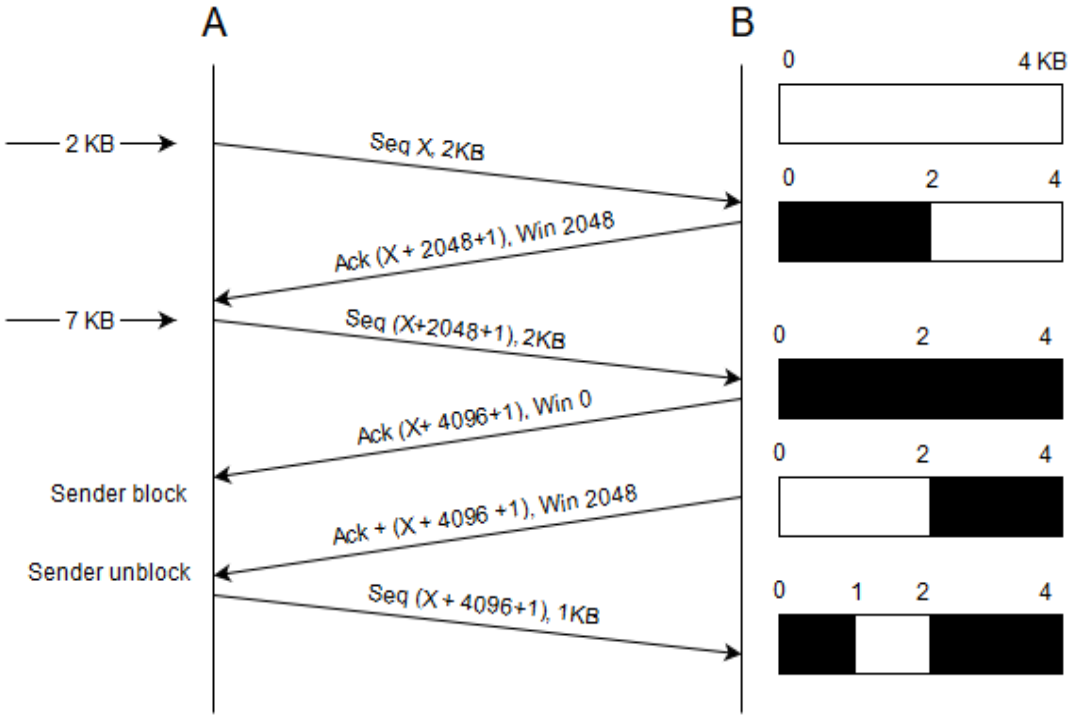
Fasi:

1. Il client invia al server un segmento con bit FIN=1, entra nello stato FIN_WAIT_1
2. Il server invia al client un riscontro, il client entra nello stato FIN_WAIT_2
3. Il server invia al client un segmento con il bit FIN=1
4. Il client riscontra il segmento ed entra nello stato TIME_WAIT. Dopo un tempo che varia dai 30 secondi ai 2 minuti la connessione viene conclusa



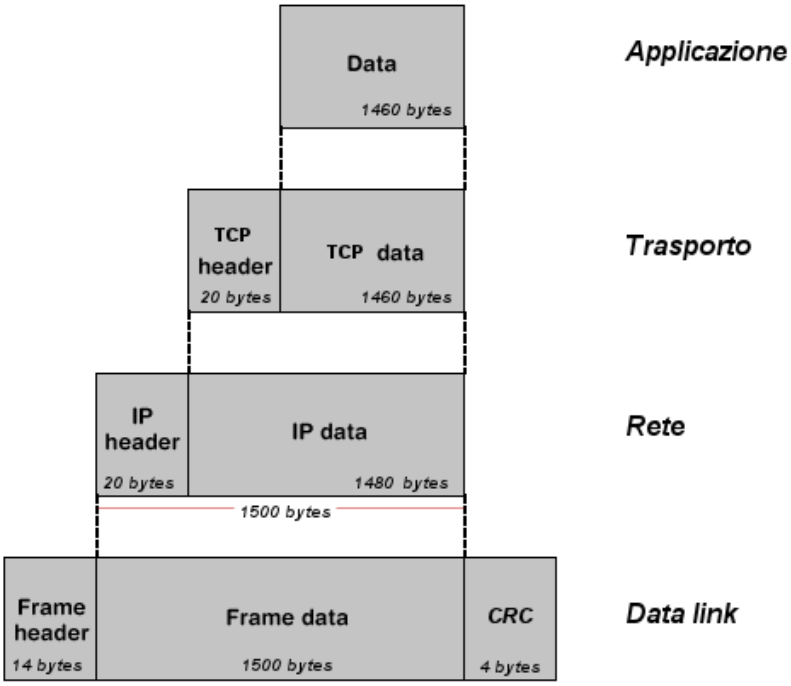
Window Size

Se viene perso un segmento contenente una windows size (vedi 5a freccia) è possibile arrivare in uno stato di deadlock in cui non viene inviato più nulla sul canale.



MTU / MSS / MWS

Il concetto di MTU genera spesso confusione, in particolare nel momento in cui lo si vuole associare ai vari livelli della pila ISO OSI. Per chiarire ogni dubbio, diciamo subito che per MTU si intende la dimensione massima (1500 byte) dei pacchetti a livello 3 (rete).
Considerando i dati al livello 2 della pila ISO OSI, la dimensione minima di una trama (frame) Ethernet è di 64 byte, mentre quella massima è di 1518 byte. Di questi, 1500 byte sono per il campo dati, mentre i restanti 18 sono costituiti da 14 byte di intestazione ethernet + 4 byte di Cyclic redundancy check (CRC). Se si utilizzano le VLAN, il tag adoperato dallo standard 802.1Q fa crescere la trama di ulteriori 4 byte, arrivando a 1522. Le dimensioni sono calcolate senza tenere



conto del preambolo, di lunghezza pari a 8 byte. Ricordiamo che lo scopo del preambolo è quello di sincronizzare il clock dei dati tra la stazione trasmittente e quella ricevente.

Poiché nelle reti Ethernet i pacchetti IP sono incapsulati in una trama Ethernet, e poiché questa può avere un massimo di 1500 byte per il campo dati (carico pagante), possiamo capire come questa dimensione sia quella massima del pacchetto IP stesso.

Entrando nel dettaglio, poiché il pacchetto IP ha un'intestazione (header) di 20 byte, il suo campo dati (carico pagante) avrà una dimensione massima di 1480 byte, corrispondenti al segmento TCP incapsulato in IP.
Anche TCP ha un'intestazione (header) di 20 byte: il suo campo dati (carico pagante) avrà quindi una dimensione massima di 1460 byte.

TCP possiede un meccanismo per calcolare il "Maximum Segment Size" (dimensione massima del segmento o MSS), corrispondente al valore dato dalla formula MSS = MTU - 40. Pertanto è al livello 4 che si decide la dimensione massima del campo dati, non appena ha inizio il processo di suddivisione dei dati stessi in blocchi incapsulati verso i livelli più bassi.

Il Max Window Size (MWS) si riferisce invece alla quantità massima di byte che si possono inviare all'interno di una finestra di TCP che è 64 KB.

Livello ISO/OSI	Livello 2	Livello 3	Livello 4
Protocollo	MAC	IP	TCP
Nome unità	Frame	Pacchetto	Segmento

TCP → Maximum Segment Size

IP → Maximum Transmission Unit

IPv6 packet → 576 (536 payload + 40 header) up to 65.575 (65.535 payload + 40 header)

TCP IPv6 Max payload → MTU IPv6 – 60 (40 ip + 20 tcp)

IPv4 packet → (536 payload + 20 header) up to 65.555 (65.535 payload + 20 header)

TCP IPv4 Max payload → MTU IPv4 – 40 (20 header tcp + 20 header ip)

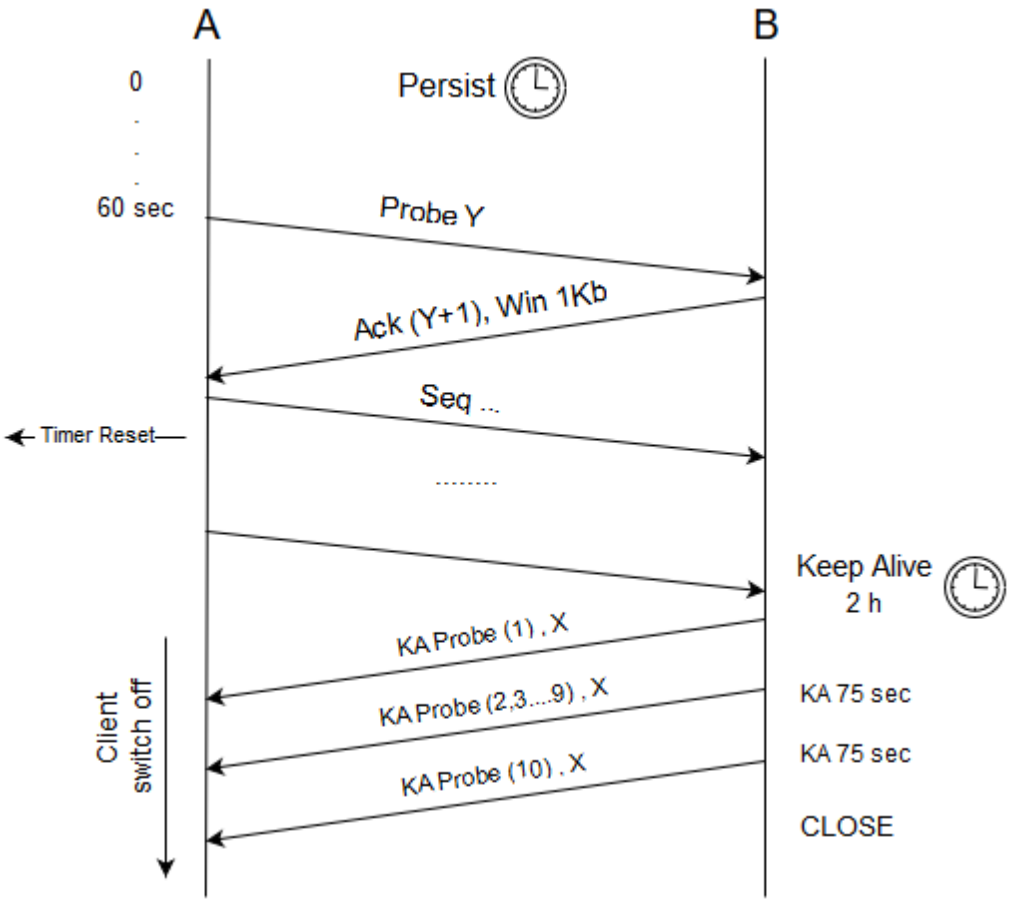
Maximum Window Size (TCP) → 64 kb

TCP fragmented → 1480 (1460 payload + 20 header)

IPv4 fragmented → 1500 (1480 payload + 20 header)

Ethernet → 64 (18 header + 46 padding) up to 1518 (1500 payload + 14 header + 4CRC) or 1522 (802.1Q)

Timing



Algoritmo di Nagle

Funziona lato sorgente.

Le applicazioni che effettuano tante piccole chiamate sincrone via rete (di tipo request/reply) possono generare un traffico composto a sua volta da piccoli pacchetti. Le performance di tali applicazioni dipendono quindi più dalla latenza introdotta nel processare tali pacchetti che dal massimo throughput gestibile dall'architettura fisica utilizzata.

L'utilizzo di piccoli pacchetti è solitamente ritenuto inefficiente dal punto di vista di chi ottimizza una rete. Ciò è particolarmente vero nei punti di raccordo del traffico IP (es. un Internet backbone) e l'algoritmo di Nagle cerca proprio di scoraggiarne l'uso quando il trasporto è TCP.

Gli stack TCP/IP solitamente implementano in modo parametrico la dimensione massima per cui un pacchetto è ritenuto "piccolo" ed è applicato questo algoritmo: ad esempio impostare tale parametro a 1 equivale a disabilitarne l'uso.

L'algoritmo di Nagle indica che lo stack di chi invia non deve trasmettere il prossimo piccolo pacchetto (come definito in precedenza) se è ancora in attesa di ricevere l'ACK di un piccolo pacchetto già inviato.

Di conseguenza chi invia deve accumulare i dati in spedizione fino a che sia soddisfatta una delle due seguenti condizioni:

- la dimensione dei dati pronti per l'invio ha superato l'MTU (Maximum Transmission Unit)
- è stato ricevuto l'ACK per tutti i piccoli pacchetti in sospeso

Un effetto non gradito di questa ottimizzazione è che le applicazioni che inviano continuamente piccoli pacchetti possono osservare una latenza maggiore di quelle che utilizzano pacchetti con dimensione dell'ordine dell'MTU. In tali casi è possibile testare le performance disabilitando l'uso dell'algoritmo (nelle modalità offerta dalle singole piattaforme) e confrontare i risultati ottenuti con la configurazione precedente per verificare se si ricade in una delle casistiche in cui l'algoritmo peggiori le performance complessive.

Tale soluzione è utile nelle applicazioni testuali (Goolge docs …) ma è problematico per i giocatori online che per ridurre il così detto LAG (LAtency Gap) disattivano l'algoritmo di Nagle.

Clark

Funziona lato destinatario.

Durante una comunicazione il trasmettitore potrebbe avere un buffer di ricezione più lento rispetto alla velocità di invio. Se ogni volta che il buffer si libera di una piccola porzione, il ricevente trasmette una win update nuova, si potrebbe generare un grande traffico dovuto proprio alla lettura lenta del buffer (Silly Window Syndrome).

Clark risolve il problema con una formula che quantifica il numero di byte da leggere prima di inviare un nuovo win update:

$$wait = MIN (½ \text{ buffer di ricezione OR } 1 MSS)$$

MSS = Maximum Segment Size

Questo algoritmo funziona sia in SELECTIVE_REPEAT lato ricezione che GO_BACKN lato invio fino all'arrivo dell'ack cumulativo.

Il recovery di un segmento avviene dopo la ricezione di 3 segmenti fuori sequenza: infatti la ricezione dei segmenti può non essere ordinata. Questa tecnica si chiama Fast Retransmit o Fast Recovery.

Calcolo RTO / RTT

A livello 4 non è possibile calcolare direttamente il Round Trip Time e il Retransmission TimeOut come nel livello 2. Si usano quindi delle formule per calcolarlo indirettamente.

Inizialmente RTO viene impostato a 3 secondi ma nei calcoli successivi verrà subito approssimato correttamente mentre l'RTT iniziale non esiste.

Dalla formula del primo RTO è possibile trovare il primo valore di D.

$$RTO = SRTT + 4D$$

quindi l'RTT si calcola come:

$$SRTT = \alpha * StimaStoricoRTT + (1 - \alpha) * ultimoRTT$$

SRTT = Smoothed Round Trip Time (stima dell'RTT ammorbidita)

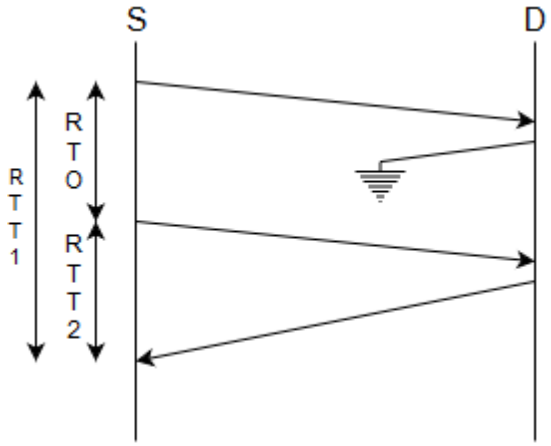
Dove α è un fattore variabile di smoothing di solito impostato a 7/8 ossia 0,9 (si da 90% di importanza alla stima dello storico dei valori RTT e il 10% all'ultima misura rilevata).

Poi si calcola la deviazione standard dell'ultima misura dell'RTT dalla media SRTT.

$$D = \alpha * oldD + (1 - \alpha) * | SRTT - ultimoRTT|$$

Karn

Cosa succederebbe nel caso in cui durante la stima del RTO (Retransmission TimeOut) si perdesse un segmento?



Se si perde l'ack come si fa a valutare il RTO?

Si prende in considerazione RTT1 o RTT2?

Se dovessimo tenere conto di RTT1 conteremmo anche gli eventuali errori che si verificano nel mentre.

Potrei scegliere RTT2. Ma se l'ack rimane congestionato e arriva dopo l'invio del segmento di RTO?

Si creerebbe una situazione (figura 2) di ack quasi istantaneo ovviamente errata.

Come si fa a dimensionare quindi l'RTT?

Karn dice che se ci sono errori, butto via le misurazioni precedenti e ricalcolo il nuovo RTT come segue fino a quando non ho più errori:

$$NEW\ RTT = OLD\ RTT * 2$$

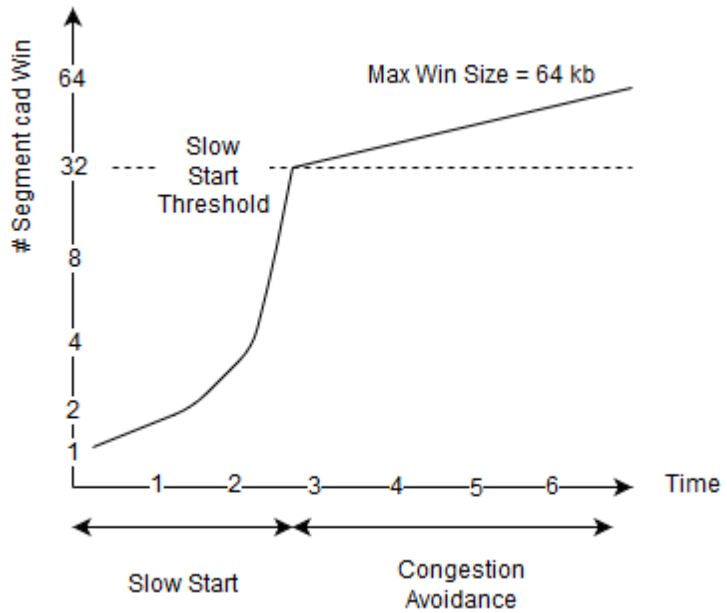
Congestione e portata

Durante la comunicazione la destinazione ha a disposizione un buffer in cui salva tutti i dati in arrivo dalla rete e periodicamente invia il win update alla sorgente. Il problema che si presenta però a livello 4 è l'incapacità da parte degli hosts di conoscere la congestione della rete: è possibile che la destinazione abbia un buffer di dimensione minore rispetto alla capacità della rete e quindi faccia del dropping.

Come arrivare quindi ad avere la conoscenza del minor numero di bytes (stima) che una rete riesce a gestire senza errori? La maggior parte degli errori deriva da congestione e non da errori fisici sui cavi.

Se TCP vede un errore sul canale, la prima cosa che identifica come errore è la congestione, abbassando il flusso di invio dei dati.

$$W_{tcp} = MIN(C_{rete}, B_d)$$



C_{rete} = Capacità rete

B_d = Buffer destinatario

Il minimo flusso di dati di tcp è il minimo tra la congestione della rete ed il buffer di destinazione.

Se il lavandino è otturato cosa posso capirlo? Aprendo piano l'acqua e mano a mano aumento: la stessa cosa fa TCP su una rete di cui non conosce il livello di congestione.

Congestion window (W_c) : finestra (Flight Size) di invio dei dati in base alla congestione della rete.

In caso di Retransmission TimeOut :

$$SST = Flight\ Size / 2$$

$$W_c = 1\ Maximum\ Segment\ Size$$

Quindi la finestra ritorna ad essere grande come il MSS e riparte con slow start.

In caso di Fast Recovery :

$$SST = Flight\ Size / 2$$

$$W_c = SST$$

Si continua con Congestion Avoidance anche se sono sotto i 32 kb.

Il wireless soffre di problemi di trasmissione al contrario della trasmissione su cavo che presenta maggiori problemi di congestione.

Opzioni

Esistono diverse opzioni in TCP tra cui:

- timestamp → invio di pochi byte senza il calcolo dei valori relativi a tcp

Chiusura

Una connessione tcp si chiude con il FIN BIT (chiusura asimmetrica).

Se il client chiude la connessione, il server continuerà a trasmettere ciò che rimane nello stream.

Chi chiude attivamente deve attendere la chiusura dell'host passivo.

Chiusura:

Fin 1 , Seq X →

RTO Fin wait

← Ack 1 , Ack X+1

← Fin 1 , Seq Y

Timed wait

RTO last ack

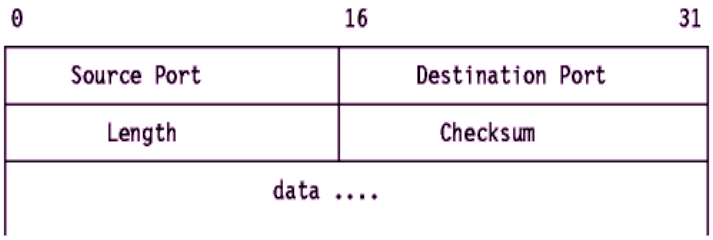
Ack 1 , Ack Y+1 →

Timed wait è un timer specifico. Un protocollo ha il compito di mantenere sincronizzati gli hosts.

Un protocollo che non una timer potrebbe utilizzare la ridondanza anche se il rischio è quello di generare traffico inutile.

UDP

User Datagram Protocol è un protocollo best effort.

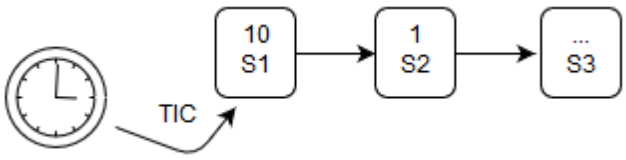


La CRC serve per la correttezza del segmento e se è sbagliato, il segmento viene buttato via, infatti anche se non c'è un recovery del segmento, comunque quelli sbagliati non vengono consegnati.

UDP viene usato per traffico realtime, stream video etc...

Timer

La gestione dei timer è particolare, infatti viene controllato un solo timer che grazie a delle strutture a lista permette la gestione di tutti gli altri.



Ad ogni tic vado a cambiare solo il primo nella lista, i successivi vengono salvati per differenza temporale dal precedente.

Esercizi

- t_x = frame / banda
- t_p = Lunghezza / C
- $U = t_x / (t_x + 2t_p)$

→ A

f = 1000 bit

b = 10kb

L = 100 mt

$t_x = 10^3 / 10^4 = 10^{-1} = 0,1$ sec

$t_p = 10^2 / 2*10^8 = 5*10^{-7}$

→ B

f = 1000 bit

b = 1 mb

L = 300 km

$t_x = 10^3 / 10^6 = 10^{-3} = 1$ ms

$t_p = 3*10^5 / 3*10^8 = 5*10^{-3} = 1$ ms di propagazione

U = 1/3 k = 3

→ C

t_p = 40 ms

b = 8 kb

f = ?

U = 0,5

$0,5 = t_x / (t_x + 80 \text{ ms}) = 0,5\ t_x + 0,04 = t_x \rightarrow 0,04 = 0,5 = 0,08$

$0,08 = f / b \rightarrow 0,08 * 8 * 10^3 = 640$ bit di frame

$t_x = 10^3 / 10^6 = 10^{-3} = 1$ ms

→ D

f = 2000

b = 1 mb / sec

t_p = 20 ms

numero di sequenza composto da 3 bits

U = ? selective repeat

finestra al max di 4 (in go_backn finestra grande al max 7)

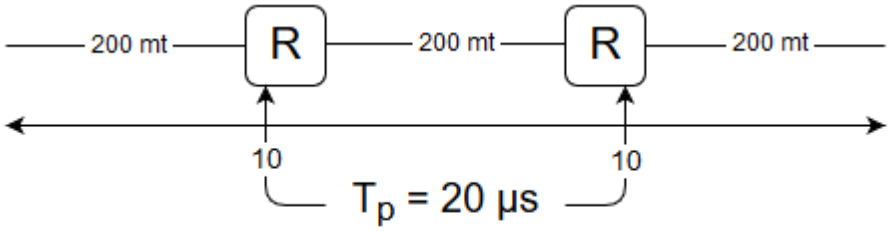
$t_x = 2*10^3 / 10^6 = 2 * 10^{-3} = 2$ ms

U = 2ms / (2ms + 40ms) = 4,7% poco uso del canale

k = 4 4 * 4,7 < 20% poco utilizzo

Potrei aumentare l'utilizzo usando 4 bit di k.

→ E



f = ?

b = 8 mb / sec

$t_p = 6*10^2 / 2*10^8 = 3 * 10^{-6} = 3$ µs + 20 µs = 23 µs

t_x = 46 µs = $2t_p$ = f/b = $46 * 10^{-6}$

$f = 46 * 10^{-6} * 8 * 10^6 = 368$

Edoardo Vignati